



(51) International Patent Classification:
G06N 3/08 (2006.01)

(21) International Application Number:
PCT/US2015/024807

(22) International Filing Date:
8 April 2015 (08.04.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/987,268 1 May 2014 (01.05.2014) US
14/315,531 26 June 2014 (26.06.2014) US

(71) Applicant: QUALCOMM INCORPORATED [US/US];
ATTN: International IP Administration, 5775 Morehouse
Drive, San Diego, California 92121-1714 (US).

(72) Inventors: JULIAN, David Jonathan; 5775 Morehouse
Drive, San Diego, California 92121-1714 (US). AN-
NAPUREDDY, Venkata Sreekanth Reddy; 5775 More-
house Drive, San Diego, California 92121-1714 (US).
PETERSON, Kristopher David; 5775 Morehouse Drive,
San Diego, California 92121-1714 (US).

(74) Agents: READ, Randol W. et al.; Patterson & Sheridan,
L.L.P., 24 Greenway Plaza, Suite 1600, Houston, Texas
77046-2472 (US).

(81) Designated States (unless otherwise indicated, for every
kind of national protection available): AE, AG, AL, AM,

AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY,
BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM,
DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT,
HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR,
KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG,
MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM,
PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC,
SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN,
TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every
kind of regional protection available): ARIPO (BW, GH,
GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ,
TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU,
TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE,
DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU,
LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK,
SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ,
GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

- as to applicant's entitlement to apply for and be granted a
patent (Rule 4.17(ii))
- as to the applicant's entitlement to claim the priority of the
earlier application (Rule 4.17(iii))

Published:

- without international search report and to be republished
upon receipt of that report (Rule 48.2(g))

(54) Title: TEMPORAL SPIKE ENCODING FOR TEMPORAL LEARNING

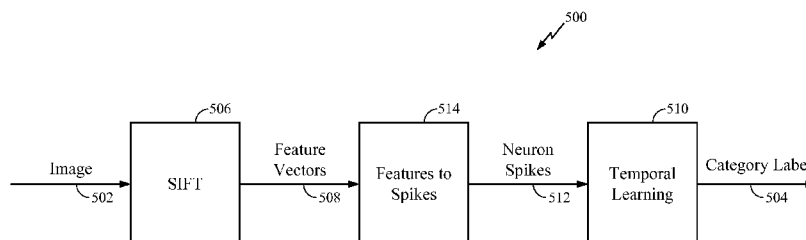


FIG. 5

(57) Abstract: Certain aspects of the present disclosure support methods and apparatus for temporal spike encoding for temporal learning in an artificial nervous system. The temporal spike encoding for temporal learning can comprise obtaining sensor data being input into the artificial nervous system, processing the sensor data to generate feature vectors, converting element values of the feature vectors into delays, and causing at least one artificial neuron of the artificial nervous system to spike at times based on the delays.



TEMPORAL SPIKE ENCODING FOR TEMPORAL LEARNING**CLAIM OF PRIORITY UNDER 35 U.S.C. § 119**

[0001] This application claims priority to U.S. Application Serial No. 14/315,531, filed June 26, 2014 which claims benefit of U.S. Provisional Patent Application Serial No. 61/987,268, filed May 1, 2014 and entitled “Temporal Spike Encoding for Temporal Learning”, both incorporated by reference in their entirety.

BACKGROUND**Field**

[0002] Certain aspects of the present disclosure generally relate to artificial nervous systems and, more particularly, to methods and apparatus for temporal spike encoding for temporal learning.

Background

[0003] An artificial neural network, which may comprise an interconnected group of artificial neurons (i.e., neural processing units), is a computational device or represents a method to be performed by a computational device. Artificial neural networks may have corresponding structure and/or function in biological neural networks. However, artificial neural networks may provide innovative and useful computational techniques for certain applications in which traditional computational techniques are cumbersome, impractical, or inadequate. Because artificial neural networks can infer a function from observations, such networks are particularly useful in applications where the complexity of the task or data makes the design of the function by conventional techniques burdensome.

[0004] One type of artificial neural network is the spiking neural network, which incorporates the concept of time into its operating model, as well as neuronal and synaptic state, thereby providing a rich set of behaviors from which computational function can emerge in the neural network. Spiking neural networks are based on the concept that neurons fire or “spike” at a particular time or times based on the state of the neuron, and that the time is important to neuron function. When a neuron fires, it generates a spike that travels to other neurons, which, in turn, may adjust their states

based on the time this spike is received. In other words, information may be encoded in the relative or absolute timing of spikes in the neural network.

SUMMARY

[0005] Certain aspects of the present disclosure provide a method for operating an artificial nervous system. The method generally includes obtaining sensor data being input into the artificial nervous system, processing the sensor data to generate feature vectors; converting element values of the feature vectors into delays, and causing at least one artificial neuron of the artificial nervous system to spike at times based on the delays.

[0006] Certain aspects of the present disclosure provide an apparatus for operating an artificial nervous system. The apparatus generally includes a sensor configured to obtain sensor data being input into the artificial nervous system, a first circuit configured to process the sensor data to generate feature vectors, a second circuit configured to convert element values of the feature vectors into delays, and a third circuit configured to cause at least one artificial neuron of the artificial nervous system to spike at times based on the delays.

[0007] Certain aspects of the present disclosure provide an apparatus for operating an artificial nervous system. The apparatus generally includes means for obtaining sensor data being input into the artificial nervous system, means for processing the sensor data to generate feature vectors, means for converting element values of the feature vectors into delays, and means for causing at least one artificial neuron of the artificial nervous system to spike at times based on the delays.

[0008] Certain aspects of the present disclosure provide a computer-readable medium having instructions executable by a computer stored thereon. The instructions are executable for obtaining sensor data being input into an artificial nervous system, processing the sensor data to generate feature vectors, converting element values of the feature vectors into delays, and causing at least one artificial neuron of the artificial nervous system to spike at times based on the delays.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] So that the manner in which the above-recited features of the present disclosure can be understood in detail, a more particular description, briefly summarized above, may be had by reference to aspects, some of which are illustrated in the appended drawings. It is to be noted, however, that the appended drawings illustrate only certain typical aspects of this disclosure and are therefore not to be considered limiting of its scope, for the description may admit to other equally effective aspects.

[0010] FIG. 1 illustrates an example network of neurons, in accordance with certain aspects of the present disclosure.

[0011] FIG. 2 illustrates an example processing unit (neuron) of a computational network (neural system or neural network), in accordance with certain aspects of the present disclosure.

[0012] FIG. 3 illustrates an example spike-timing dependent plasticity (STDP) curve, in accordance with certain aspects of the present disclosure.

[0013] FIG. 4 is an example graph of state for an artificial neuron, illustrating a positive regime and a negative regime for defining behavior of the neuron, in accordance with certain aspects of the present disclosure.

[0014] FIG. 5 illustrates an example block diagram of encoding Scale Invariant Feature Transform (SIFT) feature vectors as neuron spike times, in accordance with certain aspects of the present disclosure.

[0015] FIG. 6 illustrates an example of a parallel spike pattern for a feature vector, in accordance with certain aspects of the present disclosure.

[0016] FIG. 7 illustrates examples of parallel spike patterns with flipped mapping, in accordance with certain aspects of the present disclosure.

[0017] FIG. 8 illustrates an example graphical diagram of a parallel approach for encoding feature vectors into neuron spikes, in accordance with certain aspects of the present disclosure.

[0018] FIG. 9 illustrates an example graphical diagram of a serial approach for encoding feature vectors into neuron spikes, in accordance with certain aspects of the present disclosure.

[0019] FIG. 10 illustrates examples of a serial approach for encoding feature vectors into neuron spikes, in accordance with certain aspects of the present disclosure.

[0020] FIG. 11 illustrates an example variation of the serial approach for encoding feature vectors into neuron spikes, in accordance with certain aspects of the present disclosure.

[0021] FIG. 12 illustrates an example graphical diagram of a fixed weight clustering approach for encoding feature vectors into neuron spikes, in accordance with certain aspects of the present disclosure.

[0022] FIG. 13 illustrates a flow diagram of example operations for operating an artificial nervous system, in accordance with certain aspects of the present disclosure.

[0023] FIG. 13A illustrates example means capable of performing the operations shown in FIG. 13.

[0024] FIG. 14 illustrates an example implementation for operating an artificial nervous system using a general-purpose processor, in accordance with certain aspects of the present disclosure.

[0025] FIG. 15 illustrates an example implementation for operating an artificial nervous system where a memory may be interfaced with individual distributed processing units, in accordance with certain aspects of the present disclosure.

[0026] FIG. 16 illustrates an example implementation for operating an artificial nervous system based on distributed memories and distributed processing units, in accordance with certain aspects of the present disclosure.

[0027] FIG. 17 illustrates an example implementation of an artificial nervous system, in accordance with certain aspects of the present disclosure.

[0028] FIG. 18 illustrates a block diagram of an example hardware implementation for an artificial nervous system, in accordance with certain aspects of the present disclosure.

DETAILED DESCRIPTION

[0029] Various aspects of the disclosure are described more fully hereinafter with reference to the accompanying drawings. This disclosure may, however, be embodied in many different forms and should not be construed as limited to any specific structure or function presented throughout this disclosure. Rather, these aspects are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the disclosure to those skilled in the art. Based on the teachings herein one skilled in the art should appreciate that the scope of the disclosure is intended to cover any aspect of the disclosure disclosed herein, whether implemented independently of or combined with any other aspect of the disclosure. For example, an apparatus may be implemented or a method may be practiced using any number of the aspects set forth herein. In addition, the scope of the disclosure is intended to cover such an apparatus or method which is practiced using other structure, functionality, or structure and functionality in addition to or other than the various aspects of the disclosure set forth herein. It should be understood that any aspect of the disclosure disclosed herein may be embodied by one or more elements of a claim.

[0030] The word “exemplary” is used herein to mean “serving as an example, instance, or illustration.” Any aspect described herein as “exemplary” is not necessarily to be construed as preferred or advantageous over other aspects.

[0031] Although particular aspects are described herein, many variations and permutations of these aspects fall within the scope of the disclosure. Although some benefits and advantages of the preferred aspects are mentioned, the scope of the disclosure is not intended to be limited to particular benefits, uses or objectives. Rather, aspects of the disclosure are intended to be broadly applicable to different technologies, system configurations, networks and protocols, some of which are illustrated by way of example in the figures and in the following description of the preferred aspects. The detailed description and drawings are merely illustrative of the disclosure rather than

limiting, the scope of the disclosure being defined by the appended claims and equivalents thereof.

AN EXAMPLE NEURAL SYSTEM

[0032] FIG. 1 illustrates an example neural system 100 with multiple levels of neurons in accordance with certain aspects of the present disclosure. The neural system 100 may comprise a level of neurons 102 connected to another level of neurons 106 through a network of synaptic connections 104 (i.e., feed-forward connections). For simplicity, only two levels of neurons are illustrated in FIG. 1, although fewer or more levels of neurons may exist in a typical neural system. It should be noted that some of the neurons may connect to other neurons of the same layer through lateral connections. Furthermore, some of the neurons may connect back to a neuron of a previous layer through feedback connections.

[0033] As illustrated in FIG. 1, each neuron in the level 102 may receive an input signal 108 that may be generated by a plurality of neurons of a previous level (not shown in FIG. 1). The signal 108 may represent an input (e.g., an input current) to the level 102 neuron. Such inputs may be accumulated on the neuron membrane to charge a membrane potential. When the membrane potential reaches its threshold value, the neuron may fire and generate an output spike to be transferred to the next level of neurons (e.g., the level 106). Such behavior can be emulated or simulated in hardware and/or software, including analog and digital implementations.

[0034] In biological neurons, the output spike generated when a neuron fires is referred to as an action potential. This electrical signal is a relatively rapid, transient, all-or nothing nerve impulse, having an amplitude of roughly 100 mV and a duration of about 1 ms. In a particular aspect of a neural system having a series of connected neurons (e.g., the transfer of spikes from one level of neurons to another in FIG. 1), every action potential has basically the same amplitude and duration, and thus, the information in the signal is represented only by the frequency and number of spikes (or the time of spikes), not by the amplitude. The information carried by an action potential is determined by the spike, the neuron that spiked, and the time of the spike relative to one or more other spikes.

[0035] The transfer of spikes from one level of neurons to another may be achieved through the network of synaptic connections (or simply “synapses”) 104, as illustrated in FIG. 1. The synapses 104 may receive output signals (i.e., spikes) from the level 102 neurons (pre-synaptic neurons relative to the synapses 104). For certain aspects, these signals may be scaled according to adjustable synaptic weights $w_1^{(i,i+1)}, \dots, w_P^{(i,i+1)}$ (where P is a total number of synaptic connections between the neurons of levels 102 and 106). For other aspects, the synapses 104 may not apply any synaptic weights. Further, the (scaled) signals may be combined as an input signal of each neuron in the level 106 (post-synaptic neurons relative to the synapses 104). Every neuron in the level 106 may generate output spikes 110 based on the corresponding combined input signal. The output spikes 110 may be then transferred to another level of neurons using another network of synaptic connections (not shown in FIG. 1).

[0036] Biological synapses may be classified as either electrical or chemical. While electrical synapses are used primarily to send excitatory signals, chemical synapses can mediate either excitatory or inhibitory (hyperpolarizing) actions in postsynaptic neurons and can also serve to amplify neuronal signals. Excitatory signals typically depolarize the membrane potential (i.e., increase the membrane potential with respect to the resting potential). If enough excitatory signals are received within a certain period to depolarize the membrane potential above a threshold, an action potential occurs in the postsynaptic neuron. In contrast, inhibitory signals generally hyperpolarize (i.e., lower) the membrane potential. Inhibitory signals, if strong enough, can counteract the sum of excitatory signals and prevent the membrane potential from reaching threshold. In addition to counteracting synaptic excitation, synaptic inhibition can exert powerful control over spontaneously active neurons. A spontaneously active neuron refers to a neuron that spikes without further input, for example, due to its dynamics or feedback. By suppressing the spontaneous generation of action potentials in these neurons, synaptic inhibition can shape the pattern of firing in a neuron, which is generally referred to as sculpturing. The various synapses 104 may act as any combination of excitatory or inhibitory synapses, depending on the behavior desired.

[0037] The neural system 100 may be emulated by a general purpose processor, a digital signal processor (DSP), an application specific integrated circuit (ASIC), a field programmable gate array (FPGA) or other programmable logic device (PLD), discrete

gate or transistor logic, discrete hardware components, a software module executed by a processor, or any combination thereof. The neural system 100 may be utilized in a large range of applications, such as image and pattern recognition, machine learning, motor control, and the like. Each neuron in the neural system 100 may be implemented as a neuron circuit. The neuron membrane charged to the threshold value initiating the output spike may be implemented, for example, as a capacitor that integrates an electrical current flowing through it.

[0038] In an aspect, the capacitor may be eliminated as the electrical current integrating device of the neuron circuit, and a smaller memristor element may be used in its place. This approach may be applied in neuron circuits, as well as in various other applications where bulky capacitors are utilized as electrical current integrators. In addition, each of the synapses 104 may be implemented based on a memristor element, wherein synaptic weight changes may relate to changes of the memristor resistance. With nanometer feature-sized memristors, the area of neuron circuit and synapses may be substantially reduced, which may make implementation of a very large-scale neural system hardware implementation practical.

[0039] Functionality of a neural processor that emulates the neural system 100 may depend on weights of synaptic connections, which may control strengths of connections between neurons. The synaptic weights may be stored in a non-volatile memory in order to preserve functionality of the processor after being powered down. In an aspect, the synaptic weight memory may be implemented on a separate external chip from the main neural processor chip. The synaptic weight memory may be packaged separately from the neural processor chip as a replaceable memory card. This may provide diverse functionalities to the neural processor, wherein a particular functionality may be based on synaptic weights stored in a memory card currently attached to the neural processor.

[0040] FIG. 2 illustrates an example 200 of a processing unit (e.g., an artificial neuron 202) of a computational network (e.g., a neural system or a neural network) in accordance with certain aspects of the present disclosure. For example, the neuron 202 may correspond to any of the neurons of levels 102 and 106 from FIG. 1. The neuron 202 may receive multiple input signals 204₁-204_N (x_1 - x_N), which may be signals external to the neural system, or signals generated by other neurons of the same neural

system, or both. The input signal may be a current or a voltage, real-valued or complex-valued. The input signal may comprise a numerical value with a fixed-point or a floating-point representation. These input signals may be delivered to the neuron 202 through synaptic connections that scale the signals according to adjustable synaptic weights 206₁-206_N ($w_1 - w_N$), where N may be a total number of input connections of the neuron 202.

[0041] The neuron 202 may combine the scaled input signals and use the combined scaled inputs to generate an output signal 208 (i.e., a signal y). The output signal 208 may be a current, or a voltage, real-valued or complex-valued. The output signal may comprise a numerical value with a fixed-point or a floating-point representation. The output signal 208 may be then transferred as an input signal to other neurons of the same neural system, or as an input signal to the same neuron 202, or as an output of the neural system.

[0042] The processing unit (neuron 202) may be emulated by an electrical circuit, and its input and output connections may be emulated by wires with synaptic circuits. The processing unit, its input and output connections may also be emulated by a software code. The processing unit may also be emulated by an electric circuit, whereas its input and output connections may be emulated by a software code. In an aspect, the processing unit in the computational network may comprise an analog electrical circuit. In another aspect, the processing unit may comprise a digital electrical circuit. In yet another aspect, the processing unit may comprise a mixed-signal electrical circuit with both analog and digital components. The computational network may comprise processing units in any of the aforementioned forms. The computational network (neural system or neural network) using such processing units may be utilized in a large range of applications, such as image and pattern recognition, machine learning, motor control, and the like.

[0043] During the course of training a neural network, synaptic weights (e.g., the weights $w_1^{(i,i+1)}, \dots, w_p^{(i,i+1)}$ from FIG. 1 and/or the weights 206₁-206_N from FIG. 2) may be initialized with random values and increased or decreased according to a learning rule. Some examples of the learning rule are the spike-timing-dependent plasticity (STDP) learning rule, the Hebb rule, the Oja rule, the Bienenstock-Copper-Munro (BCM)

rule, etc. Very often, the weights may settle to one of two values (i.e., a bimodal distribution of weights). This effect can be utilized to reduce the number of bits per synaptic weight, increase the speed of reading and writing from/to a memory storing the synaptic weights, and to reduce power consumption of the synaptic memory.

Synapse Type

[0044] In hardware and software models of neural networks, processing of synapse related functions can be based on synaptic type. Synapse types may comprise non-plastic synapses (no changes of weight and delay), plastic synapses (weight may change), structural delay plastic synapses (weight and delay may change), fully plastic synapses (weight, delay and connectivity may change), and variations thereupon (e.g., delay may change, but no change in weight or connectivity). The advantage of this is that processing can be subdivided. For example, non-plastic synapses may not require plasticity functions to be executed (or waiting for such functions to complete). Similarly, delay and weight plasticity may be subdivided into operations that may operate in together or separately, in sequence or in parallel. Different types of synapses may have different lookup tables or formulas and parameters for each of the different plasticity types that apply. Thus, the methods would access the relevant tables for the synapse's type.

[0045] There are further implications of the fact that spike-timing dependent structural plasticity may be executed independently of synaptic plasticity. Structural plasticity may be executed even if there is no change to weight magnitude (e.g., if the weight has reached a minimum or maximum value, or it is not changed due to some other reason) since structural plasticity (i.e., an amount of delay change) may be a direct function of pre-post spike time difference. Alternatively, it may be set as a function of the weight change amount or based on conditions relating to bounds of the weights or weight changes. For example, a synaptic delay may change only when a weight change occurs or if weights reach zero, but not if the weights are maxed out. However, it can be advantageous to have independent functions so that these processes can be parallelized reducing the number and overlap of memory accesses.

DETERMINATION OF SYNAPTIC PLASTICITY

[0046] Neuroplasticity (or simply “plasticity”) is the capacity of neurons and neural networks in the brain to change their synaptic connections and behavior in response to new information, sensory stimulation, development, damage, or dysfunction. Plasticity is important to learning and memory in biology, as well as to computational neuroscience and neural networks. Various forms of plasticity have been studied, such as synaptic plasticity (e.g., according to the Hebbian theory), spike-timing-dependent plasticity (STDP), non-synaptic plasticity, activity-dependent plasticity, structural plasticity, and homeostatic plasticity.

[0047] STDP is a learning process that adjusts the strength of synaptic connections between neurons, such as those in the brain. The connection strengths are adjusted based on the relative timing of a particular neuron’s output and received input spikes (i.e., action potentials). Under the STDP process, long-term potentiation (LTP) may occur if an input spike to a certain neuron tends, on average, to occur immediately before that neuron's output spike. Then, that particular input is made somewhat stronger. In contrast, long-term depression (LTD) may occur if an input spike tends, on average, to occur immediately after an output spike. Then, that particular input is made somewhat weaker, hence the name "spike-timing-dependent plasticity." Consequently, inputs that might be the cause of the post-synaptic neuron's excitation are made even more likely to contribute in the future, whereas inputs that are not the cause of the post-synaptic spike are made less likely to contribute in the future. The process continues until a subset of the initial set of connections remains, while the influence of all others is reduced to zero or near zero.

[0048] Since a neuron generally produces an output spike when many of its inputs occur within a brief period (i.e., being sufficiently cumulative to cause the output,), the subset of inputs that typically remains includes those that tended to be correlated in time. In addition, since the inputs that occur before the output spike are strengthened, the inputs that provide the earliest sufficiently cumulative indication of correlation will eventually become the final input to the neuron.

[0049] The STDP learning rule may effectively adapt a synaptic weight of a synapse connecting a pre-synaptic neuron to a post-synaptic neuron as a function of time

difference between spike time t_{pre} of the pre-synaptic neuron and spike time t_{post} of the post-synaptic neuron (i.e., $t = t_{post} - t_{pre}$). A typical formulation of the STDP is to increase the synaptic weight (i.e., potentiate the synapse) if the time difference is positive (the pre-synaptic neuron fires before the post-synaptic neuron), and decrease the synaptic weight (i.e., depress the synapse) if the time difference is negative (the post-synaptic neuron fires before the pre-synaptic neuron).

[0050] In the STDP process, a change of the synaptic weight over time may be typically achieved using an exponential decay, as given by,

$$\Delta w(t) = \begin{cases} a_+ e^{-t/k_+} + \mu, & t > 0 \\ a_- e^{t/k_-}, & t < 0 \end{cases}, \quad (1)$$

where k_+ and k_- are time constants for positive and negative time difference, respectively, a_+ and a_- are corresponding scaling magnitudes, and μ is an offset that may be applied to the positive time difference and/or the negative time difference.

[0051] FIG. 3 illustrates an example graph 300 of a synaptic weight change as a function of relative timing of pre-synaptic and post-synaptic spikes in accordance with STDP. If a pre-synaptic neuron fires before a post-synaptic neuron, then a corresponding synaptic weight may be increased, as illustrated in a portion 302 of the graph 300. This weight increase can be referred to as an LTP of the synapse. It can be observed from the graph portion 302 that the amount of LTP may decrease roughly exponentially as a function of the difference between pre-synaptic and post-synaptic spike times. The reverse order of firing may reduce the synaptic weight, as illustrated in a portion 304 of the graph 300, causing an LTD of the synapse.

[0052] As illustrated in the graph 300 in FIG. 3, a negative offset μ may be applied to the LTP (causal) portion 302 of the STDP graph. A point of cross-over 306 of the x-axis ($y=0$) may be configured to coincide with the maximum time lag for considering correlation for causal inputs from layer i-1 (presynaptic layer). In the case of a frame-based input (i.e., an input is in the form of a frame of a particular duration comprising spikes or pulses), the offset value μ can be computed to reflect the frame boundary. A first input spike (pulse) in the frame may be considered to decay over time either as

modeled by a post-synaptic potential directly or in terms of the effect on neural state. If a second input spike (pulse) in the frame is considered correlated or relevant of a particular time frame, then the relevant times before and after the frame may be separated at that time frame boundary and treated differently in plasticity terms by offsetting one or more parts of the STDP curve such that the value in the relevant times may be different (e.g., negative for greater than one frame and positive for less than one frame). For example, the negative offset μ may be set to offset LTP such that the curve actually goes below zero at a pre-post time greater than the frame time and it is thus part of LTD instead of LTP.

NEURON MODELS AND OPERATION

[0053] There are some general principles for designing a useful spiking neuron model. A good neuron model may have rich potential behavior in terms of two computational regimes: coincidence detection and functional computation. Moreover, a good neuron model should have two elements to allow temporal coding: arrival time of inputs affects output time and coincidence detection can have a narrow time window. Finally, to be computationally attractive, a good neuron model may have a closed-form solution in continuous time and have stable behavior including near attractors and saddle points. In other words, a useful neuron model is one that is practical and that can be used to model rich, realistic and biologically-consistent behaviors, as well as be used to both engineer and reverse engineer neural circuits.

[0054] A neuron model may depend on events, such as an input arrival, output spike or other event whether internal or external. To achieve a rich behavioral repertoire, a state machine that can exhibit complex behaviors may be desired. If the occurrence of an event itself, separate from the input contribution (if any) can influence the state machine and constrain dynamics subsequent to the event, then the future state of the system is not only a function of a state and input, but rather a function of a state, event, and input.

[0055] In an aspect, a neuron n may be modeled as a spiking leaky-integrate-and-fire neuron with a membrane voltage $v_n(t)$ governed by the following dynamics,

$$\frac{dv_n(t)}{dt} = \alpha v_n(t) + \beta \sum_m w_{m,n} y_m(t - \Delta t_{m,n}), \quad (2)$$

where α and β are parameters, $w_{m,n}$ is a synaptic weight for the synapse connecting a pre-synaptic neuron m to a post-synaptic neuron n , and $y_m(t)$ is the spiking output of the neuron m that may be delayed by dendritic or axonal delay according to $\Delta t_{m,n}$ until arrival at the neuron n 's soma.

[0056] It should be noted that there is a delay from the time when sufficient input to a post-synaptic neuron is established until the time when the post-synaptic neuron actually fires. In a dynamic spiking neuron model, such as Izhikevich's simple model, a time delay may be incurred if there is a difference between a depolarization threshold v_t and a peak spike voltage v_{peak} . For example, in the simple model, neuron soma dynamics can be governed by the pair of differential equations for voltage and recovery, i.e.,

$$\frac{dv}{dt} = (k(v - v_t)(v - v_r) - u + I) / C, \quad (3)$$

$$\frac{du}{dt} = a(b(v - v_r) - u). \quad (4)$$

where v is a membrane potential, u is a membrane recovery variable, k is a parameter that describes time scale of the membrane potential v , a is a parameter that describes time scale of the recovery variable u , b is a parameter that describes sensitivity of the recovery variable u to the sub-threshold fluctuations of the membrane potential v , v_r is a membrane resting potential, I is a synaptic current, and C is a membrane's capacitance. In accordance with this model, the neuron is defined to spike when $v > v_{peak}$.

Hunzinger Cold Model

[0057] The Hunzinger Cold neuron model is a minimal dual-regime spiking linear dynamical model that can reproduce a rich variety of neural behaviors. The model's one- or two-dimensional linear dynamics can have two regimes, wherein the time constant (and coupling) can depend on the regime. In the sub-threshold regime, the time constant, negative by convention, represents leaky channel dynamics generally acting to return a cell to rest in biologically-consistent linear fashion. The time constant

in the supra-threshold regime, positive by convention, reflects anti-leaky channel dynamics generally driving a cell to spike while incurring latency in spike-generation.

[0058] As illustrated in FIG. 4, the dynamics of the model may be divided into two (or more) regimes. These regimes may be called the negative regime 402 (also interchangeably referred to as the leaky-integrate-and-fire (LIF) regime, not to be confused with the LIF neuron model) and the positive regime 404 (also interchangeably referred to as the anti-leaky-integrate-and-fire (ALIF) regime, not to be confused with the ALIF neuron model). In the negative regime 402, the state tends toward rest (v_-) at the time of a future event. In this negative regime, the model generally exhibits temporal input detection properties and other sub-threshold behavior. In the positive regime 404, the state tends toward a spiking event (v_s). In this positive regime, the model exhibits computational properties, such as incurring a latency to spike depending on subsequent input events. Formulation of dynamics in terms of events and separation of the dynamics into these two regimes are fundamental characteristics of the model.

[0059] Linear dual-regime bi-dimensional dynamics (for states v and u) may be defined by convention as,

$$\tau_\rho \frac{dv}{dt} = v + q_\rho \quad (5)$$

$$-\tau_u \frac{du}{dt} = u + r \quad (6)$$

where q_ρ and r are the linear transformation variables for coupling.

[0060] The symbol ρ is used herein to denote the dynamics regime with the convention to replace the symbol ρ with the sign “-” or “+” for the negative and positive regimes, respectively, when discussing or expressing a relation for a specific regime.

[0061] The model state is defined by a membrane potential (voltage) v and recovery current u . In basic form, the regime is essentially determined by the model state. There are subtle, but important aspects of the precise and general definition, but for the

moment, consider the model to be in the positive regime 404 if the voltage v is above a threshold (v_+) and otherwise in the negative regime 402.

[0062] The regime-dependent time constants include τ_- which is the negative regime time constant, and τ_+ which is the positive regime time constant. The recovery current time constant τ_u is typically independent of regime. For convenience, the negative regime time constant τ_- is typically specified as a negative quantity to reflect decay so that the same expression for voltage evolution may be used as for the positive regime in which the exponent and τ_+ will generally be positive, as will be τ_u .

[0063] The dynamics of the two state elements may be coupled at events by transformations offsetting the states from their null-clines, where the transformation variables are

$$q_\rho = -\tau_\rho \beta u - v_\rho \quad (7)$$

$$r = \delta(v + \varepsilon) \quad (8)$$

where δ , ε , β and v_- , v_+ are parameters. The two values for v_ρ are the base for reference voltages for the two regimes. The parameter v_- is the base voltage for the negative regime, and the membrane potential will generally decay toward v_- in the negative regime. The parameter v_+ is the base voltage for the positive regime, and the membrane potential will generally tend away from v_+ in the positive regime.

[0064] The null-clines for v and u are given by the negative of the transformation variables q_ρ and r , respectively. The parameter δ is a scale factor controlling the slope of the u null-cline. The parameter ε is typically set equal to $-v_-$. The parameter β is a resistance value controlling the slope of the v null-clines in both regimes. The τ_ρ time-constant parameters control not only the exponential decays, but also the null-cline slopes in each regime separately.

[0065] The model is defined to spike when the voltage v reaches a value v_s . Subsequently, the state is typically reset at a reset event (which technically may be one and the same as the spike event):

$$v = \hat{v}_- \quad (9)$$

$$u = u + \Delta u \quad (10)$$

where $\hat{v}_-$ and Δu are parameters. The reset voltage $\hat{v}_-$ is typically set to v_- .

[0066] By a principle of momentary coupling, a closed-form solution is possible not only for state (and with a single exponential term), but also for the time required to reach a particular state. The closed-form state solutions are

$$v(t + \Delta t) = (v(t) + q_\rho) e^{\frac{\Delta t}{\tau_\rho}} - q_\rho \quad (11)$$

$$u(t + \Delta t) = (u(t) + r) e^{\frac{\Delta t}{\tau_u}} - r \quad (12)$$

[0067] Therefore, the model state may be updated only upon events, such as upon an input (pre-synaptic spike) or output (post-synaptic spike). Operations may also be performed at any particular time (whether or not there is input or output).

[0068] Moreover, by the momentary coupling principle, the time of a post-synaptic spike may be anticipated so the time to reach a particular state may be determined in advance without iterative techniques or Numerical Methods (e.g., the Euler numerical method). Given a prior voltage state v_0 , the time delay until voltage state v_f is reached is given by

$$\Delta t = \tau_\rho \log \frac{v_f + q_\rho}{v_0 + q_\rho} \quad (13)$$

[0069] If a spike is defined as occurring at the time the voltage state v reaches v_s , then the closed-form solution for the amount of time, or relative delay, until a spike occurs as measured from the time that the voltage is at a given state v is

$$\Delta t_s = \begin{cases} \tau_+ \log \frac{v_s + q_+}{v + q_+} & \text{if } v > \hat{v}_+ \\ \infty & \text{otherwise} \end{cases} \quad (14)$$

where $\hat{v}_+$ is typically set to parameter v_+ , although other variations may be possible.

[0070] The above definitions of the model dynamics depend on whether the model is in the positive or negative regime. As mentioned, the coupling and the regime ρ may be computed upon events. For purposes of state propagation, the regime and coupling (transformation) variables may be defined based on the state at the time of the last (prior) event. For purposes of subsequently anticipating spike output time, the regime and coupling variable may be defined based on the state at the time of the next (current) event.

[0071] There are several possible implementations of the Cold model, and executing the simulation, emulation or model in time. This includes, for example, event-update, step-event update, and step-update modes. An event update is an update where states are updated based on events or “event update” (at particular moments). A step update is an update when the model is updated at intervals (e.g., 1ms). This does not necessarily require iterative methods or Numerical methods. An event-based implementation is also possible at a limited time resolution in a step-based simulator by only updating the model if an event occurs at or between steps or by “step-event” update.

NEURAL CODING

[0072] A useful neural network model, such as one composed of the artificial neurons 102, 106 of FIG. 1, may encode information via any of various suitable neural coding schemes, such as coincidence coding, temporal coding or rate coding. In coincidence coding, information is encoded in the coincidence (or temporal proximity) of action potentials (spiking activity) of a neuron population. In temporal coding, a neuron encodes information through the precise timing of action potentials (i.e., spikes) whether in absolute time or relative time. Information may thus be encoded in the relative timing of spikes among a population of neurons. In contrast, rate coding involves coding the neural information in the firing rate or population firing rate.

[0073] If a neuron model can perform temporal coding, then it can also perform rate coding (since rate is just a function of timing or inter-spike intervals). To provide for temporal coding, a good neuron model should have two elements: (1) arrival time of inputs affects output time; and (2) coincidence detection can have a narrow time window. Connection delays provide one means to expand coincidence detection to

temporal pattern decoding because by appropriately delaying elements of a temporal pattern, the elements may be brought into timing coincidence.

Arrival time

[0074] In a good neuron model, the time of arrival of an input should have an effect on the time of output. A synaptic input—whether a Dirac delta function or a shaped post-synaptic potential (PSP), whether excitatory (EPSP) or inhibitory (IPSP)—has a time of arrival (e.g., the time of the delta function or the start or peak of a step or other input function), which may be referred to as the input time. A neuron output (i.e., a spike) has a time of occurrence (wherever it is measured, e.g., at the soma, at a point along the axon, or at an end of the axon), which may be referred to as the output time. That output time may be the time of the peak of the spike, the start of the spike, or any other time in relation to the output waveform. The overarching principle is that the output time depends on the input time.

[0075] One might at first glance think that all neuron models conform to this principle, but this is generally not true. For example, rate-based models do not have this feature. Many spiking models also do not generally conform. A leaky-integrate-and-fire (LIF) model does not fire any faster if there are extra inputs (beyond threshold). Moreover, models that might conform if modeled at very high timing resolution often will not conform when timing resolution is limited, such as to 1 ms steps.

Inputs

[0076] An input to a neuron model may include Dirac delta functions, such as inputs as currents, or conductance-based inputs. In the latter case, the contribution to a neuron state may be continuous or state-dependent.

ENCODING OF FEATURE VECTORS AS NEURON SPIKE TIMES

[0077] Certain aspects of the present disclosure specifies methods and apparatus for encoding Scale Invariant Feature Transform (SIFT) feature vectors as neuron spike times for temporal learning of categories in an artificial nervous system, such as the artificial nervous system 100 from FIG. 1. More generally, certain aspects of the present disclosure specify methods and apparatus for encoding sensor or non-spike data

as neuron spike times. In addition, aspects of the present disclosure present methods and apparatus for efficient encoding of information in order to facilitate implementation on hardware of a given size/capacity and a given processing time.

[0078] FIG. 5 illustrates an example block diagram 500 of encoding SIFT feature vectors as neuron spike times, in accordance with certain aspects of the present disclosure. The goal can be to take an image (or, more generally, sensor data) 502 and assign a category label 504 to an object in the image 502, such as car, plane, food, or poison. The front-end processing may utilize SIFT transform 506 to obtain SIFT feature vectors 508 from the image 502. The back-end processing may utilize the neural processing temporal learning 510 for learning and determining category labels based on training images.

[0079] There is a need in the art for a method to convert the feature vectors (e.g., the SIFT feature vectors 508 from FIG. 5) from SIFT format to a spike encoded format (e.g., to neuron spikes 512 in FIG. 5) that a temporal learning neural engine of an artificial nervous system can process. As illustrated in FIG. 5, a features-to-spikes block engine 514 may be configured to convert the feature vectors 508 into the neuron spikes 512. Certain aspects of the present disclosure support several different approaches for implementation of the features-to-spikes block 514. More generally, aspects of the present disclosure provide solutions to convert other sensor or traditionally processed data to a spike-encoded format for neural networks.

TEMPORAL SPIKE ENCODING SOLUTIONS

[0080] In an aspect of the present disclosure, the temporal learning engine (e.g., the engine 510 from FIG. 5) may be able to learn and classify recurring temporal spike patterns. Further, some embodiments of the temporal learning engine may put stronger weights on earlier spike times in a spike pattern. The goal is to convert the neuron spikes into a recurring spike pattern that is close in a temporal metric for the same or similar objects, and potentially to encode more important information earlier.

[0081] Certain aspects of the present disclosure provide several different general approaches for encoding feature vectors into neuron spikes, such as: the parallel approach, the serial approach, the fixed weight clustering approach, and the K-means

histogram approach. In the case of parallel encoding, SIFT transform may convert an image into feature vectors, and top feature vectors may be then selected. In an aspect of the present disclosure, each feature vector element may be assigned to one spiking neuron, wherein an element magnitude may be mapped to a spike delay. In the case of serial encoding, SIFT transform may convert an image into feature vectors, and top feature vectors may be then selected while a single spiking neuron may be associated to more than one feature vector elements, wherein a set of element magnitudes may be mapped to a set of spike delays. In the case of fixed weight clustering approach, the feature vectors may be combined into a large common cluster space of cluster neurons to be utilized in temporal learning. In the case of K-means histogram approach, the feature vectors may be clustered into different cluster points, a histogram of the number of feature vectors near each cluster point may be created, and then the histograms may be encoded into spike times.

SIFT Feature Vectors

[0082] The SIFT feature vectors may encode interesting points in an image, where interesting points may be based, for example, on strong gradients in x, y, and Gaussian blur dimensions of the image. The number of elements in the feature vector may vary based on model parameters. As an example, each feature vector may have 128 dimensions (elements) with 7-bit values. In an aspect, the feature vector can comprise a series of gradient histograms. The 128 dimensions may comprise 16 sections of 8 values ($16 \times 8 = 128$), where these 16 sections may represent a 4×4 spatial tile around the located feature vector center and the 8 values represent 8 directions ($0, \pi/4, \dots, 7\pi/4$). In an aspect, the feature vector may also have a corresponding (x, y) location and sigma blur.

[0083] Feature vector values may be robust to rotation and scale variations. Ordering of feature vectors may not be robust and may easily change based on transformations such as rotation, yaw, and others. Many algorithms based on SIFT may utilize “bag of words” matching and ignore spatial locations, or use spatial locations as a later consistency check phase.

Parallel Encoding Approach

[0084] For parallel encoding of SIFT feature vectors into neuron spikes, each feature vector of length 128 elements may be converted into 128 artificial neurons, with each neuron spiking at an offset from a periodic reference time based on the top (most significant) k bits of the element. As an example, the top four bits may be used for mapping a feature vector element into one of 16 delay values. FIG. 6 illustrates an example 600 of a parallel spike pattern for a feature vector with spike delay from a reference time on the x-axis and node ID from 0 to 127 on the y-axis.

[0085] This mapping can be done in multiple ways, including flipping the mapping so that larger feature vector values have earlier spike times. For example, if the four most significant bits (MSBs) of the feature vector element values are used to determine a delay, then the delay may be 16-value of top four MSBs. Three mapping examples are illustrated in FIG. 7, wherein an example 700 in FIG. 7 corresponds to flipping of the mapping 600 from FIG. 6.

[0086] One advantage of the flipped mapping can be that the SIFT feature vectors have more information encoded in their larger values, and the flipping illustrated in FIG. 7 places the larger values earlier. Additionally, since small SIFT feature vector element values are common, they may be encoded by not spiking for the smallest value, or not spiking for values less than a threshold. For example, all the values that have the four MSBs equal to zero may not generate a spike.

[0087] Additional transformations (mappings) may be performed, such as taking delay equals to $16 - \log(\text{element value})$, or delay equals to $16/\text{element value}$ in order to assign non-linear mapping to delay and provide different amounts of temporal spacing for different areas of the pattern to increase and/or decrease the distance for the temporal code to learn and discriminate. Further, other transformations, such as the principle component analysis (PCA) transformation, may be used to reduce the feature vector dimension. Then, the resulting element values may be encoded into delays in the manner described above.

[0088] Since the SIFT feature vector ordering is not invariant, one approach for the temporal learning architecture to learn the set of feature vectors and process a set of feature vectors from an image can be to take the top ranked feature vectors (e.g., 500)

from an image. Aspects of the present disclosure support a number of different metrics (criteria) for ranking the feature vectors.

[0089] In an aspect of the present disclosure, a neural network 800 illustrated in FIG. 8 may be utilized for parallel encoding approach, potentially with weight sharing between the different first layer parallel feature vectors 802 to intermediate features 804. As illustrated in FIG. 8, multiple layers of temporal and/or rate learning may be utilized, with 128 neurons per feature vector 802 (one neuron per each feature vector element), one neuron per intermediate feature 804 (e.g., a total of 100 neurons for intermediate features), and one neuron per category label 806.

Serial Encoding Approach

[0090] In the case of parallel encoding approach illustrated in FIG. 8, each input neuron represents one SIFT feature vector element and spiked once per image. In the case of serial encoding approach, more than one feature vector element may be encoded per neuron by having each neuron spiking more than once for one feature vector.

[0091] FIG. 9 illustrates an example graphical diagram 900 of a serial approach for encoding feature vectors into neuron spikes in accordance with certain aspects of the present disclosure. As illustrated in FIG. 9, there may be less than 128 input neurons 902 per feature vector, and, for example, 1000 neurons 904 for intermediate features, as well as one neuron per category label 906. In an aspect, all-to-all connections with multiple synapses per node pair with different delay offsets may be implemented to account for the serial feature vector input.

[0092] The serial encoding approach illustrated in FIG. 9 utilizes fewer neurons and more processing time (larger latency) than the parallel encoding approach illustrated in FIG. 8. On the other hand, the parallel encoding approach uses more neurons and less processing time (smaller latency). As illustrated in FIG. 9, multiple layers of temporal and/or rate learning may be utilized for serial encoding of feature vectors into neuron spikes.

[0093] In one embodiment of the serial encoding approach, two or four SIFT feature vector elements may be taken and delays may be computed based on those element values using, for example, one of the mapping methods described above (e.g., the

flipped mapping illustrated in FIG. 7). The computed delays may be then concatenated in different ways. One way for concatenating delays can be to delay a first spike from a first reference point based on a first feature vector element, delay a second spike from the first spike based on a second feature vector element, and so on. A second way for concatenating delays is illustrated in examples 1002 and 1004 in FIG. 10, where, respectively, two and four feature vector elements are encoded per input neuron. In this case, a first spike can be delayed from a first reference point based on a first feature vector element value, a second spike can be delayed from a second reference point based on a second feature vector element value, and so on.

[0094] The serial encoding approach has an advantage of reducing the number of neurons needed to encode the SIFT feature vectors from an image (sensor data). For example, if two feature vector elements are encoded per neuron then the number of neurons may be halved, while if 128 elements are encoded per neuron then the number of neurons may be divided by 128.

[0095] Aspects of the present disclosure support a hybrid between the parallel and serial high level encoding approaches. In the hybrid approach, the computation time can be speeded up by having parallel temporal learning layer with weight sharing, but having fewer than the number of feature vectors.

[0096] Another variation on the serial coding approach is to interleave the spikes, as illustrated in an example 1100 in FIG. 11.

Fixed Weight Clustering Encoding Approach

[0097] One drawback to the serial and parallel encoding approaches described above is the number of intermediate neurons that learn cluster points. For example, if there are 300 intermediate neurons per feature vector and 500 feature vectors per image, then there are 150,000 intermediate neurons.

[0098] Certain aspects of the present disclosure support an encoding approach where the feature vectors may be combined to a large common cluster space in the first step using fixed random weights and synapse delays with weight and delay sharing between feature vector instances, as illustrated in a graphical diagram 1200 in FIG. 12.

[0099] The 10,000 intermediate cluster neuron point spikes may be generated in several different ways. In an aspect of the present disclosure, spikes may propagate from the input feature vector neurons 1202 as in the parallel or serial encoding methods, and let the synapses combine and generate one or more spikes at each intermediate neuron 1204 based on the synapse and neuron dynamics.

[0100] In another aspect, the outputs may be generated at the intermediate neurons 1204 for each feature vector independently as if the other feature vectors did not generate any spikes. Then, the set of spikes can be combined in several ways, such as by OR-ing together all the times when an intermediate neuron spiked for any of the feature vectors in the image, or by taking the minimum time when the intermediate cluster neuron spiked for any feature vector in the image, or applying other functions.

[0101] In yet another aspect, cluster points (e.g., 1000 of them) may be generated in the feature vector 128 dimensional Euclidean space. Then, a neuron may be created for each cluster point, and have the neuron spike at a time based on the Euclidean distance from the cluster point to the feature vector. Similar to the above, the cluster points could spike based on only the closest feature vector, based on all the feature vectors, or based on a function of the feature vectors in the image.

Histogram Encoding Approach

[0102] Aspects of the present disclosure support another encoding approach similar to the fixed random weight approach. In this case, the feature vectors may be further pre-processed, such as by k-means clustering over a training set. Then, the feature vectors may be computed for each image, the closest cluster point to the feature vector may be determined, and a histogram of the number of feature vectors near each cluster point may be created. Then, the histograms may be encoded into spike times using the methods described previously, such as using upper k bits or truncating the top number of bits and converting the value into a delay.

[0103] The advantage of this approach is similar to the fixed random weight approach, i.e., it may considerably reduce the number of input neurons for the neural network making the size easier to fit on hardware. Another advantage is performing

some of the front-end pre-processing using more traditional signal processing techniques.

[0104] Certain aspects of the present disclosure support additional extensions on the histogram based encoding approach. In an aspect, histograms of the closest and second closest cluster points may be created and those values may be encoded. In another aspect, histograms of points within a distance d_1 and beyond a distance d_1 and within a distance d_2 to the cluster points may be created and those values may be encoded. In yet another aspect, one (parallel) or more than one (serial) value may be sent per input neuron. In yet another aspect, the histogram values may be compressed to a delay range in a few ways including normalization to get the top bits and it may be saturated to get more resolution for the smaller values. Also, a bias can be added or a threshold may be used to indicate greater than or less than a given level.

[0105] It should be noted that the number of SIFT feature vectors may vary substantially based on the image size in pixels. Therefore, if images of different sizes are fed in, it may be desirable to normalize the histogram to avoid biases based on an image size. In an aspect of the present disclosure, the histogram vector may be normalized to a unit length. In another aspect, the maximum value of the histogram vector may be normalized to the value of one.

[0106] FIG. 13 is a flow diagram of example operations 1300 for operating an artificial nervous system in accordance with certain aspects of the present disclosure. The operations 1300 may be performed in hardware (e.g., by one or more neural processing units, such as a neuromorphic processor), in software, or in firmware. The artificial nervous system may be modeled on any of various biological or imaginary nervous systems, such as a visual nervous system, an auditory nervous system, the hippocampus, etc.

[0107] The operations 1300 may begin, at 1302, by obtaining sensor data being input into the artificial nervous system. At 1304, the sensor data may be processed to generate feature vectors. At 1306, element values of the feature vectors may be converted into delays. At 1308, at least one artificial neuron of the artificial nervous system may be configured to spike at times based on the delays.

[0108] In an aspect of the present disclosure, processing the sensor data to generate the feature vectors may comprise performing Scale Invariant Feature Transform (SIFT) on the sensor data (e.g., image). According to certain aspects of the present disclosure, processing the sensor data may further comprise selecting a subset of the feature vectors for converting into the delays. In an aspect, the subset of feature vectors may comprise a portion of the feature vectors associated with features of the sensor data ranked above a threshold according to specific criteria. In another aspect, the subset of feature vectors may comprise a portion of the feature vectors associated with frequency bands of the sensor data ranked above a threshold according to specific criteria.

[0109] For certain aspects of the present disclosure, processing the sensor data may comprise using prior learned sensor categories to map the sensor data to the feature vectors. In an aspect, the sensor categories may comprise histograms related to the sensor data.

[0110] In an aspect of the present disclosure, processing the sensor data may further comprise pre-distorting the element values of the feature vectors to map the element values to an implicit temporal learning distance metric. In another aspect, processing the sensor data may further comprise pre-distorting the element values of the feature vectors to achieve a specific distance metric related to the feature vectors.

[0111] In an aspect of the present disclosure, converting the element values of feature vectors into the delays may comprise linear mapping of the element values into the delays. In another aspect, converting the element values of feature vectors into the delays may comprise logarithmic mapping of the element values into the delays. In yet another aspect, converting the element values of feature vectors into the delays may comprise inverse mapping of the element values into the delays. In yet another aspect, converting the element values of feature vectors into the delays may comprise mapping of less frequent and larger of the element values into smaller of the delays.

[0112] In an aspect of the present disclosure, two or more values of the sensor data may be mapped into spiking of the at least one artificial neuron of the artificial nervous system. In another aspect, converting the element values of feature vectors into the delays may comprise mapping two or more of the element values into spiking of the at least one artificial neuron of the artificial nervous system. For certain aspects of the

present disclosure, learning of multiple parallel structures of the feature vectors, based on the at least one artificial neuron spiking, may utilize synapse weight sharing so that the structures learn same weights and are order invariant.

[0113] FIG. 14 illustrates an example block diagram 1400 of the aforementioned method for operating an artificial nervous system using a general-purpose processor 1402 in accordance with certain aspects of the present disclosure. Variables (neural signals), synaptic weights, and/or system parameters associated with a computational network (neural network) may be stored in a memory block 1404, while instructions related executed at the general-purpose processor 1402 may be loaded from a program memory 1406. In an aspect of the present disclosure, the instructions loaded into the general-purpose processor 1402 may comprise code for obtaining sensor data being input into the artificial nervous system, for processing the sensor data to generate feature vectors, for converting element values of the feature vectors into delays, and for causing at least one artificial neuron of the artificial nervous system to spike at times based on the delays.

[0114] FIG. 15 illustrates an example block diagram 1500 of the aforementioned method for operating an artificial nervous system where a memory 1502 can be interfaced via an interconnection network 1504 with individual (distributed) processing units (neural processors) 1506 of a computational network (neural network) in accordance with certain aspects of the present disclosure. Variables (neural signals), synaptic weights, and/or system parameters associated with the computational network (neural network) may be stored in the memory 1502, and may be loaded from the memory 1502 via connection(s) of the interconnection network 1504 into each processing unit (neural processor) 1506. In an aspect of the present disclosure, the processing unit 1506 may be configured to obtain sensor data being input into the artificial nervous system, to process the sensor data to generate feature vectors, to convert element values of the feature vectors into delays, and to cause at least one artificial neuron of the artificial nervous system to spike at times based on the delays.

[0115] FIG. 16 illustrates an example block diagram 1600 of the aforementioned method for operating an artificial nervous system based on distributed weight memories 1602 and distributed processing units (neural processors) 1604 in accordance with certain aspects of the present disclosure. As illustrated in FIG. 16, one memory bank

1602 may be directly interfaced with one processing unit 1604 of a computational network (neural network), wherein that memory bank 1602 may store variables (neural signals), synaptic weights, and/or system parameters associated with that processing unit (neural processor) 1604. In an aspect of the present disclosure, the processing unit(s) 1604 may be configured to obtain sensor data being input into the artificial nervous system, to process the sensor data to generate feature vectors, to convert element values of the feature vectors into delays, and to cause at least one artificial neuron of the artificial nervous system to spike at times based on the delays.

[0116] FIG. 17 illustrates an example implementation of a neural network 1700 in accordance with certain aspects of the present disclosure. As illustrated in FIG. 17, the neural network 1700 may comprise a plurality of local processing units 1702 that may perform various operations of methods described above. Each processing unit 1702 may comprise a local state memory 1704 and a local parameter memory 1706 that store parameters of the neural network. In addition, the processing unit 1702 may comprise a memory 1708 with a local (neuron) model program, a memory 1710 with a local learning program, and a local connection memory 1712. Furthermore, as illustrated in FIG. 17, each local processing unit 1702 may be interfaced with a unit 1714 for configuration processing that may provide configuration for local memories of the local processing unit, and with routing connection processing elements 1716 that provide routing between the local processing units 1702.

[0117] According to certain aspects of the present disclosure, each local processing unit 1702 may be configured to determine parameters of the neural network based upon desired one or more functional features of the neural network, and develop the one or more functional features towards the desired functional features as the determined parameters are further adapted, tuned and updated.

[0118] FIG. 18 is a block diagram 1800 of an example hardware implementation for an artificial nervous system, in accordance with certain aspects of the present disclosure. STDP updating, as described above, may occur in an Effect Plasticity Updates and Reassemble block 1802. For certain aspects, the updated synaptic weights may be stored, via a cache line interface 1804, in an off-chip memory (e.g., dynamic random access memory (DRAM) 1806).

[0119] In a typical artificial nervous system, there are many more synapses than artificial neurons, and for a large neural network, processing the synapse updates in an efficient manner is desired. The large number of synapses may suggest storing the synaptic weight and other parameters in memory (e.g., DRAM 1806). When artificial neurons generate spikes in a so-called “super neuron (SN),” the neurons may forward those spikes to the post-synaptic neurons through DRAM lookups to determine the post-synaptic neurons and corresponding neural weights. To enable fast and efficient lookup, the synapse ordering may be kept consecutively in memory based, for example, on fan-out from a neuron. Later when processing STDP updates in the Effect Plasticity Updates and Reassemble block 1802, efficiency may dictate processing the updates based on a forward fan-out given this memory layout since the DRAM or a large lookup table need not be searched to determine the reverse mapping for LTP updates. The approach shown in FIG. 18 facilitates this. The Effect Plasticity Updates and Reassemble block 1802 may query the super neurons in an effort to obtain the pre- and post-synaptic spike times, again reducing the amount of state memory involved.

[0120] The various operations of methods described above may be performed by any suitable means capable of performing the corresponding functions. The means may include various hardware and/or software component(s) and/or module(s), including, but not limited to a circuit, an application specific integrated circuit (ASIC), or processor. For example, the various operations may be performed by one or more of the various processors shown in FIGS. 14-18. Generally, where there are operations illustrated in figures, those operations may have corresponding counterpart means-plus-function components with similar numbering. For example, operations 1300 illustrated in FIG. 13 correspond to means 1300A illustrated in FIG. 13A.

[0121] For example, means for displaying may include a display (e.g., a monitor, flat screen, touch screen, and the like), a printer, or any other suitable means for outputting data for visual depiction (e.g., a table, chart, or graph). Means for processing, means for receiving, means for tracking, means for adjusting, means for updating, or means for determining may comprise a processing system, which may include one or more processors or processing units. Means for sensing may include a sensor. Means for storing may include a memory or any other suitable storage device (e.g., RAM), which may be accessed by the processing system.

[0122] As used herein, the term “determining” encompasses a wide variety of actions. For example, “determining” may include calculating, computing, processing, deriving, investigating, looking up (e.g., looking up in a table, a database or another data structure), ascertaining, and the like. Also, “determining” may include receiving (e.g., receiving information), accessing (e.g., accessing data in a memory), and the like. Also, “determining” may include resolving, selecting, choosing, establishing, and the like.

[0123] As used herein, a phrase referring to “at least one of” a list of items refers to any combination of those items, including single members. As an example, “at least one of a, b, or c” is intended to cover a, b, c, a-b, a-c, b-c, and a-b-c.

[0124] The various illustrative logical blocks, modules, and circuits described in connection with the present disclosure may be implemented or performed with a general purpose processor, a digital signal processor (DSP), an application specific integrated circuit (ASIC), a field programmable gate array signal (FPGA) or other programmable logic device (PLD), discrete gate or transistor logic, discrete hardware components or any combination thereof designed to perform the functions described herein. A general-purpose processor may be a microprocessor, but in the alternative, the processor may be any commercially available processor, controller, microcontroller, or state machine. A processor may also be implemented as a combination of computing devices, e.g., a combination of a DSP and a microprocessor, a plurality of microprocessors, one or more microprocessors in conjunction with a DSP core, or any other such configuration.

[0125] The steps of a method or algorithm described in connection with the present disclosure may be embodied directly in hardware, in a software module executed by a processor, or in a combination of the two. A software module may reside in any form of storage medium that is known in the art. Some examples of storage media that may be used include random access memory (RAM), read only memory (ROM), flash memory, EPROM memory, EEPROM memory, registers, a hard disk, a removable disk, a CD-ROM and so forth. A software module may comprise a single instruction, or many instructions, and may be distributed over several different code segments, among different programs, and across multiple storage media. A storage medium may be coupled to a processor such that the processor can read information from, and write information to, the storage medium. In the alternative, the storage medium may be integral to the processor.

[0126] The methods disclosed herein comprise one or more steps or actions for achieving the described method. The method steps and/or actions may be interchanged with one another without departing from the scope of the claims. In other words, unless a specific order of steps or actions is specified, the order and/or use of specific steps and/or actions may be modified without departing from the scope of the claims.

[0127] The functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in hardware, an example hardware configuration may comprise a processing system in a device. The processing system may be implemented with a bus architecture. The bus may include any number of interconnecting buses and bridges depending on the specific application of the processing system and the overall design constraints. The bus may link together various circuits including a processor, machine-readable media, and a bus interface. The bus interface may be used to connect a network adapter, among other things, to the processing system via the bus. The network adapter may be used to implement signal processing functions. For certain aspects, a user interface (e.g., keypad, display, mouse, joystick, etc.) may also be connected to the bus. The bus may also link various other circuits such as timing sources, peripherals, voltage regulators, power management circuits, and the like, which are well known in the art, and therefore, will not be described any further.

[0128] The processor may be responsible for managing the bus and general processing, including the execution of software stored on the machine-readable media. The processor may be implemented with one or more general-purpose and/or special-purpose processors. Examples include microprocessors, microcontrollers, DSP processors, and other circuitry that can execute software. Software shall be construed broadly to mean instructions, data, or any combination thereof, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Machine-readable media may include, by way of example, RAM (Random Access Memory), flash memory, ROM (Read Only Memory), PROM (Programmable Read-Only Memory), EPROM (Erasable Programmable Read-Only Memory), EEPROM (Electrically Erasable Programmable Read-Only Memory), registers, magnetic disks, optical disks, hard drives, or any other suitable storage medium, or any

combination thereof. The machine-readable media may be embodied in a computer-program product. The computer-program product may comprise packaging materials.

[0129] In a hardware implementation, the machine-readable media may be part of the processing system separate from the processor. However, as those skilled in the art will readily appreciate, the machine-readable media, or any portion thereof, may be external to the processing system. By way of example, the machine-readable media may include a transmission line, a carrier wave modulated by data, and/or a computer product separate from the device, all which may be accessed by the processor through the bus interface. Alternatively, or in addition, the machine-readable media, or any portion thereof, may be integrated into the processor, such as the case may be with cache and/or general register files.

[0130] The processing system may be configured as a general-purpose processing system with one or more microprocessors providing the processor functionality and external memory providing at least a portion of the machine-readable media, all linked together with other supporting circuitry through an external bus architecture. Alternatively, the processing system may be implemented with an ASIC (Application Specific Integrated Circuit) with the processor, the bus interface, the user interface, supporting circuitry, and at least a portion of the machine-readable media integrated into a single chip, or with one or more FPGAs (Field Programmable Gate Arrays), PLDs (Programmable Logic Devices), controllers, state machines, gated logic, discrete hardware components, or any other suitable circuitry, or any combination of circuits that can perform the various functionality described throughout this disclosure. Those skilled in the art will recognize how best to implement the described functionality for the processing system depending on the particular application and the overall design constraints imposed on the overall system.

[0131] The machine-readable media may comprise a number of software modules. The software modules include instructions that, when executed by the processor, cause the processing system to perform various functions. The software modules may include a transmission module and a receiving module. Each software module may reside in a single storage device or be distributed across multiple storage devices. By way of example, a software module may be loaded into RAM from a hard drive when a triggering event occurs. During execution of the software module, the processor may

load some of the instructions into cache to increase access speed. One or more cache lines may then be loaded into a general register file for execution by the processor. When referring to the functionality of a software module below, it will be understood that such functionality is implemented by the processor when executing instructions from that software module.

[0132] If implemented in software, the functions may be stored or transmitted over as one or more instructions or code on a computer-readable medium. Computer-readable media include both computer storage media and communication media including any medium that facilitates transfer of a computer program from one place to another. A storage medium may be any available medium that can be accessed by a computer. By way of example, and not limitation, such computer-readable media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium that can be used to carry or store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if the software is transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared (IR), radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. Disk and disc, as used herein, include compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk, and Blu-ray® disc where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Thus, in some aspects computer-readable media may comprise non-transitory computer-readable media (e.g., tangible media). In addition, for other aspects computer-readable media may comprise transitory computer-readable media (e.g., a signal). Combinations of the above should also be included within the scope of computer-readable media.

[0133] Thus, certain aspects may comprise a computer program product for performing the operations presented herein. For example, such a computer program product may comprise a computer readable medium having instructions stored (and/or encoded) thereon, the instructions being executable by one or more processors to

perform the operations described herein. For certain aspects, the computer program product may include packaging material.

[0134] Further, it should be appreciated that modules and/or other appropriate means for performing the methods and techniques described herein can be downloaded and/or otherwise obtained by a device as applicable. For example, such a device can be coupled to a server to facilitate the transfer of means for performing the methods described herein. Alternatively, various methods described herein can be provided via storage means (e.g., RAM, ROM, a physical storage medium such as a compact disc (CD) or floppy disk, etc.), such that a device can obtain the various methods upon coupling or providing the storage means to the device. Moreover, any other suitable technique for providing the methods and techniques described herein to a device can be utilized.

[0135] It is to be understood that the claims are not limited to the precise configuration and components illustrated above. Various modifications, changes and variations may be made in the arrangement, operation and details of the methods and apparatus described above without departing from the scope of the claims.

CLAIMS

1. A method for operating an artificial nervous system, comprising:
obtaining sensor data being input into the artificial nervous system;
processing the sensor data to generate feature vectors;
converting element values of the feature vectors into delays; and
causing at least one artificial neuron of the artificial nervous system to spike at times based on the delays.
2. The method of claim 1, wherein processing the sensor data to generate the feature vectors comprises performing Scale Invariant Feature Transform (SIFT) on the sensor data.
3. The method of claim 1, wherein processing the sensor data further comprises selecting a subset of the feature vectors for converting into the delays.
4. The method of claim 3, wherein the subset of feature vectors comprises a portion of the feature vectors associated with features of the sensor data ranked above a threshold according to specific criteria.
5. The method of claim 3, wherein the subset of feature vectors comprises a portion of the feature vectors associated with frequency bands of the sensor data ranked above a threshold according to specific criteria.
6. The method of claim 1, wherein processing the sensor data comprises using prior learned sensor categories to map the sensor data to the feature vectors.
7. The method of claim 6, wherein the sensor categories comprise histograms related to the sensor data.
8. The method of claim 1, wherein processing the sensor data further comprises pre-distorting the element values of the feature vectors to map the element values to an implicit temporal learning distance metric.
9. The method of claim 1, wherein processing the sensor data further comprises pre-distorting the element values of the feature vectors to achieve a specific distance metric related to the feature vectors.

10. The method of claim 1, wherein converting the element values of feature vectors into the delays comprises linear mapping of the element values into the delays.
11. The method of claim 1, wherein converting the element values of feature vectors into the delays comprises logarithmic mapping of the element values into the delays.
12. The method of claim 1, wherein converting the element values of feature vectors into the delays comprises inverse mapping of the element values into the delays.
13. The method of claim 1, wherein converting the element values of feature vectors into the delays comprises mapping of less frequent and larger of the element values into smaller of the delays.
14. The method of claim 1, further comprising:
mapping of two or more values of the sensor data into spiking of the at least one artificial neuron of the artificial nervous system.
15. The method of claim 1, wherein converting the element values of feature vectors into the delays comprises mapping two or more of the element values into spiking of the at least one artificial neuron of the artificial nervous system.
16. The method of claim 1, further comprising:
learning, based on the at least one artificial neuron spiking, multiple parallel structures of the feature vectors using synapse weight sharing so that the structures learn same weights and are order invariant.
17. An apparatus for operating an artificial nervous system, comprising:
a sensor configured to obtain sensor data being input into the artificial nervous system;
a first circuit configured to process the sensor data to generate feature vectors;
a second circuit configured to convert element values of the feature vectors into delays; and
a third circuit configured to cause at least one artificial neuron of the artificial nervous system to spike at times based on the delays.
18. The apparatus of claim 17, wherein the first circuit is also configured to perform Scale Invariant Feature Transform (SIFT) on the sensor data.

19. The apparatus of claim 17, wherein the first circuit is also configured to select a subset of the feature vectors for converting into the delays.
20. The apparatus of claim 19, wherein the subset of feature vectors comprises a portion of the feature vectors associated with features of the sensor data ranked above a threshold according to specific criteria.
21. The apparatus of claim 19, wherein the subset of feature vectors comprises a portion of the feature vectors associated with frequency bands of the sensor data ranked above a threshold according to specific criteria.
22. The apparatus of claim 17, wherein the first circuit is also configured to use prior learned sensor categories to map the sensor data to the feature vectors.
23. The apparatus of claim 22, wherein the sensor categories comprise histograms related to the sensor data.
24. The apparatus of claim 17, wherein the first circuit is also configured to pre-distort the element values of the feature vectors to map the element values to an implicit temporal learning distance metric.
25. The apparatus of claim 17, wherein the first circuit is also configured to pre-distort the element values of the feature vectors to achieve a specific distance metric related to the feature vectors.
26. The apparatus of claim 17, wherein the second circuit configured to convert the element values of feature vectors into the delays is also configured to perform linear mapping of the element values into the delays.
27. The apparatus of claim 17, wherein the second circuit configured to convert the element values of feature vectors into the delays is also configured to perform logarithmic mapping of the element values into the delays.
28. The apparatus of claim 17, wherein the second circuit configured to convert the element values of feature vectors into the delays is also configured to perform inverse mapping of the element values into the delays.

29. The apparatus of claim 17, wherein the second circuit configured to convert the element values of feature vectors into the delays is also configured to perform mapping of less frequent and larger of the element values into smaller of the delays.

30. The apparatus of claim 17, wherein the third circuit is also configured to map two or more values of the sensor data into spiking of the at least one artificial neuron of the artificial nervous system.

31. The apparatus of claim 17, wherein the third circuit is also configured to map two or more of the element values into spiking of the at least one artificial neuron of the artificial nervous system.

32. The apparatus of claim 17, further comprising:
a fourth circuit configured to learn, based on the at least one artificial neuron spiking, multiple parallel structures of the feature vectors using synapse weight sharing so that the structures learn same weights and are order invariant.

33. An apparatus for operating an artificial nervous system, comprising:
means for obtaining sensor data being input into the artificial nervous system;
means for processing the sensor data to generate feature vectors;
means for converting element values of the feature vectors into delays; and
means for causing at least one artificial neuron of the artificial nervous system to spike at times based on the delays.

34. A computer-readable medium having instructions executable by a computer stored thereon for:
obtaining sensor data being input into an artificial nervous system;
processing the sensor data to generate feature vectors;
converting element values of the feature vectors into delays; and
causing at least one artificial neuron of the artificial nervous system to spike at times based on the delays.

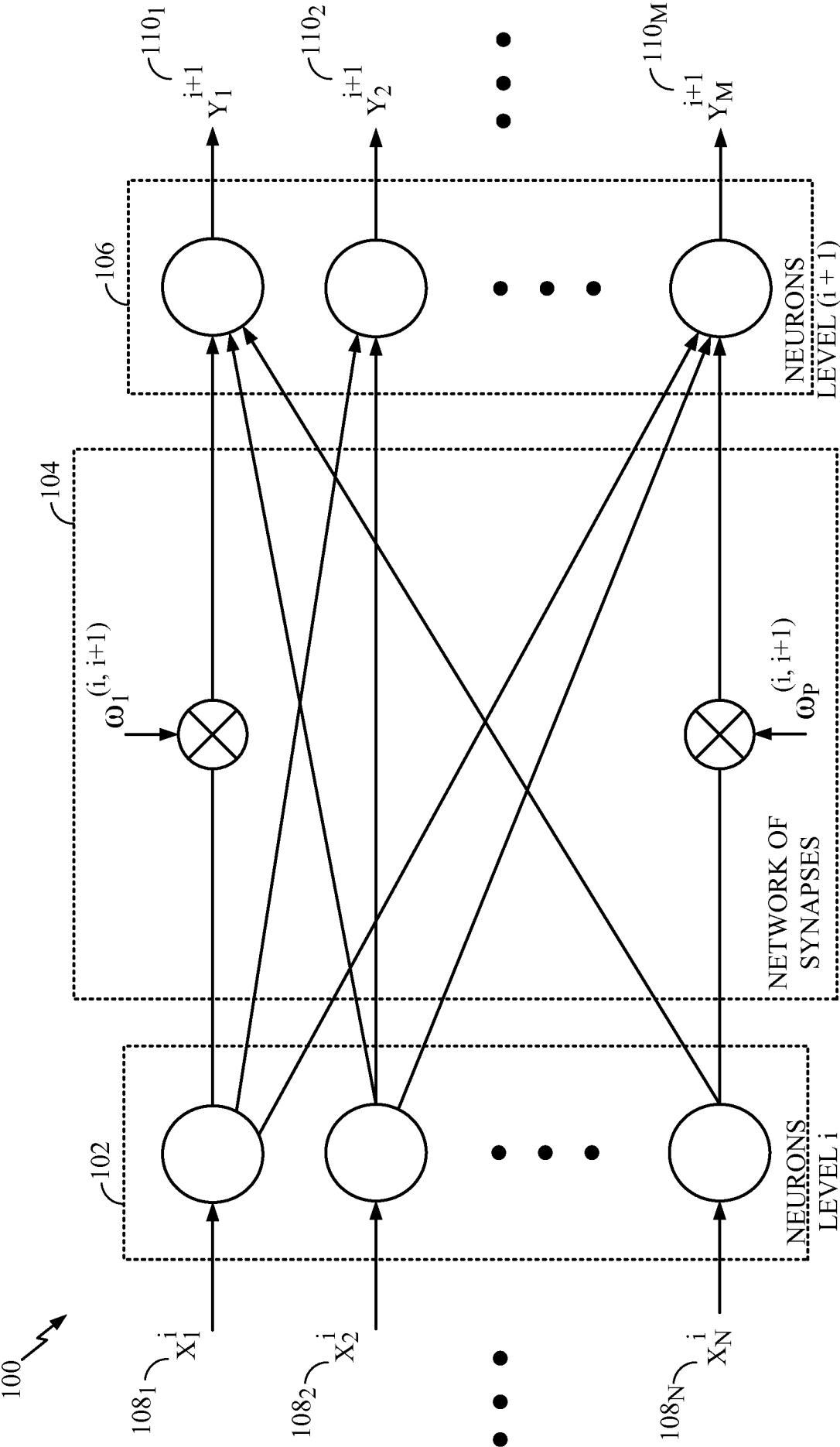


FIG. 1

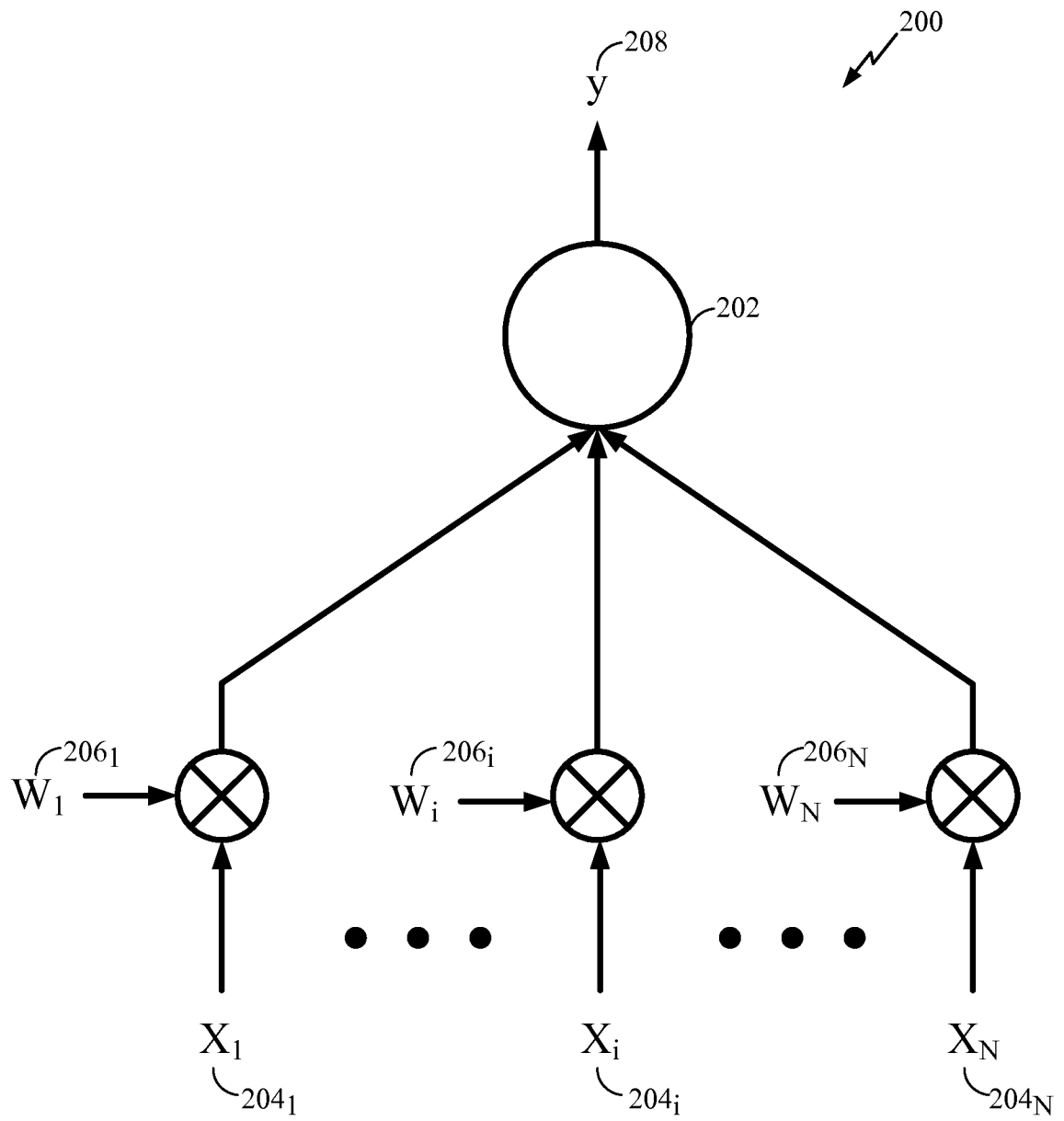


FIG. 2

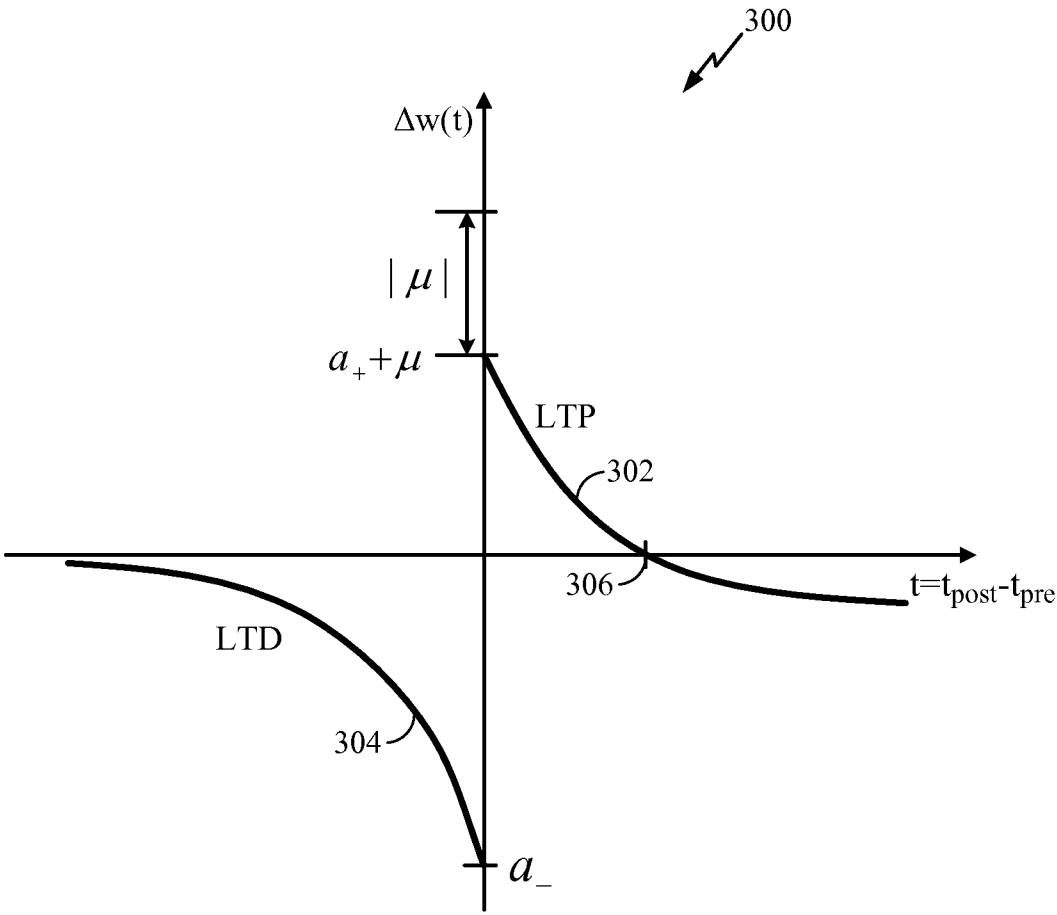


FIG. 3

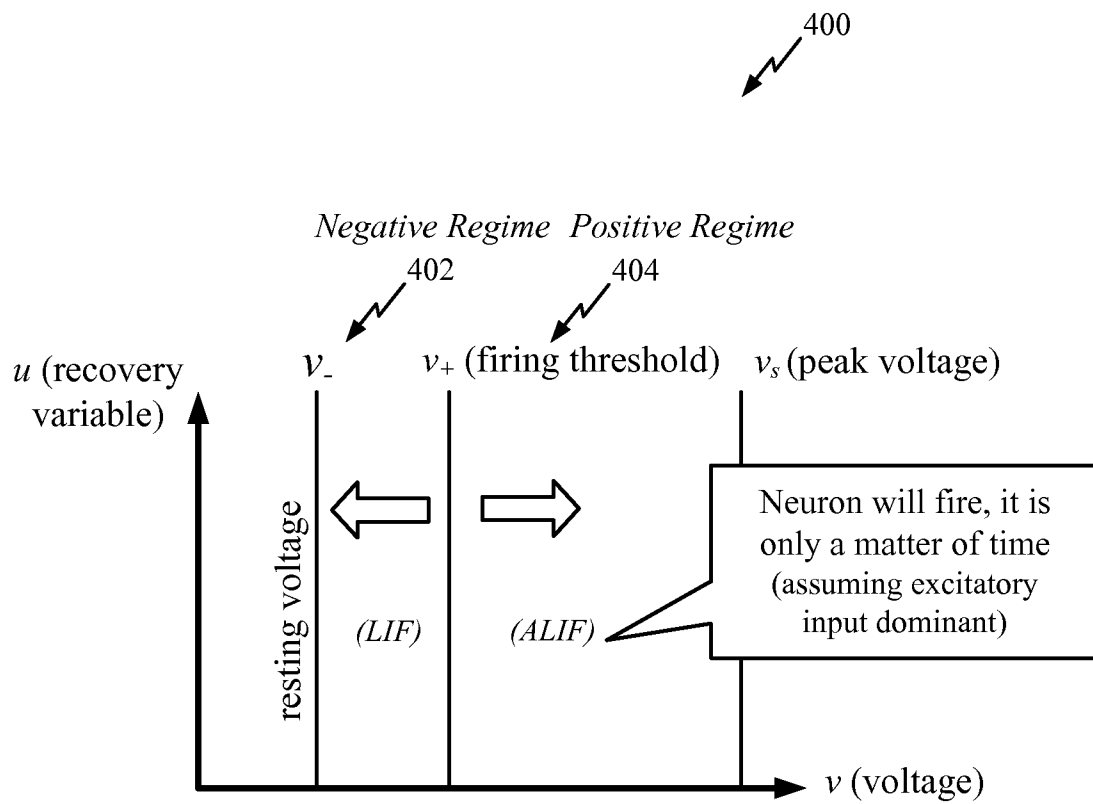


FIG. 4

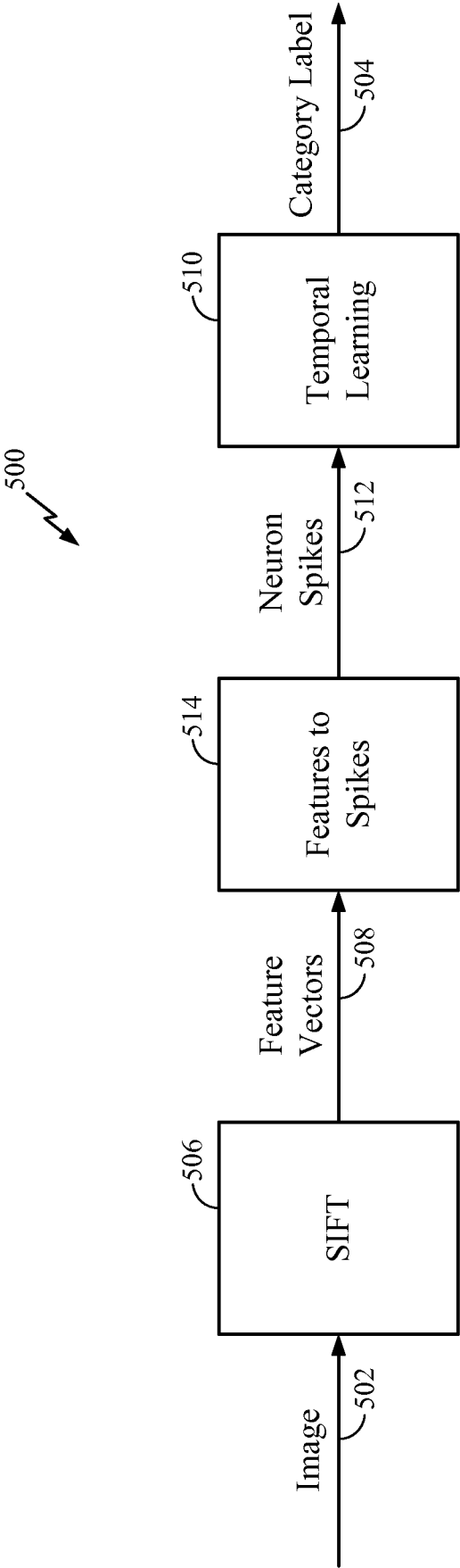


FIG. 5

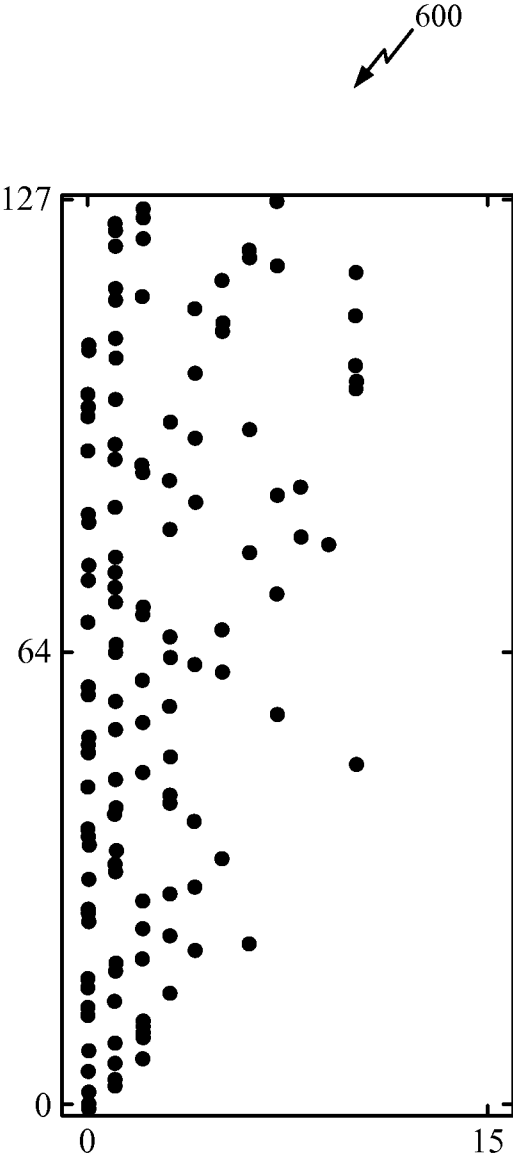


FIG. 6

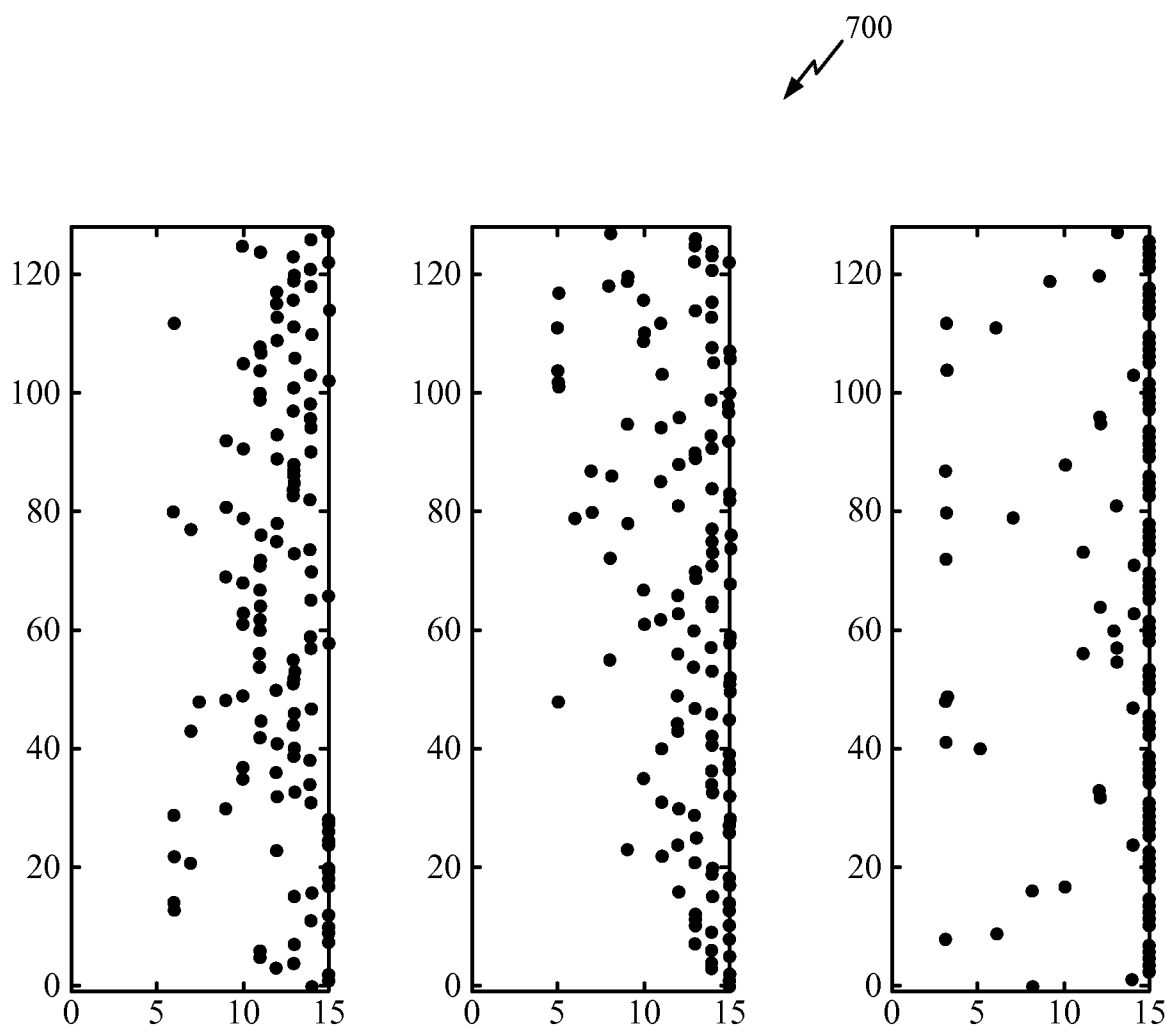


FIG. 7

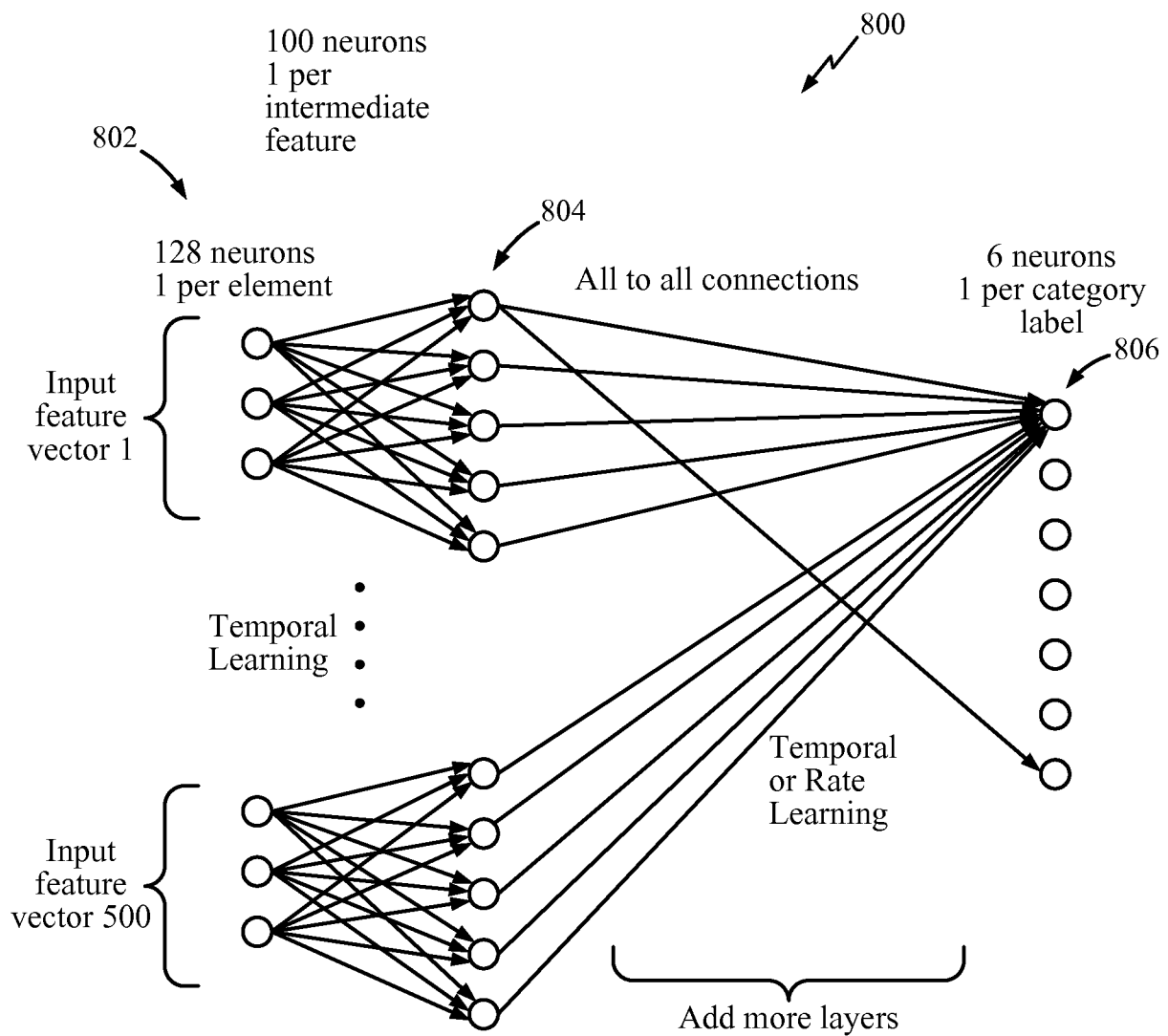


FIG. 8

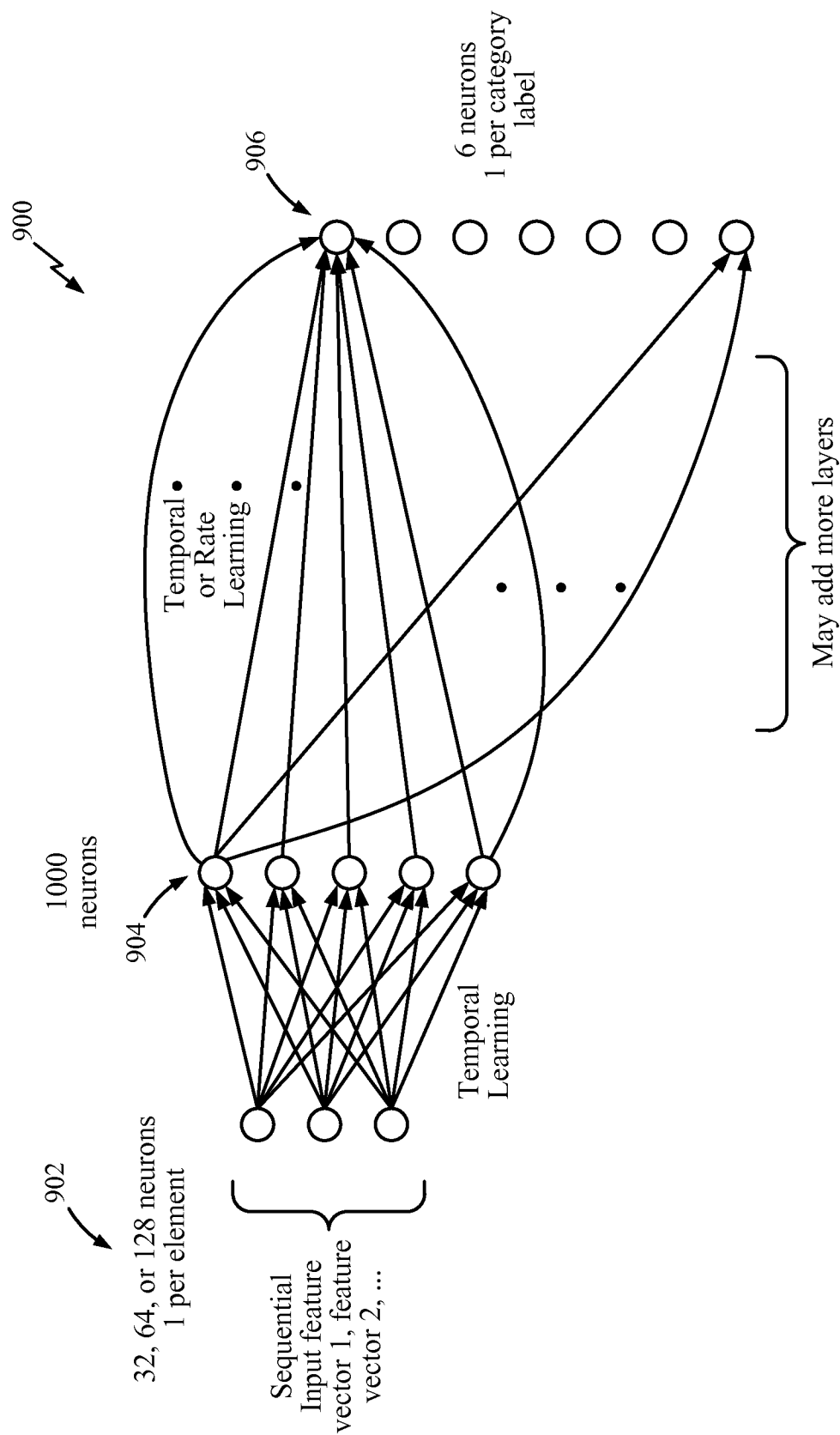


FIG. 9

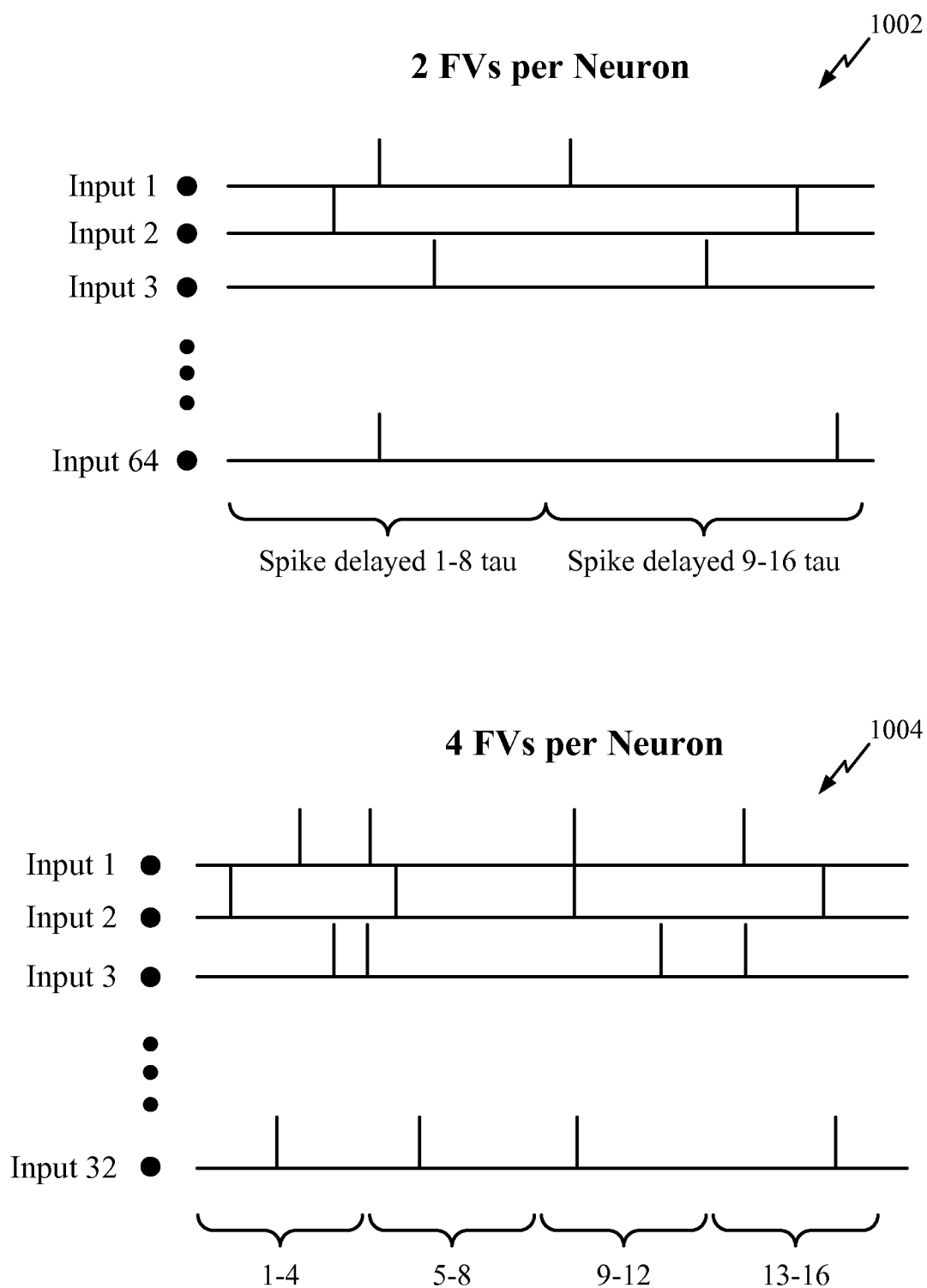


FIG. 10

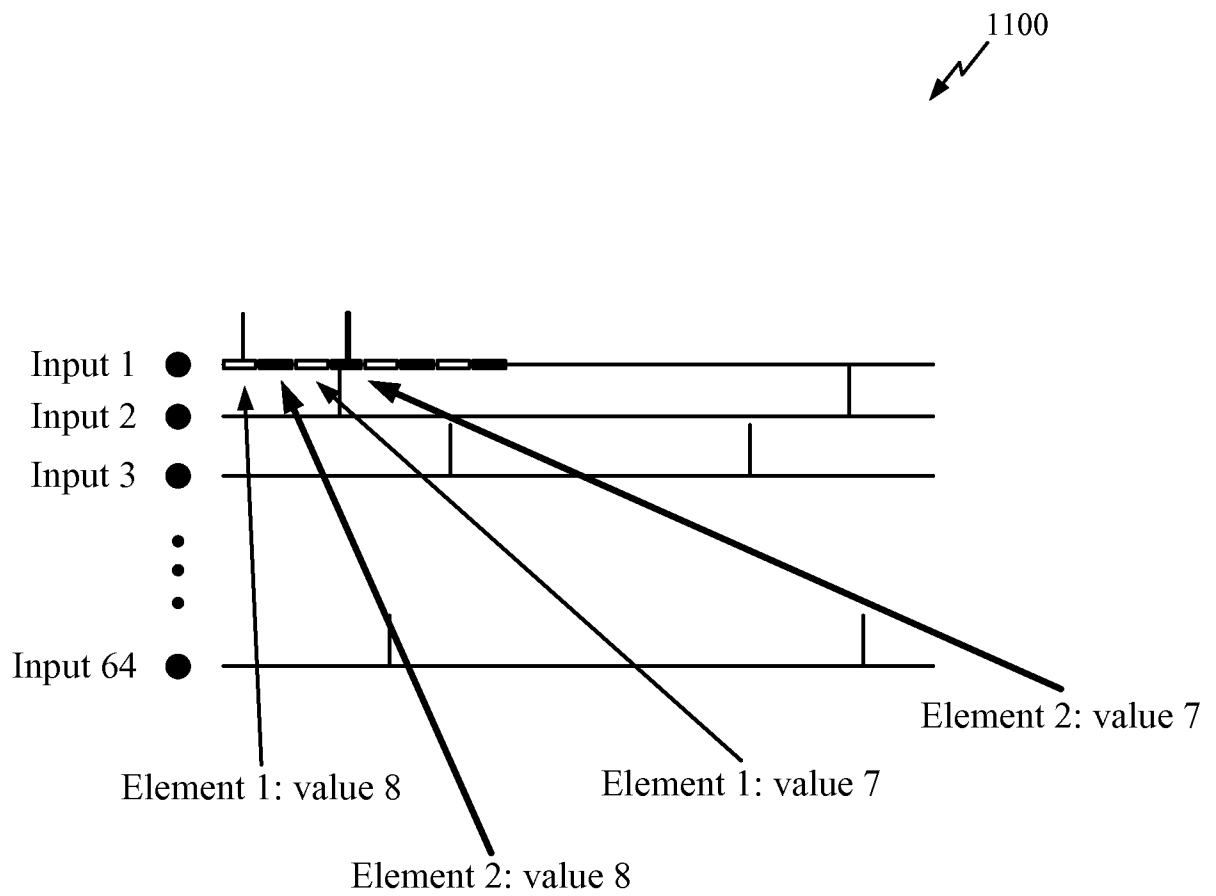


FIG. 11

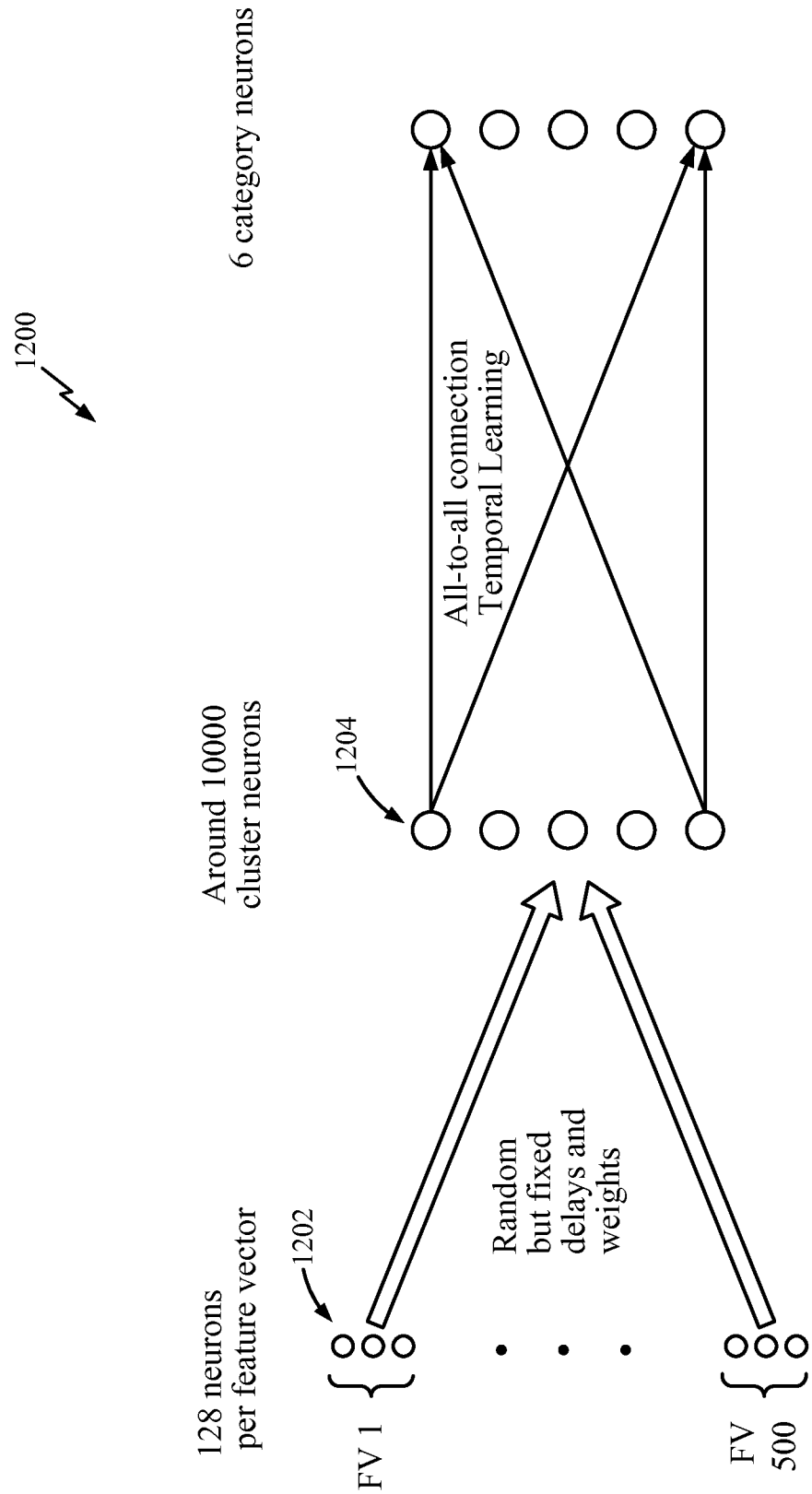


FIG. 12

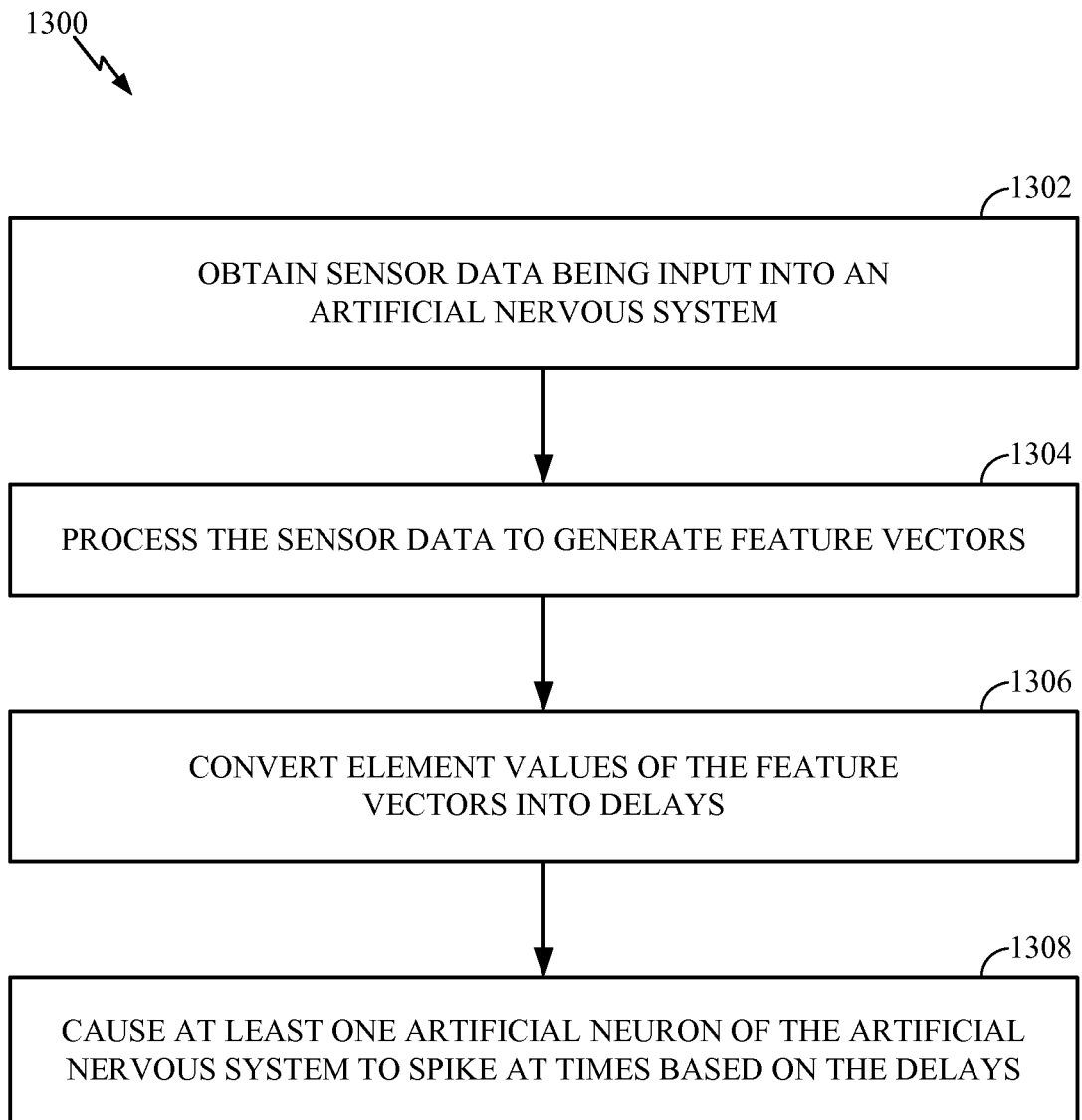


FIG. 13

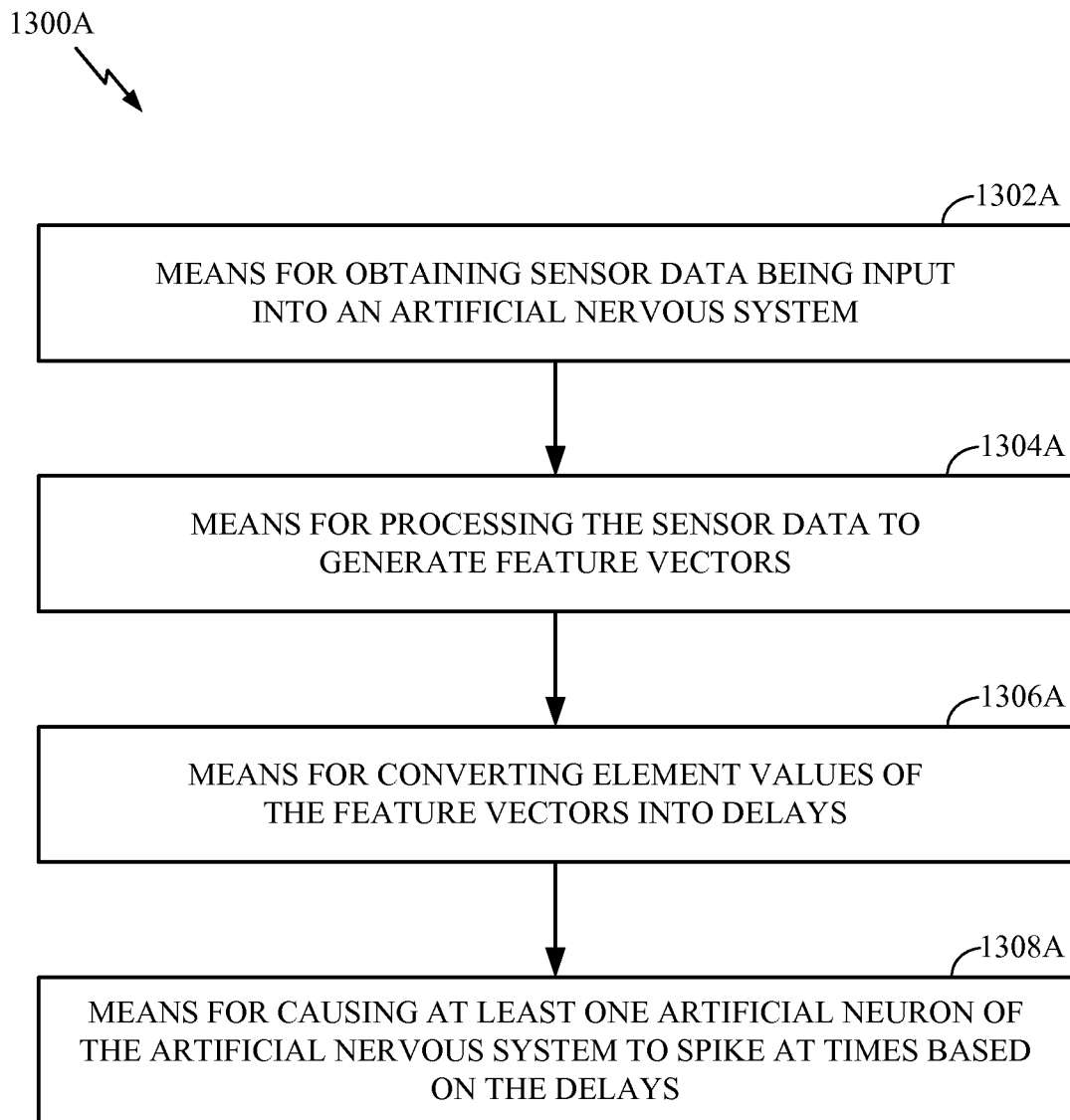


FIG. 13A

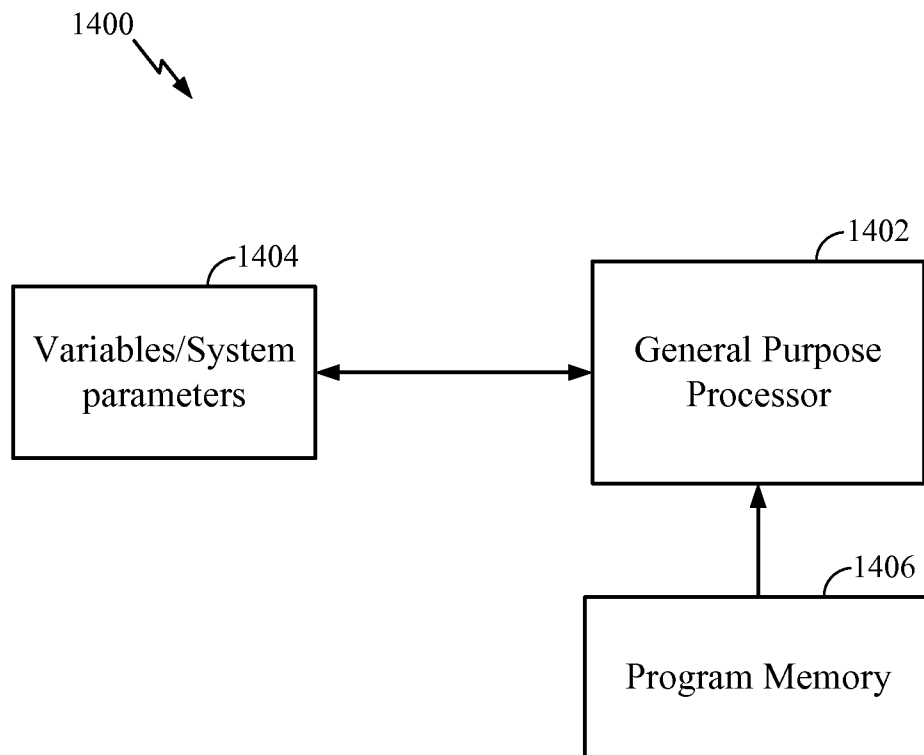


FIG. 14

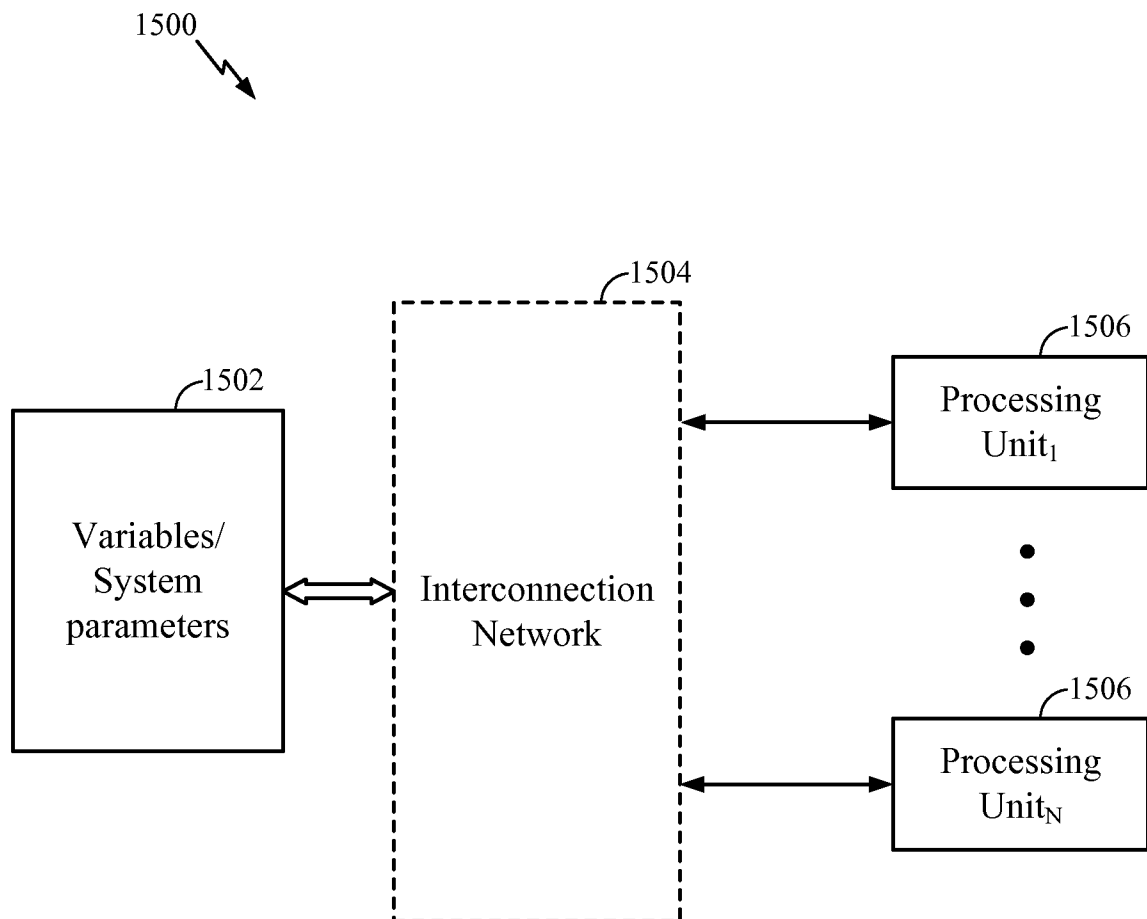


FIG. 15

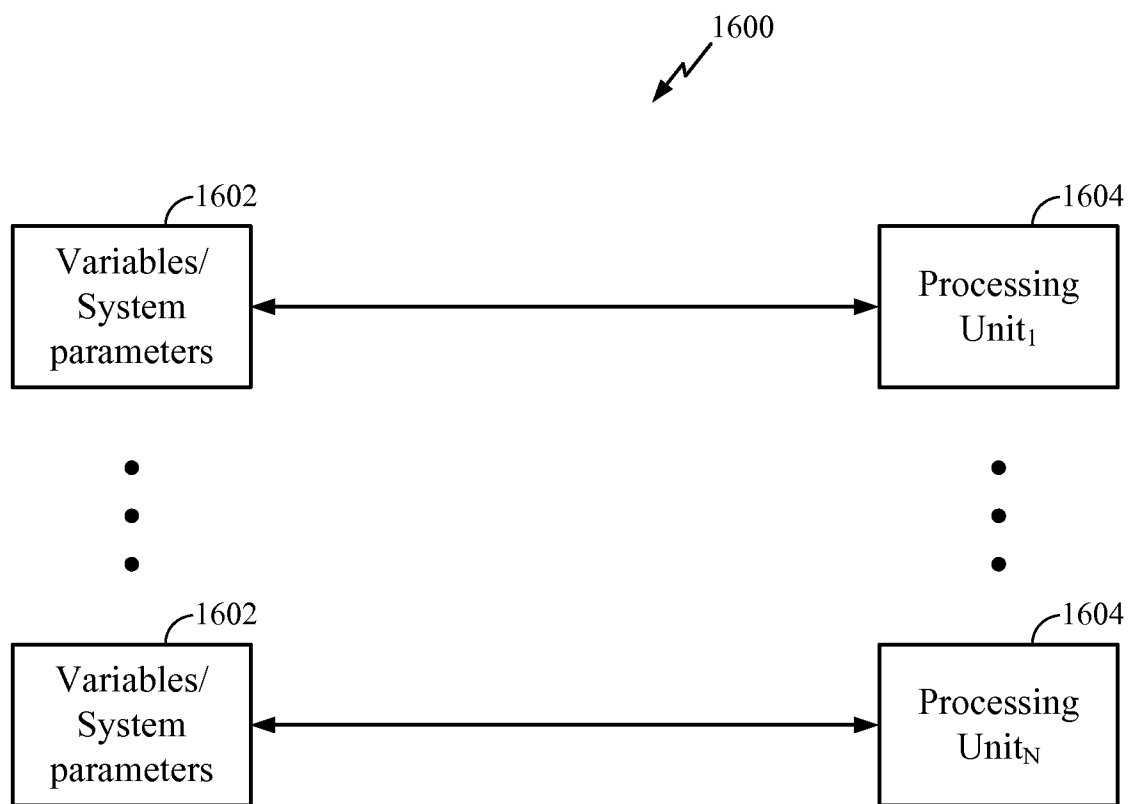


FIG. 16

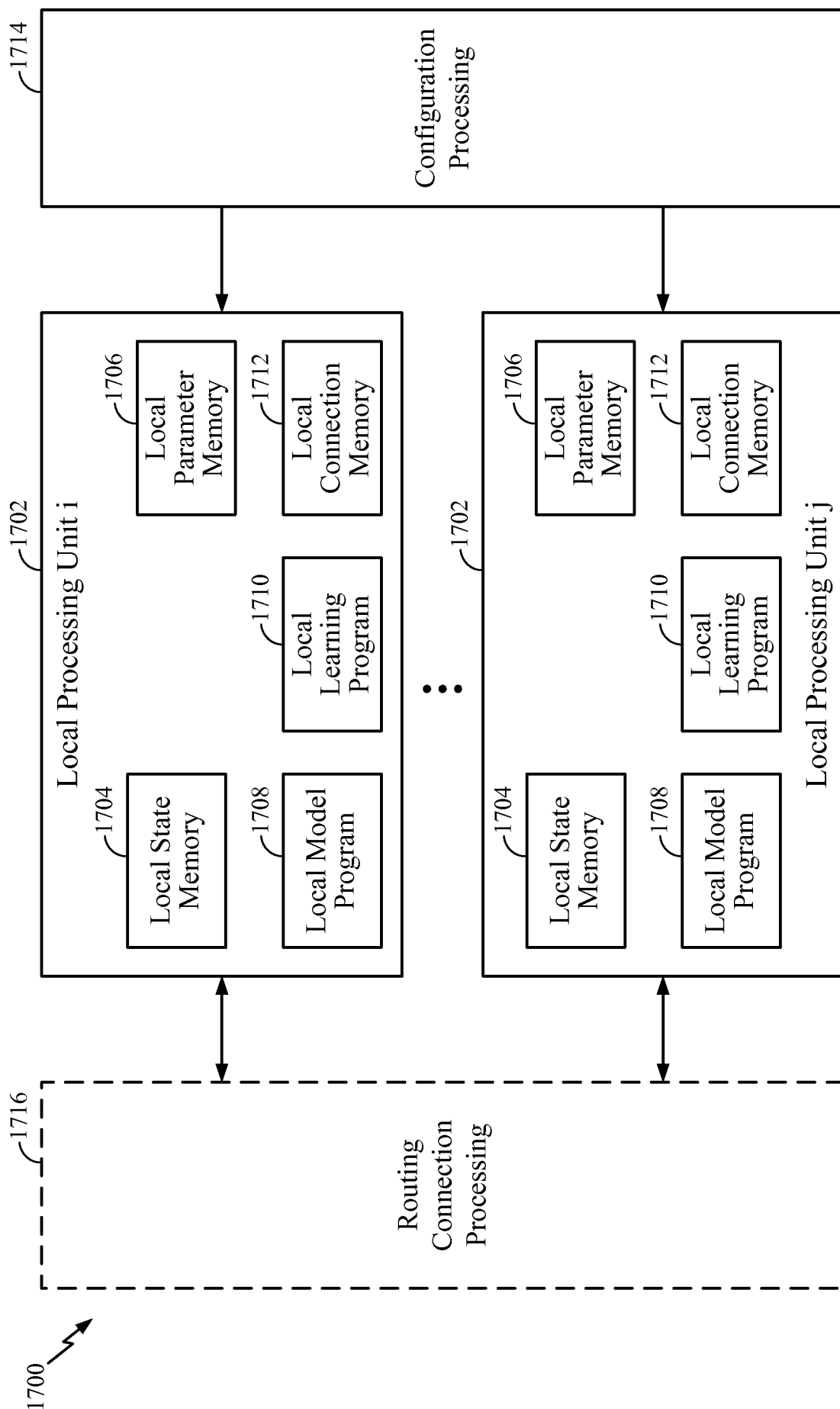


FIG. 17

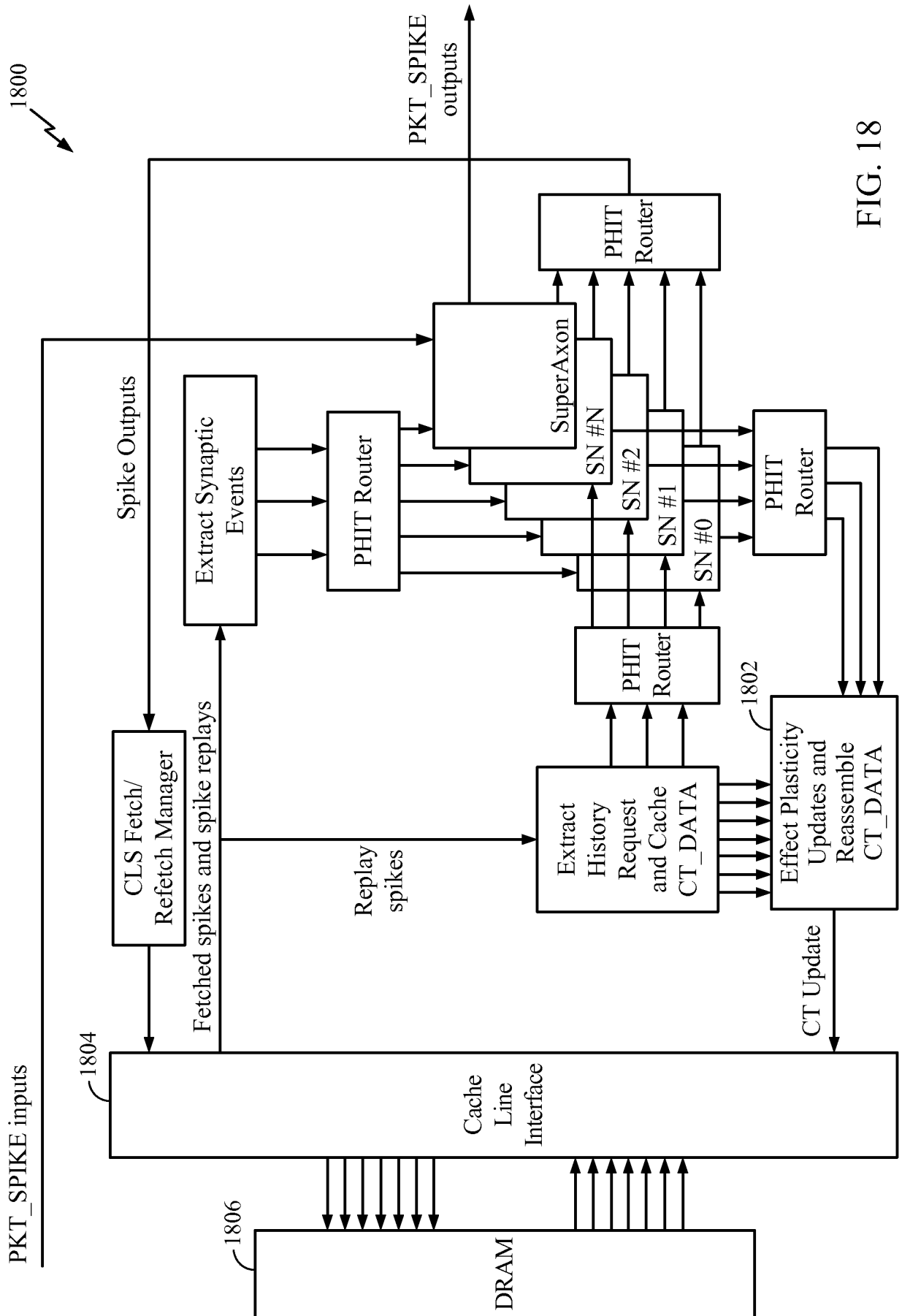


FIG. 18