(12) STANDARD PATENT (11) Application No. AU 2017230184 B2
(19) AUSTRALIAN PATENT OFFICE

(54) Title
Structure learning in convolutional neural networks

(51) International Patent Classification(s)
G06N 3/02 (2006.01)

(21) Application No: **2017230184** (22) Date of Filing: **2017.03.13**

(87) WIPO No: **WO17/156547**

(30) Priority Data

(31) Number	(32) Date	(33) Country
62/307,071	2016.03.11	US

(43) Publication Date: **2017.09.14**
(44) Accepted Journal Date: **2021.10.07**

(71) Applicant(s)
Magic Leap, Inc.

(72) Inventor(s)
Rabinovich, Andrew;Badrinarayanan, Vijay;Detone, Daniel;Rajendran, Srivignesh;Lee, Douglas Bertram;Malisiewicz, Tomasz J.

(74) Agent / Attorney
Davies Collison Cave Pty Ltd, Level 14 255 Elizabeth St, Sydney, NSW, 2000, AU

(56) Related Art
US 6138109 A
WO 2005022343 A2

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

(19) **World Intellectual Property Organization**
International Bureau

(43) **International Publication Date**
14 September 2017 (14.09.2017)

WIPO | PCT

(10) **International Publication Number**
WO 2017/156547 A1

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

(21) **International Application Number**:
PCT/US2017/022206

(22) **International Filing Date**:
13 March 2017 (13.03.2017)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:
62/307,071 11 March 2016 (11.03.2016) US

(71) **Applicant**: **MAGIC LEAP, INC.** [US/US]; 1855 Griffin Road, Suite B454, Dania Beach, Florida 33004 (US).

(72) **Inventors**: **RABINOVICH, Andrew**; 1855 Griffin Road, Suite B454, Dania Beach, Florida 33004 (US). **BADRINARAYANAN, Vijay**; 222 Escuela Avenue, Apt. 172, Mountain View, California 94040 (US). **DETONE, Daniel**; 955 Wisconsin Street, Unit 4, San Francisco, California 94107 (US). **RAJENDRAN, Srivignesh**; 100 E. Middlefield Road, Apt. 5C, Mountain View, California 94043 (US). **LEE, Douglas, Bertram**; 630 Bair Island Rd., Unit 105, Redwood City, California 94063 (US). **MALISIEWICZ, Tomasz, J.**; 315 View St., Mountain View, California 94041 (US).

(74) **Agent**: **MEI, Peter C.**; Vista IP Law Group, LLP, 2160 Lundy Ave., Suite 230, San Jose, California 95131 (US).

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

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

(54) **Title**: STRUCTURE LEARNING IN CONVOLUTIONAL NEURAL NETWORKS

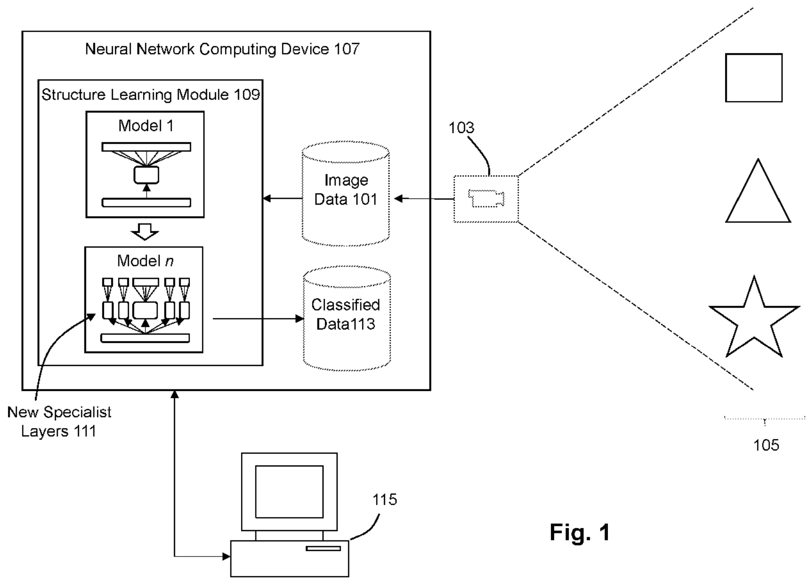
Neural Network Computing Device 107
Structure Learning Module 109
Model 1
Model *n*
Image Data 101
Classified Data113
103
New Specialist Layers 111
105
115

Fig. 1

(57) **Abstract**: The present disclosure provides an improved approach to implement structure learning of neural networks by exploiting correlations in the data/problem the networks aim to solve. A greedy approach is described that finds bottlenecks of information gain from the bottom convolutional layers all the way to the fully connected layers. Rather than simply making the architecture deeper, additional computation and capacitance is only added where it is required.

WO 2017/156547 A1

STRUCTURE LEARNING IN CONVOLUTIONAL NEURAL NETWORKS

FIELD OF THE INVENTION

[0001] This disclosure pertains to computing networks, and more particularly to neural networks configured to learn hierarchical representations from data.

BACKGROUND

[0002] Neural networks pertain to computational approaches that are loosely modeled after the neural structures of biological brain processing that can be used for solving complex computational problems. Neural networks are normally organized as a set of layers, where each layer includes a set of interconnected nodes that include various functions. Weighted connections implement functions that are processed within the network to perform various analytical operations. Learning processes may be employed to construct and modify the networks and the associated weights for the connectors within the network. By modifying the connector weights, this permits the network to learn over time from past analysis to improve future analysis results.

[0003] Neural networks may be employed to perform any appropriate type of data analysis, but is particularly suitable to be applied to complex analysis tasks such as pattern analysis and classification. Direct application of these techniques are therefore suitable, for example, to implement machine vision functions such as recognition and classification of specific objects and object classes from image data captured by digital imaging devices.

[0004] There are numerous types of neural networks that are known in the art. Deep neural networks is a type of neural network where deep learning techniques are applied to implement a cascade of many layers of nonlinear processing to perform analytical functions. Deep learning algorithms transform their inputs through more layers than shallow learning algorithms. At each layer, the signal is transformed by a processing unit, such as an artificial neuron, whose parameters are learned through training.

[0005] A convolutional neural network is a type of neural network where the connectivity pattern in the network is inspired by biological visual cortex functioning. Visual fields are constructed through the network, where the response of an individual artificial neuron to an input stimulus can be approximated mathematically by a convolution operation.

[0006] Convolutional deep neural networks have been implemented in the known art. LeNet (LeCun et al. (1998), AlexNet (Krizhevsky et al. (2012), GoogLeNet (Szegedy et al. (2015), and VGGNet (Simonyan & Zisserman (2015) are all examples of ConvNet architectures that implement different types of deep neural networks. These models are quite different (e.g., different depth, width, and activation functions). However, these models are all the same in one key respect – each one is a hand designed structure which embodies the architects' insights about the problem at hand.

[0007] These networks follow a relatively straightforward recipe, starting with a convolutional layer that learns low-level features resembling Gabor filters or some representations thereof. The later layers encode higher-level features such as object parts (parts of faces, cars, and so on). Finally, at the top, there is a layer that returns a probability distribution over classes. While there approach provide some structure, in the label space, to the output that is produced by a trained network, the issue is that this structure is seldom utilized when these networks are designed and trained.

[0008] Structure learning in probabilistic graphical models have been suggested, where the conventional algorithms for structure learning in deep convolutional networks typically fall into one of two categories: those that make the nets smaller, and those that make the nets better. One suggested approach focuses on taking unwieldy pretrained networks and squeezing them into networks with a smaller memory footprint, thus requiring fewer computational resources. This class of techniques follows the "teacher-student" paradigm where the goals is to create a student network which mimics the teacher. This means that one needs to start with both an Oracle architecture and its learned weights – training the student only happens later. When distilling an ensemble of specialists on very large datasets, the computationally expensive ensemble training step must be performed first.

[0009] Feng et al, "Learning the Structure of Deep Convolutional Networks" is an example of a technique for automatically learning aspects of the structure of a deep model. This approach uses an Indian Buffet Process to propose a new convolutional neural network model to identify a structure, where after the structure is determined, pruning is performed to create a more compact representation of the network. However, one drawback with this approach is that the number of layers remain static, where it is only the known individual layers within the static number of layers that is augmented to be more or less complex through the structure learning process. As such, this approach is unable to identify any new layers that may be needed to optimize the structure.

[0010] Therefore, there is a need for an improved approach to implement structure learning for convolutional neural networks.

SUMMARY

[0011] Some embodiments of the invention are directed to an improved approach to implement structure learning for neural networks. The approach starts out with a network, provides the network with a problem having labeled data, and then reviews the structure of the output produced by this network. The network's architecture is then modified to obtain a better solution for the specific problem. Rather than having experts come up with highly complicated and domain-specific network architectures, this approach allows the data to drive the architecture of the network that will be used for a specific task.

[0012] According to some embodiments, a neural network can be improved by (a) identifying the information gain bottleneck in its structure, (b) applying the structure of the predictions to alleviate the bottleneck, and finally (c) determining the depth of specialists pathways.

[0013] Some embodiments implement structure learning of neural networks by exploiting correlations in the data/problem the networks aim to solve, where a greedy approach is performed to find bottlenecks of information gain from the bottom convolutional layers all the way to the fully connected layers. In some embodiments, a network is created at an initial point in time, and a set of outputs are generated from the network when applied to a designated task, e.g., to perform image recognition/object classification tasks. Next, the various layers within the network model are analyzed to identify the lowest performing layer within the model. Additional structures are then injected into the model to improve the performance of the model. In particular, new specialist layers are inserted into the model at the identified vertical position to augment the performance of the model. Rather than just having one general purpose pathway to perform classification for multiple types of objects, a first new specialist layer may be added just to address classification of a first type of object and a second new specialist layer may be added just to address classification of a second type of object. By taking this action, over time, each of these specialist components becomes highly knowledgeable about its dedicated area of expertise, since the specialist is forced to learn extensive levels of detail about the specific subdomain assigned to that specialist component. In this way, the model is improved by adding new layers that will directly address areas of classification that have been specifically identified as being sub-optimal compared to other parts of the network. This same process continues through the rest of the model to identify any additional layers that should be modified and/or augmented.

[0014] In certain embodiments, a "loss" mechanism (e.g., a loss layer, a loss function, and/or cost function) is included at each layer of the network. Instead of just having a single top-level loss layer, additional loss layers are added to the other layer within the network, e.g., where a deep neural network has multiple loss layers at intermediate, and final, stages of feature extraction, where each loss layer measures the performance of the network up to that point in depth. Predictions can be generated at each loss layer and converted to the respective confusion matrice, forming a tensor T containing all confusion matrices for the network. By analyzing the structure of T and its elements, the aim is to modify and augment the existing structure of the network both in terms of depth and breadth. To maximize feature sharing and reduce computation on one hand, yet to increase accuracy on the other, the aim is to restructure the existing networks' structure. To do so, the approach partitions the networks' depth as well and breadth according its current performance. Therefore, vertical splitting is performed in some embodiments, e.g., by computing the dot product between the different layers. To partition the architecture in depth, some embodiments compare the neighboring subspaces that correspond to the consecutive loss function evaluations at neighboring layers. In addition, horizontal splitting is performed, e.g., by performing K-way Bifurcation. To improve the performance of the network at a particular layer, its structure (e.g., fully convolutional), may require augmentation. Parts of the network focus on general knowledge (generalist), while others concentrate on a small subset of labels that have high similarity among each other (specialist). Knowledge achieved by layer i will be used to perform the first horizontal partitioning of the network. The processing continues (e.g., in a recursive manner) until the top of the network is reached. At this point, the final model is stored into a computer readable medium.

[0015] Some embodiments pertain to the deep learning of the specialists. While the structure of the generalist is known to perform well on general knowledge, it is not guaranteed that this same structure will perform well in a specialist where the task of the specialist may require a more simple or complex representation. Some embodiments allow the structure of each specialist to deviate from the structure of the generalist via depth-wise splitting, in a data-driven manner.

[0016] Additional variations of these techniques may be applied in alternate embodiments. For example, for every pair of splits (vertical or horizontal), a network can be retrained to get classification at a given pathway. Techniques can be applied in certain embodiments for speeding this up and/or avoiding it at all, such as by agglomerative clustering and/or splitting. Further, given confusion matrix C_i, and its partitioning K, agglomerative clustering may be performed on each of the K parts of the C_i to estimate further splits. This leads to the cost X_u. Cost X_s is the cost of supervised grouping, learning new confusion matrices at high levels of the network. X_u is less than or equal to X_s + Tau, where Tau is the upper bound on the clustering error.

[0017] In some embodiments, variations are considered with respect to convolutional layer versus fully-connected (1x1 convolution). If splitting is required among the convolutional layers (even fully convolutional layers, such as in the case of semantic segmentation), then instead of changing the linear size of the layer (fc in this case), the depth of dimension may be changed to reflect the number of classes (this is the extension to FCN).

[0018] Further variations and embodiments may be produced using collapsing or adding or vertical layers per pathway, changing the size of layer as a function of label space, and/or extension to detection and RNN (unrolling in the same way by comparing confusions).

[0019] In yet another embodiment, techniques may be applied to identify when there may be too many layers in the network, such that fewer layers would be adequate for the required processing tasks. As noted above, one can reliably add depth to a network and see an improvement in performance given enough training data. However, this added boost in performance may come at a cost in terms of FLOPs and memory consumption. In some embodiments, the network is optimized with this tradeoff in mind with the usage of an *all-or-nothing highway network*, which learns whether or not a given layer of computation in the network is used via a binary decision. If a given computational block is used, a penalty is incurred. By varying this penalty term, one can customize the learning process with a target architecture in mind: an embedded system would prefer a much leaner architecture then a cloud-based system.

[0019A] In another embodiment there is provided a method implemented with a processor, comprising: creating a neural network; generating output from the neural network; identifying a low performing layer from the neural network, the low performing layer having a relatively lower performance than a performance of another layer in the neural network; inserting a new specialist layer at the low performing layer; and repeating the act of identifying and the act of inserting until a top of the neural network is reached.

2017230184 09 Oct 2020

[0019B] In another embodiment there is provided a system, comprising: a processor; a memory for holding programmable code; and wherein the programmable code includes instructions for creating a neural network; generating output from the neural network; identifying a low performing layer from the neural network, the low performing layer having a relatively lower performance than a performance of another layer in the neural network; inserting a new specialist layer at the low performing layer; and repeating the act of identifying and the act of inserting until a top of the neural network is reached.

[0019C] In another embodiment there is provided a computer program product embodied on a non-transitory computer readable medium, the non-transitory computer readable medium having stored thereon a sequence of instructions which, when executed by a processor causes the processor to execute a method comprising: creating a neural network; generating output from the neural network; identifying a low performing layer from the neural network, the low performing layer having a relatively lower performance than a performance of another layer in the neural network; inserting a new specialist layer at the low performing layer; and repeating the act of identifying and the act of inserting until a top of the neural network is reached.

[0020] Further details of aspects, features and advantages of the invention are described below in the detailed description, drawings, and claims. Both the foregoing general description and the following detailed description are exemplary and explanatory, and are not intended to be limiting as to the scope of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] The drawings illustrate the design and utility of various embodiments of the present invention. It should be noted that the figures are not drawn to scale and that elements of similar structures or functions are represented by like reference numerals throughout the figures. In order to better appreciate how to obtain the above-recited and other advantages and objects of various embodiments of the invention, a more detailed description of the present inventions briefly described above will be rendered by reference to specific embodiments thereof, which are illustrated in the accompanying drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

[0022] Fig. 1 illustrates an example system which may be employed in some embodiments of the invention to implement structure learning for neural networks.

[0023] Fig. 2 shows a flowchart of an approach to implement structure learning for neural networks according to some embodiments of the invention.

[0024] Fig. 3 illustrates a more detailed flowchart of an approach to implement structure learning for neural networks according to some embodiments.

[0025] Figs. 4A-4F illustrate various embodiments of the invention.

[0026] Figs. 5A-5B illustrate an approach to identify when there may be too many layers in the network.

[0027] Figs. 6A-6D illustrate general AR system component options for various embodiments.

[0028] Fig. 7 depicts a computerized system on which some embodiments of the invention can be implemented

DETAILED DESCRIPTION

[0029] Some embodiments of the invention are directed to an improved approach to implement structure learning for neural networks. The approach starts out with a network, provides the network with a problem having labeled data, and then reviews the structure of the output produced by this network. The network's architecture is then modified to obtain a better solution for the specific problem. Rather than having experts come up with highly complicated and domain-specific network architectures, this approach allows the data to drive the architecture of the network that will be used for a specific task.

[0030] Fig. 1 illustrates an example system which may be employed in some embodiments of the invention to implement structure learning for neural networks. The system may include one or more users that interface with and operate a computing system 107 or 115 to control and/or interact with the system. The system comprises any type of computing station that may be used to operate, interface with, or implement a neural network computing device 107 or user computing device 115. Examples of such computing systems include for example, servers, workstations, personal computers, or remote computing terminals connected to a networked or cloud-based computing platform. The computing system may comprise one or more input devices for the user to provide operational control over the activities of the system, such as a mouse or keyboard to manipulate a pointing object. The computing system may also be associated with a display device, such as a display monitor, for control interfaces and/or analysis results to users of the computing system.

[0031] In some embodiments, the system is employed to implement computer vision functionality. As such, the system may include one or more image capture devices, such as camera 103, to capture image data 101 for one or more objects 105 in the environment at which the system operates. The image data 101 and/or any analysis results (e.g., classification output data 113) may be stored in one or more computer readable storage mediums. The computer readable storage medium includes any combination of hardware and/or software that allows for ready access to the data that is located at the computer readable storage medium. For example, the computer readable storage medium could be implemented as computer memory and/or hard drive storage operatively managed by an operating system, and/or remote storage in a networked storage device, such as networked attached storage (NAS), storage area network (SAN), or cloud storage. The computer readable storage medium could also be implemented as an electronic database system having storage on persistent and/or non-persistent storage.

[0032] The neural network computing device 107 includes a structure learning module 109 to modify an original model 1 into an improved model *n*, where model *n* is the results of possibly multiple iterative processes to modify the layers within the model. The model *n* preferably includes a depth and breadth in knowledge, essentially a mixture of experts. The model should understand the difference between coarse categories, yet at the same time understands the difference for fine grained classes across various domains. New specialist layers 111 are added to the model as necessary to implement these goals. The design of such a system is governed by the constraint of adding resources solely where they are required. Simply expanding the network by making it arbitrarily deeper and wider does not scale due to computational constraints and thus the present approach avoids the need for extra regularization tricks.

[0033] Fig. 2 shows a flowchart of an approach to implement structure learning for neural networks according to some embodiments of the invention. The present approach implements structure learning of neural networks by exploiting correlations in the data/problem the networks aim to solve. A greedy approach is described that finds bottlenecks of information gain from the bottom convolutional layers all the way to the fully connected layers. Rather than simply making the architecture deeper arbitrarily, additional computation and capacitance is only added where it is required.

[0034] At 131, a network is created at an initial point in time. Any suitable approach can be used to create the network. For example, conventional Alexnet or Googlenet approaches may be used to generate the network.

[0035] Next, at 133, a set of outputs are generated from the network when applied to a designated task, e.g., to perform image recognition/object classification tasks. For example, assume that a number of people and animals are within an environment, and the assigned task is to analyze the image data to classify the different people and types of animals that can be observed within the environment. Each layer of the model provides certain outputs for the activities performed within that layer. The output has certain structures to it which can be reviewed to ascertain relationships between classes in the classification problem being solved.

[0036] At 135, the various layers within the network model are analyzed to identify the lowest performing layer within the model. For example, assume a model having ten layers, where the layers from layers 1 through 3 and layers 5 through 10 each provide a 10% improvement in classification accuracy, but layer 4 only provides a 1% improvement. In this situation, layer 4 would be identified as the lowest performing layer.

[0037] Next, at 137, additional structures are injected into the model to improve the performance of the model. In particular, new specialist layers are inserted into the model at the identified vertical position to augment the performance of the model.

[0038] To explain this aspect of the inventive embodiment, assume that the model is intended to perform classifications of the people and animals in the environment as illustrated in Fig. 4A. Here, the image capture device captures images of different people (e.g., a woman 401, a man 403, and a child 405). In addition, the environment includes multiple animals (e.g., a cat 407, a dog 409, and a mouse 411). Further assume that the existing model is able to successfully distinguish the people (401, 403, 405) from the animals (407, 409, 411), but appears to have a more difficult time distinguishing the different people from each other or distinguishing the different types of animals from one another. If one reviews the actual structure that can be learned from a network (e.g., an Oracle network), it is clear that the network includes learning dependence between the predictions that is being made. However, in traditional deep-learning architecture design, this is not utilized. If one looks even closer at this structure, it is evident that the system is learning concepts that are actually visually similar to one another. Referring to Fig. 4B, an example scatter plot in 3D of classes is shown to illustrate an example structure of predictions for a fully-trained AlexNet, clustered into multiple groups. The distance between points corresponds to the visual similarity between concepts. Here, it can be seen that there is a first tight clustering of the points relative to the people objects and a second tight clustering of points relative to the animal objects. It is this phenomena that may contributes to difficulties in a model being able to distinguish one person from another or one animal from another.

[0039] In this situation in some embodiments of the invention, rather than just having one general purpose pathway to perform classification for all of these types of objects, a first new specialist layer may be added just to address classification of people and a second new specialist layer may be added just to address classification of animals. One specialist (people specialist layer) would therefore be assigned to handle data for portion 413 of the chart in Fig. 4B while the second specialist (animal specialist layer) would be assigned to handle data for portion 415 in Fig. 4B. By taking this action, over time, each of these specialist components becomes highly knowledgeable about its dedicated area of expertise, since the specialist is forced to learn extensive levels of detail about the specific subdomain assigned to that specialist component. In this way, the model is improved by adding new layers that will directly address areas of classification that have been specifically identified as being sub-optimal compared to other parts of the network.

[0040] This same process continues through the rest of the model to identify any additional layers that should be modified and/or augmented. Therefore, a determination is made at 139 whether the processing has reached the top of the network. If so, then the model is finalized at 141. If not, then the process returns back to 133 to continue the process until the top of the network is reached.

[0041] This approach can be taken to modify and improve the architecture of any off-the-shelf convolutional neural network. By following the inventive approach of the present disclosure, any neural network can be improved by (a) identifying the information gain bottleneck in its structure, (b) applying the structure of the predictions to alleviate the bottleneck, and finally (c) determining the depth of specialists pathways.

[0042] Fig. 3 illustrates a more detailed flowchart of an approach to implement structure learning for neural networks according to some embodiments. For the purposes of this flow, assume that a network (e.g., a monolithic network) has already been created pursuant to any suitable approach such as Alexnet or Googlenet.

[0043] At 151, a "loss" mechanism (e.g., a loss layer, a loss function, and/or cost function) is included at each layer of the network. A loss mechanism corresponds to a function that maps an event or value to a representation of a cost or error value associated with processing within the neural network. As shown in Fig. 4C, instead of just having a single top-level loss layer 421, additional loss layers 423 are added to the other layer within the network. Therefore, this figure shows an example of a deep neural network with multiple loss layers at intermediate, and final, stages of feature extraction, where each loss layer measures the performance of the network up to that point in depth. Recall that the goal is to augment and modify the network architecture to solve a given problem as best as possible by modifying its architecture to best fit the task. Therefore, the approach analyzes the predictions, formed at the various loss layers throughout the network, and groups neuron activations based on the confusion between them.

[0044] As illustrated in Figs. 4D and 4E, predictions generated at each loss layer and converted to the respective confusion matrices (as shown in Fig. 4D), forming a tensor T containing all confusion matrices for the network, e.g., Oracle network (as shown in Fig. 4E). By analyzing the structure of T and its elements, the aim is to modify and augment the existing structure of the network both in terms of depth and breadth.

[0045] To explain, let Ci be the confusion matrix of classes and loss layer i, then:

$$A_i = C_i C_i^{\mathrm{T}} \tag{1}$$

$$D(i,i) = \sum_{i=1}^{I} A(i,:) \tag{2}$$

$$L_i = D_i^{-1/2} A_i D_i^{1/2} \tag{3}$$

$$\hat{C}_i = \mathrm{top}_{N_E}(eig(L_i)), \tag{4}$$

where A_i is the affinity matrix at loss layer i, D_i is the diagonal matrix, L_i is the graph Laplacian, and $\hat{C}_i$ is a subspace spanned by the leading eigenvectors of the graph Laplacian of the affinity matrix produced by C_i. Consequently, tensor:

$\hat{T} = \{\hat{C}_1, \hat{C}_2, \hat{C}_3, \ldots, \hat{C}_n\}$.

[0046] To maximize feature sharing and reduce computation on one hand, yet to increase accuracy on the other, the aim is to restructure the existing networks' structure. To do so, the approach partitions the networks' depth as well and breadth according its current performance.

[0047] Therefore, at 153, vertical splitting is performed, e.g., by computing the dot product between the different layers. To partition the architecture in depth, some embodiments compare the neighboring subspaces that correspond to the consecutive loss function evaluations at neighboring layers using the following equation:

$$\Phi(i, i+1) = 1 - \frac{1}{N_E} \|\hat{C}_{i+1}^{\mathrm{T}} \hat{C}_i\|_{\mathcal{F}}^2, \tag{5}$$

[0048] Here, $\hat{C}_i$ and $\hat{C}_{i+1}$ denote the approximate leading eigenvectors of the confusion matrices for loss functions at levels i and $i + 1$, and $\mathcal{F}$ denotes the Frobenius norm. Formally, $\hat{C}_i$ and $\hat{C}_{i+1}$ represent N_E-dimensional subspaces and ϕ $(i, i+1)$ is the normalized complement angle between them. It is important to note that this measure ϕ only depends on the subspace spanned by the columns of $\hat{C}_i$ and $\hat{C}_{i+1}$ and thus is invariant to rotations of the eigenvectors. Also, ϕ is constrained within [0, 1], with levels i and $i + 1$ are deemed similar in structure if ϕ $(i, i+1)$ is close to zero and is exactly 1 when ϕ $(i, i+1)$ are orthogonal. To construct a complete similarity relation between levels of scale space, all neighboring pairs of losses layers are compared using ϕ. With the established similarity relations it is now possible to address the problem of partitioning the monolithic network architecture.

[0049] Let ϕ be the vector of all sequential pairs of i and $i+1$, where $\phi_i = \phi$ $(i, i+1)$. Values of ϕ_i closest to zero indicate the lowest information gain between layers i and $i+1$. Thus, argmin(ϕ) is the optimal initial split of the monolithic architecture. Splitting the architecture in depth facilitates feature sharing while identifying points of redundancy (zero information gain).

[0050] At 155, horizontal splitting is performed, e.g., by performing K-way Bifurcation. To improve the performance of the network at a particular layer, its structure (e.g., fully convolutional), may require augmentation. Parts of the network focus on general knowledge (generalist), while others concentrate on a small subset of labels that have high similarity among each other (specialist). Knowledge achieved by layer i will be used to perform the first horizontal partitioning of the network.

[0051] Formally, given C_i, compute L_i as per Equations (1), (2), and (3) as disclosed above. An Eigengap is determined by analyzing the leading eigenvalues of the graph Laplacian L_i to determine the number of new pathways (specialists). Original data is projected onto the top N leading eigengectors of L_i; in R^N, the data is further clustered into k classes, where k equals the Eigengap. An example of such projection and grouping is illustrated in Fig. 4B. This procedure will lead to the modification of the architecture as shown in Fig. 4F, which illustrates a network 407 after the first split.

[0052] Once the first split has been established, then all new pathways are treated as the original network. The splitting procedure are applied until no more labels are left to split or 100% accuracy is achieved.

[0053] At 157, the above processing continues (e.g., in a recursive manner) until the top of the network is reached. At this point, the final model is stored into a computer readable medium.

[0054] This portion of the disclosure pertains to the deep learning of the specialists. While the structure of the generalist is known to perform well on general knowledge, it is not guaranteed that this same structure will perform well in a specialist where the task of the specialist may require a more simple or complex representation. Some embodiments allow the structure of each specialist to deviate from the structure of the generalist via depth-wise splitting, in a data-driven manner.

[0055] Let $L = \{L_1, L_2, \ldots., L_n\}$ be a set of fully-connected layers to be considered for further splitting. Consider a layer L_i in L that produces an output y. One can write the transformation that it applies to its input as $y = \sigma(f(x))$, where $\sigma(\ )$ applies a non-linearity such as ReLU and $f(x) = Wx$ where W is a learned weight matrix of dimensions $M \times N$ and x is input to this layer having dimensions $N \times 1$. To perform a split, the approach decomposes the transformation of L_i into $y = \sigma_1(g(\sigma_2(h(x)))$, where $\sigma_1(\ )$ and $\sigma_2(\ )$ are activation functions and $g(x) = W_1x$ and $h(x) = W_2x$ in which W_1 has dimensions $N \times N$ and W_2 has dimensions $M \times N$. The approach chooses:

$$\sigma_1(x) = \sigma(x) \tag{6}$$

$$g(x) = U\sqrt{\Sigma}x \tag{7}$$

$$\sigma_2(x) = Ix \tag{8}$$

$$h(x) = \sqrt{\Sigma}V^T x \tag{9}$$

[0056] Here, $W = U\,\Sigma V^T$ is the SVD factorization of W and I is the identity matrix. With this change, the transformation of layer L_i is unchanged. To increase the complexity of the learned representation of L_i one could set σ_2 as a non-linear activation function, such as ReLU. However, adding this non-linearity causes an abrupt change in the learned representation of L_i and may cause the network to restart much of its learning from scratch. Instead, one can insert a PReLU non-linearity and initialize its single parameter a to be 1, which is equivalent to I in equation 8. This provides the specialist with a smooth mechanism for introducing a new non-linearity at this layer.

[0057] Given the set of layers L, one can apply the above strategy to each layer L_i independently and greedily choose the split which provides the best improvement on the training loss. This process can be repeated recursively to our the set of layers $L_{new} = \{L_1, L_2, \ldots, L_n, L_{n+1}\}$.

[0058] Additional variations of these techniques may be applied in alternate embodiments. For example, for every pair of splits (vertical or horizontal), a network can be retrained to get classification at a given pathway. Techniques can be applied in certain embodiments for speeding this up and/or avoiding it at all, such as by agglomerative clustering and/or splitting. Further, given confusion matrix C_i, and its partitioning K, agglomerative clustering may be performed on each of the K parts of the C_i to estimate further splits. This leads to the cost X_u. Cost X_s is the cost of supervised grouping, learning new confusion matrices at high levels of the network. X_u is less than or equal to X_s + Tau, where Tau is the upper bound on the clustering error.

[0059] In some embodiments, variations are considered with respect to convolutional layer versus fully-connected (1x1 convolution). If splitting is required among the convolutional layers (even fully convolutional layers, such as in the case of semantic segmentation), then instead of changing the linear size of the layer (fc in this case), the depth of dimension may be changed to reflect the number of classes (this is the extension to FCN).

[0060] Further variations and embodiments may be produced using collapsing or adding or vertical layers per pathway, changing the size of layer as a function of label space, and/or extension to detection and RNN (unrolling in the same way by comparing confusions).

[0061] In yet another embodiment, techniques may be applied to identify when there may be too many layers in the network, such that fewer layers would be adequate for the required processing tasks. As noted above, one can reliably add depth to a network and see an improvement in performance given enough training data. However, this added boost in performance may come at a cost in terms of FLOPs and memory consumption. In some embodiments, the network is optimized with this tradeoff in mind with the usage of an *all-or-nothing highway network*, which learns whether or not a given layer of computation in the network is used via a binary decision. If a given computational block is used, a penalty is incurred. By varying this penalty term, one can customize the learning process with a target architecture in mind: an embedded system would prefer a much leaner architecture then a cloud-based system.

[0062] The issue addressed by this embodiment is to determine how deep a network should be given a computational budget for a given problem X. With the approach of using an *all-or-nothing highway network*, highway networks introduce a mixing matrix to learn how the skip connection from the previous layer should be transformed before mixing with the output of the current computational block. Consider the following equation:

$$y = F(x, W_i) + W_s x \tag{10}$$

[0063] Residual networks can find success in using the identity mapping to combine the skip connection. Although the identity mapping is less representative, it is more efficient and easier to optimize:

$$y = F(x, W_i) + x \tag{11}$$

[0064] The current approach instead parameterize a mixing matrix by a single scalar α which gates the output of the computational block (see Fig. 5A):

$$y = \alpha F(x, W_i) + x \tag{12}$$

[0065] When $\alpha = 0$, the $y = x$ and the input is simply passed to the output. When $\alpha = 1$, (eqn 12) becomes (eqn 10) and a residual unit is used for computation.

[0066] Fig. 5A illustrates a chart 501 for a network with an all-or-nothing highway connection. In this figure, a computational block is fed an input and later joined via a residual connection (elementwise addition). Before the addition, the output of the computation block is scaled by a learned parameter α which penalizes the use of this computational block. This loss is described below.

[0067] Learning is performed to determine whether or not to use a computation block. It is desirable to impose a prior on the α parameter which controls the behavior of a given layer in a deep network, and optimize this parameter jointly with the model parameters and its objective function. During training, it is desirable to encourage a binary decision for α to choosing either 0 or 1 for each depth independently. If a computational block is learned to be skipped, then one can simply remove that computation block from the model at inference time.

[0068] In a residual network, consecutive layers in general have small mappings, where the learned residual functions in general have small responses, suggesting that identity mappings provide reasonable preconditioning. This suggests that transitioning between an identity map in (eqn 10) and an identity layer and vice versa should not cause catastrophic change in the objective function. Thus the present approach introduces a piecewise smooth loss function on the α parameter which gates the output of the computational block at various depths.

[0069] In addition, it is desirable to parameterize the loss function on the α parameters such that for a different scenarios, a higher penalty is assigned to models which use more computation. In the case a light embedded platform such as smartphone, one might want a high penalty for choosing a layer. In the case of a cloud computing platform, no such penalty for using a computation block might be wanted. Given these criteria, one can use the piecewise smooth polynomial/linear function shown in Fig. 5B, which can be parameterized by the following:

```
if x < 0.:
        y = (np.absolute(x) * self.steepness)
elif x > 1.:
        y = (x - 1.) * self.steepness + + self.peak*0.125
elif x < 0.5:
        y = -self.peak * (x**2. - x)
else:
        y = -self.peak/2. * (x**2. - x) + self.peak*0.125
```

[0070] For various selections of the peak shown in Fig. 5B, a varying usage penalty is given to the model.

AGUMENTED REALITY AND COMPUTING SYSTEMS ARCHITECTURE(S)

[0071] The above-described techniques are particularly applicable to machine vision applications for virtual reality and augmented reality systems. The inventive neural network classification device may be implemented independently of AR systems, but many embodiments below are described in relation to AR systems for illustrative purposes only.

[0072] Disclosed are devices, methods and systems for classification and recognition for various computer systems. In one embodiment, the computer system may be a head-mounted system configured to facilitate user interaction with various other computer systems (e.g., financial computer systems). In other embodiments, the computer system may be a stationary device (e.g., a merchant terminal or an ATM) configured to facilitate user financial transactions. Various embodiments will be described below in the context of an AR system (e.g., head-mounted), but it should be appreciated that the embodiments disclosed herein may be used independently of any existing and/or known AR systems.

[0073] Referring now to Figs. 6A-6D, some general AR system component options are illustrated according to various embodiments. It should be appreciated that although the embodiments of Figs. 6A-6D illustrate head-mounted displays, the same components may be incorporated in stationary computer systems as well, and Figs. 6A-6D should not be seen as limiting.

[0074] As shown in Fig. 6A, a head-mounted device user 60 is depicted wearing a frame 64 structure coupled to a display system 62 positioned in front of the eyes of the user 60. The frame 64 may be permanently or temporarily coupled to one or more user identification specific sub systems depending on the required level of security. A speaker 66 may be coupled to the frame 64 in the depicted configuration and positioned adjacent the ear canal of the user 60. In an alternative embodiment, another speaker (not shown) is positioned adjacent the other ear canal of the user 60 to provide for stereo/shapeable sound control. In one or more embodiments, the user identification device may have a display 62 that is operatively coupled, such as by a wired lead or wireless connectivity, to a local processing and data module 70, which may be mounted in a variety of configurations, such as fixedly attached to the frame 64, fixedly attached to a helmet or hat 80 as shown in the embodiment depicted in Fig. 6B, embedded in headphones, removably attached to the torso 82 of the user 60 in a backpack-style configuration as shown in the embodiment of Fig. 6C, or removably attached to the hip 84 of the user 60 in a belt-coupling style configuration as shown in the embodiment of Fig. 6D.

[0075] The local processing and data module 70 may comprise a power-efficient processor or controller, as well as digital memory, such as flash memory, both of which may be utilized to assist in the processing, caching, and storage of data. The data may be captured from sensors which may be operatively coupled to the frame 64, such as image capture devices (such as cameras), microphones, inertial measurement units, accelerometers, compasses, GPS units, radio devices, and/or gyros. Alternatively or additionally, the data may be acquired and/or processed using the remote processing module 72 and/or remote data repository 74, possibly for passage to the display 62 after such processing or retrieval. The local processing and data module 70 may be operatively coupled 76, 78, such as via a wired or wireless communication links, to the remote processing module 72 and the remote data repository 74 such that these remote modules 72, 74 are operatively coupled to each other and available as resources to the local processing and data module 70.

[0076] In one embodiment, the remote processing module 72 may comprise one or more relatively powerful processors or controllers configured to analyze and process data and/or image information. In one embodiment, the remote data repository 74 may comprise a relatively large-scale digital data storage facility, which may be available through the internet or other networking configuration in a "cloud" resource configuration. In one embodiment, all data is stored and all computation is performed in the local processing and data module, allowing fully autonomous use from any remote modules.

[0077] In some embodiments, identification devices (or AR systems having identification applications) similar to those described in Figs. 6A-6D provide unique access to a user's eyes. Given that the identification/AR device interacts crucially with the user's eye to allow the user to perceive 3-D virtual content, and in many embodiments, tracks various biometrics related to the user's eyes (e.g., iris patterns, eye vergence, eye motion, patterns of cones and rods, patterns of eye movements, etc.), the resultant tracked data may be advantageously used in identification applications. Thus, this unprecedented access to the user's eyes naturally lends itself to various identification applications.

[0078] Fig. 7 is a block diagram of an illustrative computing system 1400 suitable for implementing an embodiment of the present invention. Computer system 1400 includes a bus 1406 or other communication mechanism for communicating information, which interconnects subsystems and devices, such as processor 1407, system memory 1408 (e.g., RAM), static storage device 1409 (e.g., ROM), disk drive 1410 (e.g., magnetic or optical), communication interface 1414 (e.g., modem or Ethernet card), display 1411 (e.g., CRT or LCD), input device 1412 (e.g., keyboard), and cursor control.

[0079] According to one embodiment of the invention, computer system 1400 performs specific operations by processor 1407 executing one or more sequences of one or more instructions contained in system memory 1408. Such instructions may be read into system memory 1408 from another computer readable/usable medium, such as static storage device 1409 or disk drive 1410. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and/or software. In one embodiment, the term "logic" shall mean any combination of software or hardware that is used to implement all or part of the invention.

[0080] The term "computer readable medium" or "computer usable medium" as used herein refers to any medium that participates in providing instructions to processor 1407 for execution. Such a medium may take many forms, including but not limited to, non-volatile media and volatile media. Non-volatile media includes, for example, optical or magnetic disks, such as disk drive 1410. Volatile media includes dynamic memory, such as system memory 1408.

[0081] Common forms of computer readable media include, for example, floppy disk, flexible disk, hard disk, magnetic tape, any other magnetic medium, CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, RAM, PROM, EPROM, FLASH-EPROM, any other memory chip or cartridge, or any other medium from which a computer can read.

[0082] In an embodiment of the invention, execution of the sequences of instructions to practice the invention is performed by a single computer system 1400. According to other

2017230184 09 Oct 2020

embodiments of the invention, two or more computer systems 1400 coupled by communication link 1415 (e.g., LAN, PTSN, or wireless network) may perform the sequence of instructions required to practice the invention in coordination with one another.

[0083] Computer system 1400 may transmit and receive messages, data, and instructions, including program, e.g., application code, through communication link 1415 and communication interface 1414. Received program code may be executed by processor 1407 as it is received, and/or stored in disk drive 1410, or other non-volatile storage for later execution. Computer system 1400 may communicate through a data interface 1433 to a database 1432 on an external storage device 1431.

[0084] In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. For example, the above-described process flows are described with reference to a particular ordering of process actions. However, the ordering of many of the described process actions may be changed without affecting the scope or operation of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than restrictive sense.

[0085] Throughout this specification and the claims which follow, unless the context requires otherwise, the word "comprise", and variations such as "comprises" or "comprising", will be understood to imply the inclusion of a stated integer or step or group of integers or steps but not the exclusion of any other integer or step or group of integers or steps.

[0086] The reference in this specification to any prior publication (or information derived from it), or to any matter which is known, is not, and should not be taken as, an acknowledgement or admission or any form of suggestion that that prior publication (or information derived from it) or known matter forms part of the common general knowledge in the field of endeavour to which this specification relates.

2017230184 09 Oct 2020

CLAIMS

The claims defining the invention are as follows:

1. A method implemented with a processor, comprising:

creating a neural network;

generating output from the neural network;

identifying a low performing layer from the neural network, the low performing layer having a relatively lower performance than a performance of another layer in the neural network;

inserting a new specialist layer at the low performing layer; and

repeating the act of identifying and the act of inserting until a top of the neural network is reached.

2. The method of claim 1, further comprising updating a model for the neural network to obtain an updated model, wherein the updated model comprises the new specialist layer, and at least one generalist layer.

3. The method of claim 1 or 2, wherein the new specialist layer is configured to handle a specific subdomain of data distinct from a subdomain handled by another specialist layer.

4. The method of any one of claims 1 to 3, wherein a plurality of loss layers are added to the neural network.

5. The method of claim 4, further comprising generating predictions at one of the loss layers, and converting the predictions to one or more confusion matrices forming a tensor T.

6. The method of claim 5, wherein a structure of T is analyzed to modify and augment an existing structure of the neural network both in terms of depth and breadth.

7. The method of any one of claims 1 to 6, wherein the neural network undergoes both vertical splitting and horizontal splitting.

8. The method of claim 7, wherein K-way Bifurcation is performed to implement the horizontal splitting.

9. The method of any one of claims 1 to 8, wherein each layer of the neural network is addressed independently, and a given layer of the neural network undergoes splitting by

performing a greedy choice to split the given layer which provides a best improvement on a training loss.

10. The method of any one of claims 1 to 9, wherein an all-or-nothing highway network is employed to identify layers in the neural network to be removed.

11. The method of any one of claims 1 to 10, wherein the neural network is employed to classify images captured for a virtual realty or augmented reality system.

12. A system, comprising:

a processor;

a memory for holding programmable code; and

wherein the programmable code includes instructions for creating a neural network;

generating output from the neural network; identifying a low performing layer from the neural network, the low performing layer having a relatively lower performance than a performance of another layer in the neural network; inserting a new specialist layer at the low performing layer; and repeating the act of identifying and the act of inserting until a top of the neural network is reached.

13. The system of claim 12, wherein the processor is configured to update a model for the neural network to obtain an updated model, wherein the updated model comprises the new specialist layer and at least one generalist layer.

14. The system of claim 12 or 13, wherein the new specialist layer is configured to handle a specific subdomain of data distinct from a subdomain handled by another specialist layer.

15. The system of any one of claims 12 to 14, wherein the neural network comprises a plurality of loss layers.

16. The system of claim 15, wherein the processor is configured to generate predictions at one of the loss layers, and to convert the predictions to one or more confusion matrices forming a tensor T.

17. The system of claim 16, wherein a structure of T is analyzed to modify and augment an existing structure of the neural network both in terms of depth and breadth.

18. The system of any one of claims 12 to 17, wherein the neural network is configured to undergo both vertical splitting and horizontal splitting.

19. The system of claim 18, wherein the horizontal splitting is implemented using K-way Bifurcation.

20. The system of any one of claims 12 to 19, wherein the processor is configured to address each layer of the neural network independently, and to cause a given layer of the neural network to undergo splitting by performing a greedy choice to split the given layer which provides a best improvement on a training loss.

21. The system of any one of claims 12 to 20, wherein the processor is configured to employ an all-or-nothing highway network to identify layers in the neural network to be removed.

22. The system of any one of claims 12 to 21, wherein the neural network is configured to classify images captured for a virtual realty or augmented reality system.

23. A computer program product embodied on a non-transitory computer readable medium, the non-transitory computer readable medium having stored thereon a sequence of instructions which, when executed by a processor causes the processor to execute a method comprising:

creating a neural network;

generating output from the neural network;

identifying a low performing layer from the neural network, the low performing layer having a relatively lower performance than a performance of another layer in the neural network;

inserting a new specialist layer at the low performing layer; and

repeating the act of identifying and the act of inserting until a top of the neural network is reached.

24. The computer program product of claim 23, wherein the method further comprises updating a model for the neural network to obtain an updated model, wherein the updated model comprises the new specialist layer and at least one generalist layer.

25. The computer program product of claim 23 or 24, wherein in the method, the new specialist layer is configured to handle a specific subdomain of data distinct from a subdomain handled by another specialist layer.

2017230184 09 Oct 2020

26. The computer program product of any one of claims 23 to 25, wherein in the method, a plurality of loss layers are added to the neural network.

27. The computer program product of claim 26, wherein the method further comprises generating predictions at one of the loss layers, and converting the predictions to one or more confusion matrices forming a tensor T.

28. The computer program product of claim 27, wherein a structure of T is analyzed in the method to modify and augment an existing structure of the neural network both in terms of depth and breadth.

29. The computer program product of any one of claims 23 to 28, wherein in the method, the neural network undergoes both vertical splitting and horizontal splitting.

30. The computer program product of claim 29, wherein K-way Bifurcation is performed in the method to implement the horizontal splitting.

31. The computer program product of any one of claims 23 to 30, wherein each layer of the neural network is addressed independently in the method, and a given layer of the neural network undergoes splitting by performing a greedy choice to split the given layer which provides a best improvement on a training loss.

32. The computer program product of any one of claims 23 to 31, wherein an all-or-nothing highway network is employed in the method to identify layers in the neural network to be removed.

33. The computer program product of any one of claims 23 to 32, wherein the neural network is employed in the method to classify images captured for a virtual realty or augmented reality system.

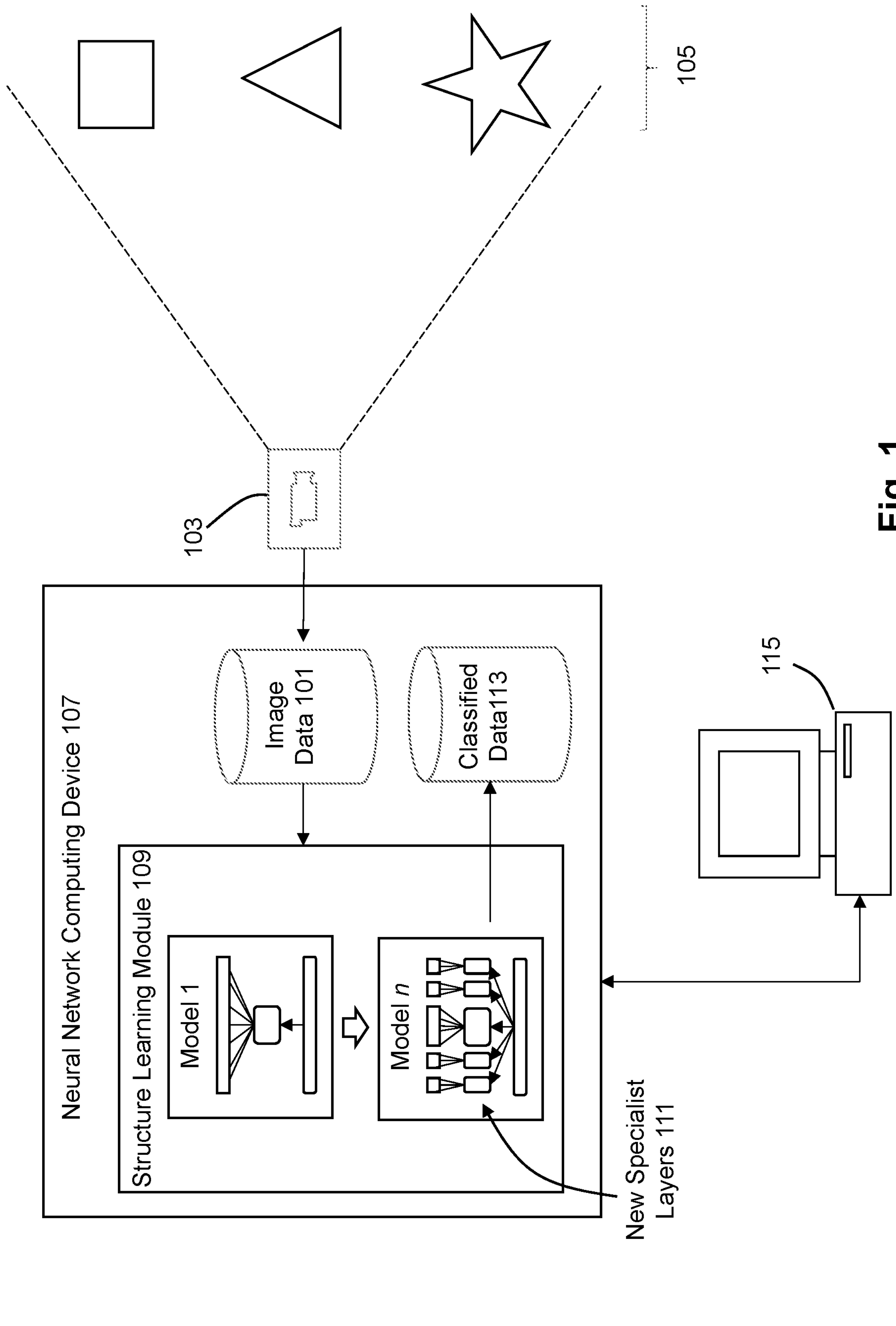
105
103
Neural Network Computing Device 107
Image Data 101
Classified Data113
Structure Learning Module 109
Model 1
Model *n*
New Specialist Layers 111
115

Fig. 1

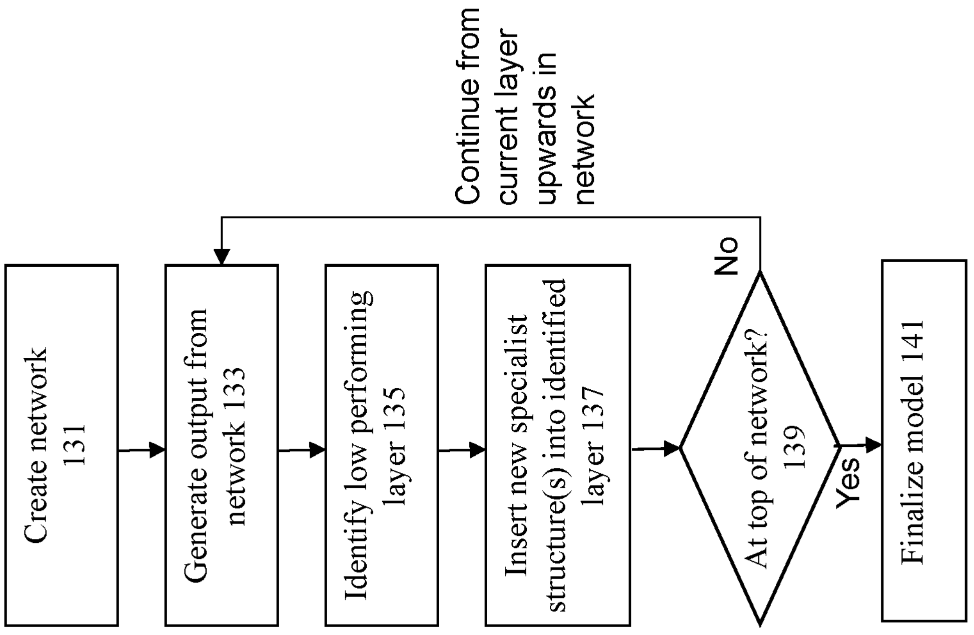
Create network 131
Generate output from network 133
Identify low performing layer 135
Insert new specialist structure(s) into identified layer 137
At top of network? 139
No
Continue from current layer upwards in network
Yes
Finalize model 141

Fig. 2

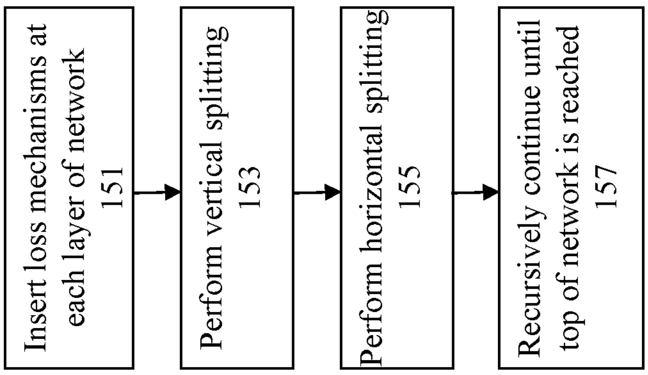
Insert loss mechanisms at each layer of network 151
Perform vertical splitting 153
Perform horizontal splitting 155
Recursively continue until top of network is reached 157

Fig. 3

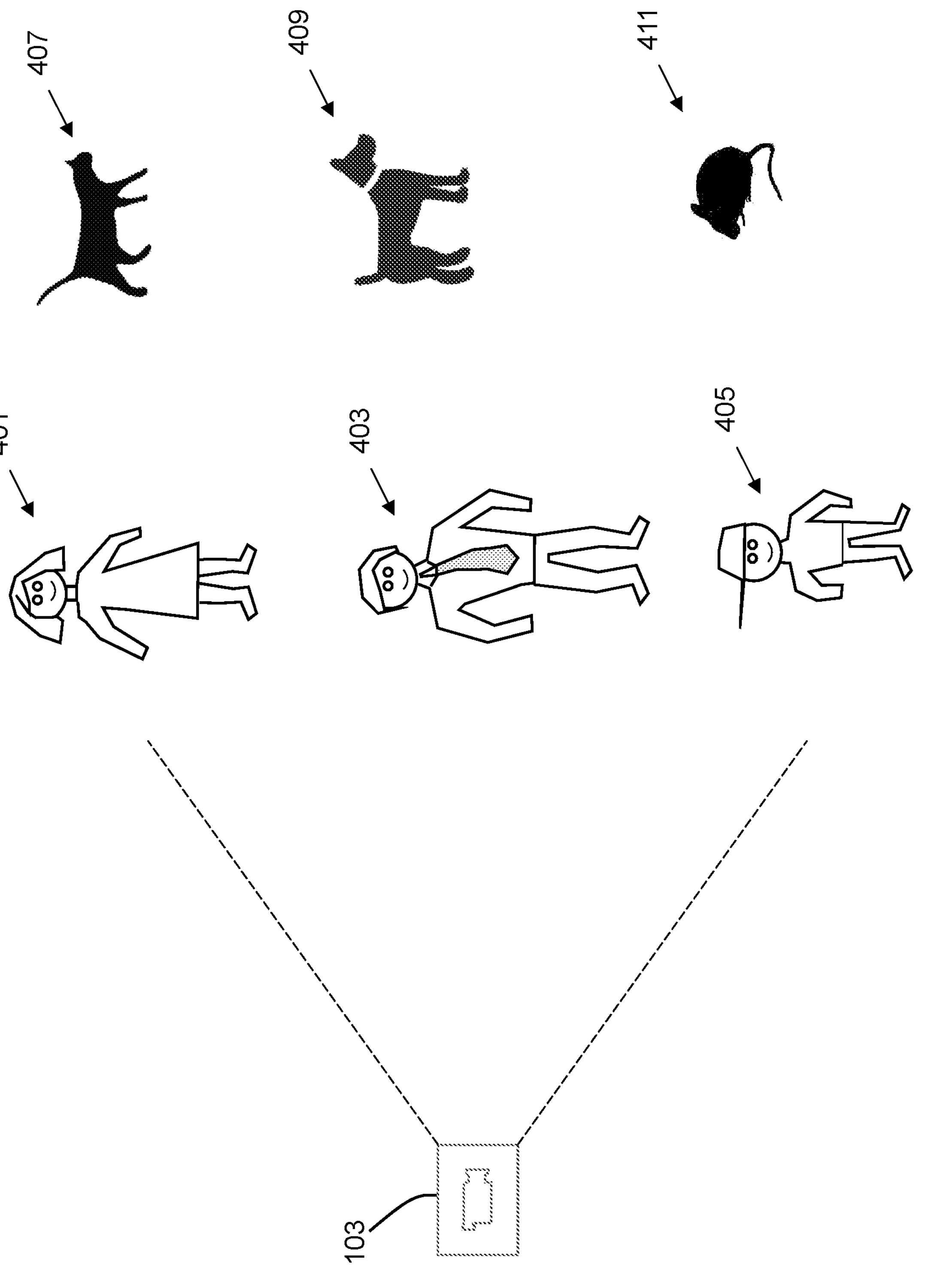

Fig. 4A

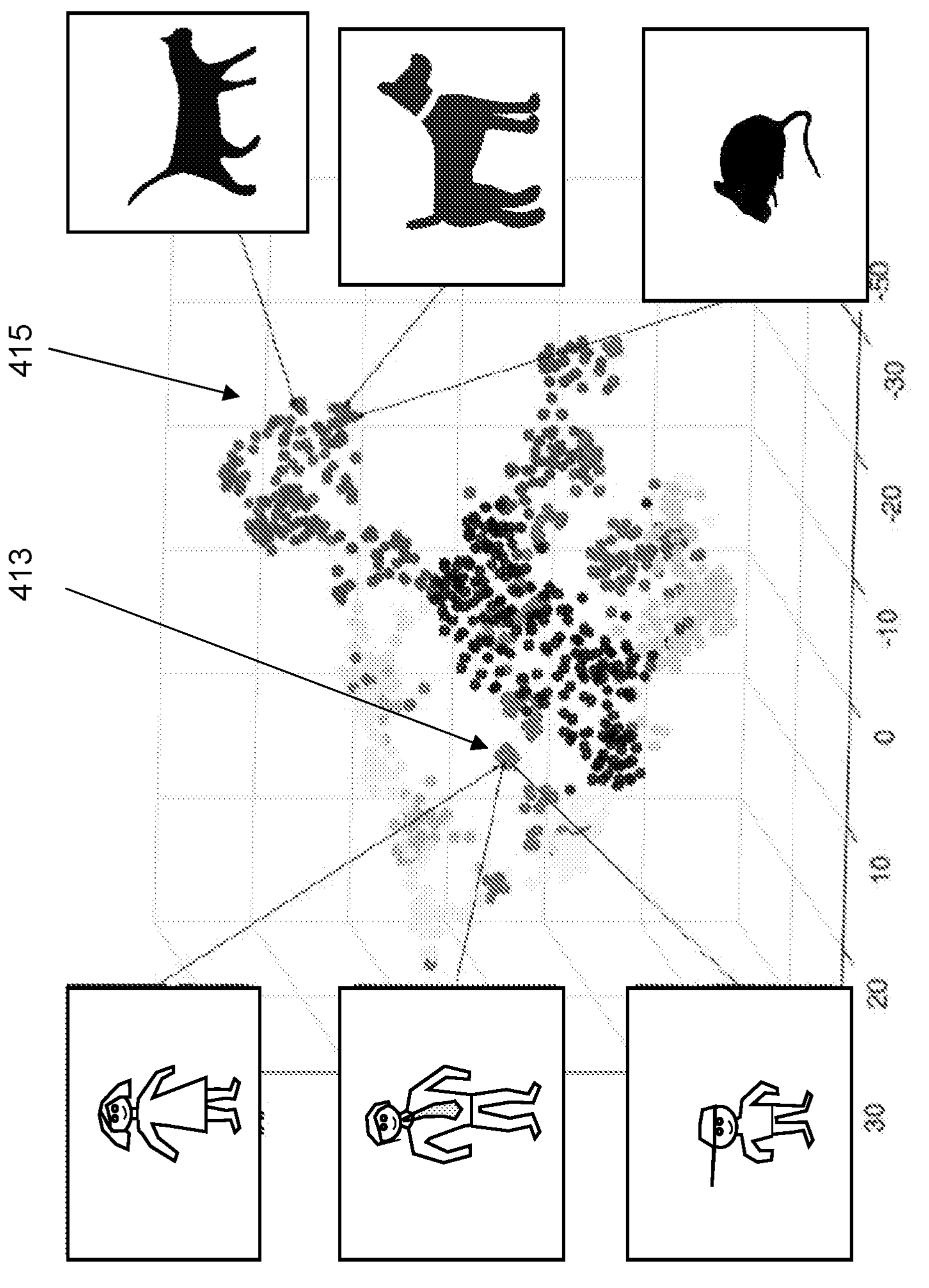

Fig. 4B

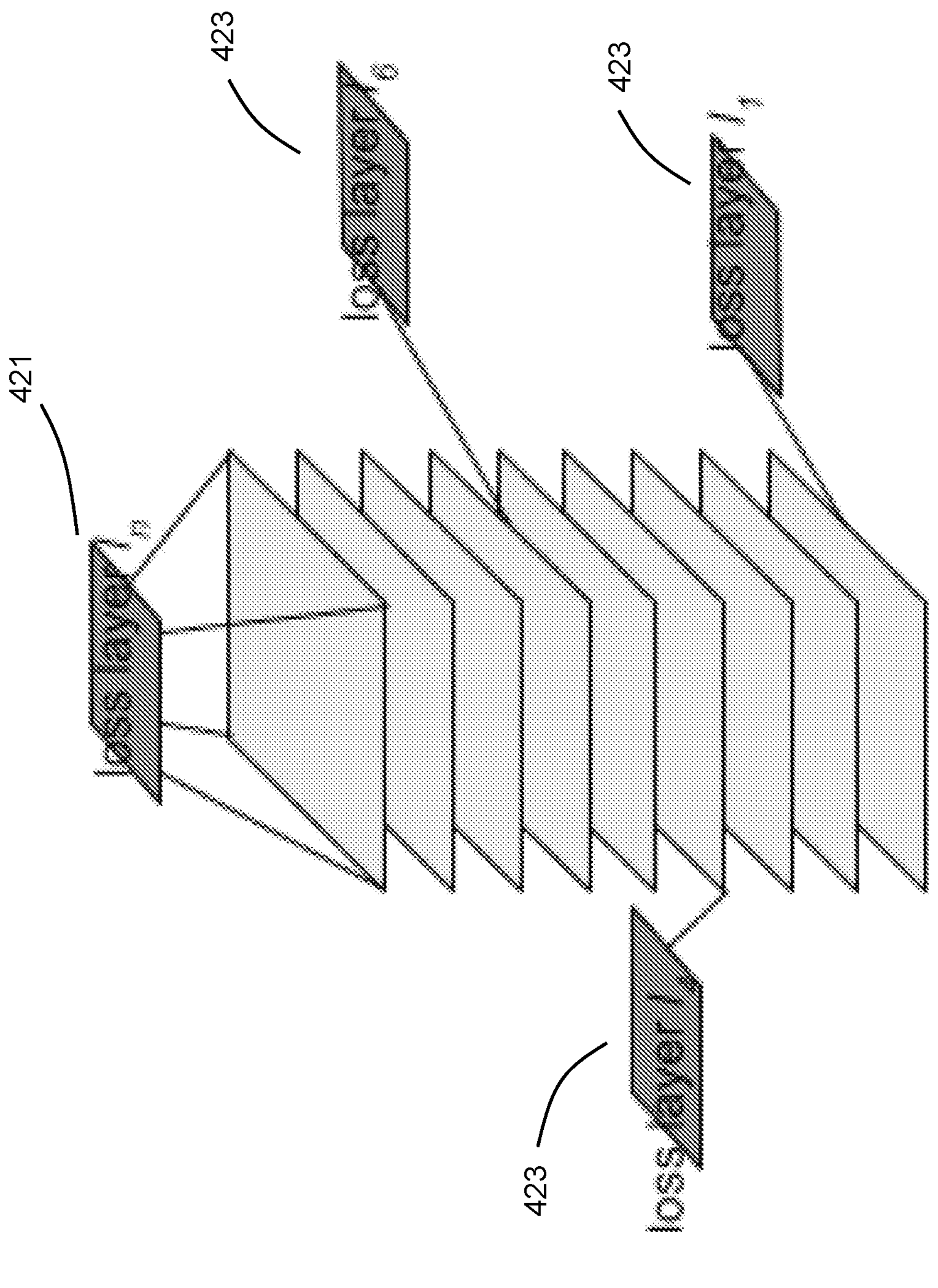
421
loss layer l_0
423
loss layer l_6
423
loss layer l_4
423
loss layer l_3

Fig. 4C

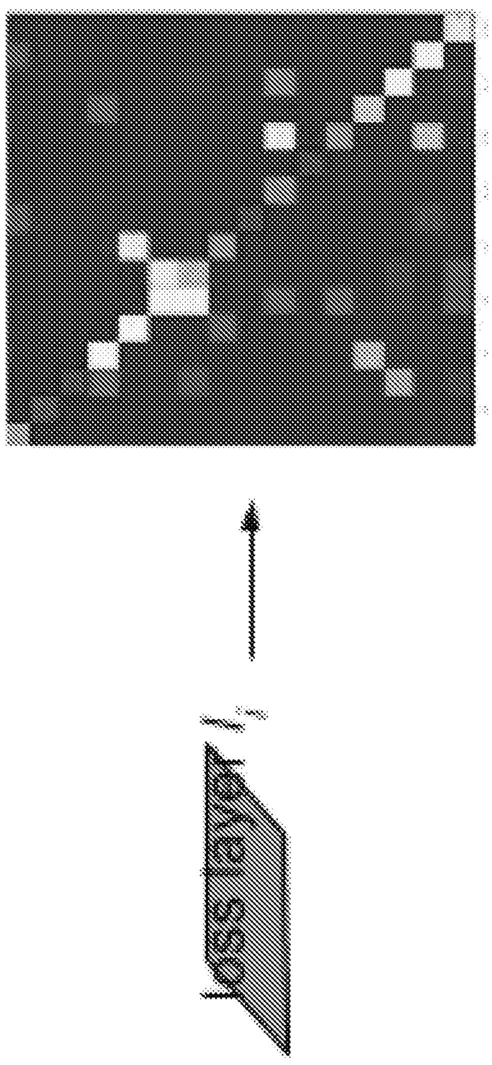

Fig. 4D

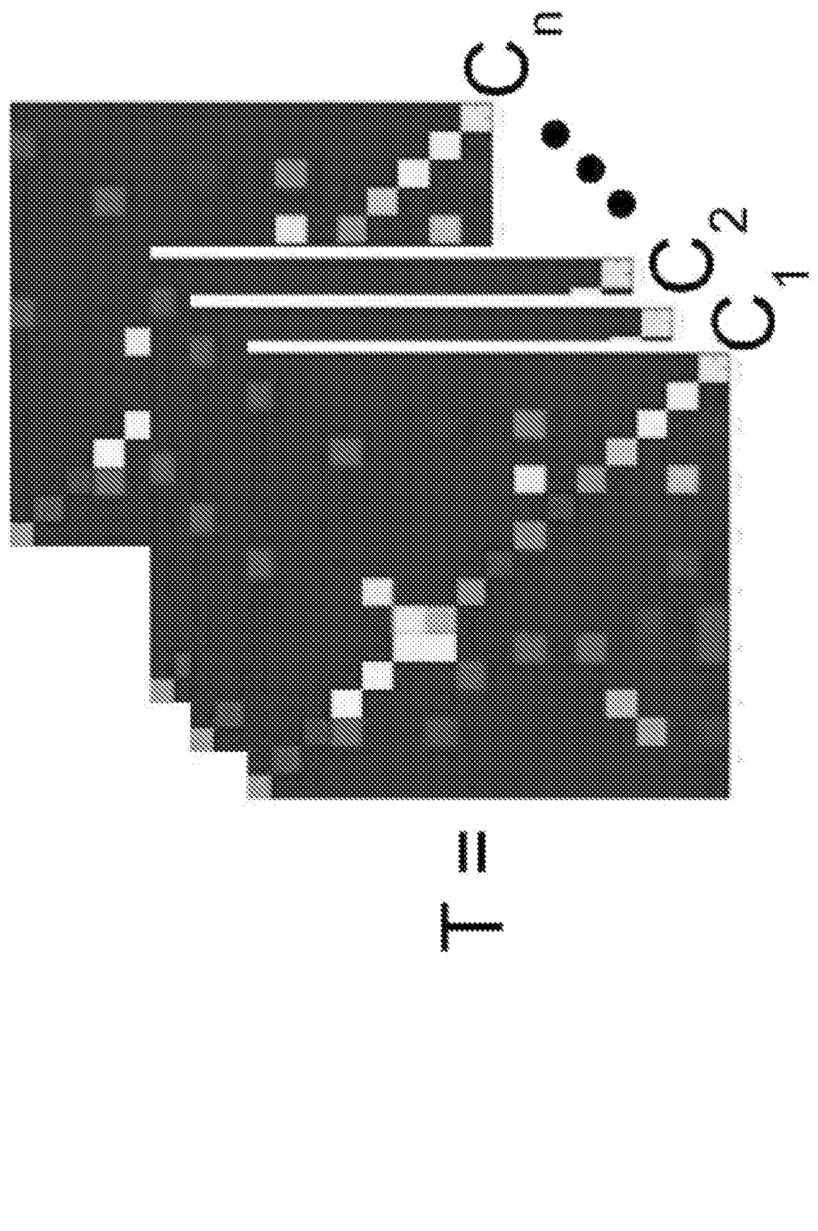

Fig. 4E

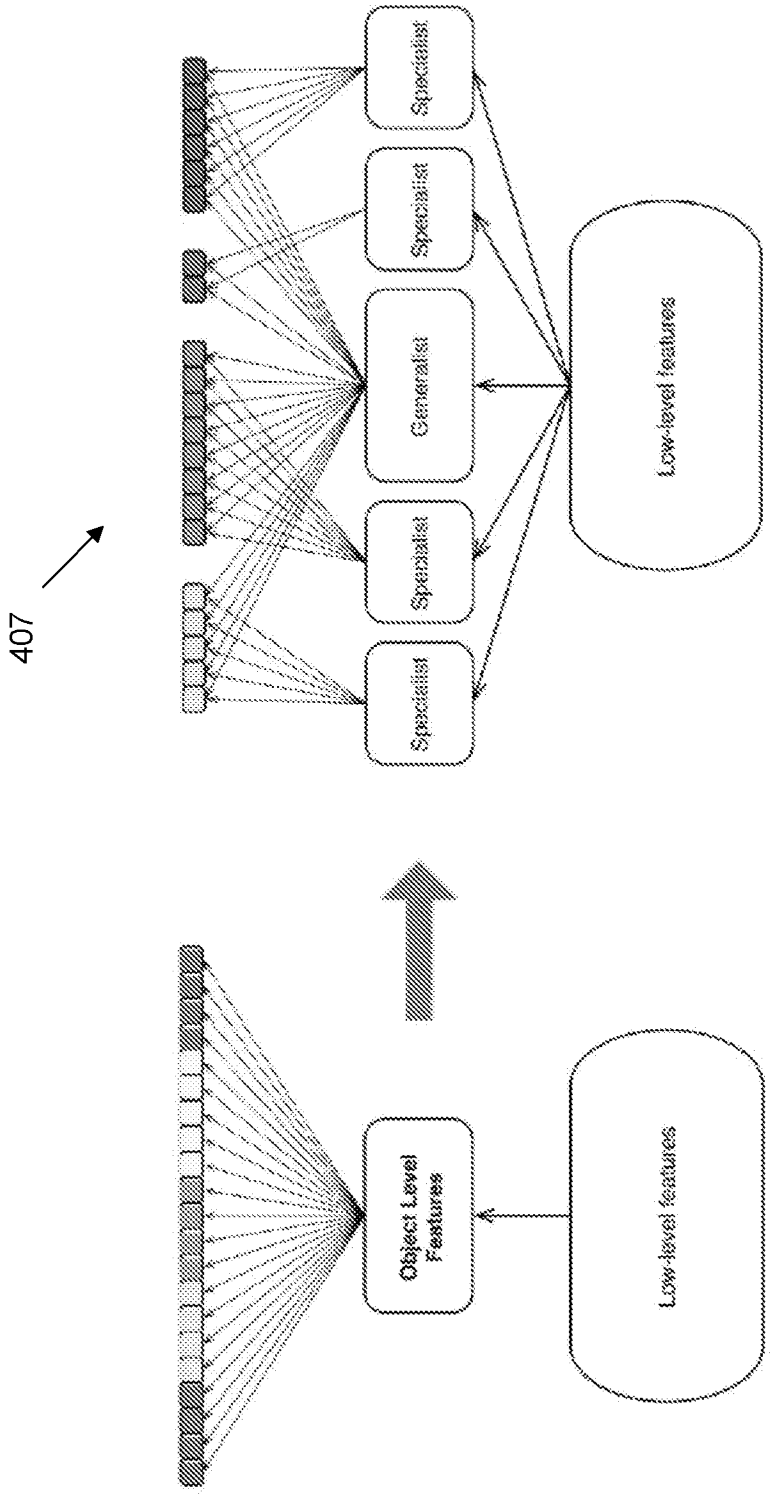
407
Specialist
Specialist
Generalist
Specialist
Specialist
Low-level features
Object Level Features
Low-level features

Fig. 4F

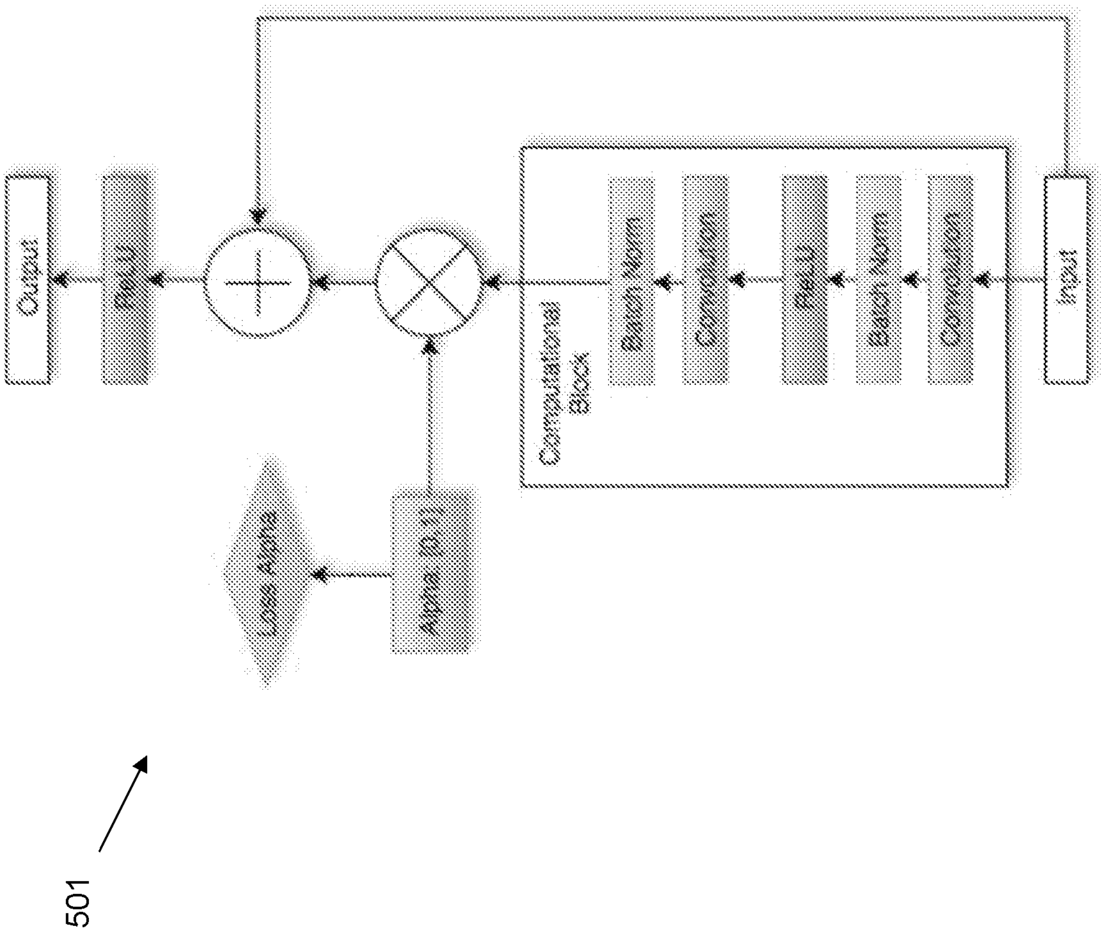
Output
ReLU
Loss Alpha
Alpha [0,1]
Computational Block
Batch Norm
Convolution
ReLU
Batch Norm
Convolution
Input
501

Fig. 5A

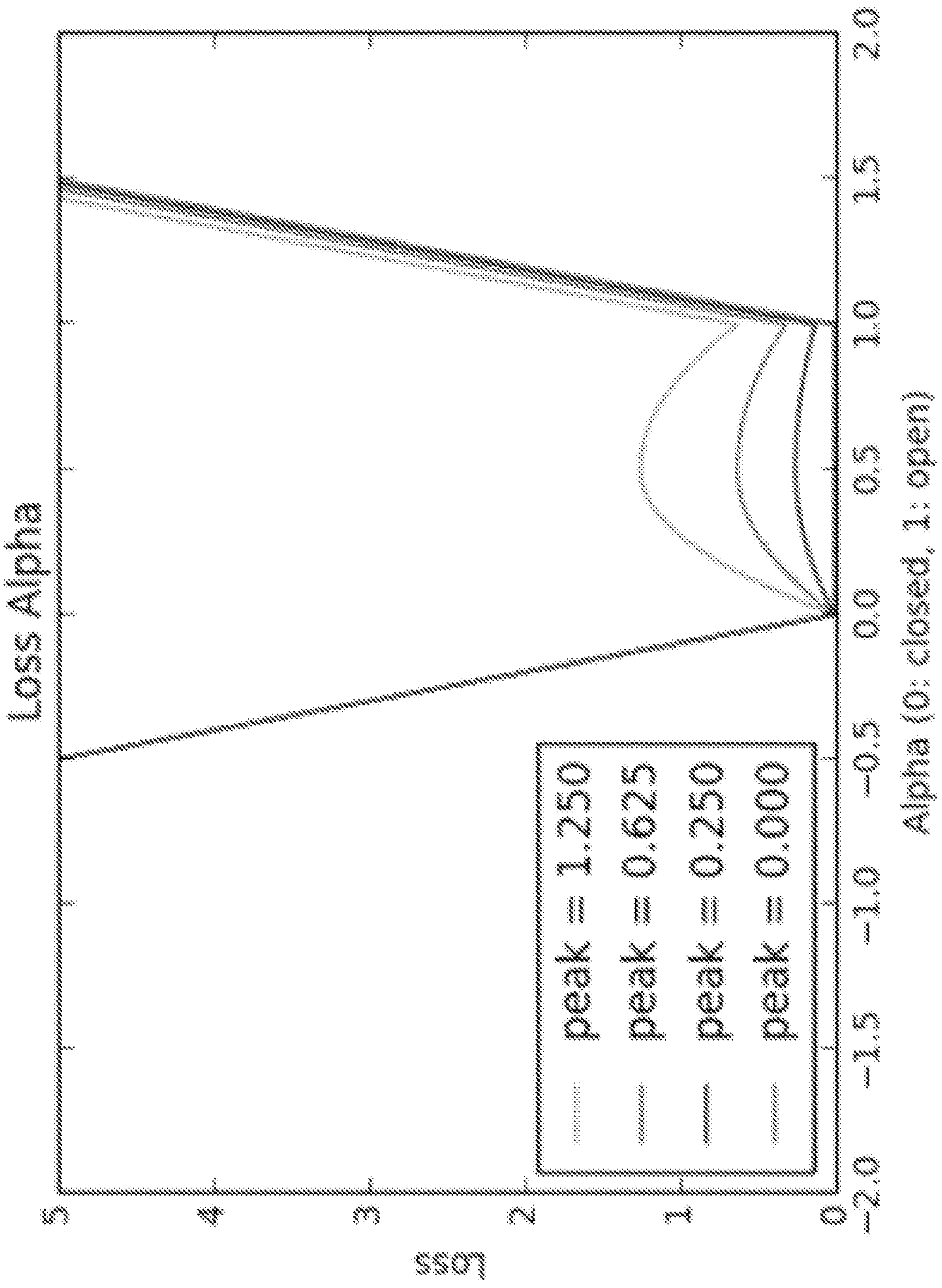
Loss Alpha
loss
Alpha (0: closed, 1: open)
peak = 1.250
peak = 0.625
peak = 0.250
peak = 0.000
0
1
2
3
4
5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Fig. 5B

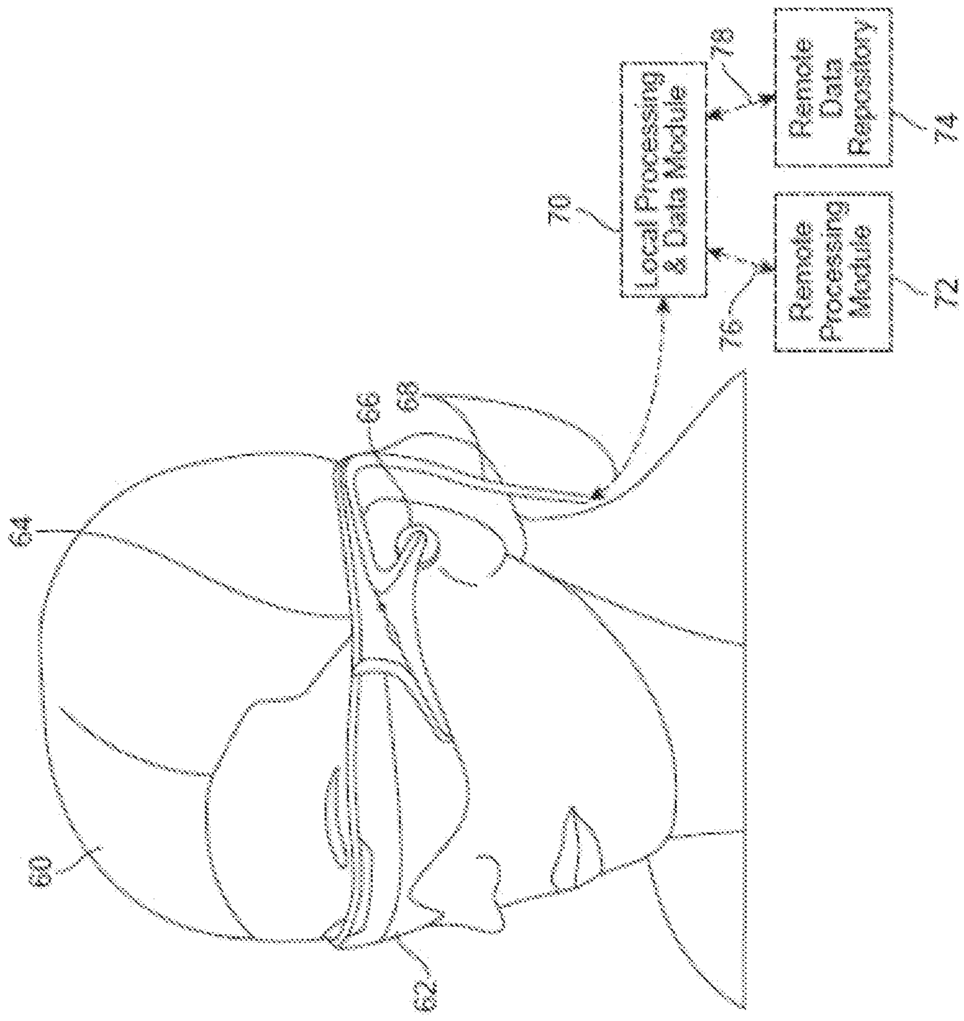
60
62
64
66
68
70
Local Processing & Data Module
76
78
Remote Processing Module
72
Remote Data Repository
74

Fig. 6A

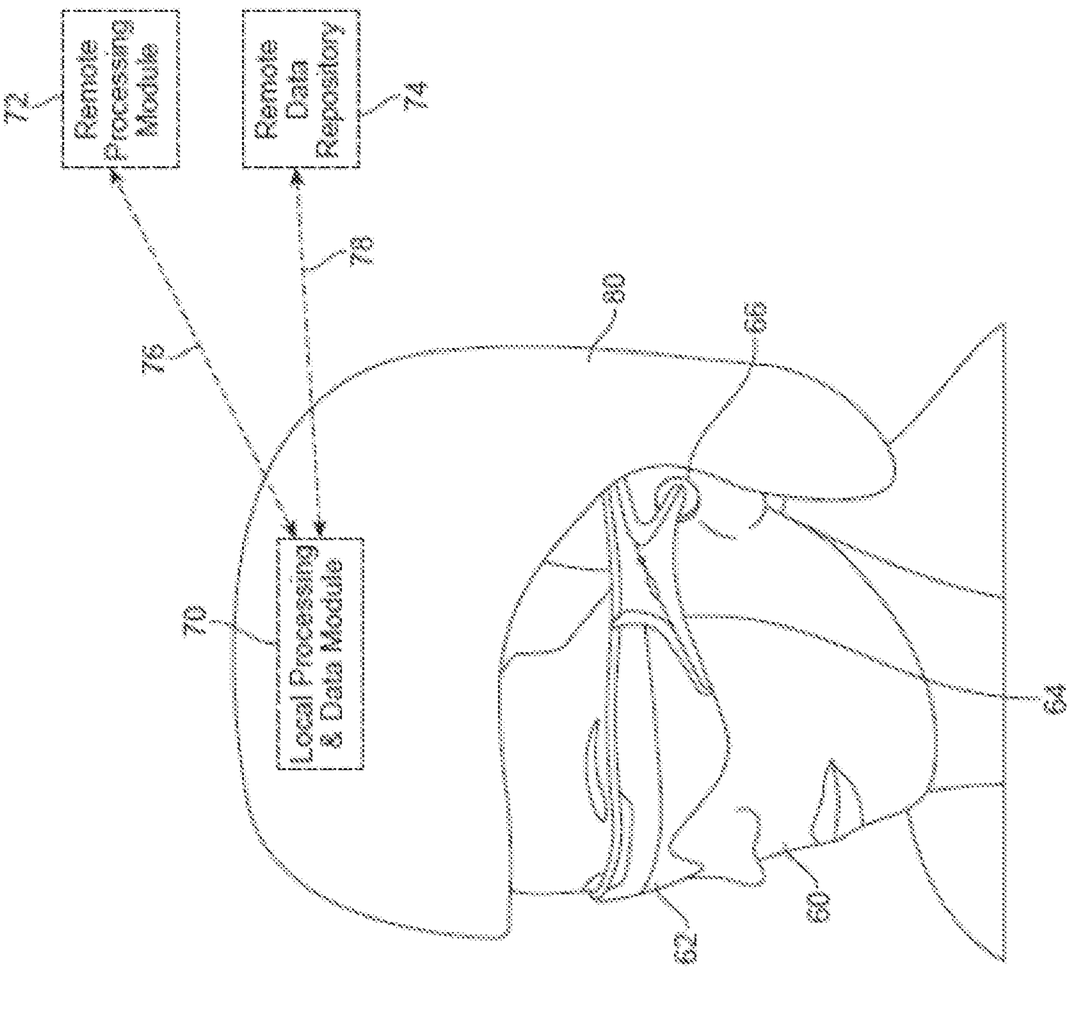
72
Remote Processing Module
74
Remote Data Repository
76
78
80
66
70
Local Processing & Data Module
64
60
62

Fig. 6B

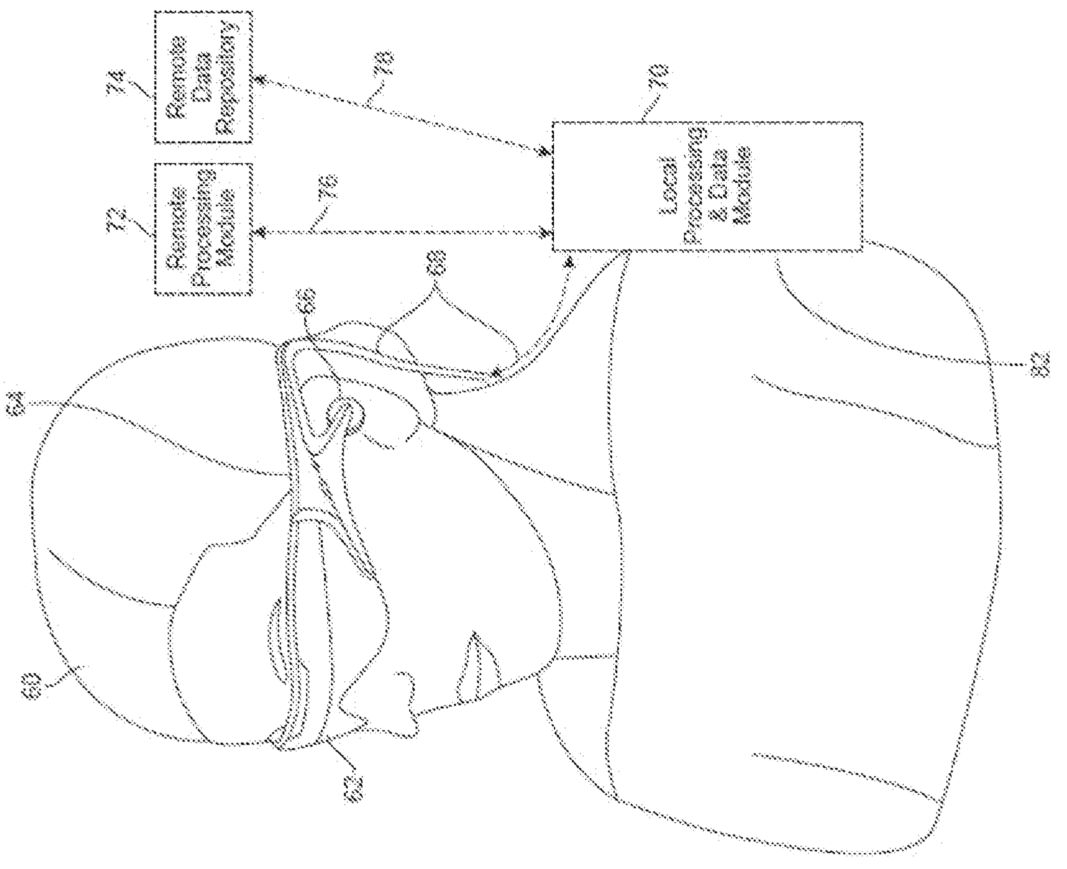
60
62
64
66
68
72
Remote Processing Module
74
Remote Data Repository
76
78
70
Local Processing & Data Module
82

Fig. 6C

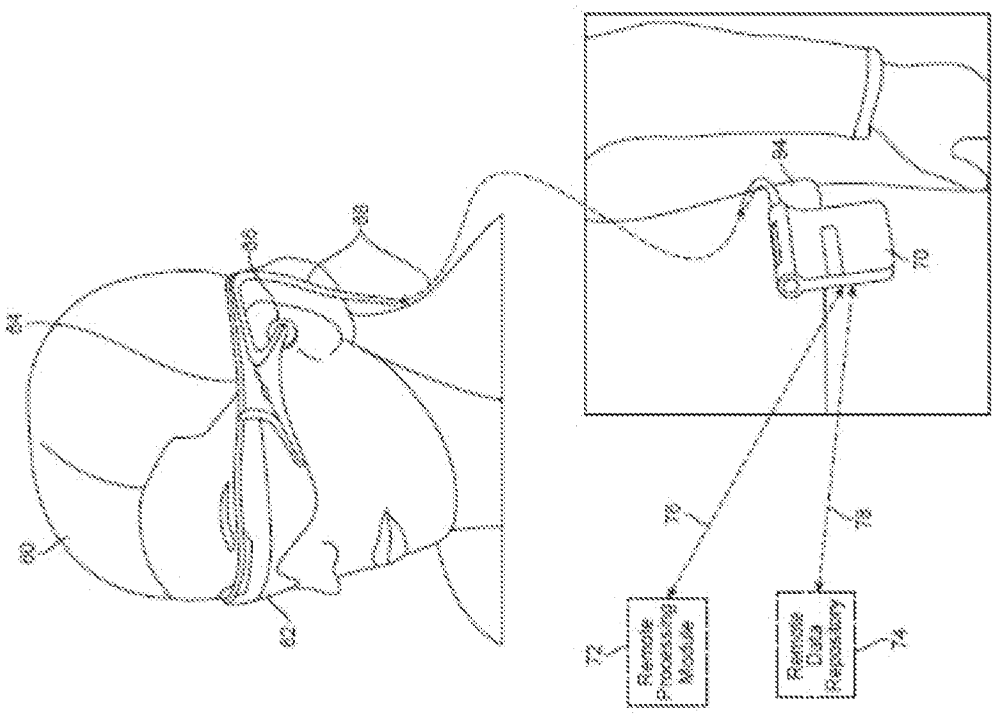

Fig. 6D

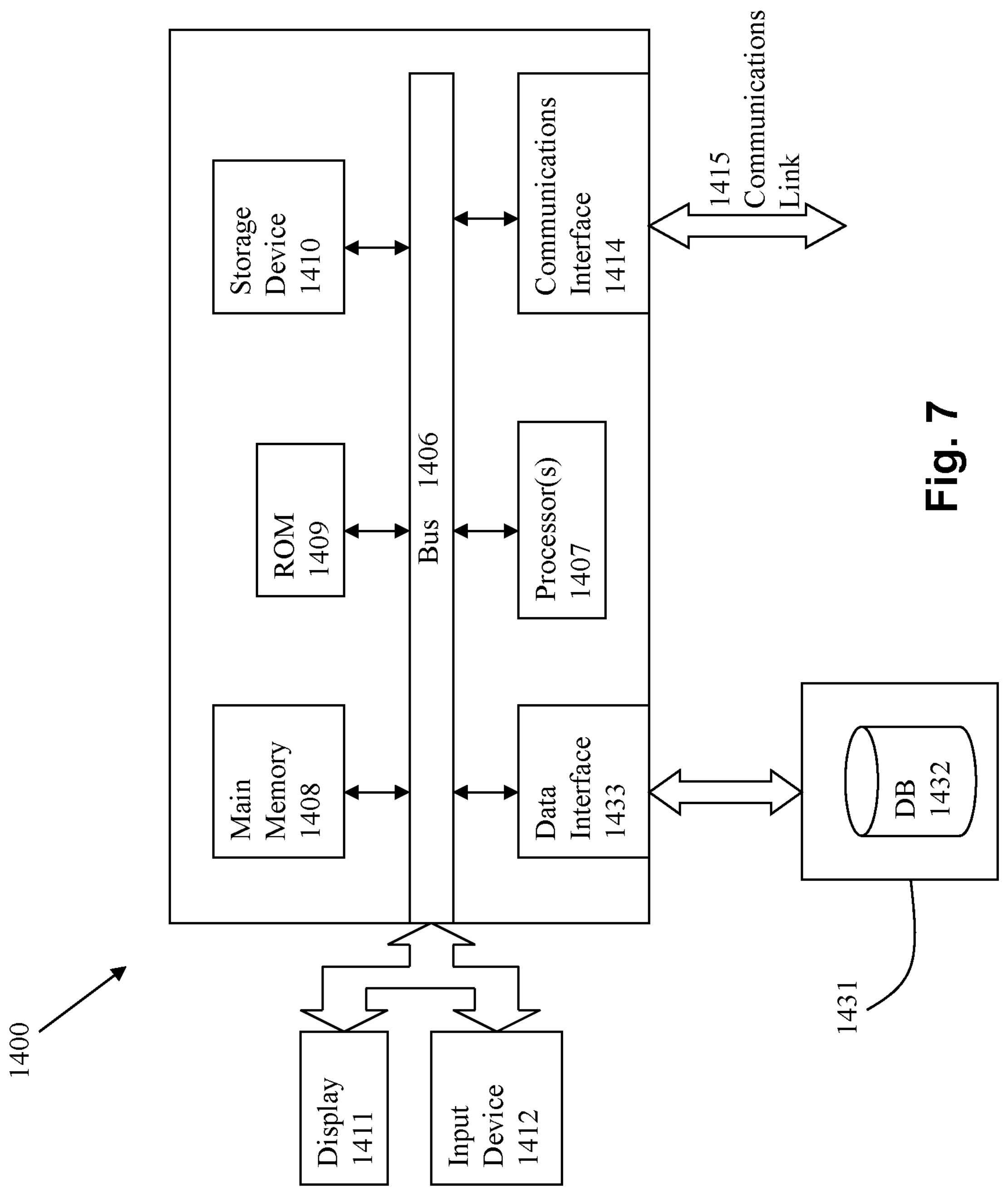
1400
Main Memory 1408
ROM 1409
Storage Device 1410
Bus 1406
Data Interface 1433
Processor(s) 1407
Communications Interface 1414
1415 Communications Link
DB 1432
1431
Display 1411
Input Device 1412

Fig. 7