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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
09 May 2019 (09.05.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/089339 A1

(51) International Patent Classification:
G06N 3/02 (2006.01) *G06F 15/18* (2006.01)

(21) International Application Number:
PCT/US2018/057485

(22) International Filing Date:
25 October 2018 (25.10.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/580,525 02 November 2017 (02.11.2017) US

(71) Applicant: **THE TRUSTEES OF PRINCETON UNIVERSITY** [US/US]; 87 Prospect Avenue, Princeton, NJ 08544 (US).

(72) Inventors: **DAI, Xiaoliang**; 1 Lawrence Drive, Apt. 1108, Princeton, NJ 08540 (US). **YIN, Hongxu**; 751 Hibben Magie Road, Princeton, NJ 08544 (US). **JHA, Niraj, K.**; 20 Norbridge Dr., Princeton, NJ 08540 (US).

(74) Agent: **DRACHTMAN, Craig, M.**; Meagher Emanuel Laks Goldberg & Liao, One Palmer Square, Suite 325, Princeton, NJ 08542 (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, 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, 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).

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

(54) **Title:** METHOD AND SYSTEM FOR NEURAL NETWORK SYNTHESIS

(57) **Abstract:** According to various embodiments, a method for generating one or more optimal neural network architectures is disclosed. The method includes providing an initial seed neural network architecture and utilizing sequential phases to synthesize the neural network until a desired neural network architecture is reached. The phases include a gradient-based growth phase and a magnitude-based pruning phase.

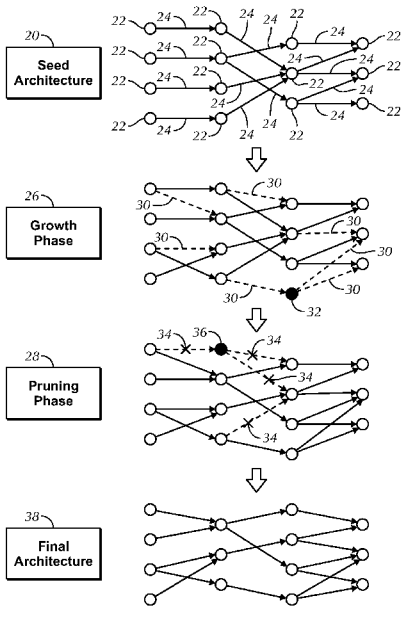


FIG. 2

WO 2019/089339 A1

METHOD AND SYSTEM FOR NEURAL NETWORK SYNTHESIS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to provisional application 62/580,525, filed on November 2, 2017, which is herein incorporated by reference in its entirety.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[0002] This invention was made with government support under Grant No. CNS-1617640 awarded by the National Science Foundation. The government has certain rights in the invention.

FIELD OF THE INVENTION

[0003] The present invention relates generally to neural networks and, more particularly, to a neural network synthesis tool based on a grow-and-prune paradigm.

BACKGROUND OF THE INVENTION

[0004] Over the last decade, deep neural networks (DNNs) have begun to revolutionize myriad research domains, such as computer vision, speech recognition, and machine translation. Their ability to distill intelligence from a dataset through multi-level abstraction can even lead to super-human performance. Thus, DNNs are emerging as a new cornerstone of modern artificial intelligence.

[0005] Though critically important, how to efficiently derive an appropriate DNN architecture from large datasets has remained an open problem. Researchers have traditionally derived the DNN architecture by sweeping through its architectural parameters and training the corresponding architecture until the point of diminishing returns in its performance. This suffers from three major problems. First, the widely used back-propagation (BP) algorithm assumes a fixed DNN architecture and only trains weights. Thus, training cannot improve the architecture. Second, a trial-and-error methodology can be inefficient when DNNs get deeper and contain millions of parameters. Third, simply going deeper and larger may lead to large, accurate, but over-parameterized DNNs.

[0006] Thus, there is a need for a method and system to efficiently determine an appropriate DNN architecture from large datasets.

SUMMARY OF THE INVENTION

[0007] According to various embodiments, a method for generating one or more optimal neural network architectures is disclosed. The method includes providing an initial seed neural network architecture and utilizing sequential phases to synthesize the neural network until a desired neural network architecture is reached. The phases include a gradient-based growth phase and a magnitude-based pruning phase.

[0008] According to various embodiments, a neural network synthesis tool (NeST) that automatically generates one or more optimal neural network architectures for a predetermined dataset is disclosed. The NeST includes one or more processes configured to provide an initial seed neural network architecture and utilize sequential phases to synthesize the neural network until a desired neural network architecture is reached. The phases include a gradient-based growth phase and a magnitude-based pruning phase.

[0009] According to various embodiments, a non-transitory computer-readable medium having stored thereon a computer program for execution by a processor configured to perform a method for generating one or more optimal neural network architectures is disclosed. The method includes providing an initial seed neural network architecture and utilizing sequential phases to synthesize the neural network until a desired neural network architecture is reached. The phases include a gradient-based growth phase and a magnitude-based pruning phase.

[0010] Various other features and advantages will be made apparent from the following detailed description and the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] In order for the advantages of the invention to be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments that are illustrated in the appended drawings. Understanding that these drawings depict only exemplary embodiments of the invention and are not, therefore, to be considered to be limiting its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings, in which:

[0012] Figure 1 is a block diagram of a system for NeST according to an embodiment of the present invention;

[0013] Figure 2 is a diagram of architecture synthesis flow in NeST according to an embodiment of the present invention;

[0014] Figure 3 is a chart of major components of the DNN architecture synthesis methodology in NeST according to an embodiment of the present invention;

[0015] Figure 4 is a diagram of grown connections from input layer to first layer of LeNet-300-100 according to an embodiment of the present invention;

[0016] Figure 5 is a methodology for neuron growth according to an embodiment of the present invention;

[0017] Figure 6 is a methodology for partial area convolution according to an embodiment of the present invention;

[0018] Figure 7 is a diagram of pruned connections and remaining connections in partial area convolution according to an embodiment of the present invention;

[0019] Figure 8 is a diagram of area-of-interests for five different kernels in first layer of LeNet-5 according to an embodiment of the present invention;

[0020] Figure 9 is a graph of growth time vs. post-growth DNN size trade-off for various seed architectures for LeNet-300-100 and LeNet-5 according to an embodiment of the present invention;

[0021] Figure 10 is a graph of compression ratio and final DNN size for different LeNet-300-100 and LeNet-5 seed architectures according to an embodiment of the present invention; and

[0022] Figure 11 is a graph of connections vs. synthesis iteration for LeNet-300-100 according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0023] Deep neural networks (DNNs) have begun to have a pervasive impact on various applications of machine learning. However, the problem of finding an optimal DNN architecture for large applications is challenging. Common approaches go for deeper and larger DNN architectures but may incur substantial redundancy. To address these problems, disclosed herein is a network growth method and system that complements network pruning to learn both weights and compact DNN architectures during training. A DNN synthesis tool (referred to herein as "NeST") is disclosed that automates the generation of compact and accurate DNNs. NeST starts with a randomly initialized sparse network called the seed architecture. It iteratively tunes the architecture with gradient-based growth and magnitude-based pruning of neurons and connections. Experimental results show that NeST yields accurate, yet very compact DNNs, with a wide range of seed architecture selection. For an LeNet-300-100 (LeNet-5) architecture, network parameters were reduced by 70.2× (74.3×)

and floating-point operations (FLOPs) by 79.4× (43.7×). For an AlexNet and VGG-16 architectures, network parameters (FLOPs) were reduced by 15.7× (4.6×) and 30.2× (8.6×), respectively. NeST's grow-and-prune paradigm delivers significant additional parameter and FLOPs reduction relative to pruning-only methods.

[0024] NeST is configured to train both DNN weights and architectures. NeST is inspired by the learning mechanism of the human brain, where the number of synaptic connections increases upon the birth of a baby, peaks after a few months, and decreases steadily thereafter. NeST starts DNN synthesis from a seed DNN architecture (birth point). It allows the DNN to grow connections and neurons based on gradient information (baby brain) so that the DNN can adapt to the problem at hand. Then, it prunes away insignificant connections and neurons based on magnitude information (adult brain) to avoid redundancy. A combination of network growth and pruning methods enables NeST to generate accurate and compact DNNs.

[0025] System Overview

[0026] Figure 1 illustrates a system 10 configured to implement NeST. The system 10 includes a device 12. The device 12 may be implemented in a variety of configurations including general computing devices such as but not limited to desktop computers, laptop computers, tablets, network appliances, and the like. The device 12 may also be implemented as a mobile device such as but not limited to a mobile phone, smart phone, smart watch, or tablet computer. The device 12 includes one or more processors 14 such as but not limited to a central processing unit (CPU), a graphics processing unit (GPU), or a field programmable gate array (FPGA) for performing specific functions and memory 16 for storing those functions. The processor 14 includes a NeST module 18 for training both DNN weights and architectures. The NeST module 18 methodology will be described in greater detail below.

[0027] Synthesis Methodology

[0028] According to various embodiments, NeST leverages both constructive and destructive structure adaptation (SA) approaches through a grow-and-prune paradigm. Unless otherwise stated, the notations given in Table 1 are adopted to represent various variables.

Table 1: Notations and descriptions

Label	Description	Label	Description
L	DNN loss function	$\mathbf{W}^l$	weights between $(l-1)^{th}$ and l^{th} layer
x_n^l	output value of n^{th} neuron in l^{th} layer	$\mathbf{b}^l$	biases in l^{th} layer
u_m^l	input value of m^{th} neuron in l^{th} layer	$\mathbb{R}^{d_0 \times d_1 \times d_2}$	d_0 by d_1 by d_2 matrix with real elements

[0029] The NeST approach is illustrated in Figure 2. Synthesis begins with an initial seed architecture 20, typically initialized as a sparse and partially connected DNN. It is also

ensured that all neurons 22 are connected via initial connections 24 in the seed architecture. Then, NeST utilizes two sequential phases to synthesize the DNN: (i) gradient-based growth phase 26, and (ii) magnitude-based pruning phase 28. In the growth phase 26, the gradient information in the architecture space is used to gradually grow new connections 30, neurons 32, and feature maps (not shown) to achieve the desired accuracy. In the pruning phase 28, the DNN inherits the synthesized architecture and weights from the growth phase and iteratively removes redundant connections 34 and neurons 36, based on their magnitudes. Finally, NeST comes to rest at a lightweight DNN model 38 that incurs no accuracy degradation relative to a fully connected model.

[0030] A flowchart illustrated in Figure 3 further shows a summary of the major components of the DNN architecture synthesis in NeST. The architecture synthesis includes gradient-based growth 26 and magnitude-based pruning 28. Gradient-based growth includes two parts: a fully connected layers aspect 40 involving connection growth and neuron growth, and a convolutional layers aspect 42 involving connection growth and feature map growth. Magnitude-based pruning 28 involves pruning of insignificant weights and partial area convolution. Each section will be described in greater detail below.

[0031] Gradient-based Growth 26

[0032] According to various embodiments, the methodologies in this section are used to grow connections 30, neurons 32, and feature maps (not shown).

[0033] The connection growth methodology greedily activates useful but currently dormant connections. This is incorporated into the learning policy: add a connection ω if it can quickly reduce the value of loss function L.

[0034] The DNN seed contains only a small fraction of active connections to propagate gradients. To locate the dormant connections that can reduce L effectively, $\partial L/\partial \omega$ is evaluated for all the dormant connections ω (computed either using the whole training set or a large batch). The above policy activates dormant connections if they the most efficient at reducing L. This can also assist with avoiding local minima and achieving higher accuracy. To illustrate this policy, the connections grown from the input to the first layer of LeNet-300-100 (for the MNIST dataset) are plotted in Figure 4. The image center has a much higher grown density than the margins, consistent with the fact that the MNIST digits are centered.

[0035] From a neuroscience perspective, the connection growth methodology coincides with the Hebbian theory: Neurons that fire together wire together. The stimulation magnitude of the m^{th} presynaptic neuron in the $(l+1)^{th}$ layer and the n^{th} postsynaptic neuron in the l^{th}

layer is defined as $\partial L/\partial u_m^{l+1}$ and x_n^l, respectively. The connections activated based on Hebbian theory would have a strong correlation between presynaptic and postsynaptic cells, thus a large value of $|(\partial L/\partial u_m^{l+1})x_n^l|$. This is also the magnitude of the gradient of L with respect to ω (ω is the weight that connects u_m^{l+1} and x_n^l:

[0036] $$|\partial L/\partial \omega| = |(\partial L/\partial u_m^{l+1})x_n^l| \quad (1)$$

[0037] The neuron growth methodology includes two steps: (i) connection establishment and (ii) weight initialization. The neuron growth policy is as follows:

[0038] In the l^{th} layer, add a new neuron as a shared intermediate node between existing neuron pairs that have high postsynaptic (x) and presynaptic ($\partial L/\partial u$) neuron correlations (each pair contains one neuron from the $(l-1)^{th}$ layer and the other from the $(l+1)^{th}$ layer). Initialize weights based on batch gradients to reduce the value of L.

[0039] The algorithm in Figure 5 incorporates this policy and illustrates the neuron growth iterations in detail. Before adding a neuron to the l^{th} layer, the bridging gradient is evaluated between the neurons at the previous $(l-1)^{th}$ and subsequent $(l+1)^{th}$ layers. The top $\beta \times 100\%$ (β is the growth ratio) correlated neuron pairs are connected through a new neuron in the l^{th} layer. The weights are initialized based on the bridging gradient to enable gradient descent, thus decreasing the value of L.

[0040] A square root rule is implemented for weight initialization to imitate a BP update on the bridging connection ω_b, which connects x_n^{l-1} and u_m^{l+1}. The BP update lead to a change in u_m^{l+1}:

[0041] $$|\Delta u_m^{l+1}|_{b.p.} = |x_n^{l-1} \times \delta\omega_b| = \eta\left|x_n^{l-1} \times G_{m,n}\right| \quad (2)$$

[0042] where η is the learning rate. In the algorithm in Figure 5, when a newly added neuron (in the l^{th} layer) is connected with x_n^{l-1} and u_m^{l+1}, their weights are initialized to the square root of the magnitude of the bridging gradient:

[0043] $$\left|\delta\omega_n^{in}\right| = |\delta\omega_m^{out}| = \sqrt{\left|G_{m,n}\right|} \quad (3)$$

[0044] where $\delta\omega_n^{in}(\delta\omega_m^{out})$ is the initialized value of the weight that connects the newly added neuron with $x_n^{l-1}(u_m^{l+1})$. The weight initialization rule leads to a change in u_m^{l+1}:

[0045] $$|\Delta u_m^{l+1}| = \left|f(x_n^{l-1} \times \delta\omega_n^{in}) \times \delta\omega_m^{out}\right| \quad (4)$$

[0046] where f is the neuron activation function. Suppose *tanh* is the activation function. Then:

[0047] $$f(x) = \tanh(x) \approx x, \text{ if } x \ll 1 \quad (5)$$

[0048] Since $\delta\omega_n^{in}$ and $\delta\omega_m^{out}$ are typically very small, the approximation in Eq. (5) leads to the following:

[0049] $$|\Delta u_m^{l+1}| \approx \left|x_n^{l-1} \times \delta\omega_n^{in} \times \delta\omega_m^{out}\right| = \frac{1}{\eta} \times |u_m^{l+1}|_{b.p.} \quad (6)$$

[0050] This is linearly proportional to the effect of a BP update. Thus, the weight initialization mathematically imitates a BP update. Though the algorithm was illustrated with the *tanh* activation function, the weight initialization rule works equally well with other activation functions, such as but not limited to rectified linear unit (ReLU) and leaky rectified linear unit (Leaky ReLU).

[0051] A birth strength factor α is used to strengthen the connections of a newly grown neuron. This prevents these connections from becoming too weak to survive the pruning phase. Specifically, after square root rule-based weight initialization, the newly added weights are scaled up by:

[0052] $$\boldsymbol{w}^{out} \leftarrow \alpha \boldsymbol{w}^{out} \times \frac{avg(abs(W^{l+1}))}{avg(abs(\boldsymbol{w}^{out}))}, \boldsymbol{w}^{in} \leftarrow \alpha \boldsymbol{w}^{in} \times \frac{avg(abs(W^{l}))}{avg(abs(\boldsymbol{w}^{in}))} \quad (7)$$

[0053] where *avg* is an operation that extracts the mean value of all non-zero elements. This strengthens new weights. In practice, $\alpha > 0.3$ was found to be an appropriate range.

[0054] Convolutional layers are related to the connection growth methodology. However, instead of neuron growth, a unique feature map growth methodology is used for convolutional layers. In a convolutional layer, input images are convolved with kernels to generate feature maps. Thus, to add a feature map, the corresponding set of kernels needs to be initialized. The feature map growth policy is summarized as follows: to add a new feature map to the convolutional layers, randomly generate sets of kernels, and pick the set of kernels that reduces L the most.

[0055] During experimentation, it is observed that the percentage reduction in L for the above policy is approximately twice as in the case of the naive approach that initializes the new kernels with random values.

[0056] Magnitude-based Pruning 28

[0057] According to various embodiments, insignificant connections and neurons are pruned away based on the magnitude of weights and outputs. The policy is summarized as follows: remove a connection (or neuron) if the magnitude of the weight (or neuron output) is smaller than a pre-defined threshold. Two variants of this policy are explained below: pruning of insignificant weights and partial-area convolution.

[0058] Magnitude-based pruning can successfully cut down memory and computational costs. This approach is extended to incorporate a batch normalization technique. Such a technique can reduce the internal covariate shift by normalizing layer inputs and improve training speed and behavior. Consider the l^{th} batch normalization layer:

[0059] $$\mathbf{u}^l = [(\mathbf{W}^l\mathbf{x}^{l-1} + \mathbf{b}^l) - \mathbf{E}] \oslash \mathbf{V} = \mathbf{W}_*^l\mathbf{x} + \mathbf{b}_*^l \quad (8)$$

[0060] where $\mathbf{E}$ and $\mathbf{V}$ are batch normalization terms, and $\oslash$ depicts the Hadamard (element-wise) division operator. Effective weights $\mathbf{W}_*^l$ and effective biases $\mathbf{b}_*^l$ are defined as:

[0061] $$\mathbf{W}_*^l = \mathbf{W}^l \oslash \mathbf{V}, \mathbf{b}_*^l = (\mathbf{b}^l - \mathbf{E}) \oslash \mathbf{V} \quad (9)$$

[0062] Connections with small effective weights are treated as insignificant. Pruning of insignificant weights is an iterative process. In each iteration, the most insignificant weights (e.g., top 1%) are only pruned for each layer, and then the whole DNN is retrained to recover its performance.

[0063] In common convolutional neural networks (CNNs), the convolutional layers typically consume about 5% of the parameters, but contribute to about 90-95% of the total FLOPs. In a convolutional layer, kernels shift and convolve with the entire input image. This process incurs redundancy, since not the whole input image is of interest to a particular kernel. One methodology is to prune all connections from a not-of-interest input image to a particular kernel. This method reduces FLOPs but incurs performance degradation. Instead of discarding an entire image, the proposed partial-area convolution methodology allows kernels to convolve with the image areas that are of interest. Such an area is referred to as *area-of-interest*. Connections are pruned to other image areas.

[0064] This process is illustrated in Figure 7. The cross shaded area in the center depicts *area-of-interest*, whereas the slant shaded area around the cross shaded area depicts parts that are not of interest. Thus, solid line connections are kept, whereas dashed line connections are pruned away.

[0065] Partial-area convolution pruning is an iterative process. One such iteration is presented in the algorithm in Figure 6. M input images are first convolved with $M \times N$ convolution kernels and $M \times N$ feature maps are generated, which are stored in a four-dimensional feature map matrix $\mathbf{C}$. The pruning threshold *thres* is set to the $(100\gamma)^{th}$ percentile of all elements in *abs*($\mathbf{C}$), where γ is the pruning ratio, typically but not limited to 1%. The elements whose values are below this threshold are marked as insignificant, and their input connections are pruned away. The whole DNN is retrained after each pruning iteration. A mask **Msk** may be utilized to disregard the pruned convolution area.

[0066] Partial-area convolution enables additional FLOPs reduction without any performance degradation. For example, FLOPs in LeNet-5 can be reduced by 2.09× when applied to MNIST. Compared to the conventional CNNs that force a fixed square-shaped *area-of-interest* on all kernels, each kernel is allowed to self-explore the preferred shape of its *area-of-interest*. Figure 8 shows the *area-of-interest* found by the layer-1 kernels in LeNet-5 when applied to MNIST. Significant overlaps are observed in the image central area, which most kernels are interested in.

[0067] Evaluation of an Exemplary Embodiment of NeST

[0068] NeST was implemented using Tensorflow and PyTorch on Nvidia GTX 1060 and Tesla P100 GPUs. NeST was used to synthesize compact DNNs for the MNIST and ImageNet datasets. DNN seed architectures were selected based on clues (e.g., depth, kernel size, etc.) from the existing LeNets, AlexNet, and VGG-16 architectures, respectively. From this implementation, NeST exhibited two major advantages.

[0069] (1) Wide seed range: NeST yields high-performance DNNs with a wide range of seed architectures. Its ability to start from a wide range of seed architectures alleviates reliance on human-defined architectures, and offers more freedom to DNN designers.

[0070] (2) Drastic redundancy removal: NeST-generated DNNs are very compact. Compared to the DNN architectures generated with pruning-only methods, DNNs generated through the grow-and-prune paradigm have much fewer parameters and require much fewer FLOPs.

[0071] For LeNets on MNIST, the seed architectures were derived from the original LeNet-300-100 and LeNet-5 networks. LeNet-300-100 is a multi-layer perceptron with two hidden layers. LeNet-5 is a CNN with two convolutional layers and three fully connected layers. The affine-distorted MNIST dataset is used, on which LeNet-300-100 (LeNet-5) can achieve an error rate of 1.3% (0.8%).

[0072] For the growth phase, first, nine (four) seed architectures were derived for LeNet-300-100 (LeNet-5). These seeds contain fewer neurons and connections per layer than the original LeNets. The number of neurons in each layer is the product of a ratio r and the corresponding number in the original LeNets (e.g., the seed architecture for LeNet-300-100 becomes LeNet-120-40 if $r = 0.4$). Only 10% of all possible connections in the seed architecture are randomly initialized. Also, it is ensured that all neurons in the network are connected.

[0073] First, r was swept for LeNet-300-100 (LeNet-5) from 0.2 (0.5) to 1.0 (1.0) with a step-size of 0.1 (0.17), and then the DNN architectures were grown from these seeds. The

impact of these seeds was studied on the GPU time for growth and post-growth DNN sizes under the same target accuracy (this accuracy is typically a reference value for the architecture). The results for LeNets are summarized in Figure 9. There were two interesting findings for the growth phase.

[0074] (1) Smaller seed architectures often lead to smaller post-growth DNN sizes, but at the expense of a higher growth time. Smaller seeds and thus smaller post-growth DNN sizes are better, since they also lead to smaller final DNN sizes.

[0075] (2) When the post-growth DNN size saturates due to the full exploitation of the synthesis freedom for a target accuracy, a smaller seed is no longer beneficial, as evident from the flat left ends of the dashed curves in Figure 9.

[0076] For the pruning phase, next, the post-growth LeNet DNNs are pruned to remove their redundant neurons/connections. It is shown the post-pruning DNN sizes and compression ratios for LeNet-300-100 and LeNet-5 for the different seeds in Figure 10. There were two major observations for the pruning phase.

[0077] (1) The larger the pre-pruning DNN, the larger is its compression ratio. This is because larger pre-pruning DNNs have a larger number of weights and thus also higher redundancy.

[0078] (2) The larger the pre-pruning DNN, the larger is its post-pruning DNN. Thus, to synthesize a more compact DNN, one should choose a smaller seed architecture within an appropriate range.

[0079] The results are compared against related results in the below Table 2. The results for the embodiment disclosed herein outperform other reference models from various design perspectives. Without any loss of accuracy, the number of connections and FLOPs were reduced of LeNet-300-100 (LeNet-5) by 70.2× (74.3×) and 79.4× (43.7×), respectively, relative to the baseline Caffe model.

Table 2: Different inference models for MNIST

Model	Method	Error	#Param	FLOPs
RBF network [7]	-	3.60%	794K	1588K
Polynomial classifier [7]	-	3.30%	40K	78K
K-nearest neighbors [7]	-	3.09%	47M	94M
SVMs (reduced set) [35]	-	1.10%	650K	1300K
Caffe model (LeNet-300-100) [36]	-	1.60%	266K	532K
LWS (LeNet-300-100) [22]	Prune	1.96%	4K	8K
Net pruning (LeNet-300-100) [5]	Prune	1.59%	22K	43K
Our LeNet-300-100: compact	**Grow+Prune**	**1.58%**	**3.8K**	**6.7K**
Our LeNet-300-100: accurate	**Grow+Prune**	**1.29%**	**7.8K**	**14.9K**
Caffe model (LeNet-5) [36]	-	0.80%	431K	4586K
LWS (LeNet-5) [22]	Prune	1.66%	4K	199K
Net pruning (LeNet-5) [5]	Prune	0.77%	35K	734K
Our LeNet-5	**Grow+Prune**	**0.77%**	**5.8K**	**105K**

[0080] For the AlexNet and VGG-16 on ImageNet, NeST is used to synthesize DNNs for the ILSVRC 2012 image classification dataset. A slim and sparse seed architecture is initialized base on the AlexNet and VGG-16. For this embodiment, the seed architecture for AlexNet contains only 60, 140, 240, 210, and 160 feature maps in the five convolutional layers, and 3200, 1600, and 1000 neurons in the fully connected layers. The seed architecture for VGG-16 uses $r = 0.75$ for the first 13 convolutional layers, and has 3200, 1600, and 1000 neurons in the fully connected layers. 30% of all the possible connections are randomly activated for both seed architectures.

[0081] Table 3 below compares the model synthesized by NeST with various AlexNet and VGG-16 based inference models. The baselines here are the AlexNet Caffe model (42.78% top-1 and 19.73% top-5 error rate) and VGG-16 PyTorch model (28.41% top-1 and 9.62% top-5 error rate). The grow-and-prune synthesis paradigm disclosed herein outperforms the pruning-only methods listed in Table 3. This may be explained by the observation that pruning methods potentially inherit a certain amount of redundancy associated with the original large DNNs. Network growth can alleviate this phenomenon.

[0082] Note that the current mask-based implementation of growth and pruning incurs a temporal memory overhead during training. If the model becomes deeper, as in the case of ResNet or DenseNet, using masks to grow and prune connections, neurons, and/or feature maps may not be economical due to this temporal training memory overhead.

Table 3: Different AlexNet and VGG-16 based inference models for ImageNet

Model	Method	ΔTop-1 err.	ΔTop-5 err.	#Param (M)	FLOPs (B)
Baseline AlexNet [37]	-	0.0%	0.0%	61 (1.0×)	1.5 (1.0×)
Data-free pruning [38]	Prune	+1.62%	-	39.6 (1.5×)	1.0 (1.5×)
Fastfood-16-AD [39]	-	+0.12%	-	16.4 (3.7×)	1.4 (1.1×)
Memory-bounded [40]	-	+1.62%	-	15.2 (4.0×)	-
SVD [41]	-	+1.24%	+0.83%	11.9 (5.1×)	-
LWS (AlexNet) [22]	Prune	+0.35%	+0.28%	6.7 (9.1×)	0.5 (3.0×)
Net pruning (AlexNet) [5]	Prune	-0.01%	-0.06%	6.7 (9.1×)	0.5 (3.0×)
Our AlexNet	Grow+Prune	-0.02%	-0.06%	3.9 (15.7×)	0.33 (4.6×)
Baseline VGG-16 [42]	-	0.0%	0.0%	138 (1.0×)	30.9 (1.0×)
LWS (VGG-16) [22]	Prune	+3.61%	+1.35%	10.3 (13.3×)	6.5 (4.8×)
Net pruning (VGG-16) [5]	Prune	+2.93%	+1.26%	10.3 (13.3×)	6.5 (4.8×)
Our VGG-16: accurate	Grow+Prune	-0.35%	-0.31%	9.9 (13.9×)	6.3 (4.9×)*
Our VGG-16: compact	Grow+Prune	+2.31%	+0.98%	4.6 (30.2×)	3.6 (8.6×)*

* Currently without partial-area convolution due to GPU memory limits.

[0083] Conclusion

[0084] The synthesis methodology disclosed herein incorporates three inspirations from the human brain.

[0085] First, the number of synaptic connections in a human brain varies at different human ages. It rapidly increases upon the baby's birth, peaks after a few months, and decreases steadily thereafter. A DNN experiences a very similar learning process in NeST, as shown in Figure 11. This curve shares a very similar pattern to the evolution of the number of synapses in the human brain.

[0086] Second, most learning processes in a human brain result from rewiring of synapses between neurons. The human brain grows and prunes away a large amount (up to 40%) of synaptic connections every day. NeST wakes up new connections, thus effectively rewiring more neurons pairs in the learning process. Thus, it mimics the "learning through rewiring" mechanism of human brains.

[0087] Third, only a small fraction of neurons are active at any given time in human brains. This mechanism enables the human brain to operate at an ultra-low power (20Watts). However, fully connected DNNs contain a substantial amount of insignificant neuron responses per inference. To address this problem, a magnitude-based pruning methodology is included in NeST to remove the redundancy, thus achieving sparsity and compactness. This leads to huge storage and computation reductions.

[0088] As such disclosed herein is a synthesis tool, NeST, to synthesize compact yet accurate DNNs. NeST starts from a sparse seed architecture, adaptively adjusts the architecture through gradient-based growth and magnitude-based pruning, and finally arrives at a compact DNN with high accuracy. For LeNet-300-100 (LeNet-5) on MNIST, the number of network parameters were reduced by 70.2× (74.3×) and FLOPs by 79.4× (43.7×). For

Amendments to the Specification:

Please replace Table 3 with the following table such that the characters are greater than 0.28 cm high in capital letters:

Table 3: Different AlexNet and VGG-16 based inference models for ImageNet

Model	Method	ΔTop-1 err.	ΔTop-5 err.	#Param (M)	FLOPs (B)
Baseline AlexNet [37]	-	0.0%	0.0%	61 (1.0x)	1.5 (1.0x)
Data-free pruning [38]	Prune	+1.62%	-	39.6 (1.5x)	1.0 (1.5x)
Fastfood-16-AD [39]	-	+0.12%	-	16.4 (3.7x)	1.4 (1.1x)
Memory-bounded [40]	-	+1.62%	-	15.2 (4.0x)	-
SVD [41]	-	+1.24%	+0.83%	11.9 (5.1x)	-
LWS (AlexNet) [22]	Prune	+0.33%	+0.28%	6.7 (9.1x)	0.5 (3.0x)
Net pruning (AlexNet) [5]	Prune	-0.01%	-0.06%	6.7 (9.1x)	0.5 (3.0x)
Our AlexNet	Grow+Prune	**-0.02%**	**-0.06%**	3.9 (15.7x)	**0.33 (4.6x)**
Baseline VGG-16 [42]	-	0.0%	0.0%	138 (1.0x)	30.9 (1.0x)
LWS (VGG-16) [22]	Prune	+3.61%	+1.35%	10.3 (13.3x)	6.5 (4.8x)
Net pruning (VGG-16) [5]	Prune	+2.93%	+1.26%	10.3 (13.3x)	6.5 (4.8x)
Our VGG-16: accurate	Grow+Prune	**-0.35%**	**-0.31%**	9.9 (13.9x)	6.3 (4.9x)*
Our VGG-16: compact	Grow+Prune	+2.31%	**+0.98%**	4.6 (30.2x)	3.6 (8.6x)*

* Currently without partial-area convolution due to GPU memory limits.

^12. 2 .

AlexNet and VGG-16 on ImageNet, the network parameters (FLOPs) were reduced by 15.7× (4.6×) and 30.2× (8.6×), respectively.

[0089] It is understood that the above-described embodiments are only illustrative of the application of the principles of the present invention. The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. All changes that come within the meaning and range of equivalency of the claims are to be embraced within their scope. Thus, while the present invention has been fully described above with particularity and detail in connection with what is presently deemed to be the most practical and preferred embodiment of the invention, it will be apparent to those of ordinary skill in the art that numerous modifications may be made without departing from the principles and concepts of the invention as set forth in the claims.

CLAIMS

What is claimed is:

1. A method for generating one or more optimal neural network architectures, comprising:

providing an initial seed neural network architecture; and

utilizing sequential phases to synthesize the neural network until a desired neural network architecture is reached, the phases comprising a gradient-based growth phase and a magnitude-based pruning phase.

2. The method of claim 1, wherein the gradient-based growth phase and magnitude-based pruning phase are iterative.

3. The method of claim 1, wherein the gradient-based growth phase comprises a fully connected layers aspect comprising connection growth and neuron growth.

4. The method of claim 3, wherein the connection growth aspect comprises a policy to add a connection if the connection can reduce a value of a loss function.

5. The method of claim 3, wherein the neuron growth aspect comprises a policy to add a new neuron as a shared intermediate node between existing neuron pairs that have high postsynaptic and presynaptic neuron correlations and to initialize weights based on batch gradients to reduce a value of a loss function.

6. The method of claim 1, wherein the gradient-based growth phase comprises a convolutional layers aspect comprising connection growth and feature map growth.

7. The method of claim 6, wherein the feature map growth aspect comprises a policy to add a new feature map to convolutional layers, randomly generate sets of kernels, and pick the set of kernels that reduces a loss function the most.

8. The method of claim 1, wherein the magnitude-based pruning phase comprises pruning of insignificant weights and partial-area convolution.

9. The method of claim 8, wherein the pruning of insignificant weights and partial-area

convolution comprises a policy to remove a connection or neuron if a magnitude of a weight or neuron output is smaller than a predefined threshold.

10. A neural network synthesis tool (NeST) that automatically generates one or more optimal neural network architectures for a predetermined dataset, the NeST comprising one or more processes configured to:

provide an initial seed neural network architecture; and

utilize sequential phases to synthesize the neural network until a desired neural network architecture is reached, the phases comprising a gradient-based growth phase and a magnitude-based pruning phase.

11. The NeST of claim 10, wherein the gradient-based growth phase and magnitude-based pruning phase are iterative.

12. The NeST of claim 10, wherein the gradient-based growth phase comprises a fully connected layers aspect comprising connection growth and neuron growth.

13. The NeST of claim 12, wherein the connection growth aspect comprises a policy to add a connection if the connection can reduce a value of a loss function.

14. The NeST of claim 12, wherein the neuron growth aspect comprises a policy to add a new neuron as a shared intermediate node between existing neuron pairs that have high postsynaptic and presynaptic neuron correlations and to initialize weights based on batch gradients to reduce a value of a loss function.

15. The NeST of claim 10, wherein the gradient-based growth phase comprises a convolutional layers aspect comprising connection growth and feature map growth.

16. The NeST of claim 15, wherein the feature map growth aspect comprises a policy to add a new feature map to convolutional layers, randomly generate sets of kernels, and pick the set of kernels that reduces a loss function the most.

17. The NeST of claim 10, wherein the magnitude-based pruning phase comprises pruning of insignificant weights and partial-area convolution.

18. The NeST of claim 17, wherein the pruning of insignificant weights and partial-area convolution comprises a policy to remove a connection or neuron if a magnitude of a weight or neuron output is smaller than a predefined threshold.

19. A non-transitory computer-readable medium having stored thereon a computer program for execution by a processor configured to perform a method for generating one or more optimal neural network architectures, the method comprising:

providing an initial seed neural network architecture; and

utilizing sequential phases to synthesize the neural network until a desired neural network architecture is reached, the phases comprising a gradient-based growth phase and a magnitude-based pruning phase.

20. The computer-readable medium of claim 19, wherein the gradient-based growth phase and magnitude-based pruning phase are iterative.

21. The computer-readable medium of claim 19, wherein the gradient-based growth phase comprises a fully connected layers aspect comprising connection growth and neuron growth.

22. The computer-readable medium of claim 21, wherein the connection growth aspect comprises a policy to add a connection if the connection can reduce a value of a loss function.

23. The computer-readable medium of claim 21, wherein the neuron growth aspect comprises a policy to add a new neuron as a shared intermediate node between existing neuron pairs that have high postsynaptic and presynaptic neuron correlations and to initialize weights based on batch gradients to reduce a value of a loss function.

24. The computer-readable medium of claim 19, wherein the gradient-based growth phase comprises a convolutional layers aspect comprising connection growth and feature map growth.

25. The computer-readable medium of claim 24, wherein the feature map growth aspect comprises a policy to add a new feature map to convolutional layers, randomly

generate sets of kernels, and pick the set of kernels that reduces a loss function the most.

26. The computer-readable medium of claim 19, wherein the magnitude-based pruning phase comprises pruning of insignificant weights and partial-area convolution.

27. The computer-readable medium of claim 26, wherein the pruning of insignificant weights and partial-area convolution comprises a policy to remove a connection or neuron if a magnitude of a weight or neuron output is smaller than a predefined threshold.

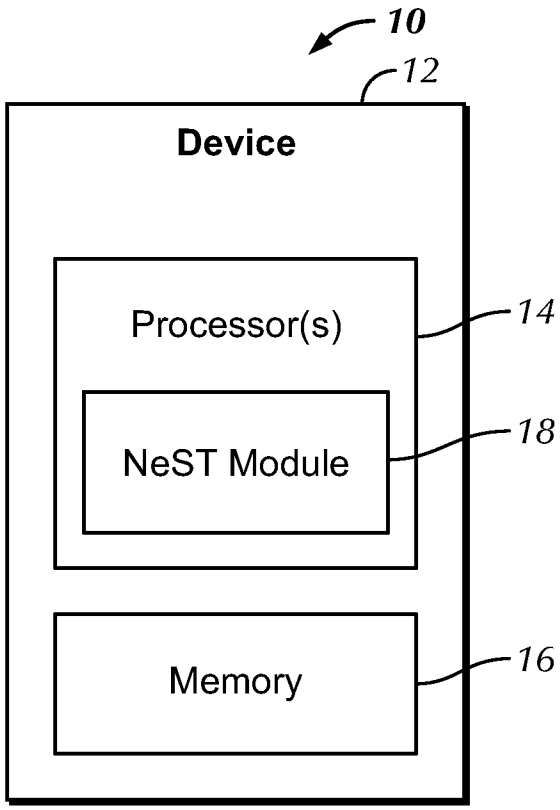


FIG. 1

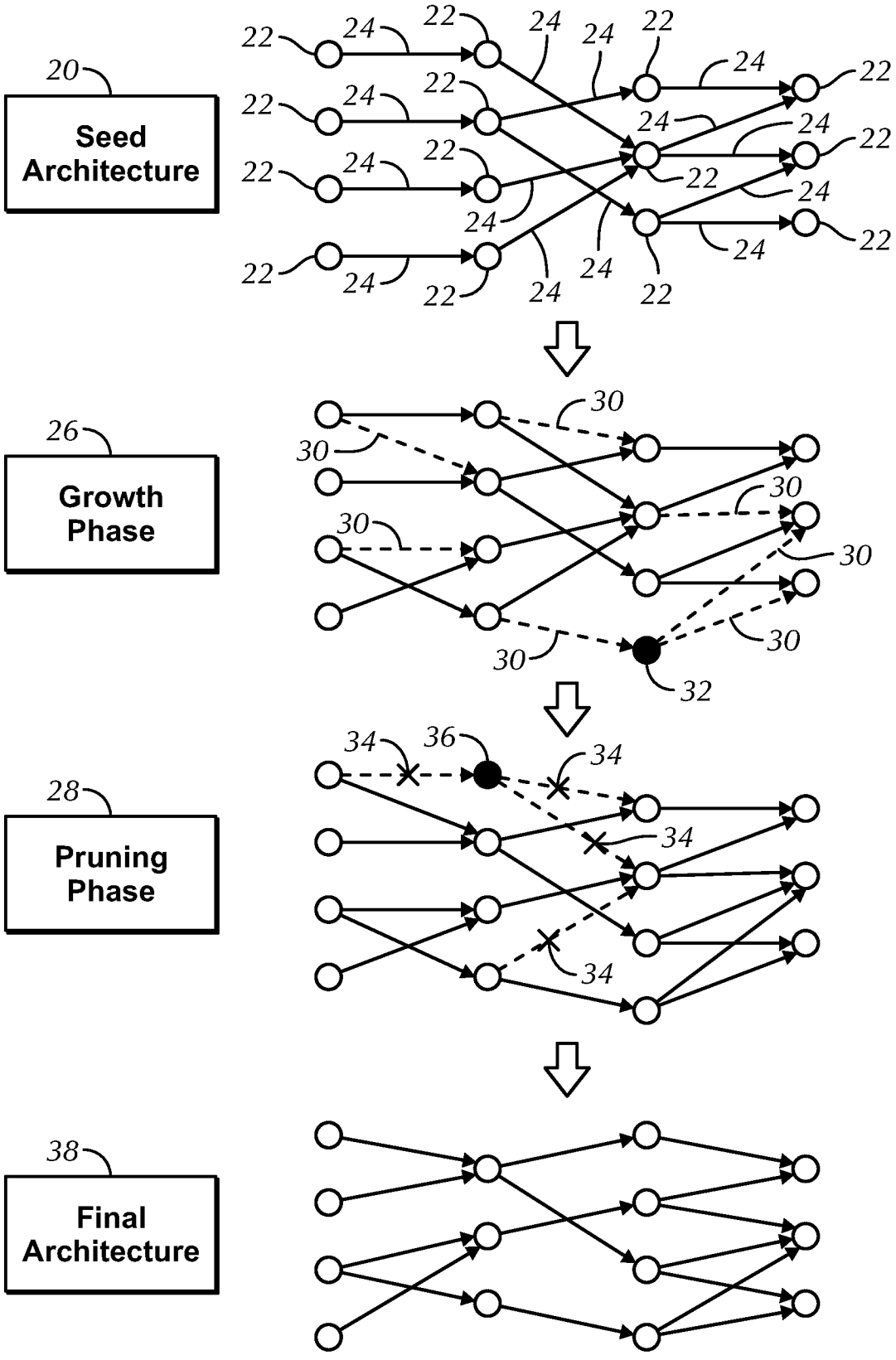

20
Seed Architecture
26
Growth Phase
28
Pruning Phase
38
Final Architecture
22
24
30
32
34
36

FIG. 2

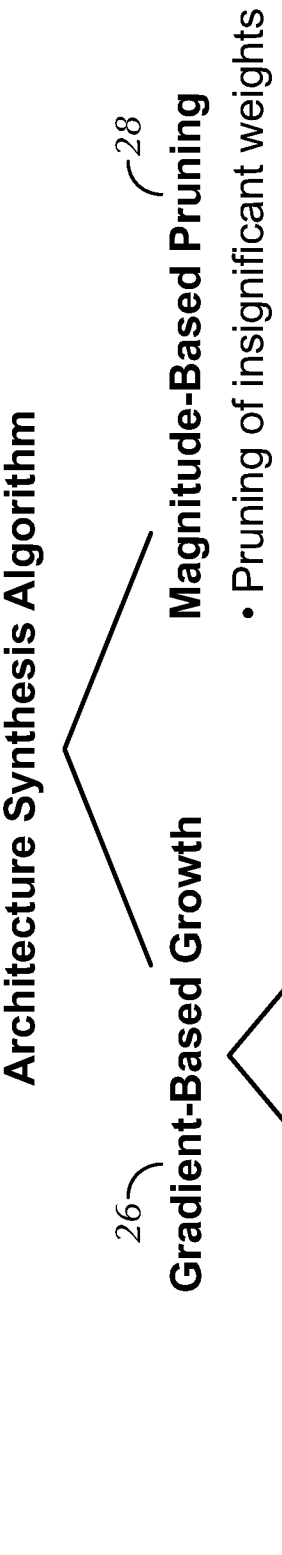

Architecture Synthesis Algorithm
26
Gradient-Based Growth
28
Magnitude-Based Pruning
• Pruning of insignificant weights
• Partial-area convolution
42
Convolutional Layers
• Connection growth
• Feature map growth
40
Fully Connected Layers
• Connection growth
• Neuron growth

FIG. 3

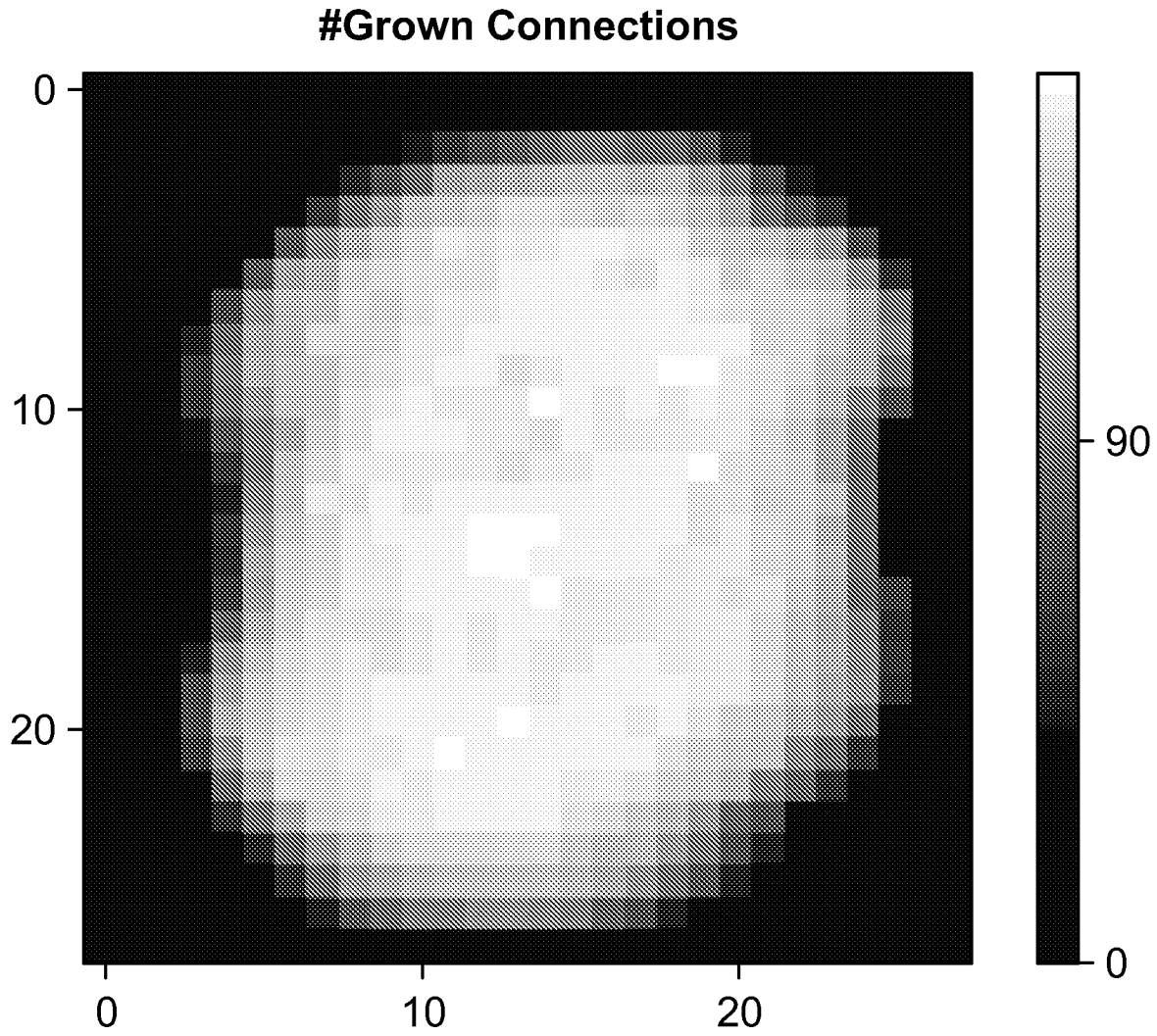


FIG. 4

Algorithm 1 Neuron growth in the l^{th} layer

Input: α - birth strength, β - growth ratio
Denote: M - number of neurons in the $(l+1)^{th}$ layer, N - number of neurons in the $(l-1)^{th}$ layer, $\mathbf{G} \in \boldsymbol{R}^{M \times N}$ - bridging gradient matrix, *avg* - extracts mean value of non-zero elements
Add a neuron in the l^{th} layer, initialize $\mathbf{w}^{out} = \vec{\mathbf{0}} \in \boldsymbol{R}^{M}$, $\mathbf{w}^{in} = \vec{\mathbf{0}} \in \boldsymbol{R}^{N}$
for $1 \le m \le M$, $1 \le n < N$ **do**

$$G_{m,n} = \frac{\partial L}{\partial u_m^{l+1}} \times x_n^{l-1}$$

end for
$thres = (\beta MN)^{th}$ largest element in abs (**G**)
for $1 \le m \le M$, $1 \le n \le N$ **do**
if $|G_{m,n}| > thres$ **then**

$$\delta w = \sqrt{|G_{m,n}|} \times rand\{1,-1\}$$

$$w_m^{out} \leftarrow w_m^{out} + \delta w, w_n^{in} \leftarrow w_n^{in} + \delta w, \times sgn(G_{m,n})$$

end if

$$\boldsymbol{w}^{out} \leftarrow \boldsymbol{w}^{out} \times \alpha \frac{avg(abs(\mathbf{w}^{l+1}))}{avg(abs(\mathbf{w}^{out}))}, \boldsymbol{w}^{in} \leftarrow \boldsymbol{w}^{in} \times \alpha \frac{avg(abs(\mathbf{w}^{l}))}{avg(abs(\mathbf{w}^{in}))}$$

end for
Concatenate network weights **W** with $\boldsymbol{w}^{in}, \boldsymbol{w}^{out}$

FIG. 5

Algorithm 2 Partial-area convolution

Input: I - M input images, **K** - kernel matrix, **Msk** - feature map mask, γ - pruning ratio
Output: Msk, F - N feature maps
Denote: $\mathbf{C} \in \boldsymbol{R}^{M \times N \times P \times Q}$ - Depthwise feature map, $\otimes$ - Hadamark (element-wise) multiplication
for $1 \le m \le M$, $1 \le n < N$ **do**
  $C_{m,n} = convolve(\mathbf{I}_m, \mathbf{K}_{m,n})$
end for
$thres = (\gamma MNPQ)^{th}$ largest element in abs (**C**)
for $1 \le m \le M$, $1 \le n \le N$, $1 \le p \le P$, $1 \le q \le Q$ **do**
  if $|C_{m,n,p,q}| < thres$ **then**
    $M sk_{m,n,p,q} = 0$
  end if
end for
$\mathbf{C} \leftarrow \mathbf{C} \otimes \mathbf{Msk}$, $\mathbf{F} \leftarrow \sum_{m=1}^{M} \mathbf{C}_m$

FIG. 6

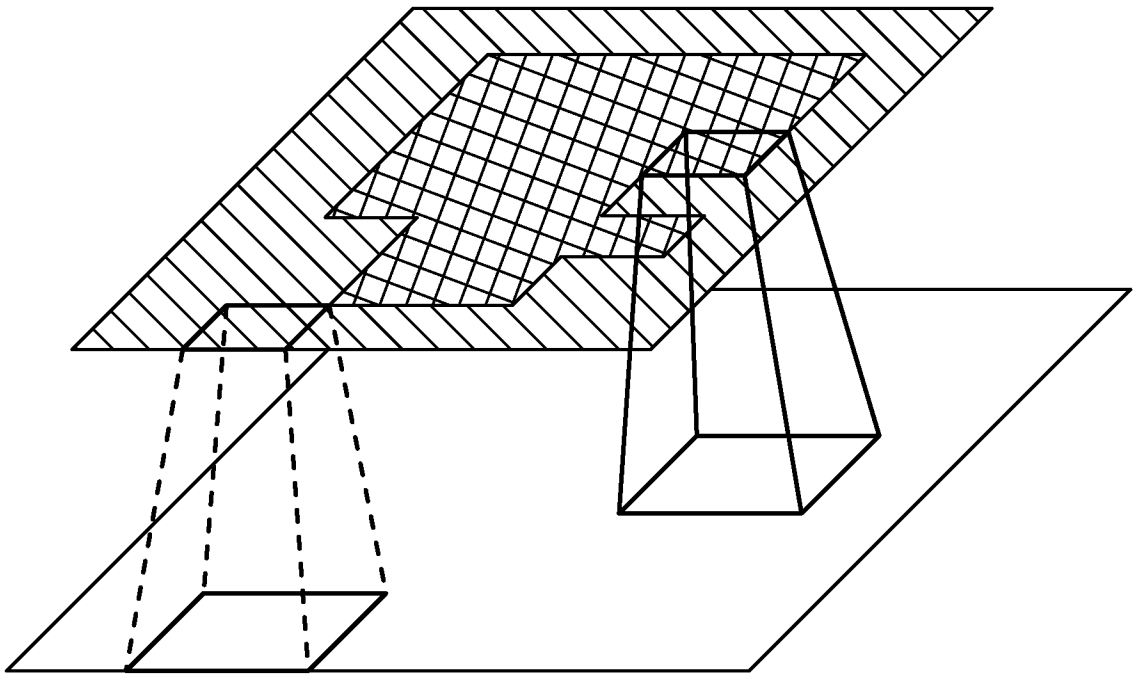


FIG. 7

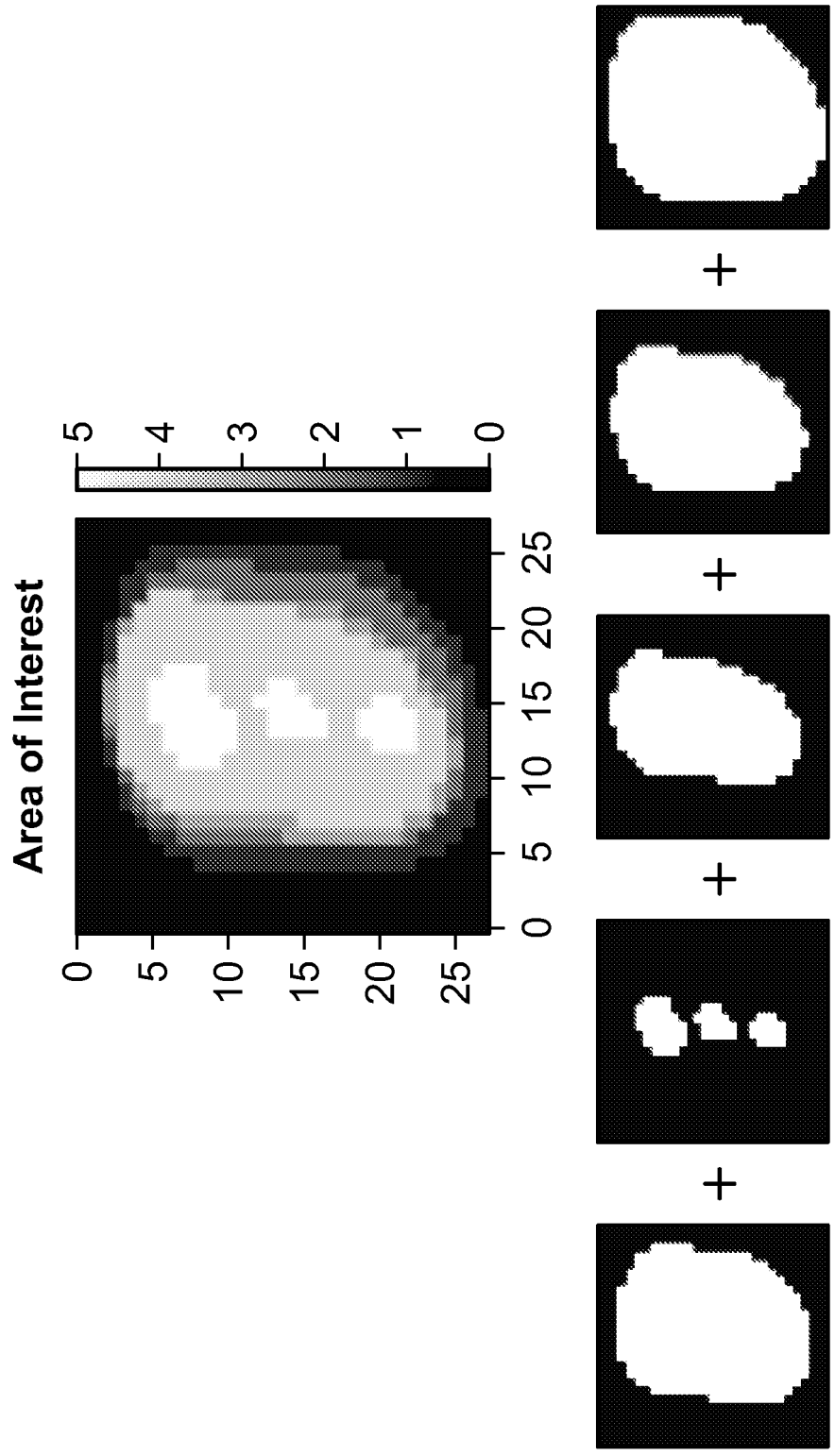


FIG. 8

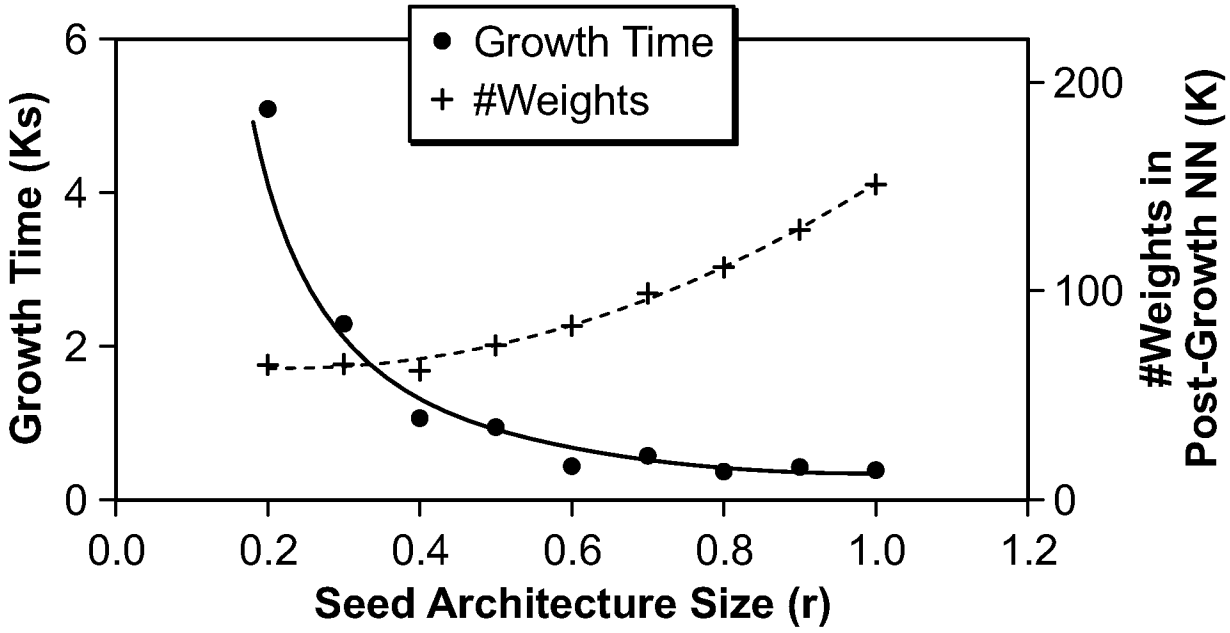

Growth Time
#Weights
Growth Time (Ks)
#Weights in Post-Growth NN (K)
Seed Architecture Size (r)
0
2
4
6
0
100
200
0.0
0.2
0.4
0.6
0.8
1.0
1.2

FIG. 9A

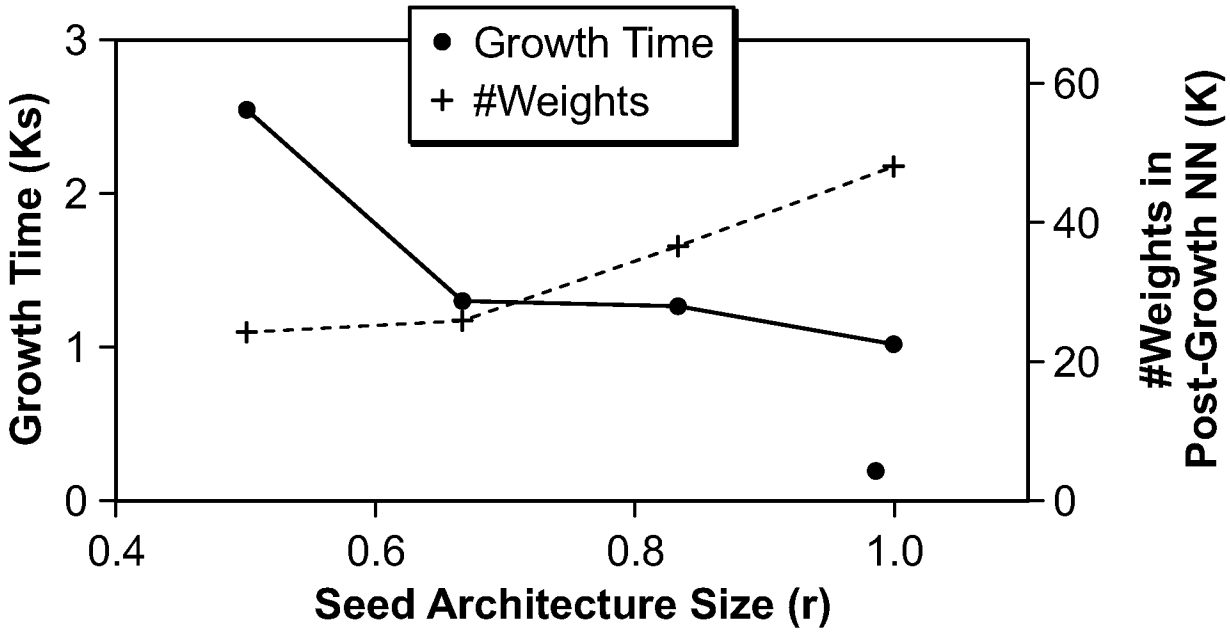

Growth Time
#Weights
Growth Time (Ks)
#Weights in Post-Growth NN (K)
Seed Architecture Size (r)
0
1
2
3
0
20
40
60
0.4
0.6
0.8
1.0

FIG. 9B

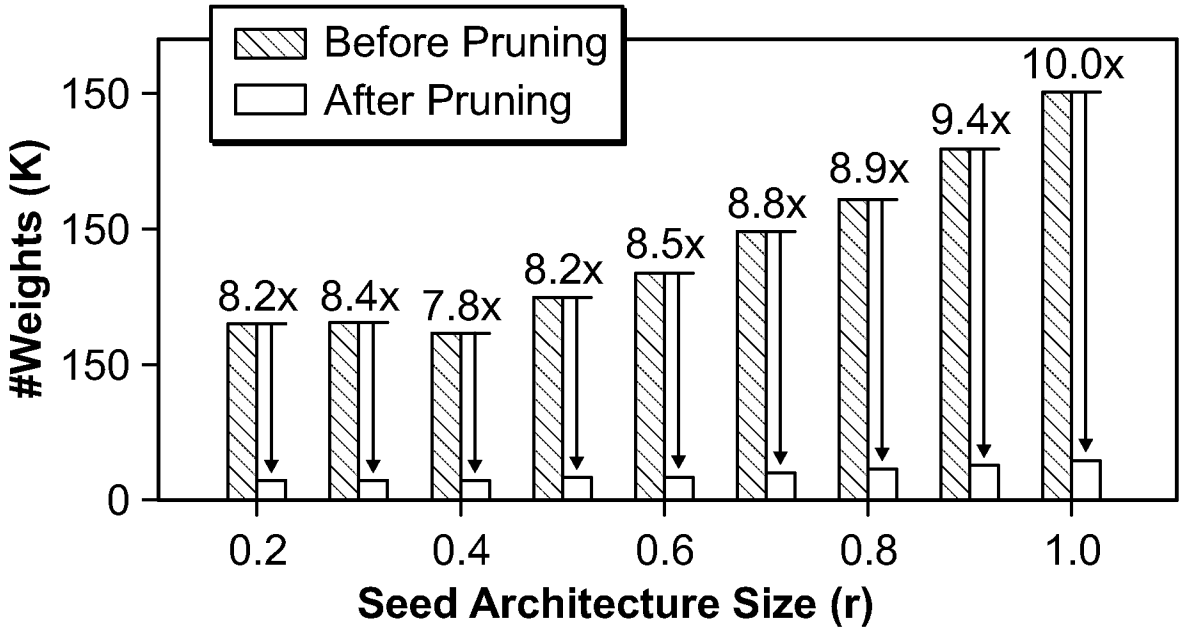


FIG. 10A

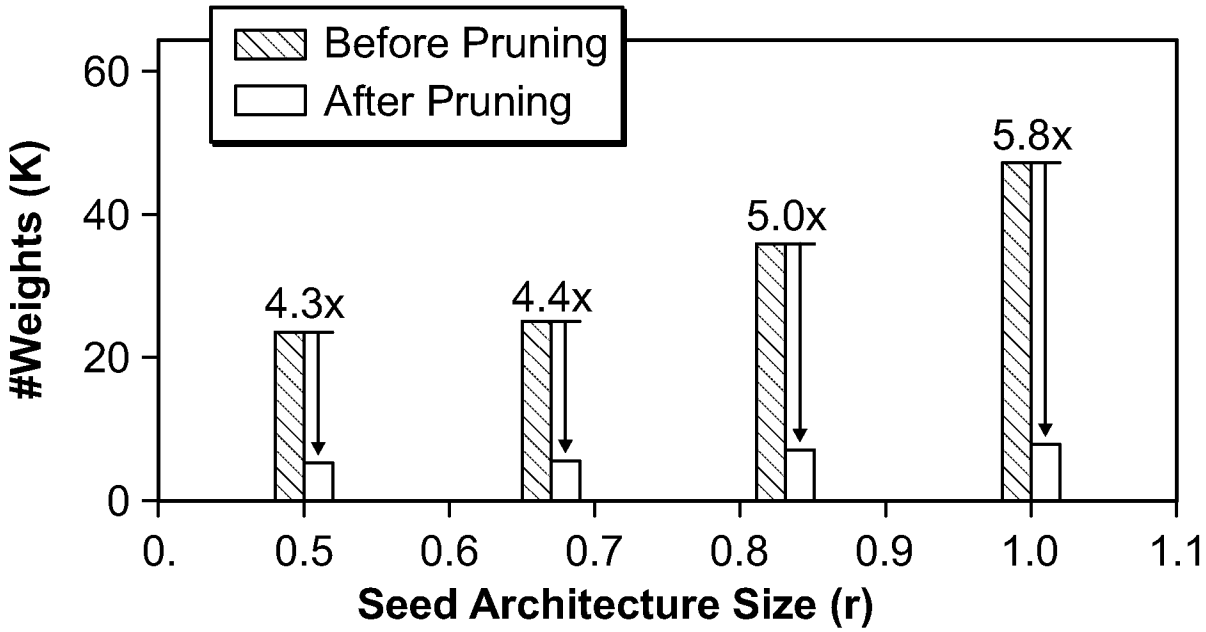


FIG. 10B

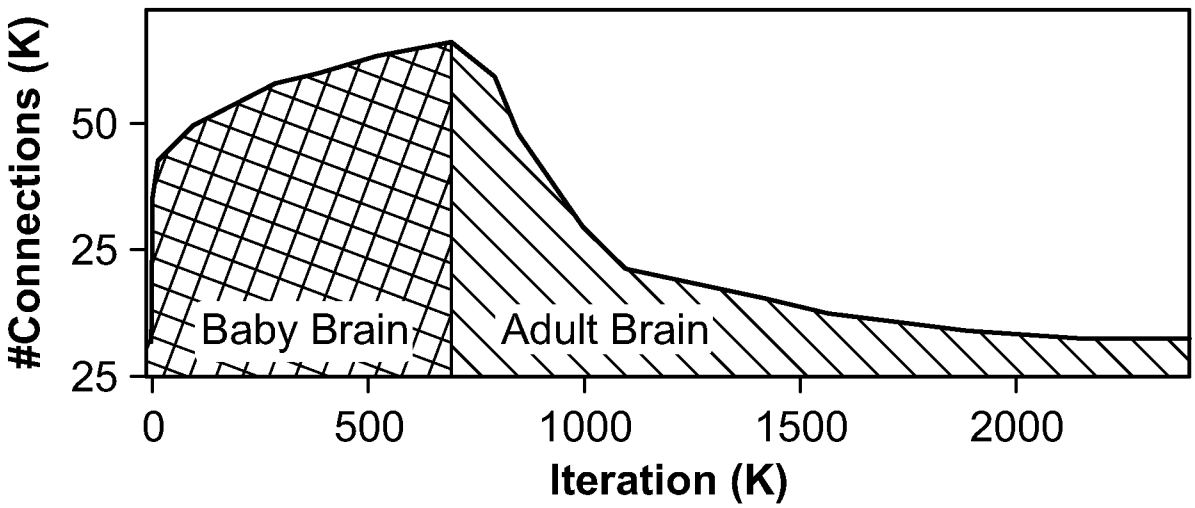

#Connections (K)
50
25
25
Baby Brain
Adult Brain
0
500
1000
1500
2000
Iteration (K)

FIG. 11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 18/57485

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06N 3/02, G06F 15/18 (2018.01)
CPC - G06N 3/02, G06N 3/063, G06N 3/08, G06N 3/0454, G06N 3/049

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)
See Search History Document

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2016/0174902 A1 (Siemens Aktiengesellschaft) 23 June 2016 (23.06.2016), entire document, especially abstract and para [0086]-[0096], [0120]-[0126].	1-27
A	US 2017/0061250 A1 (Microsoft Technology Licensing, LLC) 02 March 2017 (02.03.2017), entire document.	1-27

☐ 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
17 December 2018 (17.12.2018)	11 JAN 2019
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (January 2015)