A Survey of Partially Connected Neural Networks

D. Elizondo[1,2] E. Fiesler[1] J. Korczak[2]

[1] IDIAP
Case Postale 592
CH-1920 Martigny
Switzerland
Electronic mail: elizondo@maya.idiap.ch

[2] Université Louis Pasteur, Département d'Informatique
7, rue René Descartes, 67084 Strasbourg Cedex
France
Telephone (33) 88-41-64-37
Electronic mail: elizondo@ia5.u-strasbg.fr

Abstract

Almost all artificial neural networks are by default fully connected, which often implies a high redundancy and complexity. Little research has been devoted to the study of partially connected neural networks, with its potential advantages like reduced training and recall time, improved generalization capabilities, reduced hardware requirements, as well as being a step closer to biological reality. This publication presents an extensive survey of the various kinds of partially connected neural networks, clustered into a clear framework, followed by a detailed comparative discussion.

Keywords: neurocomputing, connectionism, connection, network topology, partial connectivity, ontogenic network, non-ontogenic network, sparse neural network, partially connected neural network, diluted neural network, randomly connected neural network.

1 Introduction

Fully connected neural networks (FCNNs) are the most commonly used neural networks. One of the principal reasons for using FCNNs is to simplify the neural network design. However, it usually allso implies a large amount of redundancy.

The term *fully connected* can have different meanings depending on the topology of the neural network model under study. It is often used loosely by different authors, and is therefore a cause of confusion. Among the different FCNNs found in the literature, there are:

1. *Plenary neural networks*, where all possible interlayer, intralayer, supralayer, and self connections are present [24],
2. *Plenary neural networks* without self-connections like associative memories,
3. *Fully interlayer connected neural networks* (FICNN), where all possible interlayer connections are present and the network contains no intralayer (including self-connections), or supralayer connections. The term FCNN is most often (loosely) used for neural networks with this kind of topology.

These three interpretations will be used throughout this paper specifying the different kinds of "fully connected" neural networks.

A partially connected neural network (PCNN) can therefore be defined as a network which contains only a sub-set of the entire set of possible connections for a particular neural network model.

The idea behind PCNNs is to have a reduced neural network topology equivalent or better in performance than the fully connected model, but with a smaller number of connections. A reduction in the number of connections in the network can help improve generalization and reduce the network complexity, hardware, and storage requirements, and training and recall time.

A variety of methods for dealing with PCNNs have been proposed, and there seems to be an increase in interest in this area. This survey presents a compilation and an overview of these methods.

In this paper, three classes of methods for adding and deleting connections are discussed, the ontogenic, the non-ontogenic, and the hybrid methods.

Ontogenic methods specify how the topology of the neural network (NN) is modified during the learning phase. Although a section on ontogenic methods is included for completeness, the emphasis of this survey is on non-ontogenic methods. An in-depth survey on ontogenic methods can be found in [22].

Non-ontogenic or static methods for connectivity reduction are those for which the topology of the network is defined prior to the learning phase. Once in the learning phase, the topology remains unchanged throughout the whole learning process.

Hybrid methods are combinations of neural networks with other artificial intelligence (AI) techniques. These AI techniques include: symbolic knowledge and genetic programming. The aim of these hybrid methods is to define both the network topology as well as the initial weight values; the former by using of a knowledge base of domain specific inference rules (a domain theory), and the latter by evolving the topology and the weights of the network.

The three classes of PCNN methods described above can be further sub-divided in the following way:

1. Non-Ontogenic Methods
   - methods based on theoretical and experimental studies,
   - methods derived from biological neural networks,
   - methods that are application dependent,
   - methods based on modularity,
   - methods developed for hardware implementation.
2. Ontogenic Methods
   - ontogenic methods based on the gradient descend algorithm,
   - ontogenic methods based on other neural network learning rules.
3. Hybrid Methods
   - methods that combine neural networks and inductive knowledge,
   - methods that combine neural networks and genetic programming.

This classification reflects the structure of this paper.

2 Non-Ontogenic Methods

2.1 Theoretical and Experimental Studies of PCNNs

This section presents the work done on PCNNs from a theoretical point of view. In general, the work is kept at a theoretical level, with either none or very little practical applications for testing the given

theory. The goal of these theoretical models is often to demonstrate the consequences of sparse connectivity, usually in terms of training dynamics, rather than to apply them to actual problems.
This section is divided into theoretical studies based on gradient descend, associative memory, and stochastic models.

2.1.1 Gradient descend based studies

Hansen *et al* present an approach for reducing the number of weights in extremely ill-posed problems [36]. These problems are characterized by having a large number of correlated inputs, denoted by N_1 and a small number of sample patterns from which the system can learn, denote by P. Their linear algebra based approach consists of transposing the problem from a high dimensional space to a low dimensional one. This is done by means of expanding the weight vectors into a P dimensional signal space vector, where P represents the number of training patterns. This P dimensional signal space corresponds to the linear subspace of the input space extended by the actual inputs in the training set. The expansion of the weight vectors reduces the dimensionality of the problem from $N_1 * N_2$ to $P * N_2$, where N_2 represents the number of hidden units, and N_1 the dimension of the input space. The approach, based on Le Cun's *weight sharing* strategy [52], is presented for both supervised, (feed-forward) with one hidden layer, and unsupervised (Sanger's [70]) networks.

An early study on determining the optimal number of connections in a feed-forward network is presented by Kung *et. al.* [46]. The authors assume that synaptic weights are the strongest among neighboring neurons, and that the strenght among connections is reduced as a function of the distance between the neurons. Thus, local connections are selected by means of the range within which connection strengths are not neglegible (what the author call *effective correlation bandwith*). The authors use locally connected networks with a few global connections, which improve the convergence rate.

A performance study on locally versus globally interlayer connected multilayer networks is presented in [65]. In a *globally connected* model, according to the terminology of the authors, each neuron is connected to every neuron in the layer immediately next to it. In a *locally connected* model, each neuron is connected to some of the neurons in the layer immediately next to it. The study includes a series of simulations, performed on encoding and pattern association problems, for comparing the two types of connection strategies. The comparison criteria include the number of training iterations, hidden layers, and hidden units needed by each strategy. Locally connected models contain multiple hidden layers. Each neuron is randomly connected to at most three nearest neighbor neurons in the next higher layer. The locally connected method compensates for the reduced connectivity by augmenting the number of hidden units in the network. This augmentation is necessary because, by limiting the connectivity level in the hidden units, one also alters the geometry of the decision regions found in most classification problems. This reduces the capacity of separability of the input space by hyperplanes, which is performed by the hidden units. The authors conclude that, despite the fact that a larger number of iterations is needed for the locally connected models to converge, a performance level comparable to that of globally connected model, can be achieved with local connectivity. The advantages offered by local over gloabal connectivity include a reduction in the interconnection complexity and in the input/output requirements for each neuron unit, and an increase in recall speed.

Redding, Kowalcyk, and Downs [66] present a polynomial time method for determining the minimal fan-in of hidden units (a single hidden layer is assumed) in the context of classification problems. The method consists in determining the degree of a polynomial that will separate the classes into linearly separable clusters. The complexity of this method for a problem with p patterns is calculated to be equal to $O(p^9)$ with some good early estimates obtainable in $O(p^3)$.

The use of parabolic neurons for reducing the number of connections in high order neural networks is proposed by Yang and Guest [87]. Parabolic neurons are second order neurons constrainted to point towards the origin of the coordinate system. Each parabolic neuron requiers $N + 2$ connections, as opposed to $N(N + 1)$ requierd by a regular second order neuron, where N is equal to the number inputs. It is assumed that in a second order network some of the weights are more important than others in

determining the general shape of a boundary solution surface of a classification problem. The simplest of all shapes on a second order network is the parabola which is close to a hyperplane. Keeping only these weights, the number of connections needed by the network reduces to $2N + 1$. By requiering the orientation of the parabola to pass through the origin of the coordinate system $N - 1$ weights can be removed. The total number of connections needed by a parabolic neuron becomes therefore $N - 2$. Based on two classification examples, the authors conclude that parabolic neurons can improve the performance of the NN with resepect to linear neurons. This type of network requiers less neurons or layers than regular back propagation to solve the same problem, and thus decreases the number of connections needed to solve a particular problem.

A straight forward method for finding minimal neural network topologies for small Boolean functions is presented by Fiesler [21]. The method consists in the generation of all possible topologies that might yield a solution to the problem. Since this is an exhaustive search method, it is limited to very small problems such as the XOR. After the generation of all possible topologies in increasing order of complexity, the function performed by the network for each input pattern is written as an equation which is subsequently subjected to a non-linear activation function. The non-linear activation function is replaced by a Heaviside function which allows the substitution of the equations containing the non-linear function by a set of inequalities which is solved by mutual substitution. The first topology that has a solution is the minimal topology for the given problem. In general, the smallest topologies can be obtained by using high order neural networks. A constructive algorithm for building partially connected high order neural networks is also presented in order to ilustrate the potencial power of these models. This algorithm allows the generation of two layer sparsely connected high order network for arbitrary Boolean functions.

2.1.2 Associative Memory

Kurten presents a study of three neural network topologies with sparse connectivity [47]. The neural network models consist of N sparsely connected logical units whose output can be either $+1$ or -1. Each of these units has exactly d_{in} arbitrarily chosen input links coming from any of the neighboring, not necessarily the nearest, units in the network to which it may connect. Each of the network models is given a set of $P = C * d_{in}$ random unbiased patterns to memorize, within at most twenty five epochs, where d_{in} is equal to the fan-in of each neuron and C represents the upper limit of the storage capacity of the network with a critical value C_g, equal to 0.42.
The first model is called the *next-neighbor square lattice model*. This model has cells residing in a two-dimensional square lattice restricted to eight next-nearest neighbor connections. Based on the poor level of performance obtained with this approach, the authors conclude that models with exclusive nearest-neighbor connections are not well suited for implementing associative memories.
The second model is called the *random neighbor long-ranged model*. In this model, the units choose their arbitrary input neighbors at random. This model's performance is better than that of the first model for certain values of C, but it performs rather poorly at levels of C superior to the critical value of C_g.
The third model is called *quasi-optimal long-ranged neighbor model*. This is an ontogenic variation of the second method, in which new random neighbors are repeatedly chosen until a near optimal connectivity pattern is found. Computer simulations suggest that for randomly chosen information, optimized sparsely connected networks outperform both their equivalent short-ranged lattice and fully connected models. The authors deduce that the degree of connectivity and thresholds of the network have to be adapted to the specific information the network has to capture, and can be modified while in an optimization process.

Minai and Levy [61] present a study on the dynamic properties of PCNNs. This study is based on the dynamics of random networks with sparse asymmetric connectivity. These networks consist of N binary neurons with a threshold θ for determining whether the neuron is active or not. Each neuron connects excitatorily to other neurons, and to itself, with a constant probability p, generating a fixed set of connections. All excitatory connections have the same fixed weight value w, where $0 < w \leq 1$. There is a single interneuron which gets firing information from all primary neurons, and provides the same level of inhibition to all the excitatory neurons at the next time step. An *interneuron* corresponds to a neuron structure along the axon of a primary neuron that distributes the primary neurons's activity to other

neurons [10]. The authors are mainly interested in the activity dynamics of the neural network model. In particular, their goal is to estimate the activity level of the model at time t given its activity at time $t-1$. This is done by modulating the degree of inhibition of the network defined by an inhibition regulator parameter called α. This parameter corresponds to the relative strength of inhibitory and excitatory weights and is equivalent to $\frac{\theta}{1-\theta}\frac{K}{w}$, where θ corresponds to the firing threshold, K is a fixed inhibitory weight, and w is the weight value for all the excitatory connections. Five activity behaviors, characterized by α, were found in these networks, ranging from cycles of low period, to those of extremely high aperiodicity.

A statistical method (based on statistical neurodynamics proposed by Amari) for studying the retrieval dynamics of an *associative neural network model* with arbitrary sparse connectivity is presented in [86]. The neural network is composed of N neurons which can have either a -1 or +1 activation value (resting or firing state respectively) with respect to the weighted sum of the outputs from the other neurons. The synaptic connections are constructed from P given memory patterns $s^1, s^2 ... s^P$, where $s_i^p = \pm 1 (i = 1, ..., N; p = 1, ..., P)$. Each of the P memory patterns is composed of N bits of information. The article presents the effect that random removal of connections between neurons (symmetric or asymmetric) can have on the performance of the model with some comments on systematic removal (defined by a Hebb based rule). This performance corresponds to the pattern recall capacity of the model without loss of generality. The authors conclude that the storage capacity per connection in the associative model is larger for small connectivity levels with noise free thresholds (these thresholds are part of the dynamics of the network model). They also conclude that the basin of attraction is larger for smaller connectivity when the loading ratio R_L is fixed. The loading ratio corresponds to the amount of information loaded per one connection and it is defined as: $R_L = NP/(cN^2) = P/(cN)$, where c corresponds to the degree of connectivity of the network, and is defined by: $c = \frac{1}{N(N-1)} \sum_i \sum_{j \neq i} c_{ij}$ (c_{ij} corresponds to the connection from the ith to the jth neuron). They show graphically that when there is static noise in the threshold, there exists an optimal value of connectivity for the performance of the network model. This value, varies according to the noise level.

Stochastic Models

Barna and Kaski present a study of the effect on sparse connectivity in a Boltzmann machine [6]. The Boltzmann machine is a probabilistic network with binary units and symmetrical bidirectional connections. Sparse connectivity is obtained by deleting both directions of some of the bidirectional connections, thereby maintaining the symmetry of the network. The connectivity density is given by the formula $W/W_{\max}$, where W represents the actual number of connections used, and $W_{\max}$ the set of all possible interlayer connections. The connectivity density of the network was varied from almost none to full interlayer connectivity. Simulation experiments suggest that, for the two problems studied, the best performance results in terms of the percentage of correctly classified inputs were obtained with low connectivity densities. Based on these experiments, the authors concluded that the performance of the network can, in general, increase when using partial connectivity with a lower connectivity density than that of an equivalent fully interlayer connected network. They also concluded that the optimal network topology, in terms of the minimum connectivity level on the hidden layer which will produce the best performance, is strongly dependent on the problem to be solved.

2.1.3 Randomly Connected PCNNs

The principal goal of random connectivity is to randomly select the connections to and from a given neuron within a probability distribution. This principle of random connectivity can be observed in biological neural networks where after birth, it appears as if the neurons are connected to each other in a random way [4]. Studies on random connectivity include Barkai Kanter and Sompolinsky [5], Venkatesh [81], Banzhaf *et al* [4], and Doyon *et al* [18].

2.1.4 Diluted PCNNs

Contrary to randomly connected PCNNs, diluted PCNNs are created by randomly cutting neuron connections within a certain probability. The principal goal of these methods is to study the storage and retrieval capacity of these network models. A study on diluted networks storage capacity is, for example, presented by Gardner [28] and one in retrieval capacity by Englisch, Xiao, and Yao [19].

2.2 PCNN Derived From Biological Neural Networks

This section presents PCNN methods build on connectivity strategies observed in, and based on, biological systems.

The role of sparse connectivity in the minimization of the statistical dependence between the inputs of a binary neural network is presented in [1] and [2]. Since the authors are interested in the theoretical model of the DG/CA3 region of the hippocampal [53], they limit their study to excitatory neurons which interact in a feedforward fashion. They investigate the interactions between the variable firing threshold, the statistical dependence of the input environments, and the amount of synaptic connectivity. Based on several experiments, the authors concluded that the quality of feedforward network transformations which reduce input dependency while minimizing the loss of information depends on:

- Output layer firing threshold (which is highly dependent on the synaptic connectivity of the neural network).
- The network connectivity.
- The level of statistical dependency in the input environment.

The best results in their experiments were obtained with a connectivity probability between 0.1 and 0.5.

A biologically inspired PCNN called *synaptic growth network* is presented in [7]. The network has two input layers, and an output layer. The first input layer corresponds to the *conditioned stimulus*, and the second one to the‘ *unconditioned stimulus*. Conditioning was first described by Pavlov in 1927 [57]. After a few trials in which a stimulus that causes a response is presented to the network, the system will automatically produce the conditioned response upon the presence of the given stimulus. This corresponds to the *conditioned stimulus*. If there is already a strong connection in the neural network between the stimulus and the response, this stimulus becomes an *unconditioned stimulus*. The selection of the neuron connectivity is based on statistical measures, and non-Hebbian processes which model biological synaptic growth. The number of binary associations that an *synaptic growth network* is capable of learning depends on the number and specificity of the *conditioned stimulus*-output connections. A set of theorems is given for selecting the number of *conditioned stimulus*-output per output neurons needed to achieve a given storage capacity for noiseless as well as for noisy inputs.

Peretto and Niez [64] present a study on the memory storage capacity of PCNNs based on observations found in biological networks. Partial connectivity is obtained using random connections. The authors conclude that the maximum number of stored bits in a network is equivalent to the number of connections regardless of the topology of the network. This conclusion is valid only for large values of γ, a parameter for measuring the magnitude ratios between connections of different orders (binary, ternary, etc.), which must exceed the value of $1/\sqrt{N}$ for fully connected networks and of $1/\sqrt{\omega N}$ for PCNNs, were N corresponds to the number of neurons, and ω to the probability for a connection to exist. They also found that the supression of a percentage of connections reduces the storage capacity of the network by the same amount.
The topic of network modularity is also addressed by the authors. They suggest that the optimal size of the modules is independent of the overall size of the network, and that therefore, it must be the same for all species depending only on the scaling behavior of the memory storage capacities.

A study on the dynamics of a randomly connected PCNN is presented by Senn *et al* [74]. The study is based on experiments performed in a slice culture of the spinal cord of an embryonic rat. The network

consists on linear threshold elements that are updated in discrete time steps. The two principal characteristics of this network are excitatory connections with equal distribution and synaptic depression after repetitive firing. The number of connetions for each neuron is assumed to follow a Poisson distribution with a mean μ. The probability of a neuron having a fan-out or a fan-in value of m is $p_m = \frac{\mu^m}{m!}e^{-\mu}$. The authors conclude that the dynamics of these networks depends on low connectivity, high synaptic depression time, and large excitatory post-synaptic potentials. They found, with a set of biologically inspired parameters, that oscillations in an embryonic nework occur.

Shiono *et al.* [75] present a study on the use of information theory to analyze neural connection structure and functioning. The neural network connectivity is deduced from simulated neural spike activity trains. Information theory states that the n point mutual information (MI) expresses the amount of information shared among n processes [60]. The authors found that a two point MI between two neurons can be used for estimating the neurons synaptic strength, and a three point MI among three neurons could be used for finding their connectivity structure. The following four topologies can be found between three neurons A, B, and C:

1 No interconnection between neurons A and B. This implies that A and B are statistically independent and that the three point MI peak is negative.

2 No interconnection between neurons A and C. This means that A and C are statistically independent and that the peak is positive.

3 No interconnection between neurons B and C. This is similar to case 2.

4 Three interconnections. In this case, the above three cases are combined.

A negative peak corresponds to either case 1 or 4. If A and B are not interconnected then the negative peak corresponds to case 1 otherwise it corresponds to case 4. A positive peak corresponds to case 2, 3 or 4. If A and C and B and C are interconnected, then the peak corresponds to case 4. If A and C are not interconnected and B and C are, then the peak corresponds to case 2. If A and C are interconnected and B and C are not, then the peak corresponds to case 3. Otherwise, if neither A and C nor B and C are interconnected, then the interconnection structure cannot be deduced except for the A-B interconnection.

2.3 Application Dependent PCNNs

Application dependent PCNNs are designed using methods which are tailored to specific problems or application areas. The aim is to minimize the number of connections in a network that is suited for a particular problem.

2.3.1 Speech Recognition and Linguistics

Among the speech recognition methods, a time-delay PCNN architecture for isolated word recognition is presented by Lang and Waibel [49]. The original neural network consists of 3 layers containing a 12 (time slice in ms) by 16 (frequency bands in kHz) input array, 8 hidden units, and 4 output units one for each of the words: B (bee), D (dee), E (ee), and V (vee) respectively. The number of inputs is reduced to 16 by connecting every one of the 16 frequency input units to each hidden unit by 3 different connections with time delays of 0, 1, and 2 ms. Each hidden unit is connected to each output unit by 5 different connections having time delays of 0, 1, 2, 3, and 4 ms. This architecture is based on replicated units trained under constraints that assure that the copies of a given unit apply the same weight pattern to successive portions of the input thus producing a share of weights. A performance comparison based on the mean squared error shows a slight improvement of the PCNN over the FICNN.

A study of PCNNs in the domain of linguistics is presented by Regier [67]. The author presents a backpropagation variation capable of learning spatial concepts from several natural languages. The goal of this study is to be able to determine, for any natural language, whether a scene description in that

language is true for a particular scene. This is implemented in a network whose inputs correspond to a scene produced by the superposition of a *trajector* and a *landmark* boundary map, and whose output indicates how appropriate a term is when describing a relation shown in an input scene. In a given scene, a landmark corresponds to the reference object, and a trajector to the object located relative to the reference object.
The network consists of three modules. The first two modules, modules one and two, are partially connected and are designed based on a spatial learning task. These two modules are fed into a third unstructured fully interlayer connected module (FICNN), which produces an output value indicating how appropriate a term would be when describing a relation shown in the input scene. Module one handles directional features, such as the location of an object relative to another, that can be extracted from the scene. Non-directional features, such as presence or absence of contact, are handled by module two. In this module, there is an LM interior map which receives inputs from the *trajector* and *landmark* boundary maps.
There are two feature maps receiving inputs from the *landmark* interior map. They both receive some function of the *landmark* interior map which corresponds to either the maximum (Feature map 1) or the average (feature map 2) response over the nodes in the map below. Each node in these two modules is connected to the node directly below in the *landmark* interior map, and the four nearest neighbors of the *landmark* node. Corresponding links, at different positions, share the same weight value. All links in the neighborhood are constrained to have the same weight value. Thus, there are only two weights to be adjusted for each feature map, one for the center, and one for all the surrounding ones. In addition, each node in the feature map is connected to the node in the same position at the *trajector* boundary map, so that if this *trajector* node is not activated, the feature map node will be inhibited.
This system, designed to learn spatial concepts from different natural languages, works well for several spatial concepts from languages that have distinctive spatial systems.

Ye *et al* present a PCNN in the domain of speech processing for isolated word recognition [89]. The application consists in the development of a neural network model for recognizing spoken numbers in chinese. The proposed PCNN uses both temporal and spectral information. The input layer is therefore represented as a two dimensional array in which the rows correspond to the time (temporal part), and the columns correspond to the frequency (spectral part).
An assumption is made about the existence of at least one hidden layer. Each hidden unit has connections from only a consecutive subset of neurons in the input layer. More than one hidden unit can have the same subset of consecutive input units to increase the discrimination capacity of the model. The union of the subset of cells should include all of the input units.
The authors do not have a general method for selecting the subset of consecutive input units for each hidden unit, and imply that this depends on the particular problem under study. They make, however, some general suggestions for selecting these subsets of cells in their isolated word recognition problem. Some of their guidelines include:

- For the temporal part:
  - A hidden unit can be connected to input neurons from several consecutive temporal input columns (ex. $t-1$, t and $t+1$).
  - More than one neuron can be connected to the same input zone (same input rows and columns) to increase discrimination capacity.
- For the spectral part:
  - The use of fewer connections in the high frequency part of the network, since its resolution is lower than that of the low frequency one.
  - A partially connected neuron can examine the variation of a particular spectral band.
  - Sparser connectivity levels can be used at the end of a word (which may contain fill constants).
  - Neurons can become specialized feature detectors. A feature might, for example, consist in comparing low and high frequencies in the spectrum.

Once the topology has been defined, backpropagation (see section 3.1 for an explanation of the backpropagation algorithm) is used to train the network. A comparison between the learning performance of a PCNN and a FICNN, for the pronunciation of chinese numbers, is presented. The PCNN shows a 30% improvement over the FICNN when testing both models on the recognition of a set of previously unseen patterns.

2.3.2 Image Processing

A PCNN for perceptual learning is discussed in [40]. The network inputs represent a retina of 32×32 pixels. The network is constructed in a pyramid like structure in which each node at layer l receives input from $2 - l$ tuples of nodes drawn from 4 node clusters in layer $l-1$, and layer l has 1/4 the number of node-clusters found in the adjacent layer $l-1$. The authors discuss three variants of the model above:

- Model 1: Use of Local receptive fields for preserving topographic mapping between layers. Each node in layer l is linked to nodes in the 4 node clusters spatially located directly below it in layer $l-1$; layer 1 contains 8 pre-wired edge detectors.
- Model 2: Same as model 1 without edge detectors.
- Model 3: Use of random receptive fields. Each node in layer l is linked to nodes in 4 randomly chosen node clusters in layer $l-1$.

In the three models eight edge detectors (either pre-wired or learnable) were provided at layer 1. All weights, with exception of those of the edge detectors, were randomly initialized. Several test runs, with a varying percentage of connectivity, were performed in order to compare the three methods together with other ontogenic variations of the same model. The highest level of performance within the three non-ontogenic methods was obtained with model number 1. Model 3 had the lowest level of performance. The three non-ontogenic methods were out-performed by their ontogenic variations. The authors conclude that random connectivity is unlikely to work for most practical problems.

A PCNN for image classification was developed by Korczak and Hammadi-Mesmoudi [45]. The problem at hand consists in land usage classification from SPOT remote sensing images (with images of 512×512 bytes). The SPOT (**S**atellite **P**our l'**O**bservation **T**errestre) is a french satellite system for spatial observation of the earth. An image is divided into smaller sub-images of 3×3 pixels each. There is one 3×3 sub-image for each of the three spectral bands. This makes a total of twenty seven pixels, each corresponding to an input unit in the neural network. The pixel to be classified always corresponds to the middle pixel in the 3×3 sub-image. Border pixels which do not have immediate neighbors are not analyzed. The network has five outputs corresponding to one of the following categories: water, forest, meadow, husbandry, or urban zone. In their model, the authors use pixel geometrical relationships as the bases for constructing the topology of the network. These relationships include vertical and horizontal alignment of pixels (three possible locations on which each alignment can occur: upper, middle, and lower part of the sub-image), diagonal axes (six possible locations), and corners (eight possible locations). In the first hidden layer there is one unit for each of the locations of all the relationships for a total of 20 hidden units. Each of these hidden units has a fan-in equal to three and a fan-out of one. A second level of hidden units is introduced which includes four units, one per relationship. The first and second node, corresponding to the horizontal and vertical relationships respectively, have a fan-in of 3, the third node, representing the diagonal axes relationship has a fan-in of 6, and the last unit used to represent the corners, has a fan-in of 8. All of the neurons in this layer have a fan-out of 5. A classification accuracy of 98% is obtained by using this method. No performance comparison is presented with a corresponding fully interlayer connected network (FICNN).

Perantonis and Lisboa [63] introduce a method for reducing the number of weights of a third order network used on invariant pattern recognition. In high order networks, the number of connections increases combinatorialy with the order of the network. In a similar way as in Spirkovska, input triplets contained inside an object are used to form triangles. A reduction in the number of weights is achieved

by assigning approximately similar triangles to the same class. Approximately similar triangles are those which have the same value for their two smallest angles within a certain degree of tolerance.

A study on the configuration and initialization of neural networks for performing morphological measurements on binary images is presented in [73]. The morphological measurements explored by the authors include: the surface area, the perimeter, and the Euler Poincare number which corresponds to the number of particles minus the number of holes in a binary image on a hexagonal lattice.
This study is based on the backpropagation learning rule with local connections and shared weights in a three layer structure with one hidden layer. The hidden layer consists of an arbitrary number of images, each of which has the same number of elements as the input image. The authors present a way to find the optimal neighborhood size needed in order to perform the morphological measurement. The neighborhood corresponds to the neurons in the input image to which each pixel of a hidden image is connected to. There is a single connection between the pixels of each hidden image and the output unit. The study also includes a way to test if a given learning set is complete, in other words, if it is representative of the space of all possible images.

Several partial connectivity strategies for high order neural networks (HONN), applied to the domain of pattern recognition, are presented by Spirkovska and Reid [76]. In order to reduce the number of connections in the network, pattern invariances over translation, scale, and rotation are hard wired into a third order network by using information about relationships expected between the input pixels. Triplets of inputs, which are located within a given object, are used to form triangles with included angles (α,β,γ). Since these three angles remain unchanged with pattern rotation, translation, and scaling, invariance is normally accomplished by using the same weight for all similar triangles found in the input set.
Even though this approach reduces the number of connections required by the network, their number remains very high. To further diminish the connections required by such a network, the authors looked at several methods for connecting only a subset of the input triplets to the output neuron. The approaches studied include:

- *Local connectivity.* In this approach, only triplets of pixels that are within a given number X of pixels away from each other are connected to the output. This number X is specified by a probability measure based on the distance between pixels composing a triangle. The triangles formed by using distant pixels are ignored.

- *Sampled connectivity.* Here, a fraction of all possible input triplets are connected, independent of the distance between them. Only a predetermined limited number of interconnections are formed.

- *Probabilistic connectivity.* The probability of a connection varies with the distance and is based on the distance between the pixels composing the triangle.

- *Regional connectivity.* This is the method which produces the best results. Triplets of pixels are connected to the output only if the distance between all of the pixels falls within a set of preselected regions. The regions are selected in such a way that they provide a good discrimination of the input space. This selection includes triplets formed by local, distant, and in between pixels. Some of the advantages presented by this method include:

  - The consideration of features of various scales within the image; thus obtaining a better sample of what the image looks like.
  - It is invariant to image translation, scaling, and rotation.
  - No extra storage is needed in order to record which connections were formed.
  - A 100% accuracy level is obtained in an image of 128×128 pixels with a connectivity reduction that drops from 10^{12} to 4.1×10^6 interconnections.

A single layer neural network PCNN for thinning binary images is proposed by Wu and Tsai [85]. The process of thinning an image consists on eliminating redundant pixels from an object shape in an image to obtain a reduced contour of the object with one pixel width. This process makes eassier the task of feature extraction in image analysis and pattern recognition. The network proposed by the authors

removes the edge points of an object shape by template matching. If the neighbors of a given point match any of a set of templates, it is removed. Some templates are of size 3×3 and others of size 3×4 or 4×3. A neighborhood of size 8 and 12 is needed for a 3×3 and a 3×4 or 4×3 template respectively. Since two of the pixels in a 3×4 or 4×3 template are irrelevant, a 10 pixel size neighborhood is sufficient for each tested pixel. Each neuron corresponds to a pixel in the input image. Thus, each neuron receives connections from its 10 nearest neighbors and itself. The results obtained with the network are the same as those obtained with the OPPTA method developed earlier by the authors.

2.3.3 Weight Sharing

Weight sharing methods have been proposed by several authors including LeCun *et al* [51] and Nowlan and Hinton [62]. The idea behind weight sharing is to share a single weight between two or more connections in the network thus reducing the number of weights that need to be adjusted. This involves a good understanding of the problem at hand so that similar weights can be combined into a single weight before training takes place.

2.4 Methods Based on Modularity

The idea behind modular neural networks is to divide the problem at hand into smaller sub-problems, each of which is to be solved by a sub-network. These sub-networks are assembled together into a global network which solves the problem. Even though individual modules are often internally fully connected, they are by definition not fully connected to all other modules. Modular neural networks are usually designed for a specific application.

A study on modular neural networks is presented by de Francesco [16]. This study presents a functional frame work which facilitates the development of modular neural systems. The system uses an extension of λ-calculus called ν-calculus to compose neural modules and was developed in CLOS, an object oriented programming language that runs under Common LISP.
Functional neural networks are considered as multidimensional mappings with states. Two types of functional networks are considered: *basic* (indivisible) and a *composition* of several other functional networks. *Basic* networks can be feed-forward, competitive, associative, or mapping. *Composed* networks can be either sequential or parallel. In a sequential composition of two modules, the outputs of the first module are directly connected to the inputs of the second module. In a parallel composition of two modules, the modules are assembled together so that the inputs and outputs of the final module correspond to the union of those of the two original modules. In both sequencial and parallel composition, each module keeps its original topology.

Fogelman *et al.* [25] present a general method for designing and training modular architectures for feature extraction. Each module may have a different neural network model like backpropagation and learning vector quantization. The architecture consists of two concatenated modules. The first module has two hidden layers, one with shared weights and the other with local connections. The second module takes its input from the last hidden layer of the first module and classifies it. The number of connections used with this architecture for an object code recognition problem is equal to 1798, as opposed to 96512 for the equivalent fully interlayer connected network.

Jacobs and Jordan [42] present a modular neural network for control tasks such as the controling of the movement of a simulated robot arm. The architecture of the system consists of two types of networks: *expert* networks and a *gating* network. The expert networks compete to learn the training patterns, and the gating one mediates this competition. The two types of networks can be either feedforward or recurrent neural networks with arbitrary connectivity. The gating network has one output unit for each expert network and the activation of these output units must be non-negative and sum to one. The outputs from the expert and gating networks are all connected to a single output vector. This output vector, denoted y, is represented by $y = \sum_i^n g_i y_i$, where g_i denotes the output vector from the gating network and y_i the

output vector of the ith expert network. Each expert network and the gating network are trainned using the back propagation learning algorithm.

A modular VLSI PCNN for implementing classification problems is presented in [59]. The goal of this study is to generate modular neural networks that can be mapped into hierarchical hardware. The design of the topology of the model is done by means of an input/output space analysis. Classification problems can be divided into two categories, the linearly separable and the non-linearly separable. The perceptron is used to define the linear separability between classes. A set of subnetworks, based on the linear or non-linear separability of the classes at hand, is then generated. These sub-networks are in turn unified by means of logical operators, such as AND and OR, to provide the final neural network topology for the given problem.

2.5 PCNNs for Hardware Implementation

Analog and digital electronic implementations of neural networks can speed up their training and use. However, as the size of the networks scales up, it becomes very hard to maintain full connectivity due to the limits imposed by physical restrictions. It is therefore necessary to develop partial connectivity strategies for electronic hardware implementations of neural networks.

A radix-r PCNN for hardware implementations of neural networks is given by Antola *et al* [3]. This approach, based on radix-r networks, assures that each output of the network is dependent on all the inputs. This method reduces the computational complexity of the feed-forward network learning step $O = W * X$, where W corresponds to an N_1 by N_1 matrix of weights (after learning), N_1 represents the number of inputs, and X is the input vector of the model. The processing needed to produce each single output value from the matrix per vector product O is decomposed in a number of iterations, each one being performed by a stage in the network. Each stage of the network is made of N nodes that calculate the weighted summation of the r inputs coming from the r nodes of the preceding stage. Therefore, the partial outputs obtained at a given stage are dependent on a finite number (r) of outputs produced at the previous stage. The topology of the model ensures that each output unit depends on all the input units **if** the number of stages is $log_r N_1$, where r is the radix chosen for the network and N_1 is the number of inputs. Each neuron has a fan-in and fan-out equal to r, and the activation function is applied **only** to the last layer. This final layer is equivalent to the hidden layer in a FFNN. Learning in the radix-r networks is much more sensitive to the initial random value of weights than in the FICNN. Hence, several runs must be done with different initial random weights in order to achieve a low error. As a rule of thumb for the weight initialization, the authors suggest the use of random weights uniformly distributed from -1 to +1. With this radix-r method, the authors obtained a complexity reduction in connectivity from N_1^2 (N connections per neuron) to $r * N_1 * log_r N_1$ (r connections per neuron) where N_1 corresponds to the number of inputs, and r to the radix value.

A study on fan-in reduction for VLSI implementation of large neural networks is presented by Beiu *et al* [8]. The authors present several transformation algorithms for reducing the fan-in of Boolean neural networks by studying one single neuron or *threshold gate* at a time. Three algorithms for fan-in reduction are presented:

- Classical Boolean decomposition. Here, a function F with 2^k inputs is rewritten in terms of two sub-functions F_1 and F_2 which have the same weights as the original function F. Thus, there is no need to calculate new weight values for F_1 and F_2. The number of layers grows linearly, and the number of gates exponentially, with respect to the number of input neurons.
- Majority gates. This approach involves the use of three input majority gates (2 out of 3) for creating *threshold gates*. Decomposition is obtained by rewriting the function F. As a result, a binary tree structure is obtained, in which one variable is eliminated at each tree level. As in the *classical boolean decomposition* method, the number of gates grows exponentially.

- Threshold gate decomposition based on the *divide and conquer* principle. This algorithm can be directly applied to symmetric and Boolean functions. According to the authors, any symmetric function can be represented as a function which depends exclusively in the number of input variables whose value is equal to one. They also suggest that any Boolean function can be implemented with only two layers of symmetric functions. The first step towards the symmetric representation, is to divide the set of input variables into two groups of equal size in the first layer of *threshold gates*, and to use *threshold gates* to compute all the possible *sums* of *one* values within the two groups. The second step is to generate all possible combinations of *threshold gates* outputs from the two groups of the first layer, in the second layer. This step is accomplished by joining the intermediate results obtained after the division, into two groups, of the input variables in the first step.

Each of the three algorithms presented in this paper, produces a tree structured network topology which differs from the other two in the complexity. The complexity of a VLSI can be measured in terms of AT and AT^2, where A corresponds to the area of the circuit, and T to the delay of the circuit. The values of AT and AT^2 can be estimated by $NG \times NL$ and $NG \times NL^2$ respectively, where NG corresponds to the number of gates and NL to the number of layers. The complexity measure for the first two algorithms is exponential with $AT = N \times 2^N$ and $AT^2 = N^2 \times 2^N$. The complexity of the third algorithm is superpolynomial with $AT = N$ and $AT^2 = N^{\frac{\log N+5}{2}}$. Thus, the best balance between number of gates and layers is obtained with the *divide and conquer* approach.

A VLSI PCNN implementation of the backpropagation algorithm is presented by Faure and Mazare [20]. The authors show a VLSI cellular array for processing neural networks on which each cell performs both the recall and the learning algorithms of a backpropagation network.
The computations are restricted to integers, and there is a limit on the number of connections that a cell can have. Each cell is physically connected to its four immediate neighbors and can be logically connected to any other cell and to the outside world by a parallel message transmission mechanism distributed among the cells.
The limitation in connectivity is overcome by the introduction of partially connected sub-layers in the initial topology. This is done by assigning a complete sub-neuron, with its own non linear activation function and connections, to each cell in the input and output sub-trees that are needed to implement the original neuron in the initial network. A program based on the simulated annealing algorithm is used to map the initial topology into its associated augmented network map for a given cell connectivity limit. This approach reduces the degree of connectivity per neuron by increasing the number of layers and neurons in the network. Hence, it does not decrease the complexity of the networks, instead, it increases the degrees of freedom in the neural network weight space.

A digital VLSI network architecture with hierarchical connectivity is presented in [58]. The main goal of this study is to show a prototype of a chip that presents hierarchical connectivity similar to that observed in biological neural networks. Fully interlayer connected neurons are represented as a hierarchy of subnetworks with low sparse connectivity between them and a high degree of internal connectivity. The high internal connectivity is based on the nearest neighbor approach provided by a mesh architecture. Due to the limit on connections, connectivity to non-neighbor neurons is based on competition among subnetworks and depends on the availability of feedthrough paths.

Cellular Neural Networks

Introduced by L. O. Chua and L. Yang in 1988 [13] [12], cellular neural networks (CNNs) are VLSI circuits that can perform signal processing in real time. They consist of cells which contain linear and nonlinear circuit elements. Every cell is locally connected only to its neighbor cells within a given, often small, distance. In contrast with other neural networks in which most of the VLSI circuit area is used for connections, the connection area needed by a CNN is very small. There is a single inner cell and the rest of the cells are boundary cells. Inner cells have a total of $(2r+1)^2$ neighbor cells, where r is a positive integer number that represents the radius of the r-neighborhood. A 3×3 grid of cells has an r value of 1, and a 4×4 a value of 2. Since the neighborhood of a cell is chosen to be as small as possible a 3×3 matrix is often used.

Cimagalli *et al* [14] present a partially connected CNN neural architecture for spatio-temporal detection. The system is capable of detecting and tracking moving objects in an image. In order to deal with spatio temporal detection, edges of the image are mapped into neuron states, as opposed to the pixels themselves as commonly used in image processing. The system has nine neurons for every pixel of the image (excluding border pixels), which represent edges directed towards eight nearest neighbors, and to the same pixel for static objects. Connections are limited to a neighborhood of order three. A constraint on direction is imposed since a moving object has a finite speed and cannot jump from a given pixel to a non adjacent one. Thus, each off-center neuron (boundary neuron) is only connected to the four neurons, of a total of nine, corresponding to a trajectory within 45 degrees from the present direction of the moving object.

Liu and Michel [54] present a synthesis procedure for sparsely interconnected neural networks. They apply this synthesis procedure to the design of CNNs for associative memories which have the basic structure of analog Hopfield neural networks. The sparse design problem can be stated as follows: Given an $n \times n$ index matrix $S = [\, S_{ij} \,]$, with $S_{ii} \neq 0$ for $i = 1, ..., n$, and m vectors $\alpha^1, ..., \alpha^m$ in B^n, choose $\{A, T, I\}$ with $T = T \mid S$ in such a manner that $\alpha^1, ..., \alpha^m$ are memory vectors, where an index matrix $S = [S_{ij}] \in R^{n \times n}$ corresponds to an index matrix if it satisfies $S_{ij} = 1$ or 0, T is the connection matrix, I is the bias vector, A is an arbitrary matrix, and $B^n = \{x \in R^n : x_i = 1 \, or \, -1, i = 1, ..., n\}$. A ten step algorithm for solving the sparse design problem is presented by the authors. Several cases synthesized by their procedure are presented in this paper. These cases involve different pre-specified constraints on the interconnecting structure of each network. A classification problem consisting in storing four patterns as memories in a 4×4 input grid is used to obtain the level of performance of the model based on the number of steps taken by each model to converge. In one of the models, the authors consider only horizontal and vertical interconnections. Convergence is achieved in 9 steps with this model. Another model is presented on which the structure of network does not contain any line-crossing in the interconnecting structure. Convergence with this model is obtained within 13 steps.

3 Ontogenic Methods

3.1 Multilayer Perceptron Based Methods

The multilayer perceptron is the most popular neural network used for supervised learning. The network is arranged in layers with inter- and sometimes surpralayer connections between the neurons. There is an input and an output layer and one or more hidden layers. During recall, information flows exclusively in a feedforward direction from the input layer towards the output layer. The most commonly used multilayer perceptron network training algorithm is the error backpropagation.

The function of the backpropagation network is to reproduce certain target output patterns at the output layer. This is done by adjusting each connection weight according to a mathematical expression that compares the activity patterns at the output layer with those of the target patterns, and propagates the difference, or error, back through the network contributing a small adjustment to the strength of each connection. The learning algorithm uses the principle of gradient descendent to alter the value of each connection weight in the direction for which the change in that particular parameter moves the output activity patterns closer to the actual patterns [83] [69] [50].

The ontogenic methods based on the backpropagation algorithm can be classified into three groups: growing, pruning, and the methods which combine both growing and pruning techniques [22].

Connection Pruning Methods

Pruning methods start by training a neural network with a topology bigger than needed. Due to it being oversized, the initial network will be less sensitive to initial conditions such as weight initialization and learning parameters. Thus, the initial network is expected to learn reasonably fast. After this, the network is trimmed until the smallest topology that correctly maps the data is found. A summary of some pruning methods that exist for backpropagation which result in PCNNs, is presented in table 1.

Method	Reference
Energy Functions	[11]
Penalty Function	[33]
Modification of backpropagation Error Function	[43]
Optimal Brain Damage	[52]
Ockhams Razor	[77]
The Upstart Algorithm	[26]
Weight Elimination	[82]
Optimal Brain Surgeon	[37]

Table 1: Multilayer Perceptron Pruning Methods

Method	Reference
Cut and/or Create Connections	[29]
Units Recruitment	[39] [41]
Removal of Worth Hidden Neuron	[31]
Addition of Hidden Units to help Avoid Local Minimum	[38]

Table 2: Growing and Pruning Multilayer Perceptron Ontogenic Methods

Pruning Methods Combined With Growing Methods

There are some methods which combine pruning techniques with growing methods. Growing methods start with a small topology which increases until the neural network achieves a good level of performance for the given method. Table 2 displays a summary of such methods.

3.2 Other ontogenic methods

Besides multilayer feedforward based methods, there are other ontogenic methods that produce PCNNs which are presented in table 3.

4 Hybrid Methods

4.1 Knowledge Based PCNNs

This section presents some of the work done on the combination of symbolic knowledge and neural networks to produce PCNNs. The idea behind these methods is to use symbolic domain knowledge as rules

Method	Reference
Use of Monte Carlo Procedures for Architecture Optimization on High Order Neural Networks	[48]
Limited Fan-in Random Wired Cascade-Correlation	[44]

Table 3: Other Ontogenic Methods

to actually create the topology of the neural network. Next, the actual neural training algorithm is used to improve the performance of the network.

The KBANN (**K**nowledge **B**ased **A**rtificial **N**eural **N**etwork) system for combining symbolic knowledge and neural networks is described in [78], [80], and [79]. KBANN is a hybrid learning method which combines explanation based learning, containing rules of a given task, with empirical based learning.

In the KBANN system a set of approximately correct rules is used to determine the structure and initial weights of a neural network. These rules are expressed as Horn clauses. A set of mathematical proofs is given in order to show the correctness of the rules-to-neural-network translator.

After the symbolic rules are inserted into the network, the network is refined using standard neural learning algorithms and a set of classified training examples. A *heuristic-like* method for initializing the network weights is provided. Once the network is trained (examples based on the backpropagation learning algorithm are presented), the refined set of rules is extracted from the network. The authors find the following correspondence between knowledge based systems and neural networks:

Supporting facts ⇒ Input units
Dependencies ⇒ Connection weights
Intermediate conclusions ⇒ Hidden units
Final conclusions ⇒ Output units

Some of the drawbacks of this method include:

- Once the topology of the model is defined by KBANN, the user is expected to specify additional hidden units at any specific level. The reason for this is that KBANN may need some more hidden units in addition to those specified by the domain theory in order to improve the networks accuracy.
- There is a need to add input units for known features not used in the rules. This is needed because a set of rules that is only approximately correct may not identify all the input features that are required for correctly learning a concept.
- Additional links, **not** specified by the translation, between all units in topologically contiguous levels (all these links are set to an initial value of 0) and at different levels, are needed. These include intra and supralayer connections. These new connections are created such that dependencies that were not previously specified can be discovered. There is no method given for selecting the minimum set of links needed.

An extension of the KBANN system, called FAKBANN, is presented by Maclin and Shavlik [55]. This system translates rule-like knowledge, represented as a finite automata, into a corresponding neural network.
The main extension of this system is that the domain theory, which uses state information represented as finite state automata, is mapped into a recurrent network where a subset of the neural network output is copied back as input to the network in the next step. This copied output represents the current state calculated by the network and can be used for calculating the succeeding state.
A state in the domain theory represents the context of the problem. Rules introduced to solve a problem take into account the state of the problem.

4.2 Genetic Programming Based Methods

Genetic algorithms (GAs) are guided random search algorithms based on natural selection and genetics. The algorithms are composed of four major operators: selection, crossover (mating two individuals), inversion, and mutation (a small amount of noise added to an individual). They start the search for a solution from a random initial population that evolves towards a population of best fitted individuals. For a more detailed description on GAs the reader can refer to Goldberg [30]

Description	*Weights*	*Reference*
Introduction of the notion of connectivity path	No	[17]
GA and permutation problems on NN	No	[32]
Bumptree NN topology optimization	No	[84]
Structural GA for designing application specific NN	Yes	[15]
ANNA ELEONORA. Parallel GA	Yes	[56]
Discussion of several hybrid GA/NN methods	Yes	[9]
GA encoding based on marker structure of DNA	No	[27]
General GA and NN introductory paper	Yes	[72]
Notion of *blueprint* for encoding NN characteristics such as structural properties and learning parameter values	No	[34] [35]

Table 4: GA based PCNN

The use of genetic algorithms to assist neural networks has been intensively studied in the past few years. These efforts concentrate on optimizing the topology of the neural network, and on finding the ideal set of weights for a given topology.

Table 4, gives a summary of some of the GA based PCNN papers. All the methods presented optimize the neural networks topology, and some optimize the initial set of weights.The conventions used in this table are:

1. *Description.* Provides a brief description of the method.
2. *Weights.* Indicates if the method optimizes the initial values for the neural network weights.
3. *Reference.* Bibliographical reference code.

Due to the extensive amount of literature found in the subject of PCNN by means of GA, only some of the most relevant publications are summarized in this table. For a detailed survey in combining GAs and NNs see [71] and [88].

5 Summary and Conclusions

This section provides a summary of the principle characteristics of the non-ontogenic methods reviewed in this survey paper which are recapitulated in table 5.

The conventions used in this table are: the bibliographical reference code (*Reference*), the type of topology of the neural network neural network studied (*Topology* [1]), the number of neuron layers (for layered topologies) of the neural network model studied (L), if the actual number is fixed, it is shown, otherwise a "x" for variable is shown, the type of connections that the proposed method uses (see [23]) (TC), intra ($\bowtie$), inter ($\diamondsuit$), and supra ($\bigcirc$), or is plenary connected ($\bigotimes$), whether the neural network has symmetric "S" or asymmetric "A" connections (*Sym.*), if the method uses random connectivity (*Ran.*), if the method uses local connectivity (*Local*), if the method uses high order connections (*HO*), the *fan in/out* value ($F_{i/o}$) of the method, and the average sparsity value (AS) of the method which represents the amount of connection reduction made overall the neural network topology. In both the ($F_{i/o}$) and the (AS) an "x" for variable, an "C" for constant, and an "L" for limited are used when the value is not specified. In general, a "Y" indicates that condition is satisfied, an "N" that it is not, an "-" that this

[1] A *uniform* topology is one on which all the neurons play the same role and can be thus viewed as a one layer network

does not apply to the method/model under study, and a "?" that the information is not available in the publication.

5.1 Non-Ontogenic Methods

5.1.1 Theoretical and Experimental Studies on PCNNs

Most of the methods studied in this section are aimed at demonstrating the consequences of partially connected networks in terms of their dynamics. The methods used for searching an optimal connectivity level include:

- random arbitrary selection of the units, often within the nearest neighborhood, to which each neuron will connect to,
- the application of the principle of *weight dilution*, which consists on the random cutting of connections within a given probability,
- the variation of the connectivity level from minimal to plenary connectivity in order to find the connectivity level which produces the best performance in terms of the percentage of well classified patterns,
- the use of exhaustive methods which generate all possible topologies that might yield a solution,
- the use of high order connections.

The results obtained in several of the studies suggest that the level of performance increases by using partial connectivity. They also suggest that there is an optimal connectivity level (which is problem dependent) for obtaining the best performance.

5.1.2 PCNN Derived From Biological Neural Networks

The methods studied in this section are inspired by the sparse connectivity level of neurons found in biological systems. They are in general aimed at studying the dynamics and storage capacity of biological PCNN. Some of the methods study the neural connection structure and functioning. Others are interested in reducing statistical dependence among inputs by means of sparse connectivity. The connectivity of these PCNN models is based on:

- simulation of neural spike activity trains,
- statistical measures (probability and distribution),
- non-Hebbian processes which model biological synaptic growth, or
- random connection selection.

5.1.3 Application Dependent PCNNs

This section presents PCNN models developed for such domains as speech recognition and image processing. The actual topology of the neural network is designed by the developer based on prior knowledge from the given application and relationships found by a-priori analysis of the data. These methods are therefore usually not applicable outside their domain. The connectivity strategies are based on:

- a subset of consecutive units to connect to each hidden unit,
- nearest neighbor connections,
- local connections,
- shared weights,
- geometrical relationships between input patterns, or
- high order connections.

Reference	*Topology*	*L*	*TC*	*Sym.*	*Ran.*	*Local*	*HO*	$F_{i/o}$	*AS*
Methods Based on Theoretical and Experimental Studies									
Multilayered Networks									
[5]	Layered	x	◇/◯	A	Y	?	?	x	x
[21]	Layered	2	◇	A	N	Y	Y	x	x
[36]	Layered	3	◇	A	N	N	N	?	?
[46]	Layered	3	◇	A	N	Y	N	C	C
[65]	Layered	x	◇	A	N	Y	N	3	x
[66]	Layered	3	◇	A	N	N	Y	x	x
[87]	Layered	3	◇	A	N	N	Y	x	x
Associative Memory									
[4]	Layered	-	⊗	A	Y	Y	Y	L	?
[6]	Layered	3	◇	S	?	Y	N	?	x
[18]	Layered	?	?	S	Y	?	N	x	x
[19]	Unstructured	1	◇	S	Y	?	N	x	x
[28]	Unstructured	1	◇	S	Y	N	N	x	x
[47]	Lattice	?	?	?	Y	?	?	x	x
[61]	Layered	?	⊗	A	Y	?	?	x	x
[81]	Unstructured	1	?	A	Y	N	N	x	x
[86]	Layered	2	◇/◯	SA	Y	?	?	?	x
Methods Derived from Biological Neural Networks									
[1] [2]	Layered	2	◇	S	Y	Y	N	x	x
[7]	Layered	2	◇	A	N	Y	N	C	C
[64]	Layered	x	◇	?	Y	N	Y	x	x
[74]	Layered	x	◇	?	Y	N	Y	x	x
[75]	Layered	2	◇/◯	A	N	N	N	x	x
Application Dependent Methods									
Speech Recognition and Linguistics									
[49]	Layered	2/3	◇	A	N	Y	N	x	x
[67]	Layered	5	◇	A	N	Y	N	C	C
[89]	Layered	3	◇	A	N	N	N	x	x
Image Processing									
[40]	Layered/Pyramid	5	◇	A	Y	N	N	x	x
[45]	Layered	4	◇	A	N	Y	N	C	x
[63]	Layered	2	◇	A	N	N	Y	-	-
[73]	Layered	3	◇	A	N	Y	N	?	?
[76]	Layered	2	◇	A	N	Y	Y	C	C
[85]	Layered	2	A	N	N	N	N	10	10
Weight Sharing									
[51]	Layered	5	◇	A	N	Y	N	C	C
[62]	Layered	3	◇	A	N	N	N	-	-
Modular Neural Networks									
[25]	Layered	x	◇	A	N	Y	N	x	x
[16]	Layered	x	◇/◯	A	N	Y	N	x	x
[42]	Layered	x	◯	A	N	Y	N	x	?
[59]	Layered	x	◇	A	N	Y	N	x	x
Methods Developed for Hardware Implementation									
[3]	Layered	x	◇	A	N	Y	N	x	x
[8]	Layered	x	◇	A	N	?	N	x	x
[20]	Layered	x	◇	A	N	N	N	x	x
[58]	Layered	x	◇	A	N	Y	N	x	x
Cellular Neural Networks									
[14]	Cellular	-	◇	A	N	Y	N	L	L
[54]	Cellular	-	◇	A	N	Y	N	L	L
Hybrid Knowledge Based Methods									
[55]	Layered	x	◇	A	N	Y	N	x	x
[78] [80] [79]	Layered	x	◇	A	N	Y	N	x	x

Table 5: Non-ontogenic SCNN methods

5.1.4 Modular based PCNNs

Modular based PCNNs are designed by dividing a task into sub-tasks. The way this division is done depends on the problem at hand and is a key aspect for the good performance of the network. These sub-tasks are represented by specialized sub-networks with possibly different topologies. The different sub-networks are ussually linked with each other by a decision layer which selects the model output value in a *winner take all* fashion. An in-depth survey on modular neural networks can be found in [68]. As mentioned by the authors, an improvement in the way tasks are decomposed is necessary to make modular networks more widely used. At present, spatial clustering appears to be the only method for task decomposition. Thus, *ah hoc* techniques are used for decomposition in function approximation problems.

5.1.5 PCNNs for Hardware Implementation

The methods presented in this section are motivated by the physical connectivity limitations presented in the VLSI technology. The main advantage of using VLSI neural network hardware implementations over software ones is the gain in training and recall speed. Some of these methods propose the restructuration of the original fully connected neural network topology into partially connected ones. The reduction in number of connections made by this restructuration produces a reduction in the degrees of freedom. This is often compensated for by an increase in the number of nodes in the topology of the model.
Other methods limit the connectivity of each neuron (chosen either randomly or by trial and error) of all the possible neighboring neurons the given neuron can connect to.
In cellular neural networks, every cell is locally connected only to its neighbor cells within a given distance.

5.1.6 Other Methods

5.2 Ontogenic Methods

As discussed by Fiesler [22], most ontogenic based PCNNs are generally aimed at improving the neural network generalization capacity by removing connections from a FCNN until the topology that correctly maps the data is found. Some of these methods remove connections from an already trained network, others remove them during the training process. Often, when connections are removed, it becomes necessary to retrain the network.

5.3 Hybrid Methods

5.3.1 Knowledge Based PCNNs

Even though these methods are still in a very early stage of development, and primarily aimed at understanding and explaining the representations formed by neural networks, they provide a sound way for defining a partially connected topology of a network based on domain knowledge.
An interesting extension of this work will be the combination with an automatic rule extractor which will automatically produce the rules from the training data set instead of the user searching for the rules.

5.3.2 Genetic Programming Based Methods

The use of genetic algorithms (GA) to assist neural networks has been intensively studied in the past few years. These efforts concentrate on optimizing the topology of the neural network, and/or on finding the ideal set of weights for a given topology. The GA based methods for optimizing the topology of the neural network summarized in this publication differ from each other mainly in the way they encode the neural network into the GA chromosome, and the way they evaluate the fitness or performance of each individual. The main drawback of these methods remains the excessive computational time required to find a good network topology. Several authors have suggested ways for reducing this computational time which include, among others, the use of specialized neural network hardware and the use of parallel techniques for programming GAs.

Acknowledgments

The authors would like to thank E. Dizdarevic, F. Hammadi-Mesmoudi, V. Honavar, A. Ketterlin, R. Mason, and D. Tveter for providing publications, technical reports, and personal communications on the subject of partial connectivity.

References

[1] Adelsberger-Mangan, D., and Levy, W. Correlation matrices and the construction of succesful network recodings. *IEEE IV* (1992), 808–813.

[2] Adelsberger-Mangan, D., and Levy, W. Information maintenance and statistical dependence reduction in simple neural networks. *Biological Cybernetics 67*, 5 (1992), 469–477.

[3] Antola, A., Stefanelli, R., and Storti-Gajani, G. Radix-r implementation of neural nets. In *Silicon Architectures for Neural Nets. Proceedings of the IFIP WG 10.5 Workshop* (Amsterdam, Netherlands, 1991), M. Sami and J. Valzadilla-Daguerre, Eds., Conf. Sponsor: IFIP, North-Holland, pp. 153–165.

[4] Banzhaf, W., Ishii, T., Nara, S., and Nakayama, T. A sparsely connected asymmetric neural network and its possible application to the processing of transient spatio-temporal signals. In *INNC 90 Paris. International Neural Network Conference* (Dordrecht, Netherlands, 1990), vol. 2, Thomsom; SUN; British Comput. Soc., Kluwer, pp. 1005–1008.

[5] Barkai, E., Kanter, I., and Sompolinsky, H. Properties of sparsely connected excitatory neural networks. *Physical Review A; Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics 41*, 2 (January 1990), 590–597.

[6] Barna, G., and Kaski, K. Choosing optimal network structure. In *Proceedings of the International Neural Network Conference (INNC)* (1990), Kluwer Academic, pp. 890–893.

[7] Bauer, S. M., and Acharya, R. Synaptic growth as a learning model. In *Artificial Neural Networks*, T. Kohonen, K. Makisara, O. Simula, and J. Kangas, Eds. Elsevier Science Publications B. V. (North Holland), 1991, pp. 1169–1172.

[8] Beiu, V., Peperstraete, J. A., and Lauwereins, R. Algorithms for fan-in reduction. In *Proceedings of the International Workshop on Neural Networks and Their Applications, Neuro-Nimes* (1992), pp. 589–600.

[9] Belew, R. K., McInerney, J., and Schraudolph, N. N. Evolving neural networks: Using genetic algorithms with connectionist learning. Tech. rep., University of California at San Diego, La Jolla, California, June 1990.

[10] Caudill, M., and Butler, C. *Understanding neural networks volume 1: basic networks*, vol. I. The MIT Press, 1992.

[11] Chauvin, Y. A back-propagation algorithm with optimal use of hidden units. In *Advances in Neural Information Processing Systems (NIPS) 1* (San Mateo, California, 1989), D. S. Touretzky, Ed., IEEE, Morgan Kaufmann, pp. 519–526.

[12] Chua, L. O., and Yang, L. Cellular neural networks: Applications. *IEEE Transactions on Circuits and Systems 35*, 10 (October 1988), 1273–1290.

[13] Chua, L. O., and Yang, L. Cellular neural networks: Theory. *IEEE Transactions on Circuits and Systems 35*, 10 (October 1988), 1257–1272.

[14] Cimagalli, V., Bobbi, M., and Balsi, M. Moda: Moving object detecting architecture. *IEEE Transactions on Circuits and Systems 40*, 3 (October 1993), 174–183.

[15] Dasgupta, D., and McGregor, D. R. Designing application specific neural networks using the structured genetic algorithm. In *International Workshop on Combinations of Genetic Algorithms and Neural Networks* (1992), IEEE Computer Society Press, pp. 1–37.

[16] de Francesco, M. *Functional Networks; A New Computational Framework for the Specification, Simulation and Algebraic Manipulation of Modular Neural Systems.* PhD thesis, Département d'Informatique, Université de Genève, Geneva, Switzerland, 1994.

[17] Dizdarevic, E. Optimisation de l'architecture de reseaux de neurones par algorithmes genetiques. Tech. rep., Universite Louis Pasteur, 7, Rue Rene Descartes, F-67084 Strasbourg, France, September 1994.

[18] Doyon, B., Cessac, B., Quoy, M., and Samuelides, M. Control of the transition to chaos in neural networks with random connectivity. *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering 3*, 2 (April 1993), 279–91.

[19] Englisch, H., Xiao, Y., and Yao, K. Strongly diluted networks with selfinteraction. NTZ-Preprint 13/1991, Naturwissenschaftlich - Theoretisches Zentrum (NTZ), Universität Leipzig, Leipzig, Germany, 1991.

[20] Faure, B., and Mazaré, G. A VLSI implementation of multi-layered neural networks: 2 - performance. In *Proceedings of the International Workshop on VLSI for Artificial Intelligence and Neural Networks* (New York, New York, 1991), J. G. Delgado-Frias and W. R. Moore, Eds., University of Oxford Department for External Studies in conjunction with the Dept. of Engineering Science and the Dept. of EE at SUNY-Binghamton, Plenum Press, pp. 377–386.

[21] Fiesler, E. Minimal and high order neural network topologies. In *Proceedings of the 1993 International Simulation Technology Conference (SIM Tec 93)* (1993), Society for Computer Simulations (SCS), pp. 173–178.

[22] Fiesler, E. Comparative bibliography of ontogenic neural networks. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN 94)* (London, U.K., 1994), M. Marinaro and P. G. Morasso, Eds., vol. 1, Springer-Verlag, pp. 793–796.

[23] Fiesler, E. Neural network classification and formalization. *Computer Standards & Interfaces, special issue on Neural Network Standards 16*, 3 (1994), 231–239.

[24] Fiesler, E., and Beale, R. *Handbook of neural computation.* Oxford University Press, New York, NY, 1996.

[25] Fogelman, F., Viennet, E., and Lamy, B. Multi-modular neural network architectures: Applications in optical character and human face recognition. *Int. Journal of Pattern Recognition and Artificial Intelligence 7*, 4 (1993), 721–755.

[26] Frean, M. The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation 2*, 2 (Summer 1990), 198–209.

[27] Fullmer, B., and Miikkulainen, R. Using marker-based genetic encoding of neural networks to evolve finite-state behaviour. In *Toward a Practice of Autonomous Systems. Proceedings of the First European Conference on Artificial Life* (Cambridge, MA, USA, 1992), F. J. Varela and P. Bourgine, Eds., MIT Press, pp. 255–262.

[28] Gardner, E. Optimal basins of attraction in randomly sparse neural network models. *J. Phys. A: Math. Gen. 22* (1989), 1969–1974.

[29] Gloger, W., and Hausler, G. Neural nets with reduced connectivity for the processing of large pictures. *International Journal of Optical Computing 2*, 4 (Dec. 1991), 425–431.

[30] Goldberg, D. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[31] Hagiwara, M. Novel back propagation algorithm for reduction of hidden units and acceleration of convergence using artificial selection. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN) San Diego* (Ann Arbor, Michigan, 1990), vol. I, IEEE/INNS, IEEE Neural Networks Council; Edward Brothers, pp. 625–630.

[32] Hancock, P. J. B. Genetic algorithms and permutation problems: a comparison of recombination operators for neural net structure specification. In *Proceedings of the international workshop on combinations of genetic algorithms and neural networks* (1992), IEEE Computer Society Press.

[33] Hanson, S. J., and Pratt, L. Y. Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems 1* (San Mateo, California, 1989), D. S. Touretzky, Ed., IEEE, Morgan Kaufmann, pp. 177–185.

[34] Harp, S. A., Samad, T., and Guha, A. Designing application-specific neural networks using the genetic algorithm. In *Advances in Neural Information Processing Systems (NIPS-IEEE)*, D. S. Touretzky, Ed., vol. 2. Morgan Kaufmann Publishers, 2929 Campus Drive, Suite 260, San Mateo, California 94403, 1990, pp. 447–454.

[35] Harp, S. A., Samad, T., and Guha, A. Genetic synthesis of neural networks. In *The Genetic Algorithms Handbook*, L. D. Davis, Ed. Van Nostrand Reinhold, NewYork, 1990, pp. 202–221.

[36] Hasen, L. K., Lautrup, B., Law, I., Morch, N., and Thomsen, J. Extremely ill-posed learning. Tech. rep., Electronics Institute, build. 349, Technical University of Denmark, DK-2800, Lyngby, Denmark, August 1994.

[37] Hassibi, B., and Stork, D. G. Second order derivatives for network pruning: Optimal brain surgeon. Tech. Rep. CRC-TR-9214, RICOH California Research Center, Menlo Park, California, U.S.A., May 7 1992.

[38] Hirose, Y., Yamashita, K., and Hijaya, S. Back-propagation algorithm which varies the number of hidden units. *Neural Networks 4* (1991), 61–66.

[39] Honavar, V., and Uhr, L. Experimantal results indicate that generation, local receptive fields and gloval convergence improve perceptual learning in connectionist networks. Tech. rep., Computer Science Department, University of Wisconsin-Madison, Wisconsin, USA, 1988.

[40] Honavar, V., and Uhr, L. A network of neuron like units that learns to perceive by generation as well as reweighting of its links. In *Proceedings of the 1988 Connectionist Models Summer School* (1988), D. Touretzky, G. Hinton, and T. Sejnowski, Eds., Morgan Kaufmann, pp. 472–484.

[41] Honavar, V., and Uhr, L. Generative learning structures and processes for generalized connectionist networks. Tech. Rep. 91-02, Department of computer science, Iowa State University, 1991.

[42] Jacobs, R., and Jordan, M. Learning piecewise control strategies in a modular neural network architecture. *IEEE Transactions on Systems, Man, and Cybernetics 23*, 2 (1993), 337–345.

[43] Ji, C., Snapp, R. R., and Psaltis, D. Generalizing smoothness constraints from discrete samples. *Neural Computation 2*, 2 (Summer 1990), 188–197.

[44] Klagges, H., and Soegtrop, M. Limited fan-in random wired cascade-correlation. In *... MicroNeuro '93 ...* (1993).

[45] Korczak, J., and Hammadi-Mesmoudi, F. A way to improve an architecture of neural network classifier for remote sensing applications. *Neural Processing Letters 1*, 1 (September 1994), 13–16.

[46] Kung, S. Y., Hwang, J., and Sun, S. W. Efficient modeling for multilayer feed-forward neural nets. In *Proceedings of ICASSP* (1988), vol. 4, pp. 2160–2163.

[47] Kurten, K. E. Quasi-optimized memorization and retrieval dynamics in sparsely connected neural network models. *Journal de Physique 51*, 15 (August 1990), 1585–1594.

[48] Kurten, K. E. Optimal architectures and high-order networks. In *Neural Network Dynamics. Proceedings of the Workshop on Complex Dynamics in Neural Networks* (Berlin, Germany, 1992), J. G. Taylor, E. R. Caianiello, R. M. J. Cotterill, and J. W. Clark, Eds., Springer-Verlag, pp. 44–56.

[49] Lang, K. J., Waibel, A. H., and Hinton, G. E. A time-delay neural network architecture for isolated word recognition. *Neural Networks 3*, 1 (1990), 23–43.

[50] Le Cun, Y. *Modéles Connexionistes de l'Apprentissage*. PhD thesis, Université Pierre et Marie Curie, Paris, France, 1987.

[51] Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation 1*, 4 (Winter 1989), 541–551.

[52] Le Cun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In *Advances in Neural Information Processing Systems (NIPS)* (San Mateo, California, 1990), D. S. Touretzky, Ed., vol. 2, IEEE, Morgan Kaufmann, pp. 598–605.

[53] Levy, W. B. A computational approach to hippocampal function. In *Computational models of learning in simple neural systems. The psychology of learning and motivation*, R. D. Hawukins and G. H. Bower, Eds. Academic Press, San Diego, CA, USA, 1989, ch. 23, pp. 243–305.

[54] Liu, D., and Michel, A. N. Sparsely interconnected neural networks for associative memories with applications to cellular neural networks. *IEEE Transactions on Circuits and Systems 41*, 4 (April 1994), 295–307.

[55] Maclin, R., and Shavlik, J. W. Using knowledge-based neural networks to improve algorithms: Refining chou-fasman algorithm for protein folding. *Machine Learning 11* (1993), 195–215.

[56] Maniezzo, V. Genetic evolution of the topology and weight distribution of neural networks. *Transactions on Neural Networks 5*, 1 (1993), 39–53.

[57] Martindale, C. *Cognitive psycology: a neural network approach*. Brooks Cole Publishing Company, 1991.

[58] Mason, R., Robertson, W., and Pincok, D. A hierarchical vlsi neural network architecture. *IEEE Journal of Solid State Circuits 27*, 1 (1992), 106–108.

[59] Mason, R. D. *Hierarchical VLSI Neural Networks*. PhD thesis, Technical University of Nova Scotia, Nova Scotia, Canada, 1990.

[60] McGill, W. J. Multivariate information transmission. *IRE Trans. Inf. Theory I* (1955), 93–111.

[61] Minai, A. A., and Levy, W. B. The dynamics of sparse random networks. *Biological Cybernetics 70* (1993), 177–187.

[62] Nowlan, S. J., and Hinton, G. E. Simplifying neural networks by soft weight sharing. *Neural Computation 4*, 4 (July 1992), 681–695.

[63] Perantonis, S. J., and Lisboa, P. J. G. Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers. *IEEE Transactions on Neural Networks 3*, 2 (March 1992), 241–251.

[64] Peretto, P., and Niez, J. J. Long term memory storage capacity of multiconnected neural networks. *Biological Cybernetics 54* (1986), 53–63.

[65] Provence, J., and Naganathan, S. Locally versus globally interlayer connected feed-forward neural networks: a performance comparison. *Proceedings of the SPIE - The International Society for Optical Engineering 1293*, 1 (1990), 278–289.

[66] Redding, N. J., Kowalczyk, A., and Downs, T. Higher order separability and minimal hidden-unit fan-in. In *Artificial Neural Networks; Proceedings of the 1991 International Conference on Artificial Neural Networks (ICANN-91)* (Amsterdam, The Netherlands, 1991), T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, Eds., vol. 1, Pattern Recognition Society of Finland and Helsinki University of Technology, North-Holland; Elsevier Science Publishing Company B.V,, pp. 25–30.

[67] Regier, T. Learning spatial concepts using a partially- structured connectionist architecture. Tech. rep., University of Berkeley, Berkeley, California, USA, October 1991.

[68] Ronco, E., and Gawthrop, P. Modular neural networks: a state of the art. Tech. Rep. CSC-95026, Glasgow University, 1995.

[69] Rumelhart, D. E., McClelland, J. L., and the PDP Research Group. *Parallel Distributed Processing*, vol. 1. The MIT Press, Cambridge, Massachusetts, 1986.

[70] Sanger, T. D. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks 2* (1989), 459–473.

[71] Schaffer, J. D., Whitley, D., and Eshelman, L. J. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In *International Workshop on Combinations of Genetic Algorithms and Neural Networks* (1992), IEEE Computer Society Press, pp. 1–37.

[72] Schiffmann, W., Joost, M., and Werner, R. Synthesis and performance analysis of multilayer neural network architectures. Tech. Rep. 16/1992, University of Kobleux, 1992.

[73] Schmitt, M., and Vallet, F. Network configuration and initialization using mathematical morphology: Theoretical study of measurement functions. In *Artificial Neural Networks*, T. Kohonen, K. Makisara, O. Simula, and J. Kangas, Eds. Elsevier Science Publications B. V. (North Holland), 1991, pp. 1045–1048.

[74] Senn, W., Wyler, K., Striet, J., Larkum, M., Lüscher, H.-R., Mey, H., Müller, L., Stainhauser, D., Vogt, K., and Wannier, T. Dynamics of a random neural network with synaptic depression. Preprint, December 8 1994.

[75] Shiono, S., Yamada, S., Nakashima, M., and Matsumoto, K. Information theoretic analysis of connection structure from spike trains. In *Advances in Neural Information Processing Systems 5*, e. a. e. Hanson S. J., Ed. John Wiley and Sons, Inc., 1992, ch. 10, pp. 153–184.

[76] Spirkovska, L., and Reid, M. B. Connectivity strategies for higher-order neural networks applied to pattern recognition. In *IJCNN International Joint Conference on Neural Networks (Cat. No.90CH2879-5)* (New York, NY, USA, 1990), vol. 1, IEEE; Int. Neural Network Soc, IEEE, pp. 21–26.

[77] Thodberg, H. H. Improving generalization of neural networks through pruning. *International Journal of Neural Systems 1*, 4 (1991), 317–326.

[78] Towell, G. G. *Symbolic knowledge and neural networks: insertion, refinement, and extraction.* PhD thesis, Univerity of Wisconsin, Madison, Wisconsin, USA, 1991.

[79] Towell, G. G., Craven, M. W., and Shavlik, J. W. Constructive induction in knowledge-based neural networks. In *Proceedings of the Eight International Workshop on Machine Learning* (San Mateo, CA, 1991), L. A. Birnbaum and G. C. Collins, Eds., Morgan Kaufmanm, pp. 213–217.

[80] Towell, G. G., and Shavlik, J. W. *Refining symbolic knowledge using neural networks.* Machine Learning Research group Working Paper, 1991.

[81] Venkatesh, S. S. Robustness in neural computation: Random graphs and sparsity. 11 pages., December 30 1990.

[82] Weigend, A. S., Rumelhart, D. E., and Huberman, B. A. Generalization by weight-elimination with application to forecasting. In *Advances in Neural Information Processing Systems (NIPS) - Natural and Synthetic 3* (San Mateo, California, 1991), R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds., Morgan Kaufmann Publishers, pp. 875–882.

[83] Werbos, P. J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, Massachusetts, August 1974.

[84] Williams, B. V., Bostock, R. T. J., Bounds, D., and Harget, A. Improving classification performance in the bumptree network by optimising topology with a genetic algorithm. In *IEEE Conf. Evolutionary Computation at WCCI* (1994).

[85] Wu, R.-Y., and Tsai, W.-H. A single-layer neural network for parallel thinning. *International Journal of Neural Systems 3*, 4 (1992), 395–404.

[86] Yanai, H. F., Sawada, Y., and Yoshizawa, S. Dynamics of an auto-associative neural network model with arbitrary connectivity and noise in the threshold. *Network: Computation in Neural Systems 2*, 3 (August 1991), 295–314.

[87] Yang, H., and Guest, C. C. High order neural networks with reduced numbers of interconnection weights. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN) San Diego* (Ann Arbor, Michigan, 1990), vol. III, IEEE+INNS, IEEE Neural Networks Council; Edward Brothers, pp. 281–286.

[88] Yao, X. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems 8*, 4 (April 1993), 539–567.

[89] Ye, H., Wang, S., and Robert, F. A pcmn neural network for isolated word recognition. *Speech Communication 9*, 2 (April 1990), 141–53.