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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 March 2020 (26.03.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/060659 A1

(51) International Patent Classification:
G06N 3/02 (2006.01) *G06F 15/18* (2006.01)
G06N 3/08 (2006.01)

(21) International Application Number:
PCT/US2019/041531

(22) International Filing Date:
12 July 2019 (12.07.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/732,620 18 September 2018 (18.09.2018) US
62/835,694 18 April 2019 (18.04.2019) US

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

(72) Inventors: **HASSANTABAR, Shayan**; 1447 Olden Avenue, Ewing, NJ 08638 (US). **WANG, Zeyu**; 10 Lawrence Drive, Apt. 401, Princeton, NJ 08540 (US). **JHA, Niraj, K.**; 20 Norbridge Drive, 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).

(54) Title: SYSTEM AND METHOD FOR SYNTHESIS OF COMPACT AND ACCURATE NEURAL NETWORKS (SCANN)

(57) Abstract: According to various embodiments, a method for generating a compact and accurate neural network for a dataset is disclosed. The method includes providing an initial neural network architecture; performing a dataset modification on the dataset, the dataset modification including reducing dimensionality of the dataset; performing a first compression step on the initial neural network architecture that results in a compressed neural network architecture, the first compression step including reducing a number of neurons in one or more layers of the initial neural network architecture based on a feature compression ratio determined by the reduced dimensionality of the dataset; and performing a second compression step on the compressed neural network architecture, the second compression step including one or more of iteratively growing connections, growing neurons, and pruning connections until a desired neural network architecture has been generated.

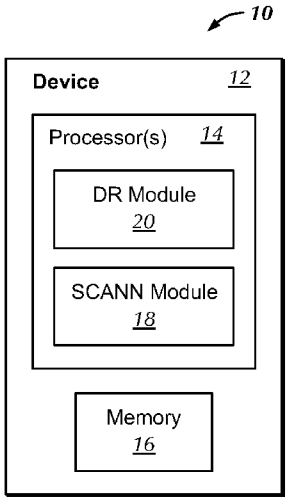

FIG. 1

WO 2020/060659 A1

[Continued on next page]

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

SYSTEM AND METHOD FOR SYNTHESIS OF COMPACT AND ACCURATE NEURAL NETWORKS (SCANN)

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to provisional applications 62/732,620 and 62/835,694, filed September 18, 2018 and April 18, 2019, respectively, which are herein incorporated by reference in their entirety.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[0002] This invention was made with government support under Grant #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 system and method that can generate compact neural networks without loss in accuracy.

BACKGROUND OF THE INVENTION

[0004] Artificial neural networks (ANNs) have a long history, dating back to 1950's. However, interest in ANNs has waxed and waned over the years. The recent spurt in interest in ANNs is due to large datasets becoming available, enabling ANNs to be trained to high accuracy. This trend is also due to a significant increase in compute power that speeds up the training process. ANNs demonstrate very high classification accuracies for many applications of interest, e.g., image recognition, speech recognition, and machine translation. ANNs have also become deeper, with tens to hundreds of layers. Thus, the phrase 'deep learning' is often associated with such neural networks. Deep learning refers to the ability of ANNs to learn hierarchically, with complex features built upon simple ones.

[0005] An important challenge in deploying ANNs in practice is their architecture design, since the ANN architecture directly influences the learnt representations and thus the performance. Typically, it takes researchers a huge amount of time through much trial-and-error to find a good architecture because the search space is exponentially large with respect to many of its hyperparameters. As an example, consider a convolutional neural network (CNN)

often used in image recognition tasks. Its various hyperparameters, such as depth, number of filters in each layer, kernel size, how feature maps are connected, etc., need to be determined when designing an architecture. Improvements in such architectures often take several years of effort, as evidenced by the evolution of various architectures for the ImageNet dataset: AlexNet, GoogleNet, ResNet, and DenseNet.

[0006] Another challenge ANNs pose is that to obtain their high accuracy, they need to be designed with a large number of parameters. This negatively impacts both the training and inference times. For example, modern deep CNNs often have millions of parameters and take days to train even with powerful graphics processing units (GPUs). However, making the ANN models compact and energy-efficient may enable them to be moved from the cloud to the edge, leading to benefits in communication energy, network bandwidth, and security. The challenge is to do so without degrading accuracy.

[0007] As the number of features or dimensions of the dataset increases, in order to generalize accurately, exponentially more data is needed. This is another challenge which is referred to as the curse of dimensionality. Hence, one way to reduce the need for large amounts of data is to reduce the dimensionality of the dataset. In addition, with the same amount of data, by reducing the number of features, the accuracy of the inference model may also improve to a degree. However, beyond a certain point, which is dataset-dependent, reducing the number of features may lead to loss of information, which may lead to inferior classification results.

[0008] At least these problems pose a significant design challenge in obtaining compact and accurate neural networks.

SUMMARY OF THE INVENTION

[0009] According to various embodiments, a method for generating a compact and accurate neural network for a dataset is disclosed. The method includes providing an initial neural network architecture; performing a dataset modification on the dataset, the dataset modification including reducing dimensionality of the dataset; performing a first compression step on the initial neural network architecture that results in a compressed neural network architecture, the first compression step including reducing a number of neurons in one or more layers of the initial neural network architecture based on a feature compression ratio determined by the reduced dimensionality of the dataset; and performing a second compression step on the compressed neural network architecture, the second compression step including one or more of iteratively growing connections, growing neurons, and pruning connections until a desired neural network architecture has been generated.

[0010] According to various embodiments, a system for generating a compact and accurate neural network for a dataset is disclosed. The system includes one or more processors configured to provide an initial neural network architecture; perform a dataset modification on the dataset, the dataset modification including reducing dimensionality of the dataset; perform a first compression step on the initial neural network architecture that results in a compressed neural network architecture, the first compression step including reducing a number of neurons in one or more layers of the initial neural network architecture based on a feature compression ratio determined by the reduced dimensionality of the dataset; and perform a second compression step on the compressed neural network architecture, the second compression step including one or more of iteratively growing connections, growing neurons, and pruning connections until a desired neural network architecture has been generated.

[0011] 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 a compact and accurate neural network for a dataset is disclosed. The method includes providing an initial neural network architecture; performing a dataset modification on the dataset, the dataset modification including reducing dimensionality of the dataset; performing a first compression step on the initial neural network architecture that results in a compressed neural network architecture, the first compression step including reducing a number of neurons in one or more layers of the initial neural network architecture based on a feature compression ratio determined by the reduced dimensionality of the dataset; performing a second compression step on the compressed neural network architecture, the second compression step including one or more of iteratively growing connections, growing neurons, and pruning connections until a desired neural network architecture has been generated.

[0012] Various other features and advantages will be made apparent from the following detailed description and the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] 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:

[0014] Figure 1 depicts a block diagram of a system for SCANN or DR+SCANN according to an embodiment of the present invention;

[0015] Figure 2 depicts a diagram illustrating hidden layers of hidden neurons according to an embodiment of the present invention;

[0016] Figure 3 depicts a methodology for automatic architecture synthesis according to an embodiment of the present invention;

[0017] Figure 4 depicts a diagram of architecture synthesis according to an embodiment of the present invention;

[0018] Figure 5 depicts a methodology for connection growth according to an embodiment of the present invention;

[0019] Figure 6 depicts a methodology for neuron growth according to an embodiment of the present invention;

[0020] Figure 7 depicts a methodology for connection pruning according to an embodiment of the present invention;

[0021] Figure 8 depicts a diagram of training schemes according to an embodiment of the present invention;

[0022] Figure 9 depicts a block diagram of DR+SCANN according to an embodiment of the present invention;

[0023] Figure 10 depicts a diagram of neural network compression according to an embodiment of the present invention;

[0024] Figure 11 depicts a table of dataset characteristics according to an embodiment of the present invention;

[0025] Figure 12 depicts a table comparing different training schemes according to an embodiment of the present invention;

[0026] Figure 13 depicts a table showing test accuracy according to an embodiment of the present invention;

[0027] Figure 14 depicts a table showing neural network parameters according to an embodiment of the present invention; and

[0028] Figure 15 depicts a table showing inference energy consumption according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0029] Artificial neural networks (ANNs) have become the driving force behind recent artificial intelligence (AI) research. With the help of a vast amount of training data, neural

networks can perform better than traditional machine learning algorithms in many applications, such as image recognition, speech recognition, and natural language processing. An important problem with implementing a neural network is the design of its architecture. Typically, such an architecture is obtained manually by exploring its hyperparameter space and kept fixed during training. The architecture that is selected is the one that performs the best on a hold-out validation set. This approach is both time-consuming and inefficient as it is in essence a trial-and-error process. Another issue is that modern neural networks often contain millions of parameters, whereas many applications require small inference models due to imposed resource constraints, such as energy constraints on battery-operated devices. Also, whereas ANNs have found great success in big-data applications, there is also significant interest in using ANNs for medium- and small-data applications that can be run on energy-constrained edge devices. However, efforts to migrate ANNs to such devices typically entail a significant loss of classification accuracy.

[0030] To address these challenges, generally disclosed herein is a neural network synthesis system and method, referred to as SCANN, that can generate compact neural networks without loss in accuracy for small and medium-size datasets. With the help of three operations, connection growth, neuron growth, and connection pruning, SCANN synthesizes an arbitrary feed-forward neural network with arbitrary depth. These neural networks do not necessarily have a multilayer perceptron structure. SCANN allows skipped connections, instead of enforcing a layer-by-layer connection structure. SCANN encapsulates three synthesis methodologies that apply a repeated grow-and-prune paradigm to three architectural starting points. Dimensionality reduction methods are also implemented to reduce the feature size of the datasets, so as to alleviate the curse of dimensionality. The approach generally includes three steps: dataset dimensionality reduction, neural network compression in each layer, and neural network compression with SCANN. The neural network synthesis system and method with dimensionality reduction may by referred to as DR+SCANN.

[0031] The efficacy of this approach is demonstrated on a medium-size MNIST dataset by comparing SCANN-synthesized neural networks to a LeNet-5 baseline. Without any loss in accuracy, SCANN generates a 46.3x smaller network than the LeNet-5 Caffe model. The efficiency is also evaluated using dimensionality reduction alongside SCANN on nine small- to medium-size datasets. Using this approach enables reduction of the number of connections in the network by up to 5078.7x (geometric mean: 82.1x), with little to no drop in accuracy. It is also shown that this approach yields neural networks that are much better at navigating the

accuracy vs. energy efficiency space. This can enable neural network based inference even for IoT sensors.

[0032] General Overview

[0033] This section is a general overview of dimensionality reduction and automatic architecture synthesis.

[0034] *Dimensionality Reduction*

[0035] The high dimensionality of many datasets used in various applications of machine learning leads to the curse of dimensionality problem. Therefore, dimensionality reduction methods may be used to improve the performance of machine learning models by decreasing the number of features. Some dimensionality reduction methods include but are not limited to Principal Component Analysis (PCA), Kernel PCA, Factor Analysis (FA), Independent Component Analysis (ICA), as well as Spectral Embedding methods. Some graph-based methods include but are not limited to Isomap and Maximum Variance Unfolding. Another nonlimiting example, FeatureNet, uses community detection in small sample size datasets to map high-dimensional data to lower dimensions. Other dimensionality reduction methods include but are not limited to stochastic proximity embedding (SPE), Linear Discriminant Analysis (LDA), and t-distributed Stochastic Neighbor Embedding (t-SNE).

[0036] *Automatic Architecture Synthesis*

[0037] There are generally three different categories of automatic architecture synthesis methods: evolutionary algorithm, reinforcement learning algorithm, and structure adaptation algorithm.

[0038] Evolutionary Algorithm

[0039] As the name implies, evolutionary algorithms are heuristic approaches for architecture synthesis influenced by biological evolution. One of the seminal works in neuroevolution is the NEAT algorithm, which uses direct encoding of every neuron and connection to simultaneously evolve the network architecture and weights through weight mutation, connection mutation, node mutation, and crossover. Extensions of the evolutionary algorithm can be used to generate CNNs.

[0040] Reinforcement Learning Algorithm

[0041] Reinforcement learning algorithms update architecture synthesis based on rewards received from actions taken. For instance, a recurrent neural network can be used as a controller to generate a string that specifies the network architecture. The performance of the generated network is used on a validation dataset as the reward signal to compute the policy gradient and update the controller. Similarly, the controller can be used with a different defined search space

to obtain a building block instead of the whole network. Convolutional cells obtained by learning performed on one dataset can be successfully transferred to architectures for other datasets.

[0042] Structure Adaptation Algorithm

[0043] Architecture synthesis can be achieved by altering the number of connections and/or neurons in the neural network. A nonlimiting example is network pruning. Structure adaptation algorithms can be constructive or destructive, or both constructive and destructive. Constructive algorithms start from a small neural network and grow it into a larger more accurate neural network. Destructive algorithms start from a large neural network and prune connections and neurons to get rid of the redundancy while maintaining accuracy. A couple nonlimiting examples of this architecture synthesis can generally be found in PCT Application Nos. PCT/US2018/057485 and PCT/US2019/22246, which are herein incorporated by reference in their entirety. One of these applications describes a network synthesis tool that combines both the constructive and destructive approaches in a grow-and-prune synthesis paradigm to create compact and accurate architectures for the MNIST and ImageNet datasets. If growth and pruning are both performed at a specific ANN layer, network depth cannot be adjusted and is fixed throughout training. This problem can be solved by synthesizing a general feed-forward network instead of an MLP architecture, allowing the ANN depth to be changed dynamically during training, to be described in further detail below. The other of these applications combines the grow-and-prune synthesis methodology with hardware-guided training to achieve compact long short-term memory (LSTM) cells. Some other nonlimiting examples include platform-aware search for an optimized neural network architecture, training an ANN to satisfy predefined resource constraints (such as latency and energy consumption) with help from a pre-generated accuracy predictor, and quantization to reduce computations in a network with little to no accuracy drop.

[0044] System Overview

[0045] Figure 1 illustrates a system 10 configured to implement SCANN or DR+SCANN. 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 can also include network appliances and Internet of Things (IoT) devices as well such as IoT sensors. 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 SCANN module 18 and optional dimensionality reduction (DR) module 20 for synthesizing neural network architectures. The SCANN module 18 and DR module 20 methodology will be described in greater detail below.

[0046] It is also to be noted the training process for SCANN or DR+SCANN may be implemented in a number of configurations with a variety of processors (including but not limited to central processing units (CPUs), graphics processing units (GPUs), and field programmable gate arrays (FPGAs)), such as servers, desktop computers, laptop computers, tablets, and the like.

[0047] SCANN Synthesis Methodology

[0048] This section first proposes a technique so that ANN depth no longer needs to be fixed, then introduces three architecture-changing techniques that enable synthesis of an optimized feedforward network architecture, and last describes three training schemes that may be used to synthesize network architecture.

[0049] *Depth Change*

[0050] To address the problem of having to fix the ANN depth during training, embodiments of the present invention adopt a general feedforward architecture instead of an MLP structure. Specifically, a hidden neuron can receive inputs from any neuron activated before it (including input neurons), and can feed its output to any neuron activated after it (including output neurons). In this setting, depth is determined by how hidden neurons are connected and thus can be changed through rewiring of hidden neurons. As shown in Figure 2, depending on how the hidden neurons are connected, they can form one hidden layer 22, two hidden layers 24, or three hidden layers 26. The one hidden layer 22 neural network is due to the neurons not being connected and all of them being in the same layer. For the two hidden layers 24 neural network, the neurons are connected in layers. Similarly, for the three hidden layers 26 neural network, the neurons are connected in three layers. The top one has one skip connection while the bottom one does not.

[0051] *Overall Workflow*

[0052] The overall workflow for architecture synthesis is shown in Figure 3. The synthesis process iteratively alternates between architecture change and weight training. Thus, the network architecture evolves along the way. After a specified number of iterations, the checkpoint that achieves the best performance on the validation set is output as the final network.

[0053] *Architecture-changing Operations*

[0054] Three general operations, connection growth, neuron growth, and connection pruning, are used to adjust the network architecture, in order to evolve a feed-forward network just through these operations. Figure 4 shows a simple example in which an MLP architecture with one hidden layer evolves into a non-MLP architecture with two hidden layers with a sequence of the operations mentioned above. It is to be noted the order of operations shown is purely for illustrative purposes and is not intended to be limiting. The operations can be performed in any order any number of times until a final architecture is determined. An initial architecture is first shown at step 28, a neuron growth operation is shown at step 30, a connection growth operation is shown at step 32, a connection pruning operation is shown at step 34, and a final architecture is shown at step 36.

[0055] These three operations will now be described in greater detail. The ith hidden neuron is denoted as n_i, its activity as x_i, and its pre-activity as u_i, where $x_i = f(u_i)$ and f is the activation function. The depth of n_i is denoted as D_i and the loss function as L. The connection between n_i and n_j, where $D_i \leq D_j$, is denoted as ω_{ij}. Masks may be used to mask out pruned weights in implementation.

[0056] Connection Growth

[0057] Connection growth adds connections between neurons that are unconnected. The initial weights of all newly added connections are set to 0. Depending on how connections can be added, at least three different methods may be used, as shown in Figure 5. These are gradient-based growth, full growth, and random growth.

[0058] Gradient-based growth adds connections that tend to reduce the loss function L significantly. Supposing two neurons n_i and n_j are not connected and $D_i \leq D_j$, then gradient-based growth adds a new connection ω_{ij} if $\left|\frac{\partial L}{\partial u_j} x_i\right|$ is large based on a predetermined threshold, for example, adding the top 20 percent of the connections based on the gradients.

[0059] Full growth restores all possible connections to the network.

[0060] Random growth randomly picks some inactive connections and adds them to the network.

[0061] Neuron Growth

[0062] Neuron growth adds new neurons to the network, thus increasing network size over time. There are at least two possible methods for doing this, as shown in Figure 6.

[0063] For the first method, drawing an analogy from biological cell division, neuron growth can be achieved by duplicating an existing neuron. To break the symmetry, random

noise is added to the weights of all the connections related to this newly added neuron. The specific neuron that is duplicated can be selected in at least two ways. Activation-based selection selects neurons with a large activation for duplication and random selection randomly selects neurons for duplication. Large activation is determined based on a predefined threshold, for example, the top 30% of neurons, in terms of their activation, are selected for duplication.

[0064] For the second method, instead of duplicating existing neurons, new neurons with random initial weights and random initial connections with other neurons may be added to the network.

[0065] Connection Pruning

[0066] Connection pruning disconnects previously connected neurons and reduces the number of network parameters. If all connections associated with a neuron are pruned, then the neuron is removed from the network. As shown in Figure 7, one method for pruning connections is to remove connections with a small magnitude. Small magnitude is based on a predefined threshold. The rationale behind it is that since small weights have a relatively small influence on the network, ANN performance can be restored through retraining after pruning.

[0067] *Training Schemes*

[0068] Depending on how the initial network architecture A_{init} and the three operations described above are chosen, one or more of three training schemes can be adopted.

[0069] Scheme A

[0070] Scheme A is a constructive approach, where the network size is gradually increased from an initially smaller network. This can be achieved by performing connection and neuron growth more often than connection pruning or carefully selecting the growth and pruning rates, such that each growth operation grows a larger number of connections and neurons, while each pruning operation prunes a smaller number of connections.

[0071] Scheme B

[0072] Scheme B is a destructive approach, where the network size is gradually decreased from an initially over-parametrized network. There are at least two possible ways to accomplish this. First, a small number of network connections can be iteratively pruned and then the weights can be trained. This gradually reduces network size and finally results in a small network after many iterations. Another approach is that, instead of pruning the network gradually, the network can be aggressively pruned to a substantially smaller size. However, to make this approach work, the network needs to be repeatedly pruned and then the network needs to be grown back, rather than performing a one-time pruning.

[0073] Scheme C

[0074] Scheme B can also work with MLP architectures, with only a small adjustment in connection growth such that only connections between adjacent layers are added and not skipped connections. For clarity, MLP-based Scheme B will be referred to as Scheme C. Scheme C can also be viewed as an iterative version of a dense-sparse-dense technique, with the aim of generating compact networks instead of improving performance of the original architecture. It is to be noted that for Scheme C, the depth of the neural network is fixed.

[0075] Figure 8 shows example of the initial and final architectures for each scheme. An initial architecture 38 and a final architecture 40 is shown for Scheme A, an initial architecture 42 and a final architecture 44 is shown for Scheme B, and an initial architecture 46 and a final architecture 48 is shown for Scheme C. Both Schemes A and B evolve general feedforward architectures, thus allowing network depth to be changed during training. Scheme C evolves an MLP structure, thus keeping the depth fixed.

[0076] Dimensionality Reduction + SCANN

[0077] This section illustrates a methodology to synthesize compact neural networks by combining dimensionality reduction (DR) and SCANN, referred to herein as DR+SCANN. Figure 9 shows a block diagram of the methodology, starting with an original dataset 50. The methodology begins by obtaining an accurate baseline architecture at step 52 by progressively increasing the number of hidden layers. This leads to an initial MLP architecture 54. The other steps are a dataset modification step 56, a first neural network compression step 58, and a second neural network compression step 60, to be described in the following sections. A final compressed neural network architecture 62 results from these steps.

[0078] *Dataset Modification 56*

[0079] Dataset modification entails normalizing the dataset and reducing its dimensionality. All feature values are normalized to the range [0,1]. Reducing the number of features in the dataset is aimed at alleviating the effect of the curse of dimensionality and increasing data classifiability. This way, an $N \times d$-dimensional dataset is mapped onto an $N \times k$-dimensional space, where $k < d$, using one or more dimensionality reduction methods. A number of nonlimiting methods are described below as examples.

[0080] Random projection (RP) methods are used to reduce data dimensionality based on the lemma that if the data points are in a space of sufficiently high dimension, they can be projected onto a suitable lower dimension, while approximately maintaining inter-point distances. More precisely, this lemma shows that the distance between the points change only by a factor of $(1 \pm \varepsilon)$ when they are randomly projected onto the subspace of $\mathcal{O}(\log \frac{n}{\varepsilon^2})$

dimensions for any $0 < \varepsilon < 1$. The RP matrix Φ can be generated in several ways. Four RP matrices are described here as nonlimiting examples.

[0081] One approach is to generate Φ using a Gaussian distribution. In this case, the entries ϕ_{ij} are i.i.d. samples drawn from a Gaussian distribution $\mathcal{N}(0, \frac{1}{k})$. Another RP matrix can be obtained by sampling entries from $\mathcal{N}(0,1)$. These entries are shown below:

$$\phi_{ij}^{1} \sim \mathcal{N}\left(0, \frac{1}{k}\right)$$

$$\phi_{ij}^{2} \sim \mathcal{N}(0,1)$$

[0082] Several other sparse RP matrices can be utilized. Two are as follows, where ϕ_{ij}'s are independent random variables that are drawn based on the following probability distributions:

$$\phi_{ij}^{3} = \begin{cases} +1 \text{ with probability } \frac{1}{2} \\ -1 \text{ with probability } \frac{1}{2} \end{cases}$$

$$\phi_{ij}^{4} = \sqrt{\frac{3}{k}} \begin{cases} +1 \text{ with probability } \frac{1}{6} \\ 0 \text{ with probability } \frac{2}{3} \\ -1 \text{ with probability } \frac{1}{6} \end{cases}$$

[0083] The other dimensionality reduction methods that can be used include but are not limited to principal component analysis (PCA), polynomial kernel PCA, Gaussian kernel PCA, factor analysis (FA), isomap, independent component analysis (ICA), and spectral embedding.

[0084] *Neural Network Compression in each Layer 58*

[0085] Dimensionality reduction maps the dataset into a vector space of lower dimension. As a result, as the number of features reduces, the number of neurons in the input layer of the neural network decreases accordingly. However, since the dataset dimension is reduced, one might expect the task of classification to become easier. This means the number of neurons in all layers can be reduced, not just the input layer. This step reduces the number of neurons in each layer of the neural network by the feature compression ratio in the dimensionality reduction step, except for the output layer. Feature compression ratio is the ratio by which the number of features in the dataset are reduced. The number of neurons in each layers are reduced by the same ratio as the feature compression ratio. Figure 10 shows an example of this process of compressing neural networks in each layer. While a compression ratio of 2 is shown, that

ratio number is only an example and is not intended to be limiting. This dimensionality reduction stage may be referred to as *DR*.

[0086] *Neural Network Compression with SCANN 60*

[0087] Several neural network architectures obtained from the output of the first neural network compression step are input to SCANN. These architectures correspond to the best three classification accuracies, as well as the three most compressed networks that meet the baseline accuracy of the initial MLP architecture, as evaluated on the validation set. SCANN uses the corresponding reduced-dimension dataset.

[0088] In Scheme A, the maximum number of connections in the networks should be set. This value is set to the number of connections in the neural network that results from the first compression step 58. This way, the final neural network will become smaller.

[0089] Schemes B and C should have the maximum number of neurons and the maximum number of connections be initialized. In addition, in these two training schemes, the final number of connections in the network also should be set. Furthermore, the number of layers in the MLP architecture synthesized by Scheme C should be predetermined. These parameters are initialized using the network architecture that is output from the first neural network compression step 58.

[0090] Experimental Results

[0091] This evaluates the performance of embodiments of SCANN and DR+SCANN on several small- to medium-size datasets. Figure 11 shows the characteristics of these datasets. The evaluation results are divided into two parts. The first part discusses results obtained by SCANN when applied to the widely used MNIST dataset. Compared to related work, SCANN generates neural networks with better classification accuracy and fewer parameters. The second part shows the results of experiments on nine other datasets. It is demonstrated that the ANNs generated by SCANN are very compact and energy-efficient, while maintaining performance. These results open up opportunities to use SCANN-generated ANNs in energy-constrained edge devices and IoT sensors.

[0092] *Experiments with MNIST*

[0093] MNIST is a dataset of handwritten digits, containing 60000 training images and 10000 test images. 10000 images are set aside from the training set as the validation set. The Lenet-5 Caffe model is adopted. For Schemes A and B, the feed-forward part of the network is learnt by SCANN, whereas the convolutional part is kept the same as in the baseline (Scheme A does not make any changes to the baseline, but Scheme B prunes the connections). For Scheme C, SCANN starts with the baseline architecture, and only learns the connections and

weights, without changing the depth of the network. All experiments use the stochastic gradient descent (SGD) optimizer with a learning rate of 0.03, momentum of 0.9, and weight decay of 1e-4. No other regularization technique like dropout or batch normalization is used. Each experiment is run five times and the average performance is reported.

[0094] The LeNet-5 Caffe model contains two convolutional layers with 20 and 50 filters, and also one fully-connected hidden layer with 500 neurons. For Scheme A, 400 hidden neurons are started with in the feed-forward part, 95 percent of the connections are randomly pruned out in the beginning, and then a sequence of connection growth is iteratively performed that activates 30 percent of all connections and connection pruning that prunes 25 percent of existing connections. For Scheme B, 400 hidden neurons are started with in the feed-forward part and a sequence of connection pruning is iteratively performed such that 3.3K connections are left in the convolutional part and 16K connections are left in the feedforward part, and connection growth is then performed such that 90 percent of all connections are restored. For Scheme C, a fully connected baseline architecture is started with and a sequence of connection pruning is iteratively performed such that 3.3K connections are left in the convolutional part and 6K connections are left in the feed-forward part, and connection growth is then performed such that all connections are restored.

[0095] Figure 12 summarizes the results. The baseline error rate is 0.72% with 430.5K parameters. The most compressed model generated by SCANN contains only 9.3K parameters (with a compression ratio of 46.3x over the baseline), achieving a 0.72% error rate when using Scheme C. Scheme A obtains the best error rate of 0.68%, however, with a lower compression ratio of 2.3x.

[0096] *Experiments with Other Datasets*

[0097] Though SCANN demonstrates very good compression ratios for LeNets on the medium-size MNIST dataset at similar or better accuracy, one may ask if SCANN can also generate compact neural networks from other medium and small datasets. To answer this question, nine other datasets are experimented with and evaluation results are presented on these datasets.

[0098] SCANN experiments are based on the Adam optimizer with a learning rate of 0.01 and weight decay of 1e-3. Results obtained by DR+SCANN are compared with those obtained by only applying SCANN, and also DR without using SCANN in a secondary compression step. Figure 13 shows the classification accuracy obtained. The MLP column shows the accuracy of the MLP baseline for each dataset. For all the other methods, two columns are presented, the left of which shows the highest achieved accuracy (H.A.) whereas the right one

shows the result for the most compressed network (M.C.). Furthermore, for the DR columns, the dimensionality reduction method employed is shown in parentheses. Figure 14 shows the number of parameters in the network for the corresponding columns in Figure 13.

[0099] SCANN-generated networks show improved accuracy for six of the nine datasets, as compared to the MLP baseline. The accuracy increase is between 0.41% to 9.43%. These results correspond to networks that are 1.2x to 42.4x smaller than the base architecture. Furthermore, DR+SCANN shows improvements on the highest classification accuracy on five out of the nine datasets, as compared to SCANN-generated results.

[0100] In addition, SCANN yields ANNs that achieve the baseline accuracy with fewer parameters on seven out of the nine datasets. For these datasets, the results show a connection compression ratio between 1.5x to 317.4x. Moreover, as shown in Figures 13 and 14, combining dimensionality reduction with SCANN helps achieve higher compression ratios. For these seven datasets, DR+SCANN can meet the baseline accuracy with a 28.0x to 5078.7x smaller network. This shows a significant improvement over the compression ratio achievable by just using SCANN.

[0101] The performance of applying DR without the benefit of the SCANN synthesis step is also reported. While these results show improvements, DR+SCANN can be seen to have much more compression power, relative to when DR and SCANN are used separately. This points to a synergy between DR and SCANN.

[0102] Although the classification performance is of great importance, in applications where computing resources are limited, e.g., in battery-operated devices, energy efficiency might be one of the most important concerns. Thus, energy performance of the algorithms should also be taken into consideration in such cases. To evaluate the energy performance, the energy consumption for inference is calculated based on the number of multiply accumulate and comparison (MAC) operations and the number of SRAM accesses. For example, a multiplication of two matrices of size $M \times N$ and $N \times K$ would require $(M \cdot N \cdot K)$ MAC operations and $(2 \cdot M \cdot N \cdot K)$ SRAM accesses. In their model, a single MAC operation, SRAM access, and comparison operation implemented in a $130nm$ CMOS process (which may be an appropriate technology for many IoT sensors) consumes 11.8 pJ, 34.6 pJ and 6.16 fJ, respectively. Figure 15 shows the energy consumption estimates per inference for the corresponding models discussed in Figures 13 and 14. DR+SCANN can be seen to have the best overall energy performance. Except for the Letter dataset (for which the energy reduction is only 17 percent), the compact ANNs generated by DR+SCANN consume one to four orders

of magnitude less energy than the baseline MLP models. Thus, this synthesis methodology is suitable for heavily energy-constrained devices, such as IoT sensors.

[0103] Conclusion

[0104] The advantages of SCANN and DR+SCANN are derived from its core benefit: the network architecture is allowed to dynamically evolve during training. This benefit is not directly available in several other existing automatic architecture synthesis techniques, such as the evolutionary and reinforcement learning based approaches. In those methods, a new architecture, whether generated through mutation and crossover in the evolutionary approach or from the controller in the reinforcement learning approach, needs to be fixed during training and trained from scratch again when the architecture is changed.

[0105] However, human learning is incremental. The brain gradually changes based on the presented stimuli. For example, studies of the human neocortex have shown that up to 40 percent of the synapses are rewired every day. Hence, from this perspective, SCANN takes inspiration from how the human brain evolves incrementally. SCANN's dynamic rewiring can be easily achieved through connection growth and pruning.

[0106] Comparisons between SCANN and DR+SCANN show that the latter results in a smaller network in nearly all the cases. This is due to the initial step of dimensionality reduction. By mapping data instances into lower dimensions, it reduces the number of neurons in each layer of the neural network, without degrading performance. This helps feed a significantly smaller neural network to SCANN. As a result, DR+SCANN synthesizes smaller networks relative to when only SCANN is used.

[0107] As such, embodiments generally disclosed herein are a system and method for a synthesis methodology that can generate compact and accurate neural networks. It solves the problem of having to fix the depth of the network during training that prior synthesis methods suffer from. It is able to evolve an arbitrary feed-forward network architecture with the help of three general operations: connection growth, neuron growth, and connection pruning. Experiments on the MNIST dataset show that, without loss in accuracy, SCANN generates a 46.3x smaller network than the LeNet-5 Caffe model. Furthermore, by combining dimensionality reduction with SCANN synthesis, significant improvements in the compression power of this framework was shown. Experiments with several other small to medium datasets show that SCANN and DR+SCANN can provide a good tradeoff between accuracy and energy efficiency in applications where computing resources are limited.

[0108] 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 a compact and accurate neural network for a dataset, the method comprising:

providing an initial neural network architecture;

performing a dataset modification on the dataset, the dataset modification comprising reducing dimensionality of the dataset;

performing a first compression step on the initial neural network architecture that results in a compressed neural network architecture, the first compression step comprising reducing a number of neurons in one or more layers of the initial neural network architecture based on a feature compression ratio determined by the reduced dimensionality of the dataset; and

performing a second compression step on the compressed neural network architecture, the second compression step comprising one or more of iteratively growing connections, growing neurons, and pruning connections until a desired neural network architecture has been generated.

2. The method of claim 1, further comprising adding one or more hidden layers to the initial neural network architecture.

3. The method of claim 1, wherein dataset modification further comprises normalizing the dataset.

4. The method of claim 1, wherein reducing dimensionality of the dataset comprises one or more of random projection, principal component analysis (PCA), polynomial kernel PCA, Gaussian kernel PCA, factor analysis, isomap, independent component analysis, and spectral embedding.

5. The method of claim 1, wherein the first compression step further comprises computing the feature compression ratio.

6. The method of claim 1, wherein the second compression step comprises increasing a size of the compressed neural network architecture.

7. The method of claim 1, wherein the second compression step comprises decreasing a size of the compressed neural network architecture.

8. The method of claim 1, wherein growing connections further comprises growing connections only between adjacent layers.

9. The method of claim 1, wherein growing connections is based on one of gradient-based growth, full growth, and random growth.

10. The method of claim 1, wherein growing neurons is based on one of duplication and random addition.

11. The method of claim 1, wherein pruning connections is based on magnitude-based pruning.

12. A system for generating a compact and accurate neural network for a dataset, the system comprising one or more processors configured to:

provide an initial neural network architecture;

perform a dataset modification on the dataset, the dataset modification comprising reducing dimensionality of the dataset;

perform a first compression step on the initial neural network architecture that results in a compressed neural network architecture, the first compression step comprising reducing a number of neurons in one or more layers of the initial neural network architecture based on a feature compression ratio determined by the reduced dimensionality of the dataset; and

perform a second compression step on the compressed neural network architecture, the second compression step comprising one or more of iteratively growing connections, growing neurons, and pruning connections until a desired neural network architecture has been generated.

13. The system of claim 12, wherein the one or more processors are further configured to add one or more hidden layers to the initial neural network architecture.

14. The system of claim 12, wherein dataset modification further comprises normalizing the dataset.

15. The system of claim 12, wherein reducing dimensionality of the dataset comprises one or more of random projection, principal component analysis (PCA), polynomial kernel PCA, Gaussian kernel PCA, factor analysis, isomap, independent component analysis, and spectral embedding.

16. The system of claim 12, wherein the first compression step further comprises computing the feature compression ratio.

17. The system of claim 12, wherein the second compression step comprises increasing a size of the compressed neural network architecture.

18. The system of claim 12, wherein the second compression step comprises decreasing a size of the compressed neural network architecture.

19. The system of claim 12, wherein growing connections further comprises growing connections only between adjacent layers.

20. The system of claim 12, wherein growing connections is based on one of gradient-based growth, full growth, and random growth.

21. The system of claim 12, wherein growing neurons is based on one of duplication and random addition.

22. The system of claim 12, wherein pruning connections is based on magnitude-based pruning.

23. A non-transitory computer-readable medium having stored thereon a computer program for execution by a processor configured to perform a method for generating a compact and accurate neural network for a dataset, the method comprising:

providing an initial neural network architecture;

performing a dataset modification on the dataset, the dataset modification comprising reducing dimensionality of the dataset;

performing a first compression step on the initial neural network architecture that results in a compressed neural network architecture, the first compression step comprising reducing a number of neurons in one or more layers of the initial neural network architecture based on a feature compression ratio determined by the reduced dimensionality of the dataset;

performing a second compression step on the compressed neural network architecture, the second compression step comprising one or more of iteratively growing connections, growing neurons, and pruning connections until a desired neural network architecture has been generated.

24. The computer-readable medium of claim 23, further comprising adding one or more hidden layers to the initial neural network architecture.

25. The computer-readable medium of claim 23, wherein dataset modification further comprises normalizing the dataset.

26. The computer-readable medium of claim 23, wherein reducing dimensionality of the dataset comprises one or more of random projection, principal component analysis (PCA), polynomial kernel PCA, Gaussian kernel PCA, factor analysis, isomap, independent component analysis, and spectral embedding.

27. The computer-readable medium of claim 23, wherein the first compression step further comprises computing the feature compression ratio.

28. The computer-readable medium of claim 23, wherein the second compression step comprises increasing a size of the compressed neural network architecture.

29. The computer-readable medium of claim 23, wherein the second compression step comprises decreasing a size of the compressed neural network architecture.

30. The computer-readable medium of claim 23, wherein growing connections further comprises growing connections only between adjacent layers.

31. The computer-readable medium of claim 23, wherein growing connections is based on one of gradient-based growth, full growth, and random growth.

32. The computer-readable medium of claim 23, wherein growing neurons is based on one of duplication and random addition.

33. The computer-readable medium of claim 23, wherein pruning connections is based on magnitude-based pruning.

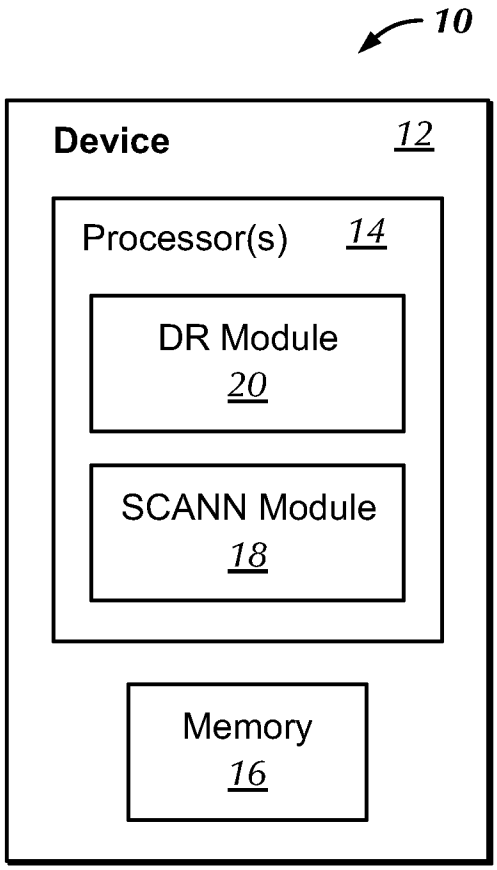
10
Device
12
Processor(s)
14
DR Module
20
SCANN Module
18
Memory
16

FIG. 1

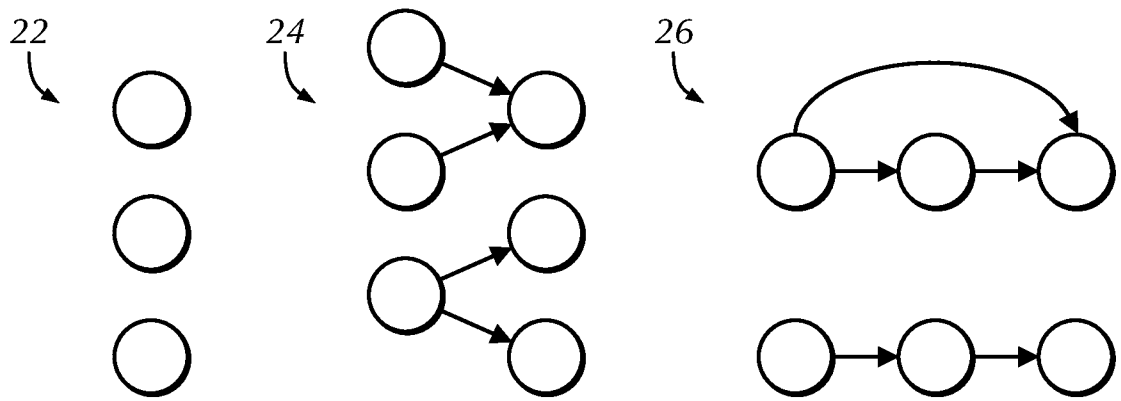

FIG. 2

Automatic Architecture Synthesis

Input: Initial network architecture A_{init}, weights W_{init}, and maximum number of iteration I_{max}

while maximum iterations I_{max} not reached **do**

(a) Perform one of the three basic architecture-changing operations

(b) Train weights of the network and test its performance on the validation set

end while

Output: Final network architecture A_{final} and associated weights W_{final} that achieve the best performance on the validation set

FIG. 3

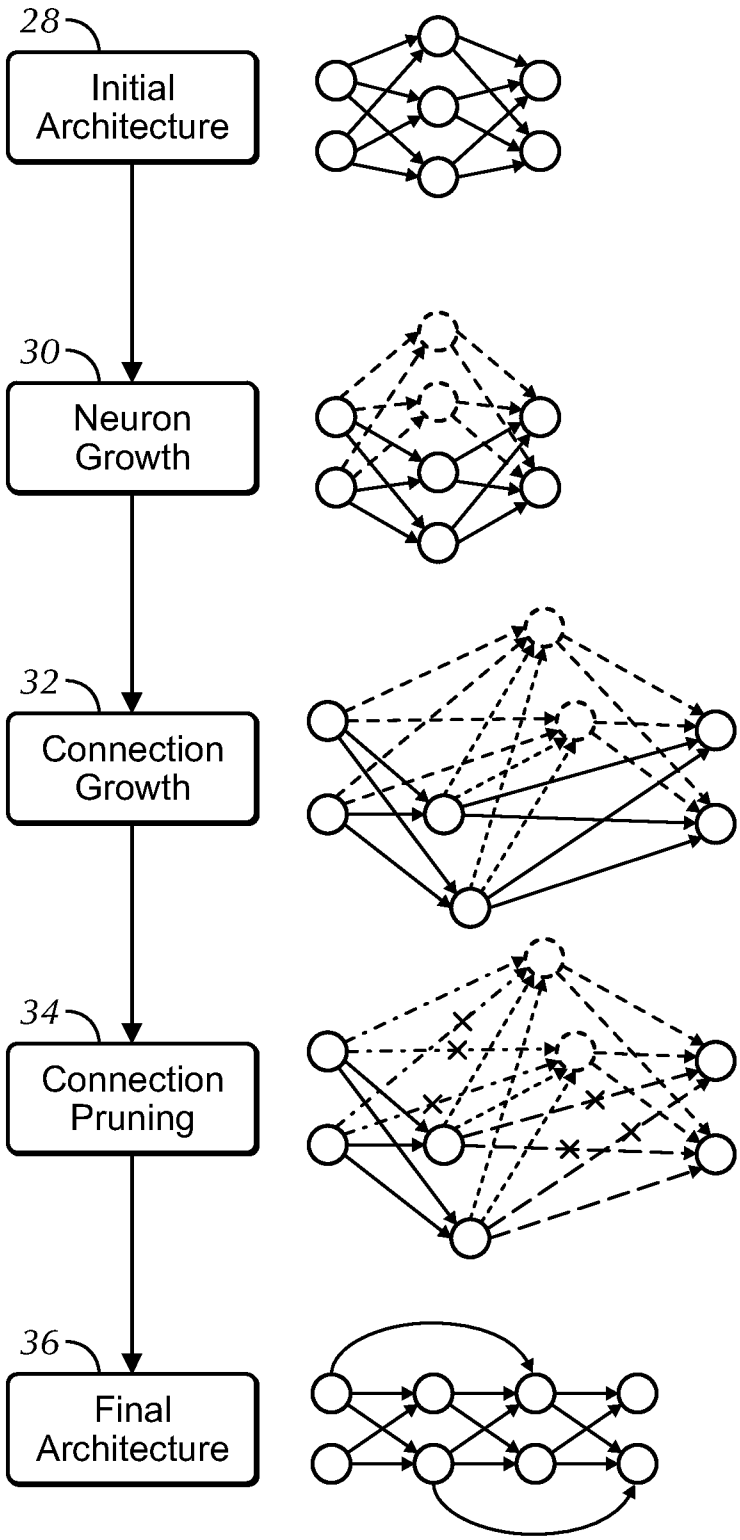
28
Initial Architecture
30
Neuron Growth
32
Connection Growth
34
Connection Pruning
36
Final Architecture

FIG. 4

Connection Growth Algorithm

Input: Network N, weight matrix W, mask matrix C,
data batch D, threshold t
if full growth **then**
set all elements to C to 1
else if random growth **then**
randomly set some elements in C to 1
else if gradient-base growth **then**
forward propagation through N using data D and
then back propagation

compute $g_{ij} = \left| \frac{\partial L}{\partial u_j} x_i \right|$

For $g_{ij} > t$, set $c_{ij} = 1$, $w_{ij} = 0$
end if
Output: Modified weight matrix W and mask matric C

FIG. 5

Neuron Growth Algorithm

Input: Network N, weight matrix W, mask matrix C,
data batch D, a candidate neuron n_j to be added
if neuron division **then**
if activation-based selection **then**
forward propagation through N using data D
$i = argmax\ u_i$
else if random selection **then**
randomly pick and active neuron n_i
end if
$c_{j\cdot} = c_{i\cdot}, c_{\cdot j} = c_{\cdot i}$
$w_{j\cdot} = w_{i\cdot} + noise,\ w_{\cdot j} = w_{\cdot i} + noise$
else if random growth **then**
randomly set elements of $c_{j\cdot}$ and $c_{\cdot j}$ to 1
randomly initialize $w_{j\cdot}$ and $w_{\cdot j}$
end if
Output: Modified weight matrix W and mask matric C

FIG. 6

Connection Pruning Algorithm

Input: Weight matrix W, mask matrix C, threshold t

for all w_{ij} **do**

if $|w_{ij}| < t$ **then**

$c_{ij} = 0$

end if

end for

Output: Modified weight matrix W and mask matric C

FIG. 7

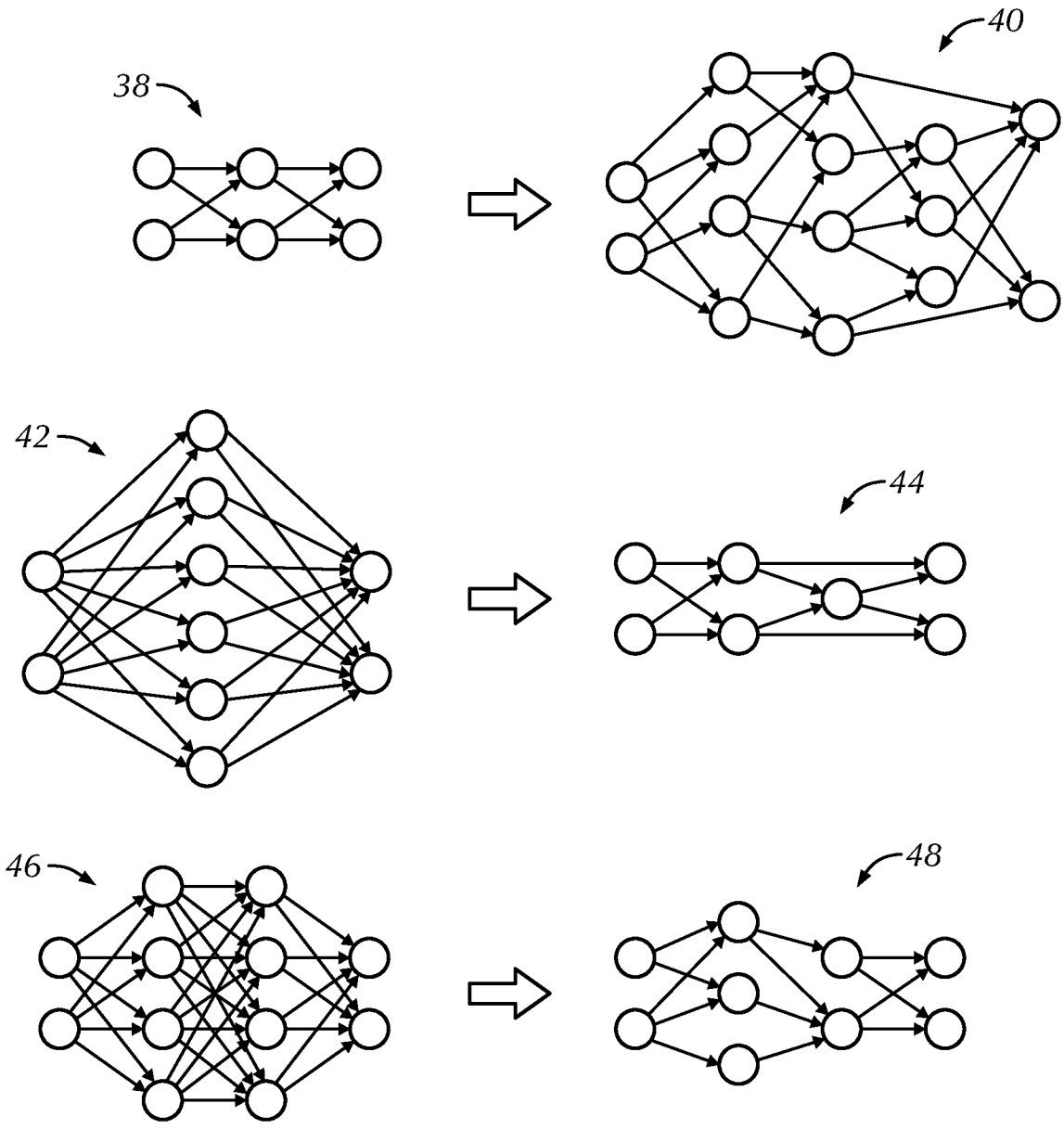

FIG. 8

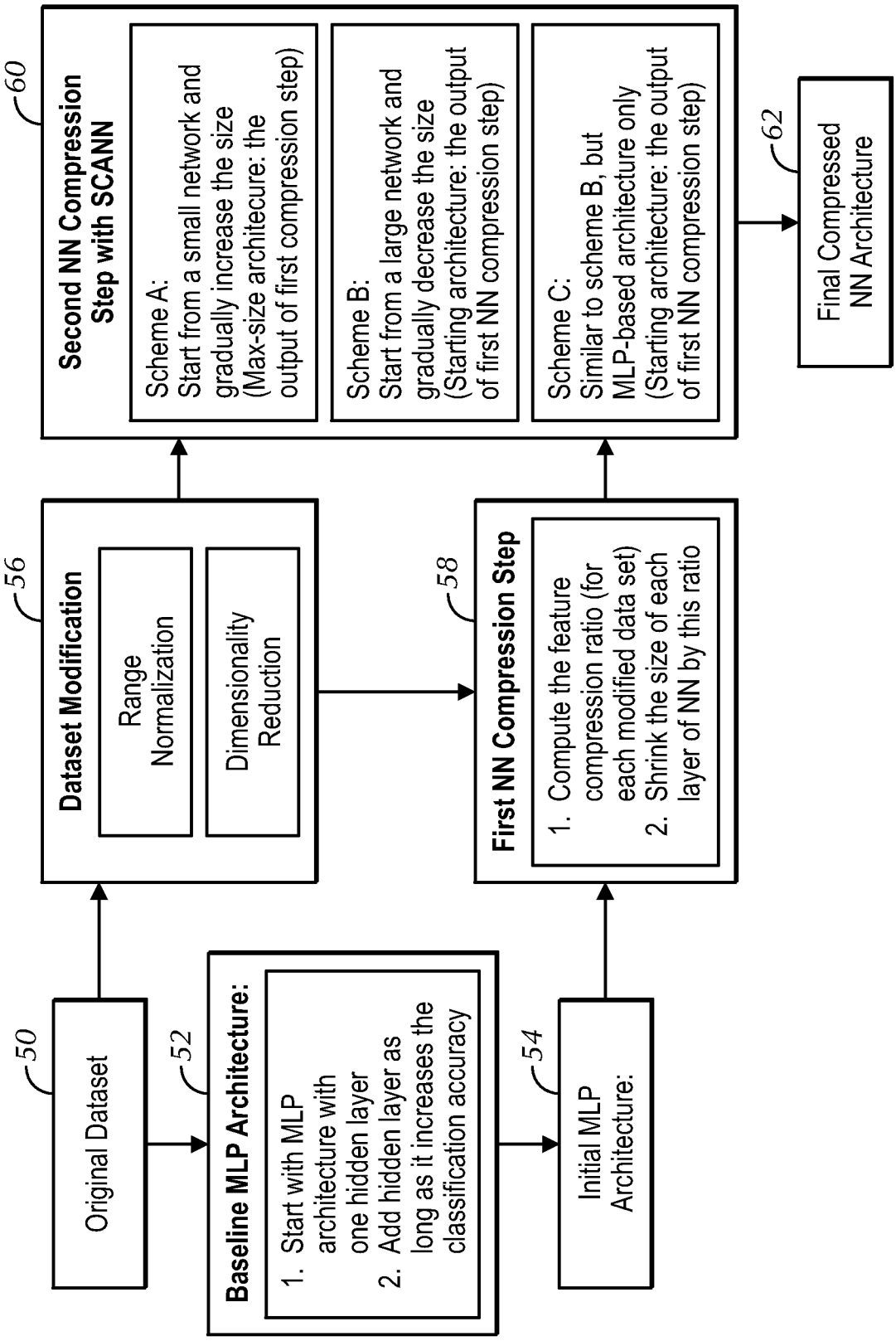
50
Original Dataset
52
Baseline MLP Architecture:
1. Start with MLP architecture with one hidden layer
2. Add hidden layer as long as it increases the classification accuracy
54
Initial MLP Architecture:
56
Dataset Modification
Range Normalization
Dimensionality Reduction
58
First NN Compression Step
1. Compute the feature compression ratio (for each modified data set)
2. Shrink the size of each layer of NN by this ratio
60
Second NN Compression Step with SCANN
Scheme A: Start from a small network and gradually increase the size (Max-size architecure: the output of first compression step)
Scheme B: Start from a large network and gradually decrease the size (Starting architecture: the output of first NN compression step)
Scheme C: Similar to scheme B, but MLP-based architecture only (Starting architecture: the output of first NN compression step)
62
Final Compressed NN Architecture

FIG. 9

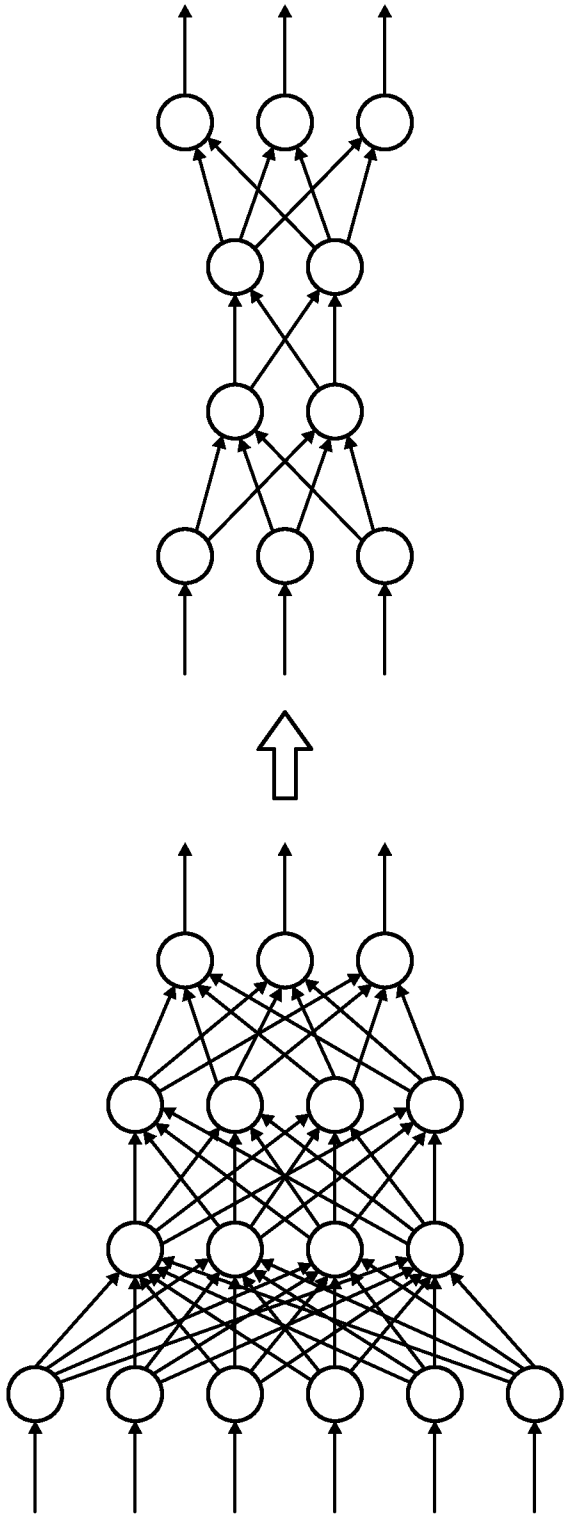

FIG. 10

Characteristics of the Datasets

Dataset	Training Set	Training Set	Test Set	Features	Classes
MNIST	50000	10000	10000	784	10
Sensorless Drive Diagnosis	40509	9000	9000	48	11
Human Activity Recognition (HAR)	5881	1471	2947	561	6
Musk v 2	4100	1000	1974	166	2
Pen-Based Recognition of Handwritten Digits	5995	1499	3498	16	10
Landsat Satellite Image	3104	1331	2000	36	6
Letter Recognition	10500	4500	5000	16	26
Epileptic Seizure Recognition	6560	1620	3320	178	2
Smartphone Human Activitiy Recognition	6121	153	3277	561	12
DNA	1400	600	1186	180	3

FIG. 11

Comparison of Different Methods on the LeNet-5 Caffe Model

Methods	Error Rate	Weights	Compression Ratio
Baseline	0.72%	430.5K	1.0x
Network Pruning [12]	0.77%	34.5K	12.5x
Scheme A	0.68%	184.6K	2.3x
Scheme B	0.72%	19.3K	22.3x
Scheme C	0.72%	9.3K	46.3x

FIG. 12

Test Accuracy Comparison

Dataset	MLP	DR (H.A.)	DR (M.C.)	SCANN (H.A.)	SCANN (M.C.)	DR+SCANN (H.A.)	DR+SCANN (M.C.)
SenDrive	93.53%	99.07% (FA)	97.99% (FA)	97.10%	93.63%	99.34%	94.20%
HAR	95.01%	95.04% (ICA)	95.04% (ICA)	95.52%	95.52%	95.28%	95.08%
Musk	98.68%	98.83% (FA)	98.78% (FA)	99.09%	98.83%	98.08%	98.08%
Pendigits	97.22%	97.51% (Isomap)	97.39% (Isomap)	97.22%	97.22%	97.93%	97.65%
Satlm	91.30%	91.10% (PCA)	91.10% (PCA)	90.10%	90.10%	89.40%	89.40%
Letter	95.24%	94.92% (PCA)	94.92% (PCA)	92.60%	92.60%	92.70%	92.70%
Seizure	87.53%	97.50% (FA)	95.42% (FA)	96.96%	96.23%	97.62%	95.72%
SHAR	90.66%	94.44% (RP)	90.69% (RP)	93.78%	90.93%	94.84%	90.93%
DNA	94.86%	94.69% (FA)	94.69% (FA)	95.86%	95.36%	93.76%	93.76%

FIG. 13

Neural Network Parameter Comparison

Dataset	MLP	DR (H.A.)	DR (M.C.)	SCANN (H.A.)	SCANN (M.C.)	DR+SCANN (H.A.)	DR+SCANN (M.C.)
SenDrive	56.9k (1x)	2.6k (21.9x)	1140 (49.9x)	10.0k (5.7x)	750 (75.9x)	2.2k (25.9x)	200 (284.5x)
HAR	212.0k (1x)	108.4k (1.9x)	108.4k (1.9x)	5.0k (42.4x)	5.0k (42.4x)	1.0k (212x)	750 (282.7x)
Musk	55.8k (1x)	17.3k (3.2x)	15.5k (3.6x)	22.0k (2.5x)	20.0k (2.8x)	600 (93.0x)	600 (93.0x)
Pendigits	4.9k (1x)	780 (6.3x)	671 (7.3x)	3.2k (1.5x)	3.2k (1.5x)	400 (12.2x)	175 (28.0x)
Satlm	3.8k (1x)	1.1k (3.4x)	1.1k (3.4x)	3.2k (1.5x)	3.2k (1.5x)	1.0k (3.8x)	1.0k (3.8x)
Letter	4.4k (1x)	3.8k (1.1x)	3.8k (1.1x)	3.8k (1.1x)	3.8k (1.1x)	3.7k (1.2x)	3.7k (1.2x)
Seizure	380.9k (1x)	10.5k (36.3x)	616 (618.3x)	3.0k (127.0x)	1.2k (317.4x)	1.8k (211.6x)	75 (5078.7x)
SHAR	214.0k (1x)	127.1k (1.7x)	3.7k (57.8x)	10.0k (21.4x)	800 (267.5x)	1.0k (214.0x)	500 (428.0x)
DNA	24.6k (1x)	22.9k (1.1x)	22.9k (1.1x)	20.0k (1.2x)	200 (123.0x)	300 (82.0x)	300 (82.0x)

FIG. 14

Inference Energy Consumption Comparison (J)

Dataset	MLP	DR (H.A.)	DR (M.C.)	SCANN (H.A.)	SCANN (M.C.)	DR+SCANN (H.A.)	DR+SCANN (M.C.)
SenDrive	4.6e-6	2.1e-7	8.9e-8	8.1e-7	6.1e-8	1.8e-7	1.6e-8
HAR	17.2e-6	8.8e-6	8.8e-6	4.0e-7	4.0e-7	8.1e-8	6.1e-8
Musk	4.5e-6	1.4e-6	1.2e-6	1.8e-6	1.6e-6	4.9e-8	4.9e-8
Pendigits	4.0e-7	6.3e-8	5.4e-8	2.6e-7	2.6e-7	3.2e-8	1.4e-8
Satlm	3.1e-7	8.9e-8	8.9e-8	2.6e-7	2.6e-7	8.1e-8	8.1e-8
Letter	3.6e-7	3.1e-7	3.1e-7	3.1e-7	3.1e-7	3.0e-7	3.0e-7
Seizure	3.1e-5	8.5e-7	5.0e-8	2.4e-7	9.7e-8	1.4e-7	6.1e-9
SHAR	1.7e-5	1.0e-5	3.0e-7	8.1e-7	6.5e-8	8.1e-8	4.0e-8
DNA	2.0e-6	1.8e-6	1.8e-6	1.6e-6	1.6e-8	2.4e-8	2.4e-8

FIG. 15

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US19/41531

A. CLASSIFICATION OF SUBJECT MATTER

IPC - G06N 3/02, 3/08, G06F 15/18 (2019.01)

CPC - G06N 3/02, 3/08, 3/0454, 3/049, 3/082; G06F 15/18

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 ---- Y	US 2016/0174902 A1 (SIEMENS AKTIENGESELLSCHAFT) 23 June 2016; paragraphs [0086], [0096]	1-3, 5, 11-14, 16, 22-25, 27, 33 --- 4, 6-10, 15, 17-21, 26, 28-32
Y	HAOZHE X et al. "A SURVEY OF DIMENSIONALITY REDUCTION TECHNIQUES BASED ON RANDOM PROJECTION" [online publication] 30 May 2018 [retrieved online: 12 September 2019] Retrieved from the Internet: <URL: https://arxiv.org/pdf/1711.02017.pdf>; page 4, paragraph 3	4, 15, 26
Y	XIAOLIANG D et al. "NeST: A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm" [online publication] 01 June 2018 [retrieved online: 12 September 2019] Retrieved from the Internet: <URL: https://arxiv.org/pdf/1711.02017.pdf>; page 3, paragraph 1, page 9, paragraph 7	6-9, 17-20, 28-31
Y	ISLAM M et al. "A New Adaptive Merging and Growing Algorithm for Designing Artificial Neural Networks" [online publication] 10 February 2009 [retrieved online: 12 September 2019] Retrieved from the Internet: <URL: https://www.researchgate.net/publication/224381435_A_New_Adaptive_Merging_and_Growing_Algorithm_for_Designing_Artificial_Neural_Networks>; page 709, paragraph 1	10, 21, 32
P, X	SHAYAN H et al. "SCANN: Synthesis of Compact and Accurate Neural Networks" [online publication] 19 April 2019 [retrieved online: 12 September 2019] Retrieved from the Internet: <URL: https://arxiv.org/pdf/1904.09090.pdf>; entire document	1-33

☐ 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
"D" document cited by the applicant in the international application
"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
12 September 2019 (12.09.2019)	08 OCT 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 Shane Thomas Telephone No. PCT Helpdesk: 571-272-4300