US 20230244935A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2023/0244935 A1**

LETOURNEAU et al. (43) **Pub. Date: Aug. 3, 2023**

(54) **TRAINING OF NEURAL NETWORK WITH POLYNOMIAL SOLVER**

(71) Applicant: **RESERVOIR LABS, INC.**, San Diego, CA (US)

(72) Inventors: **Pierre-David LETOURNEAU**, Philadelphia, PA (US); **Matthew Harper LANGSTON**, Beacon, NY (US); **Richard LETHIN**, Bronxville, NY (US); **Matthew James MORSE**, New York, NY (US)

(21) Appl. No.: **18/125,027**

(22) Filed: **Mar. 22, 2023**

Related U.S. Application Data

(63) Continuation-in-part of application No. 17/733,874, filed on Apr. 29, 2022.

(60) Provisional application No. 63/182,558, filed on Apr. 30, 2021.

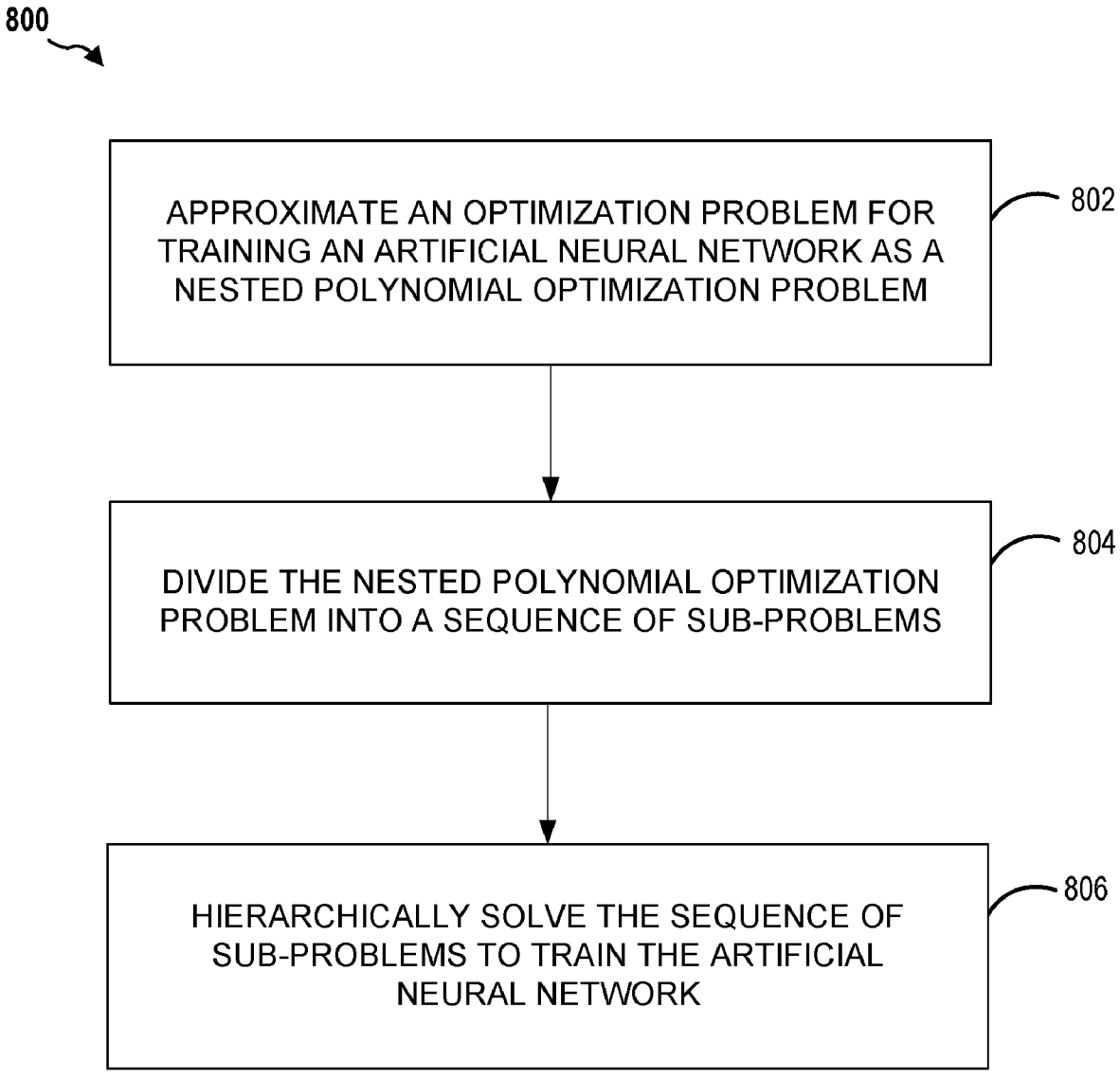
Publication Classification

(51) **Int. Cl.**
G06N 3/08 (2006.01)

(52) **U.S. Cl.**
CPC ***G06N 3/08*** (2013.01)

(57) **ABSTRACT**

A processor-implemented method includes approximating an optimization problem for training an artificial neural network as a nested polynomial optimization problem. The method also includes dividing the nested polynomial optimization problem into a sequence of sub-problems. The method further includes hierarchically solving the sequence of sub-problems to train the artificial neural network.



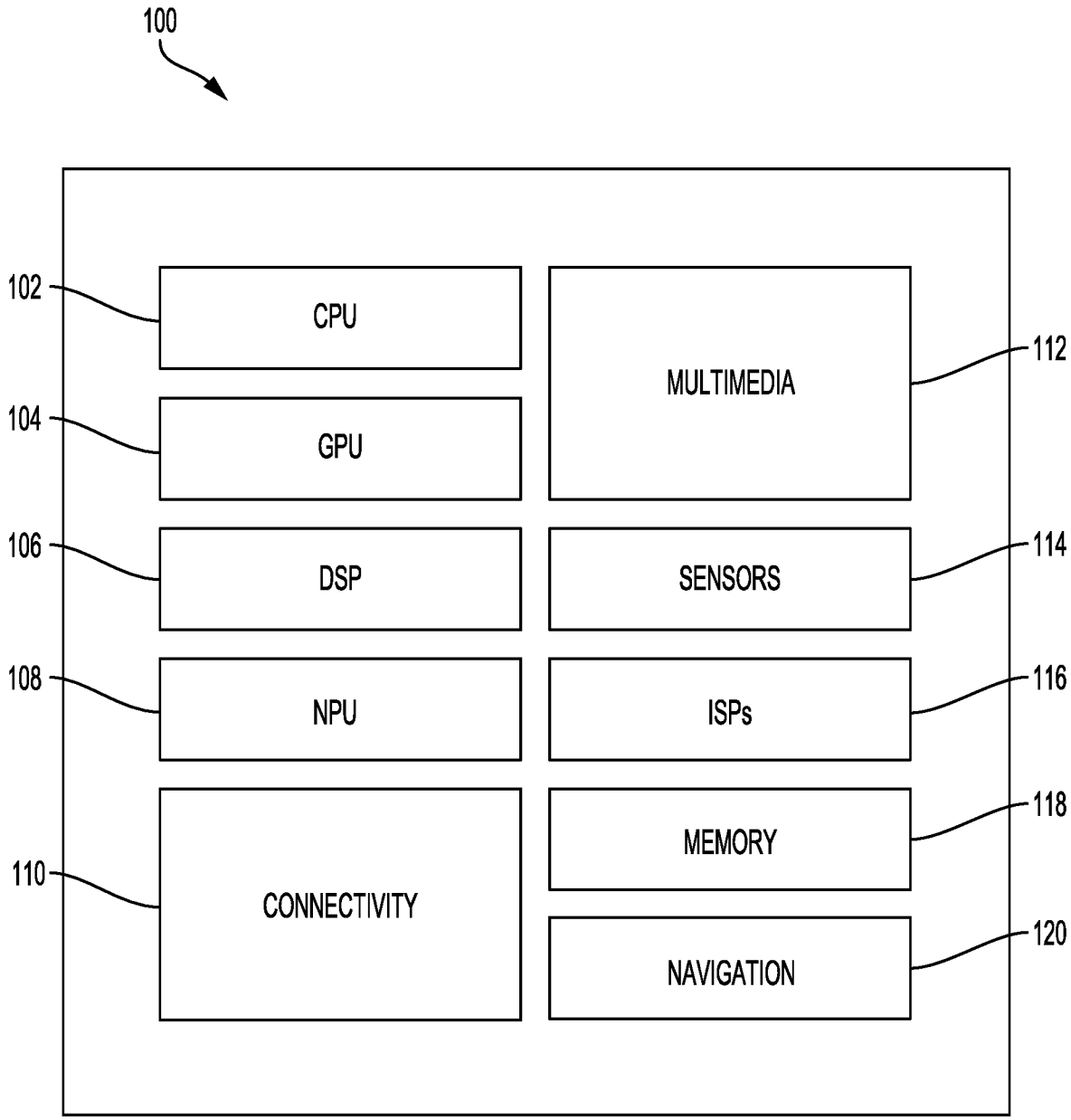

100
102
CPU
104
GPU
106
DSP
108
NPU
110
CONNECTIVITY
MULTIMEDIA
112
SENSORS
114
ISPs
116
MEMORY
118
NAVIGATION
120

FIG. 1

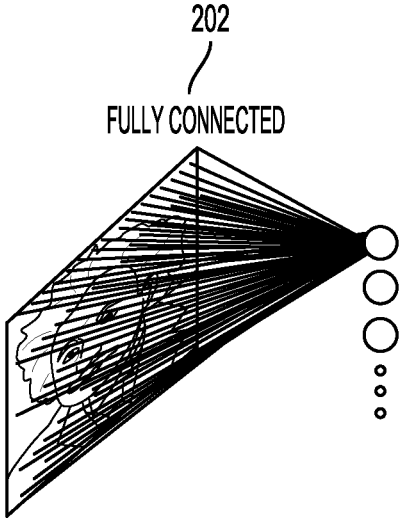


FIG. 2A

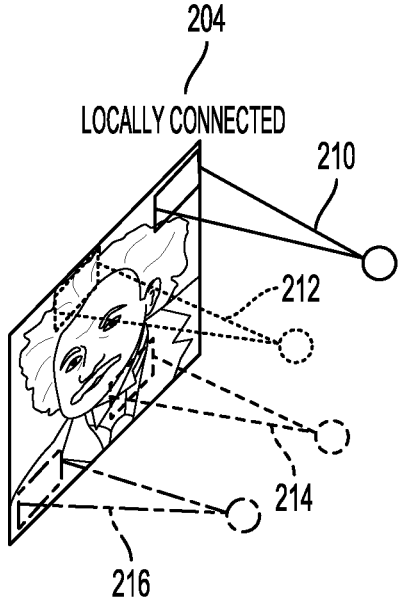


FIG. 2B

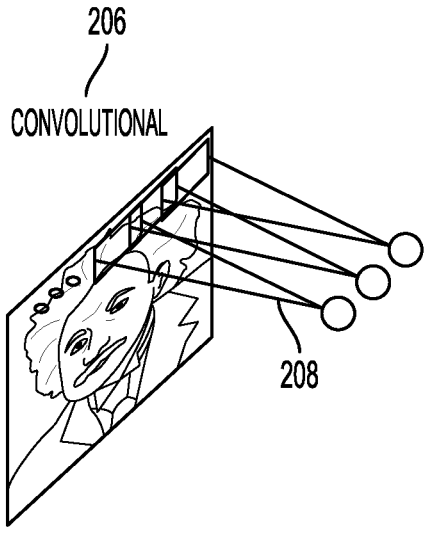


FIG. 2C

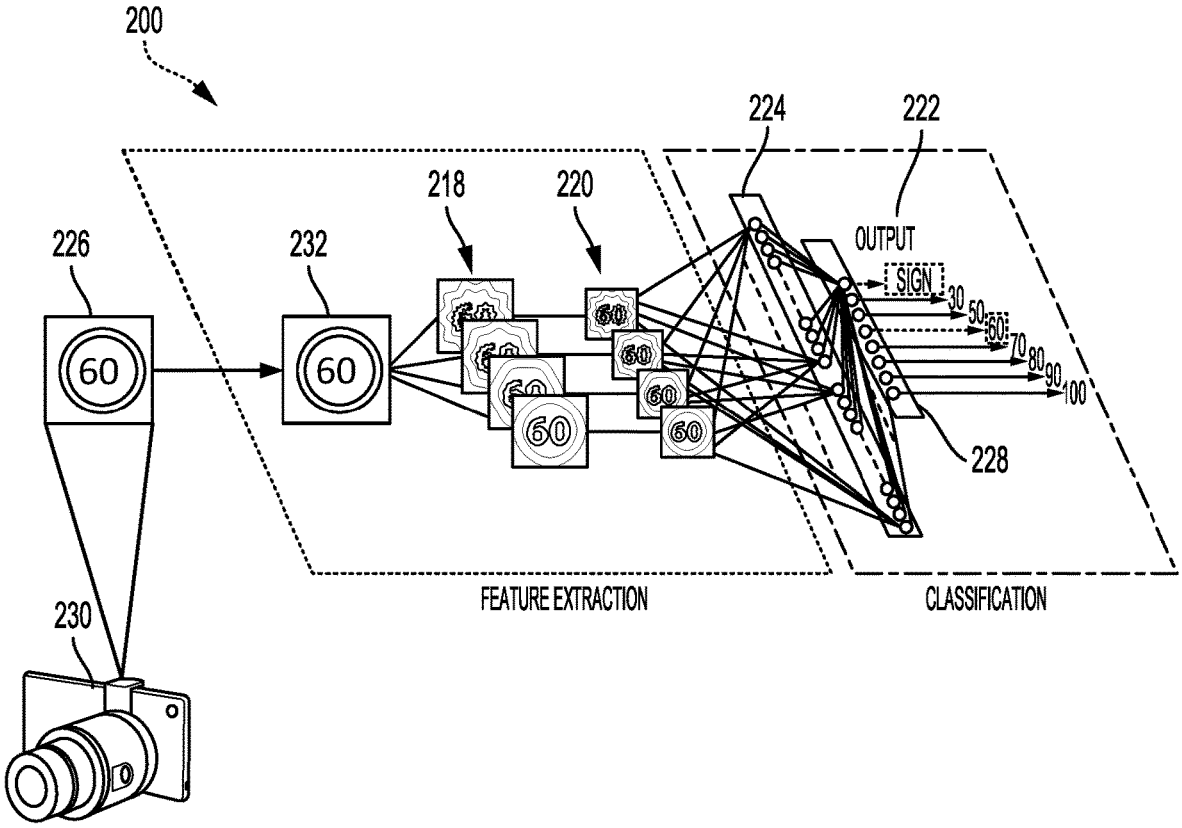

200
224
222
218
220
226
232
OUTPUT
SIGN
30
50
60
70
80
90
100
228
230
FEATURE EXTRACTION
CLASSIFICATION

FIG. 2D

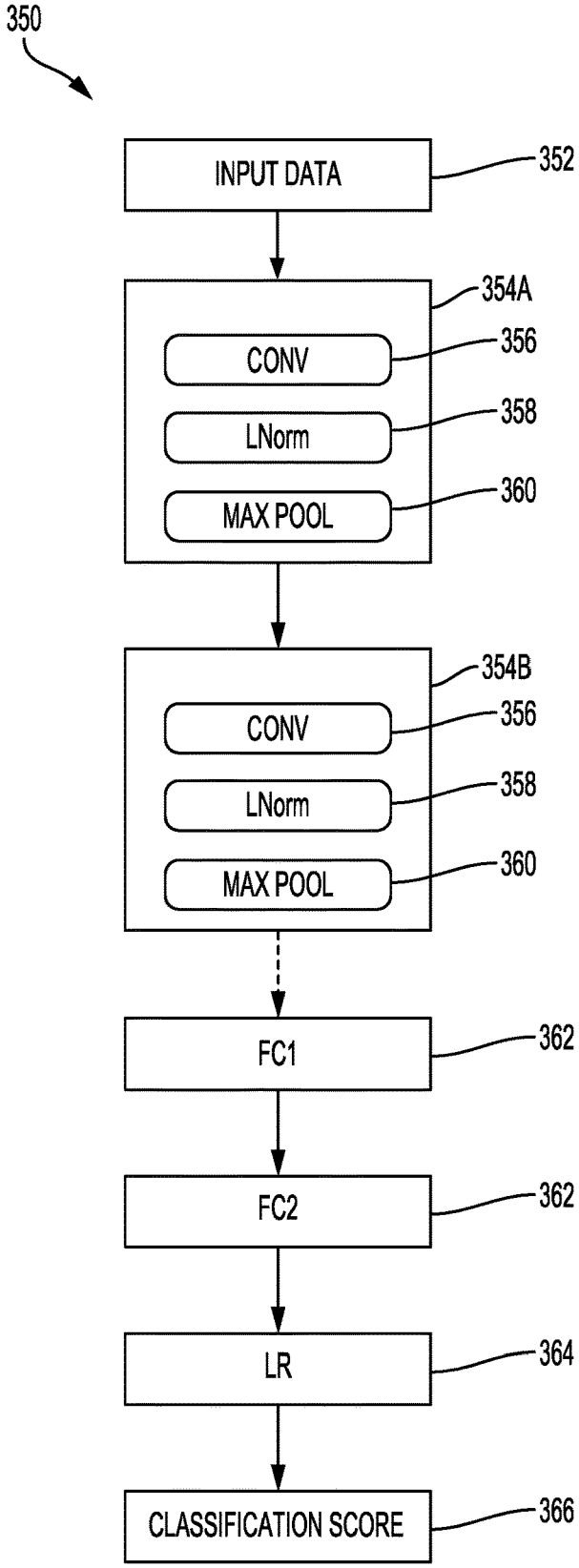

350
INPUT DATA
352
354A
CONV
356
LNorm
358
MAX POOL
360
354B
CONV
356
LNorm
358
MAX POOL
360
FC1
362
FC2
362
LR
364
CLASSIFICATION SCORE
366

FIG. 3

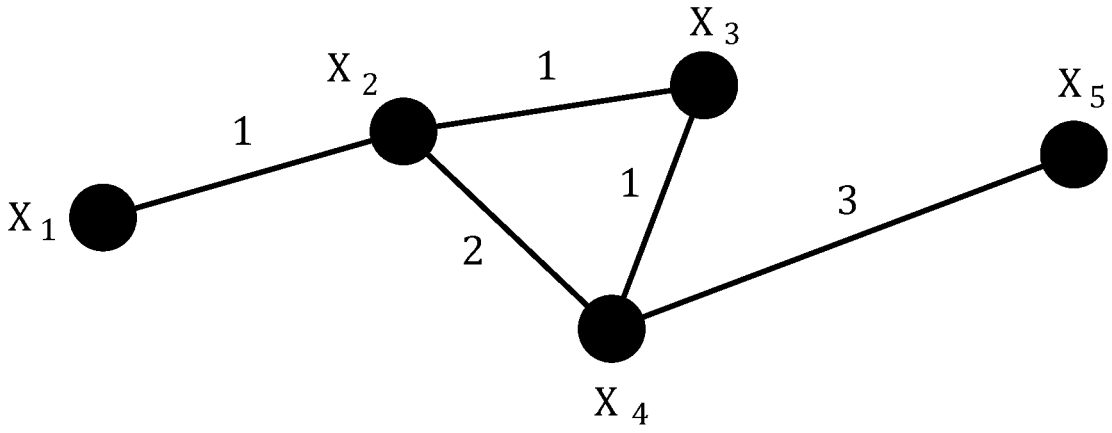


FIG. 4

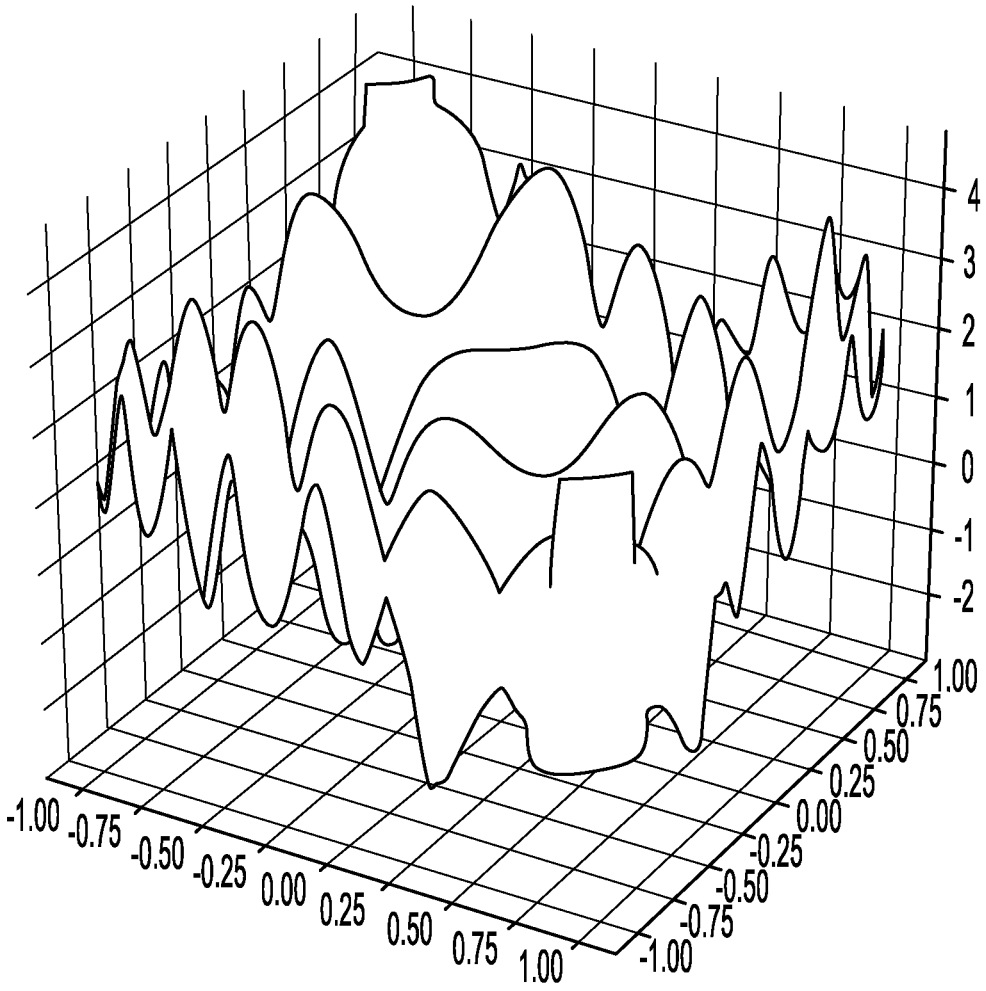

4
3
2
1
0
-1
-2
-1.00
-0.75
-0.50
-0.25
0.00
0.25
0.50
0.75
1.00
-1.00
-0.75
-0.50
-0.25
0.00
0.25
0.50
0.75
1.00

FIG. 5

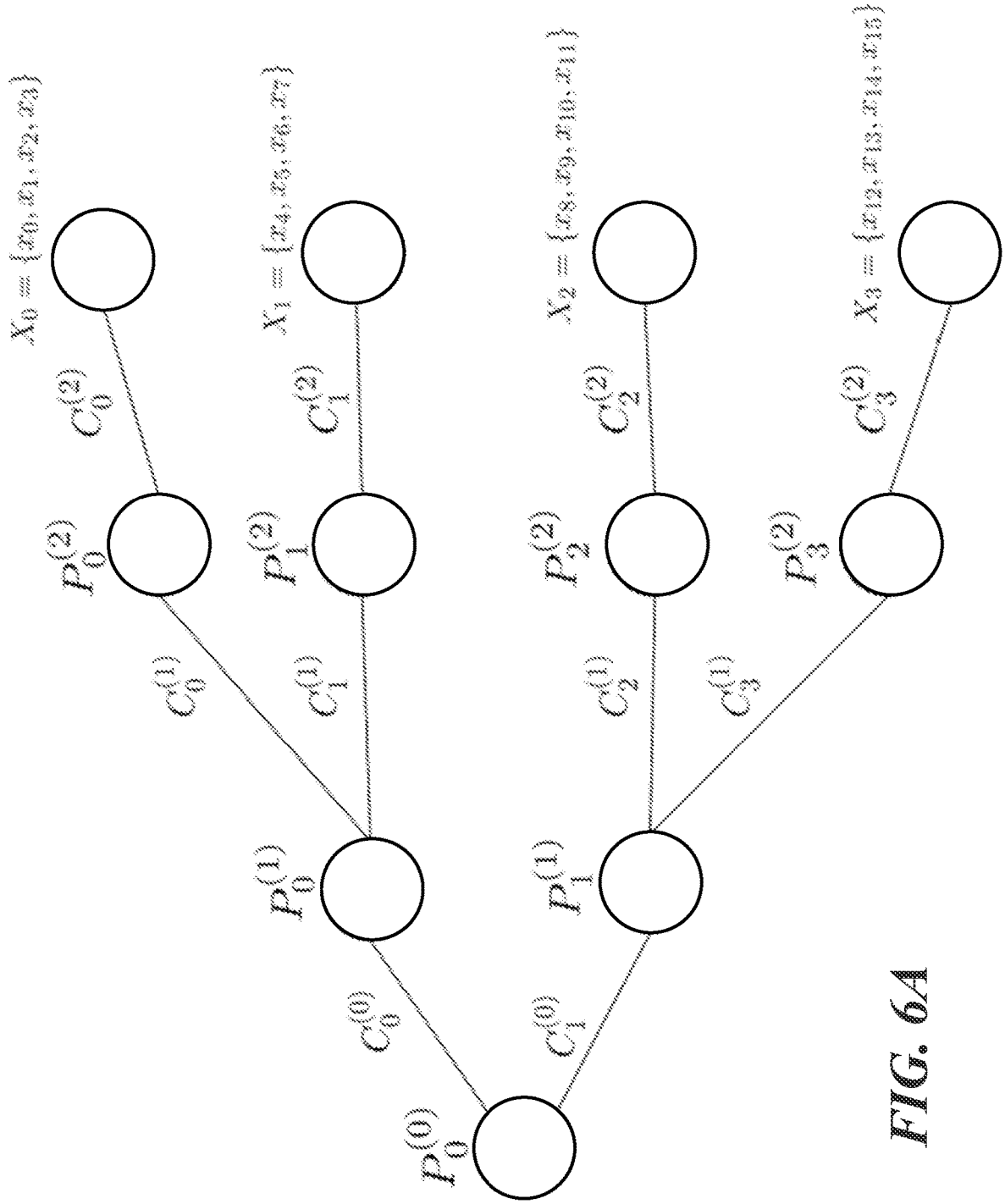

$X_0 = \{x_0, x_1, x_2, x_3\}$
$X_1 = \{x_4, x_5, x_6, x_7\}$
$X_2 = \{x_8, x_9, x_{10}, x_{11}\}$
$X_3 = \{x_{12}, x_{13}, x_{14}, x_{15}\}$
$P_0^{(2)}$
$P_1^{(2)}$
$P_2^{(2)}$
$P_3^{(2)}$
$P_0^{(1)}$
$P_1^{(1)}$
$P_0^{(0)}$
$C_0^{(2)}$
$C_1^{(2)}$
$C_2^{(2)}$
$C_3^{(2)}$
$C_0^{(1)}$
$C_1^{(1)}$
$C_2^{(1)}$
$C_3^{(1)}$
$C_0^{(0)}$
$C_1^{(0)}$

FIG. 6A

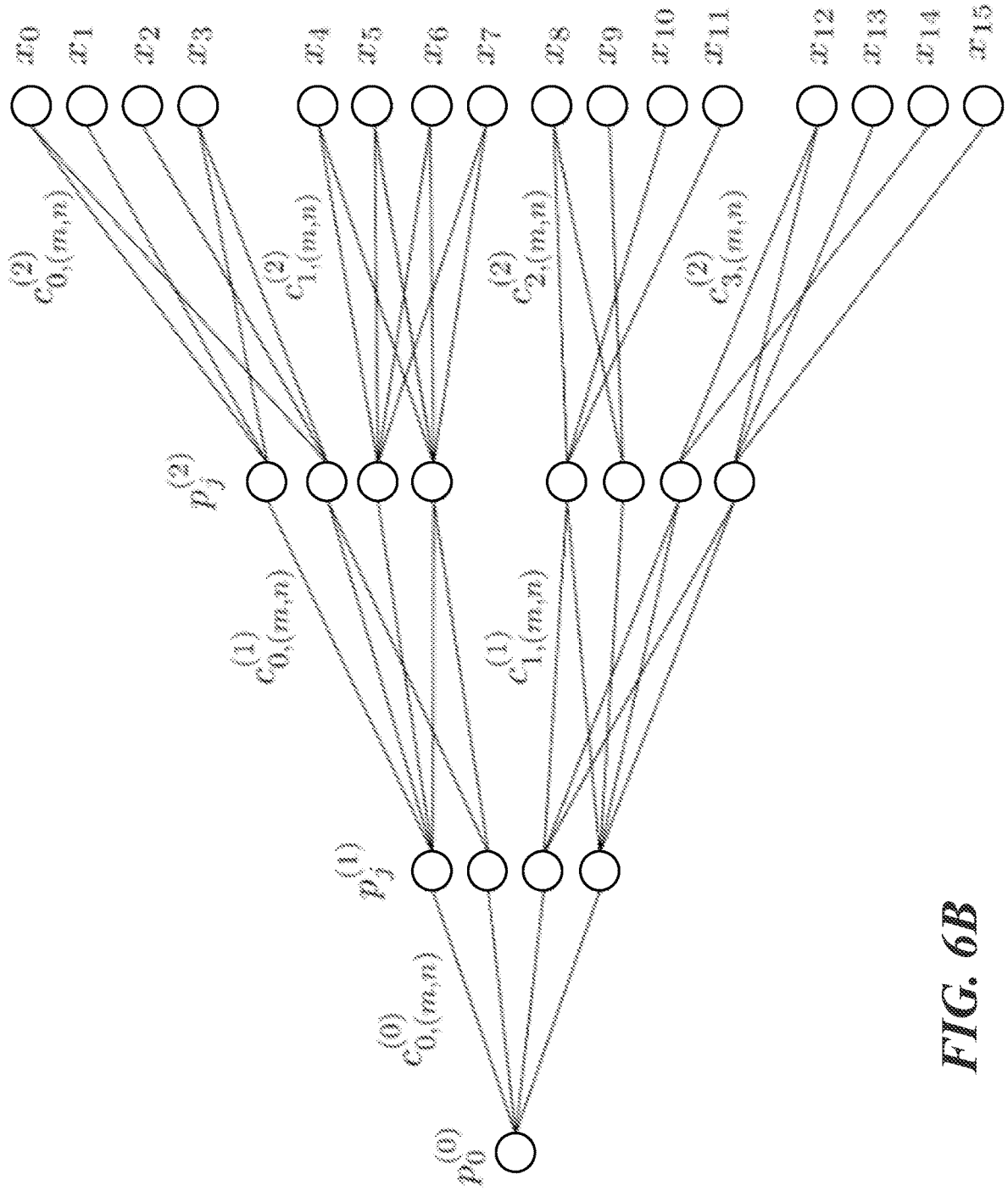

x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15}
$c^{(2)}_{0,(m,n)}$
$c^{(2)}_{1,(m,n)}$
$c^{(2)}_{2,(m,n)}$
$c^{(2)}_{3,(m,n)}$
$p^{(2)}_j$
$c^{(1)}_{0,(m,n)}$
$c^{(1)}_{1,(m,n)}$
$p^{(1)}_j$
$c^{(0)}_{0,(m,n)}$
$p^{(0)}_0$

FIG. 6B

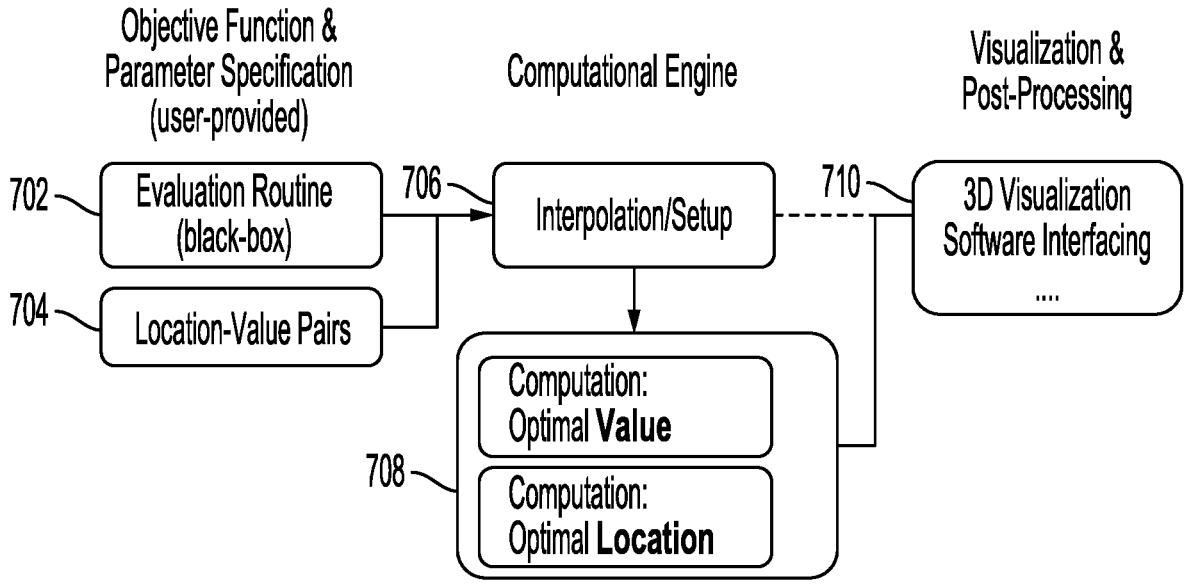

Objective Function & Parameter Specification (user-provided)
Computational Engine
Visualization & Post-Processing
702
Evaluation Routine (black-box)
704
Location-Value Pairs
706
Interpolation/Setup
710
3D Visualization Software Interfacing
....
708
Computation: Optimal Value
Computation: Optimal Location

FIG. 7

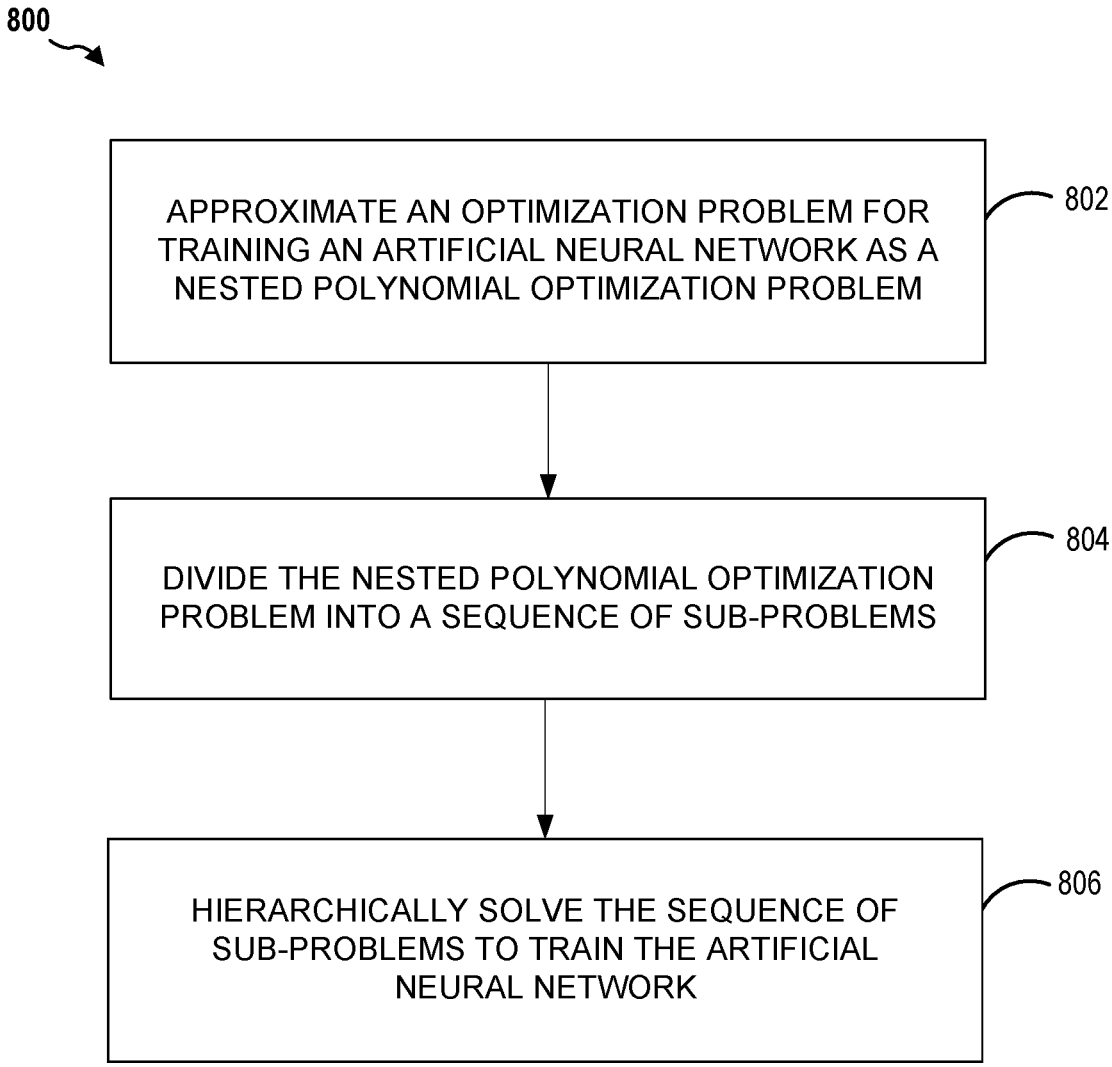

800
APPROXIMATE AN OPTIMIZATION PROBLEM FOR TRAINING AN ARTIFICIAL NEURAL NETWORK AS A NESTED POLYNOMIAL OPTIMIZATION PROBLEM
802
DIVIDE THE NESTED POLYNOMIAL OPTIMIZATION PROBLEM INTO A SEQUENCE OF SUB-PROBLEMS
804
HIERARCHICALLY SOLVE THE SEQUENCE OF SUB-PROBLEMS TO TRAIN THE ARTIFICIAL NEURAL NETWORK
806

FIG. 8

TRAINING OF NEURAL NETWORK WITH POLYNOMIAL SOLVER

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application is a continuation-in-part application of U.S. patent application Ser. No. 17/733,874, filed on Apr. 29, 2022, and titled "FAST-MULTIDIMENSIONAL GLOBAL POLYNOMIAL SOLVER (FM-GPS)," which claims the benefit of U.S. Provisional Patent Application No. 63/182,558, filed on Apr. 30, 2021, and titled "OPTIMIZATION TECHNIQUES, FM-GPS, PARFAITE, sMFFT," the disclosures of which are expressly incorporated by reference in their entireties.

FIELD OF THE DISCLOSURE

[0002] Aspects of the present disclosure generally relate to optimization techniques for machine learning, such as a fast-multidimensional global polynomial solver (FM-GPS) to improve training of neural networks.

BACKGROUND

[0003] Artificial neural networks may be comprised of interconnected groups of artificial neurons (e.g., neuron models). The artificial neural network may be a computational device or be represented as a method to be performed by a computational device. Convolutional neural networks are a type of feed-forward artificial neural network. Convolutional neural networks may include collections of neurons that each have a receptive field and that collectively tile an input space. Convolutional neural networks (CNNs), such as deep convolutional neural networks (DCNs), have numerous applications. In particular, these neural network architectures are used in various technologies, such as machine learning, artificial intelligence, training, image recognition, speech recognition, acoustic scene classification, keyword spotting, autonomous driving, and other classification tasks.

[0004] Training large, deep neural networks is time consuming and computationally expensive. Training may include a large-scale, highly non-convex optimization problem for minimizing loss. Existing techniques for solving the optimization problems offer a disadvantageous trade-off between guarantees of global optimality and scalability. It would be desirable to have a high-performance technique for efficiently solving large-scale multidimensional global, non-convex optimization problems to improve neural network training.

SUMMARY

[0005] In aspects of the present disclosure, a processor-implemented method includes approximating an optimization problem for training an artificial neural network as a nested polynomial optimization problem. The method also includes dividing the nested polynomial optimization problem into a sequence of sub-problems. The method further includes hierarchically solving the sequence of sub-problems to train the artificial neural network.

[0006] Other aspects of the present disclosure are directed to an apparatus. The apparatus has a memory and one or more processors coupled to the memory. The processor(s) is configured to approximate an optimization problem for training an artificial neural network as a nested polynomial optimization problem. The processor(s) is also configured to divide the nested polynomial optimization problem into a sequence of sub-problems. The processor(s) is further configured to hierarchically solve the sequence of sub-problems to train the artificial neural network.

[0007] Other aspects of the present disclosure are directed to an apparatus. The apparatus includes means for approximating an optimization problem for training an artificial neural network as a nested polynomial optimization problem. The apparatus also includes means for dividing the nested polynomial optimization problem into a sequence of sub-problems. The apparatus further includes means for hierarchically solving the sequence of sub-problems to train the artificial neural network.

[0008] In other aspects of the present disclosure, a non-transitory computer-readable medium with program code recorded thereon is disclosed. The program code is executed by a processor and includes program code to approximate an optimization problem for training an artificial neural network as a nested polynomial optimization problem. The program code further includes program code to divide the nested polynomial optimization problem into a sequence of sub-problems. The program code still further includes program code to hierarchically solve the sequence of sub-problems to train the artificial neural network.

[0009] Additional features and advantages of the disclosure will be described below. It should be appreciated by those skilled in the art that this disclosure may be readily utilized as a basis for modifying or designing other structures for carrying out the same purposes of the present disclosure. It should also be realized by those skilled in the art that such equivalent constructions do not depart from the teachings of the disclosure as set forth in the appended claims. The novel features, which are believed to be characteristic of the disclosure, both as to its organization and method of operation, together with further objects and advantages, will be better understood from the following description when considered in connection with the accompanying figures. It is to be expressly understood, however, that each of the figures is provided for the purpose of illustration and description only and is not intended as a definition of the limits of the present disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The features, nature, and advantages of the present disclosure will become more apparent from the detailed description set forth below when taken in conjunction with the drawings in which like reference characters identify correspondingly throughout.

[0011] FIG. **1** illustrates an example implementation of a neural network using a system-on-a-chip (SOC), including a general-purpose processor in accordance with certain aspects of the present disclosure.

[0012] FIGS. **2**A, **2**B, and **2**C are diagrams illustrating a neural network in accordance with aspects of the present disclosure.

[0013] FIG. **2**D is a diagram illustrating an exemplary deep convolutional network (DCN) in accordance with aspects of the present disclosure.

[0014] FIG. **3** is a block diagram illustrating an exemplary deep convolutional network (DCN) in accordance with aspects of the present disclosure.

[0015] FIG. **4** illustrates a graph with respect to a max-cut problem.

[0016] FIG. **5** illustrates a three-dimensional view of a global search space with multiple local minima.

[0017] FIGS. **6**A and **6**B illustrate examples of hierarchal polynomials, in accordance with aspects of the present disclosure.

[0018] FIG. **7** is a block diagram illustrating an example fast-multidimensional global polynomial solver (FM-GPS) processor-based system, in accordance with aspects of the present disclosure.

[0019] FIG. **8** illustrates a method for training a neural network, in accordance with aspects of the present disclosure.

DETAILED DESCRIPTION

[0020] The detailed description set forth below, in connection with the appended drawings, is intended as a description of various configurations and is not intended to represent the only configurations in which the concepts described may be practiced. The detailed description includes specific details for the purpose of providing a thorough understanding of the various concepts. However, it will be apparent to those skilled in the art that these concepts may be practiced without these specific details. In some instances, well-known structures and components are shown in block diagram form in order to avoid obscuring such concepts.

[0021] Based on the teachings, one skilled in the art should appreciate that the scope of the disclosure is intended to cover any aspect of the disclosure, whether implemented independently of or combined with any other aspect of the disclosure. For example, an apparatus may be implemented or a method may be practiced using any number of the aspects set forth. In addition, the scope of the disclosure is intended to cover such an apparatus or method practiced using other structure, functionality, or structure and functionality in addition to or other than the various aspects of the disclosure set forth. It should be understood that any aspect of the disclosure disclosed may be embodied by one or more elements of a claim.

[0022] The word "exemplary" is used to mean "serving as an example, instance, or illustration." Any aspect described as "exemplary" is not necessarily to be construed as preferred or advantageous over other aspects.

[0023] Although particular aspects are described, many variations and permutations of these aspects fall within the scope of the disclosure. Although some benefits and advantages of the preferred aspects are mentioned, the scope of the disclosure is not intended to be limited to particular benefits, uses or objectives. Rather, aspects of the disclosure are intended to be broadly applicable to different technologies, system configurations, networks and protocols, some of which are illustrated by way of example in the figures and in the following description of the preferred aspects. The detailed description and drawings are merely illustrative of the disclosure rather than limiting, the scope of the disclosure being defined by the appended claims and equivalents thereof.

[0024] Neural networks, as well as the training of the neural networks, may be modeled as polynomials. These polynomials can be formulated as computational problems in the form of global, non-convex optimizations. Existing techniques for solving these optimization problems offer disadvantageous trade-offs between guarantees of global optimality (e.g., finding a global optimum) and scalability.

[0025] Aspects of the present disclosure introduce a fast-multidimensional global polynomial solver (FM-GPS) that performs semi-definite convex relaxations of constrained polynomial optimization problems, as seen in the following equation:

$$\min_{\{\mu_n\}} p_n \mu_n$$

subject to

$$M(\mu_n) \geq 0,$$

where $\{p_N\}$ are the

$$\binom{D+d}{d}$$

polynomial coefficients or me degree-d polynomial p(x) in D dimensions (where

$$\binom{D+d}{d}$$

means "D+d choose d" combinations), $M(\mu\mu_n)$ is a (moment) matrix with explicit form specific to the semi-definite program (SDP) relaxation and ≥ 0 indicates positive-definiteness. The matrix $M(\mu_n)$ is an N×N, highly-structured matrix with N degrees of freedom (rather than N^2 as in the general case). The size of N dictates the cost of solving the SDP (and by extension, the cost of solving the original non-convex optimization problem). The FM-GPS may be employed to train artificial neural networks by optimizing nested polynomials. The layers of a hierarchical structure may be 'peeled' to solve a sequence of polynomial optimization problems in order to train the neural network.

[0026] The FM-GPS employs a novel algebraic sampling of the optimization landscape followed by a convex relaxation. The technique uses relaxation approaches that enable low-complexity sampling to provide information about the location and value of the global optimum while efficiently performing a typically computationally-prohibitive full exploration of the entire space. The FM-GPS exploits structural properties unique to SDP solver relaxations to produce a scalable SDP and global non-convex optimization solver. Exploitable structures include a sparsity and hierarchical structure, a block Hankel-plus-Toeplitz structure, a low-rank structure, and an underlying geometry structure. The solving may further include dimensionality reduction, and/or using a warm start.

[0027] FIG. **1** illustrates an example implementation of a system-on-a-chip (SOC) **100**, which may include a central processing unit (CPU) **102** or a multi-core CPU configured for optimization techniques for machine learning. Variables (e.g., neural signals and synaptic weights), system parameters associated with a computational device (e.g., neural network with weights), delays, frequency bin information, and task information may be stored in a memory block associated with a neural processing unit (NPU) **108**, in a memory block associated with a CPU **102**, in a memory block associated with a graphics processing unit (GPU) **104**,

in a memory block associated with a digital signal processor (DSP) **106**, in a memory block **118**, or may be distributed across multiple blocks. Instructions executed at the CPU **102** may be loaded from a program memory associated with the CPU **102** or may be loaded from a memory block **118**.

[0028] The SOC **100** may also include additional processing blocks tailored to specific functions, such as a GPU **104**, a DSP **106**, a connectivity block **110**, which may include fifth generation (5G) connectivity, fourth generation long term evolution (4G LTE) connectivity, Wi-Fi connectivity, USB connectivity, Bluetooth connectivity, and the like, and a multimedia processor **112** that may, for example, detect and recognize gestures. In one implementation, the NPU **108** is implemented in the CPU **102**, DSP **106**, and/or GPU **104**. The SOC **100** may also include a sensor processor **114**, image signal processors (ISPs) **116**, and/or navigation module **120**, which may include a global positioning system.

[0029] The SOC **100** may be based on an ARM instruction set. In aspects of the present disclosure, the instructions loaded into the general-purpose processor **102** may include code to approximate an optimization problem for training an artificial neural network as a nested polynomial optimization problem. The instructions loaded into the general-purpose processor **102** may also include code to divide the nested polynomial optimization problem into a sequence of sub-problems. The instructions loaded into the general-purpose processor **102** may further include code to hierarchically solve the sequence of sub-problems to train the artificial neural network.

[0030] Deep learning architectures may perform an object recognition task by learning to represent inputs at successively higher levels of abstraction in each layer, thereby building up a useful feature representation of the input data. In this way, deep learning addresses a major bottleneck of traditional machine learning. Prior to the advent of deep learning, a machine learning approach to an object recognition problem may have relied heavily on human engineered features, perhaps in combination with a shallow classifier. A shallow classifier may be a two-class linear classifier, for example, in which a weighted sum of the feature vector components may be compared with a threshold to predict to which class the input belongs. Human engineered features may be templates or kernels tailored to a specific problem domain by engineers with domain expertise. Deep learning architectures, in contrast, may learn to represent features that are similar to what a human engineer might design, but through training. Furthermore, a deep network may learn to represent and recognize new types of features that a human might not have considered.

[0031] A deep learning architecture may learn a hierarchy of features. If presented with visual data, for example, the first layer may learn to recognize relatively simple features, such as edges, in the input stream. In another example, if presented with auditory data, the first layer may learn to recognize spectral power in specific frequencies. The second layer, taking the output of the first layer as input, may learn to recognize combinations of features, such as simple shapes for visual data or combinations of sounds for auditory data. For instance, higher layers may learn to represent complex shapes in visual data or words in auditory data. Still higher layers may learn to recognize common visual objects or spoken phrases.

[0032] Deep learning architectures may perform especially well when applied to problems that have a natural hierarchical structure. For example, the classification of motorized vehicles may benefit from first learning to recognize wheels, windshields, and other features. These features may be combined at higher layers in different ways to recognize cars, trucks, and airplanes.

[0033] Neural networks may be designed with a variety of connectivity patterns. In feed-forward networks, information is passed from lower to higher layers, with each neuron in a given layer communicating to neurons in higher layers. A hierarchical representation may be built up in successive layers of a feed-forward network, as described above. Neural networks may also have recurrent or feedback (also called top-down) connections. In a recurrent connection, the output from a neuron in a given layer may be communicated to another neuron in the same layer. A recurrent architecture may be helpful in recognizing patterns that span more than one of the input data chunks that are delivered to the neural network in a sequence. A connection from a neuron in a given layer to a neuron in a lower layer is called a feedback (or top-down) connection. A network with many feedback connections may be helpful when the recognition of a high-level concept may aid in discriminating the particular low-level features of an input.

[0034] The connections between layers of a neural network may be fully connected or locally connected. FIG. **2**A illustrates an example of a fully connected neural network **202**. In a fully connected neural network **202**, a neuron in a first layer may communicate its output to every neuron in a second layer, so that each neuron in the second layer will receive input from every neuron in the first layer. FIG. **2**B illustrates an example of a locally connected neural network **204**. In a locally connected neural network **204**, a neuron in a first layer may be connected to a limited number of neurons in the second layer. More generally, a locally connected layer of the locally connected neural network **204** may be configured so that each neuron in a layer will have the same or a similar connectivity pattern, but with connections strengths that may have different values (e.g., **210**, **212**, **214**, and **216**). The locally connected connectivity pattern may give rise to spatially distinct receptive fields in a higher layer, because the higher layer neurons in a given region may receive inputs that are tuned through training to the properties of a restricted portion of the total input to the network.

[0035] One example of a locally connected neural network is a convolutional neural network. FIG. **2**C illustrates an example of a convolutional neural network **206**. The convolutional neural network **206** may be configured such that the connection strengths associated with the inputs for each neuron in the second layer are shared (e.g., **208**). Convolutional neural networks may be well suited to problems in which the spatial location of inputs is meaningful.

[0036] One type of convolutional neural network is a deep convolutional network (DCN). FIG. **2**D illustrates a detailed example of a DCN **200** designed to recognize visual features from an image **226** input from an image capturing device **230**, such as a car-mounted camera. The DCN **200** of the current example may be trained to identify traffic signs and a number provided on the traffic sign. Of course, the DCN **200** may be trained for other tasks, such as identifying lane markings or identifying traffic lights.

[0037] The DCN **200** may be trained with supervised learning. During training, the DCN **200** may be presented with an image, such as the image **226** of a speed limit sign,

and a forward pass may then be computed to produce an output **222**. The DCN **200** may include a feature extraction section and a classification section. Upon receiving the image **226**, a convolutional layer **232** may apply convolutional kernels (not shown) to the image **226** to generate a first set of feature maps **218**. As an example, the convolutional kernel for the convolutional layer **232** may be a 5×5 kernel that generates 28×28 feature maps. In the present example, because four different feature maps are generated in the first set of feature maps **218**, four different convolutional kernels were applied to the image **226** at the convolutional layer **232**. The convolutional kernels may also be referred to as filters or convolutional filters.

[0038] The first set of feature maps **218** may be subsampled by a max pooling layer (not shown) to generate a second set of feature maps **220**. The max pooling layer reduces the size of the first set of feature maps **218**. That is, a size of the second set of feature maps **220**, such as 14×14, is less than the size of the first set of feature maps **218**, such as 28×28. The reduced size provides similar information to a subsequent layer while reducing memory consumption. The second set of feature maps **220** may be further convolved via one or more subsequent convolutional layers (not shown) to generate one or more subsequent sets of feature maps (not shown).

[0039] In the example of FIG. **2**D, the second set of feature maps **220** is convolved to generate a first feature vector **224**. Furthermore, the first feature vector **224** is further convolved to generate a second feature vector **228**. Each feature of the second feature vector **228** may include a number that corresponds to a possible feature of the image **226**, such as "sign," "60," and "100." A softmax function (not shown) may convert the numbers in the second feature vector **228** to a probability. As such, an output **222** of the DCN **200** is a probability of the image **226** including one or more features.

[0040] In the present example, the probabilities in the output **222** for "sign" and "60" are higher than the probabilities of the others of the output **222**, such as "30," "40," "50," "70," "80," "90," and "100". Before training, the output **222** produced by the DCN **200** is likely to be incorrect. Thus, an error may be calculated between the output **222** and a target output. The target output is the ground truth of the image **226** (e.g., "sign" and "60"). The weights of the DCN **200** may then be adjusted so the output **222** of the DCN **200** is more closely aligned with the target output.

[0041] To adjust the weights, a learning algorithm may compute a gradient vector for the weights. The gradient may indicate an amount that an error would increase or decrease if the weight were adjusted. At the top layer, the gradient may correspond directly to the value of a weight connecting an activated neuron in the penultimate layer and a neuron in the output layer. In lower layers, the gradient may depend on the value of the weights and on the computed error gradients of the higher layers. The weights may then be adjusted to reduce the error. This manner of adjusting the weights may be referred to as "back propagation" as it involves a "backward pass" through the neural network.

[0042] In practice, the error gradient of weights may be calculated over a small number of examples, so that the calculated gradient approximates the true error gradient. This approximation method may be referred to as stochastic gradient descent. Stochastic gradient descent may be repeated until the achievable error rate of the entire system has stopped decreasing or until the error rate has reached a target level. After learning, the DCN may be presented with new images and a forward pass through the network may yield an output **222** that may be considered an inference or a prediction of the DCN.

[0043] Deep belief networks (DBNs) are probabilistic models comprising multiple layers of hidden nodes. DBNs may be used to extract a hierarchical representation of training data sets. A DBN may be obtained by stacking up layers of Restricted Boltzmann Machines (RBMs). An RBM is a type of artificial neural network that can learn a probability distribution over a set of inputs. Because RBMs can learn a probability distribution in the absence of information about the class to which each input should be categorized, RBMs are often used in unsupervised learning. Using a hybrid unsupervised and supervised paradigm, the bottom RBMs of a DBN may be trained in an unsupervised manner and may serve as feature extractors, and the top RBM may be trained in a supervised manner (on a joint distribution of inputs from the previous layer and target classes) and may serve as a classifier.

[0044] Deep convolutional networks (DCNs) are networks of convolutional networks, configured with additional pooling and normalization layers. DCNs have achieved state-of-the-art performance on many tasks. DCNs can be trained using supervised learning in which both the input and output targets are known for many exemplars and are used to modify the weights of the network by use of gradient descent methods.

[0045] DCNs may be feed-forward networks. In addition, as described above, the connections from a neuron in a first layer of a DCN to a group of neurons in the next higher layer are shared across the neurons in the first layer. The feed-forward and shared connections of DCNs may be exploited for fast processing. The computational burden of a DCN may be much less, for example, than that of a similarly sized neural network that comprises recurrent or feedback connections.

[0046] The processing of each layer of a convolutional network may be considered a spatially invariant template or basis projection. If the input is first decomposed into multiple channels, such as the red, green, and blue channels of a color image, then the convolutional network trained on that input may be considered three-dimensional, with two spatial dimensions along the axes of the image and a third dimension capturing color information. The outputs of the convolutional connections may be considered to form a feature map in the subsequent layer, with each element of the feature map (e.g., **220**) receiving input from a range of neurons in the previous layer (e.g., feature maps **218**) and from each of the multiple channels. The values in the feature map may be further processed with a non-linearity, such as a rectification, max(0, x). Values from adjacent neurons may be further pooled, which corresponds to down sampling, and may provide additional local invariance and dimensionality reduction. Normalization, which corresponds to whitening, may also be applied through lateral inhibition between neurons in the feature map.

[0047] The performance of deep learning architectures may increase as more labeled data points become available or as computational power increases. Modem deep neural networks are routinely trained with computing resources that are thousands of times greater than what was available to a typical researcher just fifteen years ago. New architectures

and training paradigms may further boost the performance of deep learning. Rectified linear units may reduce a training issue known as vanishing gradients. New training techniques may reduce over-fitting and thus enable larger models to achieve better generalization. Encapsulation techniques may abstract data in a given receptive field and further boost overall performance.

[0048] FIG. **3** is a block diagram illustrating a deep convolutional network **350**. The deep convolutional network **350** may include multiple different types of layers based on connectivity and weight sharing. As shown in FIG. **3**, the deep convolutional network **350** includes the convolution blocks **354**A, **354**B. Each of the convolution blocks **354**A, **354**B may be configured with a convolution layer (CONV) **356**, a normalization layer (LNorm) **358**, and a max pooling layer (MAX POOL) **360**.

[0049] The convolution layers **356** may include one or more convolutional filters, which may be applied to the input data to generate a feature map. Although only two of the convolution blocks **354**A, **354**B are shown, the present disclosure is not so limiting, and instead, any number of the convolution blocks **354**A, **354**B may be included in the deep convolutional network **350** according to design preference. The normalization layer **358** may normalize the output of the convolution filters. For example, the normalization layer **358** may provide whitening or lateral inhibition. The max pooling layer **360** may provide down sampling aggregation over space for local invariance and dimensionality reduction.

[0050] The parallel filter banks, for example, of a deep convolutional network may be loaded on a CPU **102** or GPU **104** of an SOC **100** to achieve high performance and low power consumption. In alternative embodiments, the parallel filter banks may be loaded on the DSP **106** or an ISP **116** of an SOC **100**. In addition, the deep convolutional network **350** may access other processing blocks that may be present on the SOC **100**, such as sensor processor **114** and navigation module **120**, dedicated, respectively, to sensors and navigation.

[0051] The deep convolutional network **350** may also include one or more fully connected layers **362** (FC**1** and FC**2**). The deep convolutional network **350** may further include a logistic regression (LR) layer **364**. Between each layer **356**, **358**, **360**, **362**, **364** of the deep convolutional network **350** are weights (not shown) that are to be updated. The output of each of the layers (e.g., **356**, **358**, **360**, **362**, **364**) may serve as an input of a succeeding one of the layers (e.g., **356**, **358**, **360**, **362**, **364**) in the deep convolutional network **350** to learn hierarchical feature representations from input data **352** (e.g., images, audio, video, sensor data and/or other input data) supplied at the first of the convolution blocks **354**A. The output of the deep convolutional network **350** is a classification score **366** for the input data **352**. The classification score **366** may be a set of probabilities, where each probability is the probability of the input data including a feature from a set of features.

[0052] Entire neural networks (or portions of the neural networks) may be modeled as sets of polynomials. These polynomials may be trained using computational problems formulated as global, non-convex optimizations, as seen in the following equation:

$$\min_{x\in\mathbb{R}^D} p(x) \tag{1}$$

[0053] Subject to constraints:

$$q_n(x)\geq 0, n=1,2,\ldots,N$$

$$r_m(x)=0, m=1,2,\ldots,M,$$

where $x\in\mathbb{R}^D$ is the optimization variable, $p(x)=\Sigma_{|n|\leq d} p_n T_n(x)$ is a polynomial objective function ($\{T_n(x)\}$ and d being a polynomial basis and polynomial degree, respectively), and $\{q_n(x)\},\{r_m(x)\}$ are polynomial constraints. Table 1 highlights several examples of non-convex optimization problems, together with their corresponding explicit objectives and constraints.

TABLE 1

Non-Convex Optimization Problem	Objective	Constraints: Inequality	Constraints: Equality
Mixed Integer Linear Program (MILP)	$c^T x$	$a_n^T x + b_n$, x	$\cos(2\pi x_{n_i})$
Quadratically-Constrained Quadratic Program (QCQP)	$x^T A x + b^T x$	$x^T A_n x + b_n^T x + c_n$	$a_n^T x - b_n$
Non-Linear Program (NLP)	f(x) (non-linear)	$q_n(x)$ (non-linear)	$T_m(x)$ (non-linear)

[0054] An example model corresponding to Equation (1) is a max-cut model. The max-cut problem consists into finding a set of edges that separates the vertices of a graph into two disjoint sets (cut) such that the sum of the weights of the selected set of edges is maximized. The max-cut problem can be formulated as an optimization problem of the form,

$$\text{Maximize } \sum_{n,m} w_{n,m}\left(\frac{1-x_n x_m}{2}\right) \tag{2}$$

$$\text{Subject to } x_n \in \{-1, 1\}$$

where $w_{n,m}$ is the weight of an edge connecting vertices x_n and x_m, and $x_n x_m$, may be the product of the values of the vertices.

[0055] FIG. **4** illustrates a graph with respect to a max-cut problem. For the example shown in FIG. **4**, the max-cut problem corresponds to

$$\text{Maximize } \left(\frac{1-x_1x_2}{2}\right)+\left(\frac{1-x_2x_3}{2}\right)+ 2\left(\frac{1-x_2x_4}{2}\right)+\left(\frac{1-x_3x_4}{2}\right)+3\left(\frac{1-x_4x_5}{2}\right) \tag{3}$$

$$\text{Subject to } x_1, x_2, x_3, x_4, x_5 \in \{-1, 1\}.$$

[0056] The max-cut problem is a discrete optimization problem because the variables are constrained to the values +/−1. However, as with most combinatorial optimization

problem, the max-cut problem can be re-formulated exactly as a polynomial optimization problem by re-writing the constraints as:

$$x_n \in \{-1,1\} \rightarrow x_n^2 - 1 = 0, \quad (4)$$

which corresponds to a polynomial equality constraint, and leads to a polynomial optimization problem because the objective corresponds to a polynomial.

[0057] These problems may be represented in the continuous space for applications, such as artificial intelligence (AI), deep learning, machine learning, scientific computing, engineering design, and optimal control. The problems may be represented in a discrete space for applications, such as logistics and planning, scheduling, and resource allocation. Mixed space applications may include software verification, combinatorial optimization, graph analysis, and protocol certification.

[0058] Global, non-convex optimization problems are global in the sense that a global minimum should be found over the entire search space. The optimization problems are non-convex because local minima may not be global, and they provide no information about the global minima.

[0059] FIG. **5** illustrates a three-dimensional view of a global search space with multiple local minima. FIG. **5** shows the difficulties that may be encountered with finding a global minimum. In the hyper cube of FIG. **5**, the function has two global minima and multiple local minima. The function shown in FIG. **5** is $f(x)=x^2=y^2+\sin(5\pi(x^2+y^2))+\sin(\pi(xy))+\sin(2\pi(xy)^2)$, which is minimized over the hypercube.

[0060] As seen in FIG. **5**, the current methods (e.g., gradient descent) are highly dependent on the starting point, and will only reach local minima. That is, depending on where the process starts, the process can only descend so much without ascending. Such local minima can be highly suboptimal, especially in high dimension.

[0061] Existing techniques for solving these optimization problems offer a disadvantageous trade-off between guarantees of global optimality (e.g., finding a global optimum) and scalability. For example, local and convexity based solutions, such as gradient descent and Newton/quasi-Newton solutions, do not guarantee global optimality, with issues arising at saddle points. These solutions have advantages, such as working with scalable polynomials and being efficient on convex problems. Global probabilistic/deterministic techniques, such as a Monte Carlo Markov chain, simulated annealing/branch and bound techniques, and exhaustive searches, scale exponentially, although they may guarantee global optimality. It would be desirable to have a high-performance technique and novel system for efficiently solving large-scale multidimensional global, non-convex optimization problems.

[0062] Aspects of the present disclosure introduce a fast-multidimensional global polynomial solver (FM-GPS) that performs semi-definite convex relaxations of constrained polynomial optimization problems, as seen in the following equation:

$$\min_{\{\mu_n\}} p_n \mu_n \quad (5)$$

$$\text{subject to } M(\mu_n) \succeq 0$$

where $\{p_n\}$ are the

$$\binom{D+d}{d}$$

polynomial coefficients of the degree-d polynomial p(x) in D dimensions, $M(\mu_n)$ is a (moment) matrix with explicit form specific to the semi-definite program (SDP) relaxation and $\succeq 0$ indicates positive-definiteness. The matrix $M(\rho_n)$ is an N×N, highly-structured matrix with N degrees of freedom (rather than N^2 as in the general case). The size of N dictates the processing cost of solving the SDP (and by extension, the cost of solving the original non-convex optimization problem). It may be sufficient to pick N only slightly larger than the number of coefficients. In some examples, it may be sufficient to pick

$$N = \binom{D + C(\epsilon)^d}{C(\epsilon)^d},$$

where $C_{(\epsilon)}$ is a constant that is independent of dimension, depending only on the desired accuracy ϵ. While existing theoretical estimates are super-exponential in dimension and degree, it is estimated that N scales at most polynomially in dimension. In some aspects, Chebyshev polynomials are used as the basis $\{T_n(x)\}$ for numerical stability and performance. The FM-GPS obtains an optimal value and optimal location recovery, even in the presence of multiple global minima.

[0063] According to aspects of the present disclosure, the FM-GPS employs a novel algebraic sampling of the optimization landscape followed by a convex relaxation. The technique uses relaxation approaches that enable low-complexity sampling to provide information about the location and value of the global optimum while efficiently performing a typically computationally-prohibitive full exploration of the entire space.

[0064] This solution is scalable with provable polynomial scalability. In other words, the FM-GPS possesses provable efficient (e.g., polynomial) computational bounds that are significantly better than existing techniques. The FM-GPS guarantees global optimality. That is, the FM-GPS provides deterministic guarantees of global optimality (e.g., value and location) within a prescribed user-specified error level. This is in contrast to random techniques (e.g., Monte Carlo Markov chain), and non-random techniques that are computationally expensive or limited to local optima.

[0065] The FM-GPS is versatile and broadly applicable. That is, the solution works with a complex non-convex objective and constraints (e.g., discrete sets, disconnected components, non-convex domains), as well as with a continuous objective. The FM-GPS solution is a high-performance, scalable implementation.

[0066] The fast-multidimensional global polynomial solver (FM-GPS) exploits structural properties unique to semi-definite program (SDP) solver relaxations to produce a scalable SDP and global non-convex optimization solver. Exploitable structures include a sparsity and hierarchical structure, a block Hankel-plus-Toeplitz structure, a low-rank structure, and an underlying geometry structure. The solving may further include dimensionality reduction, and/or using a warm start. Of course, other structures may be exploitable, beyond what is listed here.

[0067] With respect to sparsity and hierarchy, many practical problems exhibit chordal sparsity, coefficient sparsity and/or a hierarchical structure (e.g., Neural/Bayesian networks). These structures may be used to reduce the size and cost of enforcing semi-definite constraints. For a block Hankel-plus-Toeplitz structure, the matrix M(p) possesses only N degrees of freedom together with a Hankel-plus-Toeplitz block structure that may be used to significantly speed up the computation of eigenvalue decompositions through fast matrix-vector products (e.g., iterative techniques) and randomized linear algebra. A low-rank structure may be exploited because optimal matrices are rank-1 (e.g., moment matrices of a Dirac delta). Performance may be gained by restricting the search to low-rank points and may accelerate the eigenvalue decomposition. Low-rank-based penalties may also be used to increase convergence rates. Optimal sets may have dimensions bounded by the minimum between N and total number of global optima of the original problem. This characteristic may be used to reduce the dimensionality. Facial reduction may also be used to achieve a similar goal.

[0068] To enable a warm start, the SDP may be approximated to various degrees by linear programs (LPs) and second-order cone programs (SOCPs), either statically or adaptively. Efficient numerical solvers may treat the approximate problems leading to a warm start for the SDP.

[0069] The underlying geometric structure may be exploited because the geometry of convex relaxation SDPs is closely associated with that of the original problem. Such knowledge may be used to inform the solution of the convex relaxation, e.g., large and flat optimal region.

[0070] The fast-multidimensional global polynomial solver (FM-GPS) provides significant improvements to existing applications, such as electronic design automation (EDA), software verification, planning, and cryptography. The FM-GPS enables the solving of very large problems, such as the training of neural networks, millions of times faster with greater accuracy than is currently possible today with existing heuristics. Because problems, such as those in AI and machine learning (ML), are framed as optimization (e.g., optimizing the weights of a deep learning neural network to minimize the error for inferences on a training set), this may result in ML systems that can learn millions of times faster than today's systems. Improvements in optimization can further benefit many other fields, such as engineering design in aerospace.

[0071] The FM-GPS reduces problems in the domain to polynomial optimization, a form in which the global optimum can be rapidly identified. FM-GPS is a high-performance semi-definite program (SDP) solver that provides significant computational cost reductions/speedups, as well as lower memory requirements (e.g., approximately 10 times to 1000 times lower). The FM-GPS is able to perform convex relaxations for matrix completion and recommender systems, sum of squares (SOS) programs for control, and very large scale integration (VLSI) design. The FM-GPS offers high-fidelity and globally optimal solutions, including sparse deep neural networks, high-performance graph algorithms, and guaranteed optimal control strategies and optimal design. The FM-GPS provides discrete optimization, orders-of-magnitude beyond existing scales for integer and mixed integer programming, logistics, planning, and scheduling. The FM-GPS provides an efficient approximation of non-deterministic polynomial (NP)-hard problems, such as quadratically-constrained quadratic programs (QCQP), and discrete optimization, such as combinatorial optimization and graph problems (e.g., vertex coloring, max-cut), etc. The FM-GPS also reformulates existing problems, such as non-linear inverse problems (e.g., imaging), quantum chromodynamics (QCD), etc., as well as fast bounding schemes for existing branch and bound (B&B) algorithms. The FM-GPS works well with global optimization problems such as integer programming, e.g., software verification and certification, such as with a software defined radio (SDR). For non-linear optimization, the FM-GPS has applications in material design and machine learning, scientific computing and simulations, imaging and positioning, and numerical analysis. Partial differential equation (PDE) constrained optimization, control systems and robotics, data science, and network design and maintenance also fall within the realm of non-convex optimization problems.

[0072] In some aspects, the FM-GPS is based on the mathematical Lasserre framework. In these aspects, the FM-GPS approximates general global optimization problems by global polynomial optimization problems, which are translated into equivalent (convex) semi-definite programs (SDP) that can be solved efficiently using existing technology. Global polynomial optimization, however, in its greatest generality, is NP-hard. Nonetheless, the relaxation from the global polynomial optimization to the SDP can be performed in a highly computationally-efficient manner for a broad class of problems (e.g., a compact subset on the hypercube; in some aspects, an arbitrary compact semi-algebraic set may replace the hypercube). The FM-GPS guarantees global optimality together with lower computational costs than existing techniques. Given its advantageous scaling, stability, low cost, global optimality, and modular software design, FM-GPS opens the door to the study of optimization problems that were previously considered intractable, in addition to providing the capability of solving existing problems at a fraction of the cost of existing techniques.

[0073] As described above, the FM-GPS optimization technology tool consists of processes for solving non-convex, constrained optimization problems. Aspects of the present disclosure provide an efficient convex relaxation of the problem seen in Equation (1), by which it is reformulated to take the form of a semi-definite program (SDP):

$$\min_{\{\mu_n\}} p_n \mu_n$$

$$\text{subject to } M(\mu_n) \succeq 0.$$

When the reformulation is performed appropriately, any optimal value $\{\mu_n^*\}$ of the SDP provides information about the value and the location of min p(x) within the constrained set.

[0074] The technique scales as follows: first, regular multidimensional functions may be represented efficiently using polynomial expansions. For instance, in the case of an analytic function, a Chebyshev expansion with maximum Euclidean degree d=O(log(∈)) (independent of dimension) is sufficient for obtaining an ∈-accurate approximation $\|\cdot\|_\infty$-norm independently of dimension, where $\|\cdot\|_\infty$-norm corresponds to the maximum of the absolute value of the

function on its domain. In particular, this implies that the number of coefficients is polynomial in dimension.

[0075] Second, computing the Chebyshev polynomial coefficients of a function is a numerically stable linear operation. Thus, one only needs to know approximately as many function values as coefficients in order to compute them. Therefore, only polynomially many (e.g., in dimension) function values are necessary in order to interpolate a regular function accurately in $\|\cdot\|_\infty$-norm. Computing those function values corresponds to exploring the entire landscape.

[0076] Third, once polynomial coefficients have been computed, the FM-GPS can find the location of a global optimum. In the case of neural networks, the SDP has the following characteristics: the total number of weights in the network corresponds to the dimension of the full problem; weights deeper in the network have a higher associated polynomial degree in the objective that grows linearly with depth; and the size of the data does not affect the size of the SDP (neither the dimension, nor the degree of the interpolant). In the case of an additive training function of the form:

$$e(w) := \sum_{n=1}^{N} \rho(X_n; w), \tag{6}$$

where $\rho(x_n; w)$ represents the training error at data point x_n, for weights w, the algorithm scales linearly on the number of data points N. In particular, objective function evaluation can be carried out once in parallel because the interpolations points $\{w_k\}$ are known a priori (computing $\rho(x_n; w_k)$ is parallel in n and k).

[0077] In some aspects of the present disclosure, the FM-GPS builds on recently-developed technology from Lasserre, but offers significant improvements, including the following characteristics, which allow the FM-GPS to surpass existing techniques and are key to tackling global optimization problems efficiently and at practical scales. The improvements include advantageous scaling, in other words, possessing provably efficient computational bounds that are significantly better than existing techniques. The advantages of FM-GPS in computational bounds are asymptotic, so as the problem size grows, the advantage grows. Speedups of 1,000,000 times are possible with the FM-GPS techniques.

[0078] Another improvement of the FM-GPS is global optimality, such that output is guaranteed to be within a prescribed error of the true solution. The FM-GPS outperforms currently-existing techniques that are either computationally expensive or restricted to local optima. Another improvement, optimal location, computes both the optimal value and an optimal location. Current techniques are limited to computing the optimal value, and can find an optimal location only if stringent conditions are satisfied.

[0079] The improvement of numerical stability allows the FM-GPS to solve problems of an arbitrary large degree (e.g., objective and constraints). Finally, software performance is improved such that existing, powerful convex optimization technology can be leveraged and tailored to produce a high-performance implementation.

[0080] The FM-GPS method generally relies on an efficient SDP solver as its computational backbone. Unfortunately, existing solvers suffer from both instability issues and disadvantageous scaling. The reasons underlying these issues are numerous, but can be classified into three main categories: 1) the generality-specialization trade-off, 2) a memory bottleneck, and 3) problem size. Each is described below, along with differences between the problems targeted by these existing solvers and the problem at hand. Specifically, SDPs associated with polynomial optimization problems possess much more structure than the general case considered by existing solvers. This structure can be leveraged to create specialized solvers that are both more stable and orders-of-magnitudes faster than those currently available.

[0081] The generality-specialization trade-off occurs because existing solvers are designed with broad applicability in mind and cannot leverage the structure of specific problems for the most part. The high cost associated with existing solvers may be overcome in the context of polynomial optimization in a variety of ways that involve leveraging the special structure, including: the block Hankel-plus-Toeplitz structure of the moment matrix; the existence and/or search for low-rank solutions, and a linear programming (LP) warm start.

[0082] The memory bottleneck occurs because many existing solvers are second order and rely on the knowledge of the Hessian, which consumes a large amount of storage and is often not tractable large scale. The high cost associated with a second-order solver may be significantly lowered through the use of first-order solvers (e.g., projected gradient descent). These have the advantage of only specifying knowledge of the value of the optimization functional and its gradient, both of which are known analytically due to the linear nature of the functional. Further, first-order solvers do not store or compute a large Hessian matrix, an operation which may carry a cost as high as $O(N^2)$.

[0083] Current solvers may only handle problems with a limited number, e.g., 10^3, of variables. A polynomial optimization problem's size, however, scales polynomially in dimension. Although this scaling is highly-advantageous when compared with other global optimization schemes providing guarantees of global optimality (e.g., naive search), this characteristic significantly reduces the reach of existing techniques. Problem size is a fundamental issue associated with dense polynomial optimization. Dense means that all coefficients associated with basis polynomials of degree at most d, are non-zero. The problem size may be overcome through the use of hierarchical polynomials. Hierarchical polynomials are a family of polynomials that possess a hierarchical structure associated with a function composition reminiscent of neural networks, but with polynomial activation functions.

[0084] FIGS. **6**A and **6**B illustrate examples of hierarchal polynomials, in accordance with aspects of the present disclosure. FIG. **6**A illustrates an analytical representation at a macroscopic level. FIG. **6**B illustrates a graphical representation at a microscopic level where each edge in FIG. **6**A has been expanded to show a dependence on each scalar input/output. Hierarchical polynomial-based optimization relies on "unwrapping" this hierarchical structure to break a large global optimization problem into a sequence of much smaller, more manageable problems, as seen in the following equation:

$$p_0^{(0)}(\Sigma_{k_1} C_{0,(k_0,k_1)}\, p_{k_1}^{(1)}(\cdots \Sigma_{k_{L-1}} C_{jL-1,(k_{L-2},k_{L-1})}\, p_{k_{L-1}}^{(L-1)}(\Sigma_{k_L} {}_{iL}C_{jL,(k_{L-1},k_L)}\, {}^{(L)}X_{K_L}^{(L)}))) \tag{7}$$

[0085] In FIGS. **6**A and **6**B, the nodes are vector polynomial functions (FIG. **6**A) P_0, P_1, P_2, and P_3 taking a vector

input and outputting vectors, or scalar polynomials (FIG. **6**B) p_0 taking a scalar input and outputting a scalar. The edges indicate composition, e.g., the output of a node is the input of a connected node if there is an edge between the two. The quantities C_0, C_1, C_2, . . . C_L are linear transformations, e.g., matrices. The parameters x_0, x_1, . . . are inputs to the neural network and the elements of the matrices C are the parameters, also referred to as weights. In training, the weights are allowed to vary, and the total function is a polynomial with variables corresponding to the weights. The parameter L represents the number of layers, in other words, the depth of the composition. The parameters (m,n) seen in FIG. **6**B correspond to the sub-indices $\{k_{L-1}, k_{L-j}\}$ in Equation (7), although (m,n) correspond to elements in the matrix C. In FIG. **6**B each actual edge in the neural network is shown with a single weight associated with a corresponding matrix element. All explicit indices are not shown to avoid cluttering the figure. That is, the notation in FIG. **6**B is descriptive rather than explanatory.

[0086] The issues of current solvers may be addressed by improving computational capabilities by developing SDP solvers, designed specifically for FM-GPS. Moreover, a search strategy based on "space-filling curves" may be used to tailor the dimensionality of the problem to be better suited to strengths of the FM-GPS. Finally, combination strategies may be employed, including using FM-GPS to quickly locate a global basin of attraction and then perform a stochastic gradient descent (SGD) for a local optimal refinement strategy.

[0087] Currently, neural networks (NNs) are often trained with an SGD, especially for learning in classification realms, even when regimes are over-parameterized, producing an optimization landscape prone to be exploited by descent techniques. FM-GPS, however, changes the paradigm that is assumed through SGD by changing some of the fundamental, underlying assumptions, including computing a global minimum, removing the need for over parametrization. Moreover, with FM-GPS, it may be possible to achieve levels of accuracy similar to the state-of-the-art and SGD, but with significantly fewer weights by finding a global minimum. If FM-GPS can fit an NN using fewer weights (e.g., fewer degrees of freedom), then less data may be needed. Using fewer weights improves both the training time as well as the inference time (because evaluations become cheaper). In some implementations, it is possible to achieve levels of accuracy similar to the state of the art, but with significantly fewer weights by finding a global minimum.

[0088] According to aspects of the present disclosure, the FM-GPS may supplant existing global optimization routines with minimal changes to existing code, due to the modular structure of the FM-GPS. That is, FM-GPS may be implemented as a software tool that specifies little input from the user. An exemplary design consists of three (3) modules: an objective function and parameter specification module, a computational engine, and a visualization and post-processing module.

[0089] FIG. **7** is a block diagram illustrating an example FM-GPS processor-based system, in accordance with aspects of the present disclosure. The objective function and parameter specification module includes an evaluation routine **702** and a location value pairs module **704**. The user-provided box routine takes as inputs points in parameter space and outputs the objective value, or a set of pre-computed location-value pairs characterizing the objective. The user also provides a few parameters specific to the global optimization routine (e.g., maximum number of SDP iterations, desired accuracy, etc.).

[0090] The computational engine performs a large percentage of the overall computations of the software tool. The main component is an efficient interpolation sub-routine **706** that translates the user input into the proper format (e.g., SDP), as well as a high-performance SDP solver implementation specifically tailored to our formulation. Outputs of the interpolation sub-routine **706** are received at a module **708**. The module **708** computes an optimal value and an optimal location. An optimal value is the smallest value the objective can reach among all feasible points. An optimal location is where this value is achieved, and may not be unique. For instance (x−1)*(x+1) has minimum value over the real numbers equal to 0, but two optimal locations (where this value is achieved) at +1 and −1. The FM-GPS starts by finding the optimal value, and then performs additional steps to find at least one location (e.g., random among all possible optimal locations) where this value is achieved.

[0091] The visualization and post-processing module consists of a set of tools made available to the user to examine the computed solution and/or feed the solution to another piece of software for further computations. Examples include visualization tools **610** in low dimensions, as well as an a posteriori solution certification.

[0092] Positive polynomials constitute powerful mathematical tools that find many applications in the context of global optimization, control theory, and algebraic geometry, to name but a few. One major question associated with positive polynomials is their representation through sums-of-squares (SOS) polynomials. The interest in this question arises from the fact that multidimensional SOS polynomials, which form a subset of positive polynomials, can be easily characterized through the positive-definiteness of an associated bilinear form through linear matrix inequalities, while a complete and succinct characterization of positive polynomials remains difficult, except in very special cases. In this sense, it is often much more advantageous to work with SOS polynomials than with positive polynomials directly.

[0093] It is known that polynomials p(x) positive over compact semi-algebraic sets generated by polynomials constraints $g_i(x)$ can often be represented through a combination of SOS polynomials and algebraic constraints as seen in the following equation (notation described below)

$$P(x)-\sigma_0(x)\,|\,\Sigma_{i=1}{}^{l}\sigma_i(x)g_i(x),\forall x\in\cap_{i=1}{}^{l}\{x\in[-1,1]^D:gi(x)\geq 0\} \tag{8}$$

[0094] An important result in this regard has been shown by an existence proof for such an exact representation for some properly-chosen SOS polynomials $\{\sigma_i(x)\}$. Another result in the same category has demonstrated the denseness of sums-of-squares in the $\|\cdot\|_1$ coefficient norm in the cone of polynomials non-negative on $[-1,1]^D$. Lasserre further expanded on the latter with a similar denseness result over $\mathbb{R}^D$, modulo a sums-of-squares perturbation, together with explicit converging sequences. See "A Sum of Squares Approximation of Nonnegative Polynomials," SIAM review 49.4 (2007) pp. 651-669. Following this, estimates have been introduced on the degree of the sums-of-squares involved in those expressions for polynomials positive over the hypercube and compact semi-algebraic subsets of the

hypercube, respectively. These bounds, however, are all at least exponential in the dimension and degree of the positive polynomial.

[0095] Aspects of the present disclosure address the problem of efficiently approximating polynomials that are positive over compact semi-algebraic sets through combinations of Sums-Of-Squares (SOS) polynomials and polynomial constraints. New estimates are introduced for the degree required for approximating such polynomials that are significantly more efficient than existing results. In particular, the results can be used to provide polynomial bounds on the complexity of many important computational problems, including problems in algebraic geometry as well as global polynomial optimization.

[0096] For all that follows, define $\mathbb{P}_{D,d}$ 0<d, D∈ ℕ to be the vector space of multivariate polynomial with real coefficients defined over the D-dimensional hypercube $[-1,1]^D$ and having a degree at most d, equipped with the maximum norm, for example:

$$\mathbb{P}_{D,d}:=\{p(x)\in \mathbb{R}[x]:x\in[-1,1]^D,\deg(p(x))\leq d\} \tag{9}$$

with norm:

$$\|p(x)\| = \|p(x)\|L^{\infty}[-1,1]^D = \max_{x\in[-1,1]^D}|p(x)|. \tag{10}$$

[0097] For a fixed, finite set of polynomials $\{g_i(x)\}_{i=1}^{l}\in \mathbb{P}_{D,d}$, I∈ ℕ define the compact semi-algebraic set (assumed to be non-empty):

$$\mathbb{K}:\cap_{i=1}^{l}\{x\in[-1,1]^D:g_i(x)\geq 0\}. \tag{11}$$

[0098] Refer to the polynomials $\{g_i(x)\}$ as polynomial constraints. The polynomial p(x) is said to be positive (respectively strictly positive) over the semi-algebraic set 𝕂 is p(x)≥0 (respectively >) for all x∈ 𝕂. A sums-of-squares (SOS) polynomial belonging to $\mathbb{P}_{D,2d}$ is a polynomial of the form,

$$\sum_{j=1}^{J}(q_j(x))^2, \tag{12}$$

for polynomials $q_i(x)\in \mathbb{P}_{D,d}$ and J∈ ℕ. SOS polynomials from a subset (e.g., sub-cone) of the set (e.g., cone) of positive polynomials.

[0099] Aspects of the present disclosure rely on the following assumption:

[0100] Assumption 1. Consider the non-trivial polynomials p(x), $\{g_i(x)\}_{1=1}^{l}\in \mathbb{P}_{D,d}$. Assumption 1 is satisfied if there exists some 0<∈*<1 such that p(x)>0 for all x in,

$$\mathbb{K}_{\in}:\cap i=1\ \mathbb{K}_{\in^{(i)}}:\cap_{i=1}^{l}\{x\in[-1,1]^D:g_i(x)\geq-\in\},(13)$$

and all 0<∈≤∈*.

[0101] Assumption 1 may appear restrictive, but it is not. There always exists a strictly positive number ∈* such that it holds. This number, which only depends on the specific nature of the polynomials involved, may however be arbitrarily small. In this sense, Assumption 1 makes the dependence of the degree scaling on the properties of the polynomial family explicit, and allows one to compute estimates on a case by case basis.

Theorem 1. If p(x), $\{g_i(x)\}_{1=1}^{l}\in \mathbb{P}_{D,d}$ satisfy Assumption 1 for some 0<∈*<1, then for any 0<∈<∈*, there exists sum-of-squares polynomials $\{_i^2(x)\}_{i=1}^{l}$ and $\theta^2(x)$ of degrees:

$$deg(\varsigma_i(x)) = O\left(\xi\left(\|p(x)\|,\|g(x)\|,\frac{1}{\epsilon}\right)deg(g_i(x))\right),\ i=1,\dots,I \tag{14}$$

$$deg(\theta(x)) = O\left(k\left(\|p(x)\|,\{\|g_i(x)\|\},\frac{1}{\epsilon}\right)\max\left(\{deg(g_i(x))\}_{i=1}^{I},deg(p(x))\right)\right)$$

such that,

$$\left\|p(x)-\left(\theta^2(x)+\sum_{i=1}^{I}\xi_i^2(x)g_i(x)\right)\right\|\leq\epsilon, \tag{15}$$

where $\{\xi_i(\cdot)\}_{i=1}^{l}$ and k(·) are explicit polynomials independent of D and d.

[0102] This result states that, under some mild assumption, positive polynomials over compact semi-algebraic sets may be approximated in a pointwise fashion through a combination of SOS polynomials and polynomial constraints. The number of summands involved in the approximation is at most equal to the number of algebraic constraints plus one. Further, given some fixed desired accuracy, the degree of the SOS polynomials is bounded by a quantity proportional to the degree of the original polynomials involved. The degree is otherwise independent of dimension to a theorem where the bound is super-exponential both in the dimension and the degree of the original polynomials.

[0103] Aspects of the present disclosure introduce new quantitative estimates for producing efficient pointwise approximations of polynomials that are positive over compact semi-algebraic subsets of the hypercube through combinations of sums-of-squares polynomials and polynomial constraints. The approximation involves polynomials, whose degree is bounded polynomially in the degree of the original polynomial and constraints.

[0104] Aspects of the present disclosure introduce techniques for solving complex, multiscale, and high-dimensional global optimization problems that are scalable, dynamically adaptable, and able to rapidly locate a global optimum for any sufficiently regular (e.g., continuous) high-dimensional and non-convex function. These aspects can be applied to the rapid training of deep neural networks, for example, for image recognition.

[0105] One component of the proposed techniques is a novel hierarchical structure, (e.g., nested polynomials). This structure enables efficient approximation of regular optimization functions, and then discretizes the resulting (approximate) optimization problem into a sequence of small optimization sub-problems: multidimensional polynomial optimization problems. Another component is global multidimensional polynomial optimization (e.g., convex relaxation), which efficiently handles multidimensional polynomial optimization sub-problems. This global multidimensional polynomial optimization improves the scalability, scope, and stability of existing techniques and integrates them within the hierarchical framework for maximum stability, performance, and usability.

[0106] In the following non-limiting description, the problem area is training deep neural networks for image recognition, with the aim of reducing the computational cost by

orders of magnitude. In this context, the training problem may be expressed within an optimization framework as follows: For $\{X_i\}$ a set of images, and $\{ _i\}$, a set of vectors indicating the presence or absence of a class within each corresponding image, and for $v_w(\cdot)$, a vector-valued neural network feedforward function depending on the set of weights w, the training problem becomes one of finding the best set of weights w* such that, $w^*=\text{argmin}_w\Sigma_i\text{Err}(v_w{}^*X_i)$, $_i$), where Err( ) is a measure of error. This type of problem is high-dimensional, nonconvex, computationally difficult, and can neither be globally nor efficiently optimized through current methods. Consequently, the problem benefits from the method of training a neural network with a polynomial solver in at least two ways: a significant increase in performance (e.g., training time, power usage, memory, etc.), and the identification of a global optimum that leads to better prediction power, smaller networks (e.g., fewer weights), and less training data.

[0107] Many problems in machine learning, data analysis, and various engineering domains can be posed within a global optimization framework. Current optimization methods, however, cannot solve these problems in a tractable or scalable fashion. The proposed techniques offer both scalability and optimality. In the selected problem space, a current method of choice is stochastic gradient descent (SGD), a convex optimization method relying on small random subsets (batches) of the data for gradient computations. This approach, however, possesses multiple drawbacks, such as a tendency to get stuck at local minima. To mitigate the effects of these issues, brute-force hardware solutions are employed to rapidly perform a very large number of simple computations. This solution, however, is computationally expensive and may lead to suboptimal results given the local nature of SGD. The proposed approach overcomes both issues.

[0108] Although the present disclosure is described with respect to training neural networks, and more particularly, image recognition, the present disclosure has broader applicability. The polynomial solver training method is applicable to any global optimization problem as long as the underlying function of interest is sufficiently regular, for example, a hierarchical representation, such a neural network or a structure approximating a neural network. The proposed method will benefit multiple applications including representation/evaluation of high-dimensional probability densities for a Partially Observable Markov Decision Process, engineering design, and optimal control, among other applications.

[0109] Aspects of the present disclosure implement a scalable global interpolation/optimization process based on nested polynomials and the multidimensional polynomial optimization techniques described above. Aspects of the present disclosure result in the locating of a global optimum, have low computational complexity, are scalable in dimension, accuracy, and size, are dynamically adaptive, enable access to observations and/or evaluation routines, and are inherently parallelizable.

[0110] Nested polynomials are a family of polynomials possessing a hierarchical structure (e.g., nesting) associated with function composition, as seen in FIGS. **6**A and **6**B, where FIG. **6**A illustrates a standard representation and FIG. **6**B illustrates an expanded representation. In this sense, nested polynomial-based optimization relies on the idea of "unwrapping" this hierarchical structure to break a large optimization (e.g., interpolation) problem into a sequence of smaller problems. In this context, each step involves a few sub-problems of smaller size that consist of optimizing a multidimensional polynomial. Nested polynomials offer a way to efficiently approximate functions possessing a large number of degrees of freedom (e.g., regular functions) using significantly fewer parameters. The hierarchical structure allows one to break a large optimization problem into a sequence of smaller problems.

[0111] FIGS. **6**A and **6**B show a nested structure (e.g., higher-dimensional tree structure). Solving an optimization problem implies traversing the tree in a specific order, solving a short sequence of optimization sub-problems at each node. The size of these sub-problems is characterized by the number of interactions among nodes (FIG. **6**B). By carefully tuning the nested structure and the number of interactions, it is possible to keep the size of the sub-problems, and hence the computational cost of the method, low while retaining the approximation power necessary to represent functions of interest.

[0112] These characteristics allow efficient and rapid representation of optimization functions (e.g., approximation power), and rapidly identifying an optimum (e.g., hierarchical structure). In addition, the proposed methods are designed to provide the user with guarantees of global optimality.

[0113] The proposed approach is based on the two novel algorithms, namely, a fast multidimensional polynomial optimization (MPO) algorithm (also referred to as Fast Multidimensional Global Polynomial Solver or FM-GPS), and a nested polynomial-based interpolation/optimization algorithm.

[0114] As discussed above, the techniques may be used to stably and rapidly compute solutions to problems (e.g., sub-problems within the unwrapping) of the form, argmin_x q(x), where q(x) is a multidimensional polynomial (dense, not nested) supported on a bounded subset of a Euclidean space of low dimension (generally the hypercube). As stated previously, the proposed approach will generalize, stabilize, and increase the performance of recently-developed, yet inadequate, techniques. To improve numerical stability, the optimization domain may be constrained to the hypercube and a stable orthogonal basis may be used to represent polynomial functions over this domain. For a different compact domain, an affine transformation and a penalty function may be used. This leads to a method with global stability for all degrees and in all dimensions. The relationship between Chebyshev polynomials and the fast Fourier transform (FFT) may be exploited to speed up computations.

[0115] Size and scaling is improved through a change of variables. The problem may be periodized. If $T_n(x)$ is a Chebyshev polynomial of degree n, then $T_n(\cos(\theta))=\cos(n\theta)$. A rank-1 lattice may be used to convert the problem to an equivalent one-dimensional (1D) formulation. As a consequence, the reformulated problem possesses a bounded number of variables and a Hankel-plus-Toeplitz structure.

[0116] To obtain the bounded number of variables, results of sum-of-squares (SOS) expansions in **1**D may be used, and the degree of the polynomials involved may be bounded by that of the original polynomial divided by two. This leads to a sequence containing at most two semi-definite programs (SDPs) with sizes bounded by the number of non-zero coefficients of the original polynomial.

[0117] The matrix A(x) possesses a Hankel-plus-Toeplitz structure. This can be exploited to reduce the cost of assessing definiteness from $O(M)^3$ to at most $O(M^2 \log(N))$ using recently-developed fast algorithms for structured matrices.

[0118] Optimum extraction is also obtained. Isolated optimal solutions (Dirac measures at optimal points) correspond to a sum of two rank-1 matrices (A(x) in the SDP introduced above), and the rank-1 matrices are in one-to-one correspondence with extreme points of the solution set. Therefore, having identified the minimum value though the first SDP, a second SDP (restricted to the optimal set) is solved. The solution corresponds to an extreme point. From this, the resulting matrices may be readily factored (being of rank-1), and an optimal location (or global optimum) may be obtained.

[0119] In short, if q(x) is a polynomial of degree d in D dimensions possessing

$$Q = \binom{D+d}{d} = O(D^d)$$

coefficients, the technique reformulates the global polynomial optimization problem (as described below) into two SDPs involving a number of variable M polynomials in the dimension and bounded by the number of non-zero coefficients of q(x), e.g., Q.

[0120] A nested polynomial-based interpolation/optimization technique is now described in more detail. Nested polynomials are functions of the form,

$$p_0^{(0)}\Big(\Sigma_{k_1} C_{0,(k_0,k_1)}^{(L-1)} p_{k_1}^{(1)}\Big(\cdots \Sigma_{k_{L-1}} C_{j_{L-1},(k_{L-2},k_{L-1})}^{(L)} p_{k_{L-1}}^{(L-1)}\Big(\Sigma_{k_L} C_{j_L,(k_{L-1},k_L)}^{(L)} \gamma_{k_L}^{(L)}\Big)\Big)\Big) \quad (16)$$

where $C_j^{(i)}$ are matrices (e.g., of neural network weights), $p_{il}^{(l)}(\cdot)$ are polynomials (e.g., approximating an activation function), and $\gamma^{(L)}$ is a vector (e.g., an input image). Referring to the graph representation of a nested structure in FIG. **6**A, it is noted that such representation is always a tree, although not necessarily a binary tree, as it can represent higher-dimensional spaces. As discussed above, nested polynomials constitute a basis to represent regular functions. Further, the structure enables scalability, allowing rapid approximating/reformulating of an optimization problem as a nested polynomial optimization problem (through interpolation), and then splitting the resulting problem into a sequence of small sub-problems involving the minimization of multidimensional polynomials.

[0121] Notation for nested polynomial-based interpolation or optimization is introduced in Table 2.

TABLE 2

Notation	Description
1	A superscript ⑦ represents the level in the tree (from left to right, starting from 0), and the subscript ⑦ represents the index of the quantity (node or edge) increasing from top to bottom, starting from 0.
2	To each, non-leaf node ⑦ corresponds a vector-valued polynomial function (denoted ⑦) of the node's children.
3	To each edge corresponds a matrix ⑦ of weights representing linear interactions between the arguments of ⑦ and the children of node ⑦.
4	Leaf nodes represent disjoint subsets of the original independent variables.
5	⑦ indicates the set of variables associated with the node ⑦ indicates the set of weights associated with the edge ⑦.
6	⑦ represents the output of a node ⑦ when seen as a variable by the parent node.
7	⑦ represents a scalar polynomial function, and ⑦ represent the elements of matrix ⑦
8	⑦ and ⑦ indicate the sets of parents, siblings, and children of the node ⑦ respectively.

TABLE 2-continued

⑦ indicates text missing or illegible when filed

[0122] FIG. **6**B shows the nesting structure ater fully expanding the graph. The lack of an edge in this case indicates that the corresponding weight/matrix entry is zero. With the notation of Table 2, the function p_0(**[text missing or illegible when filed]**)(·) associated with the node p_0(**[text missing or illegible when filed]**) takes the form,

$$P_0^{(1)}\big(\pi(P_0^{(2)}), \pi(⑦)\big) = \big(p_0^{(1)}(⑦⑦⑦), p_1^{(1)}(⑦⑦⑦)\big), \quad (17)$$

⑦ indicates text missing or illegible when filed

where $\Pi(P_0^{(2)}$=(x(**[text missing or illegible when filed]**),x**[text missing or illegible when filed]**(**[text missing or illegible when filed]**)), $\Pi\{P_1$(**[text missing or illegible when filed]**) }={x**[text missing or illegible when filed]**(**[text missing or illegible when filed]**),x(**[text missing or illegible when filed]**)} represent the vector-valued outputs of nodes and ($P_0^{(2)}$) and ($P_1^{(3)}$), e.g., the children of $P_0^{(1)}$, respectively. Also, $\Pi(C_0$(**[text missing or illegible when filed]**))={C(**0**,**[text missing or illegible when filed]**)$^{(3)}$,C_0(**0**,**[text missing or illegible when filed]**)(**[text missing or illegible when filed]**),C_0(**0**,**[text missing or illegible when filed]**)(**[text missing or illegible when filed]**),C_0(**[text missing or illegible when filed]**,**3**)(**[text missing or illegible when filed]**)}, $\Pi(X_1)=\{x_4,x_5,x_6,x_7\}$, por($P_0^{(1)}$)=($P_0^{(0)}$), sib(P**[text missing or illegible when filed]**(**[text missing or illegible when filed]**))={P(**[text missing or illegible when filed]**)} and child(P_0(**[text missing or illegible when filed]**))={$P_0^{(2)}$,$P_1^{(2)}$}.

[0123] Example pseudocode for the technique is provided below.

Denote the current node by ⑦ and the current set by ⑦, where ⑦ is the optimal set for the previous layer node ⑦.
Denote by ⑦(⑦) the (marginal) set associated with ⑦, where the marginal set represents the data from the domain that obtains the data of the range.
Denote ⑦(⑦) the vector of length M associated with the set of variables: ⑦ := {⑦, . . . , ⑦}, in other words, determine if all subnodes have been visited.

1. Let ⑦ ← ⑦, and ⑦ ← {⑦ is a minimum value for ⑦(⑦)},
2. WHILE (NOT(⑦ = ⑦ AND all children of ⑦ have been visited)) - corresponding to a visit in each super node

-continued

```
3.    IF ? is not at the leaf level
4.      IF all children of ? have been visited
5.        Set ?(?) ← ?(?(child(?)), ?) ← ?(?).
6.      ELSE
7.          - Identify Q, the top-most child of ? which has not yet been visited.
8.          - Let ?(?(Q)) be the (previously-assigned) values of the variables
              associated with the visited siblings of Q.
9.          - Let ?(Q) be the set of unvisited siblings of Q.
10.         - Compute the marginal of the current set ? at node Q, e.g.,
11.       ?(Q) ← {Π(Q) :? Π(?∩?(Q?) s.t. ? ∈ ?(?∩?(Q)), Π(Q), Π(?∩?(Q))) ∈ ?}
12.         - Set ? ← Q, ? ← ?(Q)
13.     ENDIF
14.   ELSE - call to FM-GPS
15.     Compute a point ?? such that ?(??) ∈ ?.
16.     Set ?(Π(?)) ← ??, ? ← par(?)
17.   ENDIF
18. ENDWHILE
```

? indicates text missing or illegible when filed

[0124] The above pseudo code is an example of an optimization procedure. Interpolation may be carried out in a similar manner (e.g., the elements of the matrices $C_j^{(i)}$ become the variables, and the variables $\{x_i\}$ become coefficients). The optimization cost and the cost of interpolation, which are of the same order, may be estimated as follows: to each node is associated O(1) sub-problems (denote a bound on cost by $C_{sub}<\infty$). Assuming a domain dimension of N, there can be at most O(N) nodes because the number of nodes decreases roughly by a constant factor at each level. Thus, the entire cost is of order $O(C_{sub}N)=O(N)$.

[0125] Observations and technical details of nested polynomial-based interpolation/optimization are now described. The sets introduced in the pseudocode described above are characterized through polynomial approximations of their indicator functions, which is computationally advantageous. The indicator function of a set is equal to 1 if the argument belongs to the set and 0 otherwise. For example, refer to the problem of finding an element belonging to a set B. In this framework, this is equivalent to locating a maximum of the polynomial approximation of the indicator function **[text missing or illegible when filed]**$_B(\cdot)$ of B, e.g., it is a polynomial optimization problem. The characterization of marginal sets (e.g., the computation of a polynomial approximation to indicator functions) may be achieved using polynomial optimization. It suffices to choose classical interpolation points, to solve a polynomial optimization problem for each point (e.g., maximum of marginal set indicator with interpolation points fixed), and to interpolate using the resulting values.

[0126] Every time line **15** is reached in the pseudocode, the partial coordinates of a solution point are computed. These coordinates are accumulated until an exit statement is encountered, at which point the full optimal location has been computed. A careful analysis of the procedure shows that the way marginal sets are constructed guarantees that subsequent sets will never be empty. This guarantees that the process can always be traversed, and a global optimum can always be found. In practice, a sublevel set is used to begin with rather than the optimal level set. This allows control of accuracy, and bounding variables to compact regions. An appropriate sublevel set bound may be found rapidly using a bisection-like technique. Whenever the function to be optimized is not in nested polynomial form, the optimization step is preceded with an interpolation step to produce a nested-polynomial approximation. Once completed, the optimization of the resulting nested polynomial approximation occurs, as shown in the pseudocode. The use of indicator functions for the marginal set in an interpolation context leads to dynamic adaptability; in this case, the addition of information (data, interpolation points, etc.) translates into the need to compute elements belonging to the intersection of marginal sets. This, however, is no more expensive than in an optimization setting for elements belonging to the intersection of sets $\{B_i\}$**[text missing or illegible when filed]**^{L} corresponding to $\{x:\Sigma_{i=1}^{L}{}_B(x)=L\}$, and the dimension and degree of the polynomial approximation of $\Sigma_{i=1}^{L}$ **[text missing or illegible when filed]** B(x) is no different from that of a single set (e.g., being a sum).

[0127] FIG. **8** illustrates a method **800** for training of a neural network with a polynomial solver. As shown in FIG. **8**, the method **800** may include approximating an optimization problem for training an artificial neural network as a nested polynomial optimization problem (block **802**).

[0128] In some aspects, the method **800** may include dividing the nested polynomial optimization problem into a sequence of sub-problems (block **804**). For example, the sequence of sub-problems may comprise multidimensional polynomial optimization problems and/or nested polynomials.

[0129] In some aspects, the method **800** may include hierarchically solving the sequence of sub-problems to train the artificial neural network (block **806**). For example, the hierarchically solving may include receiving as input, a global polynomial optimization problem that approximates the optimization problem for training the artificial neural network; and relaxing the global polynomial optimization problem including polynomial constraints with multiple semi-definite programs. The hierarchically solving may further include solving the of semi-definite programs based on a pre-defined structure; outputting a solution indicating a location of a global optimum of the optimization problem; and training the artificial neural network based on the solution.

[0130] The various operations of methods described above may be performed by any suitable means capable of performing the corresponding functions. The means may include various hardware and/or software component(s) and/or module(s), including, but not limited to, a circuit, an

application specific integrated circuit (ASIC), or processor. Generally, where there are operations illustrated in the figures, those operations may have corresponding counterpart means-plus-function components with similar numbering.

[0131] Example Aspects

[0132] Aspect 1: A processor-implemented method, comprising: approximating an optimization problem for training an artificial neural network as a nested polynomial optimization problem; dividing the nested polynomial optimization problem into a sequence of sub-problems; and hierarchically solving the sequence of sub-problems to train the artificial neural network.

[0133] Aspect 2: The processor-implemented method of Aspect 1, in which the sequence of sub-problems comprises multidimensional polynomial optimization problems.

[0134] Aspect 3: The processor-implemented method of Aspect 1 or 2, in which the sequence of sub-problems comprises nested polynomials.

[0135] Aspect 4: The processor-implemented method of any of the preceding Aspects, in which the hierarchically solving comprises: receiving as input, a global polynomial optimization problem that approximates the optimization problem for training the artificial neural network; relaxing the global polynomial optimization problem including polynomial constraints with a plurality of semi-definite programs; solving the plurality of semi-definite programs based on a pre-defined structure and outputting a solution indicating a location of a global optimum of the optimization problem; and training the artificial neural network based on the solution.

[0136] Aspect 5: The processor-implemented method of any of the preceding Aspects, in which the pre-defined structure comprises at least one of a sparse structure, a hierarchical structure, a block Hankel-plus-Toeplitz structure, a low-rank structure, or an underlying geometry structure.

[0137] Aspect 6: An apparatus, comprising: a memory; and at least one processor coupled to the memory, the at least one processor configured: to approximate an optimization problem for training an artificial neural network as a nested polynomial optimization problem; to divide the nested polynomial optimization problem into a sequence of sub-problems; and to hierarchically solve the sequence of sub-problems to train the artificial neural network.

[0138] Aspect 7: The apparatus of Aspect 6, in which the sequence of sub-problems comprises multidimensional polynomial optimization problems.

[0139] Aspect 8: The apparatus of Aspect 6 or 7, in which the sequence of sub-problems comprises nested polynomials.

[0140] Aspect 9: The apparatus of any of the Aspects 6-8, in which the at least one processor is further configured: to receive as input, a global polynomial optimization problem that approximates the optimization problem for training the artificial neural network; to relax the global polynomial optimization problem including polynomial constraints with a plurality of semi-definite programs; to solve the plurality of semi-definite programs based on a pre-defined structure and outputting a solution indicating a location of a global optimum of the optimization problem; and to train the artificial neural network based on the solution.

[0141] Aspect 10: The apparatus of any of the Aspects 6-9, in which the pre-defined structure comprises at least one of a sparse structure, a hierarchical structure, a block Hankel-plus-Toeplitz structure, a low-rank structure, or an underlying geometry structure.

[0142] Aspect 11: An apparatus, comprising: means for approximating an optimization problem for training an artificial neural network as a nested polynomial optimization problem; means for dividing the nested polynomial optimization problem into a sequence of sub-problems; and means for hierarchically solving the sequence of sub-problems to train the artificial neural network.

[0143] Aspect 12: The apparatus of Aspect 11, in which the sequence of sub-problems comprises multidimensional polynomial optimization problems.

[0144] Aspect 13: The apparatus of Aspect 11 or 12, in which the sequence of sub-problems comprises nested polynomials.

[0145] Aspect 14: The apparatus of any of the Aspects 11-13, in which the means for hierarchically solving comprises: means for receiving as input, a global polynomial optimization problem that approximates the optimization problem for training the artificial neural network; means for relaxing the global polynomial optimization problem including polynomial constraints with a plurality of semi-definite programs; means for solving the plurality of semi-definite programs based on a pre-defined structure and outputting a solution indicating a location of a global optimum of the optimization problem; and means for training the artificial neural network based on the solution.

[0146] Aspect 15: The apparatus of any of the Aspects 11-14, in which the pre-defined structure comprises at least one of a sparse structure, a hierarchical structure, a block Hankel-plus-Toeplitz structure, a low-rank structure, or an underlying geometry structure.

[0147] Aspect 16: A non-transitory computer-readable medium having program code recorded thereon, the program code executed by a processor and comprising:

[0148] program code to approximate an optimization problem for training an artificial neural network as a nested polynomial optimization problem; program code to divide the nested polynomial optimization problem into a sequence of sub-problems; and program code to hierarchically solve the sequence of sub-problems to train the artificial neural network.

[0149] Aspect 17: The non-transitory computer-readable medium of Aspect 16, in which the sequence of sub-problems comprises multidimensional polynomial optimization problems.

[0150] Aspect 18: The non-transitory computer-readable medium of Aspect 16 or 17, in which the sequence of sub-problems comprises nested polynomials.

[0151] Aspect 19: The non-transitory computer-readable medium of any of the Aspects 16-18, in which the program code to hierarchically solving comprises: program code to receive as input, a global polynomial optimization problem that approximates the optimization problem for training the artificial neural network; program code to relax the global polynomial optimization problem including polynomial constraints with a plurality of semi-definite programs; program code to solve the plurality of semi-definite programs based on a pre-defined structure and outputting a solution indicating a location of a global optimum of the optimization problem; and program code to train the artificial neural network based on the solution.

[0152] Aspect 20: The non-transitory computer-readable medium of any of the Aspects 16-19, in which the predefined structure comprises at least one of a sparse structure, a hierarchical structure, a block Hankel-plus-Toeplitz structure, a low-rank structure, or an underlying geometry structure.

[0153] As used, the term "determining" encompasses a wide variety of actions. For example, "determining" may include calculating, computing, processing, deriving, investigating, looking up (e.g., looking up in a table, a database or another data structure), ascertaining and the like. Additionally, "determining" may include receiving (e.g., receiving information), accessing (e.g., accessing data in a memory) and the like. Furthermore, "determining" may include resolving, selecting, choosing, establishing, and the like.

[0154] As used, a phrase referring to "at least one of" a list of items refers to any combination of those items, including single members. As an example, "at least one of: a, b, or c" is intended to cover: a, b, c, a-b, a-c, b-c, and a-b-c.

[0155] The various illustrative logical blocks, modules and circuits described in connection with the present disclosure may be implemented or performed with a general-purpose processor, a digital signal processor (DSP), an application specific integrated circuit (ASIC), a field programmable gate array signal (FPGA) or other programmable logic device (PLD), discrete gate or transistor logic, discrete hardware components or any combination thereof designed to perform the functions described. A general-purpose processor may be a microprocessor, but in the alternative, the processor may be any commercially available processor, controller, microcontroller, or state machine. A processor may also be implemented as a combination of computing devices, e.g., a combination of a DSP and a microprocessor, a plurality of microprocessors, one or more microprocessors in conjunction with a DSP core, or any other such configuration.

[0156] The steps of a method or algorithm described in connection with the present disclosure may be embodied directly in hardware, in a software module executed by a processor, or in a combination of the two. A software module may reside in any form of storage medium that is known in the art. Some examples of storage media that may be used include random access memory (RAM), read only memory (ROM), flash memory, erasable programmable read-only memory (EPROM), electrically erasable programmable read-only memory (EEPROM), registers, a hard disk, a removable disk, a CD-ROM and so forth. A software module may comprise a single instruction, or many instructions, and may be distributed over several different code segments, among different programs, and across multiple storage media. A storage medium may be coupled to a processor such that the processor can read information from, and write information to, the storage medium. In the alternative, the storage medium may be integral to the processor.

[0157] The methods disclosed comprise one or more steps or actions for achieving the described method. The method steps and/or actions may be interchanged with one another without departing from the scope of the claims. In other words, unless a specific order of steps or actions is specified, the order and/or use of specific steps and/or actions may be modified without departing from the scope of the claims.

[0158] The functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in hardware, an example hardware configuration may comprise a processing system in a device. The processing system may be implemented with a bus architecture. The bus may include any number of interconnecting buses and bridges depending on the specific application of the processing system and the overall design constraints. The bus may link together various circuits including a processor, machine-readable media, and a bus interface. The bus interface may be used to connect a network adapter, among other things, to the processing system via the bus. The network adapter may be used to implement signal processing functions. For certain aspects, a user interface (e.g., keypad, display, mouse, joystick, etc.) may also be connected to the bus. The bus may also link various other circuits such as timing sources, peripherals, voltage regulators, power management circuits, and the like, which are well known in the art, and therefore, will not be described any further.

[0159] The processor may be responsible for managing the bus and general processing, including the execution of software stored on the machine-readable media. The processor may be implemented with one or more general-purpose and/or special-purpose processors. Examples include microprocessors, microcontrollers, DSP processors, and other circuitry that can execute software. Software shall be construed broadly to mean instructions, data, or any combination thereof, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Machine-readable media may include, by way of example, random access memory (RAM), flash memory, read only memory (ROM), programmable read-only memory (PROM), erasable programmable read-only memory (EPROM), electrically erasable programmable Read-only memory (EEPROM), registers, magnetic disks, optical disks, hard drives, or any other suitable storage medium, or any combination thereof. The machine-readable media may be embodied in a computer-program product. The computer-program product may comprise packaging materials.

[0160] In a hardware implementation, the machine-readable media may be part of the processing system separate from the processor. However, as those skilled in the art will readily appreciate, the machine-readable media, or any portion thereof, may be external to the processing system. By way of example, the machine-readable media may include a transmission line, a carrier wave modulated by data, and/or a computer product separate from the device, all which may be accessed by the processor through the bus interface. Alternatively, or in addition, the machine-readable media, or any portion thereof, may be integrated into the processor, such as the case may be with cache and/or general register files. Although the various components discussed may be described as having a specific location, such as a local component, they may also be configured in various ways, such as certain components being configured as part of a distributed computing system.

[0161] The processing system may be configured as a general-purpose processing system with one or more microprocessors providing the processor functionality and external memory providing at least a portion of the machine-readable media, all linked together with other supporting circuitry through an external bus architecture. Alternatively, the processing system may comprise one or more neuromorphic processors for implementing the neuron models and

models of neural systems described. As another alternative, the processing system may be implemented with an application specific integrated circuit (ASIC) with the processor, the bus interface, the user interface, supporting circuitry, and at least a portion of the machine-readable media integrated into a single chip, or with one or more field programmable gate arrays (FPGAs), programmable logic devices (PLDs), controllers, state machines, gated logic, discrete hardware components, or any other suitable circuitry, or any combination of circuits that can perform the various functionality described throughout this disclosure. Those skilled in the art will recognize how best to implement the described functionality for the processing system depending on the particular application and the overall design constraints imposed on the overall system.

[0162] The machine-readable media may comprise a number of software modules. The software modules include instructions that, when executed by the processor, cause the processing system to perform various functions. The software modules may include a transmission module and a receiving module. Each software module may reside in a single storage device or be distributed across multiple storage devices. By way of example, a software module may be loaded into RAM from a hard drive when a triggering event occurs. During execution of the software module, the processor may load some of the instructions into cache to increase access speed. One or more cache lines may then be loaded into a general register file for execution by the processor. When referring to the functionality of a software module below, it will be understood that such functionality is implemented by the processor when executing instructions from that software module. Furthermore, it should be appreciated that aspects of the present disclosure result in improvements to the functioning of the processor, computer, machine, or other system implementing such aspects.

[0163] If implemented in software, the functions may be stored or transmitted over as one or more instructions or code on a computer-readable medium. Computer-readable media include both computer storage media and communication media including any medium that facilitates transfer of a computer program from one place to another. A storage medium may be any available medium that can be accessed by a computer. By way of example, and not limitation, such computer-readable media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium that can be used to carry or store desired program code in the form of instructions or data structures and that can be accessed by a computer. Additionally, any connection is properly termed a computer-readable medium. For example, if the software is transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared (IR), radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. Disk and disc, as used, include compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk, and Blu-ray® disc where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Thus, in some aspects, computer-readable media may comprise non-transitory computer-readable media (e.g., tangible media). In addition, for other aspects computer-readable media may comprise transitory computer-readable media (e.g., a signal). Combinations of the above should also be included within the scope of computer-readable media.

[0164] Thus, certain aspects may comprise a computer program product for performing the operations presented. For example, such a computer program product may comprise a computer-readable medium having instructions stored (and/or encoded) thereon, the instructions being executable by one or more processors to perform the operations described. For certain aspects, the computer program product may include packaging material.

[0165] Further, it should be appreciated that modules and/or other appropriate means for performing the methods and techniques described can be downloaded and/or otherwise obtained by a user terminal and/or base station as applicable. For example, such a device can be coupled to a server to facilitate the transfer of means for performing the methods described. Alternatively, various methods described can be provided via storage means (e.g., RAM, ROM, a physical storage medium such as a compact disc (CD) or floppy disk, etc.), such that a user terminal and/or base station can obtain the various methods upon coupling or providing the storage means to the device. Moreover, any other suitable technique for providing the methods and techniques described to a device can be utilized.

[0166] It is to be understood that the claims are not limited to the precise configuration and components illustrated above. Various modifications, changes, and variations may be made in the arrangement, operation, and details of the methods and apparatus described above without departing from the scope of the claims.

What is claimed is:

1. A processor-implemented method, comprising:

approximating an optimization problem for training an artificial neural network as a nested polynomial optimization problem;

dividing the nested polynomial optimization problem into a sequence of sub-problems; and

hierarchically solving the sequence of sub-problems to train the artificial neural network.

2. The processor-implemented method of claim **1**, in which the sequence of sub-problems comprises multidimensional polynomial optimization problems.

3. The processor-implemented method of claim **1**, in which the sequence of sub-problems comprises nested polynomials.

4. The processor-implemented method of claim **1**, in which the hierarchically solving comprises:

receiving as input, a global polynomial optimization problem that approximates the optimization problem for training the artificial neural network;

relaxing the global polynomial optimization problem including polynomial constraints with a plurality of semi-definite programs;

solving the plurality of semi-definite programs based on a pre-defined structure and outputting a solution indicating a location of a global optimum of the optimization problem; and

training the artificial neural network based on the solution.

5. The processor-implemented method of claim **4**, in which the pre-defined structure comprises at least one of a

sparse structure, a hierarchical structure, a block Hankel-plus-Toeplitz structure, a low-rank structure, or an underlying geometry structure.

6. An apparatus, comprising:

a memory; and

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

to approximate an optimization problem for training an artificial neural network as a nested polynomial optimization problem;

to divide the nested polynomial optimization problem into a sequence of sub-problems; and

to hierarchically solve the sequence of sub-problems to train the artificial neural network.

7. The apparatus of claim **6**, in which the sequence of sub-problems comprises multidimensional polynomial optimization problems.

8. The apparatus of claim **6**, in which the sequence of sub-problems comprises nested polynomials.

9. The apparatus of claim **6**, in which the at least one processor is further configured:

to receive as input, a global polynomial optimization problem that approximates the optimization problem for training the artificial neural network;

to relax the global polynomial optimization problem including polynomial constraints with a plurality of semi-definite programs;

to solve the plurality of semi-definite programs based on a pre-defined structure and outputting a solution indicating a location of a global optimum of the optimization problem; and

to train the artificial neural network based on the solution.

10. The apparatus of claim **9**, in which the pre-defined structure comprises at least one of a sparse structure, a hierarchical structure, a block Hankel-plus-Toeplitz structure, a low-rank structure, or an underlying geometry structure.

11. An apparatus, comprising:

means for approximating an optimization problem for training an artificial neural network as a nested polynomial optimization problem;

means for dividing the nested polynomial optimization problem into a sequence of sub-problems; and

means for hierarchically solving the sequence of sub-problems to train the artificial neural network.

12. The apparatus of claim **11**, in which the sequence of sub-problems comprises multidimensional polynomial optimization problems.

13. The apparatus of claim **11**, in which the sequence of sub-problems comprises nested polynomials.

14. The apparatus of claim **11**, in which the means for hierarchically solving comprises:

means for receiving as input, a global polynomial optimization problem that approximates the optimization problem for training the artificial neural network;

means for relaxing the global polynomial optimization problem including polynomial constraints with a plurality of semi-definite programs;

means for solving the plurality of semi-definite programs based on a pre-defined structure and outputting a solution indicating a location of a global optimum of the optimization problem; and

means for training the artificial neural network based on the solution.

15. The apparatus of claim **14**, in which the pre-defined structure comprises at least one of a sparse structure, a hierarchical structure, a block Hankel-plus-Toeplitz structure, a low-rank structure, or an underlying geometry structure.

16. A non-transitory computer-readable medium having program code recorded thereon, the program code executed by a processor and comprising:

program code to approximate an optimization problem for training an artificial neural network as a nested polynomial optimization problem;

program code to divide the nested polynomial optimization problem into a sequence of sub-problems; and

program code to hierarchically solve the sequence of sub-problems to train the artificial neural network.

17. The non-transitory computer-readable medium of claim **16**, in which the sequence of sub-problems comprises multidimensional polynomial optimization problems.

18. The non-transitory computer-readable medium of claim **16**, in which the sequence of sub-problems comprises nested polynomials.

19. The non-transitory computer-readable medium of claim **16**, in which the program code to hierarchically solve comprises:

program code to receive as input, a global polynomial optimization problem that approximates the optimization problem for training the artificial neural network;

program code to relax the global polynomial optimization problem including polynomial constraints with a plurality of semi-definite programs;

program code to solve the plurality of semi-definite programs based on a pre-defined structure and outputting a solution indicating a location of a global optimum of the optimization problem; and

program code to train the artificial neural network based on the solution.

20. The non-transitory computer-readable medium of claim **14**, in which the pre-defined structure comprises at least one of a sparse structure, a hierarchical structure, a block Hankel-plus-Toeplitz structure, a low-rank structure, or an underlying geometry structure.

* * * * *