PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F	A2	(11) International Publication Number: **WO 99/12085**
		(43) International Publication Date: 11 March 1999 (11.03.99)

(21) International Application Number: PCT/IL98/00430

(22) International Filing Date: 3 September 1998 (03.09.98)

(30) Priority Data:
60/057,818 4 September 1997 (04.09.97) US

(71) Applicant *(for all designated States except US)*: CAMELOT INFORMATION TECHNOLOGIES LTD. [IL/IL]; Matam, Advanced Technology Center, 31905 Haifa (IL).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: BAHRAV, Yuval [IL/IL]; Shlomtzion Street 24, 34406 Haifa (IL). GUR, Moshe [IL/IL]; Barazani Street 1, 69121 Tel Aviv (IL).

(74) Agents: FENSTER, Paul et al.; Fenster & Company Patent Attorneys, Ltd., P.O. Box 10256, 49002 Petach Tikva (IL).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: HETEROGENEOUS NEURAL NETWORKS

(57) Abstract

A method of constructing a neural network, comprising: providing a definition file, containing high-level specifications regarding a desired neural network; and compiling the definition file using software to generate a neural network which meets the specifications. Preferably, the specifications comprise a statistical specification of neuron interconnections.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

HETEROGENEOUS NEURAL NETWORKS

RELATED APPLICATIONS

The present invention claims the benefit under 119(e) of U.S. Provisional patent application number 60/057,818, of like title and filed on September 4, 1997, the disclosure of which is incorporated herein by reference.

FIELD OF THE INVENTION

This invention is related to the field of neural network design and, in particular, to automated generation of neural networks.

BACKGROUND OF THE INVENTION

Neural networks are a tool developed in the 1960's to deal with challenging pattern recognition tasks. In the 1960's it had become apparent that conventional computational solutions were not suitable in cases where "soft" decisions are required, in particular, identification and classification tasks where a system has to deal with noisy information or where very few of the inputs are expected to match a template. Since the brain is capable of solving such problems, neural networks are traditionally modeled after some aspects of the brain. However, as the brain is a very complex organ and not much is known about its inner workings, the modeling usually captures only a very few of the aspects of the brain. Many of the most common neural network in use today bear little or no resemblance to any biological structures and function in a manner which is mostly unrelated to any biological functioning.

A standard definition of neural networks, as found for example in "Neural Networks", by Simon Haykin, p. 2, Macmillan Collage publishing Company, New York (1994) is:

"A neural network is a massively distributed processor that has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two aspects:

1. Knowledge is acquired by the networks through a learning process.

2. Inter-neuron connections strengths known as synaptic weights are used to store the knowledge."

One advantage of neural networks, as implemented in the art, is their ability to learn and identify patterns even without a prior definition of the patterns which might exist in an input data set. Another advantage of neural networks is their ability to recognize patterns which are similar, but not identical, to a previously learned pattern. Neural networks are therefore used in systems that perform such tasks as hand-writing recognition, radar signature detection, object recognition and target identification. A most important advantage of neural

networks is that each neuron operates in parallel to all the other neurons, making each neural network a massively parallel architecture.

A typical neural network has the following structure: an input layer, at which input is presented to the neural network, an output layer at which outputs are produced by the neural network and a classification section between the input and output layers, in which a plurality of patterns are stored and which classification layer generates an output value set in response to an input value set. Classification, as used herein, includes pattern matching and optimization. Neural networks typically has a two stage life cycle: a first, training, stage in which the neural network learns a wide variety of input patterns and a second, active, stage in which the neural network does not learn any more but only generates output sets in response to input sets, according to data stored in the training stage. In some cases, the neural network is periodically retrained during its active use.

The learning which neural networks perform is a set of actions dedicated to reducing the error on an empirical parameter. In each neural network architecture this parameter and the actions taken are different according to the learning rules devised by the net's designer. The learning is manifested by changing the weights (strength) of the connections between the different processing units (neurons), in response to the presentation of an input pattern or a repetitive set of input patterns. Each class of network architecture includes a learning rule which states the algorithm by which the weight of a connection is changed in response to different parameters.

Most neural networks are constructed by taking a standard architecture and defining various parameters of the architecture. The definition of an architecture includes the interconnection between the neurons, the learning function and how an individual neuron acts upon its input. The number of neurons and the number of layers (in multi-layer networks) in a neural network are usually parameters of the network.

There are several common architectures for neural networks: Back Error Propagation, Hopfield (has only one layer), Self Organizing Map, Boltzman Machine, RBF nets and ART nets. These architectures are characterized by the response of an individual neuron upon its input, by the type of interconnection between individual components of the network and by the training process which is used. In all these architectures, each of the learned patterns is stored in the entire network. Usually, the patterns are stored by the configuration of connection strengths between individual neurons.

Neural networks comprise a plurality of individual "neurons". Each neuron has many inputs and a single output. However, the single output can be connected in parallel to the

inputs of many other neurons. In operation, a typical neuron weighs and sums the inputs and generates an output responsive to the summed inputs, i.e., it performs a transformation function. In some systems, a neuron generates an output only if the weighted sum is above a certain threshold. Messages between the neurons may be encoded by an amplitude or by a frequency. In most (or all) practical systems, the data is encoded using an amplitude value. Some small scale experimental implementations of frequency encoded neural networks have been built for research purposes. However, even in these cases, no general architecture for frequency encoded neural networks has been proposed. In addition, since these implementations are experimental, they do not have any interaction with the outside world. In the brain, the data is encoded using a frequency encoding.

Many neural networks operate by converging to a stable state, at which point the presented output is the "correct" output. However, in some cases, a neural network may get caught in a loop, whereby all the neurons fire together and the neural network jumps between states. This situation is called phase-locking and it is a major limitation on the ability of neural networks to operate in an unsupervised manner.

A typical neural network has a very small number of neurons (<100), is homogeneous in structure, is hand crafted for a particular task and is then taught the patterns of the task. However, even such hand crafting does not imply freedom of design. Rather, a designer chooses a pre-defined architecture and then sets various parameters such as the number of layers and the number of neurons in each layer. The designer is usually forced to design an algorithmic preprocessor to normalize the data, so the chosen architecture can recognize and perform successful learning on the data. Clearly, such construction methods are inefficient from an industrial point of view and require a high level of competence. The designer can, of course, develop his own unique architecture which is designed for the task at hand, however this is practical only in research setting, since designing such an architecture can take many years. In any case, when a designer builds an application specific neural network he modifies parameters of a general architecture and then the resulting neural network is trained to do the task.

B. J. Copeland, in "Artificial Intelligence", published by Blackwell publishers,. 238 Main street, Cambridge, Massachusetts 02142 USA, in 1993, asserts in chapter 10.5 that the brain has a very complex geometrical-connective structure, while neural networks do not have any structure. In particular, he notes that in the brain, neurons are more often connected to nearby neurons and less often connected to far neurons, thus creating groups in the brain.

It should be noted that multi-layer neural networks cannot be considered to have a task-related structure. Multi-layer neural networks have all the neurons in one layer connected to all the neurons in the next layer, but inside a layer, there is no connection between neurons. Thus, the interconnection structure is a uniform one, at least with respect to being independent of the specific function of the neural network. It is the general architecture which could be considered to have a structure, however, this structure is unrelated to the function performed by the network, except that a multi-layer structure is used.

John Kelly, in "Artificial Intelligence A Modern Myth", published by Ellis Horwood Limited, Market Cross House, Cooper Street, Chichester, West Sussex, PO191EB, England in 1993, asserts that not much is known about brain functions outside of the sensory systems, so that saying that an architecture [of a neural network] imitates the brain is pure conjecture. In addition, he notes that the wide variety of behaviors exhibited by individual neurons is almost completely ignored in neural network modeling.

SUMMARY OF THE INVENTION

It is an object of some embodiments of the present invention to provide a method of constructing large and complex neural networks.

It is another object of some embodiments of the present invention to provide a method of automatically generating a neural network.

Another object of some embodiments of the present invention is to provide a neural network which is useful for processing of data, beyond simple pattern recognition, preferably, without requiring any training of the neural network.

It is also an object of some aspect of the present invention to provide a non-homogeneous neural network and method for construction thereof.

Preferred embodiments of the present invention have various aspects, some of which are detailed below. It should however be appreciated that not every embodiment of the present invention has all the aspects described below. Rather, some preferred embodiments of the present invention may have only a few (or none) of these aspects.

A first aspect of the invention relates to the use of non-homogenous neural networks. Conventional neural networks comprises a plurality of similar neurons, which are interconnected in a set manner, typically, the individual connection strengths are random. Usually, each neuron is connected to all the other neurons, in its layer or in adjacent layers, depending on the type of network. Rather than being such a homogenous neural network, a neural network in accordance with a preferred embodiment of the invention comprises a plurality of types of neurons. Different types of neurons typically have different properties,

such as response threshold, output potential amplitude and duration, outgoing weight and connection morphology. Alternatively or additionally, different parts of the neural network have different connection morphologies between individual neurons.

A second aspect of the invention relates to the construction of non-homogenous neural networks. In accordance with a preferred embodiment of the invention, a neural network is automatically generated from a definition file, i.e., compiled. Preferably, the definition file includes a statistical description of connection probabilities between different neurons. Preferably, the distribution creates functional groups of neurons. Typically, neurons are more interconnected within each group and less interconnected between groups. Alternatively or additionally, the definition file includes a description of various types of neurons, which may be characterized, *inter alia*, by their excitation threshold, their transfer function and the number of connections thereto. Additionally, the definition file may include definitions for individual neurons, i.e. a functional group having only a single neuron. Preferably, the definitions are stored in using an object-oriented data structure, in which each type of neuron is a variant of previously defined types. Preferably, the definition file is created by a user manipulating a spatial representation of a neural network. In a preferred embodiment of the invention, the defined structure is related to the function which the network performs.

In a preferred embodiment of the invention, the neural network is described as a three-(or higher) dimensional body, which has neurons distributed evenly therethrough. This design methodology whereby the neural network is treated as a physical object, is unique to some preferred embodiments of the present invention.

A third aspect of the invention relates to the type of neural network which can be constructed using methods according to various embodiments of the present invention. Using various aspects of the present invention, it is possible to design and build a neural network which performs a particular function. Such a neural network is an analogue of a computer program, in as much as a programmer builds a description file, similar to writing a program and then complies it to yield an object which performs a desired function. Examples of such functions include scaling and rotating of input. In particular, such a neural network will not generally be required to undergo any amount of training at all. To the extent it is deemed necessary, such a neural network may include a section which can be trained and which performs any required pattern recognition function. Typically, such a section is small, however, it may include any portion of the neural network and may even encompass the entire network.

In addition, such a neural network can be integrated with executable code, such as compiled C++ code, whereby the neural network is treated as a subroutine or, alternatively, where the C++ code is used to perform a complex calculation. In one example, the neural network is compiled into an object which can accept an input pattern and produce an output file in any desired format. Preferably, the object includes algorithmic pre- and post- processing portions, which convert the input from an external format to an internal format and/or which convert the output from an internal format to an external format. The object may be executed from a command line prompt or from a program written in a high level language. In addition, such an object may be used in parallel by more than one program. It should be appreciated that such a neural network can include tens of thousands of neurons. For example, an IBM-PC compatible computer with 32MB of memory and a Pentium® 166 processor successfully executed a neural network with 50,000 neurons and approximately 500,000 connections. On such a system, a time tick (defined latter) takes approximately between 500-1500msec, depending on the level of excitation of the network.

It should be noted that a neural network in accordance with a preferred embodiment of the invention has (at least in part) a pre-designed, non-adaptable, functional architecture. Since the structure is not generally data specific it is more robust to noise and other changes and less limited in the range of data patterns it can process.

A fourth aspect of the present invention relates to the type of neural network communication scheme preferred for some aspects of the present invention. In a preferred embodiment of the invention, communication between neurons is encoded using frequency/pulse train encoding, whereby, a higher value is preferably encoded as a higher frequency. In a preferred embodiment of the invention, a random time delay is applied at the output of each neuron, in order to prevent phase locking. Thus, it is not possible for all the neurons to work in complete synchrony. In one preferred embodiment of the invention, the delay is constant for each neuron and may be determined when the neural network is constructed. Alternatively, the delay is randomly generated for each neuron, each time it generates an output. Another advantage of such a random delay is that it assists the neural network in escaping local minima. Alternatively or additionally, a refractory period is supplied after the neuron fires, thereby giving the neural network a non-linear response.

A fifth aspect of the present invention relates to a highly efficient method of simulating a neural network using a serial computer. As a result, a standard desktop computer can very quickly simulate a network having tens of thousands of neurons. More powerful computers may, of course, simulate even larger networks. As another result, embedded real-time neural

networks are also practical. In a preferred embodiment of the invention, an individual neuron is simulated as having at least one time-coded input buffer. Inputs to a neuron are added into the buffer, such that each element of the buffer corresponds to the sum of the signals which arrived at a particular time. Each input is multiplied, prior to being added to the buffer, by a time-decay weighting factor, such that earlier signals are given a higher weight. Preferably, the peak weight is not at the earliest time, but somewhat, later. Thus, the weighting function preferably, rises and then falls, with time. If the value in the buffer which corresponds to the current time is above a threshold, the neuron generates an output. Alternatively, other, user supplied, decision functions may be used. As can be appreciated, this type of design requires only addition and multiplication operations and is therefore easy to implement efficiently. In addition, such a design is easy to optimize and does not require many computational resources.

A sixth aspect of the invention relates to using a real time scale. As described above, each input to a neuron is time stamped. Preferably, the neural network operates in discrete time steps. Since time is directly modeled in some embodiments of the present invention, true Hebbian learning can be practiced, where a the weight between two neurons is increased if they fire at a correlated frequency and/or substantially at the same time. In some Hebbian learning schemes, weights are increased also if the two neurons are inhibited from firing at the same time. Alternatively or additionally, the connection weight between two neurons is decreased if their firing is not correlated. Preferably, a Hebbian network has two layers, an input and an output layer, where each neuron in the input layer is connected to a corresponding neuron in the output layer and each neuron in the output layer is connected to each neuron in the output layer. However, the input may be directly projected unto the learning layer. Preferably, all the connections weights are negative (inhibitory connections) at the onset of training. Alternatively or additionally, the Hebbian algorithm includes a restraining factor which limits connection weights are limited to a maximum value. Prior art neural networks do not utilize time modeling and, therefore, true Hebbian learning cannot be performed. In addition, a neural network in accordance with a preferred embodiments of the invention generates output even before the network stabilized. In a prior art neural network, such output cannot have any meaning.

A seventh aspect of the present invention relates to a new programming methodology for neural networks, which is made possible by the previously described aspects of the present invention. The heterogeneous structure of neural network in accordance with some aspects of the present invention, make it possible to design modular neural networks. In particular, such a neural network may includes modules copied from a library of neural networks. Further, a

neural network, in accordance with a preferred embodiment of the invention, may have a modular-hierarchical structure. For debugging purposes it is then possible to debug each module individually. Further, since different structures in the neural network have different functions, the functioning of such structures, and even of individual neurons may be analyzed to aid debugging. One result of the new design methodologies is that making a change in a neural network is a matter of minutes, not days or weeks. This is possible because only a single module of the neural network must be changed, the neural network does not usually need to be retrained (since usually most or all of it are non-trainable) and each module has an internal structure which may be intuited by a human operator. Thus, an iterative design process may be used for designing neural networks. Training where necessary is also preferably shortened by an order of magnitude, since most of the training time (in prior art neural networks) is usually expended on training the neural network to approximate the learned function. This approximation is automatically provided for in preferred embodiments of the present invention, since the neural network preferably has a structure which approximates performing the desired function.

In a preferred embodiment of the invention, various aspect of neural network design are automated. For example, there is no generally agreed upon method of determining a minimum required number of neurons, in a neural network, for a particular task. In a preferred embodiment of the invention, a user generates rules for a neural network, stipulates a maximum number of required neurons and provides example data sets. An automated program then generates a plurality of neural networks, each with a different number of neurons, and performs a search for the neural network with the smallest number of neurons which still meets the criteria.

An eighth aspect of the present invention relates to particular neural networks which perform data processing of a type not previously possible using a neural network, which data processing is of a type crucial for generalized pattern recognition tasks. A neural network in accordance with a preferred embodiment of the invention, can also handle data representation issues, such as pre-processing the angle of an input image. The problems of scale and rotation invariance have not previously been solvable using neural network techniques. In most, if not all, practical neural networks, data must be normalized (by an algorithmic program) for the NN to properly operate.

A ninth aspect of some preferred embodiments of the present invention relates to learning by varying parameters of a neural network, alternatively or additionally to synapse weight, for example synapse delay, neuron threshold, output shape and/or duration, refractory

period and/or other operational parameters of neurons. In a preferred embodiment of the invention, these parameters are modified as a function of a global state of the network, for example a back-propagation error function and/or as a function of local states, for example a level and/or pattern of sub-threshold activity. In a preferred embodiment of the invention, these learning methods are applied to a heterogeneous neural network. Alternatively or additionally, these learning methods are applied to other types of neural networks, for example, non-structured neural networks. Training techniques used may be any of those known in the art, for example back propagation.

There is therefore provided in accordance with a preferred embodiment of the invention a method of constructing a neural network, comprising:

providing a definition file, containing high-level specifications regarding a desired neural network; and

compiling the definition file using software to generate a neural network which meets the specifications.

Preferably, the specifications comprise a statistical specification of neuron interconnections. Preferably, the statistical specification comprises a specification of connections from outputs of neurons to inputs of other neurons. Alternatively or additionally, the statistical specification comprises a specification of connections to inputs of neurons from outputs of other neurons. Alternatively or additionally, the statistical specification comprises a spatial distribution.

In a preferred additional or alternative embodiment of the invention, the definition file comprises definitions of a plurality of neuron types. Preferably, the definition file comprises definitions of at least 5 neuron types. Preferably, the definition file comprises definitions of at least 15 neuron types. Preferably, the definition file comprises definitions of at least 50 neuron types. Alternatively or additionally, the definition of neuron types comprises a hierarchical definition of neuron types.

Alternatively or additionally, the definition file comprises a modular definition file, each module comprising a definition of a neural network module and the definition file defining at least one interconnection between individual modules. Alternatively or additionally, the definition file comprises a definition of parameters for a trainable and uniform neural-network architecture. Alternatively or additionally, the definition file comprises user-defined functions which perform portions of a neuron's activity.

In a preferred embodiment of the invention, the method comprises: testing said generated neural network; modifying said definition file; and repeating said compiling, said testing and said modifying until said generated neural network meets a predetermined criteria.

Alternatively or additionally, compiling comprises generating error messages responsive to mistakes in the definition file. Preferably, the errors are caused by a non-existence of a neural network meeting the specifications.

Alternatively or additionally, compiling comprises: generating an array of neurons; and assigning a particular type definition to selected one of the neurons.

Alternatively or additionally, compiling comprises: generating an array of neurons; and defining forward connections between a first plurality of neurons and a second plurality of neurons.

Alternatively or additionally, compiling comprises: generating an array of neurons; and

defining backward connections to a third plurality of neurons from a fourth plurality of neurons.

Alternatively or additionally, compiling comprises: generating an array of neurons; and defining direct connections between at least one neuron and at least a second neuron.

There is also provided in accordance with a preferred embodiment of the invention, a method of neural network design, comprising:

determining a function to be performed by a neural network;

designing a neural network architecture suitable for performing the function; and

generating a neural network having the designed architecture.

Preferably, designing an architecture comprises imposing a structure on a homogeneous neural network. Alternatively or additionally, designing an architecture comprises interconnecting at least two existing neural network modules. Alternatively or additionally, designing an architecture comprises defining functional groupings of neurons in the neural network.

There is also provided in accordance with a preferred embodiment of the invention, a method of neural network construction, comprising:

providing a first neural network;

providing a second neural network; and

interconnecting the first and second neural networks.

Preferably, the first neural network and the second neural network each comprise neurons having different characteristics. Alternatively or additionally, providing a second neural network, comprises selecting the second neural network from a library of neural

networks. Alternatively or additionally, providing a first neural network comprises debugging the first neural network prior to said interconnecting. Alternatively or additionally, providing a first neural network comprises training the first neural network prior to said interconnecting.

Alternatively or additionally, interconnecting the first and second neural networks comprises providing a third neural network which matches the inputs and outputs of the first and second neural networks.

In a preferred embodiment of the invention, the first and second neural networks each have a scale and comprising changing the scale of the second neural network to match the scale of the first neural network.

There is also provided in accordance with a preferred embodiment of the invention, a method of designing a neural network, comprising:

providing a initial neural network;

providing a range of allowed values for at least one parameter, which parameter defines allowed mutations in the initial neural network; and

searching an answer-space defined by the allowed range to find a neural network which is more optimal, in a predefined manner, than the initial neural network and which found neural network performs a predefined function.

Preferably, searching comprises:

automatically generating a new neural network; and

determining if the new neural network performs the predefined function.

Preferably, automatically generating a new neural network comprises modifying an existing neural network.

There is also provided in accordance with a preferred embodiment of the invention, a method of neural network construction, comprising:

defining a function to be performed by a neural network; and

constructing a neural network to perform the function, wherein constructing does not include training the neural network to perform the function.

There is also provided in accordance with a preferred embodiment of the invention, a method of neural network construction, comprising:

spatially defining a group of neurons of the neural network, which group comprise only a portion of the neurons in the neural network; and

setting a characteristic of the defined group.

Preferably, the characteristic comprises a spatial distribution function. Alternatively or additionally, the characteristic comprises a compiled high-level language function which

determines the output response of the neurons in the group. Alternatively or additionally, the characteristic comprises the probability of connection from the selected neurons to other neurons. Alternatively or additionally, the characteristic comprises the probability of connection from to selected neurons from other neurons. Alternatively or additionally, the characteristic comprises an integrator function of the neurons. Alternatively or additionally, the characteristic comprises a firing threshold of the neurons. Alternatively or additionally, the characteristic comprises a number of connections from said neurons. Alternatively or additionally, the characteristic comprises a delay distribution function of outputs from said neurons. Alternatively or additionally, the characteristic comprises a distribution of connection weights for said neurons.

In a preferred embodiment of the invention, the method includes constructing an integrated circuit which performs the function of said neural network. Preferably, said circuit comprises individual circuits for individual neurons of said neural network.

There is also provided in accordance with a preferred embodiment of the invention, a neural network constructed according to any of the above described methods.

There is also provided in accordance with a preferred embodiment of the invention, a computer-readable media having a computer program stored therein, wherein said program, when executed on a general purpose computer for which the program is adapted, causes the computer to simulate a neural network, constructed, as described above.

There is also provided in accordance with a preferred embodiment of the invention a method of testing a neural network having a plurality of input lines, comprising:

providing an image;

projecting the image unto the plurality of input lines; and

analyzing an output of the neural network. Preferably, the image comprises a sequence of images.

There is also provided in accordance with a preferred embodiment of the invention a method of testing a neural network, comprising:

providing a neural network having a plurality of individual neurons, a plurality of input lines and a plurality of output lines; and

tracing the output of at least one neuron, which neuron is not directly connected to an output line.

There is also provided in accordance with a preferred embodiment of the invention, a method of testing a neural network, comprising:

providing a neural network having a plurality of individual neurons, a plurality of input lines and a plurality of output lines; and

forcing an input to at least one neuron, which neuron is not directly connected to an input line.

There is also provided in accordance with a preferred embodiment of the invention, a method of testing a neural network, comprising:

replacing at least a portion of the neural network with a stub; and

providing an input data set to the neural network; and

analyzing an output from the neural network, which output is responsive to the input data set.

There is also provided in accordance with a preferred embodiment of the invention, a method of simulating a neuron which generates an output responsive to inputs the neuron receives from a plurality of input lines, comprising:

adding the inputs to at least one time-stamped buffer, responsive to the time at which the inputs are received by the neurons; and

generating an output responsive to the values in the buffer.

Preferably, generating a response comprises generating a response if the value of the buffer at the current time is above a threshold. Alternatively or additionally, generating a response comprises generating a response if a sum of values in the buffer from a range of times at and prior to the current time, is above a threshold. Alternatively or additionally, adding comprises weighting the inputs with a weight responsive to their reception time. Alternatively or additionally, generating an output comprises generating an output at a stochastic delay.

There is also provided in accordance with a preferred embodiment of the invention, an electro-mechanical device, comprising:

a user input panel;

a mechanical portion; and

a neural network which controls the mechanical portion responsive to user inputs from the user input panel, wherein the electro-mechanical device does not utilize an algorithmic circuit for controlling the mechanical portion.

Preferably, the neural network operates substantially in real-time.

There is also provided in accordance with a preferred embodiment of the invention, apparatus for generating a neural network, comprising:

storage for a definition file; and

a compiler which generates a neural network meeting the specifications in the definition file.

Preferably, the apparatus comprises a syntax analyzer which analyses the syntax of the definition file. Alternatively or alternatively, the apparatus comprises an error-message generator which generates error messages responsive to the analysis of said syntax analyzer

There is also provided in accordance with a preferred embodiment of the invention a neural network comprising:

a first, non-trainable, plurality of interconnected neurons; and

a second, trainable, plurality of interconnected neurons.

Preferably, the training of said second plurality of neural networks is controlled by the first plurality of neural networks. Alternatively or additionally, at least one of said second first plurality of neurons and said second plurality of neurons comprises a neuron having at least one trainable connection and at least one non-trainable connection.

There is also provided in accordance with a preferred embodiment of the invention, a neural network comprising a plurality of interconnected neurons, wherein said neurons do not perform a pattern matching function and are not trainable.

There is also provided in accordance with a preferred embodiment of the invention, a neural network comprising a plurality of interconnected neurons, wherein most of the neurons are grouped into a plurality of functional groups and wherein most of the connections of neurons in each functional group are with neurons in the same functional group.

There is also provided in accordance with a preferred embodiment of the invention, a neural network comprising a plurality of interconnected neurons which generate output signals, wherein the output signals are frequency encoded spike trains and wherein each neuron delays its output by a different amount.

Preferably, each neuron stochastically delays its output.

There is also provided in accordance with a preferred embodiment of the invention, a neural network, for controlling an industrial process comprising:

a plurality of interconnected neurons which generate output signals, wherein the output signals are frequency encoded spike trains;

a input device; and

an output device,

wherein said neural network generates controls said output device, responsive to said input device to achieve a desired effect on the industrial process.

There is also provided in accordance with a preferred embodiment of the invention, a neural network comprising:

at least one neuron of a first type;

at least one neuron of a second type; and

at least one neuron of a third type.

Preferably, said different types have different integrator functions. Alternatively or additionally, said different types have different numbers of connections therefrom.

There is also provided in accordance with a preferred embodiment of the invention, a method of training a neural network comprising:

providing a neural network having a design function and having a trainable portion; and

training the neural network, wherein said training is performed while the neural network performs its designated function.

There is also provided in accordance with a preferred embodiment of the invention, a method of training a neural network comprising:

providing a neural network having a design function and having a trainable portion; and

training the neural network, wherein said training is controlled by the neural network, without external intervention.

There is also provided in accordance with a preferred embodiment of the invention, a method of training a neural network comprising:

providing a neural network having a trainable portion comprising a plurality of neurons, each of said neurons having at least one non-synapse-weight parameter; and

training the neural network by modifying at least one of said at least one non-synapse-weight parameters of said neurons.

Preferably, said training comprises modifying at least one synapse-weight parameter of said neurons. Alternatively or additionally, said neural network comprises a heterogeneous neural network. Alternatively or additionally, said training is responsive to a global state of said neural network. Preferably, said global state comprises an error function of said network.

In a preferred embodiment of the invention, said training is responsive to a local state of said neural network. Preferably, said training is responsive to a state of a neuron being trained. Alternatively or additionally, said training is responsive to a state of a neuron which provides an input to a neuron being trained. Alternatively or additionally, said training is responsive to a state of a neuron which receives an output from a neuron being trained.

Alternatively or additionally, said training is responsive to a correlation between a neuron being trained and a second neuron. Alternatively or additionally, said training comprises back propagation. Alternatively or additionally, said at least one non-synapse-weight parameter comprises a response threshold. Alternatively or additionally, said at least one non-synapse-weight parameter comprises an output signal duration. Alternatively or additionally, said at least one non-synapse-weight parameter comprises an output signal wave-form. Alternatively or additionally, said at least one non-synapse-weight parameter comprises a synapse delay.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be more clearly understood from the detailed description of the preferred embodiments with reference to the accompanying figures, in which:

Fig. 1 is a schematic drawing of a three-layer prior art neural network;

Fig. 2 is a schematic high-level depiction of a neural network in accordance with a preferred embodiment of the invention;

Fig. 3 is a flow chart of a typical neural network construction process in accordance with a preferred embodiment of the present invention;

Fig. 4 is a schematic diagram of a preferred, efficient implementation of a neuron, in accordance with a preferred embodiment of the invention;

Fig. 5 is a schematic high level design of a neural network which scales its input;

Fig. 6 is a schematic diagram of a detail of the connection of a particular scale layer to an input layer in the neural network of Fig. 5; and

Fig. 7 is a schematic high-level diagram of a neural network which functions as a simple calculator.

DETAIL DESCRIPTIONS OF PREFERRED EMBODIMENTS

Fig. 1 is a schematic drawing of a prior art neural network 20. Neural network 20 includes a plurality of input lines 22, a plurality of output lines 24 and a plurality of individual neurons 26. In the example of Fig. 1, neurons 26 are arranged in three layers, a layer 28 which is connected to input lines 22, an output layer 32 which is connected to output lines 24 and an intermediate layer 30 connected between layers 28 and 32. Every neuron in layer 28 is connected to every neuron in layer 30. It should be appreciated that neural network 20 has no structure beyond this layer structure and that each layer does not form a functional group, inasmuch as there are no connections between neurons in the same layer. Such a neural network may be used for approximation or classification tasks, whereby input which is presented at input lines 22 generates an output at output lines 24. Neural network 20 is a three layer neural network, however, it should be noted that there is no structure in neural network

20 beyond that conferred by the layers. In particular, it should be noted that the layer structure is a general characteristic of the architecture of this type of NN, it is not related to the specific task which the NN performs, beyond the decision to use such an architecture for the specific task.

A distinction should be made between two types of neural networks, practical neural networks and models of brain functions. What differentiates these two types is that practical neural networks have an input and an output and the processing they perform affects devices outside the neural network. Brain models do not have and real input or output and their functioning is not related to an actual process being performed outside of the brain model simulation.

Most conventional neural networks have only a few tens of or a few hundred neurons. As can be appreciated, when a neural network, such as neural network 20 is made larger, the number of interconnections between neurons increases to an unmanageable degree. In addition, there is no satisfactory method of debugging any size network, let alone a very large network. One issue to be noted is that neural networks are typically constructed manually, however, there are no accepted methods of building a neural network using team work or by any person other than a very experienced neural network designer. The type of architecture chosen is usually bases on the designer's experience and intuition. Once the designer selected the architecture type, he then selects values for various parameters of the network, such as the number of neurons and generates a network having these parameters. Then, the network is trained to perform a specific task. If the network is not suitable, the training step will fail and the designer can either choose a new network architecture or try changing the parameters.

Another issue to be noted is that many neural networks utilize learning algorithms which seek a minima for some error criteria. In some cases, the function searches for a local minima, in others, it searches for a global minima. As the network becomes larger, the probability of the search mechanism becoming trapped in an incorrect local minima increases exponentially. In addition, some networks, such as Hopfield networks experience the spontaneous appearance of spurious states (local minima) which are unrelated to the task of the neural network. Again, as the network becomes larger, the probability of a spurious state appearing increases greatly.

Fig. 2 is a schematic higher-level depiction of a neural network 40 in accordance with a preferred embodiment of the invention. Neural network 40 is not just an interconnected group of neurons, as in the prior art, rather, neural network 40 has a distinct structure and is made up of sub-components. Blocks 42, 44, 46 and 48 are all conglomerates of neurons and not

individual neurons as shown in Fig. 1. A plurality of input lines 50 may be connected to more than one neuron block and, likewise, a plurality of output lines 52 may also be connected to more than one neuron block and may also interconnect neuron blocks. As will be described below in more detail, each of blocks 42-48 preferably has an individual function and/or structure.

There are several major advantages to using a neural network with an internal structure, such as neural network 40. First, neural network 40 may be described and analyzed in a hierarchical manner. Hierarchy is a major tool for dealing with engineering aspects of complex structures. Another such tool is modularization. Neural network 40 is adapted to be treated as a modular structure. For example, each of blocks 42-48 may be constructed and debugged individually and then connected up to form neural network 40. Many techniques which were developed for large and complex engineering projects can now, for a first time, be applied to the design and construction of neural networks, in particular CASE (computer aided software engineering) tools. An exemplary tool is the use of libraries, whereby a system is built up of components, many of which were designed and constructed for a previous system and may now be reused. As might be appreciated, components in libraries have the additional advantage of having been thoroughly debugged. Components and libraries may also be provided by third parties. A more detailed description of such tools and techniques is described latter in this specification.

Another advantage to using structured neural networks is that the operation of such a neural network is more intuitively understandable by a human programmer. For example, given two neural networks which calculate the function (a+b)*c, a first neural network, in accordance with the present invention, will typically be divided into two blocks, one which performs the addition and one which performs the multiplication. While a second neural network, in accordance with the prior art, will typically perform the entire calculation using one group of neurons, trained to perform this particular calculation, by pattern matching. A human programmer will find it much easier to "understand" the first neural network and can also easily modify it to perform a different function. Further, as explained herein, the calculated function can easily be converted to (a+b)/c, by replacing the "multiplication" neuron block with a "division" neuron block from a library.

In order to understand various aspects of the present invention, it is useful to assign a type to individual neurons in a neural network. The "type" of a neuron is determined by its various properties, including the number of input connections, the threshold of excitation and/or any other activation function and/or the output function. In addition, the neurons are

usually organized into functional groups, each of which usually performs a particular function. One example of a functional group is a column, such as found in the human visual cortex. Each such column can detect if the orientation of an input is a particular orientation. There is no necessary equivalence between functional groups and modules (described below). Usually, but not always, a functional group may be characterized by the density of interconnections between neurons within a group and by the density of connections between neurons of different groups. Typically, the number of connections within a group is larger than outside a group, as is also typical of modules. However, this is not necessarily the case.

As can be appreciated, individually defining neurons and their interconnections can be a very tedious, if not an impossible, task for large numbers of neurons. In a preferred embodiment of the invention, instead of defining all the individual connections, only the probabilities of connections between the different types of neurons are defined. A compiler then generates a neural network which meets the criteria set forth. Preferably, the definitions are stored in a definition file. Additionally, the connection profiles of individual neurons may also be defined in the definition file. As can be appreciated, by defining the probability of interconnections, the neurons are also functionally grouped. In a preferred embodiment of the invention, connections are defined between hierarchies of types, rather than between individual types. Thus, a plurality of types may be grouped together to define a super-type and the probability of interconnection between such a super-type and a second super-type or an individual type are then defined. Grouping may be by spatial location (defined below), functional relation ship, type-definition hierarchy, or any other method.

A functional grouping may include more than one type of neuron. Conversely, similar types of neurons may belong to different functional groups, in which case they are usually defined as having different types. One or more of the following parameters are usually defined for the interconnections and the neurons:

(a) a probability of a (forward) connection between an output of a neuron of a first type and an input of a neuron of a second type; alternatively or additionally to defining a second "type", the connection may be defined between the first type and a functional grouping or a group of types;

(b) a probability of a (backwards) connection between an input of a first neuron of one type and an output of a neuron of a second type;

(c) distributions of neurons types between the functional groupings;

(d) learning properties of an interconnection between two neurons, including, learning function, such as a back-propagation learning function, weight increment and

decrement rules, stop learning criteria, ability to learn is preferably also defined as a distribution on the interconnections, however, it may be defined as a property of a neuron type;

(e) an integrator/output function of a neuron, including an integrator function without a threshold, may be provided as a high-level language function, such as C++; and

(f) excitation threshold;

(g) output signal shape, amplitude and duration, typically an alpha function which has a fast rise, a slow descent, all unipolar and repeating at a certain frequency;

(h) synapse properties (may be defined by connection type and/or by target/source neuron type), including: delay time distribution, weight; and

(i) for input cells, a spatial-temporal response profile.

In addition, the definition file includes other information, such as the number of neurons, the location of the input and output points, global values such as an delay distribution (defined later). In a preferred embodiment of the invention, the definition file has an hierarchical object oriented structure. For example, the characteristics of a particular neuron may be described as being a variant of a different neuron type, with only a few differences (a sub-class in OOP terminology).

Functional grouping of neurons, in accordance with the present invention, may be achieved, *inter alia*, by using one or more of three methods: (a) by creating modular units which are self-contained and which include input and output lines (b) by imposing structure upon an initially homogeneous network or (c) by directly defining interconnection probabilities between neuron types and individual neurons.

In a preferred embodiment of the present invention, and as described in more detail below, method (b) is provided for by assigning spatial coordinates to each neuron. Structure is imposed by selecting spatial groups of neurons and defining their properties, including, a spatial distribution of interconnection probabilities. Preferably, the neurons are assigned three-dimensional coordinates, however, fewer or more coordinates may be used. Preferably, the neural groups are selected and manipulated using a graphical display which shows a projection and/or slices of the spatial representation of the neural network. In such a display, different neuron types are preferably assigned different colors.

The spatial distributions may include direction and/or distance information. Distance usually corresponds to functional grouping, where nearby neurons are more likely to belong to the same functional grouping than further-away neurons. In some preferred embodiment of the inventions parameters other than interconnection probabilities are defined as distributions,

including, the number of connections to a neuron, its excitation threshold or various parameters which may be associated with its output function.

It may not be possible to construct a network using the definition file, in which case, the compiler generates error messages, as described below. Preferably, the compiler also generates warning messages.

Fig. 3 is a flow chart of a typical neural network construction process in accordance with a preferred embodiment of the present invention. The process with first be generally described and then each individual step will be described in greater detail.

It will be appreciated by a person skilled in the art that not all of the described features are necessary to practice the invention. Rather, various embodiments of the present invention may utilize different selected ones of the features described herein.

The process starts when a user generates a definition file of a desired network. Then the user complies the definition file to create an exemplary neural network. Since the compilation is a stochastic process, different compilations will yield similar, but not same neural networks. However, by using a same random number seed for the random number generator in the compilation process, different compilations can be made to yield equal neural networks. If the compiler cannot generate a neural network based on the definition file, the user is so inform and the user must modify the definition file. If the compilation succeeds, the user (preferably) defines at least one input set for testing the neural network. If the user so desires, the input set may be used to debug the neural network, i.e., to determine if it complies with its specification. If at any point in the debugging the user decides that the neural network does not meet its specifications, he may modify the definition file. This debugging is typically repeated with many sets of input data.

Additionally, the user will usually wish to ascertain that the specifications of the neural network are reasonable. Therefore, the user may analyze the results obtained to ensure that not only is the neural network performing its desired function, but that the desired function is actually desired, in view of the rest of a total system in which the neural network will be integrated. This analysis may also be performed using multiple sets of data and, if the user so decides, he may modify the definition file and recompile the neural network.

Once the neural network is debugged and pronounced suitable, it may be integrated into the total system or into some portion of it. Debugging of the total system may also suggest changes in the neural network, which might require modifying the definition file. It should be appreciated that this above described process is a variation of the iterative model for software design, which has not hereto been feasible for the design and construction of neural networks.

NEURAL NETWORK DEFINITION

In accordance with a preferred embodiment of the invention, a neural network editor is provided for generating and maintaining a definition file of a neural network. Preferably, such an editor is a spatial editor whereby a three-dimensional model of a neural network is manipulated by a user. In addition, such an editor preferably also enables text based editing of the definition file. Preferably, the editor includes a grammar checker for checking the grammar of the definition file. A description of one such editing process, in accordance with a preferred embodiment of the invention, follows.

First, the user defines the number of neurons in the network and describes a general three-dimensional form (typically a simple shape, such as a sphere, a cube or a box). Alternatively, the user may select an existing neural network for modification or as a base for design. Additionally or alternatively, the user may select the number of neurons in the net only at the end of the design process. Preferably, the user defines an initial neural network size and modifies this size during the design process. Once a basic neural network design is generated or selected, some or all of the following steps may be performed, in any order decided upon by the user, until the design is complete. Where spatial selections are described, a user may select a portion of a displayed neural network using a pointing device or he may define the spatial form of the selection in functional terms, i.e., ranges of distances and angles. Once such a selection is displayed, a user can preferably modify it using the pointing device.

(a) The user spatially selects a portion of the (displayed) neural network and defines its type. Usually, such a definition is associated with a displayed color. The user may also define various a parameters of the neuron type, such as excitation threshold, number of connections and other parameters as described above. This process may also be performed on an individual neuron basis.

(b) The user controls the amount of data displayed and/or its level of detail. This can be done by controlling the visibility of a selected type of neuron, a grouping of neurons or a type of interconnection. In addition, if functional groups are grouped together in a hierarchical structure, the level of detail of that structure can also be controlled.

(c) The user manipulate a (three-dimensional) view of the neural network..

(d) The user deletes individual neurons, groups of neurons or types of neurons.

(e) The user adds individual neurons.

(f) The user changes global definitions of the neural network, usually using a text-based editor.

(g) The user save a portion (or all) of the neural network in a file. Preferably, using such a saved portion as a building block for a future neural network.

(h) The user loads a neural network portion from a library and interconnects the new neural networks with the one from the library. A neural network portion may also comprise many neural networks. Such a neural network can also be a conventional neural network, either trained, untrained or partially trained.

(i) The user defines an output function for a neuron type or for an individual neuron.

(j) The user defines where the input and the output of the neural network will be. It should be noted that since the user can view any portion or projections of the network, defining the outputs may be only a semantic issue in some cases.

(k) The user selects a group of neurons to be a memory unit and defines how these neurons will be trained.

(l) The user defines or selects a learning function for any trainable portion of the neural network. It should be noted that a neuron may have both trainable and non-trainable connections, since in most cases, the training is in the connections, not the neurons.

(m) The user defines or selects a function which initializes the weights of connections for a group of neurons according to a function, preferably a spatial distribution function. Such a function might be one which modifies existing weight definitions of the neurons' connections. Typically this step is performed for trainable portions, but also possibly for some non-trainable portions of the net.

(n) The user groups several neurons or neuron types as a super-group, for purposes of manipulation.

A neural network portion may be stored as a definition file. Typically, with a random seed which will generate the particular stored network. Alternatively, the individual neurons with their connections and their connection weights may be stored. This is especially useful for storing trained neural networks.

Modification of individual neurons is especially important when modeling sensory neurons. Or when defining neurons which connect to external input and outputs which are not neural network based. In a preferred embodiment of the invention, sensory neurons are defined using mathematical functions. For example, the excitation threshold of a group of input neurons may be defined to change over the range of 1-10, with an overlap of 10% between receptive field areas of (functionally) adjacent neurons. The inputs of the individual input neurons are typically spatially distributed, as are the input weights of these neurons.

Alternatively or additionally, the distribution of connection type, whether inhibitory or excitatory is also spatially distributed. In a preferred embodiment of the invention, the user selects from one of several predefined distributions of the input neurons and their connection types and weights. Alternatively, the user may supply any user defined distribution.

Two types of structured construction of neural networks should be differentiated. A first type is "modular" construction in which finished modules are connected together. Each individual module is usually separately debugged and usually also stored in a library. A second type is "fuzzy grouping" in which groups of neurons are selected and the spatial distribution of interconnections therebetween are defined. "Fuzzy grouping", unlike "modular" design, does not necessarily define sharp transitions between groups of neurons having different functions. In fact, an overlap between two functional groups is possible.

Another aspect of neural network definition relates to defining architectures, especially architectures for trainable neural networks. In a typical prior art architecture, the following three definitions are usually lumped together:

(a) the physical structure of the network, including how the neurons are interconnected;

(b) how the network is trained, for example, by forcing values at its output; and

(c) the training function: how it is decided if to change a weight and how the weight is changed.

In accordance with a preferred embodiment of the invention, each of these three aspects of the architecture may be separately modified, especially the training function. Thus, effectively, new neural network architectures may be created. In a preferred embodiment of the invention, a back-propagation architecture is modified by changing (only) the learning function to a Hebbian learning function.

Once the neural network is defined the user compiles the definition file to generate a neural network.

NEURAL NETWORK COMPILATION

Not all definition files describe a legal neural network. Some definition files may simply contain syntax errors. In some cases, the probability distributions described do not allow any neural network. In other cases, the compiler may not be able to generate a proper neural network in a reasonable amount of time. In such cases, the compiler preferably indicates to the user where the error occurred in the definition file and, preferably, indicates to the user what a suitable correction might be, for example, what range of values for a parameter will enable a neural network to be generated.

The following data files are preferably used as an input to the compiler (and described otherwise herein collectively as a definition file):

(a) A general network architecture file, which describes the topology and size of the net and the types of all the neurons in the net.

(b) A neuron definition file(s) which defines the characteristics of each neuron type, for example, the integrator type and learning function.

(c) A (forward) connection definition file(s), which describes, for each type of connection, parameters which describe the connection, including, weight and statistical delay function.

(d) A backwards connection file, which describes, for each type of connection, the neurons which are to provide the input data (by type and/or spatial distribution), the average number of output connections for each source neuron, a scoring function to use on each interconnection and/or a statistical delay function. It should be noted that backward connections are useful for defining connections from an input layer to a particular functional group of neurons.

(e) A direct connection file, which describes direct connections between individual neurons. Preferably, two types of interconnections are defined, excitatory and inhibitory. In addition, the weight of the interconnection is preferably defined, either as an absolute or as a probability. Preferably, a statistical delay function for the interconnections is also defined.

A compilation process preferably includes the following steps:

(a) Reading and verification of all data input files. In this step the compiler reads the definition files and checks them syntactically and for errors in data. Function definitions, such as an integrator function, are usually verified when they are used. Data and definitions are typically verified in this step.

(b) Generating a net database structure, responsive to the data in the general architecture definition file.

(c) Forward connecting. The forward connections are defined and the connection characteristics, such as weight are initialized. Typically, each target neuron is assigned a probability of being connected to a source neuron and the neurons with the highest score are then defined as being connected. The assigned probability and the selection of a neuron as a target neuron is dependent on a spatial distribution function, which may be user supplied.

(d) Backward connecting. Each neuron "selects" source neurons so that the average number of connections into each neuron is the defined average. Typically, when both forward and backward connections are defined, the number of actual connections is the sum of the two

numbers. Alternatively, the compiler tries to assure that the forward and backward connections will match. Typically, only an average number of connections can be assured.

(e) Additional connections. Any connections which are defined to be performed after compilation, such as changes made on a compiled network, or block copies (defined below) are applied to the network at this point.

(f) Additional table preparations. In this step, various data may be arranged in tables for more efficient use during run time. One example is a list of all the incoming connections to a neuron, which is useful for various types of learning functions.

The compiled neural network may be a stand-alone program for a particular machine. Alternatively, it may be a VHDL definition file for generating a hardware embodiment of the neural network, using further hardware design tools, as known in the art. Preferably, a hardware embodiment of the neural network is in the form of a data table, which defines the neural network and an emulator which uses the data table to perform the neural network functions. Such an emulator may be a standard hardware unit, with the data table being downloaded as firmware. Alternatively, at least some of the data table may be modified as part of a training session. Alternatively or additionally, the VHDL file is of a hardwired version of the network.

In a preferred embodiment of the invention, such a neural network is used as an embedded neural network for the electronic control of an electronic device, such as an air-conditioner, a washing machine or an automobile. A neural network is especially useful when fuzzy-logic type control of a device is required. In an embedded embodiment, the output functions used by neurons are preferably limited to those which are easily and simply implemented in hardware. Preferably, in non-research settings, the compiled neural network and the run time will include a minimum of debugging utilities.

It should be noted that a neural network in accordance with the present invention is inherently better adapted to hardware implementation than conventional neural networks. First, only a small number of the connections between individual neurons are trainable, so most connections can be hardwired. In some cases, there are no trainable connections. Second, as described below, an individual neuron, in accordance with a preferred embodiment of the invention, is very simple and it will be appreciated by a person skilled in the art that such a neuron can be implemented using only a small area of an IC. Third, as described herein, the neural network is typically structured so that most interconnections are local, i.e., to nearby neurons, so the number of long-distance connections is small.

For these same reasons, a neural network in accordance with a preferred embodiment of the invention is also more suitable for a distributed computation embodiment, inasmuch as the neural network is already divided into modules having a limited amount of communications therebetween and all of which are to be executed in parallel.

Once a neural network is compiled, the user may, in a preferred embodiment of the invention, redefine the properties of individual neurons and/or add and/or delete individual neurons, by directly modifying the neural network and without recompiling the entire definition file. Typically this is done using the neural network editor, described above. Typically, such changes are stored in a special section of the definition file and are automatically applied to the neural network when the definition file is next compiled.

A special type of modification is a block copy. In many neural networks, there are significant repetitions of structure. In some cases it is desirable to copy a portion of the network, or only definitions thereof before compilation (during network design). However, in many cases, it is desirable to copy a block of actual compiled connections (block copy), after the definition file is compiled. In a preferred embodiment of the invention, the user defines a network which is large enough for the original and copied blocks and defines connections only for the original block. After compilation, the connections and/or type definitions of the original block, are copied to the unused area of the network. Alternatively, only some of the definitions may be copied. One issue to be considered is connections between neurons inside the copied block and neurons outside the copied block. In one preferred embodiment of the invention, connections are absolute. Thus, if a neuron inside the block was connected to a particular input neuron, then the copied neuron will also be connected to the same neuron. Alternatively or additionally, at least some of the external connections may be defined to be relative. Thus, if the copied block is 10 neurons away, in a particular direction, from the original block, the connection will be transferred to an input neuron which is also 10 neurons away. Alternatively or additionally, a more complex functional relationship may be defined between the original connection locations and the copied connection locations.

TEST-INPUT DESIGN

A neural network development system in accordance with a preferred embodiment of the invention, preferably includes a dedicated input design editor for generating test-input data sets for debugging and testing of a neural network. The input editor preferably enables the user to define spatial input patterns and how they change in time. In addition, the user can select an input set from a library of input sets. The selected input set can then be modified using functions which change the input set spatially or temporally to better match the current

testing/debugging situation. In one preferred embodiment of the invention, the height and/or width of the input data is temporally modified. Data can also be imported, for example by projecting images or image sequences unto the inputs of a neural network. The data in the input set can be associated with individual input lines. Alternatively, the user defines the geometry of the input lines (matrix, linear, etc.) and allows the input design tool to project the data unto the input lines. This technique is especially important when the same data is used to test a plurality of neural networks having different scales. The projection method may also be user defined, for example, to include a smoothing function.

Preferably, the user can define an input resolution and/or a background intensity level.

NEURAL NETWORK DEBUGGING

In any system which does not translate specifications directly into executable programs, any generated programs must be debugged to verify that they work (i.e., do not hang or cause faults) and that they meet the specifications. In prior art neural networks, since nothing is known of their inner-workings, only the output of the network is analyzed. However, in accordance with a preferred embodiment of the present invention, since the neural network is constructed to have and has a structure and functional groups, the inner-workings of the neural network are preferably debugged. The inner-workings may be analyzed, either as modules or even down to the level of an individual neuron. The outputs may be analyzed in real-time or they may be stored in data files and analyzed after the execution of the neural network is over. Off-line analysis of module outputs is important in order to determine mismatch between two modules of the neural network.

Debugging may be approached using one or more of the following techniques:

(a) single stepping through the operation of the network;

(b) tracing the output (spike train) and/or input of individual neurons;

(c) tracing the output of groups of neurons, possibly the output being reprojected in a user defined manner;

(d) replacing modules with stubs, which stubs may be emulated by programs, by a data table or by a different module;

(e) dictating input values to various neurons, or, more preferably, dictating outputs of various neurons;

(f) tracing and/or debugging of the output functions of individual neurons, which may be written in a "standard" high-level language;

(g) partially retracing the execution of a neural network and presenting new input or changing various parameters of the neural network to determine their effect on the output of the neural network;

(h) analyzing a correlation between two or more neurons, preferably shown as a correlation diagram of their outputs and/or their inputs; and

(i) viewing a graphical representation of the connections of a single neuron or a (specified) group of neurons and analyzing changes in the activity on the connections and/or changes in weights (in a learning portion of the neural network).

As with the input, the output of the neural network or of portions thereof may be projected in real-time to generate an image or a sequence of images. Alternatively, a designer may view the activity of a slice or a spatial sector of the neural network, as it changes with time.

One important issue which should be tested during debugging is the tendency of the neural network to hang, i.e., not reach an end-state. Neural networks according to the present invention are less prone to hanging, as described below, however, the tendency to hang is important and testing for it is usually performed by generating a very large range of input sets and determining the distribution of relaxation times of the neural network. Typically, once an input set which generates long relaxation times is identified, new input sets which are similar to that input set are generated.

In a preferred embodiment of the invention, such a debugging activity is performed automatically by the debugger.

The debugging process described above may be repeated for many input sets and the neural network modified if necessary until the neural network is deemed to be debugged. It should be appreciated that in some cases, debugging will necessitate changes in the definition of the neural network. Some of such changes may require debugging the neural network again with previously used input sets. In such cases, the output of the neural network may be compared to the output of the unmodified neural network, either in real-time or off-line.

It should be appreciated that a neural network may be debugged either before, after or during training (if such training is relevant for the particular neural network). When debugging a neural network during training, an additional feature which is preferably provided, is the ability to partially undo the training of a neural network.

RESULT ANALYSIS

The output of the neural network, obtained using the debugger, may be exported to an analysis program, such as a spreadsheet or a mathematical package. These results are

preferably compared to the specifications of the neural network to determine if the user has properly defined the neural network. If the neural network does not meet the specifications, the user must modify the neural network. The results may also be analyzed to determine the efficiency of the neural network. Efficiency may be measured by the number of neurons required to perform a task and by the time it takes to perform the task.

In a preferred embodiment of the invention, the analysis of the results is used by an automatic programming system to automatically modify the neural network to better match its specifications. Automatic modification is made possible by the combination of two factors (a) the generation of the neural network from a compiled definition file and (b) the structure of the file being mostly parameter definitions and not structured programming steps.

One example of automatic modification is in scaling of the neural network. A user defines a maximum number of neurons which he is willing to have in a particular neural network and also defines output specifications for various input sets to the neural network. The user then designs a neural network and allows the system to automatically generate a neural network having a minimum number of neurons. In accordance with one preferred embodiment of the invention, the system first generates a neural network having the maximum number of neurons and then the system thins the neural network by randomly deleting neurons from the neural network and checking to see if it still matches the specifications of the original neural network. In accordance with another preferred embodiment of the invention, the system recompiles the neural network with a smaller number of neurons instead of thinning it, until it can no longer meet the specification. The optimal number of neurons is preferably determined using a search techniques as known in the art, for example, binary search. The user may also define portions of the neural network which may be thinned and portions which cannot be thinned, for example, the input neurons may be designated as being non-thinable.

Another example of automatic modification is scale increasing, where a user defines and debugs a small-scale neural network and then allows the system to generate a new neural network having a larger scale and/or resolution.

Yet another example of automatic modification is optimization-type searching. In this technique, the user defines a neural network and defines permissible ranges for various of the parameters therein. The system searches the space formed by the allowed ranges of parameters to find a set of parameters which better meets the specifications required of the neural network. The system then uses this set of parameters to generate a neural network.

Another aspect of the present invention relates to optimization techniques for pruning the compiled neural network. One technique comprises running several test data sets on the

neural network and erasing all neurons which fire less than a certain amount. Another technique comprises running several test data sets on the neural network and erasing a connection if it was used less than a predetermined number of times or if it affected the output of less than a certain number of neurons a certain number of times.

INTEGRATION

Once a neural network module is generated it can be integrated into a real-world system. Such a system may be a more complex neural network or it may be a "standard" software system which uses the neural network for a particular, designated task. Additionally or alternatively, the system may be a neural network system which uses "standard" software for a particular designated task.

As described above, a neural network module may be selected from a library to be integrated with a parent neural network. In some cases, a library module may be automatically scaled to match the parent neural network, for example, by generating a new neural network module with the correct scale. In some cases, a neural network may be scaled simply by increasing the number of neurons of each type, in the network's definition file. One such example is a when the neural network comprises a plurality of columns and each column has a different type of neuron. In other cases the library module may be non-scaleable, for example, as a result of some types of manually design. In such cases, a special, matching, neural network may be used to interconnect the library module to the parent neural network. The matching neural network typically scales input and/or output between the parent neural network and the library module. Matching neural networks can be automatically generated from a generic specification by defining the number of inputs and the number of outputs of the matching network. Such a matching neural network will usually both scale the input to the output and apply a smoothing function.

One example of such a neural network comprises a single layer. The inputs of each neuron in this layer are connected, using a Gaussian distribution function to outputs of the parent network. The size of the source area is dependent on the scaling factor. For example, in a one dimensional matching example, if the input is half the size of the output, than the size of the source area is a little over two neurons.

Additionally or alternatively, a matching neural network may generated by supplying a standard neural network and training it to convert an input data pattern to an output data pattern. Alternatively or additionally, a matching neural network may be manually constructed.

In a preferred embodiment of the invention, when a neural network is attached to an existing neural network and has a different size therefrom, blank neurons are automatically provided to buffer the smaller of the two neural networks. Preferably, some or all of these blank neurons, if not otherwise interconnected later, are deleted by the user.

One important issue in multi-module neural network is the synchronization between different modules. In the prior art, neural networks are treated as a single unit, whose output is meaningful only after the network output stabilizes. In contrast, in a preferred embodiment of the invention, the operation of various modules of a neural network are synchronized. preferably, one of two types of synchronization is used, either a system clock type synchronization, where all of the modules (but not necessarily functional groups within the modules) are synchronized to a single clock, or a serve-when-ready type synchronization, where a particular module operates when all of its inputs are ready. It should be appreciated that these two synchronization methods can be combined. In addition, there can exist several groups of modules, each with its own local clock. In a serve-when ready type synchronization, some modules may operate even if not all their inputs are ready. In such cases, there is preferably an input to these modules which indicates that certain inputs are not ready yet.

These synchronization techniques may be applied using the network itself or with the aid of an external, algorithmic, program.

In a preferred embodiment of the invention, a buffer module is provided between two connected modules. This buffer module has the property that its output is substantially one value when its inputs are changing and the input values once these values have stabilized. Thus, the connected module can easily change its functioning responsive to the stabilization of its input. A buffer module can constructed, for example, using a neural network having a plurality of layers, where a delay is provided between a first layer and a second layer and wherein the output of the buffer module is the input of the module, inhibited by the subtraction of the first two layers. If the two layers have substantially the same data, then the input is stabilized and no inhibitory signal will be generated. If however, the two layers are different, an inhibitor signal will be generated and the output of the buffer module can be some preset value.

Such a buffer module can be enhanced to form a latch-buffer, where the previously stabilized input is provided until a new stabilized input is available. The latch may also be activated by an external signal, which may be supplied by an external clock.

In a preferred embodiment of the invention, a separate module detects, from a plurality of such buffers and/or module, the state of readiness of data and decides which modules are

allowed to proceed with their operation. This type of centralized control is especially important when the neural network must interact with real-world tasks.

A clock is preferably provided by a dedicated neural network module which generates a periodic signal. This signal preferably only controls the transfer of data between module, while within modules, data transfer may be controlled either using internal clocks or without control.

In one preferred embodiment of the invention, one module "runs" a second module as a subroutine, at each run possibly controlling the activity of the module through specific inputs. Preferably, the one module does not start a second ruin until the first run is over. The one module may run the subroutine module with a plurality of different data sets and then generate an output responsive to which of the runs was most successful.

Combining a neural network with an existing "standard" software system requires that the inputs to- and the outputs from- the neural network be translated between binary notation and the notation used by the neural network. The input data may then be projected onto the network, or provided as input to selected neurons. Preferably, several converters are provided for converting from standard file formats to the input format preferred by the neural network. As mentioned above, the transfer (output) function of an individual neuron may be any user defined function, including a compiled C++ function. Data-input to a neural network may thus be achieved by providing the neural network with an individual neuron whose output function reads data from the surrounding "standard" software system. Output from the neural network may be achieved in a similar manner.

In a preferred embodiment of the invention, the entire activity of the neural network is recorded as an output file. Alternatively or additionally and especially where the neural network interacts with an outside system, the user preferably defines one or more output functions which translate the activity of the network to a particular data output. Such a function may be, *inter alia*, a selection of a few of the neuron outputs, a projection of such outputs, statistics of the activity of all or part of the neural network or even a post-processed output from at least a portion of the network.

Synchronization between a neural network portion of the system and a "standard software" portion of the system may also be achieved using special neurons. In addition, the output of individual neurons may be delayed using their individual output function. In this manner, it is possible to control the flow of data and/or functioning of a neural network using a high-level language. High-level languages, such as C++, are typically more suited to process

control than are neural networks. Such a C++ function may be used to supply clock signal and make decisions regarding data flow.

When the "standard software" component is to be integrated into a neural network system, the connection therebetween can be achieved using similar methods to those described above with respect to a neural network component in a software system.

It should be appreciated that a neural network system in accordance with a preferred embodiment of the invention may incorporated several different types of neural networks and/or standard software and/or prior-art type neural networks. In a preferred embodiment of the invention, the neural network generation system includes general architectures for prior-art type neural networks. Thus the system as described above may be used to construct, test and debug any type of neural network.

By the introduction of structure and modular- and hierarchical- design to neural networks, the present invention enables the use of many CASE (computer aided software engineering) tools in the field of neural network design.

Some of the most important tools have already been described: using libraries for standard components; reusing modules and/or designs of modules; using an iterative design process; dividing work between different people; hierarchical decomposition of a project; modular construction of a project; debugging, especially using program stubs; and test-data set design.

In addition, other CASE tools and techniques may be used, including tracking changes of the definition files and their effect on the neural networks; version control; and library modules.

IMPLEMENTATION DETAILS

Although the above described methods of neural network design and construction are suitable for many implementations of neural networks, the inventors have found the implementation described below to be especially useful and efficient.

A first design issue is the use of spike train encoding. In most neural network systems, the neurons generate either a binary value or a single multi-valued output (amplitude encoding). In a preferred embodiment of the present invention, a neuron generates a train of pulses in which the data is encoded by variations in frequency and length of the pulses. Preferably, the train of pulses is continuous with the data being encoded by changes in the frequency of the spikes.

A second design issue is preventing phase-locking. Spike-train encoding is inherently less susceptible to phase-locking than amplitude encoding. In addition, in a preferred

embodiment of the invention, a random delay is attached to the output function of the neuron. The addition of such a delay does not adversely affect the data carried by the spike train but it does reduce the probability of all the neurons firing in phase with each other. The random delay may be embodied as a different constant delay for each neuron (or even neuron type) or it may be embodied as a random delay each time the neuron generates an output.

It should be noted that other aspect of the invention may be performed using amplitude encoding or combined amplitude and frequency encoding.

A third design issue is providing time-decay of the input. In a preferred embodiment of the invention, signals which reach a neuron at a later time are reduced in importance by a reduction in amplitude. Preferably, a biological type function is used, which reduces the amplitude of both early- and late- arriving spike signals. Alternatively, any user defined decay may be used, including a Poisson decay and a Gaussian decay function, which reduces the amplitude of both early- and late- arriving spike trains.

Fig. 4 is a schematic diagram of a preferred, efficient, implementation of a neuron 60, in accordance with a preferred embodiment of the invention. Neuron 60 is shown receiving inputs 61, 63 and 65 from three neurons 62, 64 and 66 (respectively). Neuron 60 includes a time-stamped circular buffer 68 into which all such input is placed. When the sum of the inputs is above a threshold 76, neuron 60 generates an output 78. Each data sample from inputs 61, 63 and 65 is time-stamped and weighed with a decay value responsive to that time before being summed with the values already in the buffer. As described above, the decay value may be generated from a user defined table or it may be a peaked distribution. Preferably, neuron 60 only checks the current data value to see if it is above a threshold. Alternatively, a moving window average of some sort may be used. Typically, a certain propagation delay time is associated with the propagation of input from its arrival at neuron 60 until it can be used. In a preferred embodiment of the invention, a refractory period is defined for each neuron, which limits its ability to generate a second output signal at too short a time after generating a first output signal.

In a preferred embodiment of the invention the only delay between the generation of inputs 61, 63 and 65 and their insertion into buffer 68 is a time delay associated with preventing phase-lock. Alternatively or additionally, the distance between neurons is interpreted as adding a delay factor. Such a distance may be defined using purely geometrical terms (as appropriate for the n-space of the neural network). Additionally or alternatively, the distance may be defined by the belonging of two neurons to different modules. Alternatively

or additionally, a different delay and/or delay distribution is assigned per type of connection between neurons.

In a preferred embodiment of the invention, the integrator function compares the sums of the inputs to the threshold 76. Alternatively. Any other type of integrator function, possibly user defined, is used. In one example, the integrator function models a resting potential, by modifying the effective threshold. Alternatively, other aspects of the cell membrane may be modeled by the integrator function. In another example, the integrator function uses a windowed average of the time-stamped buffer rather than a single value.

In a preferred embodiment of the invention, more than one time-stamped buffer is used. In one example, each type of input connection has its own input buffer. In another example, inhibitory inputs have a different buffer than excitatory inputs. The sum of the inhibitory inputs may be subtracted from the sum of the excitatory inputs to model a single buffer. Alternatively, inhibitory inputs are assigned a greater weight when the excitatory inputs are lower. In one preferred embodiment of the invention, each buffer has a different refractory period, such that after a generation of an output signal, the values in the buffer are considered zero. In a preferred embodiment of the invention, the refractory period is modeled by changing threshold 76. Thus, a relative refractory period is also possible.

It will be appreciated by a person skilled in the art that since only addition and multiplication (by a weight) need to be performed, simulating the operation of neuron 60 can be done very efficiently. Alternatively, an implementation of neuron 60 can be made fast and simple.

One special type of neuron is an input neuron. In a preferred embodiment of the invention an input neuron has its inputs arranged in a particular spatial distribution. Preferably, there is at least some overlap between two adjacent neurons. Alternatively or additionally, when two neurons are sensitive to different input amplitudes, there is preferably at least some overlap between their sensitivity ranges. It should be noted that spatial distributions may also have more than two dimensions. Thus, when data is projected onto a neural network, a solid shape may be projected unto a three-dimensional array of input lines. It should be noted that such spatial distribution functions can perform filtering, such as smoothing, hi-pass and low-pass filtering.

In a preferred embodiment of the invention, an input neuron has a integrator function which converts an amplitude input signal into a frequency modulated signal. Preferably, this is achieved by integrating all the values in the input buffer and generating a spike when the values pass a threshold. The next integration cycle uses only new data. In a preferred

embodiment of the invention, the neuron has a baseline output frequency. This baseline frequency is modified responsive to the input. If the input has a low amplitude, the output frequency is lowered below the baseline, if a higher amplitude than the frequency is increased. In a preferred embodiment of the invention, the input neuron includes a decay function, such that old data in the input buffer is ignored. Alternatively or additionally, the neuron adapts to a certain input level by applying a decay function to the input, when the input is constant.

In a preferred embodiment of the invention, the neural network is a stand-alone program which can run on any general purpose computer, for example, in the WINDOWS 95® system, it preferably includes a data file and an associated DLL. In a preferred embodiment of the invention, an encapsulated solution is provided, such that the generated neural network is ready to run. Prior art applications typically generate a c-code program which needs to be complied by a user. In a preferred embodiment of the invention, the encapsulated neural network includes a input and/or output transformation filters which convert "algorithmic style" data to data suitable for a particular neural network and vice versa. As used herein, the term neural network includes, *inter alia*, a hardware device which emulates the function of a neural network, a hardware neural network and a software program which emulates a neural network. Typically, such software is provided for a general propose computer.

In a preferred embodiment of the invention, the runtime process includes the following steps:

(a) Reading input into the net. In this step, the input is read. Preferably, the input function is user supplied. Such an input function can read ASCII format, a database format or an internal format generated by the input-editor. All the data is preferably loaded at one time so that a data vector, having a time dimension is generated. In a network having a two-dimensional input layer, the data vector will be three dimensional: X, Y & T. The T dimension is discrete and depends on the time-tick resolution. Alternatively, and especially in real-time neural networks, where the data is not previously available, data is loaded and converted for each tick in real-time.

(b) Initializing all run-time functions and data.

(c) Validating the running functions. Each function is validated to ensure that it is properly called and has all of its associated data files. Preferably, user defined functions include a validation portion which is run at this time.

(d) For each time tick, for each neuron, all the input data to a neuron are entered into the buffer(s) of the neuron; the integrator on the neuron is run; and the outputs from the neuron are generated.

(e) Outputs are sent to a data file and/or another program (as described above) either for each time tick or after the run is over for example, if the network output stabilizes, the input data is finished and/or a predetermined amount of time passes. As described above, the activity of each neuron in the net can also be saved.

In a preferred embodiment of the invention, a user selects only a subgroup of neurons to be active during a run. Preferably, the user selects a module unit of the neural network. Preferably, the selected group of neurons is run on a recording of a previous run of a neural network. In such a case, the neuron repeats tits activity in that run. Then, the output of this module is used as an input for the testing of a second module.

It should be noted that two runs may not have exactly the same output, due to the stochastic nature of the network. However, the average output will generally be the same.

For a learning neural network the above described process may be different. Typically, a learning neural network is run in two modes, learn and normal. Normal was described above. However, it should be noted that some neural networks in accordance with a preferred embodiment of the invention are always in a learning mode.

In a learning mode, a set of inputs (training set) is preferably defined to be run on the network consecutively. Learning functions (preferably user defined) are called during the run time to effect the learning process. At least a primary set of parameters of these functions is preferably defined in the system. Preferably, user defined functions can be run at the start and/or end of each training set, the start and/of end of each input set and/or at the start and end of each time tick. These functions may also be used to determine if to repeat an input set, skip to a next input or a next training set and other training control functions. Preferably, different functions may be defined for each connection type. Another function, preferably run during a time tick, is preferably defined to calculate neuron-dependent calculation. This function may in turn use other functions which determine if to change connection weights and/or to effect these changes. Typically, most of these myriad of functions will perform no actions.

Alternatively or additionally, to learning by changing synapse weights, other parameters of neurons and/or inter-neuron connections may be modified as part of a learning scheme. These parameters include, for example, synapse delay, threshold values, refractory period and/or shape and/or duration of the neuron output signal. The modification of the parameters may be a function of global and/or local state of the neural network. Global states

include, for example, the performance of the entire network and/or modules thereof, for example an error function used for supervised learning such as back propagation. Local states include for example an activity level and/or distribution of activity of the current neuron, characteristics of the activity at below threshold levels, the activity of neighboring neurons, such as those which provide input to and/or receive output from the current neuron and/or activity levels of groups of neighboring neurons, especially complete modules. Alternatively or additionally, a user may defining a training connection between non-neighboring neurons and/or groups of neurons. One type of preferred training relationship is a Hebbian-like learning function in which a parameter is increased, decreased and/or optimized as a function of correlation between instantaneous characteristics of two neurons, for example, their firing.

A shape of an output function may be modified, for example, by selecting between several preset function forms. Alternatively or additionally, the output function may be parametrized and these parameters may be varied. In one example, the decay length of the output signal may be such a parameter.

In a preferred embodiment of the invention, the modification of parameters of neurons as part of a learning process may be defined to be non-monotonic and/or non-linear functions. In one example, a synapse threshold is decreased if the average level of activity of the synapse is between a minimum and a maximum values. These values may also be calculated on the fly, based, for example, on the instantaneous activity level in the entire neural network.

In the above described embodiments, only a few of the many available parameters known to apply to neurons and to complex neurological structures have been used. As can be appreciated, the above-described framework may be easily expanded to include these parameters. However, the use of a structured neural network is deemed to be the most important single valuable parameter. In addition, the above implementation includes a particular set of parameters which the inventors have determined as being suitable for neural network design.

The present invention includes the use modeling of many such parameters, including: resting potential, threshold, refractory period, temporal encoding of information, stochastic delay in firing, usage of post synaptic potential and not binary outputs, distance dependent synaptic delays and statistical definition of spatial distribution of the connections. In particular, it should be noted that the spatial description of a neural network in accordance with a preferred embodiment of the invention, make it simple to model the effect of spatial distributions of neurotransmitters and their diffusion through the brain. This might be achieved, for example by defining a time-varying spatial distribution function of the neurons,

determining the local concentration at each neuron and modifying the behavior of each neuron responsive to that local concentration and a model of the effect of that concentration on that neuron type. In a preferred embodiment of the invention, neuron types may, as part of their output function, affect these spatial distributions, by a feedback mechanism, such as modeling the (local) release of selected neurotransmitters and/or activation of enzymes which breakdown such neurotransmitters.

DESIGNING NEURAL NETWORKS

Designing structured neural networks to perform particular tasks is a new field and, as such, there is no existing rule book. However, the following guidelines can be suggested.

(a) In many cases the task can be broken down into modules, preferably in a hierarchical manner using methodology similar to that well known in the art of design of computer software and complicated electronic hardware.

(b) Where possible, such breakdown should include as many existing neural network modules as possible.

(c) Within a particular module, various parallel-programming techniques may be used. In particular, the following process may be used:

(1) Decide what the positive portion of the output of a network should look like, responsive to its input.

(2) Decide what the blank portions of the output should look like.

(3) Analyze the outputs and inputs of the net to define a discrete set of input and/or output conditions.

(4) Parametrically define a group of cells, that for a particular input condition generates the desired output. Used the parametrized definition, a different group of cells may be defined for each input condition.

(5) Define a criteria for a group of cells to inhibit another group.

(6) Connect, in parallel, a plurality of groups of cells, so that only one group can generate an output.

EXAMPLES OF STRUCTURED NEURAL NETWORKS

Like integrated circuits and computer programs, structured neural networks may take a variety of forms and perform a variety of functions, limited only by the ability of a neural network designer. The following examples show neural networks which operate in a manner completely different from conventional neural networks or which perform functions which it has not been possible to implement hereto.

SCALING NEURAL NETWORK

In classical pattern matching, the following procedure for matching an input stimuli to stored representations often used: scaling the input to a certain size; rotating the input to a certain angle (the size and angle should be the same as for stored representations; and comparing the scaled and rotated representation to the stored representations. Typically only the last step has been performed using neural networks. In accordance with a preferred embodiment of the invention, also the first and second steps are performed using a neural network.

Fig. 5 is a schematic high level design of a neural network 100 which scales its input. Network 100 comprises an input layer 102, including an area 104 on which the input stimuli is projected. Network 100 also comprises a plurality of scale layers 106, 110 and 114 and a n output layer 118 on which the output of the network will be projected. The purpose of network 100 is to generate an output in which the input best utilizes the size of the output area. Thus similar inputs will generally be scaled to similar sizes.

Network 100 operates in the following manner, each possible scale of image is generated and the image scale which is best is projected unto the output. To this end, the number of the scale layers is dependent on the scale resolution desired. Each of the scale layers is connected to input layer 102, with a different scale. For example, input stimuli 104 is enlarged to a region 108 in layer 106 and reduced in size in regions 112 and 116 of scale layers 110 and 114, respectively. These connections are indicated by the lines marked "data". Each layer inhibits all the layers which have a scale smaller than itself, these inhibitory connections are marked as "inhibit". All the layers are projected unto the output layers, with their outputs summed. These connections are marked as "project". In a preferred embodiment, scale-layers also inhibit layers with a larger scale (not shown), however, this inhibition is preferably weaker than the inhibition of smaller-scale scale layers.

In operation, the layer with the highest activity will inhibit the operation of all the other layers, so that the output layer will contain substantially only the data from the layer with the highest activity.

Fig. 6 is a schematic diagram of a detail of the connection of a particular scale layer 120 to input layer 102 in neural network 100 of Fig. 5. For clarity, only the connections of a single line 122 of neurons to a single line 124 of neurons is shown. In this simple 2:1 scaling, each neuron in the scale layer receives input from two neurons in the input layer. As can be appreciated, a smoothing function may be applied during scaling, such as by partially overlapping the inputs of the neurons in line 124.

In the above description is has been assumed that the input stimuli is centered in the input layer. As can be appreciated, centering can be achieved in the same manner as scaling. Instead of scale-layers, shift layers are used. Preferably, the inhibitory connections between a shift-layer and other layers are weighed so that data in the center is preferred over data at the edges.

ROTATING NEURAL NETWORK

A neural network which rotates its input may be constructed similarly to the scaling neural network, except, that instead of scale-layers, rotate layers are used. The data in each rotate layer is at a different orientation relative to the input layer. In addition the inhibition is between all the different orientations and not only in a particular order of inhibition.

In a preferred embodiment of the invention, the aspect ratio of the rotated image is the same as the input layer. However, there exists a trade-off between the width of the rotate layer and the precision of the rotation. If the aspect ration of the rotate layer the same as the input layer, the network will generally be more robust. However, if the rotate layer uses a large aspect ration, the precision of the rotation will be higher. A large aspect ratio can be achieved, for example, by reducing the resolution of the rotated image, during the rotation.

A combined scale-rotate network may be generated using a single network with a plurality of scale-rotate layers, each with a different scale and rotation. Alternatively, two networks, one scaling and one rotating are connected together as two modules.

Another possible rotation network is based on a Nobel winning model of how the visual cortex determines an orientation. In this model, there is an input layer and a plurality of rotate-column (which correspond to rotate-layers). However, the connection between the layers is statistical rather than deterministic. Each rotate-column corresponds to a slice of the image on the input layer. Such a slice is a projection of a rectangular portion of the input layer, which portion has a particular orientation. Each neuron in the rotate column is randomly (at least without any logic currently known) connected to a number of input neurons. Thus, for a particular rotate column, the average rotation of the neurons is a particular angle. However, the column does not correspond to a rotated image. Possibly, the column also encode the input image in some unknown manner.

It is not thought that there is competition between the orientation column at this stage in the visual processing. However, in a practical rotating network, such competition is preferably provided.

It should be noted that this network does not rotate an image, it only provides the rotation angle. However, such a determined rotation angle can serve as excitatory and inhibitory connections to control a second network which does perform rotation.

MEMORY UNIT NEURAL NETWORK

In conventional neural network, the function of the entire neural network may be summed up as "pattern recognition". In order for a conventional neural network to perform its function it must be trained. In addition, material which is learned by such a conventional neural network is stored over the entire neural network. In a neural network in accordance with a preferred embodiment of the invention, many functions besides pattern recognition may be performed by a neural network. These functions are performed by the neural network as-is, without requiring the neural network to be trained. Such a neural network may of course incorporate a trainable portion. However, a neural network, in accordance with a preferred embodiment of the invention, can utilize a trainable segment for storing data, for example, when comparing two images which are received with a time delay.

Fig. 7 is a schematic high-level diagram of a neural network which functions as a calculator which can either add or subtract a number from a currently displayed sum. Portion 80, indicated by the dotted line, is the neural network. Neural network 80 has one output, to a display 82 and three inputs, a numerical input 84, a input 88 which indicates if an "add" button was pressed and an input 86 which indicates if a "subtract" button was pressed. Neural network 80 includes a memory 90 which continuously outputs its contents, to display 82, to a subtraction unit 94 and to an addition unit 92. Units 92 and 94 are hard-wired to add (or subtract) numerical input 84 and the input from memory 90. However, unless either the "add" or the "subtract" button is pressed they output the value in memory 90. Thus, the value in memory 90 is not changed. If such a button is pressed, memory is then "trained" with the new value. Since memory unit 90 is completely under control of neural network 80, it can be reset and the training started and stopped at will. Preferably, trainable interconnections are defined to be trainable by defining the properties of the neuron which generates the signal on the interconnection. Input and/or output connections of a neuron may be defined as trainable. As with regular connection types, the connection of trainable interconnections may be user determined using a spatial distribution. A trainable neuron, in accordance with a preferred embodiment of the invention, includes inputs which reset its learned output function and/or set it to a desired value and/or instruct the neuron to begin training or to stop training.

Alternatively, a neural network in accordance with a preferred embodiment of the invention has a portion which is defined by the user as trainable, in the usual sense of the

word. The user then preferably defines the type of training and the training function. The training might then be performed before incorporating the neural network in a system, it may be performed during a calibration stage of the system or it may be performed as part of the regular operation of the system.

The above examples may not be the ideal way to perform the above described functions. However, these examples illustrate the breadth, scope and applicability of the present invention.

The present invention has been described in terms of preferred, non-limiting embodiments thereof. It should be understood that features described with respect to one embodiment may be used with other embodiments and that not all embodiments of the invention have all of the features shown in a particular figure. In particular, the scope of the invention is not defined by the preferred embodiments but by the following claims. When used in the following claims, the terms "comprises", "comprising", "includes", "including" or the like means "including but not limited to".

CLAIMS

1. A method of constructing a neural network, comprising:

providing a definition file, containing high-level specifications regarding a desired neural network; and

compiling the definition file using software to generate a neural network which meets the specifications.

2. A method according to claim 1, wherein the specifications comprise a statistical specification of neuron interconnections.

3. A method according to claim 2, wherein the statistical specification comprises a specification of connections from outputs of neurons to inputs of other neurons.

4. A method according to claim 2, wherein the statistical specification comprises a specification of connections to inputs of neurons from outputs of other neurons.

5. A method according to claim 2, wherein the statistical specification comprises a spatial distribution.

6. A method according to claim 1, wherein the definition file comprises definitions of a plurality of neuron types.

7. A method according to claim 6, wherein the definition file comprises definitions of at least 5 neuron types.

8. A method according to claim 6, wherein the definition file comprises definitions of at least 15 neuron types.

9. A method according to claim 6, wherein the definition file comprises definitions of at least 50 neuron types.

10. A method according to claim 6, wherein the definition of neuron types comprises a hierarchical definition of neuron types.

11. A method according to claim 1, wherein the definition file comprises a modular definition file, each module comprising a definition of a neural network module and the definition file defining at least one interconnection between individual modules.

12. A method according to claim 1, wherein the definition file comprises a definition of parameters for a trainable and uniform neural-network architecture.

13. A method according to claim 1, wherein the definition file comprises user-defined functions which perform portions of a neuron's activity.

14. A method according to claim 1, comprising:
testing said generated neural network;
modifying said definition file; and
repeating said compiling, said testing and said modifying until said generated neural network meets a predetermined criteria.

15. A method according to claim 1, wherein compiling comprises:
generating error messages responsive to mistakes in the definition file.

16. A method according to claim 15, wherein the errors are caused by a non-existence of a neural network meeting the specifications.

17. A method according to claim 1, wherein compiling comprises:
generating an array of neurons; and
assigning a particular type definition to selected one of the neurons.

18. A method according to claim 1, wherein compiling comprises:
generating an array of neurons; and
defining forward connections between a first plurality of neurons and a second plurality of neurons.

19. A method according to claim 1, wherein compiling comprises:

generating an array of neurons; and

defining backward connections to a third plurality of neurons from a fourth plurality of neurons.

20. A method according to claim 1, wherein compiling comprises:

generating an array of neurons; and

defining direct connections between at least one neuron and at least a second neuron.

21. A method of neural network design, comprising:

determining a function to be performed by a neural network;

designing a neural network architecture suitable for performing the function; and

generating a neural network having the designed architecture.

22. A method according to claim 21, wherein designing an architecture comprises imposing a structure on a homogeneous neural network.

23. A method according to claim 21, wherein designing an architecture comprises interconnecting at least two existing neural network modules.

24. A method according to claim 21, wherein designing an architecture comprises defining functional groupings of neurons in the neural network.

25. A method of neural network construction, comprising:

providing a first neural network;

providing a second neural network; and

interconnecting the first and second neural networks.

26. A method according to claim 25, wherein the first neural network and the second neural network each comprise neurons having different characteristics.

27. A method according to claim 25, wherein providing a second neural network, comprises selecting the second neural network from a library of neural networks.

28. A method according to claim 25, wherein providing a first neural network comprises debugging the first neural network prior to said interconnecting.

29. A method according to claim 25, wherein providing a first neural network comprises training the first neural network prior to said interconnecting.

30. A method according to claim 25, wherein interconnecting the first and second neural networks comprises providing a third neural network which matches the inputs and outputs of the first and second neural networks.

31. A method according to claim 25, wherein the first and second neural networks each have a scale and comprising changing the scale of the second neural network to match the scale of the first neural network.

32. A method of designing a neural network, comprising:

providing a initial neural network;

providing a range of allowed values for at least one parameter, which parameter defines allowed mutations in the initial neural network; and

searching an answer-space defined by the allowed range to find a neural network which is more optimal, in a predefined manner, than the initial neural network and which found neural network performs a predefined function.

33. A method according to claim 32, wherein searching comprises:

automatically generating a new neural network; and

determining if the new neural network performs the predefined function.

34. A method according to claim 33, wherein automatically generating a new neural network comprises modifying an existing neural network.

35. A method of neural network construction, comprising:

defining a function to be performed by a neural network; and

constructing a neural network to perform the function, wherein constructing does not include training the neural network to perform the function.

36. A method of neural network construction, comprising:

spatially defining a group of neurons of the neural network, which group comprise only a portion of the neurons in the neural network; and

setting a characteristic of the defined group.

37. A method according to claim 36, wherein the characteristic comprises a spatial distribution function.

38. A method according to claim 36, wherein the characteristic comprises a compiled high-level language function which determines the output response of the neurons in the group.

39. A method according to claim 36, wherein the characteristic comprises the probability of connection from the selected neurons to other neurons.

40. A method according to claim 36, wherein the characteristic comprises the probability of connection to selected neurons from other neurons.

41. A method according to claim 36, wherein the characteristic comprises an integrator function of the neurons.

42. A method according to claim 36, wherein the characteristic comprises a firing threshold of the neurons.

43. A method according to claim 36, wherein the characteristic comprises a number of connections from said neurons.

44. A method according to claim 36, wherein the characteristic comprises a delay distribution function of outputs from said neurons.

45. A method according to claim 36, wherein the characteristic comprises a distribution of connection weights for said neurons.

46. A method according to any of claims 1-45, comprising, constructing an integrated circuit which performs the function of said neural network.

47. A method according to claim 46, wherein said circuit comprises individual circuits for individual neurons of said neural network.

48. A neural network constructed according to the method of any of claims 1-45.

49. A computer-readable media having a computer program stored therein, wherein said program, when executed on a general purpose computer for which the program is adapted, causes the computer to simulate a neural network according to claim 48.

50. A method of testing a neural network having a plurality of input lines, comprising:
providing an image;
projecting the image unto the plurality of input lines; and
analyzing an output of the neural network.

51. A method according to claim 50, wherein the image comprises a sequence of images.

52. A method of testing a neural network, comprising:
providing a neural network having a plurality of individual neurons, a plurality of input lines and a plurality of output lines; and
tracing the output of at least one neuron, which neuron is not directly connected to an output line.

53. A method of testing a neural network, comprising:
providing a neural network having a plurality of individual neurons, a plurality of input lines and a plurality of output lines; and
forcing an input to at least one neuron, which neuron is not directly connected to an input line.

54. A method of testing a neural network, comprising:
replacing at least a portion of the neural network with a stub; and

providing an input data set to the neural network; and

analyzing an output from the neural network, which output is responsive to the input data set.

55. A method of simulating a neuron which generates an output responsive to inputs the neuron receives from a plurality of input lines, comprising:

adding the inputs to at least one time-stamped buffer, responsive to the time at which the inputs are received by the neurons; and

generating an output responsive to the values in the buffer.

56. A method according to claim 55, wherein generating a response comprises generating a response if the value of the buffer at the current time is above a threshold.

57. A method according to claim 55, wherein generating a response comprises generating a response if a sum of values in the buffer from a range of times at and prior to the current time, is above a threshold.

58. A method according to claim 55, wherein adding comprises weighting the inputs with a weight responsive to their reception time.

59. A method according to claim 55, wherein generating an output comprises generating an output at a stochastic delay.

60. An electro-mechanical device, comprising:

a user input panel;

a mechanical portion; and

a neural network which controls the mechanical portion responsive to user inputs from the user input panel, wherein the electro-mechanical device does not utilize an algorithmic circuit for controlling the mechanical portion.

61. A device according to claim 60, wherein the neural network operates substantially in real-time.

62. Apparatus for generating a neural network, comprising:

storage for a definition file; and

a compiler which generates a neural network meeting the specifications in the definition file.

63. Apparatus according to claim 62, comprising a syntax analyzer which analyses the syntax of the definition file.

64. Apparatus according to claim 63, comprising an error-message generator which generates error messages responsive to the analysis of said syntax analyzer

65. A neural network comprising:

a first, non-trainable, plurality of interconnected neurons; and

a second, trainable, plurality of interconnected neurons.

66. A neural network according to claim 65, wherein the training of said second plurality of neural networks is controlled by the first plurality of neural networks.

67. A neural network according to claim 65, wherein at least one of said second first plurality of neurons and said second plurality of neurons comprises a neuron having at least one trainable connection and at least one non-trainable connection.

68. A neural network comprising a plurality of interconnected neurons, wherein said neurons do not perform a pattern matching function and are not trainable.

69. A neural network comprising a plurality of interconnected neurons, wherein most of the neurons are grouped into a plurality of functional groups and wherein most of the connections of neurons in each functional group are with neurons in the same functional group.

70. A neural network comprising a plurality of interconnected neurons which generate output signals, wherein the output signals are frequency encoded spike trains and wherein each neuron delays its output by a different amount.

71. A neural network according to claim 70, wherein each neuron stochastically delays its output.

72. A neural network, for controlling an industrial process comprising:

a plurality of interconnected neurons which generate output signals, wherein the output signals are frequency encoded spike trains;

a input device; and

an output device,

wherein said neural network generates controls said output device, responsive to said input device to achieve a desired effect on the industrial process.

73. A neural network comprising:

at least one neuron of a first type;

at least one neuron of a second type; and

at least one neuron of a third type.

74. A neural network according to claim 73, wherein said different types have different integrator functions.

75. A neural network according to claim 73, wherein said different types have different numbers of connections therefrom.

76. A method of training a neural network comprising:

providing a neural network having a design function and having a trainable portion; and

training the neural network, wherein said training is performed while the neural network performs its designated function.

77. A method of training a neural network comprising:

providing a neural network having a design function and having a trainable portion; and

training the neural network, wherein said training is controlled by the neural network, without external intervention.

78. A method of training a neural network comprising:

providing a neural network having a trainable portion comprising a plurality of neurons, each of said neurons having at least one non-synapse-weight parameter; and

training the neural network by modifying at least one of said at least one non-synapse-weight parameters of said neurons.

79. A method according to claim 78, wherein said training comprises modifying at least one synapse-weight parameter of said neurons.

80. A method according to claim 78, wherein said neural network comprises a heterogeneous neural network.

81. A method according to claim 78, wherein said training is responsive to a global state of said neural network.

82. A method according to claim 81, wherein said global state comprises an error function of said network.

83. A method according to claim 78, wherein said training is responsive to a local state of said neural network.

84. A method according to claim 83, wherein said training is responsive to a state of a neuron being trained.

85. A method according to claim 83, wherein said training is responsive to a state of a neuron which provides an input to a neuron being trained.

86. A method according to claim 83, wherein said training is responsive to a state of a neuron which receives an output from a neuron being trained.

87. A method according to claim 83, wherein said training is responsive to a correlation between a neuron being trained and a second neuron.

88. A method according to any of claims 78-87, wherein said training comprises back propagation.

89. A method according to any of claims 78-87, wherein said at least one non-synapse-weight parameter comprises a response threshold.

90. A method according to any of claims 78-87, wherein said at least one non-synapse-weight parameter comprises an output signal duration.

91. A method according to any of claims 78-87, wherein said at least one non-synapse-weight parameter comprises an output signal wave-form.

92. A method according to any of claims 78-87, wherein said at least one non-synapse-weight parameter comprises a synapse delay.

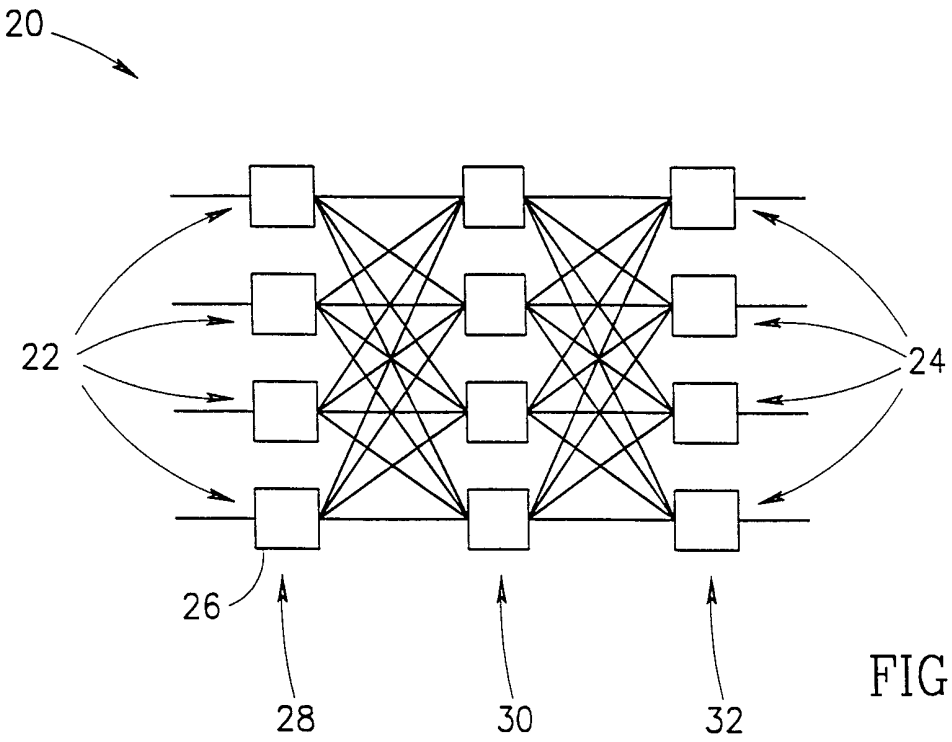

FIG.1

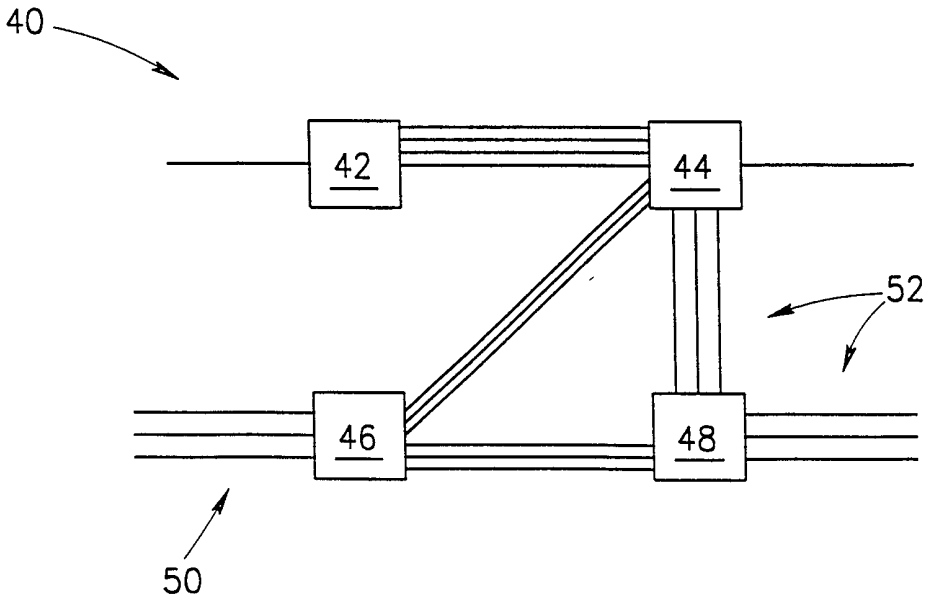

FIG.2

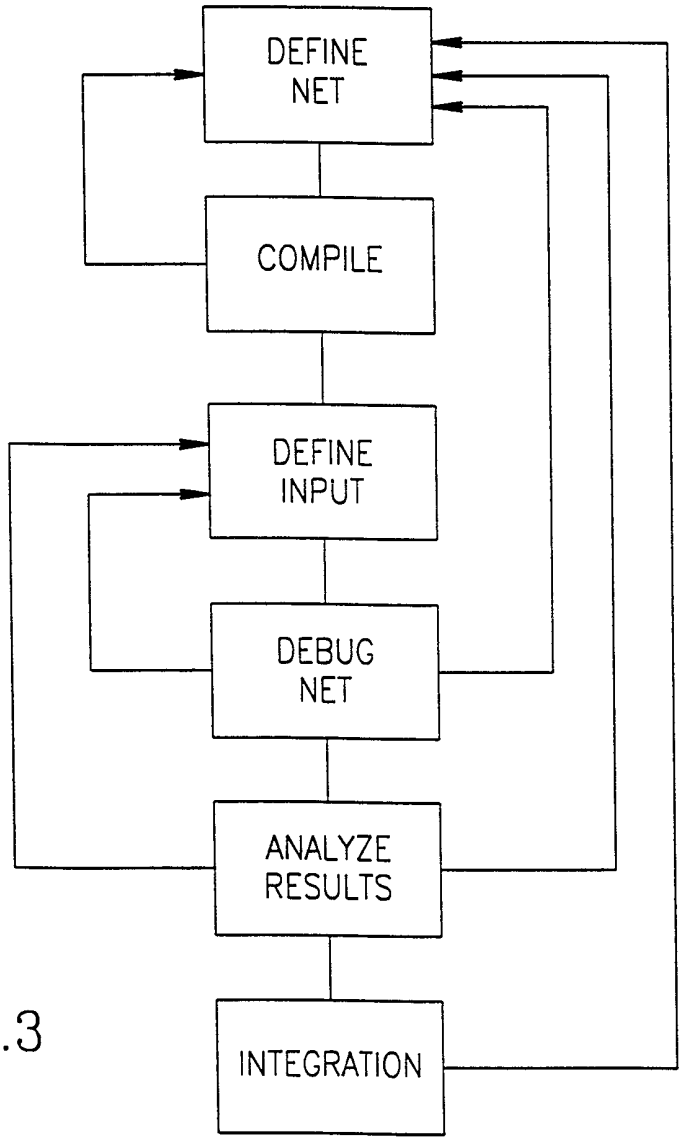
DEFINE NET
COMPILE
DEFINE INPUT
DEBUG NET
ANALYZE RESULTS
INTEGRATION

FIG.3

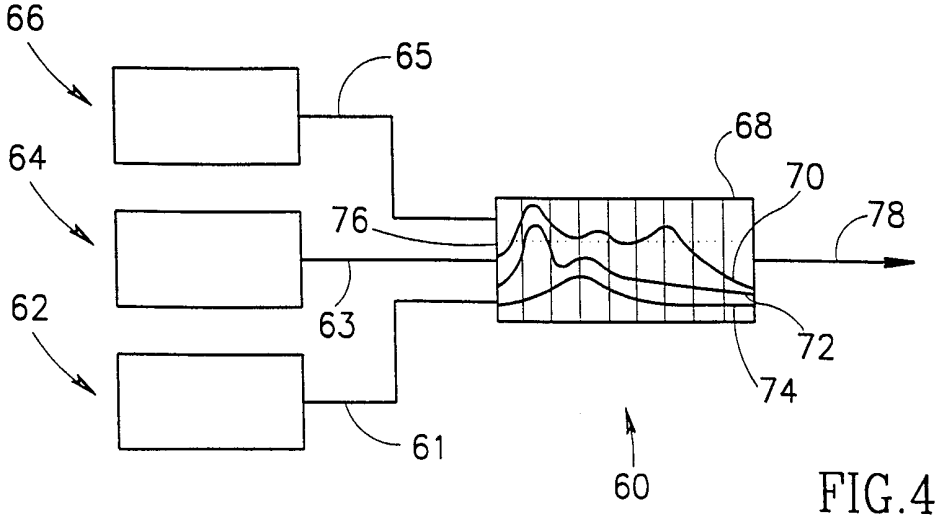
66
65
64
68
70
76
78
63
62
72
74
61
60

FIG.4

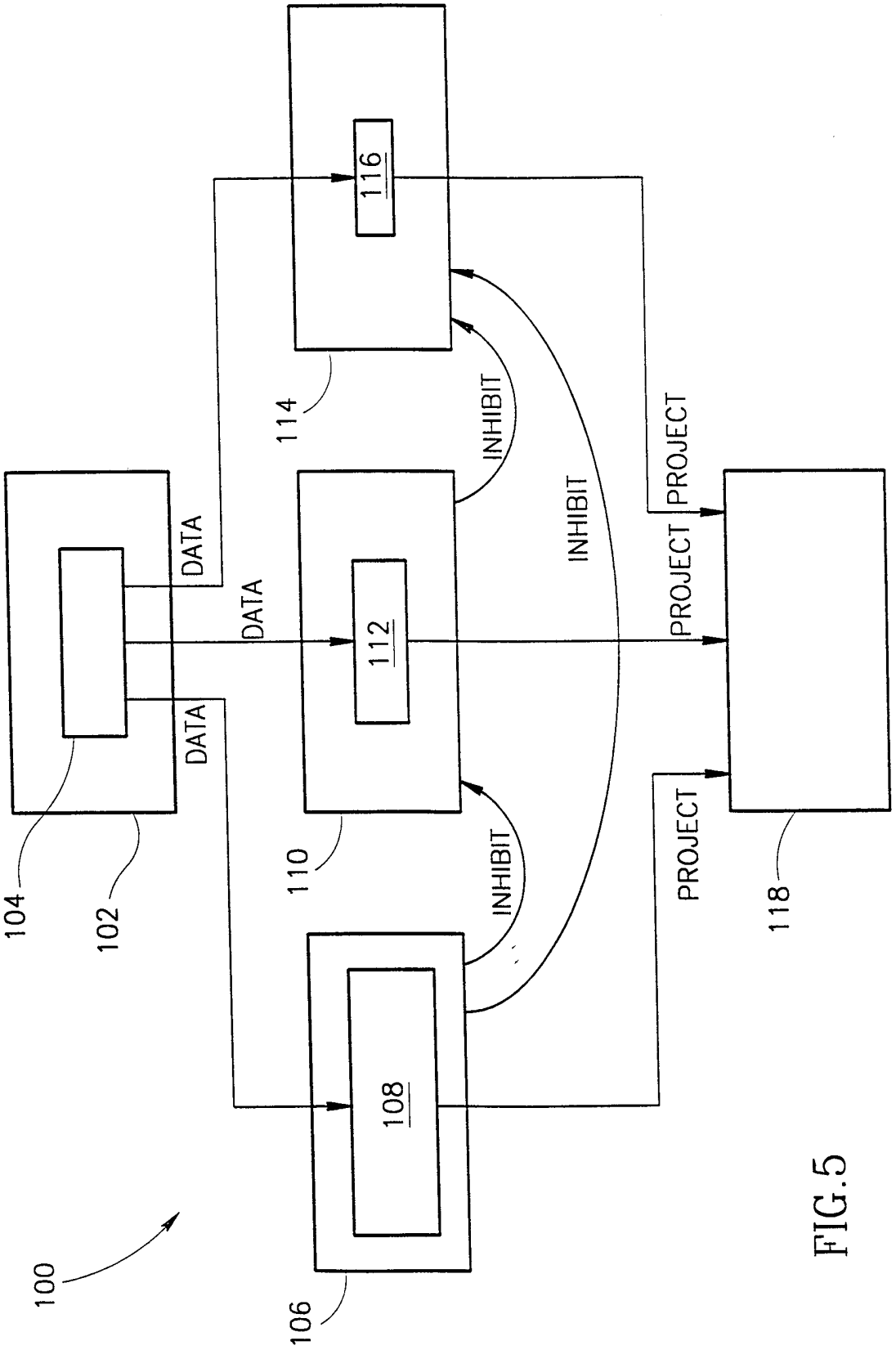
100
102
104
106
108
110
112
114
116
118
DATA
DATA
DATA
INHIBIT
INHIBIT
INHIBIT
PROJECT
PROJECT
PROJECT

FIG.5

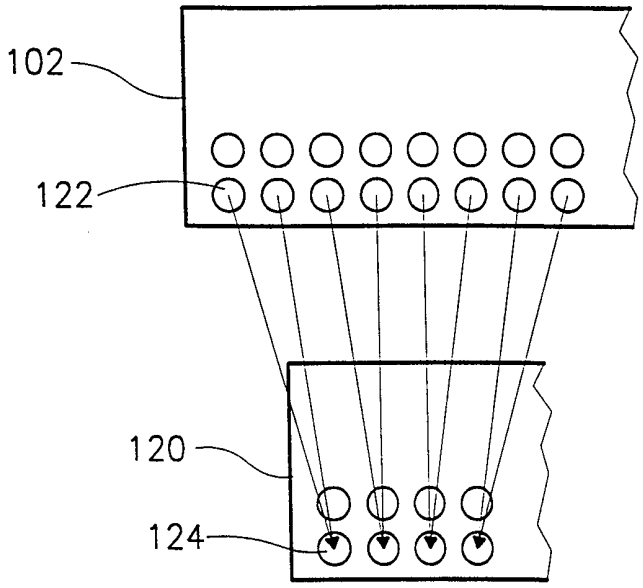

FIG.6

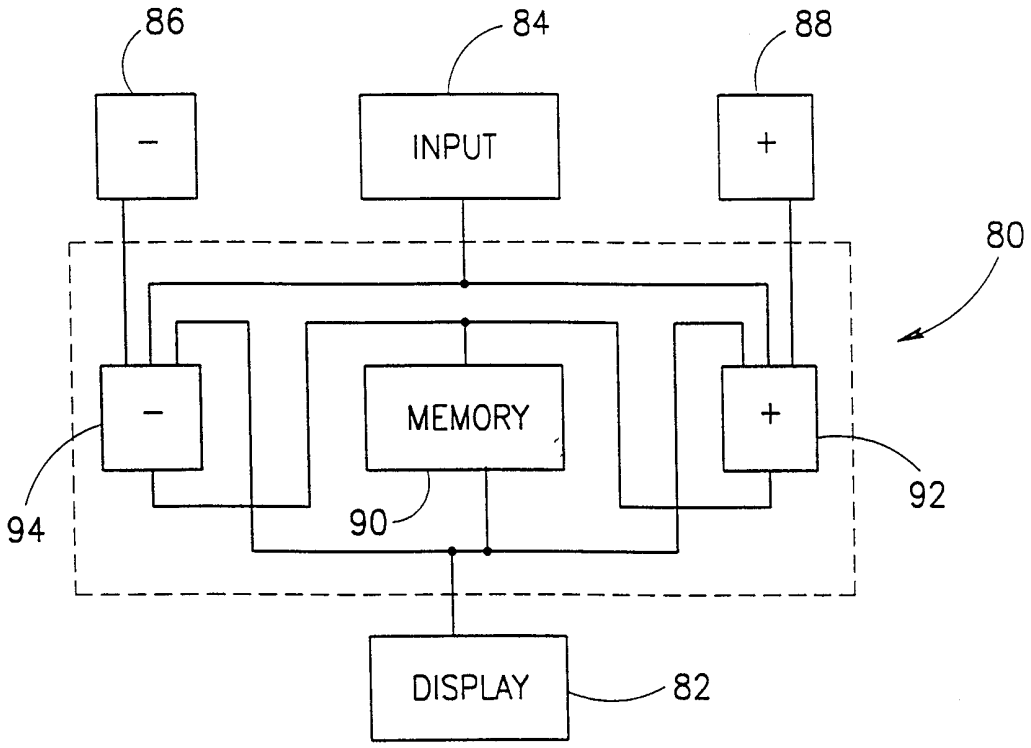

FIG.7