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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 January 2022 (27.01.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/018548 A1

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

(21) International Application Number:
PCT/IB2021/056026

(22) International Filing Date:
06 July 2021 (06.07.2021)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/054,247 21 July 2020 (21.07.2020) US

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, New York, New York 10504 (US).

(71) Applicants *(for MG only)*: **IBM ISRAEL SCIENCE AND TECHNOLOGY LTD.** [IL/IL]; Haifa University Campus, 165 Aba Khoushy Ave, Mount Carmel, 3498825 Haifa (IL). **IBM (CHINA) INVESTMENT COMPANY LTD.** [CN/CN]; 25/F, Pangu Plaza, No.27, Central North 4th Ring Road, Chaoyang District, Beijing 100101 (CN).

(72) Inventors: **BOHNSTINGL, Thomas**; IBM Research GmbH, Saeumerstrasse 4, Rueschlikon, 8803 Zurich (CH). **WOZNIAK, Stanislaw**; IBM Research GmbH, Saeumerstrasse 4, Rueschlikon, 8803 Zurich (CH). **PANTAZI, Angeliki**; IBM Research GmbH, Saeumerstrasse 4, Rueschlikon, 8803 Zurich (CH). **ELEFTHERIOU, Evangelos Stavros**; IBM Research GmbH, Saeumerstrasse 4, Rueschlikon, 8803 Zurich (CH).

(54) Title: ONLINE TRAINING OF NEURAL NETWORKS

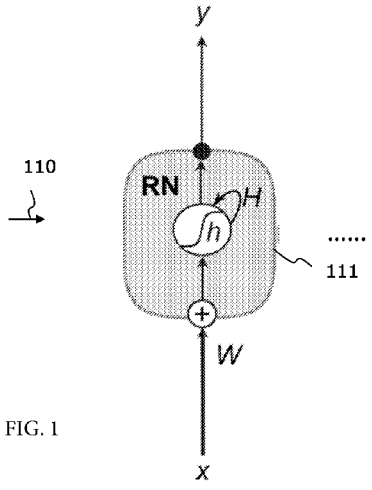
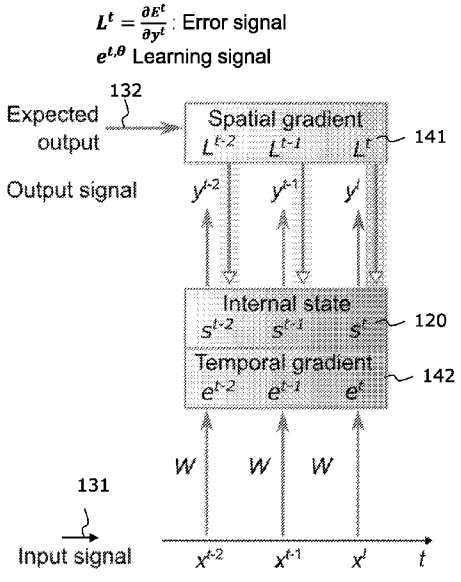


FIG. 1



100

(57) **Abstract:** A computer-implemented method for training parameters of a recurrent neural network (100) is provided. The network (100) comprises one or more layers (110) of neuronal units (111). Each neuronal unit (111) has an internal state (120), which may also be denoted as unit state (120). The method comprises providing training data comprising an input signal (131) and an expected output signal (132) to the recurrent neural network (100). The method further comprises computing, for each neuronal unit (111), a spatial gradient component (141) and computing, for each neuronal unit (111), a temporal gradient component (142). The method further comprises updating the temporal and the spatial gradient component (141) for each neuronal unit (111) at each time instance of the input signal (131). The computing of the spatial and the gradient component (141) may be performed independently from each other. A neural network (100) and a related computer program product are also provided.

WO 2022/018548 A1

[Continued on next page]

(74) **Agent: GILBOA, Eyal**; IBM Israel Science And Technology Ltd., Haifa University Campus, 165 Aba Khoushy Ave, Mount Carmel, 3498825 Haifa (IL).

(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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, 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, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, 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).

Published:

— *with international search report (Art. 21(3))*

— *in black and white; the international application as filed contained color or greyscale and is available for download from PATENTSCOPE*

ONLINE TRAINING OF NEURAL NETWORKS

CROSS-REFERENCE TO RELATED APPLICATIONS

[001] The present application is a non-provisional of U.S. Provisional Application 63/054247, "ONLINE TRAINING OF RECURRENT NEURAL NETWORKS," which was filed 21 July 2020, hereby incorporated by reference in its entirety for all purposes.

BACKGROUND

[002] The invention is notably directed to a computer-implemented method for training of neural networks, in particular recurrent neural networks.

[003] The invention further concerns a related neural network and a related computer program product.

[004] Over recent years, the number of applications utilizing artificial neural networks (ANNs) has grown rapidly. Especially in tasks such as speech recognition, language translation or building neural computers, recurrently connected ANNs, so-called RNNs, have demonstrated astounding performance levels.

[005] Recurrent neural networks (RNNs) have played an important role in advances of artificial intelligence in recent years. One known approach for training RNNs is gradient-based training utilizing backpropagation of errors through time (BPTT).

[006] BPTT has however limitations, as it needs to keep track of all past activities by unrolling the network in time, which can become very deep with increasing input sequence length. For example, a two-second-long spoken input sequence with 1 ms time steps will result in a 2000-layer-deep unrolled network.

[007] Accordingly, propagating errors backwards in time may lead to system-locking problems, rendering BPTT rather unusable for online learning scenarios. Variants that enable online training have recently regained the attention of the research community. One known

approach focuses on approximating BPTT through online algorithms. Another approach takes inspiration from biology and investigates spiking neural networks (SNNs).

[008] Accordingly, there remains a need for advantageous methods for training neural networks, in particular for online training.

SUMMARY

[009] According to an aspect, the invention is embodied as a computer-implemented method for training a neural network. The network comprises one or more layers of neuronal units. Each neuronal unit has an internal state, which may also be denoted as unit state. The method comprises providing training data comprising an input signal and an expected output signal to the neural network. The method further comprises computing, for each neuronal unit, a spatial gradient component and computing, for each neuronal unit, a temporal gradient component. The method further comprises updating the temporal and the spatial gradient component for each neuronal unit at each time instance of the input signal.

[0010] Accordingly, methods according to embodiments of the invention are based on a separation of spatial and temporal gradient components. This may facilitate a more profound understanding of feedback mechanisms. Furthermore, it may facilitate an efficient implementation on hardware accelerators such as memristive arrays. Methods according to embodiments of the invention may be in particular used for online training. Methods according to embodiments of the invention may be in particular used to train training parameters of the neural network.

[0011] Methods according to embodiments of the invention process as input signals temporal data. Temporal data may be defined as data that represents a state or a value in time or in other words as data relating to time instances. The input signals may be in particular continuous input

data streams. The input signal is processed by the neural network at time instances or in other words time steps.

[0012] According to an embodiment, the computing of the spatial and the temporal gradient component is performed independently from each other. This has the advantage that these gradient components may be computed in parallel which reduces the computational time.

[0013] According to embodiments the spatial gradient components establish learning signals and the temporal gradient components eligibility traces.

[0014] Methods according to embodiments of the invention may be in particular used for low complexity devices such as Internet of Things (IoT) devices as well as edge Artificial Intelligence (AI)- devices.

[0015] According to embodiments, the method comprises updating training parameters of the neural network at specific or predefined time instances, in particular at each time instance. The updating may be performed in particular as a function of the spatial and the temporal gradient components.

[0016] The training parameters that may be trained according to embodiments encompass in particular input weights and/or recursive weights of the neuronal units. By updating the training parameters at each time instance, the neuronal units learn at each time instance or in other words at each time step.

[0017] According to embodiments, the spatial gradient components are based on connectivity parameters of the neural network, for example the connectivity of the individual neuronal units. According to embodiments, the connectivity parameters describe in particular parameters of the architecture of the neural network. According to embodiments, the connectivity parameters may be defined as number or the set of transmission lines that allow for information exchange between individual neuronal units. According to embodiments, the spatial gradient components are components which take into consideration the spatial aspects of the neural network, in particular interdependencies between the individual neuronal units at each time instance.

[0018] According to embodiments the temporal gradient components are based on the temporal dynamics of the neuronal units. According to embodiments, temporal gradient components are components which take into consideration the temporal dynamics of the neuronal units, in particular the temporal evolution of the internal states/unit states.

[0019] According to embodiments, the method comprises computing, at each time instance, a spatial gradient component for each of the one or more layers and computing, at each time instance, for each of the one or more layers, a temporal gradient component. Hence at each time instance/time step the method computes a temporal gradient component and a spatial gradient component per layer. The spatial gradient components/the learning signal may be specific for each layer and propagates from the last layer to the input layer without going back in time, i.e. it represents the spatial gradient passing through the network architecture.

[0020] According to embodiments, each layer may compute its own temporal gradient component/eligibility trace, which is solely dependent on contributions of the respective layer, i.e. it represents the temporal gradient passing through time for the same layer. According to embodiments, the spatial gradient components may be shared for two or more layers.

[0021] According to embodiments, the method may be used for single layer as well as multi-layer networks.

[0022] According to embodiments, the method may be applied to recurrent neural networks, spiking neural networks and hybrid networks, comprising or consisting of units that have a unit state and units that do not have a unit state

[0023] According to embodiments, the method or parts of the method may be implemented on neuromorphic hardware, in particular on arrays of memristive devices.

[0024] For shallows networks, methods according to embodiments of the invention may maintain equivalent gradients as the backpropagation through time (BPTT) technique

[0025] According to an embodiment of another aspect of the invention a neural network, in particular a recurrent neural network is provided. The neural network comprises one or more layers of neuronal units. Each neuronal unit has an internal state, which may also be denoted as unit state. The neural network is configured to perform a method comprising providing training data comprising an input signal and an expected output signal to the neural network. The method further comprises computing, for each neuronal unit, a spatial gradient component and computing, for each neuronal unit, a temporal gradient component. The method further comprises updating the temporal and the spatial gradient component for each neuronal unit at each time instance of the input signal. The computing of the spatial and the gradient component may be performed independently from each other.

[0026] According to embodiments, the neural network may be a recurrent neural network, a spiking neural network or a hybrid neural network.

[0027] According to an embodiment of another aspect of the invention, a computer program product for training a neural network is provided. The computer program product comprises a computer readable storage medium having program instructions embodied therewith, the program instructions executable by the neural network to cause the neural network to perform a method comprising steps of receiving training data comprising an input signal and an expected output signal. The method comprises further steps of computing, for each neuronal unit, a spatial gradient component and computing, for each neuronal unit, a temporal gradient component, Further steps include updating the temporal and the spatial gradient component for each neuronal unit at each time instance of the input signal. According to embodiments the computing of the spatial and the temporal gradient component may be performed independently from each other.

[0028] Embodiments of the invention will be described in more detail below, by way of illustrative and non-limiting examples, with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0029] FIG. 1 illustrates the gradient flow of a computer-implemented method for training a neural network according to an embodiment of the invention;

[0030] FIG. 2 illustrates the gradient flow of a computer-implemented method for training a neural network according to an embodiment of the invention;

[0031] FIG. 3 shows a spiking neuronal unit of a spiking neural network;

[0032] FIG 4a shows test results of methods according to embodiments of the invention compared with back propagation through time (BPPT) techniques;

[0033] FIG. 4b shows further test results of methods according to embodiments of the invention compared with back propagation through time (BPPT) techniques;

[0034] FIG. 5 shows test result of another task concerning handwritten digit classification;

[0035] FIG. 6 illustrates how methods according to embodiments of the invention can be implemented on neuromorphic hardware;

[0036] FIG. 7 shows a simplified schematic diagram of a neural network according to an embodiment of the invention;

[0037] FIG. 8 shows a flow chart of method steps of a computer-implemented method for training parameters of a recurrent neural network;

[0038] FIG. 9 shows an exemplary embodiment of a computing system for performing a method according to embodiments of the invention;

[0039] FIG. 10 and FIG. 11 show exemplary detailed derivation of methods according to embodiments of the invention for deep neural networks.

DETAILED DESCRIPTION

[0040] Embodiments of the invention provide a method for training, in particular online training of neural networks, in particular recurrent neural networks (RNNs). The method may be in the following also denoted as OSTL. Methods according to embodiments of the invention provide an advantageous algorithm which can be used for online learning applications by separating spatial and temporal gradients.

[0041] FIG. 1 illustrates the gradient flow of a computer-implemented method for training a neural network 100 according to an embodiment of the invention. For FIG. 1 it is assumed that the neural network 100 is a recurrent neural network (RNN) with a single layer 110 comprising neuronal units 111. The neural network is unfolded for three time steps t.

[0042] Each neuronal unit 111 has an internal state S, 120. The method comprises providing training data comprising an input signal x^t, 131 and an expected output signal 132 to the neural network. Then, the method computes for each neuronal unit 110 a spatial gradient component L^t, 141 and a temporal gradient component e^t, 142. Furthermore, at each time instance t of the input signal 131, the temporal gradient components 130 and the spatial gradient components 131 are updated for each neuronal unit 110.

[0043] The objective of the learning/training is to train parameters θ of the neural network such that it minimizes the error E^t between the current output signal y^t at a time t and the input signal x^t.

[0044] In RNNs, the network error E^t at time t is often a function of the output y^t of the neuronal units in the output layer, i.e. $E^t = f(y^t)$. In addition, many neuronal units in RNNs may contain an internal state s^t on which the output depends, i.e. $y^t = f(s^t)$. This internal state of the neuronal units may be a recursive function of itself that in addition depends on its inputs signal x^t and recursively on its output signals through trainable input weights W and trainable recurrent weights H, respectively.

[0045] According to embodiments, an equation governing the internal state can be formulated as $s^t = f(x^t, s^{t-1}, y^{t-1}, W, H)$, for example $s^t = W x^t + H y^{t-1}$.
For the sake of notational simplicity, all the trainable parameters of the RNN 100 may be in the following collectively described by a variable θ. This simplifies the above equation to $s^t = f(x^t, s^{t-1}, y^{t-1}, \theta)$.

[0046] Moreover, the notation of the output y^t may be extended according to embodiments to also allow for a direct dependency on the trainable parameters, i.e. $y^t=f(s^t, \theta)$, for example $y^t = \sigma(s^t + b)$.

[0047] Using this notation, the required change of the parameters θ to minimize E may be computed based on the principle of gradient descent as

$$\Delta\theta = -\eta \frac{\mathrm{d}E}{\mathrm{d}\theta}. \qquad (1)$$

[0048] From this, embodiments of the invention use the backpropagation through time (BPTT) technique as a starting point for the derivation and express dE/dθ as

$$\frac{\mathrm{d}E}{\mathrm{d}\theta} = \sum_{1 \le t \le T} \frac{\partial E^t}{\partial y^t} \left[\frac{\partial y^t}{\partial s^t} \frac{\mathrm{d}s^t}{\mathrm{d}\theta} + \frac{\partial y^t}{\partial \theta} \right], \qquad (2)$$

where the summation over time ranges from the first time step t=1 until the last time step t=T. Then Equation 2 is expanded below and a recursion is unraveled that can be exploited to form an online reformulation of BPTT. For the sake of brevity, we outline only the main steps for a single unit, but the detailed derivation is given in the supplementary material. further below. In particular, it can be shown that

$$\frac{\mathrm{d}s^t}{\mathrm{d}\theta} = \sum_{1 \le i \le t} \left(\prod_{t \ge t' > i} \frac{\mathrm{d}s^{t'}}{\mathrm{d}s^{t'-1}} \right) \left(\frac{\partial s^i}{\partial \theta} + \frac{\partial s^i}{\partial y^{i-1}} \frac{\partial y^{i-1}}{\partial \theta} \right). \qquad (3)$$

[0049] Equation 3 can be rewritten in a recursive form as follows

$$\epsilon^{t,\theta} := \frac{\mathrm{d}s^t}{\mathrm{d}\theta} = \left(\frac{\mathrm{d}s^t}{\mathrm{d}s^{t-1}} \epsilon^{t-1,\theta} + \left(\frac{\partial s^t}{\partial \theta} + \frac{\partial s^t}{\partial y^{t-1}} \frac{\partial y^{t-1}}{\partial \theta} \right) \right). \qquad (4)$$

This leads to an expression of the gradient as

$$\frac{\mathrm{d}E}{\mathrm{d}\theta} = \sum_t L^t e^{t,\theta}, \tag{5}$$

where

$$e^{t,\theta} := \frac{\mathrm{d}y^t}{\mathrm{d}\theta} = \frac{\partial y^t}{\partial s^t}\epsilon^{t,\theta} + \frac{\partial y^t}{\partial \theta} \tag{6}$$

$$L^t := \frac{\partial E^t}{\partial y^t}. \tag{7}$$

[0050] Hence according to embodiments, the computing of the spatial and the gradient component may be performed independently from each other.

[0051] In the example of standard RNNs, the explicit form of these equations is

$$\frac{\mathrm{d}s_l^t}{\mathrm{d}s_l^{t-1}} = H_l h'^{t-1}_l$$

$$\epsilon^{t,W} = \frac{\mathrm{d}s^t}{\mathrm{d}s^{t-1}}\epsilon^{t-1,W} + x^t \qquad e^{t,W} = \sigma'^t \epsilon^{t,W}$$

$$\epsilon^{t,H} = \frac{\mathrm{d}s^t}{\mathrm{d}s^{t-1}}\epsilon^{t-1,H} + y^{t-1} \qquad e^{t,H} = \sigma'^t \epsilon^{t,H}$$

$$\epsilon^{t,b} = \frac{\mathrm{d}s^t}{\mathrm{d}s^{t-1}}\epsilon^{t-1,b} + H\sigma'^{t-1} \qquad e^{t,b} = \sigma'^t \epsilon^{t,b} + \sigma'^t$$

[0052] According to embodiments the notation takes inspiration from the standard nomenclature of biological systems, where the change of synaptic weights is often decomposed into a learning signal and an eligibility trace. In the simplest case, eligibility traces are low-pass filtered versions of the neural activities, while learning signals represent spatially delivered reward signals.

Therefore, according to embodiments the temporal gradients denoted $e^{t,\theta}$ in Equation 6 may be associated with eligibility traces and the spatial gradients denoted as L^t in Equation 7 may be associated with learning signals.

[0053] Similar to biological systems, the parameter change $dE/d\theta$ according to Equation 5 is calculated as the sum over time of products of the eligibility trace and the learning signal. This enables the parameter updates to be computed online, as shown in Figure 1.

[0054] Furthermore, it should be noted that the derivation in equation 6 is exact.

[0055] As can be seen in FIG. 1, at each time step the temporal gradients may be combined with the spatial gradients of this time step and do not need to go back until the beginning of the input sequence/input signal as required according to the known backpropagation though time technique.

[0056] FIG. 2 illustrates the gradient flow of a computer-implemented method for training a neural network 200 according to an embodiment of the invention. For FIG. 2 it is assumed that the neural network 200 is a recurrent neural network (RNN) with multiple layers.

[0057] More particularly, Figure 2 illustrates the gradient flow for a two-layer RNN comprising first layer 210 with neuronal units 211 and a second layer 220 with neuronal units 221. The layers 210 and 220 are unfolded for three time steps and the spatial and temporal gradients are separated.

[0058] Each neuronal unit 211 has an internal state S_1, 230. Each neuronal unit 221 has an internal state S_2, 231. The method comprises providing training data comprising an input signal x^t, 141 and an expected output signal 142 to the neural network 200. Then, the method computes for each neuronal unit 211 a spatial gradient component L_1^t, 151 and for each neuronal unit 221 a spatial gradient component L_2^t, 152. Furthermore, the method computes for each neuronal unit 211 a temporal gradient component e_1^t, 161 and for each neuronal unit 221 a temporal gradient component e_2^t, 162.

[0059] Furthermore, at each time instance t of the input signal 141, the temporal gradient components 161, 162 and the spatial gradient components 151, 152 are updated for each neuronal unit 211, 221 respectively.

[0060] Many state-of-the-art applications rely on more complicated multi-layer architectures. To extend methods according to embodiments of the invention to deep architectures, the definitions of the state s^t and the output y^t may be revisited as follows. The error E^t in deep architectures is only a function of the last output layer k, i.e. $E^t = f(y_k^t)$ and each layer l has its own trainable parameters θ_l. The input to layer l is the output of the previous layer y_{l-1}^t and for the first layer, the external input is used $y_0^t = x^t$.

[0061] Thus, the definitions may be adapted to

$$s_l^t = f(s_l^{t-1}, y_l^{t-1}, y_{l-1}^t, \theta_l) \quad (8)$$

$$y_l^t = f(s_l^t, \theta_l)., \quad (9)$$

[0062] For a single-layer neural network, the separation of spatial and temporal components comes if one follows the derivations outlined by Equations 3 to 5.

[0063] However, for a multi-layer architecture, the term $ds^t/d\theta$ in Equation 3 may involve different layers l and m, e.g. $ds_l^t/d\theta_m$, and thereby introduces dependencies across layers, see supplementary material.

[0064] In order to maintain the benefits discussed above, the clear separation of spatial and temporal gradients is also introduced for multi-layer architectures according to embodiments of the invention. Accordingly, similar steps as described above for a single layer RNN are performed using the generalized state and output Equations 8 and 9. Following the detailed derivations in the supplementary material, the following eligibility traces and learning signals are obtained for layer l:

$$e_l^{t,\theta} = \left(\frac{\partial y_l^t}{\partial s_l^t} \epsilon_l^{t,\theta} + \frac{\partial y_l^t}{\partial \theta_l} \right) \tag{10}$$

$$L_l^t = \frac{\partial E^t}{\partial y_k^t} \left(\prod_{(k-l+1) \geq m' \geq 1} \frac{\partial y_{k-m'+1}^t}{\partial s_{k-m'+1}^t} \frac{\partial s_{k-m'+1}^t}{\partial y_{k-m'}^t} \right), \tag{11}$$

where

$$\epsilon_l^{t,\theta} = \left(\frac{\mathrm{d}s_l^t}{\mathrm{d}s_l^{t-1}} \epsilon_l^{t-1,\theta} + \left(\frac{\partial s_l^t}{\partial \theta_l} + \frac{\partial s_l^t}{\partial y_l^{t-1}} \frac{\partial y_l^{t-1}}{\partial \theta_l} \right) \right). \tag{12}$$

Then, it can be shown that

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_t \left[L_l^t e_l^{t,\theta} + R \right]. \tag{13}$$

[0065] As one can see by comparing equations 5 to 13, the approach according to embodiments of the invention concerning multiplying a learning signal L_l^t with an eligibility trace $e_l^{t,\theta}$ stays the same in case of deep networks.

[0066] The learning signal L_l^t is specific for each layer and propagates from the last layer to the input layer without going back in time, i.e. it represents the spatial gradient passing through the network architecture. Furthermore, each layer computes its own eligibility trace $e_l^{t,\theta}$, which is solely dependent on contributions of the respective layer l, i.e. it represents the temporal gradient passing through time for the same layer.

[0067] However, additional terms are also involved in Equation 13, which either contain a mix of spatial and temporal gradients and generally require to go back in time. These terms are collected in the residual term R.

[0068] In order to maintain the separation between spatial and temporal gradients, Equation 13 is simplified according to embodiments by omitting the term R. Thus, the following formulation for multi-layer networks is obtained according to embodiments:

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_t L_l^t e_l^{t,\theta}. \qquad (14)$$

[0069] Hence according to embodiments of the invention the residual term R is consciously omitted, and the mixed spatial and temporal gradient components are not taken into consideration during learning/training. However, investigations of the inventors of the present invention have resulted in the insight that this is an advantageous approach. In particular, with such an approach it is known what is omitted. Furthermore, simulations of the inventors have provided empirical evidence that a competitive performance to BPTT may be achieved even without these terms, as will be explained further below.
Moreover, according to embodiments the residual term R may also be approximated, hence allowing to even better approximate the gradients from Equation 13.

[0070] FIG. 3 shows a spiking neuronal unit SNU, 310 of a spiking neural network 300. With reference to FIG. 3 it will be shown that methods according to embodiment can be applied to spiking neural networks (SNN). Dashed lines in FIG. 3 indicate connections with time-lag, while bold lines indicate parametrized connections. The SNU 310 comprises a block input 320, a block output 321, a reset gate 322 and a membrane potential 323.

[0071] While historically, SNNs were often trained with variants of spike timing- dependent plasticity, recently gradient-based training for SNNs has been proposed, e.g. in the document: Wozniak, S., Pantazi, A., Bohnstingl, T., and Eleftheriou, E. Deep learning incorporating biologically-inspired neural dynamics. arXiv, Dec 2018. URL https://arxiv. org/abs/1812.07040.

[0072] Such a method aims to bridge the ANN world with the SNN world by recasting the SNN dynamics with ANN-based building blocks, forming the spiking neuronal unit SNU, 310. The SNUs 310 of the spiking neural network 300 receive a plurality of input signals

[0073] With this approach, SNUs enable gradient-based learning, This allows to exploit the power of known optimization techniques for ANN, while still reproducing the dynamics of the leaky integrate-and-fire (LIF) neuron model, which is well-known in neuroscience.

[0074] As shown above methods according to embodiments of the invention may be used for generic RNNs, but can also be applied according to embodiments to train deep SNNs formulated as RNNs. This will be shown in the following. We start from the state and output equations of an SNU layer l, compare (Wozniak et al., 2018):

$$s_l^t = g(W_l y_{l-1}^t + H_l y_l^{t-1} + l(\tau) s_l^{t-1}(1 - y_l^{t-1})) \quad (15)$$

$$y_l^t = h(s_l^t + b_l). \quad (16)$$

[0075] By using Equations 15 and 16, we derive the eligibility traces according to Equation 10, as

$$e_l^{t,W} = h'^t_l \epsilon_l^{t,W} \quad (17)$$

$$e_l^{t,H} = h'^t_l \epsilon_l^{t,H} \quad (18)$$

$$e_l^{t,b} = h'^t_l \epsilon_l^{t,b} + h'^t_l, \quad (19)$$

where

$$\epsilon_l^{t,W} = g'^t_l \cdot \left[\frac{\mathrm{d}s_l^t}{\mathrm{d}s_l^{t-1}} \epsilon_l^{t-1,W} + y_{l-1}^t \right]$$

$$\epsilon_l^{t,H} = g'^t_l \cdot \left[\frac{\mathrm{d}s_l^t}{\mathrm{d}s_l^{t-1}} \epsilon_l^{t-1,H} + y_l^{t-1} \right]$$

$$\epsilon_l^{t,b} = g'^t_l \cdot \left[\frac{\mathrm{d}s_l^t}{\mathrm{d}s_l^{t-1}} \epsilon_l^{t-1,b} - l(\tau) s_l^{t-1} h'^{t-1}_l + H_l h'^{t-1}_l \right]$$

and

$$\frac{\mathrm{d}s_l^t}{\mathrm{d}s_l^{t-1}} = l(\tau) \left(1 - y_l^{t-1} - s_l^{t-1} h'^{t-1}_l \right) + H_l h'^{t-1}_l. \quad (20)$$

It should be noted that the short-hand notation of

$$\frac{\mathrm{d}g(\chi_l^t)}{\mathrm{d}\chi_l^t} = g'^t_l \text{ and } \frac{\mathrm{d}h(\chi_l^t)}{\mathrm{d}\chi_l^t} = h'^t_l.$$

has been used.

[0076] For a mean squared error loss function, e.g.

$$E^t = (\hat{y}^t - y_k^t)^2, \text{ where } \hat{y}^t$$

is the target output, the learning signal can be calculated as:

$$L_l^t = -2(\hat{y}^t - y_k^t) \tag{21}$$

$$\left[\prod_{(k-l+1)\geq m'\geq 1} {h'_{k-m'+1}}^t {g'_{k-m'+1}}^t W_{k-m'+1}\right] \cdot \tag{22}$$

[0077] For a deep neural network with k layers consisting of RNNs or recurrent SNUs, methods according to embodiments of the invention have time complexity of O(kn4). This time complexity is determined by the network structure itself and is primarily dominated by the recurrency matrix H_l. If feed-forward architectures are used according to embodiments, the terms involving H_l vanish, and the equations of SNU become

$$s_l^t = g(W_l y_{l-1}^t + l(\tau) s_l^{t-1}(1 - y_l^{t-1})) \tag{23}$$

$$y_l^t = h(s_l^t + b_l). \tag{24}$$

[0078] These equations then lead to the following eligibility traces

$$e_l^{t,W} = {h'_l}^t \epsilon_l^{t,W} \tag{25}$$

$$e_l^{t,b} = {h'_l}^t \epsilon_l^{t,b} + {h'_l}^t, \tag{26}$$

where

$$\epsilon_l^{t,W} = {g'_l}^t \cdot \left[\frac{\mathrm{d}s_l^t}{\mathrm{d}s_l^{t-1}} \epsilon_l^{t-1,W} + y_{l-1}^t\right]$$

$$\epsilon_l^{t,b} = {g'_l}^t \cdot \left[\frac{\mathrm{d}s_l^t}{\mathrm{d}s_l^{t-1}} \epsilon_l^{t-1,b} - l(\tau) s_l^{t-1} {h'_l}^{t-1}\right],$$

with.

$$\frac{\mathrm{d}s_l^t}{\mathrm{d}s_l^{t-1}} = l(\tau)\left(1 - y_l^{t-1} - s_l^{t-1}h'^{t-1}_l\right). \qquad (27)$$

[0079] This greatly reduces the time complexity from $O(kn^4)$ to $O(kn^2)$. Using feed-forward SNU network architecture does not necessarily prevent solving temporal tasks. Such networks have long been used in SNNs and it implies that the network should rely on the internal states of the units, implemented using self-recurrency, rather than on layer-wise recurrency matrices H_l.

[0080] It should be noted that according to embodiments, the learning signal may be computed without the matrices W, e.g. based on some randomization or approximations of W. More particularly, the learning signal may be computed based on different matrices that are not used in the forward path. In other words, the forward path may use matrices W, while the learning signal is computed on different matrices B. The matrices B might be trainable or not.

[0081] According to embodiments, methods as presented above may also be used for hybrid networks. In this respect, a very common scenario in deep RNNs or SNNs is that they are often coupled with layers of stateless neurons at the output, for example sigmoid or softmax layers. Methods according to embodiments of the invention can also be applied without any modifications to train these hybrid networks containing one or more layers of stateless neurons. In particular, the state and output equations of these layers simplify to

$$s_l^t = f(y_{l-1}^t, \theta_l) \text{ and } y_l^t = f(s_l^t, \theta_l),$$

which causes the term

$$\frac{\mathrm{d}s_l^t}{\mathrm{d}s_l^{t-1}}$$

in Equation 12 to vanish and the eligibility traces and learning signals can be calculated as

$$\epsilon_l^{t,\theta} = \frac{\partial y_l^t}{\partial s_l^t}\epsilon_l^{t,\theta} + \frac{\partial y_l^t}{\partial \theta_l} \tag{28}$$

$$L_l^t = \frac{\partial E^t}{\partial y_k^t}\left(\prod_{(k-l+1)\geq m'\geq 1} \frac{\partial y_{k-m'+1}^t}{\partial s_{k-m'+1}^t}\frac{\partial s_{k-m'+1}^t}{\partial y_{k-m'}^t}\right), \tag{29}$$

with

$$\epsilon_l^{t,\theta} = \frac{\partial s_l^t}{\partial \theta_l}. \tag{30}$$

[0082] It should be noted that a stateless layer will not introduce any residual terms R. This has the effect that when adding such a layer to the network, even between RNN layers, the gradients for the subsequent layers remain unchanged.

[0083] FIG. 4a shows test results of methods according to embodiments of the invention compared with back propagation though time (BPPT) techniques. More particularly, FIG. 4a concerns music prediction based on the JSB dataset as introduced in the document: Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription In Proceedings of the 29th International Conference on International Conference on Machine

[0084] Learning, ICML'12, pp. 1881–1888, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851. For this the standard training/testing data split was used. For the test a hybrid architecture comprising a feed-forward SNU layer with 150 units and a stateless layer sigmoid layer with 88 units on top. To obtain a baseline, the same network, including all its hyperparameters, was trained with methods according to embodiments of the invention and BPTT for 1000 epochs. The Y-axis denotes the negative log-likelihood, averaged over 10 random initial conditions. The bar 411 shows the result for the training of the BPTT method, while bar 412 shows the result for the training of methods according to embodiments of the

invention. Furthermore, the bar 413 shows the result for the test run of the BPTT method, while bar 414 shows the result for the test run of methods according to embodiments of the invention.

[0085] As shown in Figure 4a, the results obtained with methods according to embodiments of the invention are practically on par with these obtained with BPTT. Note that task proves the gradient equivalence of BPTT and of methods according to embodiments of the invention for a hybrid architecture with a single RNN layer and a stateless layer on top.

[0086] As shown in FIG. 4b, this task may be used to demonstrate the reduced computational complexity of methods according to embodiments of the invention for feed-forward SNNs. To this end, the number of required floating point operations MFLOP (y-axis) was measured, using the built-in TensorFlow profiler, for one parameter updated across different input sequence lengths (x-axis) of the JSB input sequence, see Figure 4b. As can be seen from line 421, BPTT needs to perform temporal unrolling, hence the linear dependence on the length of the sequence T, whereas methods according to embodiments of the invention as shown by line 422 do not and hence it remains steady. However, in practical implementations one may need to accumulate the updates from methods according to embodiments of the invention over time, which results in the same complexity as BPTT. Note that the initially higher cost of methods according to embodiments of the invention is due to implementation overheads, as methods according to embodiments of the invention are not contained in the standard toolbox of TensorFlow. Nevertheless, the obtained plot is consistent with theoretical complexity analysis.

[0087] FIG. 5 shows test result of another task concerning handwritten digit classification based on the MNIST dataset as introduced in the document: Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient based learning applied to document recognition. Proc. IEEE, 86(11):2278–2324, Nov 1998. ISSN 1558-2256. doi: 10.1109/5.726791.

[0088] Again the standard training/testing data split was used. According to the test a feed-forward architecture of five layers of SNUs with 256 units was employed and trained for 50 epochs averaging over 10 random initial conditions. Similar to the task illustrated with reference to FIG. 4a and 4b, the accuracy of methods according to embodiments of the invention matches the one of BPTT. The y-axis denotes the accuracy (percentage), the x-axis the number of epochs, the line 510 the results for BPTT and the line 520 the results for methods according to embodiments of the invention.

[0089] FIG. 6 illustrates how methods according to embodiments of the invention can be implemented on neuromorphic hardware. The neuromorphic hardware may comprise in particular a crossbar array comprising a plurality of row lines 610, a plurality of column lines 620 and a plurality of junctions 630 arranged between the plurality of row lines 610 and the plurality of column lines 620. Each junction 630 comprises a resistive memory element 640, in particular a serial arrangement of a resistive memory element and an access element comprising an access terminal for accessing the resistive memory element. The resistive elements may be e.g. phase-change memory elements, conductive bridge random access memory elements (CBRAM), metal-oxide resistive random access memory elements (RRAM), magneto-resistive random access memory elements (MRAM), ferroelectric random access memory elements (FeRAM) or optical memory elements.

[0090] According to embodiments the input weights and the recursive weights may be placed on the neuromorphic device, in particular as resistance states of the resistive elements.

[0091] According to such an embodiment the trainable input weights W_l and the trainable recurrent weights H_l are mapped to the resistive memory elements 640.

[0092] FIG. 7 shows a simplified schematic diagram of a neural network 700 according to an embodiment of the invention. The neural network 700 comprises an input layer 710 comprising a plurality of neuronal units 10, one or more hidden layers 720 comprising a plurality of neuronal units 10 and an output layer 730 comprising a plurality of neuronal units 10. The neural network 700 comprises a plurality of electrical connections 20 between the neuronal units 10. The electrical connections 20 connect the outputs of neurons from one layer, e.g. from the input layer 710, to the inputs of neuronal units from the next layer, e.g. one of the hidden layers 720. The neural network 700 may be in particular embodied as recurrent neural network.

[0093] Accordingly, the network 700 comprises recurrent connections from one layer to the neuronal units from the same or a previous layer as illustrated in a schematic way by the arrows 30.

[0094] FIG. 8 shows a flow chart of method steps of a computer-implemented method for training parameters of a recurrent neural network.

[0095] The method starts at a step 810.

[0096] At a step 820, training data is received by or in other words provided to the neural network. The training data comprises an input signal and an expected output signal.

[0097] At a step 830, the neural network computes for each neuronal unit a spatial gradient component.

[0098] At a step 840, the neural network computes for each neuronal unit a temporal gradient component.

[0099] At a step 850, the neural network updates the temporal and the spatial gradient component for each neuronal unit at each time instance of the input signal.

[00100] According to an embodiment, the updates of the parameters of the neural network can be accumulated and deferred until a later time step T. The computing of the spatial and the gradient component is performed independently from each other.

[00101] The steps 820 to 850 are repeated at loops 860. More particularly, the steps 820 to 850 may be repeated at specific or predefined time instances and in particular at each time instance.

[00102] Referring now to Fig. 9, an exemplary embodiment of a computing system 900 for performing a method according to embodiments of the invention is illustrated. The computing system 900 may form a neural network according to embodiments. The computing system 900 may be operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well-known computing systems, environments, and/or configurations that may be suitable for use with computing system 900 include, but are not limited to, personal computer systems, server computer systems, thin clients, thick clients, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputer

systems, mainframe computer systems, and distributed cloud computing environments that include any of the above systems or devices, and the like.

[00103] The computing system 900 may be described in the general context of computer system-executable instructions, such as program modules, being executed by a computer system. Generally, program modules may include routines, programs, objects, components, logic, data structures, and so on that perform particular tasks or implement particular abstract data types. The computing system 900 may be shown in the form of a general-purpose computing device. The components of server computing system 900 may include, but are not limited to, one or more processors or processing units 916, a system memory 928, and a bus 918 that couples various system components including system memory 928 to processor 916.

[00104] Bus 918 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus.

[00105] Computing system 900 typically includes a variety of computer system readable media. Such media may be any available media that is accessible by computing system 900, and it includes both volatile and non-volatile media, removable and non-removable media.

[00106] System memory 928 can include computer system readable media in the form of volatile memory, such as random access memory (RAM) 930 and/or cache memory 932. Computing system 900 may further include other removable/non-removable, volatile/non-volatile computer system storage media. By way of example only, storage system 934 can be provided for reading from and writing to a non-removable, non-volatile magnetic media (not shown and typically called a "hard drive"). Although not shown, a magnetic disk drive for reading from and writing to a removable, non-volatile magnetic disk (e.g., a "floppy disk"), and an optical disk drive for reading from or writing to a removable, non-volatile optical disk such as a CD-ROM, DVD-ROM or other optical media can be provided. In such instances, each can be connected to bus 918 by one or more data media interfaces. As will be further depicted and described below, memory 928 may include at least one program product having a set (e.g., at

least one) of program modules that are configured to carry out the functions of embodiments of the invention.

[00107] Program/utility 940, having a set (at least one) of program modules 942, may be stored in memory 928 by way of example, and not limitation, as well as an operating system, one or more application programs, other program modules, and program data. Each of the operating system, one or more application programs, other program modules, and program data or some combination thereof, may include an implementation of a networking environment. Program modules 942 generally carry out the functions and/or methodologies of embodiments of the invention as described herein. Program modules 942 may carry out in particular one or more steps of a computer-implemented for training recurrent neural networks, e.g. one or more steps of the method as described with reference to FIGS 1, 2 and 8.

[00108] Computing system 900 may also communicate with one or more external devices 915 such as a keyboard, a pointing device, a display 924, etc.; one or more devices that enable a user to interact with computing system 900; and/or any devices (e.g., network card, modem, etc.) that enable computing system 900 to communicate with one or more other computing devices. Such communication can occur via Input/Output (I/O) interfaces 922. Still yet, computing system 900 can communicate with one or more networks such as a local area network (LAN), a general wide area network (WAN), and/or a public network (e.g., the Internet) via network adapter 920. As depicted, network adapter 920 communicates with the other components of computing system 900 via bus 918. It should be understood that although not shown, other hardware and/or software components could be used in conjunction with computing system 900. Examples include, but are not limited to: microcode, device drivers, redundant processing units, external disk drive arrays, RAID systems, tape drives, and data archival storage systems, etc.

[00109] The present invention may be a system, a method, and/or a computer program product at any possible technical detail level of integration. The computer program product may include a computer readable storage medium (or media) having computer readable program instructions thereon for causing a processor to carry out aspects of the present invention.

[00110] The computer readable storage medium can be a tangible device that can retain and store instructions for use by an instruction execution device. The computer readable storage

medium may be, for example, but is not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing. A non-exhaustive list of more specific examples of the computer readable storage medium includes the following: a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), a static random access memory (SRAM), a portable compact disc read-only memory (CD-ROM), a digital versatile disk (DVD), a memory stick, a floppy disk, a mechanically encoded device such as punch-cards or raised structures in a groove having instructions recorded thereon, and any suitable combination of the foregoing. A computer readable storage medium, as used herein, is not to be construed as being transitory signals per se, such as radio waves or other freely propagating electromagnetic waves, electromagnetic waves propagating through a waveguide or other transmission media (e.g., light pulses passing through a fiber-optic cable), or electrical signals transmitted through a wire.

[00111] Computer readable program instructions described herein can be downloaded to respective computing/processing devices from a computer readable storage medium or to an external computer or external storage device via a network, for example, the Internet, a local area network, a wide area network and/or a wireless network. The network may comprise copper transmission cables, optical transmission fibers, wireless transmission, routers, firewalls, switches, gateway computers and/or edge servers. A network adapter card or network interface in each computing/processing device receives computer readable program instructions from the network and forwards the computer readable program instructions for storage in a computer readable storage medium within the respective computing/processing device.

[00112] Computer readable program instructions for carrying out operations of the present invention may be assembler instructions, instruction-set-architecture (ISA) instructions, machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, configuration data for integrated circuitry, or either source code or object code written in any combination of one or more programming languages, including an object oriented programming language such as Smalltalk, C++, or the like, and procedural programming languages, such as the "C" programming language or similar programming languages. The computer readable program instructions may execute entirely on the user's computer, partly on

the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider). In some embodiments, electronic circuitry including, for example, programmable logic circuitry, field-programmable gate arrays (FPGA), or programmable logic arrays (PLA) may execute the computer readable program instructions by utilizing state information of the computer readable program instructions to personalize the electronic circuitry, in order to perform aspects of the present invention.

[00113] Aspects of the present invention are described herein with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems), and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer readable program instructions.

[00114] These computer readable program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks. These computer readable program instructions may also be stored in a computer readable storage medium that can direct a computer, a programmable data processing apparatus, and/or other devices to function in a particular manner, such that the computer readable storage medium having instructions stored therein comprises an article of manufacture including instructions which implement aspects of the function/act specified in the flowchart and/or block diagram block or blocks.

[00115] The computer readable program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other device to cause a series of operational steps to be performed on the computer, other programmable apparatus or other device to produce a computer implemented process, such that the instructions which execute on the computer,

other programmable apparatus, or other device implement the functions/acts specified in the flowchart and/or block diagram block or blocks.

[00116] The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods, and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of instructions, which comprises one or more executable instructions for implementing the specified logical function(s). In some alternative implementations, the functions noted in the blocks may occur out of the order noted in the Figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts or carry out combinations of special purpose hardware and computer instructions.

[00117] The descriptions of the various embodiments of the present invention have been presented for purposes of illustration, but are not intended to be exhaustive or limited to the embodiments disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope and spirit of the described embodiments. The terminology used herein was chosen to best explain the principles of the embodiments, the practical application or technical improvement over technologies found in the marketplace, or to enable others of ordinary skill in the art to understand the embodiments disclosed herein.

[00118] In general, modifications described for one embodiment may be applied to another embodiment as appropriate.

[00119] In the following a detailed derivation of methods according to embodiments of the invention for deep neural networks, in particular for recurrent networks comprising multi-layer architectures, will be provided as supplement.

Many state-of-the-art applications rely on multi-layer networks, in which the error E^t is only a function of the last output layer k, i.e., $E^t = E^t(y_k^t)$. According to embodiments of the invention, the state and output equations are adapted as follows

$$s_l^t = s_l^t(s_l^{t-1}, y_l^{t-1}, y_{l-1}^t, \theta_l) \tag{31}$$

$$y_l^t = y_l^t(s_l^t, \theta_l). \tag{32}$$

Using this reformulation, Equation 2 can be generalized as follows

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_{1 \le t \le T} \frac{\mathrm{d}E^t}{\mathrm{d}\theta_l} = \sum_{1 \le t \le T} \frac{\partial E^t}{\partial y_k^t} \left[\frac{\partial y_k^t}{\partial s_k^t} \frac{\mathrm{d}s_k^t}{\mathrm{d}\theta_l} + \frac{\partial y_k^t}{\partial \theta_l} \right]. \tag{33}$$

For the last layer of a multi-layer network, where $k = l$, Equation 33 corresponds to Equation 2 for a single layer. However, for the hidden layers, i.e., $k \neq l$, the term $\frac{\mathrm{d}s_k^t}{\mathrm{d}\theta_l}$ is expanded as follows

$$\frac{\mathrm{d}s_k^t}{\mathrm{d}\theta_l} = \frac{\partial s_k^t}{\partial \theta_l} + \frac{\partial s_k^t}{\partial y_k^{t-1}} \frac{\mathrm{d}y_k^{t-1}}{\mathrm{d}\theta_l} + \frac{\partial s_k^t}{\partial y_{k-1}^t} \frac{\mathrm{d}y_{k-1}^t}{\mathrm{d}\theta_l}. \tag{34}$$

We define a recursive term $\underset{m}{\chi_l^{t,\theta}}$ as

$$\begin{aligned}
\underset{m}{\chi_l^{t,\theta}} &:= \frac{\mathrm{d}s_l^t}{\mathrm{d}\theta_m} \\
&= \frac{\partial s_l^t}{\partial s_m^{t-1}} \underset{m}{\chi_l^{t-1,\theta}} + \frac{\partial s_l^t}{\partial y_l^{t-1}} \left(\frac{\partial y_l^{t-1}}{\partial s_l^{t-1}} \underset{m}{\chi_l^{t-1,\theta}} + \frac{\partial y_l^{t-1}}{\partial \theta_m} \right) \\
&+ \frac{\partial s_l^t}{\partial y_{l-1}^t} \left(\frac{\partial y_{l-1}^t}{\partial s_{l-1}^t} \underset{m}{\chi_{l-1}^{t,\theta}} + \frac{\partial y_{l-1}^t}{\partial \theta_m} \right) + \frac{\partial s_l^t}{\partial \theta_m},
\end{aligned} \tag{35}$$

with the following properties

$$\epsilon_l^{t,\theta} := \underset{l}{\chi_l^{t,\theta}} \tag{36}$$

$$\epsilon_l^{t,\theta} := \frac{\mathrm{d}y_l^t}{\mathrm{d}\theta_l} = \left(\frac{\partial y_l^t}{\partial s_l^t} \underset{l}{\chi_l^{t,\theta}} + \frac{\partial y_l^t}{\partial \theta_l} \right) \tag{37}$$

$$\underset{m}{\chi_l^{t<1,\theta}} = 0$$

$$\underset{l+1}{\chi_l^{t,\theta}} = 0$$

$$\underset{m}{\chi_{l<1}^{t,\theta}} = 0$$

$$\underset{m<1}{\chi_l^{t,\theta}} = 0.$$

The term $\underset{m}{\chi_l^{t,\theta}}$ for $k \neq l$ contains a recursion in time, but additionally it contains a recursion in space, i.e., it depends on other layers, for example the $(k-1)$-th layer.

If we insert the term $\underset{m}{\chi_l^{t,\theta}}$ in Equation 33 we obtain

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_{1 \le t \le T} \frac{\partial E^t}{\partial y_k^t} \left[\frac{\partial y_k^t}{\partial s_k^t} \underset{l}{\chi_k^{t,\theta}} + \frac{\partial y_k^t}{\partial \theta_l} \right]. \tag{38}$$

The right-hand side of Equation 38 is expanded to a more complex expression

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_{1\leq t\leq T}\left[\frac{\mathrm{d}E^t}{\mathrm{d}y_k^t}\left[\frac{\partial y_k^t}{\partial s_k^t}\frac{\partial s_k^t}{\partial s_k^{t-1}}\chi_l^{t-1,\theta} + \frac{\partial y_k^t}{\partial s_k^t}\frac{\partial s_k^t}{\partial y_k^{t-1}}\left(\frac{\partial y_k^{t-1}}{\partial s_k^{t-1}}\chi_l^{t-1,\theta} + \frac{\partial y_k^{t-1}}{\partial \theta_l}\right)\right.\right.$$
$$+ \frac{\partial y_k^t}{\partial s_k^t}\frac{\partial s_k^t}{\partial y_{k-1}^t}\left(\frac{\partial y_{k-1}^t}{\partial s_{k-1}^t}\chi_l^{t,\theta}_{k-1} + \frac{\partial y_{k-1}^t}{\partial \theta_l}\right) + \left.\left.\frac{\partial y_k^t}{\partial s_k^t}\frac{\partial s_k^t}{\partial \theta_l}\right]\right.$$
$$\left. + \frac{\partial y_k^t}{\partial \theta_l}\right], \tag{39}$$

where the two recurrencies — $\chi_{l}^{t,\theta}_{k-1}$ in space, and $\chi_l^{t-1,\theta}$ in time — become apparent. When expanding $\chi_{l}^{t}_{k-1}$ far enough in space, it eventually reaches $\chi_l^{t,\theta} = \epsilon_l^t$. Therefore, we can rewrite Equation 39 as

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_{1\leq t\leq T}\left[\frac{\mathrm{d}E^t}{\mathrm{d}y_k^t}\left(\prod_{(k-l+1)\geq m'\geq 1}\frac{\partial y_{k-m'+1}^t}{\partial s_{k-m'+1}^t}\frac{\partial s_{k-m'+1}^t}{\partial y_{k-m'}^t}\right)\left(\frac{\partial y_l^t}{\partial s_l^t}\chi_l^{t,\theta} + \frac{\partial y_l^t}{\partial \theta_l}\right) + R\right], \tag{40}$$

where we collect all the remaining terms into a residual term R. In addition, we define a generalized learning signal L_l^t and a generalized eligibility trace $e_l^{t,\theta}$ as

$$L_l^t = \frac{\partial E^t}{\partial y_k^t}\left(\prod_{(k-l+1)\geq m'\geq 1}\frac{\partial y_{k-m'+1}^t}{\partial s_{k-m'+1}^t}\frac{\partial s_{k-m'+1}^t}{\partial y_{k-m'}^t}\right) \tag{41}$$

$$e_l^{t,\theta} = \left(\frac{\partial y_l^t}{\partial s_l^t}\epsilon_l^{t,\theta} + \frac{\partial y_l^t}{\partial \theta_l}\right), \tag{42}$$

see Equations 10-11. This allows to express the parameter update as

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_{1\leq t\leq T}\left[L_l^t e_l^{t,\theta} + R\right], \tag{43}$$

see Equation 13. By omitting the residual term R according to embodiments, we arrive at Equation 14.

CLAIMS

What is claimed is:

1. A computer-implemented method for training a neural network, the network comprising one or more layers of neuronal units, wherein each neuronal unit has an internal state, wherein the method comprises:

providing training data comprising an input signal and an expected output signal to the neural network;

computing, for each neuronal unit, a spatial gradient component;

computing, for each neuronal unit, a temporal gradient component; and

updating the temporal and the spatial gradient component for each neuronal unit at each time instance of the input signal.

2. The computer-implemented method according to claim 1, wherein the computing of the spatial and the gradient component is performed independently from each other.

3. The computer-implemented method according to claim 1, further comprising updating a predefined set of training parameters of the neural network as a function of the spatial and the temporal gradient component.

4. The computer-implemented method according to claim 3, further comprising updating the predefined set of training parameters of the neural network at specific or predefined time instances as a function of the spatial and the temporal gradient components.

5. The computer-implemented method according to claim 4, further comprising updating the predefined set of training parameters of the neural network at each time instance as a function of the spatial and the temporal gradient components.

6. The computer-implemented method according to claim 1, wherein the method comprises:

computing, at each time instance, a spatial gradient component for each of the one or more layers;

and

computing, at each time instance, a temporal gradient component for each of the one or more layers.

7. The computer-implemented method according to claim 1, wherein:

the spatial gradient component is based on connectivity parameters of the neural network; and

the temporal gradient component is based on parameters related to temporal dynamics of the neuronal units.

8. The computer-implemented method according to claim 1, wherein the network comprises a single layer of neuronal units and computing the spatial gradient component comprises computing:

$$L^t := \frac{\partial E^t}{\partial y^t}$$

wherein

t denotes the respective time instance;

L^t denotes the spatial gradient component at time instance t;

E^t denotes the network error, in particular the error between an expected output signal and the current output signal at time instance t;

and

y^t denotes the current output signal at time instance t.

9. The computer-implemented method according to claim 1, further comprising updating a predefined set of training parameters of the neural network as a function of the spatial and the temporal gradient component, wherein the network comprises a single layer of neuronal units and computing the temporal gradient component comprises computing:

$$e^{t,\theta} := \frac{\mathrm{d}y^t}{\mathrm{d}\theta} = \frac{\partial y^t}{\partial s^t}\epsilon^{t,\theta} + \frac{\partial y^t}{\partial \theta}$$
$$\epsilon^{t,\theta} := \frac{\mathrm{d}s^t}{\mathrm{d}\theta} = \left(\frac{\mathrm{d}s^t}{\mathrm{d}s^{t-1}}\epsilon^{t-1,\theta} + \left(\frac{\partial s^t}{\partial \theta} + \frac{\partial s^t}{\partial y^{t-1}}\frac{\partial y^{t-1}}{\partial \theta}\right)\right)$$

wherein

t denotes the respective time instance;

y^t denotes the current output signal at time instance t;

s^t denotes the unit state at time instance t;

θ denotes the training parameters of the network; and

$\epsilon^{t,\theta} := \frac{\mathrm{d}s^t}{\mathrm{d}\theta}$.

10. The computer-implemented method according to claim 9, wherein updating the training parameters comprises computing;

$$\Delta\theta = \alpha \sum_t L^t e^{t,\theta},$$

wherein α is a learning rate.

11. The computer-implemented method according to claim 1, wherein the network comprises a plurality of layers of neuronal units and computing the spatial gradient component comprises computing:

$$L_l^t = \frac{\partial E^t}{\partial y_k^t}\left(\prod_{(k-l+1)\geq m'\geq 1}\frac{\partial y_{k-m'+1}^t}{\partial s_{k-m'+1}^t}\frac{\partial s_{k-m'+1}^t}{\partial y_{k-m'}^t}\right)$$

wherein:

L_l^t denotes the spatial gradient component of layer l at time instance t;

E^t denotes a network error, in particular the error between an expected output signal and the current output signal at time instance t;

t denotes the respective time instance;

y_k^t denotes the current output signal of layer k;

s_k^t denotes the unit state of layer k;

k denotes the last layer/output layer of the network; and

m' denotes intermediate layers of the network ranging from 1 to (*k-l+1*)

12. The computer-implemented method according to claim 1, wherein the network comprises a plurality of layers of neuronal units and computing the temporal gradient component comprises computing:

$$e_l^{t,\theta} = \left(\frac{\partial y_l^t}{\partial s_l^t} \epsilon_l^{t,\theta} + \frac{\partial y_l^t}{\partial \theta_l} \right)$$

$$\epsilon_l^{t,\theta} = \left(\frac{\mathrm{d}s_l^t}{\mathrm{d}s_l^{t-1}} \epsilon_l^{t-1,\theta} + \left(\frac{\partial s_l^t}{\partial \theta_l} + \frac{\partial s_l^t}{\partial y_l^{t-1}} \frac{\partial y_l^{t-1}}{\partial \theta_l} \right) \right)$$

wherein

t denotes the respective time instance;

l denotes the respective layer;

y^t denotes the current output signal;

s^t denotes the current unit state;

θ denotes training parameters of the network; and

$$\epsilon^{t,\theta} := \frac{\mathrm{d}s^t}{\mathrm{d}\theta}.$$

13. The computer-implemented method according to claim 1, further comprising updating a predefined set of training parameters of the neural network as a function of the spatial and the temporal gradient component, wherein updating the training parameters comprises computing:

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_t \left[L_l^t e_l^{t,\theta} + R \right]$$

wherein R is a residual term.

14. The computer-implemented method according to claim 13, wherein the residual term R is approximated with a combination of eligibility traces and learning signals.

15. The computer-implemented method according to claim 1, further comprising updating a predefined set of training parameters of the neural network as a function of the spatial and the temporal gradient component, wherein updating the network parameters comprises computing:

$$\Delta\theta_l = \alpha \sum_t L_l^t e_l^{t,\theta},$$

wherein α is a learning rate.

16. The computer-implemented method according to claim 1, wherein the neural network is selected from the group consisting of: a recurrent neural network, a hybrid network, a spiking neural network and a generic recurrent network, the generic recurrent network in particular comprising or consisting of long-short-term-memory units and gated recurrent units.

17. The computer-implemented method according to claim 1, further comprising updating a predefined set of training parameters of the neural network as a function of the spatial and the temporal gradient component, wherein the network comprises a plurality of layers of neuronal units and computing the temporal gradient component comprises computing:

$$e_l^{t,\theta} = \frac{\partial y_l^t}{\partial s_l^t} \epsilon_l^{t,\theta} + \frac{\partial y_l^t}{\partial \theta_l} + \frac{\partial y_l^t}{\partial y_l^{t-1}} e_l^{t-1,\theta}$$

wherein:

t denotes the respective time instance;

l denotes the layer;

y^t denotes the current output signal;

s^t denotes the current unit state;

θ denotes the trainable parameters of the network; and

$$\epsilon^{t,\theta} := \frac{ds^t}{d\theta}.$$

18. A neural network comprising one or more layers of neuronal units, wherein each neuronal unit has an internal state, wherein the neural network is configured to perform a method for training a neural network, the method comprising

providing training data comprising an input signal and an expected output signal to the neural network;

computing, for each neuronal unit, a spatial gradient component;

computing, for each neuronal unit, a temporal gradient component; and

updating the temporal and the spatial gradient component for each neuronal unit at each time instance of the input signal.

19. The neural network according to claim 18, wherein the neural network is further configured to update parameters of the neural network at each time instance as a function of the spatial and the temporal gradient components.

20. A computer program product for training a recurrent neural network, the computer program product comprising a computer readable storage medium having program instructions embodied therewith, the program instructions executable by the neural network to cause the neural network to perform a method comprising:

receiving training data comprising an input signal and an expected output signal;

computing, for each neuronal unit, a spatial gradient component;

computing, for each neuronal unit, a temporal gradient component; and

updating the temporal and the spatial gradient component for each neuronal unit at each time instance of the input signal.

21. The computer program product according to claim 20, the program instructions executable by the neural network to cause the neural network to update parameters of the neural network at each time instance as a function of the spatial and the temporal gradient components.

22. A computing system configured to perform a computer-implemented method for training parameters of a neural network, the network comprising one or more layers of neuronal units, wherein each neuronal unit has an internal state, wherein the method comprises:

providing training data comprising an input signal and an expected output signal to the neural network;

computing, for each neuronal unit, a spatial gradient component;

computing, for each neuronal unit, a temporal gradient component; and

updating the temporal and the spatial gradient component for each neuronal unit at each time instance of the input signal.

23. The computing system according to claim 22, the computing system comprising a memristive memory array.

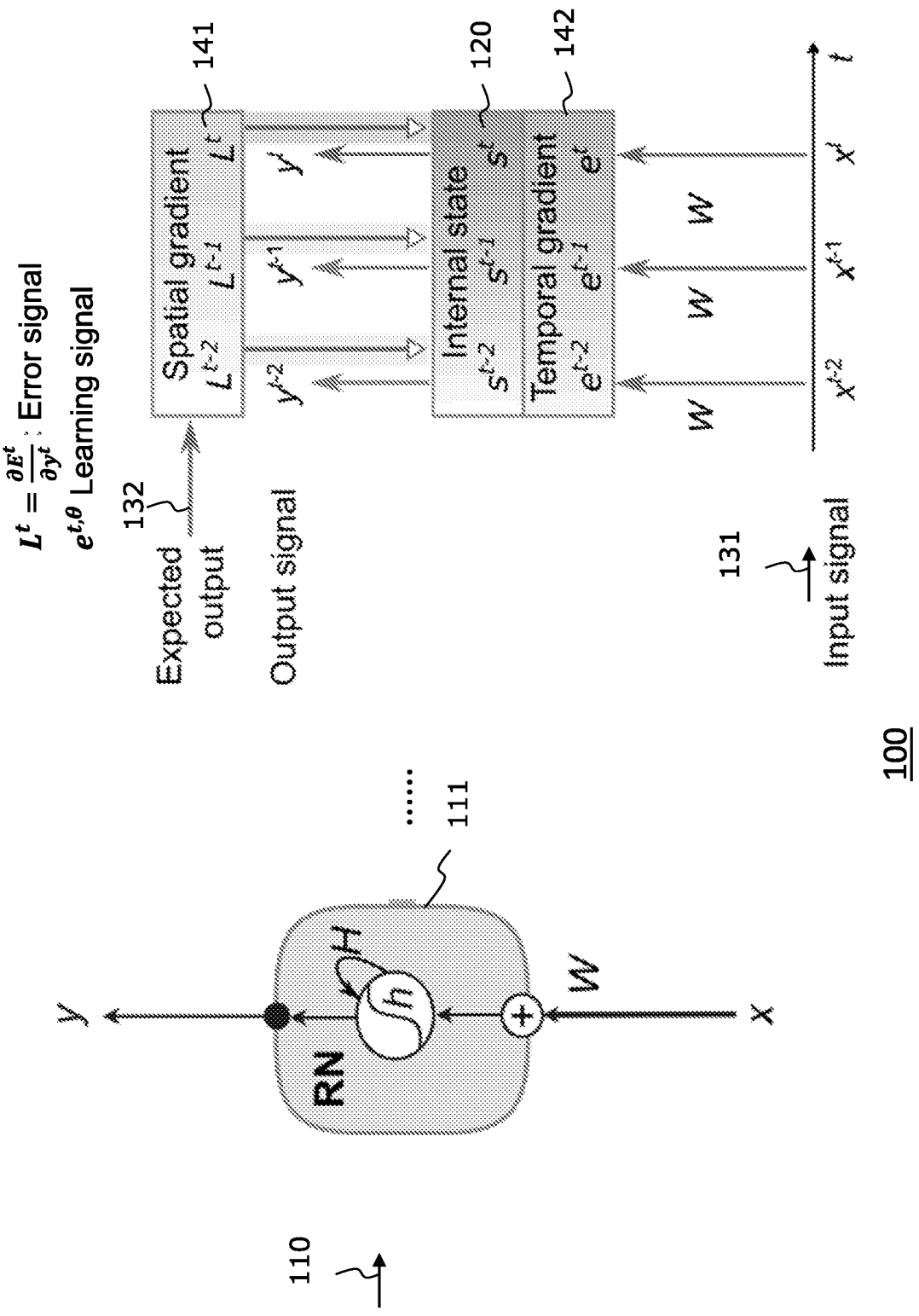

$L^t = \frac{\partial E^t}{\partial y^t}$: Error signal
$e^{t,\theta}$ Learning signal
132
Expected output
Output signal
Spatial gradient
L^{t-2} L^{t-1} L^t
141
y^{t-2} y^{t-1} y^t
Internal state
s^{t-2} s^{t-1} s^t
120
Temporal gradient
e^{t-2} e^{t-1} e^t
142
W W W
x^{t-2} x^{t-1} x^t t
131
Input signal
100
110
RN
H
f
W
x
y
111

FIG. 1

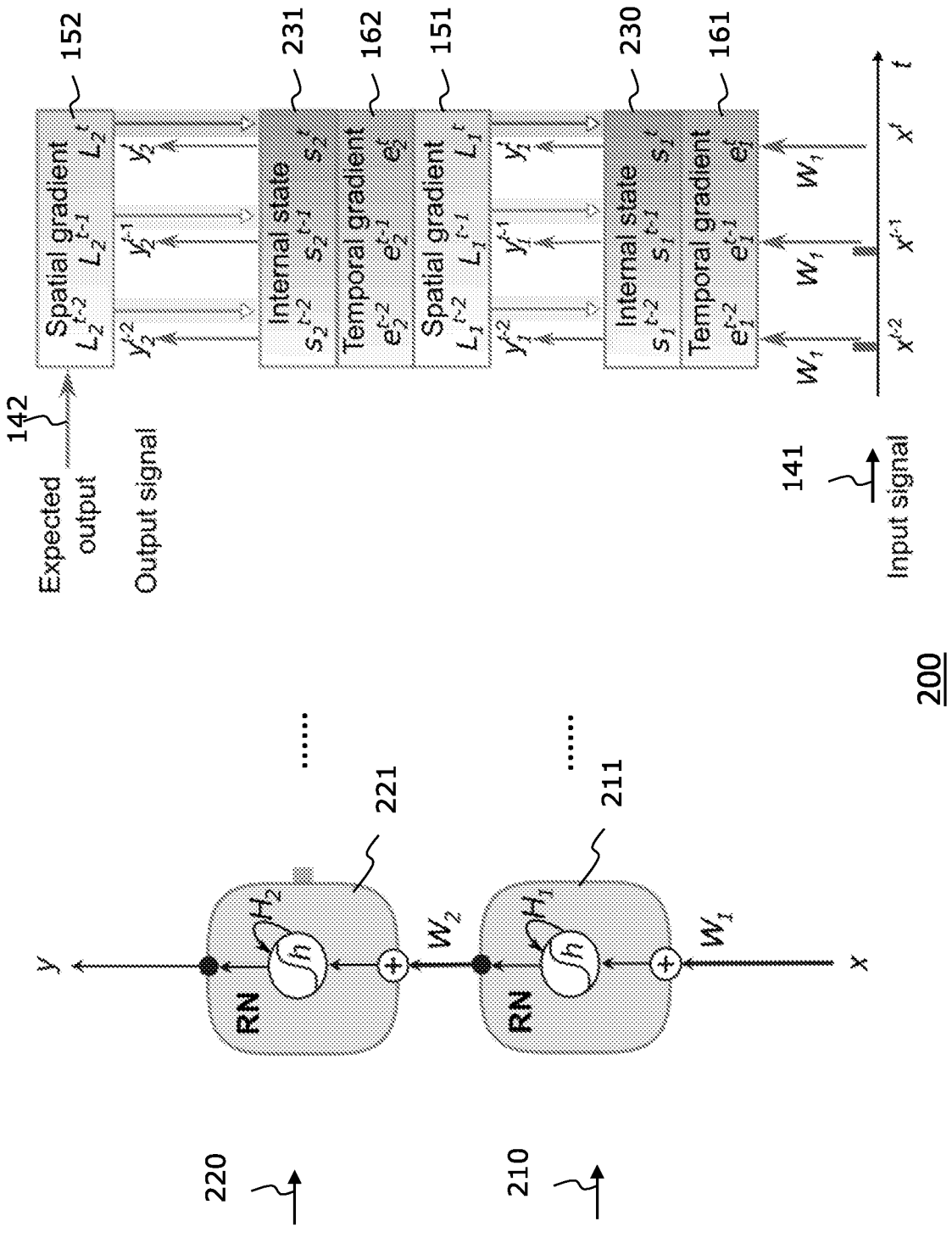

200
Input signal
141
Expected output
142
Output signal
Spatial gradient
Internal state
Temporal gradient
Spatial gradient
Internal state
Temporal gradient
152
231
162
151
230
161
RN
RN
221
211
220
210
W1
W2
H1
H2
x
y
t

FIG. 2

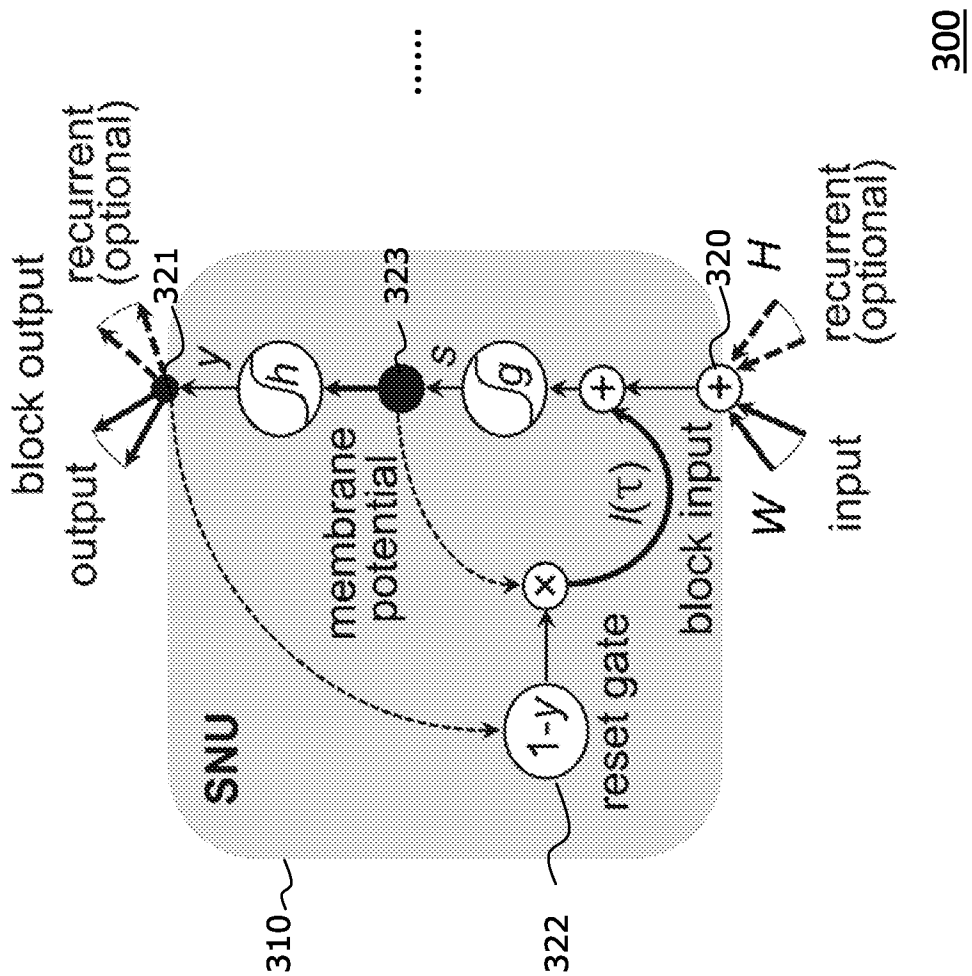
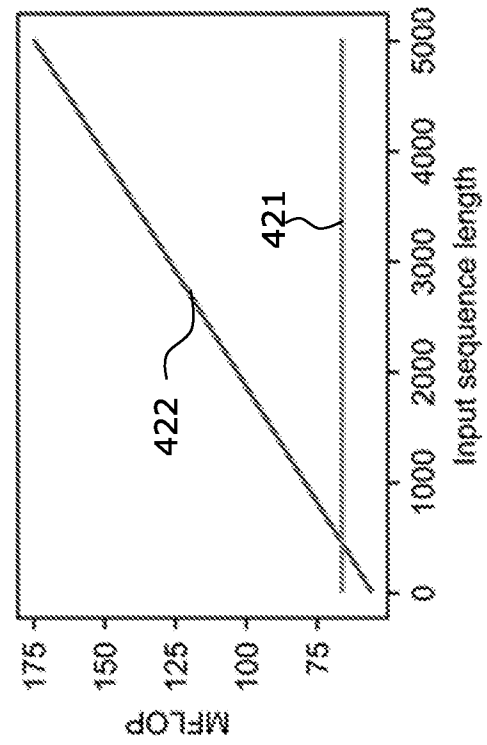
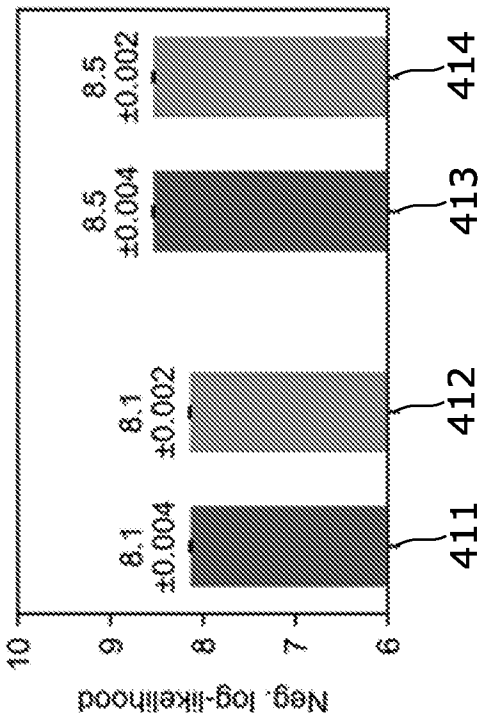

300
.....
block output
output
recurrent
(optional)
321
y
h
323
membrane
potential
s
g
+
+
320
H
recurrent
(optional)
input
W
SNU
x
1-y
reset gate
l(τ)
block input
310
322

FIG. 3



420

FIG. 4b



410

FIG. 4a

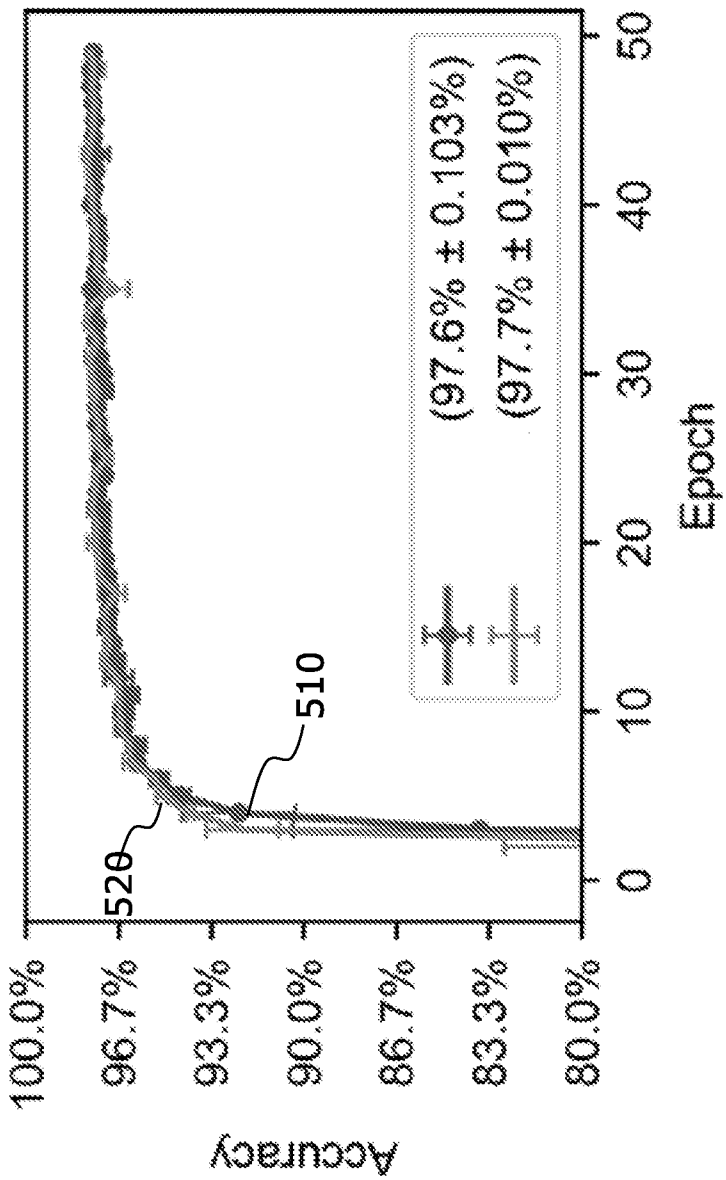

100.0%
96.7%
93.3%
90.0%
86.7%
83.3%
80.0%
Accuracy
0
10
20
30
40
50
Epoch
520
510
(97.6% ± 0.103%)
(97.7% ± 0.010%)

FIG. 5

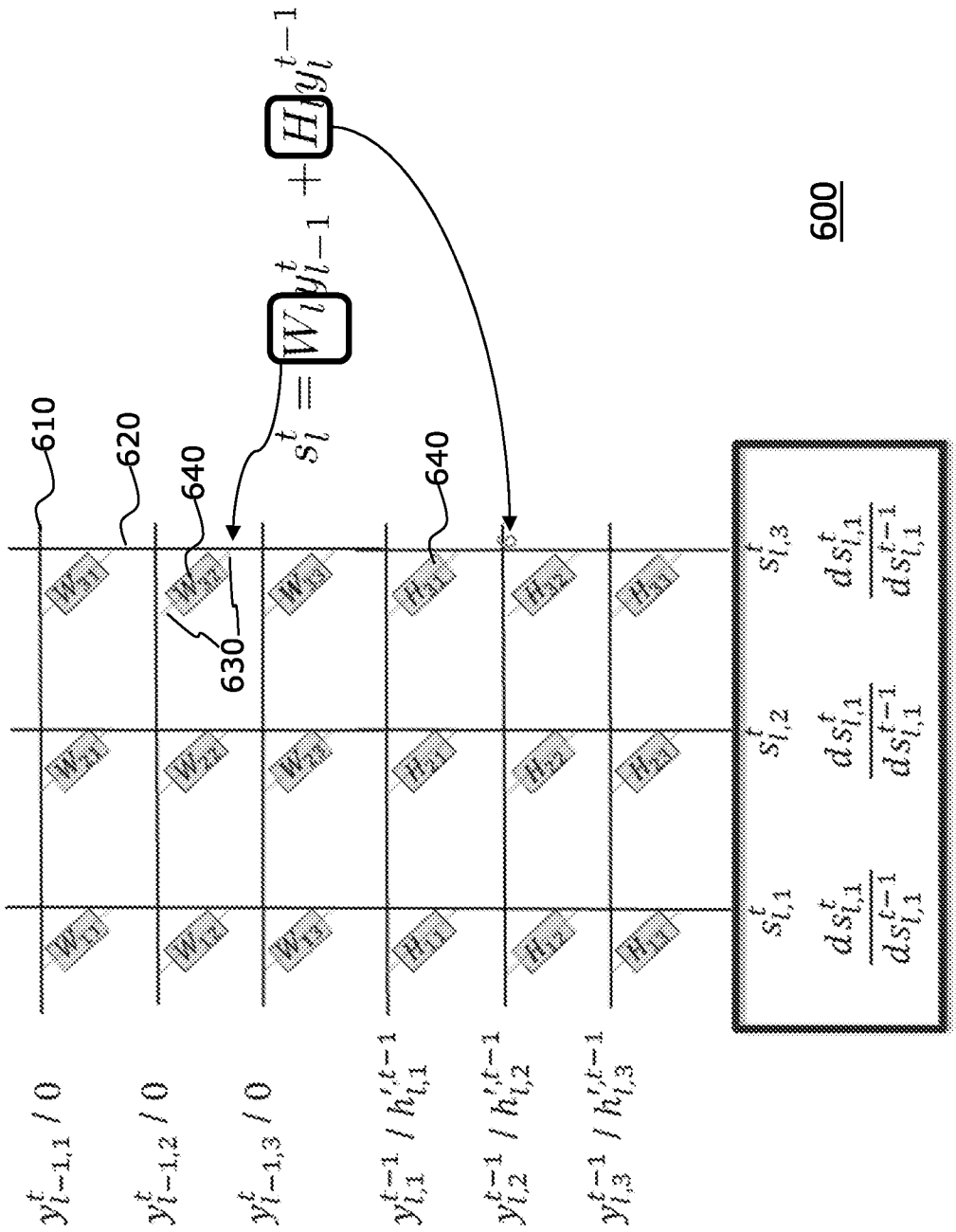

600
610
620
640
640
630
$s_l^t = W_l y_{l-1}^t + H y_l^{t-1}$
$y_{l-1,1}^t$ / 0
$y_{l-1,2}^t$ / 0
$y_{l-1,3}^t$ / 0
$y_{l,1}^{t-1}$ / $h_{l,1}^{t-1}$
$y_{l,2}^{t-1}$ / $h_{l,2}^{t-1}$
$y_{l,3}^{t-1}$ / $h_{l,3}^{t-1}$
$s_{l,1}^t$ $s_{l,2}^t$ $s_{l,3}^t$
$\frac{ds_{l,1}^t}{ds_{l,1}^{t-1}}$ $\frac{ds_{l,1}^t}{ds_{l,1}^{t-1}}$ $\frac{ds_{l,1}^t}{ds_{l,1}^{t-1}}$

FIG. 6

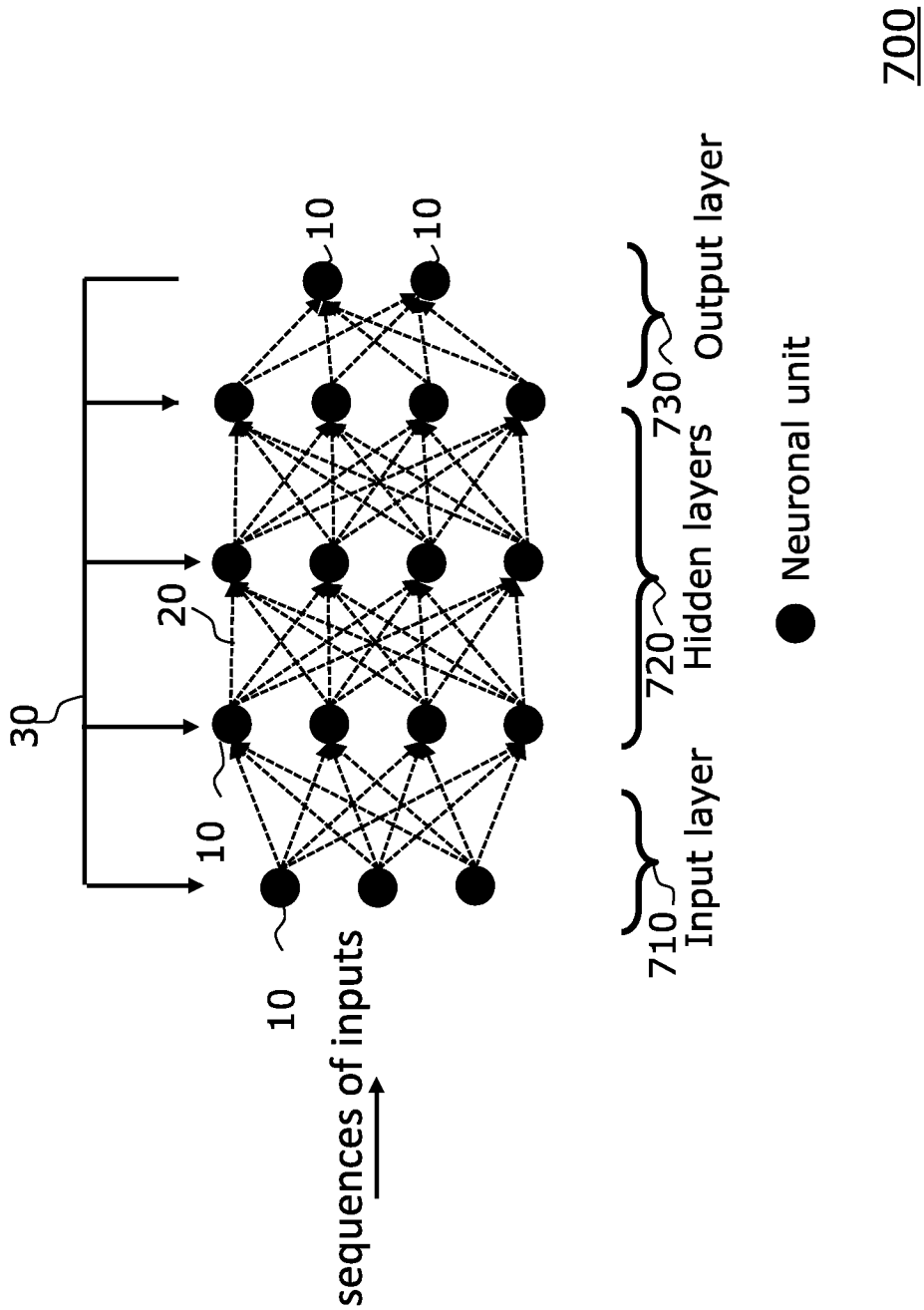

700
Neuronal unit
Input layer
Hidden layers
Output layer
710
720
730
10
20
30
sequences of inputs

FIG. 7

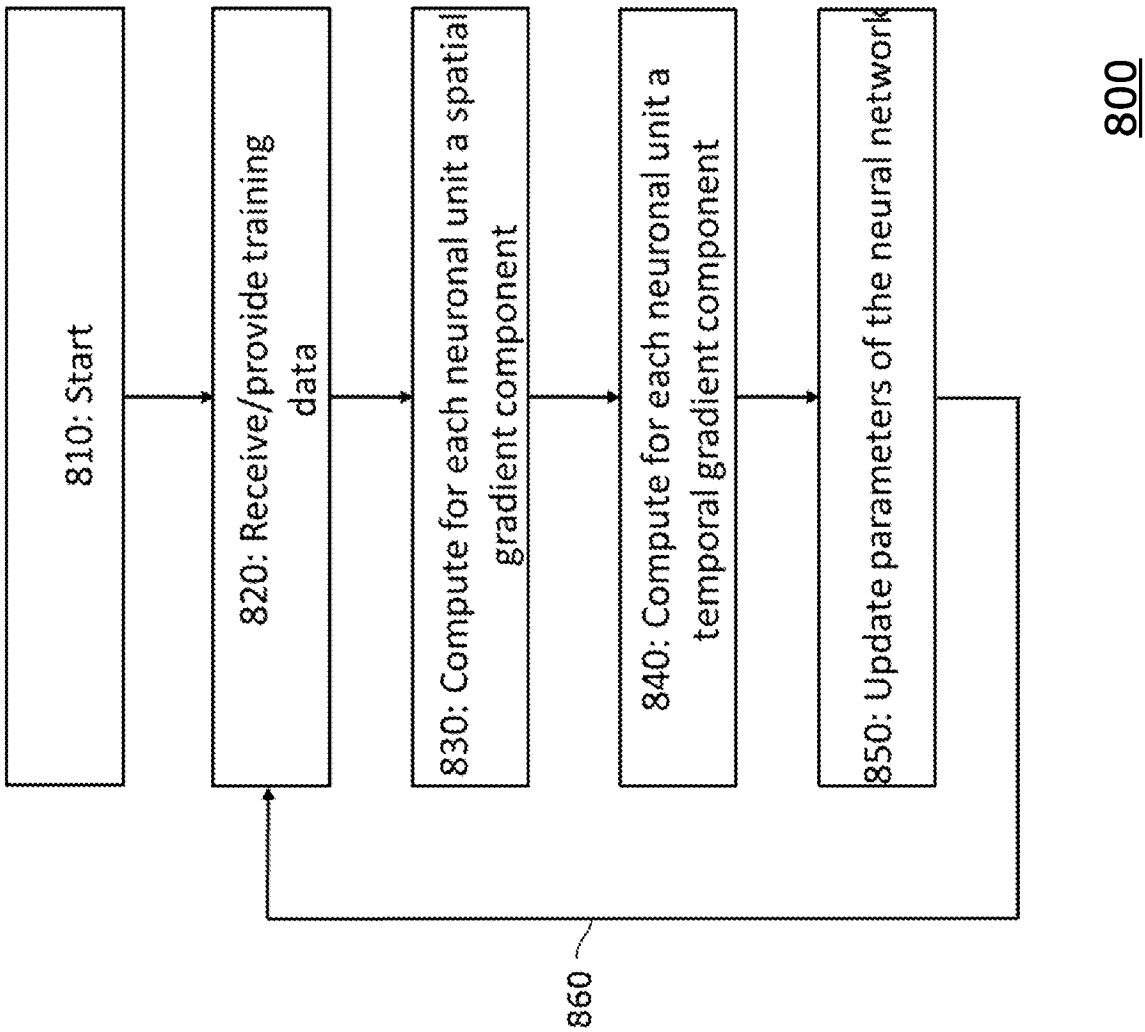

810: Start
820: Receive/provide training data
830: Compute for each neuronal unit a spatial gradient component
840: Compute for each neuronal unit a temporal gradient component
850: Update parameters of the neural network
860
800

FIG. 8

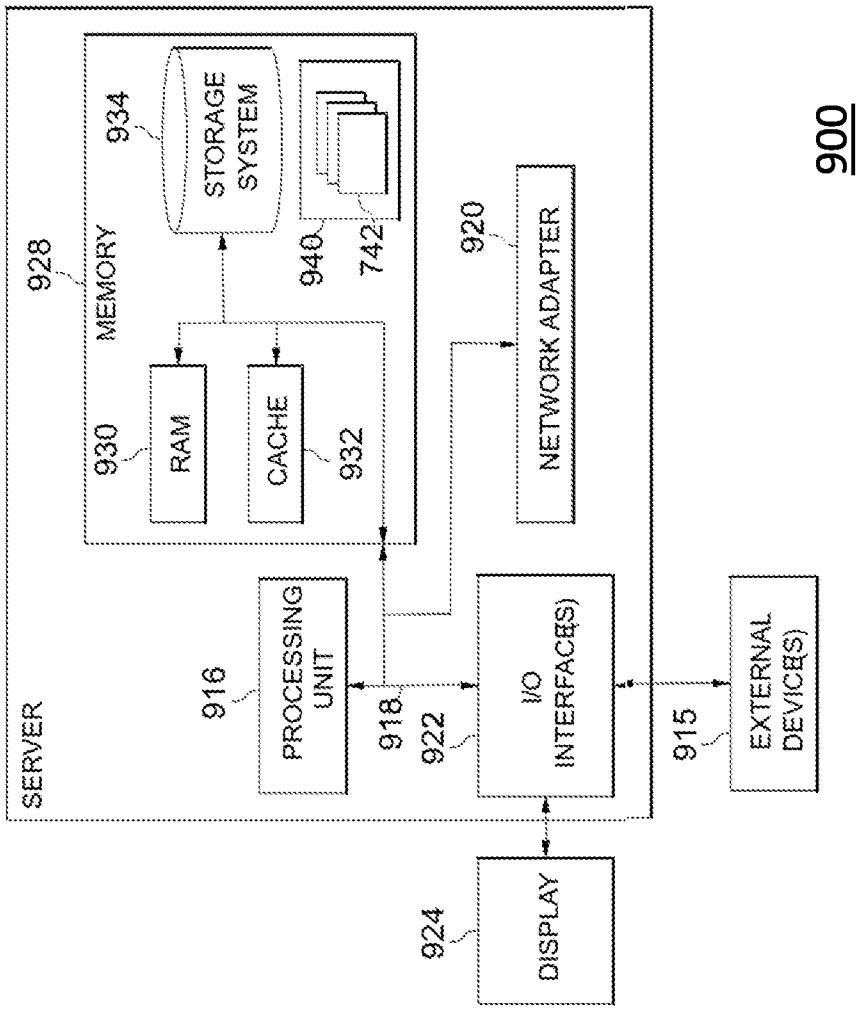

900
SERVER
928
MEMORY
934
STORAGE SYSTEM
940
742
930
RAM
CACHE
932
920
NETWORK ADAPTER
916
PROCESSING UNIT
918
922
I/O INTERFACE(S)
915
EXTERNAL DEVICE(S)
924
DISPLAY

FIG. 9

1001

Many state-of-the-art applications rely on multi-layer networks, in which the error E^t is only a function of the last output layer k, i.e., $E^t = E^t(y_k^t)$. According to embodiments of the invention, the state and output equations are adapted as follows

$$s_l^t = s_l^t(s_l^{t-1}, y_l^{t-1}, y_{l-1}^t, \theta_l) \tag{31}$$

$$y_l^t = y_l^t(s_l^t, \theta_l). \tag{32}$$

Using this reformulation, Equation 2 can be generalized as follows

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_{1 \le t \le T} \frac{\mathrm{d}E^t}{\mathrm{d}\theta_l} = \sum_{1 \le t \le T} \frac{\partial E^t}{\partial y_k^t} \left[\frac{\partial y_k^t}{\partial s_k^t} \frac{\mathrm{d}s_k^t}{\mathrm{d}\theta_l} + \frac{\partial y_k^t}{\partial \theta_l} \right]. \tag{33}$$

For the last layer of a multi-layer network, where $k = l$, Equation 33 corresponds to Equation 2 for a single layer. However, for the hidden layers, i.e., $k \neq l$, the term $\frac{\mathrm{d}s_k^t}{\mathrm{d}\theta_l}$ is expanded as follows

$$\frac{\mathrm{d}s_k^t}{\mathrm{d}\theta_l} = \frac{\partial s_k^t}{\partial \theta_l} + \frac{\partial s_k^t}{\partial y_k^{t-1}} \frac{\mathrm{d}y_k^{t-1}}{\mathrm{d}\theta_l} + \frac{\partial s_k^t}{\partial y_{k-1}^t} \frac{\mathrm{d}y_{k-1}^t}{\mathrm{d}\theta_l}. \tag{34}$$

We define a recursive term $\chi_{lm}^{t,\theta}$ as

$$\begin{aligned} \chi_{lm}^{t,\theta} &:= \frac{\mathrm{d}s_l^t}{\mathrm{d}\theta_m} \\ &= \frac{\partial s_l^t}{\partial s_m^{t-1}} \chi_{lm}^{t-1,\theta} + \frac{\partial s_l^t}{\partial y_l^{t-1}} \left(\frac{\partial y_l^{t-1}}{\partial s_l^{t-1}} \chi_{lm}^{t-1,\theta} + \frac{\partial y_l^{t-1}}{\partial \theta_m} \right) \\ &\quad + \frac{\partial s_l^t}{\partial y_{l-1}^t} \left(\frac{\partial y_{l-1}^t}{\partial s_{l-1}^t} \chi_{l-1m}^{t,\theta} + \frac{\partial y_{l-1}^t}{\partial \theta_m} \right) + \frac{\partial s_l^t}{\partial \theta_m}, \end{aligned} \tag{35}$$

with the following properties

$$c_l^{t,\theta} := \chi_{ll}^{t,\theta} \tag{36}$$

$$e_l^{t,\theta} := \frac{\mathrm{d}y_l^t}{\mathrm{d}\theta_l} = \left(\frac{\partial y_l^t}{\partial s_l^t} \chi_{ll}^{t,\theta} + \frac{\partial y_l^t}{\partial \theta_l} \right) \tag{37}$$

$$\begin{aligned} \chi_{lm}^{t<1,\theta} &= 0 \\ \chi_{ll+1}^{t,\theta} &= 0 \\ \chi_{lm}^{t,\theta} &= 0 \\ \chi_{lm<1}^{t,\theta} &= 0. \end{aligned}$$

The term $\chi_{lm}^{t,\theta}$ for $k \neq l$ contains a recursion in time, but additionally it contains a recursion in space, i.e., it depends on other layers, for example the $(k-1)$-th layer.

If we insert the term $\chi_{lm}^{t,\theta}$ in Equation 33 we obtain

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_{1 \le t \le T} \frac{\partial E^t}{\partial y_k^t} \left[\frac{\partial y_k^t}{\partial s_k^t} \chi_{kl}^{t,\theta} + \frac{\partial y_k^t}{\partial \theta_l} \right]. \tag{38}$$

FIG. 10

1003

The right-hand side of Equation 38 is expanded to a more complex expression

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_{1\leq t\leq T}\left[\frac{\mathrm{d}E^t}{\mathrm{d}y_k^t}\left[\frac{\partial y_k^t}{\partial s_k^t}\frac{\partial s_k^t}{\partial s_k^{t-1}}\chi_l^{t-1,\theta} + \frac{\partial y_k^t}{\partial s_k^t}\frac{\partial s_k^t}{\partial y_k^{t-1}}\left(\frac{\partial y_k^{t-1}}{\partial s_k^{t-1}}\chi_l^{t-1,\theta} + \frac{\partial y_k^{t-1}}{\partial \theta_l}\right)\right.\right.$$
$$+ \frac{\partial y_k^t}{\partial s_k^t}\frac{\partial s_k^t}{\partial y_{k-1}^t}\left(\frac{\partial y_{k-1}^t}{\partial s_{k-1}^t}\chi_l^{t,\theta}_{k-1} + \frac{\partial y_{k-1}^t}{\partial \theta_l}\right) + \frac{\partial y_k^t}{\partial s_k^t}\frac{\partial s_k^t}{\partial \theta_l}\Big]$$
$$\left. + \frac{\partial y_k^t}{\partial \theta_l}\right], \tag{39}$$

where the two recurrencies — $\chi_{l}^{t,\theta}_{k-1}$ in space, and $\chi_l^{t-1,\theta}$ in time — become apparent. When expanding $\chi_{l}^{t}_{k-1}$ far enough in space, it eventually reaches $\chi_l^{t,\theta} = e_l^t$. Therefore, we can rewrite Equation 39 as

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_{1\leq t\leq T}\left[\frac{\mathrm{d}E^t}{\mathrm{d}y_k^t}\left(\prod_{(k-l+1)\geq m'\geq 1}\frac{\partial y_{k-m'+1}^t}{\partial s_{k-m'+1}^t}\frac{\partial s_{k-m'+1}^t}{\partial y_{k-m'}^t}\right)\left(\frac{\partial y_l^t}{\partial s_l^t}\chi_l^{t,\theta} + \frac{\partial y_l^t}{\partial \theta_l}\right) + R\right], \tag{40}$$

where we collect all the remaining terms into a residual term R. In addition, we define a generalized learning signal L_l^t and a generalized eligibility trace $e_l^{t,\theta}$ as

$$L_l^t = \frac{\partial E^t}{\partial y_k^t}\left(\prod_{(k-l+1)\geq m'\geq 1}\frac{\partial y_{k-m'+1}^t}{\partial s_{k-m'+1}^t}\frac{\partial s_{k-m'+1}^t}{\partial y_{k-m'}^t}\right) \tag{41}$$

$$e_l^{t,\theta} = \left(\frac{\partial y_l^t}{\partial s_l^t}e_l^{t,\theta} + \frac{\partial y_l^t}{\partial \theta_l}\right), \tag{42}$$

see Equations 10-11. This allows to express the parameter update as

$$\frac{\mathrm{d}E}{\mathrm{d}\theta_l} = \sum_{1\leq t\leq T}\left[L_l^t e_l^{t,\theta} + R\right], \tag{43}$$

see Equation 13. By omitting the residual term R according to embodiments, we arrive at Equation 14.

FIG. 11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB2021/056026

A. CLASSIFICATION OF SUBJECT MATTER

G06N 3/08(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI, EPODOC, CNPAT, CNKI, IEEE: neural, network, layer?, train+, input+, output+, spatial gradient, temporal gradient, updat+, time, instance, predefined

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 106991474 A (HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY) 28 July 2017 (2017-07-28) claims 1-5, description, paragraphs [0045]-[0083], figures 1-6	1-23
A	CN 111126223 A (SHANXI UNIVERSITY) 08 May 2020 (2020-05-08) the whole document	1-23
A	US 8612286 B2 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 17 December 2013 (2013-12-17) the whole document	1-23
A	CN 106418849 A (XIAN SUANNIER ELECTRONIC TECHNOLOGY CO., LTD.) 22 February 2017 (2017-02-22) the whole document	1-23
A	WO 2017201511 A1 (GOOGLE LLC) 23 November 2017 (2017-11-23) the whole document	1-23

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 September 2021	**12 October 2021**

Name and mailing address of the ISA/CN	Authorized officer
National Intellectual Property Administration, PRC **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China**	**LI, Min**
Facsimile No. **(86-10)62019451**	Telephone No. **86-(10)-53961526**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/IB2021/056026

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 106991474 A	28 July 2017	CN 106991474 B	24 September 2019
CN 111126223 A	08 May 2020	None	
US 8612286 B2	17 December 2013	US 2010114671 A1	06 May 2010
CN 106418849 A	22 February 2017	None	
WO 2017201511 A1	23 November 2017	EP 3446259 A1	27 February 2019
		CN 109313721 A	05 February 2019
		US 2019220748 A1	18 July 2019