Preliminary version of chapter B2 of the Handbook of Neural Computation.

Neural Network Topologies

E. Fiesler

IDIAP, C.P. 592, CH-1920 Martigny, Switzerland

1 Introduction

A neural network is a network of neurons. This high level definition applies to both biological neural networks and artificial neural networks. This chapter is mainly concerned with the various ways in which neurons can be interconnected to form the networks or network topologies used in artificial neural networks, even though some underlying principles are also applicable to their biological counterparts. The term 'neural network' is therefore used to stand for 'artificial neural network' in the remainder of this chapter, unless explicitly stated otherwise. The main purpose of this chapter is to provide a base for the rest of the handbook and of the next chapter in specific, in which the training of artificial neural networks is discussed.

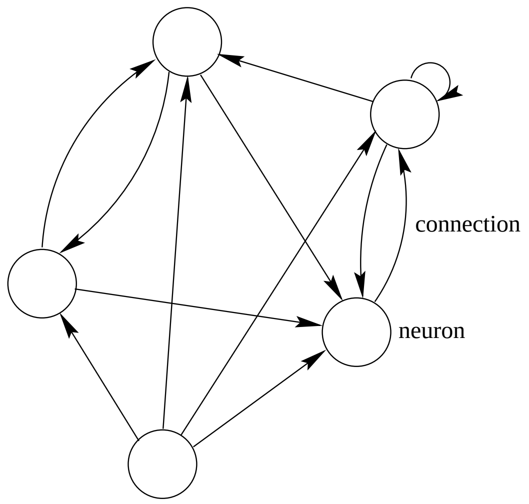

Figure 1: An unstructured neural network topology with five neurons.

Figure 1 shows an example neural network topology. A node in such a network is usually called *artificial neuron*, or simply *neuron*, a tradition that is continued in this handbook; see section B.1: The Artificial Neuron [Hyperlink to ??]. The widely accepted term 'artificial neuron' is specific to the field of artificial neural networks and therefore preferred over its alternatives. Nevertheless, given the length of this term and the need to frequently use it, it is not surprising that its abbreviated form, 'neuron', is often used as a substitute instead. However, given that the primary meaning of the word 'neuron' is a biological cell from the central nervous system of animals, it is good practice to clearly specify the meaning of the term 'neuron' when using it. Instead of '(artificial) neuron', other terms are also used:

- *Node* This is a generic term, related to the word 'knot' and used in a variety of contexts, one of them being graph theory, which offers a mathematical framework to describe neural network topologies (see section 8.4).
- *Cell* An even more generic term, that is more naturally associated with the building blocks of organisms.
- *Unit* A very general term used in numerous contexts.
- *Neurode* A nice short term coined by Caudill and Butler [Caudill-90], which contains elements of both the words 'neuron' and 'node', giving a cybernetic flavor to the word 'neuron'.

The first three words are generic terms, borrowed from other fields, which can serve as alternative terminology as long as their meaning is well defined when used in a neural network context. The neologism 'neurode' is specifically created for artificial neural networks, but unfortunately not widely known and accepted.

A connectionist system, better known as artificial neural network, is in principle an abstract entity. It can be described mathematically and can be manifested in various ways, as for example in hardware and software implementations. An artificial neural network comprises a collection of artificial neurons connected by a set of links, which function as communication channels. Such a link is called *interconnection* or shortly *connection*.

In addition to the current sections of this chapter, a section with the title "Choosing a Network Topology: experimental considerations" is scheduled to be included in the first supplement of this handbook.

2 Topology

A *neural network topology* represents the way in which neurons are connected to form a network. In other words, the neural network topology can be seen as the relationship between the neurons by means of their connections. The topology of a neural network plays a fundamental role in its functionality and performance, as illustrated throughout the handbook.

The generic terms 'structure' and 'architecture' are used as synonyms for network topology. However, caution should be taken when using these terms since their meaning is not well defined as they are also often used in contexts where they encompass more than the neural network topology alone or refer to something different altogether. They are for example often used in the context of hardware implementations ('computer architectures') or their meaning includes, besides the network topology, also the learning rule (see for example [Zurada-92]).

More precisely, the topology of a neural network consists of its *frame* or *framework* of neurons, together with its *interconnection structure* or *connectivity*:

$$\text{neural network topology} \begin{cases} \text{neural framework} \\ \text{interconnection structure} \end{cases}$$

The next two subsections are devoted to these two constituents respectively.

2.1 Neural Framework

Most neural networks, including many biological ones, have a layered topology. There are a few exceptions where the network is not explicitly layered, but those can usually be interpreted as having a layered topology, as for example in some associative memory networks ??, which can be seen as a one layer[1] neural network where all neurons function both as input and output units.

At the framework level, neurons are considered as abstract entities, thereby not considering possible differences between them. The framework of a neural network can therefore be described by the number of neuron layers, denoted by L, and the number of neurons in each of the layers, denoted by N_l, where l is the index indicating the layer number:

$$\text{neural framework} \begin{cases} \text{number of neuron layers} & : L \\ \text{number of neurons per layer} & : N_l \text{ where } 1 \leq l \leq L \end{cases}$$

The number of neurons in a layer (N_l) is also called the *layer size*.

The following neuron types can be distinguished:

- *Input neuron* A neuron that receives external inputs from outside the network.
- *Output neuron* A neuron that produces some of the outputs of the network.
- *Hidden neuron* A neuron that has no direct interaction with the 'outside world'; only with other neurons within the network.

Similar terminology is used at the layer level for *multilayer neural networks*:

- *Input layer* A layer consisting of input neurons.
- *Hidden layer* A layer consisting of hidden neurons.
- *Output layer* A layer consisting of output neurons.

In multilayer and most other neural networks the neuron layers are ordered and can be numbered: the input layer having index one, the first hidden layer index two, the second hidden layer index three, and so forth until the output layer, which is given the highest index L, equal to the total number of layers in the network. The number of neurons in the input layer can thus be denoted as N_1, the number of neurons in the first hidden layer as N_2 and of the second hidden layer as N_3, etc., until the output layer, whose size would be N_L. In figure 2 a four layer neural network topology is shown, together with the layer sizes.

[1] See section 8.1 for an explanation of why layers are considered to be neuron layers here.

Layer name	l	N_l
output layer	$4 = L$	$N_4 = N_L = 1$
second hidden layer	3	$N_3 = 2$
first hidden layer	2	$N_2 = 4$
input layer	1	$N_1 = 2$

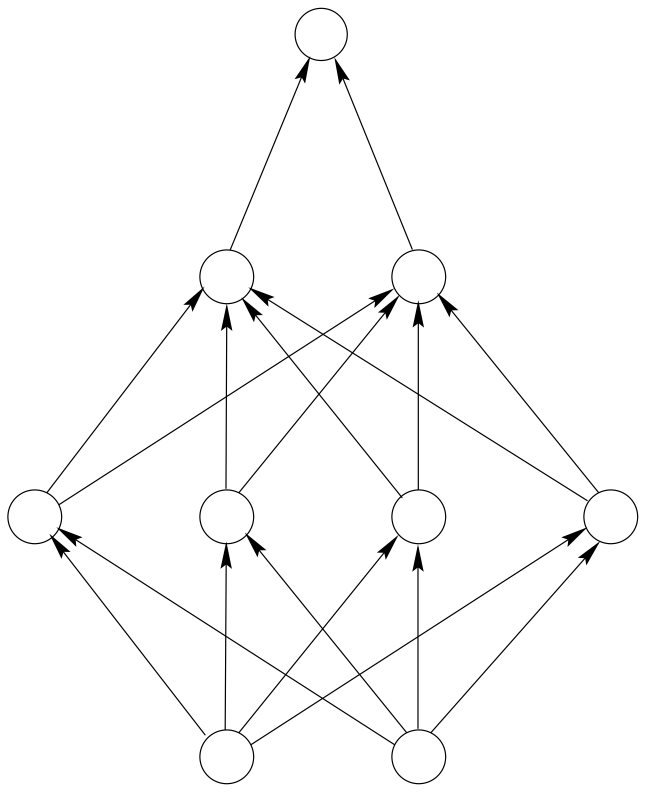

Figure 2: A fully interlayer connected topology with four layers.

Combining all layer sizes yields

$$N = \sum_{l=1}^{L} N_l, \tag{1}$$

where N is the total number of neurons in the network. Besides being more clear, the indexed notation for layer sizes is preferred since the number of layers in neural networks varies from one model to another and there are even some models that adapt their topology dynamically during the training process, thereby varying the number of layers (see section 7). Also, if one assigns a different variable to each layer (for example l, m, n, ...), one soon runs out of variables and into notational conflicts; this is especially the case for generic descriptions of multilayer neural networks and *deep networks*, which are networks with many layers.

In some neural networks, neurons are grouped together, like in layered topologies, but there is no well-defined way to order these groups. The groups of neurons in networks without an ordered structure are called '*clusters*', '*slabs*', or '*assemblies*', which are therefore generic terms which include the layer concept as a special case.

The neurons within a layer, or cluster, are usually not ordered; all neurons being equally important. However, the neurons within a cluster are sometimes numbered for convenience to be able to uniquely address them, as for example in computer simulations. Layers are likewise shapeless and can be represented in various ways. Exceptions are the input and output layers, which are special since the application constraints can suggest a specific shape, which can be one-, two-, or higher dimensional. Note however, that this structural shape is usually only present in pictorial representations of the neural network, since the individual neurons are still equally important and 'unaware' of each other's presence with respect to relative orientation. An exception could be an application specific partial connectivity where only certain neurons are connected to each other, thereby embedding positional information, like the feature detectors in [LeCun-89].

Likewise, there is also no fixed way of representing neural networks in pictorial form. Neural networks are most often drawn bottom up, with the input layer at the bottom and the output layer at the top, as in figure 2. Besides this, a left to right representation is also used, especially for optical neural networks ?? since the direction of the passing light in optical diagrams is by default assumed to be from left to right. Besides these, other pictorial orientations are also conceivable. This representational flexibility is also present in graph theory (see section 8.4).

2.2 Interconnection Structure

The interconnection structure of a neural network determines the way in which the neurons are linked. Based on a layered structure, several different kinds of connections can be distinguished (see figure 3 for an illustration):

- *Interlayer connection* Connects neurons in adjacent layers whose indices differ by one.
- *Intralayer connection* This is a connection between neurons in the same layer.
- *Selfconnection* This is a connection that connects a neuron to itself. It is a special kind of intralayer connection.
- *Supralayer connection* This is a connection between neurons that are in distinct layers that are not adjacent; in other words these connections 'cross' or 'jump' at least one hidden layer.

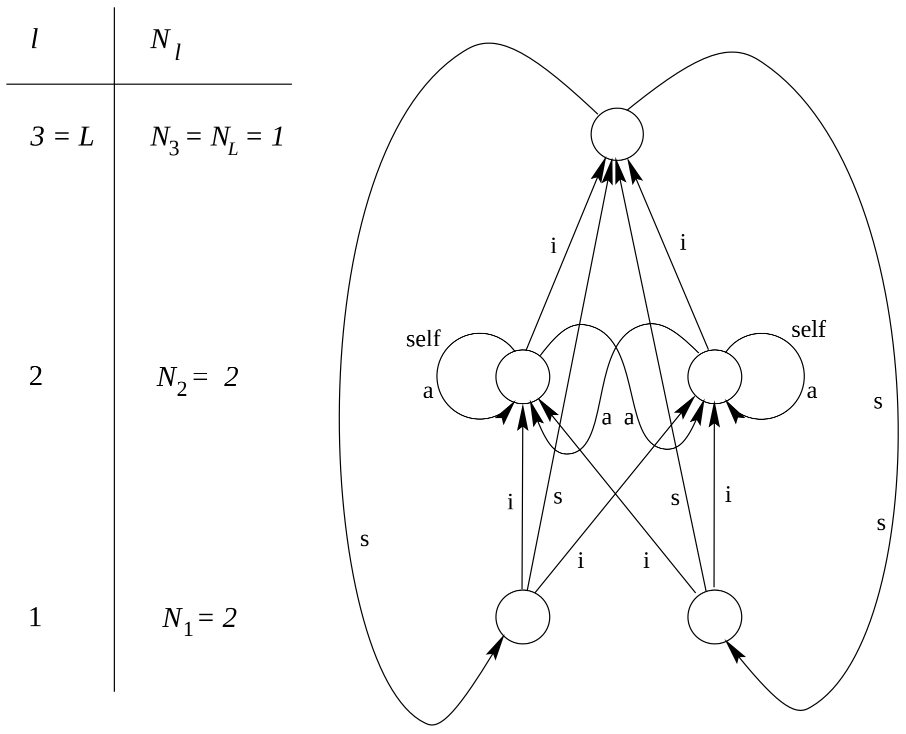

Figure 3: A three layer neural network topology with six interlayer connections (i), four supralayer connections (s) between the input and output layer, and four intralayer connections (a) including two selfconnections (self) in the hidden layer.

With each connection an *(interconnection) strength* or *weight* is associated which is a weighting factor that reflects its importance. This weight is a scalar value (a number), which can be positive (*excitatory*) or negative (*inhibitory*). If a connection has a zero weight is it considered to be nonexistent at that point in time.

Note that the basic concept of layeredness is based on the presence of interlayer connections. In other words, every layered neural network has at least one interlayer connection between adjacent layers. If interlayer connections are absent between any two adjacent clusters in the network, a spatial reordering can be applied to the topology, after which certain connections become the interlayer connections of the transformed, layered, network.

3 Symmetry and Asymmetry

The information flow through a connection can be symmetric or asymmetric. Before elaborating on this, it should be stated that 'information transfer' or 'flow', in the following discussion, refers to the *forward propagation*, where network outputs are produced in reaction to external inputs or stimuli given to the neural network. This in contrast to the information used to update the network parameters as determined by the neural network *learning rule* ??={hyperlink:B3: NN training}.

A connection in a neural network is either *unidirectional* when it is only used for information transfer in one direction at all times, or *multidirectional*[2] where it can be used in more than one direction. A multidirectional connection can either have one weight value that is used for information flow in all directions, which is the symmetric case (see figure 4), or separate weight values for information flow in specific directions, which is the asymmetric case (see figure 5).

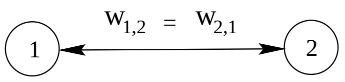

Figure 4: A symmetric connection between two neurons.

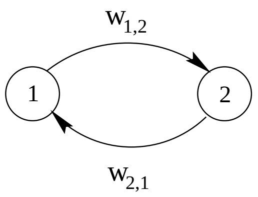

Figure 5: Two asymmetric connections between two neurons.

Hence, a *symmetric connection* is a multidirectional connection which has one weight value associated with it that is the same when used in any of the possible directions. All other connections are *asymmetric connections*, which can be either unidirectional connections (see figure 4) or multidirectional connections with more than one weight value per connection (see figure 5). Note that a multidirectional connection can be represented by a set of unidirectional connections, which is closer to biological reality where synapses are also unidirectional. In a unidirectional connection the information flows from its *source neuron* to its *sink neuron* (see figure 6).

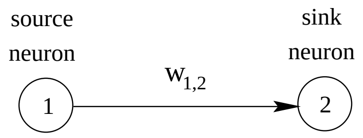

Figure 6: A unidirectional connection between a source and a sink neuron.

The definitions regarding symmetry can be extended to the network level: A *symmetric neural network* is a network with only symmetric connections, whereas an *asymmetric neural network* has at least one asymmetric connection. Most neural networks are asymmetric, having a unidirectional information flow or a multidirectional one with distinct weight values.

An important class of neural networks are the so called *feedforward neural networks* with unidirectional information flow from input to output layer. The name 'feedforward' is somewhat confusing since the best known algorithm for training a feedforward neural network is the backpropagation learning rule (see section ??), whose name indicates the backward propagation of (error gradient) information from the output layer, via the hidden layers back to the input layer, which is used to update the network parameters. The opposite of feedforward is 'feedback'; a term used for those networks that contain loops where information is fed back to neurons in previous layers. This terminology is not recommended since it is most often used for networks which have unidirectional supralayer connections from the output to the input layer, thereby excluding all other possible topologies with loops from the definition. Preferred is the term *recurrent neural network* for networks that contain at least one loop. Some common examples of recurrent neural networks are symmetric neural networks with bidirectional information flow, networks with selfconnections, and networks with unidirectional connections from output back to input neurons.

[2]The term 'multidirectional' is used here instead of bidirectional to include the case of high order connections (see section 4).

4 High Order Topologies

Most neural networks have only *first order connections* which link one source neuron to one sink neuron. However, it is also possible to connect more than two neurons by a *high order connection*[3] (see figure 7).

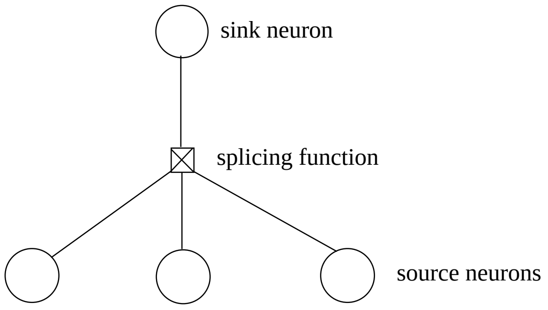

Figure 7: A third order connection.

High order connections are typically asymmetric, linking a set of source neurons to a sink neuron. The *connection order* (ω) is defined as the cardinality of the set of its source neurons, which is the number of elements in that set. As an example, figure 7 shows a third order connection. The information produced by the source neurons is combined by a *splicing function* which has ω inputs and one output. The most commonly used splicing function for high order neural networks is multiplication, where the connection outputs the product of the values produced by its source neurons. The set of source neurons of a high order connection are usually all located in one layer. The connectivity definitions of section 2.2 apply therefore also to high order connections.

The concept of higher orders can also be extended to the network level. A *high order neural network* has at least one high order connection and the *neural network order* (Ω) is determined by the highest order connection in the network:

$$\Omega = \max_{w} \omega_w, \tag{2}$$

where w ranges over all weights in the network.

Having high order connections gives the network the ability to extract higher order information from the input data set, which is a powerful feature.

Layered high order neural networks with multiplication as splicing function are also called *Sigma-Pi ($\Sigma\Pi$) neural networks*, since a summation (Σ) of products (Π) is used in the forward propagation:

$$a_j = \sum_{\{s_j\}} w_{\{s_j\}j} \prod_{i \in \{s_j\}} a_i, \tag{3}$$

where a_j is the activation value of the sink neuron, $\{s_j\}$ is the set of source neurons, $w_{\{s_i\}j}$ the associated weight, and a_i the activation values of the source neurons. The layer indices are omitted in this formula for notational simplicity. In section 8.2 notational issues concerning weights are discussed. For more information on Sigma-Pi neural networks, see [Rumelhart-86], which is based on [Williams-83].

The history of high order neural networks includes [Poggio-75] where the term 'high order' is used, and [Feldman-82] where multiplication is used as splicing function and the connections are named *conjunctive connections*. An important and fundamental contribution to the area of high order neural networks, which has given rise to a wider dissemination of them, is the work by Lee *et al.* [Lee-86].

For completeness *functional-link networks* [Pao-89] and *product unit neural networks* [Durbin-89] are mentioned here since they can be considered as special cases of high order neural networks. In these types of networks there is no combining of information from several source neurons taking place, but incoming information from a single source is transformed by means of a nonlinear splicing function.

5 Fully Connected Topologies

The simplest topologies are the fully connected ones, where all possible connections are present. However, depending on the neural framework and learning rule, the term *fully connected neural network* is used for several different interconnection schemes, and it is therefore important to distinguish between these.

The most commonly used topology is the *fully interlayer connected* one, where all possible interlayer connections are present but no intra- or supralayer ones. This is the default interconnectivity scheme for most non-recurrent multilayer neural networks.

[3]The term *higher order* is sometimes used in stead of 'high order'.

A truly fully connected or *plenary neural network* has all possible inter-, supra-, and intralayer connections including selfconnections. However, only a few neural networks have a plenary topology. A slightly more popular 'fully connected' topology is a plenary neural network without selfconnections, as used for example for some associative memories (??).

5.1 Connection Counting

In order to compare different neural network topologies, and more specific their complexities, it is useful to know how many connections a certain topology comprises. The connection counting is based on 'fully connected' topologies since they are the most commonly used and since they enable a fair and yet simple comparison. Fully interlayer connected topologies are considered as well as the various combinations of interlayer connections together with intra- and supralayer connections (see section 2.2); and 'fully connected' means here that all possible connections of each of those kinds are present in the topology. Before starting the counting of the connections, a few related issues need to be discussed and defined.

The total number of weights in a network can be denoted by W. For most neural networks this number is equal to the number of connections, since one weight is associated with one connection. In neural networks with *weight sharing* [Rumelhart-86], where a group of connections shares the same weight, the number of weights can be smaller than the number of connections. However, even in this case it is common practice to assign a separate weight to each connection and to update shared weights together and in an identical way. Given this, the number of connections is again equal to the number of weights and the same notation (W) can be used for both.

When counting the number of weights, it has to be decided whether to also count the neuron biases. The bias of a neuron, which determines its threshold level, can also be regarded as a special weight and its value is often modified in the same way as normal weights. This can be explained in the following way. The weighted sum of inputs to a neuron n, which has $W_{l,n}$ input providing connections, can be denoted as

$$\sum_{i=1}^{W_{l,n}} w_{i,n}\, a_i - \Theta_n = \sum_{i=1}^{W_{l,n}} w_{i,n}\, a_i + \Theta_n(-1), \tag{4}$$

where a_i the activation value of the neuron providing the i-th input, and $w_{i,n}$ is the weight between that neuron providing the i-th input to neuron n and neuron n itself[4]. Renaming Θ_n as $w_{0,n}$ and assuming a_0 to be a virtual activation with a constant value of -1, equation 4 becomes equal to:

$$\sum_{i=0}^{W_n} w_{i,n}\, a_i \tag{5}$$

Hence, the bias of a neuron can be seen as the weight of a virtual connection that receives its input from a virtual or dummy neuron that has a constant activation value of -1. In this section biases are not counted as weights. They can be included in the connection counting by initializing the appropriate summation indices with zero instead of one.

For networks where intralayer connections are present, two cases need to be distinguished: with and without selfconnections. Both cases can be conveniently combined in one formula by using the $\pm$-symbol, as utilized in the following section. If selfconnections are present, the addition has to be used, else the subtraction has to be used.

The maximum number of connections in asymmetric neural networks is twice that of their symmetric counterparts; except for selfconnections, which are intrinsically directed. Asymmetric topologies are therefore not elaborated upon in this context. The most common neural networks have symmetric first order topologies, which will be discussed first, followed by symmetric high order ones.

5.1.1 Counting Symmetric First Order Connections

The simplest and most widely used topologies have interlayer connections only. The total number of possible interlayer connections can be obtained by multiplying the layer sizes of each pair of adjacent layers and summing these over the whole network:

$$W = \sum_{l=1}^{L-1} W_l = \sum_{l=1}^{L-1} N_l\, N_{l+1}, \tag{6}$$

where W_l stands for the number of connections between layer l and $l+1$.

[4] See section 8.2 for a discussion on notational issues concerning weights.

When intralayer connections are also present, a number equal to the number of possible connections within a layer ($\frac{N_l}{2}(N_l \pm 1)$) has to be added for each layer in the network, and the total becomes:

$$\sum_{l=1}^{L} \frac{N_l}{2}(N_l \pm 1) + \sum_{l=1}^{L-1} N_l\, N_{l+1} \;=\; \frac{(N_L)^2 \pm N}{2} + \sum_{l=1}^{L-1} N_l \left(N_l + \frac{N_{l+1}}{2} \right). \tag{7}$$

The number of connections in networks with both interlayer and supralayer connections can be calculated by summing over all the layer sizes, multiplied by the sizes of all the layers of a higher index:

$$\sum_{l=1}^{L-1} N_l \sum_{m=l}^{L-1} N_{m+1} \;=\; \sum_{m=1}^{L-1} N_{m+1} \sum_{l=1}^{m} N_l. \tag{8}$$

Plenary neural networks have all possible connections and are equivalent to a fully connected undirected graph with N nodes (see section 8.4), which has

$$\frac{N}{2}(N \pm 1) \tag{9}$$

connections.

In summary, the number of connections in (fully connected) first order topologies is quadratic in the number of neurons:

$$W = O(N^2), \tag{10}$$

where $O()$ is the 'order' notation as used in complexity theory (see for example [Aho-74]).

5.1.2 Counting High Order Connections

In this section the counting of connections is extended to high order topologies. In order to focus the high order connection counting on the most common case, all the source neurons of a high order connection are assumed here to share the same layer and the possibility of having multiple instances of the same source neuron providing input to one high order connection is excluded.

It is illustrative to first examine the case of one single sink neuron in a high order network. The total number of possible connections of order ω that can provide information for one specific sink neuron is equal to the number of possibilities to combine the corresponding source neurons. This number is equal to

$$\left(\begin{array}{c} n \\ \omega \end{array} \right) := \frac{n!}{\omega!(n-\omega)!}, \tag{11}$$

where n is the number of potential source neurons. Note that ω can be maximally n.

Adding up these numbers over all possible orders, the maximum number of connections associated with a high order neuron[5] then becomes

$$\sum_{i=1}^{\Omega} \left(\begin{array}{c} n \\ i \end{array} \right). \tag{12}$$

Since Ω is bounded by n, the total number of high order connections is bounded by

$$\sum_{i=1}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) = 2^n - 1. \tag{13}$$

The virtual bias-connection of the neuron can be added to this sum to obtain the crisp maximum of 2^n.

To obtain the connectivity count of a high order topology, these high order neurons need to be combined into a network. Given the scope of this handbook, only the most prevalent case, that of asymmetric[6] fully interlayer connected high order networks is presented here. For a more elaborate treatment of this subject the reader is referred to [Fiesler-96.4], which also contains a comparison between the various topologies based on these connection counts.

The number of connections in a fully interlayer connected neural network of order Ω is:

$$\sum_{l=1}^{L-1} N_{l+1} \sum_{i=1}^{\Omega} \left(\begin{array}{c} N_l \\ i \end{array} \right). \tag{14}$$

In general, the number of connections in (fully connected) high order topologies is exponential in the number of neurons:

$$W = O(2^N). \tag{15}$$

[5]Note that the concept of 'order' can be seen from the connection point of view as well as from the neuron point of view.

[6]High order connections are usually unidirectional and counting multidirectional high order connections is complicated since the set of source neurons can no longer be assumed to share the same layer.

6 Partially Connected Topologies

Even though most neural network topologies are fully connected according to any of the definitions given in section 5, this choice is usually an arbitrary one and based on simplicity. Partially connected topologies offer an interesting alternative with a reduced degree of redundancy and hence a potential for increased efficiency. As shown in sections 5.1.1 and 5.1.2, the number of connections in fully connected neural networks is quadratic in the number of neurons for first order networks and exponential for high order networks. Although it is outside the scope of this chapter to discuss the amount of redundancy desired in neural networks, one can imagine that so many connections are in many cases an overkill with a serious overhead in training and using the network. On the other hand, partial connectedness brings along the difficult question of which connections to use and which not. Before giving an overview of the different strategies followed in creating partially connected topologies, a number of metrics are presented, providing a base for studying them.

6.1 Connectivity Metrics

Some basic neural network connectivity metrics are presented in this section. They can be used for the analysis and comparison of partially connected topologies, but are also applicable to the various kinds of fully connected topologies discussed in section 5.

The *degree* of a neuron is equal to the number of connections linked to it. More specifically, the degree of a neuron can be subdivided into an *in-degree* (d^{IN}) or *fan-in*, which is the number of connections that can provide information for the neuron, and an *out-degree* (d^{OUT}) or *fan-out*, which is the number of connections that can receive information from the neuron. It therefore holds that

$$d_n = d_n^{IN} + d_n^{OUT}, \tag{16}$$

where d_n is the degree of neuron n. For the network as a whole, the *average degree* ($\overline{d}$) can be defined as

$$\overline{d} = \frac{\sum_{l=1}^{L} \sum_{i=1}^{N_l} (d_{l,i}^{IN} + d_{l,i}^{OUT})}{N}, \tag{17}$$

where $d_{l,i}$ denotes the degree of neuron i in layer l.
Another useful metric is the *connectivity density* of a topology, which is defined as

$$\frac{W}{W_{\max}}, \tag{18}$$

where W is the number of connections in the network and $W_{\max}$ the total number of possible connections for that interconnection scheme, which are given in sections 5.1.1 and 5.1.2.

The last metric given here is the *connectivity level*, which provides a ratio of the number of connections with respect to the number of neurons in the network:

$$\frac{W}{N}. \tag{19}$$

6.2 A Classification of Partially Connected Neural Networks

As mentioned earlier, choosing a suitable partially connected topology is not a trivial task. This task is most difficult if one strives at finding a scheme for choosing such a topology a priori, that is, independent of the application. Most approaches leading to partially connected topologies are therefore assuming a number of constraints, which can aid in the topology choice. Based on this, the methods for constructing partially connected networks can be classified as follows:

- **Methods based on theoretical and experimental studies.** These methods usually assume a fixed, possibly random, connectivity distribution with either a constant degree or connectivity level. The created networks are typically used for theoretical studies to determine fundamental aspects of these networks, as for example their storage capacity.

- **Methods derived from biological neural networks.** The goal of these methods is to mimic biological neural networks as well as possible, or at least to use certain criteria from biology as constraints to aid the network building.

- **Application dependent methods.** This is an important class of methods where the choice of topology is directly based on information obtained from a given application domain.

- **Methods based on modularity.** Modular neural networks, which are discussed in a later section, are a special kind of partially connected neural networks that can be seen as a subclass of the application dependent models. They consist of a set of modules, which can each be either fully or partially connected internally. The modules themselves are typically sparsely connected to each other, again often based on application dependent knowledge. See also sections 7 and ??.
- **Methods developed for hardware implementation.** These methods are based on constraints that arise from hardware limitations in analog or digital electronic, optical, or other hardware implementations. An important subclass are the *locally connected* neural networks, like *cellular neural networks* (see chapter ??), that minimize the amount of wiring needed for the network, which is of fundamental importance for electronic implementations.
- **Ontogenic methods.** An important class of methods, where the topology is dynamically adapted during the training process by adding and/or deleting connections and/or neurons, are the *ontogenic methods*. The ontogenic methods that include the removal and/or addition of individual connections provide an automatic way to create partially connected neural networks. The various kinds of ontogenic neural networks are discussed in chapters ?? and ??.

An extensive review of partially connected neural networks, based on this classification, can be found in [Elizondo-96]. A short summary this work, restricted to non-ontogenic methods, is [Elizondo-95].

Besides these purely neural network based methods, other artificial intelligence techniques, like genetic programming and inductive knowledge, have been used to aid the construction of partially connected networks.

For completeness, a technique that does not necessarily reduce the number of connections but reduces the number of modifiable parameters by reducing the number of weights needs to be mentioned here, which is weight sharing (see also section 5.1). Using this technique, groups of connections are assigned only one updatable weight. These groups of connections can for example act as feature detectors in pattern recognition applications.

7 Special Topologies

Besides the common layered topologies, which are usually at least fully interlayer connected, there exists a variety of other topologies that are not necessarily layered, or at least not homogeneously layered. In this section a number of these are discussed.

Modular neural networks are composed of a set of smaller subnetworks (the modules), each performing a subtask of the complete problem. The topology design of modular neural networks is typically based on knowledge obtained from a specific application or application domain. Based on this knowledge, the problem is split up into subproblems, each assigned to a neural module. These individual modules do not have to belong to the same category and their topologies can therefore differ considerably. The global interconnectivity of the modular network, that links the modules, is often irregular as it is usually tuned to the application. The overall topology of modular neural networks is therefore often irregular and without a uniform layered structure. See section ?? for a detailed discourse on modular neural networks.

Somewhat related to modular neural networks are *composite neural networks*. A composite neural network consists of a concatenation of two or more neural network models, each with its associated topology, thereby forming a new neural network model. A layered structure can therefore be observed at the component level, since they are stacked, but the internal topologies of the components themselves can differ from each other, yielding an inhomogeneous global topology. Examples of composite neural networks can be found in sections ?? and ??. Composite neural networks are often called *hybrid neural networks*, a context dependent term that is even more popular for describing combinations of neural networks with other artificial intelligence techniques like expert systems, evolutionary systems etc. In this handbook, the term 'hybrid neural network' is therefore reserved for these systems; see part ??.

Another kind of topology that is sometimes used in the context of neural computation is the *tree*, which refers to the graph theoretical definition[7] of a connected acyclic graph. The typical tree topology used is a *rooted* one, where connections branch off from one point or a set of points. These points are usually the output neurons of the network. Tree based topologies are usually deep and sparse, and the neurons have a restricted fan-in and fan-out. If these networks are trees according to the definition, that is, without cross-connections between the branches of the tree, it can be argued whether they should be classified as neural networks or as *decision trees* [Kanal-79] [Breiman-84]. In this context it should be mentioned that it is in some cases possible to convert the tree based topology into a conventional layered neural network topology; see for example [Frean-90].

An important class of networks which can have a nonstandard topology are the ontogenic neural networks, as discussed in the previous section, where the topology can change over time during the training process.

[7]See section 8.4 for the relationship between graph theory and neural network topologies.

See chapters ?? and ?? for examples of ontogenic neural networks. Even though their topology is dynamic, it is usually homogeneous at each point in time during the training; this in contrast with modular neural networks, which are usually inhomogeneous.

One of the fundamental motivations behind ontogenic neural networks is to overcome the notorious problem of finding a suitable topology for solving a given problem. The ultimate goal is to find the optimal topology, which is usually the *minimal topology* that allows a successful solution of the problem. For this reason, but also for establishing a base for comparing the resulting topologies of different ontogenic training methods, it is important to define the minimal topology [Fiesler-93.3]:

Definition: A *minimal neural network topology* for a given problem is a topology with a minimal computational complexity that enables the problem to be solved adequately.

In practice, the topological complexity of neural networks can be estimated by the number of high complexity operations, like multiplications, to be performed during one recall phase. In case the splicing function is either the multiplication operation or a low complexity operation, the count can be restricted to the number of multiplications only. For first order networks, where the number of multiplications to be performed in the recall process is almost equal to the number of weighted connections, this can be further simplified to:

Definition: A *minimal first order neural network topology* for a given problem is a neural network topology with a minimal number of weighted connections that solves the problem adequately.

To illustrate the concept of minimal topology, the well-known *eXclusive OR* (XOR) problem can be used. The exclusive OR function has two Boolean inputs and one Boolean output which yields FALSE either when both inputs are TRUE or when both inputs are FALSE, and yields TRUE otherwise. This function is the simplest example of a *nonlinearly separable* problem. And, since nonlinearly separable problems can not be solved by first order perceptrons without hidden layers [Minsky-69], the minimal topology of a perceptron that can solve the XOR problem has either hidden layers or high order connections.

In the following three examples, binary (0, 1) inputs, outputs, and activation values are assumed, as well as a *hard-limiting threshold* or *Heaviside function* ($\mathcal{H}$) as activation function:

$$\mathcal{H}(x) = \begin{cases} 0 & \text{if} \quad x \leq \Theta \\ 1 & \text{if} \quad x > \Theta \end{cases} , \tag{20}$$

and the activation value of a neuron in layer $l+1$ is calculated by the following forward propagation formula:

$$a_{l+1_j} = \mathcal{H}\left(\sum_i W_{l_{i,j}} a_{l_i} \right) , \tag{21}$$

where a_{l_i} is the activation value of neuron i in layer l, and $W_{l_{i,j}}$ the weight of the connection between this neuron and neuron j in layer $l+1$; in accordance with the abbreviated notation of section 8.2.
Figure 8 shows the minimal topology of an interlayer connected first order neural network able to solve the XOR problem, and figure 9 the smallest first order solution which uses supralayer connections.

Figure 10 shows the smallest high order solution with two first order connections and one second order connection.

8 A Formal Framework

Even though artificial neural networks have been studied for several decades, a unifying formal theory is still missing. An important reason for this is the nonlinear nature of neural networks, which makes them difficult to study analytically, since most of our mathematical knowledge relates to linear mathematics. This lack of formalization is further illustrated by the upsurge in progress in neurocomputing during the same period when computers became popular and widespread, since they enable the study of neural networks by simulating their nonlinear dynamics. It is therefore important to strive for a formal theoretical framework that will aid the development of formal theories and analytical studies of neural networks. A first step towards this goal is the standardization of terminology, notations, and several higher level neural network concepts to enable smooth information dissemination within the neural network community, including users, that consists of people with a wide variety of backgrounds and interests. The IEEE Neural Network Council Standardization Committee is aiming at this goal (see Appendix ??). A further step towards this goal is a formal definition of a neural network that is broad enough to encompass virtually all existing neural network models, yet detailed enough to be useful. Such a topology based definition, supported by a consistent terminology and notation, can be found in [Fiesler-94.2]; other examples of formal definitions can be found in [Valiant-88], [Hong-88], [Farmer-90], and [Smith-92].

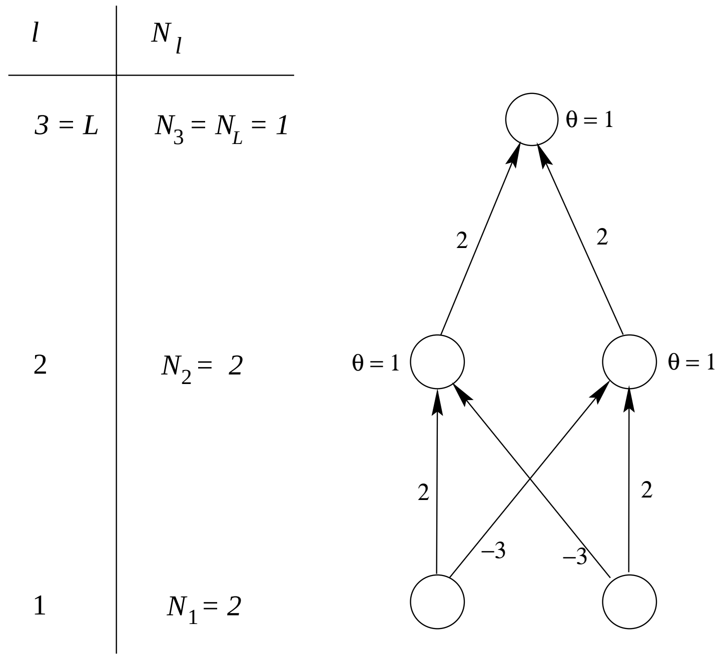

Figure 8: A first order neural network with a minimal interlayer connected topology that can solve the XOR problem. It has three layers and six interlayer connections.

A deep-rooted nomenclature issue, that of the definition of a layer, will be addressed in the next section. Further, in order to illustrate the concept of a consistent and mnemonic notation, the notational issue of weights, the most important neural network parameters, is discussed in the subsequent section, which is followed by a structured way to visualize and study weights and network connectivity. Lastly, the relationship between neural network topologies and graph theory is outlined, which offers a mathematical base for neural network formalization from the topology point of view.

8.1 Layer Counting

A fundamental terminology issue which gives rise to a lot of confusion throughout the neural network literature is that of the definition of a *layer* and, related to that, how to count layers in a network. The problem is rooted in the generic nature of the word 'layer', since it can refer to at least three network elements:

1. A layer of neurons
2. A layer of connections and their weights
3. A combination of a layer of neurons plus their connections and weights

Some of these interpretations need further explanation. The second meaning, that of the connections and associated weights is difficult to call a layer if there are other connections present besides interlayer connections only, for example like intralayer connections, which are inherently intertwined with a layer of neurons. Defining a layer as a set of connections plus weights is therefore very limited in scope and its use should be discouraged. For both the second and the third meaning, the relationship between the neurons and 'their' connections needs to be defined. In this context of layers, all incoming connections, that is, those that are capable of providing information to a layer of neurons, are usually the ones that are associated with that layer. Nevertheless, independent of which meaning is used, an important part of this terminology issue can be solved by simply defining what one means by a layer.

An early neural network in history with a layered topology was the Perceptron [Rosenblatt-58], which is sometimes called 'single layer Perceptron'. It has a layer of input units that duplicate and fan-out incoming information, and a layer of output units that perform the (nonlinear) weighted sum operation (see section ??). The name 'single layer Perceptron' reflects the third meaning of the word 'layer' as given above, and is based on not counting the input layer as a layer, which is explained below. Since the conception of the Perceptron, many other neural network models have been introduced. The topology of some of these models does not match with the layer concept given by the third interpretation. This is for example the case for networks which have intralayer connections in the input (neuronal) layer or where a certain amount of processing takes place in the input layer, like the Boltzmann Machine and related stochastic neural network models (see section ??)

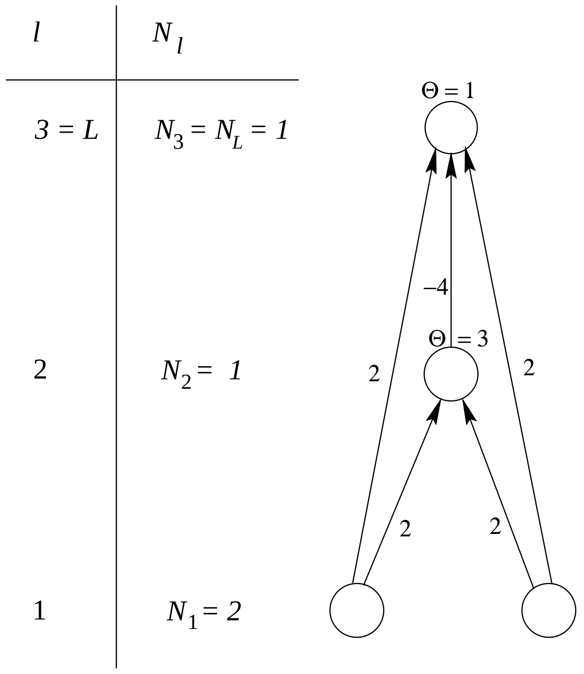

Figure 9: A first order neural network with a minimal topology that can solve the XOR problem. It has three layers, three interlayer connections, and two supralayer connections.

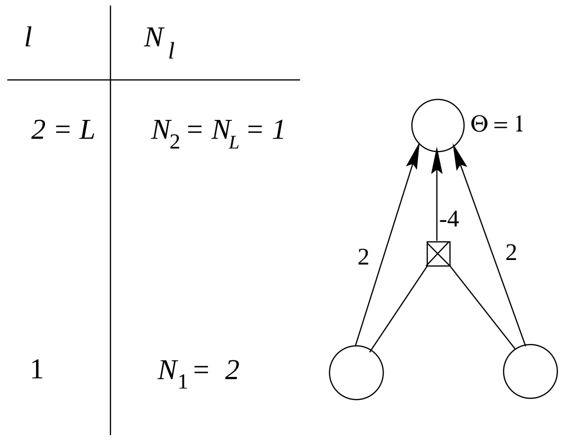

Figure 10: A high order neural network with a minimal topology that can solve the XOR problem. It has two layers, two first order connections, and one second order connection.

and like recurrent neural networks that feed information from the output layer back to the input layer (see section ??).

Currently, the most popular neural network models belong to the family of multilayer neural networks. The terminology associated with these models includes the terms 'input layer', together with 'hidden layer' and 'output layer' (see sections 2.1 and Backpropagation-Networks), which corresponds to the first interpretation of the word 'layer' as a layer of neurons.

The issue of defining a layer also gives rise to the problem of counting the number of layers, which is mainly caused by the dilemma of whether one should count the input layer as a layer. The argument against counting the input layer is that in many neural network models the input layer is used for duplicating and fanning out information and does not perform any further information processing. However, since there are neural network models where the input neurons are also processing units, as explained above, the best solution is to include the input layer in the counting. This policy has therefore been adopted by this handbook.

The layer counting problem manifests itself mainly when one wants to label or classify a neural network as having a certain number of layers. An easy way to circumvent the layer counting problem is therefore to count the number of hidden layers instead of the total number of layers. This approach avoids the issue of whether to count the input layer.

In can be concluded that the concept of a layer should be based on a layer of neurons. For a number of popular neural network models it would be possible to also include the incoming interlayer connections into the layer concept, but its use should be discouraged given its limited scope of validity. In general it is best to clearly define what is understood by a layer, and in order to avoid the layer counting problem one can count the number of hidden layers instead.

8.2 Weight Notation

To underline the importance and to illustrate the use of a consistent and mnemonic notation, the notation of the most fundamental and abundant neural network parameters, that of the weights, is discussed in this section.

A suitable and commonly used notation for a connection weight is the letter w, which is also mnemonic, using the first letter of the word 'weight'. Depending on the topology, there are several ways to uniquely address a specific weight in the network.

The best and most general way is to specify the position of both the source and the sink neuron that are linked by the connection associated with a weight, by specifying the layer and neuron indices of both: $w_{l_i m_j}$, where l and m are the indices of the source and sink layer respectively and i and j the neuron indices within these layers. This notation specifies a weight in a unique way for all the different kinds of first order connections as defined in section 2.2. This notation is used in figure 3.

For neural networks with only interlayer connections, the notation can be simplified if necessary. Since the difference between the layer indices (l and m) is always one for these networks, one of the two indices could be omitted: $w_{l_{ij}}$. In case this abbreviated notation is used, it is important to clearly specify which layer the index l represents; whether it represents the layer containing the source or the sink neuron.

A further notational simplification is possible for first order networks with one neuronal layer or networks without any cluster structure, where all neurons in the network are equally important. The weights in these networks can be simply addressed by w_{ij}, where the i and j indices point to the two neurons linked by the connection 'carrying' this weight.

High order connections require a more elaborate notation since they combine the information of several source neurons. Hence, the set of source neurons ($\{s_i\}$) needs to be included in the notation and the weight of a high order connection can be denoted as: $w_{\{s_i\}m_j}$. When desired, this notation can be abbreviated for certain kinds of networks, analogous to first order connections as described above.

Similarly to the weight notation, mnemonic notations for other network parameters are also recommended and used in this handbook (see ??).

8.3 Connectivity matrices

A compact way to represent the connections and/or weights in a neural network is by means of a *connectivity matrix*. For first order neural networks this is a two-dimensional array where each element represents a connection or its associated weight. A global connectivity matrix describes the complete network topology with all neuron indices enumerated along each of its two axis. Note that a symmetric neural network has a symmetric connectivity matrix and an asymmetric neural network an asymmetric one. Feedforward neural networks can be represented by a triangular matrix without diagonal elements. Table 1) shows an example for the fully interlayer connected topology of figure 2.

For layered networks, the order of the neuron indices should reflect the sequential order of the layers, starting with the input layer neurons on one end of the matrix and ending with the output neurons at the other end of the matrix. The matrix can be subdivided into blocks based on the layer boundaries; see table 1. In such a matrix, subdivided into blocks, the diagonal elements, which are the matrix elements with identical indices, represent the selfconnections and the diagonal blocks containing these diagonal elements contain the intralayer connections. The interlayer connections are found in the blocks that are horizontally or vertically adjacent to the diagonal blocks. All other blocks represent supralayer connections. Table 2 shows the global connectivity matrix for the network depicted in figure 3.

For layered neural networks with only interlayer connections, individual connectivity matrices can be constructed for each of the connection sets between adjacent layers.

The connectivity matrices for high order neural networks need to have a dimensionality of $\Omega + 1$, corresponding to the maximum number of source neurons (Ω) plus one sink neuron.

Based on the definitions of section 2.2, the *span* of a connection, measured in number of layers, can be defined as the difference between the indices of the layers in which the neurons that are linked by that connection are located. That is, the span of a connection which connects layer l with layer m is $|l - m|$. For example, interlayer connections have a span of one, intralayer connections a zero span, and supralayer connections a span of two or more. Different kinds of supralayer connections can be distinguished based on their span. The span of a connection can be easily visualized with the aid of a global connectivity matrix, since it is equal to the horizontal or vertical distance, in blocks, from the matrix element corresponding to that connection to the closest diagonal element of the connectivity matrix. The span of a high order connection,

	1,1	1,2	2,1	2,2	2,3	2,4	3,1	3,2	4,1
1,1			•	•	•	•			
1,2			•	•	•	•			
2,1							•	•	
2,2							•	•	
2,3							•	•	
2,4							•	•	
3,1									•
3,2									•
4,1									

Table 1: Connectivity matrix for the four layer fully interlayer connected neural network topology as depicted in figure 2. On the vertical axis the source neurons are listed by a tuple consisting of the layer number followed by the neuron number in that layer. On the horizonal axis the sink neurons are listed using the same notation. A '•' symbol marks the presence of a connection in the topology.

which is equal to the maximum difference between any of the indices of the layers it connects, is more difficult to visualize given the increased dimensionality of the connectivity matrix.

8.4 Neural Networks as Graphs

Graph theory[8] provides an excellent framework for studying and interpreting neural network topologies. A neural network topology is in principle a graph $(\mathcal{N}, \mathcal{W})$, where $\mathcal{N}$ is the set of neurons and $\mathcal{W}$ the set of connections, and when the network has a layered structure it becomes a *layered graph* [Fiesler-93.2]. More specifically, neural networks are directed layered graphs, specifying the direction of the information flow. In case the information between neurons can flow in more than one direction, there are two possibilities:

1. if distinct weight values are used for the information flow (between some neurons) in more than one direction, the topology remains a directed graph but with multiple connections between those neurons that can have a multidirectional information flow;

[8] See for example [Harary-69].

	1,1	1,2	2,1	2,2	3,1
1,1			•	•	•
1,2			•	•	•
2,1			•	•	•
2,2			•	•	•
3,1	•	•			

Table 2: Global connectivity matrix for the layered neural network topology with various kinds of connections as depicted in figure 3. The notation is the same as in table 1.

2. if the same weight value is used in all directions, the topology becomes symmetric (see section 3) and corresponds to the topology of an undirected graph.

Figure 1 shows a neural network topology without a layered structure, which is a directed graph. If all possible connections are present, as in a plenary neural network, its topology is equivalent to a *fully connected graph*.

References

[Aho-74] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. "*The Design and Analysis of Computer Algorithms.*". Computer Science and Information Processing. Addison-Wesley Publishing Company, Reading, Massachusetts, 1974.

[Breiman-84] L. Breiman, J. H. Friedman, R. A. Olsen, and C. J. Stone. "*Classification and Regression Trees.*". Wadsworth, Belmont, CA, 1984.

[Caudill-90] Maureen Caudill and Charles Butler. "*Naturally Intelligent Systems.*". A Bradford Book. The MIT Press, Cambridge, Massachusetts, 1990.

[Durbin-89] R. Durbin and David E. Rumelhart. "Product Units: A Computationally Powerful and Biologically Plausible Extension to Backpropagation Networks." *Neural Computation*, volume 1, pages 133–142, 1989.

[Elizondo-95] D. Elizondo, E. Fiesler, and J. Korczak. "Non-Ontogenic Sparse Neural Networks." In *Proceedings of the International Conference on Neural Networks*, (IEEE; Perth; 27 November – 1 December, 1995), (ISBN: 0-7803-2768-3), volume 1, pages 290–295. IEEE, Piscataway, NJ, 1995.

[Elizondo-96] D. Elizondo, E. Fiesler, and J. Korczak. "A Survey of Partially Connected Neural Networks.", . In preparation.

[Farmer-90] J. Doyne Farmer. "A Rosetta Stone For Connectionism." *Physica D*, volume 42, pages 153–187, 1990.

[Feldman-82] Jerome A. Feldman and D. H. Ballard. "Connectionist Models and Their Properties." *Cognitive Science*, volume 6, pages 205–254, 1982.

[Fiesler-93.2] E. Fiesler. "Layered Graphs with a Maximum Number of Edges." In Hervé Dedieu (editor), *Circuit Theory and Design 93; Proceedings of the 11th European Conference on Circuit Theory and Design*, (Davos, Switzerland; 30 August – 3 September, 1993), (ISBN: 0-444-81664-X), volume part I, pages 403–408. Elsevier, Amsterdam, The Netherlands, 1993.

[Fiesler-93.3] E. Fiesler. "Minimal and High Order Neural Network Topologies." In *Proceedings of the Fifth Workshop on Neural Networks: Academic/Industrial/NASA/Defense; An International Conference on Computational Intelligence: Neural Networks, Fuzzy Systems, Evolutionary Programming and Virtual Reality (WNN93/FNN93)*, (San Francisco, California; 7–10 November, 1993), (ISBN: 1-56555-059-5), number 2204 in SPIE Proceedings, pages 173–178. Simulation Councils, Inc. / The Society for Computer Simulation, San Diego, California, 1993.

[Fiesler-94.2] E. Fiesler. "Neural Network Classification and Formalization." *Computer Standards & Interfaces*, volume 16, number 3, pages 231–239, June 1994.

[Fiesler-96.4] E. Fiesler, H. J. Caulfield, A. Choudry, and J. P. Ryan. "Maximal Interconnection Topologies for Neural Networks.", . In preparation.

[Frean-90] Marcus Frean. "The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks." *Neural Computation*, volume 2, number 2, pages 198–209, Summer 1990.

[Harary-69] Frank Harary. "*Graph Theory.*". Addison-Wesley, 1969.

[Hong-88] Jiawei Hong. "On Connectionist Models." *Communications on Pure and Applied Mathematics*, volume 41, pages 1039–1050, 1988.

[Kanal-79] L. N. Kanal. "Problem-Solving Models and Search Strategies for Pattern Recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 1, pages 194–201, April 1979.

[LeCun-89] Yann Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition." *Neural Computation*, volume 1, number 4, pages 541–551, Winter 1989.

[Lee-86] Y. C. Lee, Gary Doolen, H. Chen, G. Sun, Tom Maxwell, H. Lee, and C. Lee Giles. "Machine Learning Using a Higher Order Correlation Network." In Doyne Farmer, Alan Lapedes, Norman Packard, and Burton Wendroff (editors), *Physica D: Nonlinear Phenomena*, (the Center for Nonlinear Studies; Los Alamos, New Mexico; May 20–24, 1985), volume 22, pages 276–306. Elsevier Science Publishers (North Holland), Amsterdam, 1986.

[Minsky-69] Marvin L. Minsky and Seymour A. Papert. "*Perceptrons.*". MIT Press, Cambridge, Massachusetts, 1969.

[Pao-89] Yoh-Han Pao. "*Adaptive Pattern Recognition and Neural Networks.*". Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1989.

[Poggio-75] T. Poggio. "On Optimal Nonlinear Associative Recall." *Biological Cybernetics*, volume 19, pages 201–209, 1975.

[Rosenblatt-58] Frank Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." *Psychological Review*, volume 65, pages 386–408, 1958.

[Rumelhart-86] David E. Rumelhart, James L. McClelland, and the PDP Research Group. "*Parallel Distributed Processing: Explorations in the Microstructure of Cognition.*" volume 1: Foundations. The MIT Press, Cambridge, Massachusetts, 1986.

[Smith-92] Leslie S. Smith. "A Framework for Neural Net Specification." *IEEE Transactions on Software Engineering*, volume 18, number 7, pages 601–612, July 1992.

[Valiant-88] L. G. Valiant. "Functionality in Neural Nets." In *Proceedings of the Seventh National Conference of the American Association for Artificial Intelligence (AAAI)-88*, (AAAI; St. Paul, Minnesota; August 21–26, 1988), (ISBN: ISBN: 0-929280-00-8), volume 2, pages 629–634. Morgan Kaufmann publishers, San Mateo, California, 1988.

[Williams-83] Ronald J. Williams. "Unit Activation Rules for Cognitive Network Models." ICS Technical Report 8303, Institute for Cognitive Science, University of California, San Diego (UCSD), La Jolla, California, 1983.

[Zurada-92] Jacek M. Zurada. "*Introduction to Artificial Neural Systems.*". West Publishing Company, St. Paul, Minneapolis, U.S.A., 1992.