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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 November 2017 (23.11.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/201511 A1

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

(21) International Application Number:
PCT/US2017/033703

(22) International Filing Date:
19 May 2017 (19.05.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/339,785 20 May 2016 (20.05.2016) US

(71) Applicant: **GOOGLE LLC** [US/US]; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US).

(72) Inventors: **DENIL, Misha Man Ray**; 6 Pancras Square, London N1C 4AG (GB). **SCHAUL, Tom**; 6 Pancras Square, London N1C 4AG (GB). **ANDRYCHOWICZ, Marcin**; 7 Pancras Square, Kings Cross, London N1C 4AG (GB). **DE FREITAS, Joao Ferdinando Gomes**; 6 Pancras Square, London N1C 4AG (GB). **COLMENAREJO, Sergio Gomez**; 6 Pancras Square, London N1C 4AG (GB). **HOFFMAN, Matthew, William**; 7 Pancras Square, Kings Cross, London N1C 4AG (GB). **PFAU, David, Benjamin**; 7 Pancras Square, Kings Cross, London N1C 4AG (GB).

(74) Agent: **SHOGHI, Pooya** et al.; Fish & Richardson P.C., P.O. Box 1022, 3300 RBC Plaza, Minneapolis, MN 55440-1022 (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, DJ, 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, 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, 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).

(54) Title: TRAINING MACHINE LEARNING MODELS

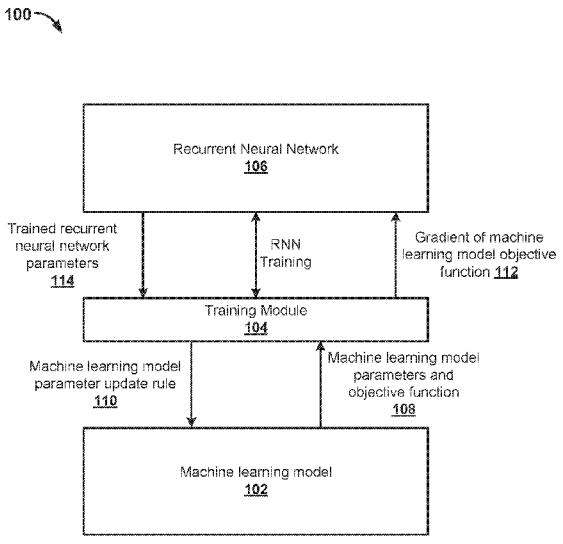

FIG. 1A

(57) Abstract: Methods, systems, and apparatus, including computer programs encoded on computer storage media for training machine learning models. One method includes obtaining a machine learning model, wherein the machine learning model comprises one or more model parameters, and the machine learning model is trained using gradient descent techniques to optimize an objective function; determining an update rule for the model parameters using a recurrent neural network (RNN); and applying a determined update rule for a final time step in a sequence of multiple time steps to the model parameters.

WO 2017/201511 A1

[Continued on next page]

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

TRAINING MACHINE LEARNING MODELS

BACKGROUND

[0001] This specification relates to neural networks.

[0002] Neural networks are machine learning models that employ one or more layers of nonlinear units to predict an output for a received input. Some neural networks include one or more hidden layers in addition to an output layer. The output of each hidden layer is used as input to the next layer in the network, i.e., the next hidden layer or the output layer. Each layer of the network generates an output from a received input in accordance with current values of a respective set of parameters.

SUMMARY

[0003] This specification describes how a system implemented as computer programs on one or more computers in one or more locations can replace hard-coded parameter optimization algorithms, e.g., gradient descent optimization algorithms, with a trainable deep recurrent neural network. Hand-designed update rules for the parameters of a machine learning model are replaced with a learned update rule.

[0004] In general, one innovative aspect of the subject matter described in this specification can be embodied in methods including obtaining a machine learning model, wherein (i) the machine learning model comprises one or more model parameters, and (ii) the machine learning model is trained using gradient descent techniques to optimize an objective function; for each time step in a plurality of time steps: determining an update rule for the model parameters for the time step using a recurrent neural network (RNN), comprising: providing as input to the RNN, a gradient of the objective function with respect to the model parameters for the time step; generating a respective RNN output from the provided input for the time step, wherein the RNN output comprises an update rule for the model parameters at the time step that is dependent on one or more RNN parameters; training the RNN using the generated output and a RNN objective function that depends on each preceding time step in the plurality of time steps, comprising determining RNN parameters that minimize the RNN objective function for the time step using gradient descent techniques; based on the determined RNN parameters, determining an update rule for the model parameters that minimizes the objective function for the time step; and applying the determined update rule for the time step to the model parameters.

[0005] Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods. A system of one or more computers can be configured to perform particular operations or actions by virtue of software, firmware, hardware, or any combination thereof installed on the system that in operation may cause the system to perform the actions. One or more computer programs can be configured to perform particular operations or actions by virtue of including instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions.

[0006] The foregoing and other embodiments can each optionally include one or more of the following features, alone or in combination. In some implementations applying the determined update rule for a final time step in the plurality of time steps to the model parameters generates trained model parameters.

[0007] In some implementations the machine learning model comprises a neural network.

[0008] In some implementations the determined update rule for the model parameters that minimizes the objective function is given by

$$\theta_{t+1} = \theta_t + g_t(\nabla \mathrm{f}(\theta_t), \phi)$$

wherein θ_t represents model parameters at time *t*, $\nabla \mathrm{f}(\theta_t)$ represents the gradient of objective function *f*, ϕ represents RNN parameters and g_t represents the RNN output for the time step *t*.

[0009] In some implementations the RNN operates coordinate-wise on the objective functions parameters.

[0010] In some implementations the RNN implements separate activations for each model parameter.

[0011] In some implementations applying the determined update rule for the time step to the model parameters comprises using a long short-term memory (LSTM) neural network.

[0012] In some implementations the LSTM network comprises two LSTM layers.

[0013] In some implementations the LSTM neural network shares parameters across different coordinates of the objective function.

[0014] In some implementations a subset of cells in each LSTM layer comprise global averaging units, wherein a global average unit is a unit whose update includes a step

that averages the activations of the units globally at each step across the different coordinate wise LSTMs.

[0015] In some implementations a same update rule is applied independently on each coordinate.

[0016] In some implementations the RNN is invariant to the order of the model parameters.

[0017] In some implementations the method further comprises providing a previous hidden state of the RNN as input to the RNN for the time step.

[0018] In some implementations the determined update rule for the model parameters that minimizes the objective function for the time step depends on a hidden state of the RNN for the time step.

[0019] In some implementations the RNN objective function is given by

$$\mathcal{L}(\phi) = E_f\left[\sum_{t=1}^{T} w_t f(\theta_t)\right]$$

where $\theta_{t+1} = \theta_t + g_t$, $\begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi)$, ϕ represents the RNN parameters, $f(\theta_t)$ represents the machine learning model objective function that depends on the machine learning model parameters θ at time t, $w_t \in \mathbb{R}_{\geq 0}$ represents weights associated with each time step t, g_t represents a RNN output for time t, h_t represents a hidden state of the RNN at time t, m represents the RNN and $\nabla_t = \nabla_\theta f(\theta_t)$.

[0020] In some implementations the method further comprises preprocessing the input to the RNN to disregard gradients that are smaller than a predetermined threshold.

[0021] In some implementations a trained machine learning model may be output that is based upon the obtained machine learning model with updated parameters based upon the implementations described above. The machine learning model may be used to process input data to generate output data. The input data may be data associated with a real-world environment and the output data may provide an output associated with the real-world environment.

[0022] The subject matter described in this specification can be implemented in particular embodiments so as to realize one or more of the following advantages.

[0023] A system for training machine learning models using a recurrent neural network, as described in this specification, may outperform systems that train machine

learning models using other methods, e.g., using hard-coded optimization algorithms. For example, machine learning models that have been trained using a recurrent neural network may perform respective machine learning tasks more accurately and efficiently.

[0024] A system for training machine learning models using a recurrent neural network, as described in this specification, may achieve a high degree of transfer. For example, a recurrent neural network trained on machine learning tasks with a first number of task parameters may be generalizable to machine learning tasks with a second, higher number of task parameters. Alternatively or in addition, the recurrent neural network may be generalizable to further machine learning tasks and/or different types of neural network inputs. Embodiments may therefore provide improvements in generation of machine learning models that may provide improved performance for processing data.

[0025] The details of one or more embodiments of the subject matter of this specification are set forth in the accompanying drawings and the description below. Other features, aspects, and advantages of the subject matter will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0026] FIG. 1A is an illustration of an example system for training a machine learning model.

[0027] FIG. 1B is an illustration of a data flow graph for computing the gradient of a recurrent neural network objective function.

[0028] FIG. 2 is a flow diagram of an example process for training a machine learning model.

[0029] FIG. 3 is a flow diagram of an example process for determining an update rule for machine learning model parameters using a recurrent neural network.

[0030] Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0031] FIG. 1A is a block diagram of an example system 100 for training a machine learning model. The system 100 is an example of a system implemented as computer programs on one or more computers in one or more locations, in which the systems, components, and techniques described below can be implemented.

[0032] The system 100 includes a machine learning model 102, a training module 104, and a recurrent neural network (RNN) 106. The machine learning model 102 can be trained to perform a machine learning task. For example, the machine learning model may be trained to perform classification tasks. The classification tasks are typically tasks associated with real-world input data such as speech recognition, image recognition or natural language processing, regression tasks, or robot learning tasks. For example, the machine learning models may include deep neural networks e.g., convolutional neural networks, or support vector machines.

[0033] The machine learning model 102 has a set of machine learning model parameters. For example, in cases where the machine learning model 102 includes a neural network, the machine learning model parameters may include neural network weights for the neural network. As another example, in cases where the machine learning model 102 includes a support vector machine, the machine learning model parameters may include kernel parameters or soft margin parameters for the support vector machine.

[0034] The machine learning model 102 can be trained to perform the machine learning task using gradient descent techniques to optimize a machine learning model objective function. For example, in cases where the machine learning model 102 is a neural network, the machine learning model may be trained to perform a respective machine learning task using backpropagation of errors. During a backpropagation training process, training inputs are processed by the neural network to generate respective neural network outputs. The outputs are then compared to a desired or known output using an objective function, e.g., a loss function, and error values are determined. The error values are used to calculate a gradient of the objective function with respect to the neural network parameters. The gradient is then used as input to an update rule to determine an update for the neural network parameters that minimizes the objective function. One example of a conventional update rule is given by equation (1) below.

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t) \qquad (1)$$

In equation (1), θ_t represents the neural network parameters at time t, α_t represents a learning rate at time t, and $f(\theta_t)$ represents the objective function.

[0035] The training module 104 communicates with the machine learning models 102 and the RNN 106. The training module 104 is configured to train the machine learning

model 102 by determining a learned parameter update rule for the machine learning model parameters using the RNN 106. The learned parameter update rule for the machine learning model parameters can be implemented over a sequence of time steps $t=1, \ldots, T$ to adjust the values of the machine learning model parameters from initial or current values, e.g., at time $t=1$, to trained values, e.g., at time $t=T$. A learned update rule for a set of machine learning model parameters for time step $t+1$ is given by equation (2) below.

$$\theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi) \qquad (2)$$

In equation (2), θ_t represents the machine learning model parameters at time t, $\nabla f(\theta_t)$ represents the gradient of the machine learning model objective function f, ϕ represents RNN 106 parameters and g_t represents a RNN output for the time step t in accordance with current values of the RNN parameters.

[0036] To determine the above learned update rule for time $t+1$, the training module 104 is configured to compute or obtain a gradient of the machine learning model objective function at time t with respect to the machine learning model parameters at time t. For example, the training module 104 may be configured to receive data representing machine learning model parameters at time t and objective function at time t 108, and to compute data representing the gradient of the machine learning model objective function with respect to the machine learning model parameters at time t. The training module 104 is configured to provide obtained or computed gradients to the RNN 106 as input. For example, the training module 104 may be configured to provide data representing the gradient of the machine learning model objective function at time t 112 as input to the RNN 106.

[0037] The RNN 106 is configured to process the received data representing the gradient of the machine learning model objective function at time t 112 to generate a respective RNN output for time t that is dependent on the one or more RNN parameters ϕ, e.g., as represented by g_t described above with reference to equation (2). Processing received RNN inputs to generate respective RNN outputs is described in more detail below.

[0038] The training module 104 is configured to update the values of the RNN parameters ϕ whilst training the machine learning model 102. Updating the values of the RNN parameters includes determining values of the RNN parameters ϕ that minimize a RNN objective function using gradient descent techniques. In some implementations the RNN

objective function is given by equation (3) below.

$$\mathcal{L}(\phi) = E_f\left[\sum_{t=1}^{T} w_t f(\theta_t)\right] \qquad (3)$$

where

$$\theta_{t+1} = \theta_t + g_t,$$
$$\begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi).$$

In equation (3), ϕ represents the RNN parameters, $f(\theta_t)$ represents the machine learning model objective function that depends on the machine learning model parameters θ at time t, $w_t \in \mathbb{R}_{\geq 0}$ represents weights, e.g., predetermined weights, associated with each time step t, g_t represents a RNN output for time t, h_t represents a hidden state of the RNN at time t, m represents the RNN and $\nabla_t = \nabla_\theta f(\theta_t)$.

[0039] The training module 104 is configured to determine the learned update rule for time $t+1$ in equation (2) above using the values of the RNN parameters ϕ for time t and gradients of respective machine learning model objective functions f. The learned update rule 110 may then be applied to the machine learning model parameters to update the machine learning model 102. This process may be iteratively repeated over a sequence of time steps $t=1, .., T$ to generate a trained machine learning model. In some implementations the number of time steps T may be a predetermined number, e.g., a number chosen based on available memory in the system 100. For example, T may be chosen as the highest number possible, given the available memory constraint. In some cases a trained machine learning model may generated when the machine learning model converges, e.g., the machine learning model parameters converge towards trained values. In these cases, the number of time steps T depends on the convergence rate.

[0040] As described above, the recurrent neural network 106 has RNN parameters, e.g., RNN weights. The RNN 106 is configured to receive a RNN input at each time step in a sequence of multiple time steps, e.g., data representing a gradient of a machine learning model objective function with respect to machine learning model parameters 112. In some implementations the RNN 106 may be invariant to the order of the machine learning model parameters. That is, interfacing between the RNN 106 and the machine learning model 102 may require fixing a particular order of the parameters of the machine learning model 102, e.g., numbering parameters of the machine learning model 102 and putting them into a

list. The ordering may be arbitrary, e.g., a predetermined order, but must be fixed so that outputs of the RNN 106 may be matched to parameters of the machine learning model 102. Invariance of the RNN 106 to the order of the machine learning model parameters enables the same results regardless of which ordering is picked.

[0041] The RNN 106 processes each received RNN input to generate a respective RNN output for the time step in accordance with the RNN parameters, e.g., an update rule for the machine learning model parameters that is dependent on one or more of the RNN parameters. The RNN 106 may be trained to generate RNN outputs from received inputs using gradient descent techniques to optimize a RNN objective function.

[0042] In some implementations the RNN 106 may be a fully connected RNN. In other implementations the RNN 106 may be a coordinate-wise RNN that operates on each RNN parameter individually. This architecture may be used when the number of RNN parameters exceeds a parameter threshold, e.g., when the number of RNN parameters is of the order of tens of thousands of parameters. The RNN 106 may be configured to operate on RNN parameters individually by using separate activations for each machine learning model parameter. In this manner, the RNN 106 may be invariant to the order of parameters in the network, since a same parameter update rule may be used independently on each parameter.

[0043] In some implementations the RNN 106 may include one or more Long Short-Term Memory (LSTM) neural network layers, e.g., two LSTM neural network layers. A LSTM neural network layer is a neural network layer that has one or more LSTM memory blocks. In these implementations, at each time step, the RNN 106 may be configured to receive as input data representing a gradient of machine learning model objective functions with respect to a single machine learning model parameter together with a previous hidden state of the RNN. The RNN 106 may then generate as output an update for the corresponding machine learning model parameter. In some implementations the LSTM neural network layers may share layer parameters across different coordinates of the machine learning model objective function. In some implementations the LSTM neural network layers may include one or more global averaging cells, i.e., units whose update includes a step that averages the activations of the units globally at each step across the different coordinate wise LSTMs.

[0044] In some implementations the RNN inputs, e.g., data representing the gradient of the machine learning model objective function 112 and data representing generated update rules for the machine learning model parameters, may be rescaled using one or more constants. For example, the training module 104 may be configured to rescale the RNN inputs or outputs to ensure that the RNN inputs and outputs are neither too small nor too

large. For example, the training module 104 may be configured to preprocess the RNN inputs according to equation (4) below.

$$\nabla^k \rightarrow \begin{cases} \left(\frac{\log|\nabla|}{p}, \quad \text{sgn}(\nabla)\right) & \text{if } |\nabla| \geq e^{-p} \\ (-1, e^p \nabla) & \text{otherwise} \end{cases} \tag{4}$$

In equation (4) $p > 0$ is a parameter controlling how small gradients are disregarded. In some implementations $p = 10$. Equation (4) considers the magnitude and direction components of the gradient separately – in some cases the magnitude component is a problematic component, so it gets mapped into log space (softly from above and truncated from below). The direction component, which is important for optimization, is preserved. Preprocessing RNN according to equation (4) reduces the range in which the scale of the gradient can change over training.

[0045] FIG. 1B is an illustration of an example data flow graph 150 for computing the gradient of a recurrent neural network objective function. For example, the data flow graph 150 may be used to compute the gradient of RNN 106 using machine learning model parameters of machine learning model 102.

[0046] In the example data flow graph 150, θ_t represents machine learning model 102 parameters at time t, and f_t represents machine learning model 102 objective function at time t, ∇_t represents the gradient of the objective function f_t with respect to the parameters θ_t, h_t represents the state of the RNN at time t, g_t represents a RNN output at time t, and m represents the RNN 106.

[0047] FIG. 2 is a flow diagram of an example process 200 for training a machine learning model. For convenience, the process 200 will be described as being performed by a system of one or more computers located in one or more locations. For example, a machine learning model training module, e.g., the training module 104 of FIG. 1A, can perform the process 200.

[0048] The system obtains data specifying a machine learning model (step 202). For example, the machine learning model may include a machine learning model that may be trained to perform a machine learning task, including a classification task such as speech recognition, image recognition or natural language processing, regression task or robot learning task.

[0049] The machine learning model has a respective set of machine learning model parameters. For example, as described above with reference to FIG. 1A, in some implementations the machine learning model may include a neural network. In these implementations the machine learning model parameters may include neural network parameters, e.g., neural network weights, for the neural network. The machine learning model is a machine learning model that is trained using gradient descent techniques to optimize a respective objective function.

[0050] For each time step in a sequence of time steps, the system determines an update rule for the machine learning model parameters for the time step using a recurrent neural network (RNN) (step 204). The RNN includes one or more RNN parameters and is trained using gradient descent techniques to optimize a RNN objective function. The update rule for the machine learning model parameters for the time step is a parameterized update rule – that is a function of update rule parameters – that may be used to adjust the values of the machine learning model parameters. Determining the update rule for the machine learning model parameters using the RNN includes training the RNN to determine RNN parameters that minimize the RNN objective function, and using trained RNN parameters to determine a final update rule that is used to generate the trained machine learning model. Determining an update rule for model parameters using a RNN is described in more detail below with reference to FIG. 3.

[0051] For each time step in the sequence of time steps, the system applies the determined update rule for the time step to the machine learning model parameters (step 206). In some implementations a same update rule is applied independently to each of the machine learning model parameters, e.g., using coordinate-wise network architecture as described above with reference to FIG. 1A.

[0052] Sequential application of the determined update rules for each time step in the sequence of time steps $t=1, \ldots, T$, adjusts the values of the machine learning model parameters from initial values, e.g., at time $t=1$, to trained values, e.g., at time $t=T$. Once trained, the machine learning model may be used to perform its respective machine learning task.

[0053] FIG. 3 is a flow diagram of an example process 300 for determining an update rule for a set of machine learning model parameters using a recurrent neural network (RNN). The example process 300 may be performed for each time step in a sequence of multiple time steps. For convenience, the process 300 will be described as being performed by a system of one or more computers located in one or more locations. For example, a machine learning

model training module, e.g., the training module 104 of FIG. 1A, can perform the process 300.

[0054] The system provides a gradient of the machine learning model objective function with respect to the machine learning model parameters for the time step *t* as input to the RNN (step 302). Optionally, the system may further provide a previous hidden state of the RNN as input to the RNN for the time step.

[0055] The system generates a respective RNN output from the provided input for the time step (step 304). The RNN output corresponds to g_t in equation (2) above, and may be used to determine the update rule given by equation (2) above for the machine learning model parameters at the time step that is dependent on one or more RNN parameters.

[0056] The system trains the RNN using the generated output and a RNN objective function that depends on each preceding time step in the sequence of multiple time steps (step 306). In some implementations the RNN objective function is given by equation (3) above, which is repeated below for clarity.

$$\mathcal{L}(\phi) = E_f\left[\sum_{t=1}^{T} w_t f(\theta_t)\right] \qquad (3)$$

$$\text{where } \theta_{t+1} = \theta_t + g_t, \ \begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi).$$

In equation (3), ϕ represents the RNN parameters, $f(\theta_t)$ represents the machine learning model objective function that depends on the machine learning model parameters θ at time *t*, $w_t \in \mathbb{R}_{\geq 0}$ represents weights, e.g., predetermined weights, associated with each time step, g_t represents the RNN output for the time *t*, h_t represents a hidden state of the RNN at time t, *m* represents the RNN and the notation $\nabla_t = \nabla_\theta f(\theta_t)$ is used. In some implementations $w_t > 0$, e.g., at intermediate points along the trajectory. For example, in some cases $w_t = 1$ for all *t*.

[0057] The system trains the RNN by determining values of the RNN parameters ϕ that minimize the RNN objective function $\mathcal{L}(\phi)$ for the time step using gradient descent techniques. For example, the system may compute a gradient estimate $\partial\mathcal{L}(\phi)/\partial\theta$ by sampling a random function *f* and applying backpropagation techniques, as described above with reference to FIGS. 1A and 1B. In some implementations it is assumed that gradients of the machine learning model does not depend on the RNN parameters ϕ, i.e., $\partial\nabla_t/\partial\phi = 0$.

[0058] Based on the determined RNN parameters ϕ, the system determines an update rule for the machine learning model parameters that minimizes the machine learning model objective functions for the time step (step 308). In some implementations the determined update rule for the machine learning model parameters that minimizes the machine learning model objective functions is given by equation (2) above, which is repeated below for clarity.

$$\theta_{t+1} = \theta_t + g_t(\nabla \mathrm{f}(\theta_t), \phi) \qquad (2)$$

In equation (2), θ_t represents machine learning model parameters at time t, $\nabla \mathrm{f}(\theta_t)$ represents the gradient of objective function f, as described above with reference to step 402, ϕ represents the determined values of the RNN parameters and g_t represents the RNN output for the time step. Although not shown in the above equation, in some implementations the determined update rule for the model parameters that minimizes the objective functions for the time step further depends on a hidden state h_t of the RNN for the time step.

[0059] In some implementations a learned update rule, as given by equation (2) above, may be applied to other machine learning models that are configured to perform similar machine learning tasks, e.g., machine learning tasks with a similar structure. For example, the learned update rule may be applied to a second machine learning model that is configured to perform a same machine learning task as the first machine learning model (e.g., the machine learning model 102 of FIG. 1A), but where the second machine learning model includes a different number of hidden units or neural network layers than the first machine learning model. As another example, the learned update rule may be applied to a second machine learning model that is configured to perform a same machine learning task as the first machine learning model but where the second machine learning model includes a different activation function to the first machine learning model.

[0060] Applying the learned update rule to other machine learning models in these examples can be achieved using the coordinate-wise RNN described above with reference to FIG 1, e.g., a neural network that uses a single coordinate to define the RNN and shares RNN parameters across different machine learning model parameters. Different behavior on each coordinate may be achieved using separate activation functions for each objective function parameter. The learned update rule may be implemented for each coordinate using the RNN, e.g., a two-layer LSTM network, using forget gate architecture. The network takes as input

the machine learning model gradient for a single coordinate as well as the previous hidden state and outputs the update for the corresponding machine learning model parameter.

[0061] Embodiments of the subject matter and the functional operations described in this specification can be implemented in digital electronic circuitry, in tangibly-embodied computer software or firmware, in computer hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them.

[0062] Embodiments of the subject matter described in this specification can be implemented as one or more computer programs, i.e., one or more modules of computer program instructions encoded on a tangible non transitory program carrier for execution by, or to control the operation of, data processing apparatus. Alternatively or in addition, the program instructions can be encoded on an artificially generated propagated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus for execution by a data processing apparatus. The computer storage medium can be a machine-readable storage device, a machine-readable storage substrate, a random or serial access memory device, or a combination of one or more of them. The computer storage medium is not, however, a propagated signal.

[0063] The term "data processing apparatus" encompasses all kinds of apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit). The apparatus can also include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

[0064] A computer program (which may also be referred to or described as a program, software, a software application, a module, a software module, a script, or code) can be written in any form of programming language, including compiled or interpreted languages, or declarative or procedural languages, and it can be deployed in any form, including as a stand alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program may, but need not, correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data, e.g., one or more scripts stored in a markup language document, in a single file dedicated to the program in question, or in multiple coordinated files, e.g., files

that store one or more modules, sub programs, or portions of code. A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

[0065] As used in this specification, an "engine," or "software engine," refers to a software implemented input/output system that provides an output that is different from the input. An engine can be an encoded block of functionality, such as a library, a platform, a software development kit ("SDK"), or an object. Each engine can be implemented on any appropriate type of computing device, e.g., servers, mobile phones, tablet computers, notebook computers, music players, e-book readers, laptop or desktop computers, PDAs, smart phones, or other stationary or portable devices, that includes one or more processors and computer readable media. Additionally, two or more of the engines may be implemented on the same computing device, or on different computing devices.

[0066] The processes and logic flows described in this specification can be performed by one or more programmable computers executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

[0067] Computers suitable for the execution of a computer program include, by way of example, can be based on general or special purpose microprocessors or both, or any other kind of central processing unit. Generally, a central processing unit will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a central processing unit for performing or executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Moreover, a computer can be embedded in another device, e.g., a mobile telephone, a personal digital assistant (PDA), a mobile audio or video player, a game console, a Global Positioning System (GPS) receiver, or a portable storage device, e.g., a universal serial bus (USB) flash drive, to name just a few.

[0068] Computer readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks;

and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[0069] To provide for interaction with a user, embodiments of the subject matter described in this specification can be implemented on a computer having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input. In addition, a computer can interact with a user by sending documents to and receiving documents from a device that is used by the user; for example, by sending web pages to a web browser on a user's client device in response to requests received from the web browser.

[0070] Embodiments of the subject matter described in this specification can be implemented in a computing system that includes a back end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front end component, e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation of the subject matter described in this specification, or any combination of one or more such back end, middleware, or front end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN") and a wide area network ("WAN"), e.g., the Internet.

[0071] The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

[0072] While this specification contains many specific implementation details, these should not be construed as limitations on the scope of any invention or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this specification in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination.

Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[0073] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system modules and components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

[0074] Particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In certain implementations, multitasking and parallel processing may be advantageous.

WHAT IS CLAIMED IS:

1. A method implemented by one or more computers, comprising:

obtaining a machine learning model, wherein (i) the machine learning model comprises one or more model parameters, and (ii) the machine learning model is trained using gradient descent techniques to optimize an objective function;

for each time step in a plurality of time steps:

determining an update rule for the model parameters for the time step using a recurrent neural network (RNN), comprising:

providing as input to the RNN, a gradient of the objective function with respect to the model parameters for the time step;

generating a respective RNN output from the provided input for the time step, wherein the RNN output comprises an update rule for the model parameters at the time step that is dependent on one or more RNN parameters;

training the RNN using the generated output and a RNN objective function that depends on each preceding time step in the plurality of time steps, comprising determining RNN parameters that minimize the RNN objective function for the time step using gradient descent techniques;

based on the determined RNN parameters, determining an update rule for the model parameters that minimizes the objective function for the time step; and

applying the determined update rule for the time step to the model parameters.

2. The method of claim 1, wherein applying the determined update rule for a final time step in the plurality of time steps to the model parameters generates trained model parameters.

3. The method of claim 1 or 2, wherein the machine learning model comprises a neural network.

4. The method of claim 1, 2 or 3, wherein the determined update rule for the model parameters that minimizes the objective function is given by

$$\theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi)$$

wherein θ_t represents model parameters at time t, $\nabla f(\theta_t)$ represents the gradient of objective function f, ϕ represents RNN parameters and g_t represents the RNN output for the time step t.

5. The method of any preceding claim, wherein the RNN operates coordinate-wise on the objective function parameters.

6. The method of any preceding claim, wherein the RNN implements separate activations for each model parameter.

7. The method of any preceding claim, wherein applying the determined update rule for the time step to the model parameters comprises using a long short-term memory (LSTM) neural network.

8. The method of claim 7, wherein the LSTM neural network comprises two LSTM layers.

9. The method of claim 7 or 8, wherein the LSTM neural network shares parameters across different coordinates of the objective function.

10. The method of claim 7, 8 or 9, wherein a subset of cells in each LSTM layer comprise global averaging units, wherein a global average unit is a unit whose update includes a step that averages the activations of the units globally at each step across the different coordinate wise LSTMs.

11. The method of any preceding claim, wherein a same update rule is applied independently on each coordinate.

12. The method of any preceding claim, wherein the RNN is invariant to the order of the model parameters.

13. The method of any preceding claim, further comprising providing a previous hidden state of the RNN as input to the RNN for the time step.

14. The method of any preceding claim, wherein the determined update rule for the model parameters that minimizes the objective function for the time step depends on a hidden state of the RNN for the time step.

15. The method of any preceding claim, wherein the RNN objective function is given by

$$\mathcal{L}(\phi) = E_f\left[\sum_{t=1}^{T} w_t f(\theta_t)\right]$$

where $\theta_{t+1} = \theta_t + g_t$, $\begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} = m(\nabla_t, h_t, \phi)$, ϕ represents the RNN parameters, $f(\theta_t)$ represents the machine learning model objective function that depends on the machine learning model parameters θ at time t, $w_t \in \mathbb{R}_{\geq 0}$ represents weights associated with each time step t, g_t represents a RNN output for time t, h_t represents a hidden state of the RNN at time t, m represents the RNN and $\nabla_t = \nabla_\theta f(\theta_t)$.

16. The method of any preceding claim, further comprising preprocessing the input to the RNN to disregard gradients that are smaller than a predetermined threshold.

17. A system comprising one or more computers and one or more storage devices storing instructions that are operable, when executed by the one or more computers, to cause the one or more computers to perform the operations comprising a method according to any preceding claim.

18. A computer storage medium encoded with instructions that, when executed by one or more computers, cause the one or more computers to perform the operations comprising a method according to any one of claims 1 to 16.

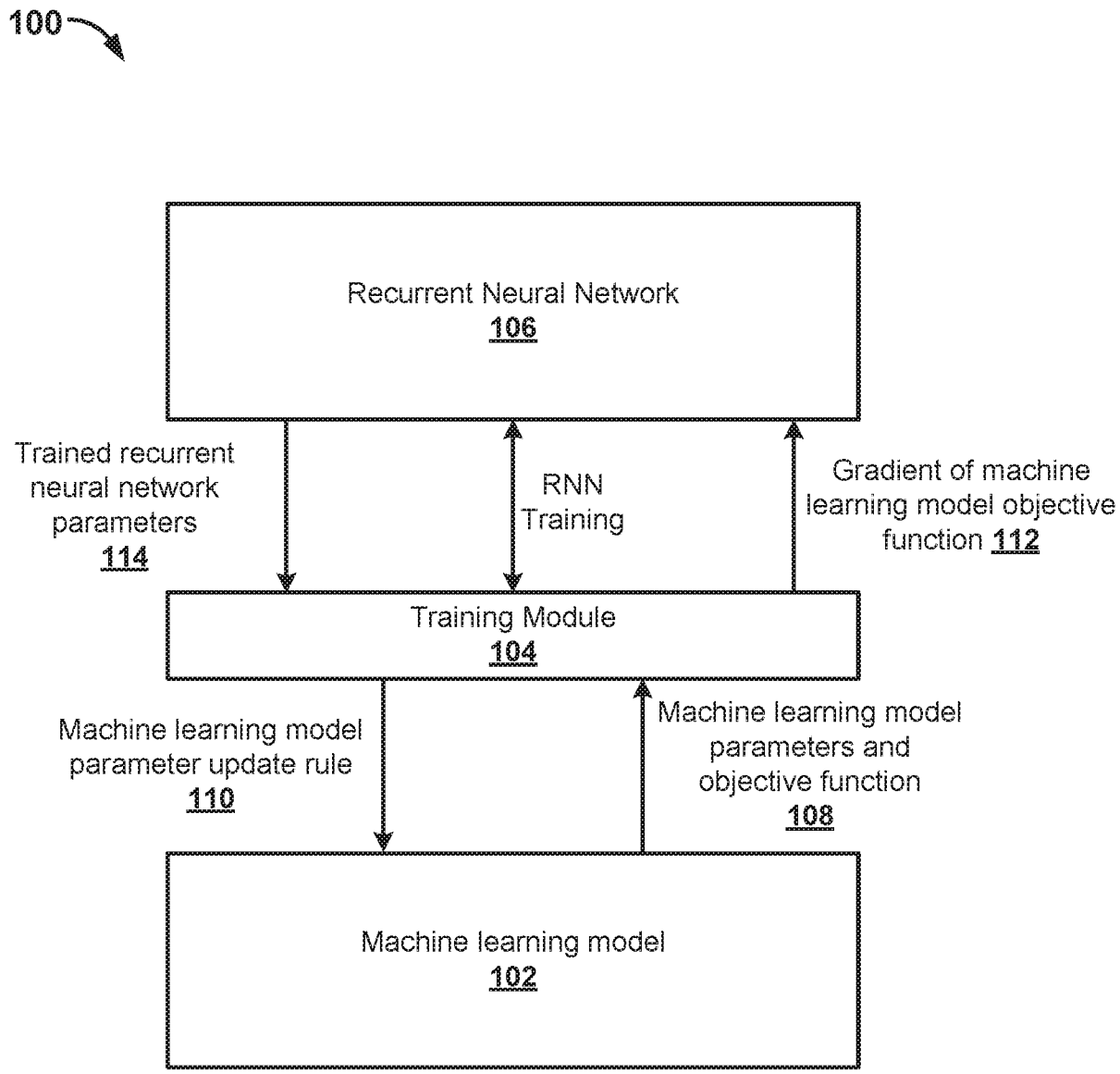
100
Recurrent Neural Network
106
Trained recurrent neural network parameters
114
RNN Training
Gradient of machine learning model objective function 112
Training Module
104
Machine learning model parameter update rule
110
Machine learning model parameters and objective function
108
Machine learning model
102

FIG. 1A

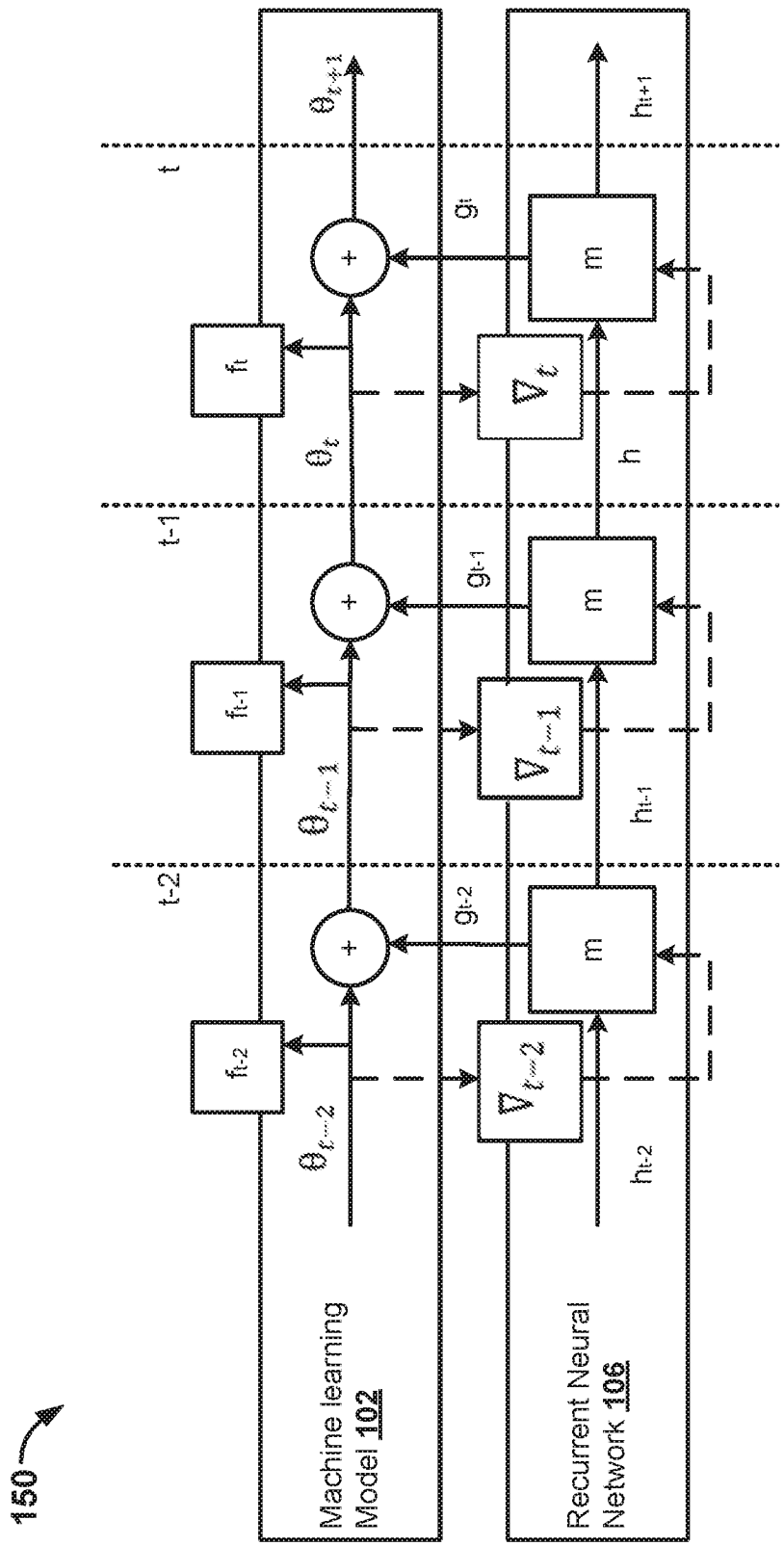
150
Machine learning Model 102
Recurrent Neural Network 106
t
t-1
t-2
ft
ft-1
ft-2
θt
θt-1
θt-2
θt+1
gt
gt-1
gt-2
ht+1
ht
ht-1
ht-2
m
+
∇t
∇t-1
∇t-2

FIG. 1B

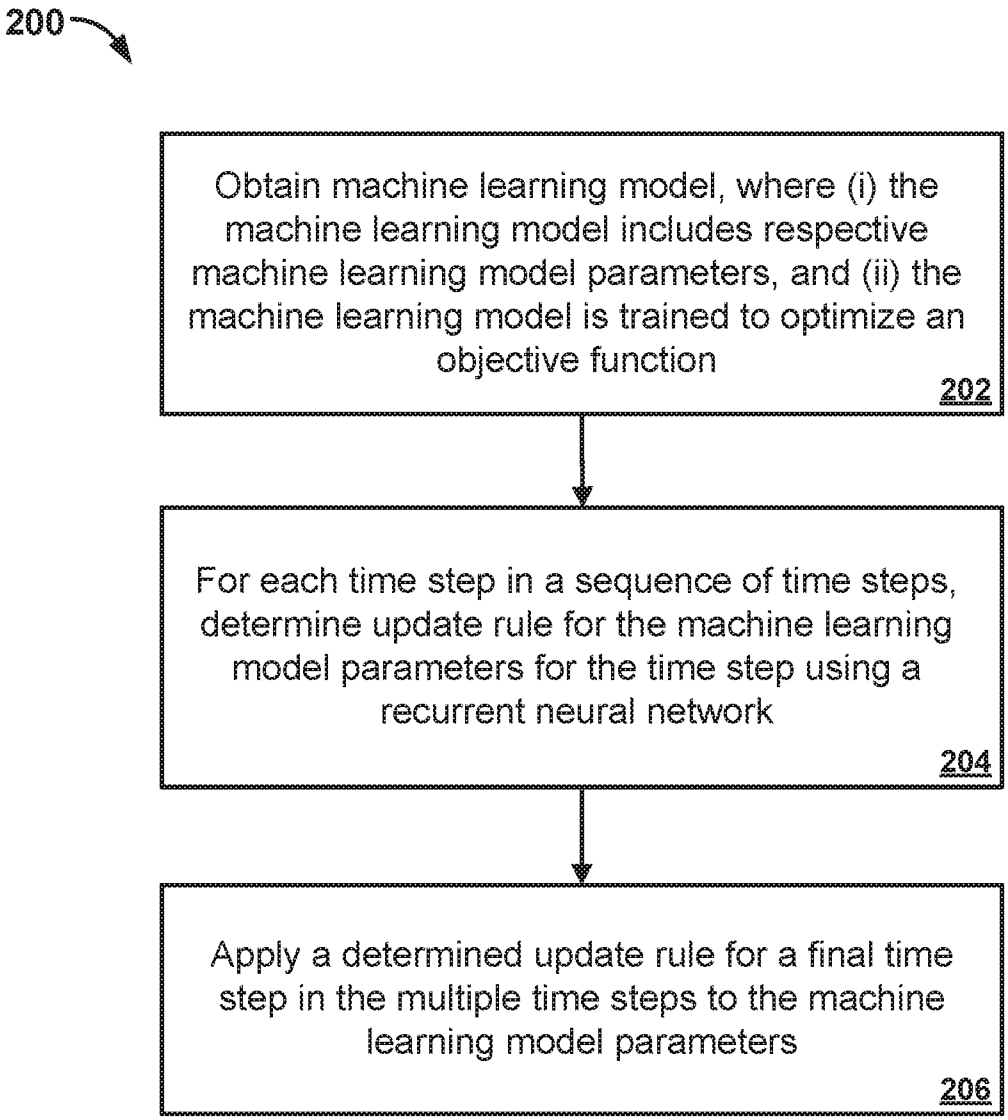
200
Obtain machine learning model, where (i) the machine learning model includes respective machine learning model parameters, and (ii) the machine learning model is trained to optimize an objective function
202
For each time step in a sequence of time steps, determine update rule for the machine learning model parameters for the time step using a recurrent neural network
204
Apply a determined update rule for a final time step in the multiple time steps to the machine learning model parameters
206

FIG. 2

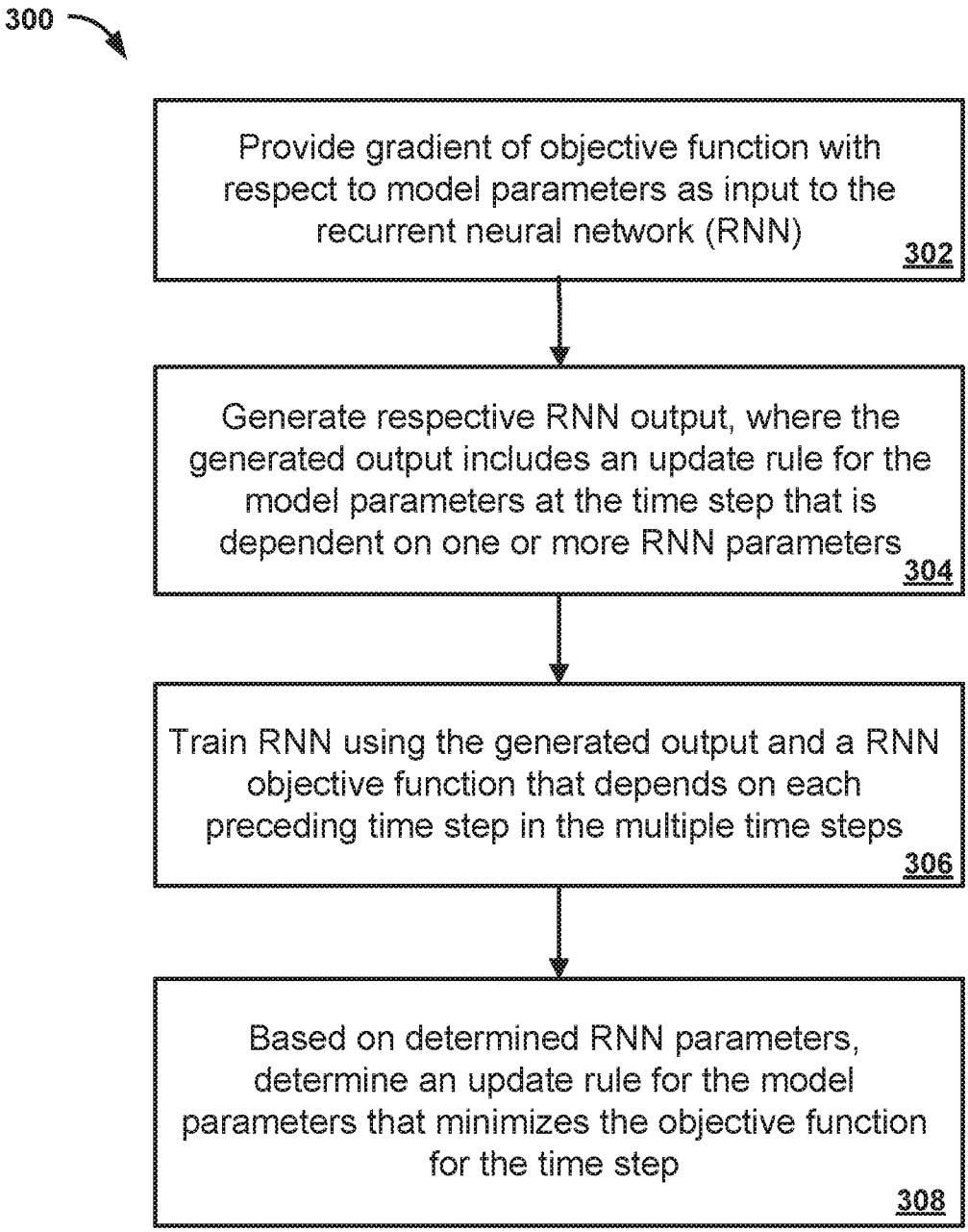
300
Provide gradient of objective function with respect to model parameters as input to the recurrent neural network (RNN)
302
Generate respective RNN output, where the generated output includes an update rule for the model parameters at the time step that is dependent on one or more RNN parameters
304
Train RNN using the generated output and a RNN objective function that depends on each preceding time step in the multiple time steps
306
Based on determined RNN parameters, determine an update rule for the model parameters that minimizes the objective function for the time step
308

FIG. 3

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2017/033703

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06N3/04 G06N3/08
ADD.

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)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Sepp Hochreiter ET AL: "Learning to Learn Using Gradient Descent", ICANN 2001, LNCS 2130, 25 August 2001 (2001-08-25), pages 87-94, XP055394688, Retrieved from the Internet: URL:http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=9D7D685FB075EEA3ABF7C81FDA6F0716?doi=10.1.1.5.323&rep=rep1&type=pdf [retrieved on 2017-07-28] the whole document ----- -/--	1-18

[X] 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
4 August 2017	16/08/2017

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Manfrin, Max

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2017/033703

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,P	MARCIN ANDRYCHOWICZ ET AL: "Learning to learn by gradient descent by gradient descent", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 14 June 2016 (2016-06-14), XP080708467, the whole document --- Jie Fu ET AL: "Deep Q-Networks for Accelerating the Training of Deep Neural Networks", , 5 June 2016 (2016-06-05), XP055396379, Retrieved from the Internet: URL:https://arxiv.org/pdf/1606.01467v1.pdf [retrieved on 2017-08-04] the whole document ---	1-18 1-18
X,P		