(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 October 2015 (01.10.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/148189 A2

(51) **International Patent Classification**:
G06N 3/08 (2006.01) *G06N 3/04* (2006.01)

(21) **International Application Number**:
PCT/US2015/021077

(22) **International Filing Date**:
17 March 2015 (17.03.2015)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:
61/969,747 24 March 2014 (24.03.2014) US
14/513,155 13 October 2014 (13.10.2014) US

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

(72) **Inventors**: **ANNAPUREDDY, Venkata Sreekanta Reddy**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **JULIAN, David Jonathan**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **TOWAL, Regan Blythe**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **LIU, Yinyin**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(74) **Agents**: **LENKIN, Alan M.** et al.; Seyfarth Shaw LLP, Suite 3500, 2029 Century Park East, Los Angeles, California 90067-3021 (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**: DIFFERENTIAL ENCODING IN NEURAL NETWORKS

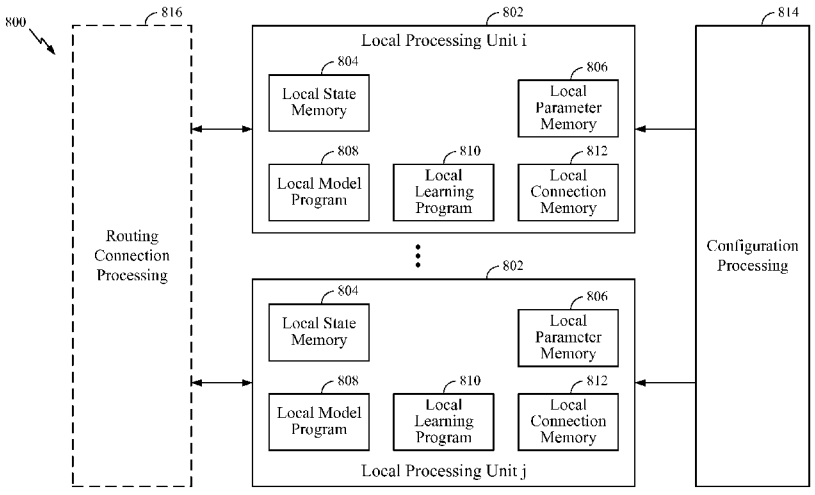

FIG. 8

(57) **Abstract**: Differential encoding in a neural network includes predicting an activation value for a neuron in the neural network based on at least one previous activation value for the neuron. The encoding further includes encoding a value based on a difference between the predicted activation value and an actual activation value for the neuron in the neural network.

WO 2015/148189 A2

DIFFERENTIAL ENCODING IN NEURAL NETWORKS

CROSS REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit under 35 U.S.C. § 119(e) to U.S. Provisional Patent Application No. 61/969,747, entitled "DIFFERENTIAL ENCODING IN NEURAL NETWORKS," filed on March 24, 2014, the disclosure of which is expressly incorporated by reference herein in its entirety.

BACKGROUND

Field

[0002] Certain aspects of the present disclosure generally relate to neural system engineering and, more particularly, to systems and methods for differential encoding in neural networks.

Background

[0003] An artificial neural network, which may comprise an interconnected group of artificial neurons (i.e., neuron models), 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.

SUMMARY

[0004] A method of performing differential encoding in a neural network in accordance with an aspect of the present disclosure includes predicting an activation value for a neuron in the neural network based on at least one previous activation value for the neuron. Such a method further includes encoding a value based on a difference

between the predicted activation value and an activation value for the neuron in the neural network.

[0005] An apparatus for performing differential encoding in a neural network in accordance with an aspect of the present disclosure includes a memory and at least one processor coupled to the memory. The processor(s) is configured to predict an activation value for a neuron in the neural network based on at least one previous activation value for the neuron. The processor(s) is also configured to encode a value based on a difference between the predicted activation value and an activation value for the neuron in the neural network.

[0006] An apparatus for performing differential encoding in a spiking neural network in accordance with another aspect of the present disclosure includes means for predicting an activation value for a neuron in the neural network based on at least one previous activation value for the neuron. Such an apparatus further includes means for encoding a value based on a difference between the predicted activation value and an activation value for the neuron in the neural network.

[0007] A computer program product for performing differential encoding in a spiking neural network in accordance with another aspect of the present disclosure includes a non-transitory computer readable medium having program code encoded thereon. The program code includes program code to predict an activation value for a neuron in the neural network based on at least one previous activation value for the neuron. The program code also includes program code to encode a value based on a difference between the predicted activation value and an activation value for the neuron in the neural network.

[0008] This has outlined, rather broadly, the features and technical advantages of the present disclosure in order that the detailed description that follows may be better understood. Additional features and advantages of the disclosure will be described below. It should be appreciated by those skilled in the art that this disclosure may be readily utilized as a basis for modifying or designing other structures for carrying out the same purposes of the present disclosure. It should also be realized by those skilled in the art that such equivalent constructions do not depart from the teachings of the disclosure as set forth in the appended claims. The novel features, which are believed to

be characteristic of the disclosure, both as to its organization and method of operation, together with further objects and advantages, will be better understood from the following description when considered in connection with the accompanying figures. It is to be expressly understood, however, that each of the figures is provided for the purpose of illustration and description only and is not intended as a definition of the limits of the present disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The features, nature, and advantages of the present disclosure will become more apparent from the detailed description set forth below when taken in conjunction with the drawings in which like reference characters identify correspondingly throughout.

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

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

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

[0013] FIGURE 4 illustrates an example of a positive regime and a negative regime for defining behavior of a neuron model in accordance with certain aspects of the present disclosure.

[0014] FIGURE 5 illustrates an example implementation of designing a neural network using a general-purpose processor in accordance with certain aspects of the present disclosure.

[0015] FIGURE 6 illustrates an example implementation of designing a neural network where a memory may be interfaced with individual distributed processing units in accordance with certain aspects of the present disclosure.

[0016] FIGURE 7 illustrates an example implementation of designing a neural network based on distributed memories and distributed processing units in accordance with certain aspects of the present disclosure.

[0017] FIGURE 8 illustrates an example implementation of a neural network in accordance with certain aspects of the present disclosure.

[0018] FIGURE 9 illustrates a method for performing differential encoding in accordance with aspects of the present disclosure.

DETAILED DESCRIPTION

[0019] The detailed description set forth below, in connection with the appended drawings, is intended as a description of various configurations and is not intended to represent the only configurations in which the concepts described herein may be practiced. The detailed description includes specific details for the purpose of providing a thorough understanding of the various concepts. However, it will be apparent to those skilled in the art that these concepts may be practiced without these specific details. In some instances, well-known structures and components are shown in block diagram form in order to avoid obscuring such concepts.

[0020] Based on the teachings, one skilled in the art should appreciate that the scope of the disclosure is intended to cover any aspect of the disclosure, 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. In addition, the scope of the disclosure is intended to cover such an apparatus or method practiced using other structure, functionality, or structure and functionality in addition to or other than the various aspects of the disclosure set forth. It should be understood that any aspect of the disclosure disclosed may be embodied by one or more elements of a claim.

[0021] 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.

[0022] 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, TRAINING AND OPERATION

[0023] FIGURE 1 illustrates an example artificial neural system 100 with multiple levels of neurons in accordance with certain aspects of the present disclosure. The neural system 100 may have 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 FIGURE 1, although fewer or more levels of neurons may exist in a 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.

[0024] As illustrated in FIGURE 1, each neuron in the level 102 may receive an input signal 108 that may be generated by neurons of a previous level (not shown in FIGURE 1). The signal 108 may represent an input current of the level 102 neuron. This current 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). In some modeling approaches, the neuron may continuously transfer a signal to the next level of neurons. This signal is typically a function of the membrane potential. Such behavior can be emulated or simulated in hardware and/or software, including analog and digital implementations such as those described below.

[0025] 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,

nerve impulse, having an amplitude of roughly 100 mV and a duration of about 1 ms. In a particular embodiment of a neural system having a series of connected neurons (e.g., the transfer of spikes from one level of neurons to another in FIGURE 1), every action potential has basically the same amplitude and duration, and thus, the information in the signal may be represented only by the frequency and number of spikes, or the time of spikes, rather than by the amplitude. The information carried by an action potential may be determined by the spike, the neuron that spiked, and the time of the spike relative to other spike or spikes. The importance of the spike may be determined by a weight applied to a connection between neurons, as explained below.

[0026] 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 FIGURE 1. Relative to the synapses 104, neurons of level 102 may be considered presynaptic neurons and neurons of level 106 may be considered postsynaptic neurons. The synapses 104 may receive output signals (i.e., spikes) from the level 102 neurons and scale those signals according to adjustable synaptic weights $w_1^{(i,i+1)},\ldots, w_P^{(i,i+1)}$ where P is a total number of synaptic connections between the neurons of levels 102 and 106 and i is an indicator of the neuron level. In the example of FIGURE 1, i represents neuron level 102 and i+1 represents neuron level 106. Further, the scaled signals may be combined as an input signal of each neuron in the level 106. Every neuron in the level 106 may generate output spikes 110 based on the corresponding combined input signal. The output spikes 110 may be transferred to another level of neurons using another network of synaptic connections (not shown in FIGURE 1).

[0027] Biological synapses can mediate either excitatory or inhibitory (hyperpolarizing) actions in postsynaptic neurons and can also serve to amplify neuronal signals. Excitatory signals 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 time 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 a 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 a 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.

[0028] 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 alike. 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.

[0029] 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, where synaptic weight changes may relate to changes of the memristor resistance. With nanometer feature-sized memristors, the area of a neuron circuit and synapses may be substantially reduced, which may make implementation of a large-scale neural system hardware implementation more practical.

[0030] 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, where a particular functionality may be based on synaptic weights stored in a memory card currently attached to the neural processor.

[0031] FIGURE 2 illustrates an exemplary diagram 200 of a processing unit (e.g., a neuron or neuron circuit) 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 FIGURE 1. The neuron 202 may receive multiple input signals 204_1-204_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, a conductance, a voltage, a real-valued, and/or a 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 206_1-206_N (W_1-W_N), where N may be a total number of input connections of the neuron 202.

[0032] 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, a conductance, a voltage, a real-valued and/or a complex-valued. The output signal may be 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.

[0033] The processing unit (neuron) 202 may be emulated by an electrical circuit, and its input and output connections may be emulated by electrical connections with synaptic circuits. The processing unit 202 and its input and output connections may also be emulated by a software code. The processing unit 202 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 202 in the computational network may be an analog electrical circuit. In another aspect, the processing unit 202 may be a digital electrical circuit. In yet another aspect, the processing unit 202 may be a mixed-signal electrical circuit with both analog and digital components. The computational network may include 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.

[0034] During the course of training a neural network, synaptic weights (e.g., the weights $w_1^{(i,i+1)},\ldots, w_P^{(i,i+1)}$ from FIGURE 1 and/or the weights 206_1-206_N from FIGURE 2) may be initialized with random values and increased or decreased according to a learning rule. Those skilled in the art will appreciate that examples of the learning rule include, but are not limited to the spike-timing-dependent plasticity (STDP) learning rule, the Hebb rule, the Oja rule, the Bienenstock-Copper-Munro (BCM) rule, etc. In certain aspects, the weights may settle or converge to one of two values (i.e., a bimodal distribution of weights). This effect can be utilized to reduce the number of bits for each synaptic weight, increase the speed of reading and writing from/to a memory storing the synaptic weights, and to reduce power and/or processor consumption of the synaptic memory.

Synapse Type

[0035] In hardware and software models of neural networks, the processing of synapse related functions can be based on synaptic type. Synapse types may be 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 multiple types is that processing can be subdivided. For example, non-plastic synapses may not use 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 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, formulas, or parameters for the synapse's type.

[0036] 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) s structural plasticity (i.e., an amount of delay change) may be a direct function of pre-post spike time difference. Alternatively, structural plasticity 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 synapse delay may change only when a weight change occurs or if weights reach zero but not if they are at a maximum value. However, it may 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

[0037] 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 for 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.

[0038] STDP is a learning process that adjusts the strength of synaptic connections between neurons. 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. On the other hand, 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, and hence the name "spike-timing-dependent plasticity." Consequently, inputs that might be the cause of the postsynaptic neuron's excitation are made even more likely to contribute in the future, whereas inputs that are not the cause of the postsynaptic 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 an insignificant level.

[0039] Because a neuron generally produces an output spike when many of its inputs occur within a brief period (i.e., being cumulative sufficient to cause the output),

the subset of inputs that typically remains includes those that tended to be correlated in time. In addition, because 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.

[0040] The STDP learning rule may effectively adapt a synaptic weight of a synapse connecting a presynaptic neuron to a postsynaptic neuron as a function of time difference between spike time t_{pre} of the presynaptic neuron and spike time t_{post} of the postsynaptic 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 presynaptic neuron fires before the postsynaptic neuron), and decrease the synaptic weight (i.e., depress the synapse) if the time difference is negative (the postsynaptic neuron fires before the presynaptic neuron).

[0041] 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_- $\tau_{sign(\Delta t)}$ 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.

[0042] FIGURE 3 illustrates an exemplary diagram 300 of a synaptic weight change as a function of relative timing of presynaptic and postsynaptic spikes in accordance with the STDP. If a presynaptic neuron fires before a postsynaptic 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 presynaptic and postsynaptic 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.

[0043] As illustrated in the graph 300 in FIGURE 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. In the case of a frame-based input (i.e., an input that 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 postsynaptic 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 to 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

[0044] 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 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.

[0045] 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.

[0046] 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:

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

where α and β are parameters, $w_{m,n}$ is a synaptic weight for the synapse connecting a presynaptic neuron *m* to a postsynaptic 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.

[0047] It should be noted that there is a delay from the time when sufficient input to a postsynaptic neuron is established until the time when the postsynaptic 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.:

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

$$\frac{du}{dt} = a\left(b(v - v_r) - u\right), \tag{4}$$

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

[0048] 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 a 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.

[0049] As illustrated in FIGURE 4, the dynamics of the model 400 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.

[0050] Linear dual-regime bi-dimensional dynamics (for states v and u) may be defined by convention as:

$$\tau_\rho \frac{dv}{dt} = v + q_\rho \tag{5}$$

$$-\tau_u \frac{du}{dt} = u + r, \tag{6}$$

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

[0051] 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.

[0052] 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.

[0053] 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.

[0054] 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:

$$q_\rho = -\tau_\rho \beta u - v_\rho \tag{7}$$

$$r = \delta(v + \varepsilon), \tag{8}$$

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.

[0055] 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.

[0056] The model may be defined to spike when the voltage v reaches a value v_S. Subsequently, the state may be reset at a reset event (which may be one and the same as the spike event):

$$v = \hat{v}_{-} \tag{9}$$

$$u = u + \Delta u\,, \tag{10}$$

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

[0057] 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 to reach a particular state. The close form state solutions are:

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

$$u(t+\Delta t) = \left(u(t) + r\right)e^{-\frac{\Delta t}{\tau_{u}}} - r\,. \tag{12}$$

[0058] Therefore, the model state may be updated only upon events, such as an input (presynaptic spike) or output (postsynaptic spike). Operations may also be performed at any particular time (whether or not there is input or output).

[0059] Moreover, by the momentary coupling principle, the time of a postsynaptic 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:

$$\Delta t = \tau_{\rho} \log \frac{v_f + q_{\rho}}{v_0 + q_{\rho}}\,. \tag{13}$$

[0060] 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:

$$\Delta t_S = \begin{cases} \tau_+ \log \frac{v_S + q_+}{v + q_+} & \text{if} \quad v > \hat{v}_+ \\ \infty & \text{otherwise} \end{cases} \tag{14}$$

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

[0061] 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.

[0062] 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., 1 ms). This does not necessarily utilize 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.

DIFFERENTIAL ENCODING IN NEURAL NETWORKS

[0063] Aspects of the present disclosure are directed to differential encoding in neural networks.

[0064] In some aspects, neural networks learn or solve many inference tasks including object classification, speech recognition, and handwriting recognition. In many applications, the neural networks makes "sense" from a continuous stream of sensory information. For example, and not by way of limitation, a robot (or

smartphone) may use a neural network to extract high-level features or category labels on a sequence of images (i.e., image classification). In such scenarios, the neural network can exploit the temporal structure of the input data stream. Because the data stream does not change much from instance to instance, or changes in predictable ways, e.g., motion predictions, the present disclosure may send differential or difference results rather than sending all data values in each instance. The present disclosure may also be applied to differential encoding for machine learning networks. For example computing Scale-Invariant Feature Transform (SIFT) features on images can use differential encoding of the SIFT values and locations based on differences to prior images or may be based on motion-based forward estimates.

[0065] Neural networks have layers of neurons where the bottom layer represents the raw data, and the higher layers represent features. The bottom layer may be a lower layer in the network, and the layer receiving the outputs from the bottom layer may be a higher layer in the network. For example, the "bottom" layer may be an intermediate hidden level that has had some pre-processing or initial feature extraction, and the "upper" layer may be a layer that receives inputs from that "bottom" layer. When inferring a sensory stream with temporal structure, each neuron may predict activation based on a history of activations for that neuron. In such cases, propagating the activation value to other neurons is less efficient than sending the difference (or error) between the actual activation value and the predicted value based on history.

[0066] Depending on the how good the prediction is, communication between levels of the neural network is decreased. If the communication between the neurons is binary (i.e., spike or no-spike), then a difference/error approach in accordance with the present disclosure propagates fewer spikes through differential encoding. As the predicted values approach 100% accuracy at a layer of neurons, there is less need for computation at the neurons in higher layers. If the neurons are non-binary, then differential encoding uses fewer bits to achieve the same level of precision when compared to sending a full set of activation values.

[0067] The present disclosure may send encoded values, which may be the difference between the predicted activation value and the activation value, between layers in the neural network. Further, there may be occasion in an aspect of the present disclosure to alter the information being sent between layers in the neural network. The

activation value may be a difference value, or may be the activation value itself, or may be other data. The determination of what activation value, difference in activation value, or value in general may be based on a number of factors. These factors include the number of bits of the activation value or difference in activation values, a threshold value used to determine whether any data is sent between layers of the neural network, the activation function used to determine the activation value, receipt of an input to an input neuron, a bit width of the activation value, or other factors. For example, a threshold can be set based upon a number of bits. That is, the difference is sent when the difference exceeds a specific value that depends on a number of bits available for communication for a particular neuron.

[0068] The activation value, as well as the predicted activation value, may be determined using one or more activation functions. One or more of the activation functions may be a non-linear function. The activation function may be implemented using a filter, which may also determine the encoding of the activation value and/or the differentially encoded activation value.

[0069] Sending or other distribution of data within the neural network may be on a continuous basis, a periodic basis, or an intermittent basis. That is, state information may be periodically (intermittently) synchronized across the network. Further, encoding of the activation value and/or the differentially encoded activation value may be delayed from receipt of input data.

[0070] Although differential encoding may reduce the amount of data transmitted within a neural network, the present disclosure also envisions that design options may include sending more data within the network while reducing the computations for encoding or determining the data to be sent. For example, no predictions on activation value may be sent, and actual data may be merely forwarded through the system as the data is received. This approach results in large data throughput and little computation. Design trade-offs may be made between data transmission and data computation within the neural network to satisfy various neural network designs.

[0071] Within a neural network, some of the activation functions for some neurons may change "mode" during operation of the neural network. Further, some neurons may always operate in one mode while other neurons operate in another mode. For

example, some neurons may only send differentially encoded data, while other neurons may send the entire activation value. Some neurons may switch modes during operation, e.g., sending the entire activation value until a certain point, and then sending differentially encoded activation values after that point. The change in the data being transmitted within the neural network may be based on a classification of data within the neural network, or on other factors, including available computational power, transmission reliability, size of the neural network, or other constraints.

[0072] FIGURE 5 illustrates an example implementation 500 of the aforementioned differential encoding using a general-purpose processor 502 in accordance with certain aspects of the present disclosure. Variables (neural signals), synaptic weights, system parameters associated with a computational network (neural network), delays, frequency bin information node state information, bias weight information, connection weight information, and/or firing rate information may be stored in a memory block 504, while instructions executed at the general-purpose processor 502 may be loaded from a program memory 506. In an aspect of the present disclosure, the instructions loaded into the general-purpose processor 502 may comprise code for receiving input events at a node, applying bias weights and connection weights to the input events to obtain intermediate values, determining a node state based on the intermediate values, and computing an output event rate representing a posterior probability based on the node state to generate output events according to a stochastic point process.

[0073] FIGURE 6 illustrates an example implementation 600 of the aforementioned differential encoding where a memory 602 can be interfaced via an interconnection network 604 with individual (distributed) processing units (neural processors) 606 of a computational network (neural network) in accordance with certain aspects of the present disclosure. Variables (neural signals), synaptic weights, system parameters associated with the computational network (neural network) delays, frequency bin information, node state information, bias weight information, connection weight information, and/or firing rate information may be stored in the memory 602, and may be loaded from the memory 602 via connection(s) of the interconnection network 604 into each processing unit (neural processor) 606. In an aspect of the present disclosure, the processing unit 606 may be configured to receive input events at a node, apply bias weights and connection weights to the input events to obtain intermediate values, determine a node state based at least in part on the intermediate values, and compute an

output event rate representing a posterior probability based on the node state to generate output events according to a stochastic point process.

[0074] FIGURE 7 illustrates an example implementation 700 of the aforementioned differential encoding. As illustrated in FIGURE 7, one memory bank 702 may be directly interfaced with one processing unit 704 of a computational network (neural network). Each memory bank 702 may store variables (neural signals), synaptic weights, and/or system parameters associated with a corresponding processing unit (neural processor) 704 delays, frequency bin information, node state information, bias weight information, connection weight information, and/or firing rate information. In an aspect of the present disclosure, the processing unit 704 may be configured to receive input events at a node, apply bias weights and connection weights to the input events to obtain intermediate values, determine a node state based at least in part on the intermediate values, and compute an output event rate representing a posterior probability based on the node state to generate output events according to a stochastic point process.

[0075] FIGURE 8 illustrates an example implementation of a neural network 800 in accordance with certain aspects of the present disclosure. As illustrated in FIGURE 8, the neural network 800 may have multiple local processing units 802 that may perform various operations of methods described herein. Each local processing unit 802 may comprise a local state memory 804 and a local parameter memory 806 that store parameters of the neural network. In addition, the local processing unit 802 may have a local (neuron) model program (LMP) memory 808 for storing a local model program, a local learning program (LLP) memory 810 for storing a local learning program, and a local connection memory 812. Furthermore, as illustrated in FIGURE 8, each local processing unit 802 may be interfaced with a configuration processor unit 814 for providing configurations for local memories of the local processing unit, and with a routing connection processing unit 816 that provide routing between the local processing units 802.

[0076] According to certain aspects of the present disclosure, each local processing unit 802 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.

[0077] In an aspect of the present disclosure, a general framework for predictive differential encoding in neural networks is as follows. An artificial neuron receives an input x(t) and emits an output y(t), where t represents time. The output y(t) may be a non-linear function of x(t), such as the sigmoid function:

$$y(t) = \sigma(x(t)) = e^x/(1 + e^x). \quad (15a)$$

or the rectifier nonlinearity function

$$y(t) = \max(0, x(t)) \quad (15b)$$

[0078] The neuron may emit a binary output y(t) stochastically by using the sigmoid expression, or other expression, as a probability that the output y(t) is one.

[0079] The input x(t) may be a weighted linear combination of the output of other neurons:

$$x_i(t) = \Sigma w_{ij} y_j(t) \quad (16)$$

[0080] where w_{ij} represents a weight for the ith and jth neurons and j is the index of all neurons connected to the ith neuron.

[0081] An aspect of the present disclosure allows the predictive differential encoding framework to work with an arbitrary artificial neuron model. Adding a state variable to the artificial neurons allows the neuron to maintain a history log. A function s(t) denotes the state-variable or state-variables. Each neuron keeps track of the history through the state variables, and predicts the input x^(t) it is going to receive and the output $\hat{y}(t)$ it is going to emit. The predictions reduce the amount of communication between the neurons (i.e., each neuron now emits only the error in prediction $\delta y(t) = y(t) - \hat{y}(t)$ and not the actual output y(t)). The state variable stores the input history for deterministic models and also the output history in the case of stochastic models.

[0082] Because the neurons emit the error in prediction, they now receive a weighted combination of the errors in prediction z(t) instead of x(t), as in equation (16):

$$z_i(t) = \Sigma w_{ij}\delta y_j(t). \quad (17)$$

[0083] If z(t) is exactly equal to the error in the prediction of the input x(t), then x(t) is reconstructed exactly by:

$$x(t) = x\hat{}(t) + z(t). \quad (18)$$

[0084] The condition to satisfy where $\delta x(t) = x(t) - x\hat{}(t)$ is:

$$z(t) = \delta x(t). \quad (19)$$

[0085] In equation 16, z(t) is what the neuron receives whereas δx(t) is what we want the neuron to receive. If equation (19) is satisfied, then the predictive differential encoding method is an exact but alternative implementation compared to the standard method of emitting actual output values. Even if equation (19) is not satisfied, the predictive differential encoding method gives an approximate implementation.

[0086] If the same linear functions are used for predicting x(t) and y(t), then equation (19) is satisfied. The predictive differential encoding method of the present disclosure becomes an alternative but exact implementation method. If the prediction is linear, then the approximation should be exact. If the prediction is non-linear, the approximation may not be exact.

[0087] The neuron can store the entire history or only a partial history. Let L represent an amount of history a neuron stores (i.e., the neuron keeps track of the inputs and outputs over the last L time steps). This makes the state variable function:

$$s(t) = \{x(t-1), y(t-1), x(t-2), y(t-2), \bullet\bullet\bullet, x(t-L), y(t-L)\}. \quad (20)$$

[0088] If the neuron input-output relationship is deterministic (i.e., if y(t) is a deterministic function of x(t)), then it is sufficient to store only the input history. A deterministic algorithm computes a mathematical function with a unique output value for any given input, and the algorithm produces this particular value as output.

[0089] If the input-output relationship is stochastic, as in Restricted Boltzmann Machines (RBMs) or Deep Belief Nets (DBNs), then the output history is also stored. A stochastic process, unlike the deterministic process, has some uncertainty: even if the initial condition (or starting point) is known, there are several (often infinitely many) directions in which the process may evolve.

[0090] The present disclosure also contemplates using the differential encoding framework on a regional and/or global level. For example, a motion vector estimate of the image can be used to determine the average vector change either for the entire image or for regional portions of the image. The global or local information may be provided to all of the neurons and then both the pre and postsynaptic neuron could use these regional/global differences to make better predictions. Further, the postsynaptic neurons may provide differential feedback. In such an aspect, presynaptic neurons may use post-synaptic neuron differential outputs for state estimation. This may reduce the amount of communication between layers in the neural network.

[0091] Given the history s(t), each neuron predicts the input and output at time t using a linear filter:

$$\hat{x}(t) = \Sigma\alpha_k x(t-k) \text{ for } k = 1 \text{ to } L \quad (21)$$

$$\hat{y}(t) = \Sigma\alpha_{ky}(t-k) \text{ for } k = 1 \text{ to } L. \quad (22)$$

[0092] Within this predictive framework, equation (19) is satisfied. The filter coefficients $\alpha 1, \alpha 2, \bullet\bullet\bullet, \alpha L$ can be learned over time or chosen *a priori*. For example, and not by way of limitation, if L is 1 and $\alpha 1 = 1$ (i.e., each neuron uses the previous time period input and output as the best predictions), then:

$$x\hat{}(t) = x(t-1) \quad (23a)$$

$$\hat{y}(t) = y(t-1). \quad (23b)$$

As another example, if $L = 2$, $\alpha 1 = 2$, $\alpha 2 = -1$ (i.e., each neuron assumes the input is changing in a linear fashion, then:

$$x\hat{}(t) = 2x(t-1) - x(t-2) = x(t-1) + (x(t-1) + x(t-2)) \quad (24a)$$

$$\hat{y}(t) = 2y(t-1) - y(t-2) = y(t-1) + (y(t-1) + y(t-2)) \qquad (24b)$$

[0093] Equation (19) is not satisfied if different neurons use different prediction filters. To arrive at an exact prediction with the differential encoding method of the present disclosure, it may be desired to use the same filter over the entire neural network.

[0094] Each neuron may use different x-filters and y-filters for predicting x^(t) and y^(t), respectively. These prediction filters can be the same or different for different neurons. For a neuron's reconstruction of 'actual input' to be exact, the x-filter should match with all y-filters of all the fan-in neurons. One method to ensure the match is by using the same prediction filter throughout the network. In a layered neural network, another method to achieve the matching is by using the same x-filter for all neurons in one layer, and using the same as y-filter for all neurons in the previous layer.

[0095] These prediction filters can be fixed or adaptive, and can be linear or non-linear. In the case on non-linear filters, a prediction for each input synapse is desired. For linear filters, joint predictions can be provided. In one configuration, the baseline solution includes a single fixed filter for all neurons. Another solution estimates the filter coefficient values online. In yet another aspect, filters may be configured or even optimized for different environments, such as a set of filters for indoors versus outdoors, or static versus moving. These filters may be determined by having a codebook of pre-defined filters, a method to determine the environment, and choosing a specific filter based on the environment. In another aspect, the filters are selected based on an output of the classifier (i.e., neural network).

[0096] In one configuration, the pre-defined filters are exponential in shape. The exponential shape has a decaying factor, thereby forcing more delta updates. In one example, a .9 coefficient is supplied. The exponential shape may reduce or even eliminate instability and long term error propagation. The exponential shape also allows one-step updates to a future value so that updates only occur when non-zero inputs are received. It should be noted that there is a tradeoff between decay factor and bit rate. That is, a higher decay factor will result in more communications, whereas a lower decay rate results in fewer transmissions. In one aspect, different neurons may

use different decay factors for the exponential distribution. In another aspect, the filters are learned online. For example, a robot may use a high decay factor if it is moving fast, and a low decay factor if it is standing still or moving slowly.

[0097] Differential encoding saves on the resources for communication between neurons. However, the differential encoding also adds overhead. Additional memory stores the state variable or the history of input/output values. Additional computation computes the predictions and error in predictions. The amount of increase may be reduced some by using an exponential filter shape. Thus, there is a trade-off between the benefits of differential encoding versus. additional overhead.

[0098] Differential encoding may be employed for only a subset of neurons. For example, consider the scenario where the neural network is simulated using multiple cores (or machines), and different cores simulate different neurons. The cost of communication is higher for those neurons that communicate across the cores or machines (i.e., these neurons have input or output synapses connecting neurons in other cores). In this scenario, differential encoding can be employed only for the neurons communicating across the cores or machines. Moreover, neurons that change frequently may not be good candidates for differential encoding. Different neurons can also use different bit widths for communication or even different filters. In yet another aspect, neurons can change modes where in one mode they send differential updates whereas in another mode they send actual results. The mode change can be based on a trigger, for example, based on whether classification results (i.e., output from the neural network) are satisfactory.

[0099] Instead of each neuron predicting input and output, a collection of neurons can jointly predict their collective inputs and outputs. Specifically, in a layered neural network, each layer of neurons can jointly predict the vector input and the vector output based on their joint history of vectors inputs and outputs. The linear predictive framework is naturally extended to the vector input/output scenario by replacing scalar filter coefficients with matrices, i.e., $\alpha 1, \alpha 2, \cdots, \alpha L$ are now matrices.

[00100] An example application where the joint prediction has an advantage over individual prediction is the vision application of inference over a video. Consider the scenario where a person or an object is moving in an otherwise static environment. A

layered neural network, such as Deep Convolutional Network (DCN) can be used, where a collection of neurons in a layer represents a spatial response map. In this case, the filter matrices are selected based on motion vectors from image to image. These motion vectors can be obtained bottom-up from standard motion estimation techniques available from the video compression literature. Or the motion vectors can be obtained top-down from the output of the DCN. Note that a DCN can be trained to predict objects and their locations in images. The neuron can predict based on additional global inputs, such as motion vectors, as well as feedback from the neurons to which it is sending information.

[00101] FIGURE 9 illustrates a method 900 for performing differential encoding in a neural network in accordance with aspects of the present disclosure. In block 902, an activation value is predicted for a neuron in the neural network, based on at least one previous activation value for the neuron. In block 904, a value is encoded based on a difference between the predicted activation value and an activation value for the neuron in the neural network.

[00102] In one configuration, a method for differential encoding includes a means for predicting an activation value for a neuron and means for encoding an error. In one aspect, the predicting means and/or the encoding means may be the general-purpose processor 502, program memory 506, memory block 504, memory 602, interconnection network 604, processing units 606, processing unit 704, local processing units 802, and or the routing connection processing units 816 configured to perform the functions recited. In another configuration, the aforementioned means may be any module or any apparatus configured to perform the functions recited by the aforementioned means.

[00103] The neural network as described in the present disclosure may be any type of neural network, including a multi-layer perceptron network, a deep convolutional network, a deep belief network, and a recurrent neural network, etc. Further, although described with respect to a neuron predicting inputs and outputs for itself based on history, and propagating only errors in that neuron's outputs, a neuron may use the errors and predictions of other neurons to predict its own inputs and outputs.

[00104] 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. Generally, where there are operations illustrated in the figures, those operations may have corresponding counterpart means-plus-function components with similar numbering.

[00105] 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. Additionally, "determining" may include receiving (e.g., receiving information), accessing (e.g., accessing data in a memory) and the like. Furthermore, "determining" may include resolving, selecting, choosing, establishing and the like.

[00106] 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.

[00107] 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).

[00108] 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, erasable programmable read-only memory (EPROM), electrically erasable programmable read-only memory (EEPROM), 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.

[00109] 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.

[00110] The functions described herein and in Appendix A and Appendix B 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.

[00111] 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, random access memory (RAM), flash memory, read only memory (ROM), programmable read-only memory (PROM), erasable programmable read-only memory (EPROM), electrically erasable programmable Read-only memory (EEPROM), 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.

[00112] 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. Although the various components discussed may be described as having a specific location, such as a local component, they may also be configured in various ways, such as certain components being configured as part of a distributed computing system.

[00113] 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 comprise one or more neuromorphic processors for implementing the neuron models and models of neural systems described herein. As another alternative, the processing system may be implemented with an application specific integrated circuit (ASIC) 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 field programmable gate arrays (FPGAs), programmable logic devices (PLDs), 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.

[00114] 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.

[00115] 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. In addition, 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.

[00116] 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.

[00117] 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 user terminal and/or base station 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 user terminal and/or base station 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.

[00118] 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

WHAT IS CLAIMED IS:

1. A method of performing differential encoding in a neural network, comprising:

predicting an activation value for a neuron in the neural network based at least in part on at least one previous activation value for the neuron; and

encoding a value based at least in part on a difference between the predicted activation value and an activation value for the neuron in the neural network.

2. The method of claim 1, further comprising sending the encoded value between layers of the neural network.

3. The method of claim 2, in which the sent encoded value is at least one of the difference between the predicted activation value and the activation value and a thresholded difference between the predicted activation value and the activation value.

4. The method of claim 3, in which the sent encoded value is selected based at least in part on a number of bits of the encoded value.

5. The method of claim 1, in which the activation value is based at least in part on a nonlinear function.

6. The method of claim 1, in which predicting the activation value is performed based at least in part on receipt of an input.

7. The method of claim 1, further comprising encoding the value based at least in part on a bit width of the value.

8. The method of claim 1, in which the encoding is performed based at least in part on a neural network output based trigger.

9. The method of claim 1, in which the encoding is performed intermittently.

10. The method of claim 1, in which the encoding is delayed with respect to an input to the neural network.

11. The method of claim 1, in which the encoding is further based at least in part on an output of the neural network.

12. The method of claim 1, in which the at least one previous activation value comprises an input history when an input-output relationship is deterministic.

13. The method of claim 1, in which the at least one previous activation value comprises an input history and an output history when an input-output relationship is stochastic.

14. The method of claim 1, further comprising computing the predicted activation value based at least in part on a predicted input value.

15. The method of claim 14, further comprising computing an actual input value by combining the encoded value with the predicted input value.

16. The method of claim 14, in which computing the predicted input value and the predicted activation value comprises using a linear combination of a plurality of previous input values and a plurality of previous activation values for the neuron.

17. The method of claim 1, in which the predicted activation value for the neuron is based at least in part on a state for the neuron and an input to the neuron.

18. The method of claim 17, in which the state for the neuron is updated based on at least one of a previous state, an input value, an output value, a predicted activation value, and a targeted activation value.

19. The method of claim 17, in which the state for the neuron comprises at least one of an input history, an output history, a predicted activation value history, and a targeted activation value history.

20. The method of claim 19, in which the predicting is based at least in part on a state of another neuron.

21. The method of claim 1, in which the predicted activation value is based at least in part on a linear combination of a plurality of previous actual activation values or a linear combination of previous input values.

22. The method of claim 1, in which predicting the activation value comprises using an additional value provided to the neuron.

23. The method of claim 22, further comprising computing the additional value based at least in part on image motion estimation.

24. The method of claim 22, in which the additional value comprises a feedback signal from another neuron.

25. An apparatus for performing differential encoding in a neural network, comprising:

a memory; and

at least one processor coupled to the memory, the at least one processor being configured:

to predict an activation value for a neuron in the neural network based at least in part on at least one previous activation value for the neuron; and

to encode a value based at least in part on a difference between the predicted activation value and an activation value for the neuron in the neural network.

26. An apparatus for performing differential encoding in a spiking neural network, comprising:

means for predicting an activation value for a neuron in the neural network based at least in part on at least one previous activation value for the neuron; and

means for encoding a value based at least in part on a difference between the predicted activation value and an activation value for the neuron in the neural network.

27. A computer program product for performing differential encoding in a spiking neural network, comprising:

a non-transitory computer readable medium having encoded thereon program code, the program code comprising:

program code to predict an activation value for a neuron in the neural network based at least in part on at least one previous activation value for the neuron; and

program code to encode a value based at least in part on a difference between the predicted activation value and an activation value for the neuron in the neural network.

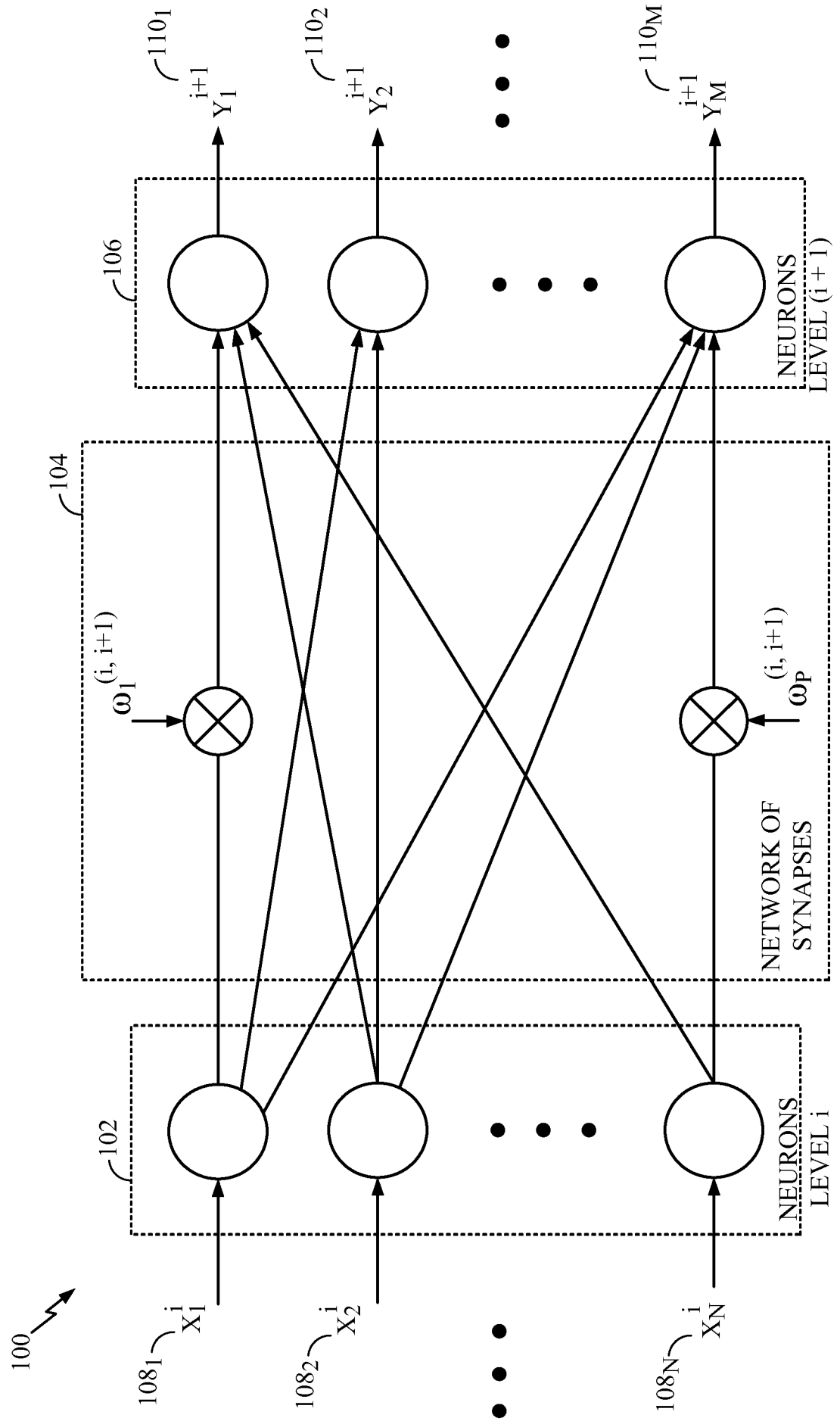
100
108_1 X_1^i
108_2 X_2^i
108_N X_N^i
102
NEURONS LEVEL i
104
$\omega_1^{(i, i+1)}$
$\omega_P^{(i, i+1)}$
NETWORK OF SYNAPSES
106
NEURONS LEVEL (i + 1)
110_1 Y_1^{i+1}
110_2 Y_2^{i+1}
110_M Y_M^{i+1}

FIG. 1

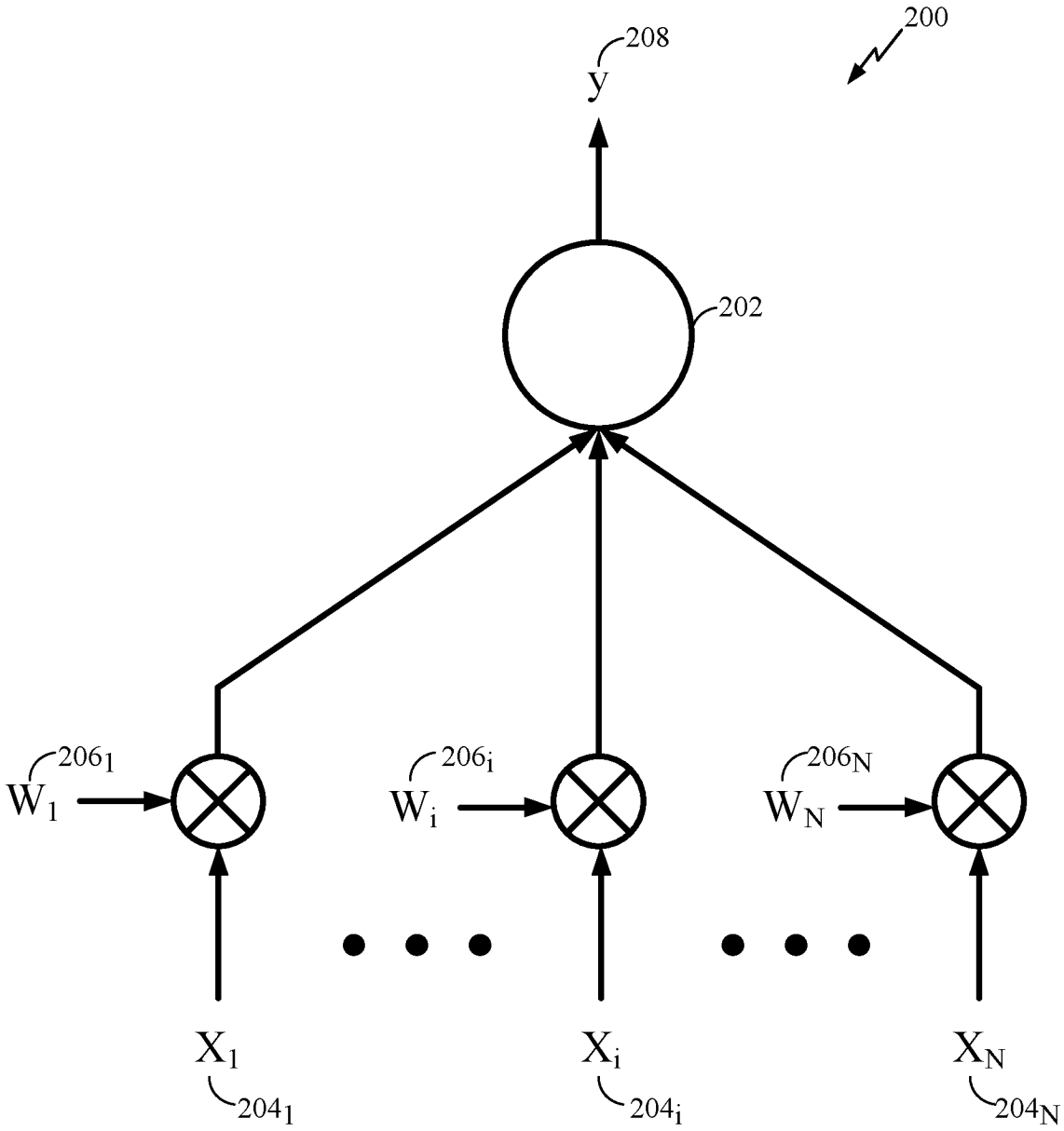

FIG. 2

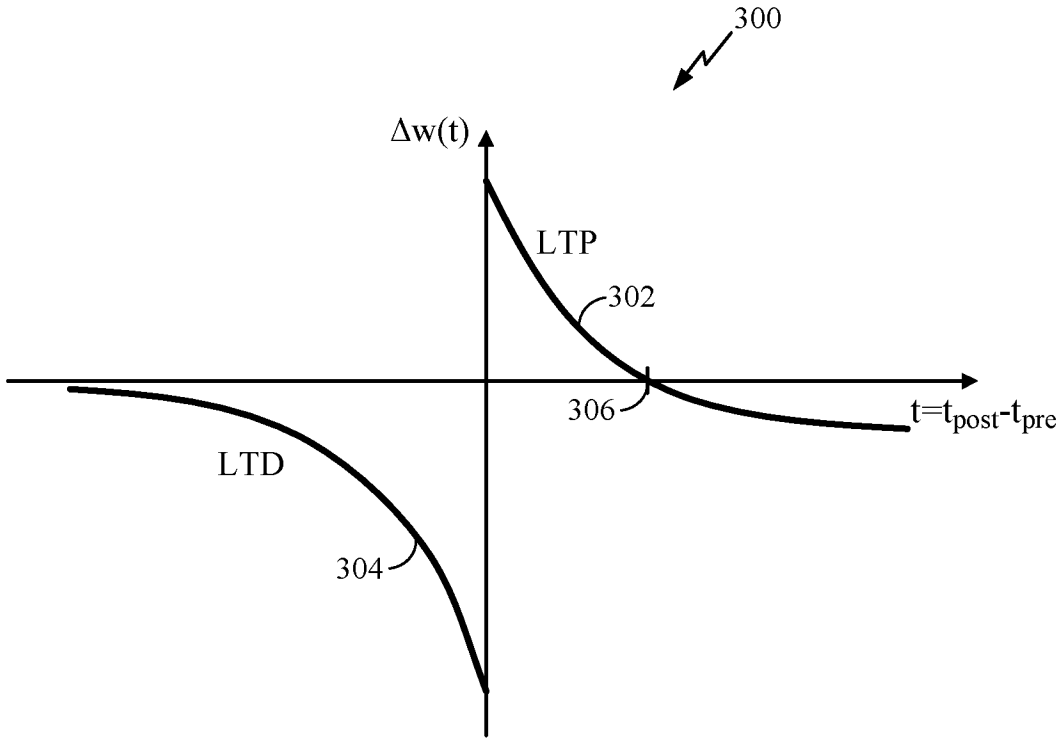

FIG. 3

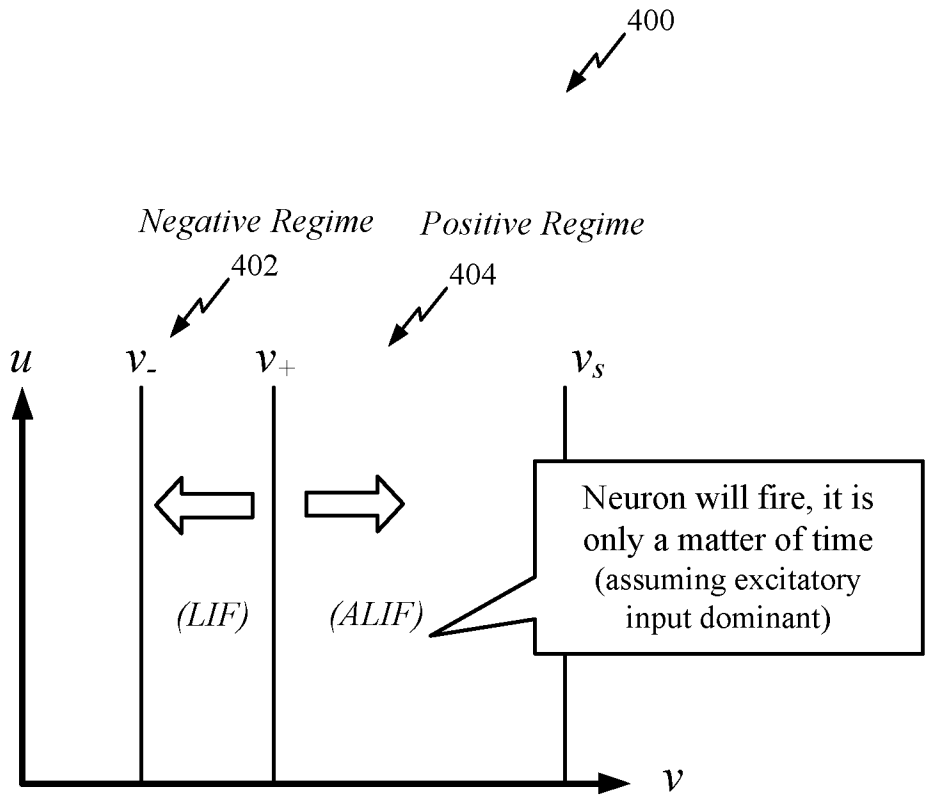
400
Negative Regime
Positive Regime
402
404
u
v-
v+
vs
(LIF)
(ALIF)
Neuron will fire, it is only a matter of time (assuming excitatory input dominant)
v

FIG. 4

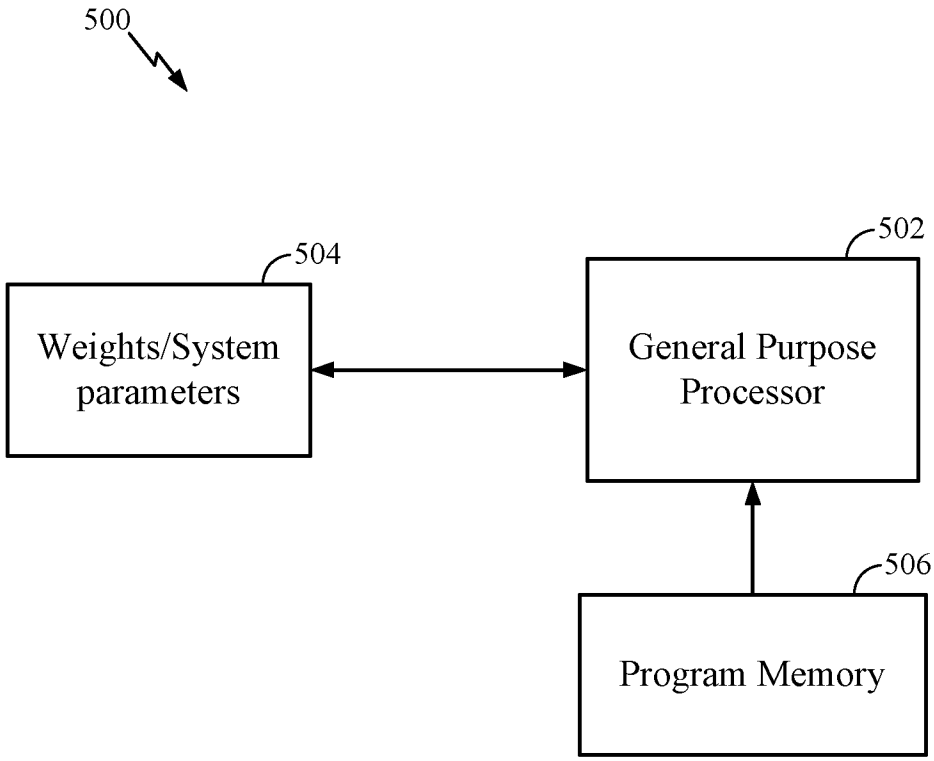
500
504
Weights/System parameters
502
General Purpose Processor
506
Program Memory

FIG. 5

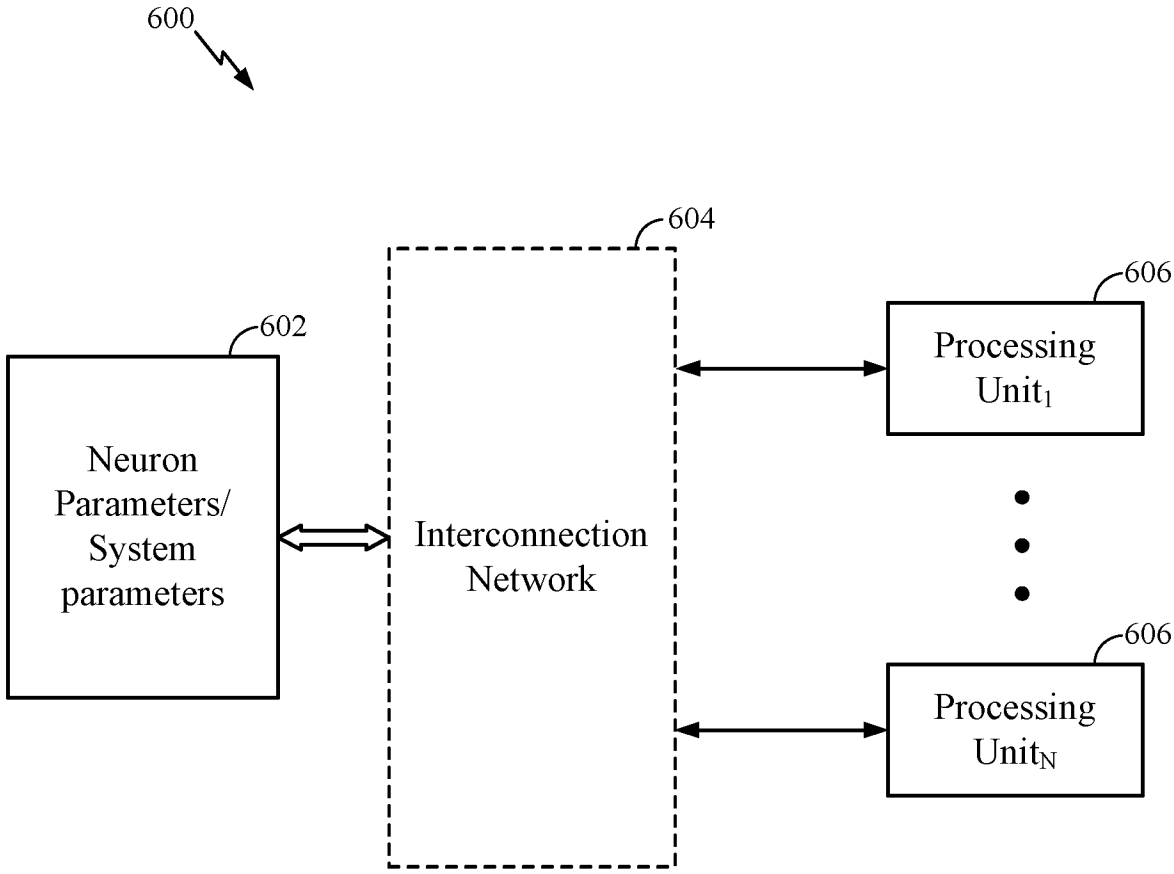
600
604
606
602
Neuron Parameters/ System parameters
Interconnection Network
Processing Unit1
Processing UnitN
606

FIG. 6

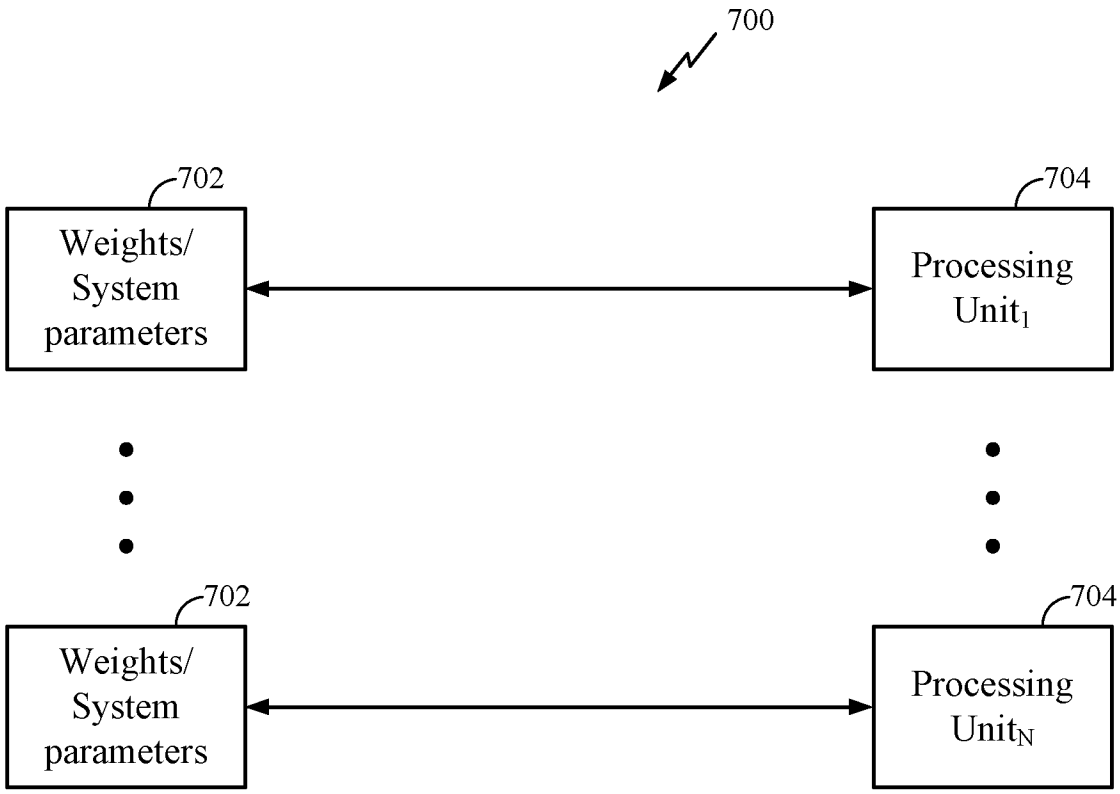
700
702
Weights/
System
parameters
704
Processing
Unit1
702
Weights/
System
parameters
704
Processing
UnitN

FIG. 7

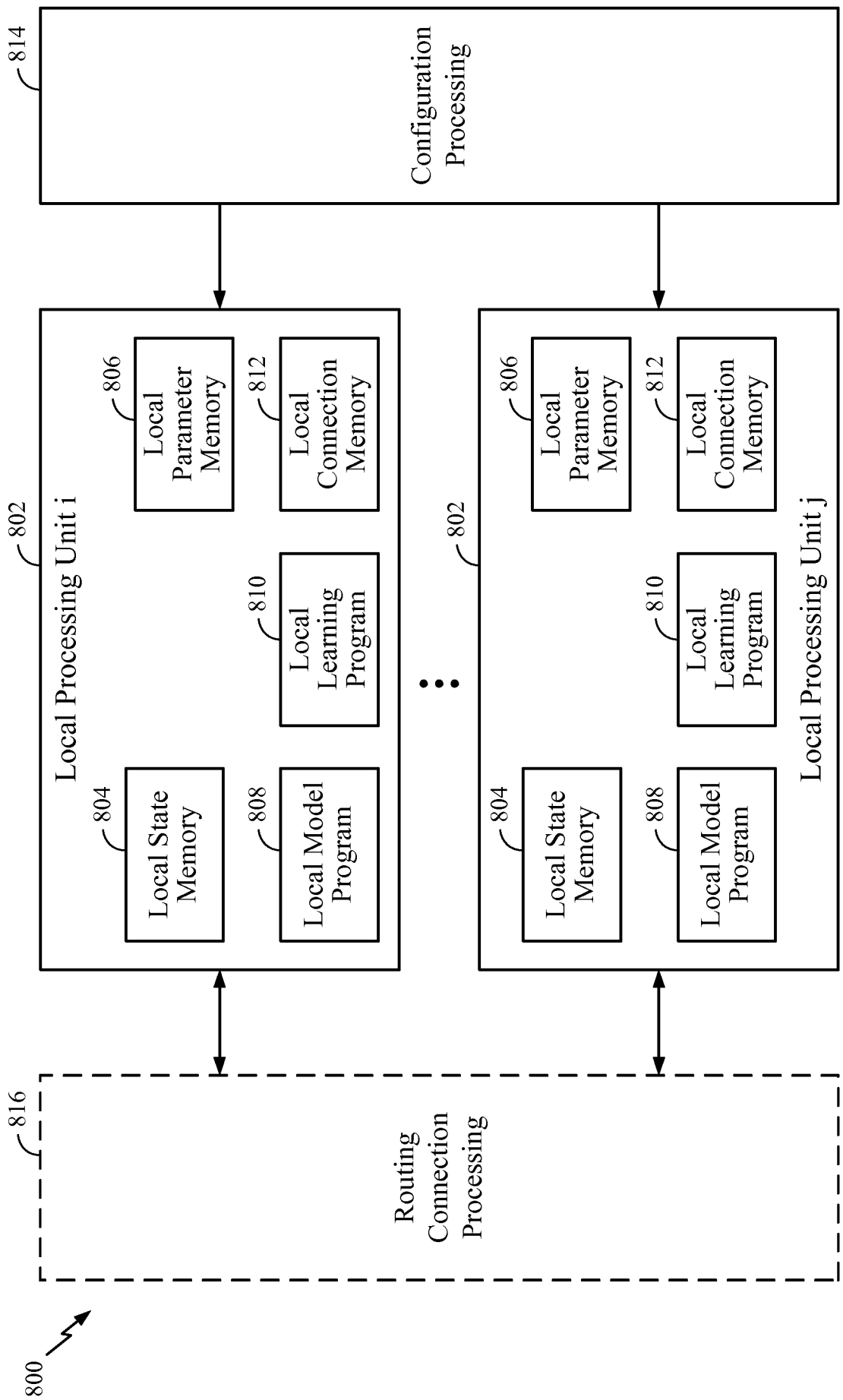
800
816
Routing Connection Processing
802
Local Processing Unit i
804
Local State Memory
806
Local Parameter Memory
808
Local Model Program
810
Local Learning Program
812
Local Connection Memory
802
Local Processing Unit j
804
Local State Memory
806
Local Parameter Memory
808
Local Model Program
810
Local Learning Program
812
Local Connection Memory
814
Configuration Processing

FIG. 8

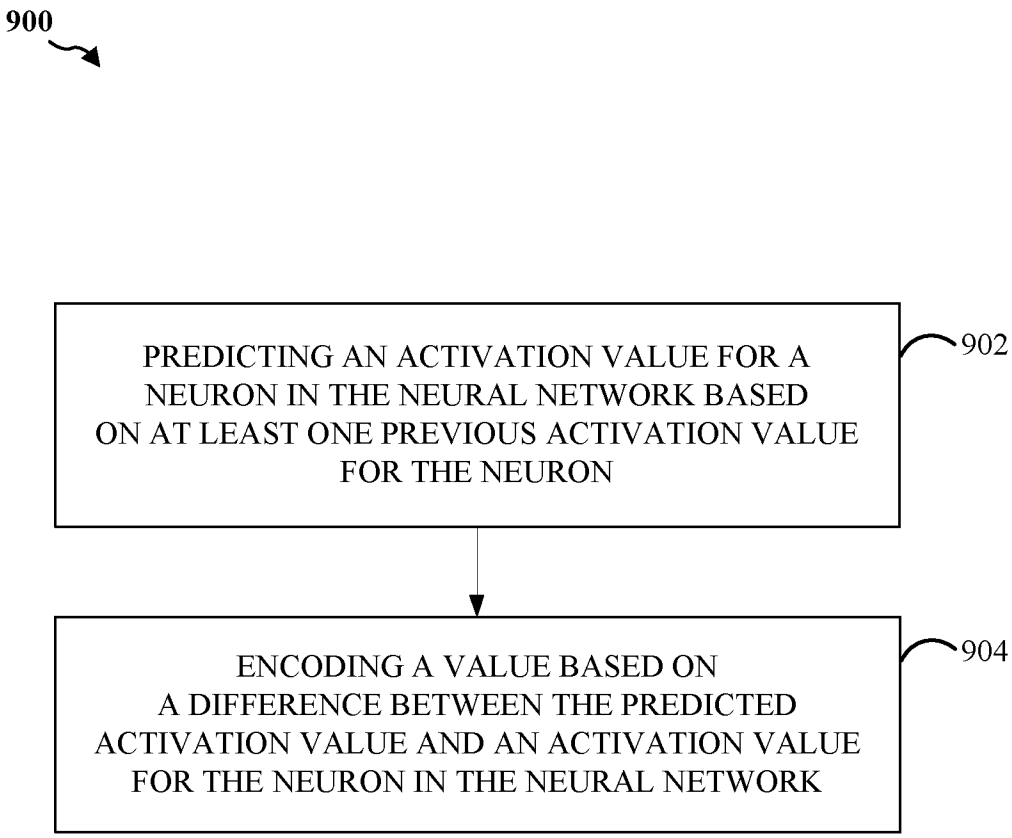
900
PREDICTING AN ACTIVATION VALUE FOR A NEURON IN THE NEURAL NETWORK BASED ON AT LEAST ONE PREVIOUS ACTIVATION VALUE FOR THE NEURON
902
ENCODING A VALUE BASED ON A DIFFERENCE BETWEEN THE PREDICTED ACTIVATION VALUE AND AN ACTIVATION VALUE FOR THE NEURON IN THE NEURAL NETWORK
904

FIG. 9