US 20250259064A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2025/0259064 A1**

Versace et al. (43) **Pub. Date: Aug. 14, 2025**

(54) **SYSTEMS AND METHODS FOR DEEP NEURAL NETWORKS ON DEVICE LEARNING (ONLINE AND OFFLINE) WITH AND WITHOUT SUPERVISION**

(71) Applicant: **Neurala, Inc.**, Boston, MA (US)

(72) Inventors: **Massimiliano Versace**, Milton, MA (US); **Daniel Glasser**, Boston, MA (US); **Vesa TORMANEN**, Roslindale, MA (US); **Anatoli GORCHET**, Newton, MA (US); **Heather Ames Versace**, Milton, MA (US); **Jeremy Wurbs**, Worcester, MA (US)

(73) Assignee: **Neurala, Inc.**, Boston, MA (US)

(21) Appl. No.: **19/174,310**

(22) Filed: **Apr. 9, 2025**

Related U.S. Application Data

(63) Continuation of application No. 16/952,250, filed on Nov. 19, 2020, now Pat. No. 12,288,162, which is a continuation of application No. PCT/US2019/033345, filed on May 21, 2019.

(60) Provisional application No. 62/680,937, filed on Jun. 5, 2018, provisional application No. 62/674,346, filed on May 21, 2018.

Publication Classification

(51) **Int. Cl.**

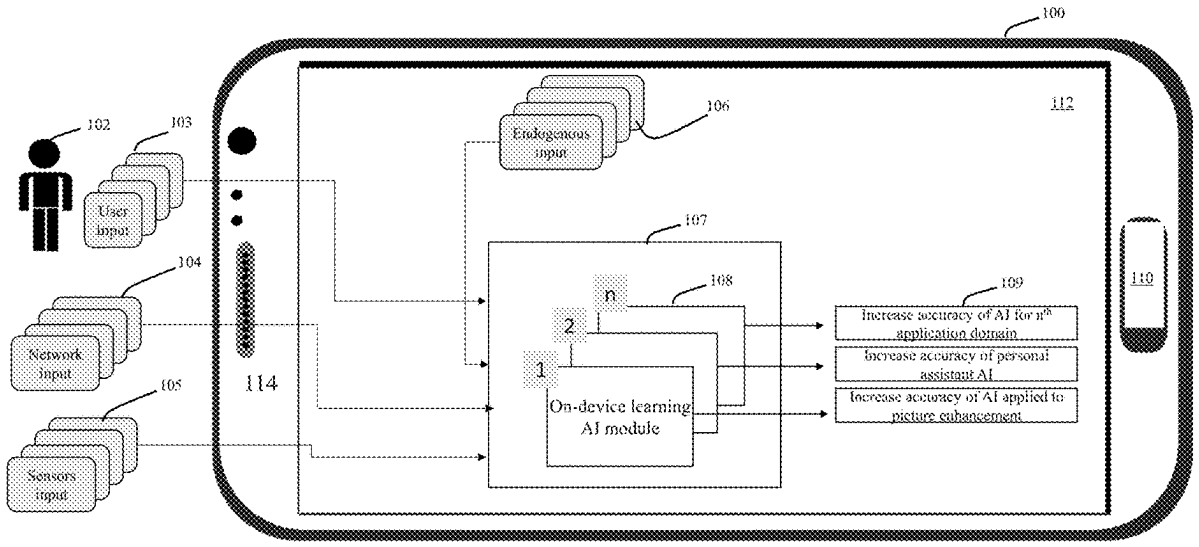
G06N 3/084	(2023.01)
G06F 18/214	(2023.01)
G06F 18/22	(2023.01)
G06F 18/2411	(2023.01)
G06N 3/08	(2023.01)
G06V 10/764	(2022.01)
G06V 10/82	(2022.01)
G06V 10/96	(2022.01)

(52) **U.S. Cl.**

CPC ***G06N 3/084*** (2013.01); ***G06F 18/214*** (2023.01); ***G06F 18/22*** (2023.01); ***G06F 18/2411*** (2023.01); ***G06N 3/08*** (2013.01); ***G06V 10/764*** (2022.01); ***G06V 10/82*** (2022.01); ***G06V 10/96*** (2022.01)

(57) **ABSTRACT**

An artificial neural network (ANN) that learns at the Edge (e.g., on a smart phone) can be faster and use less network bandwidth than an ANN trained on a server and distributed to the Edge. Learning at the compute edge can be accomplished by executing Lifelong Deep Neural Network (L-DNN) technology at the compute edge. L-DNN technology uses a representation-rich, DNN-based subsystem with a fast-learning subsystem to learn new features quickly without forgetting previously learned features. Compared to a conventional DNN, L-DNN uses much less data to build robust networks, has dramatically shorter training time, and learns on-device instead of on servers without re-training or storing data. An edge device with L-DNN can learn continuously after deployment, eliminating costs in data collection and annotation, memory, and compute power. This fast, local, on-device learning can be used in unsupervised mode to make personal assistants more intelligent and enhance frequently used apps.



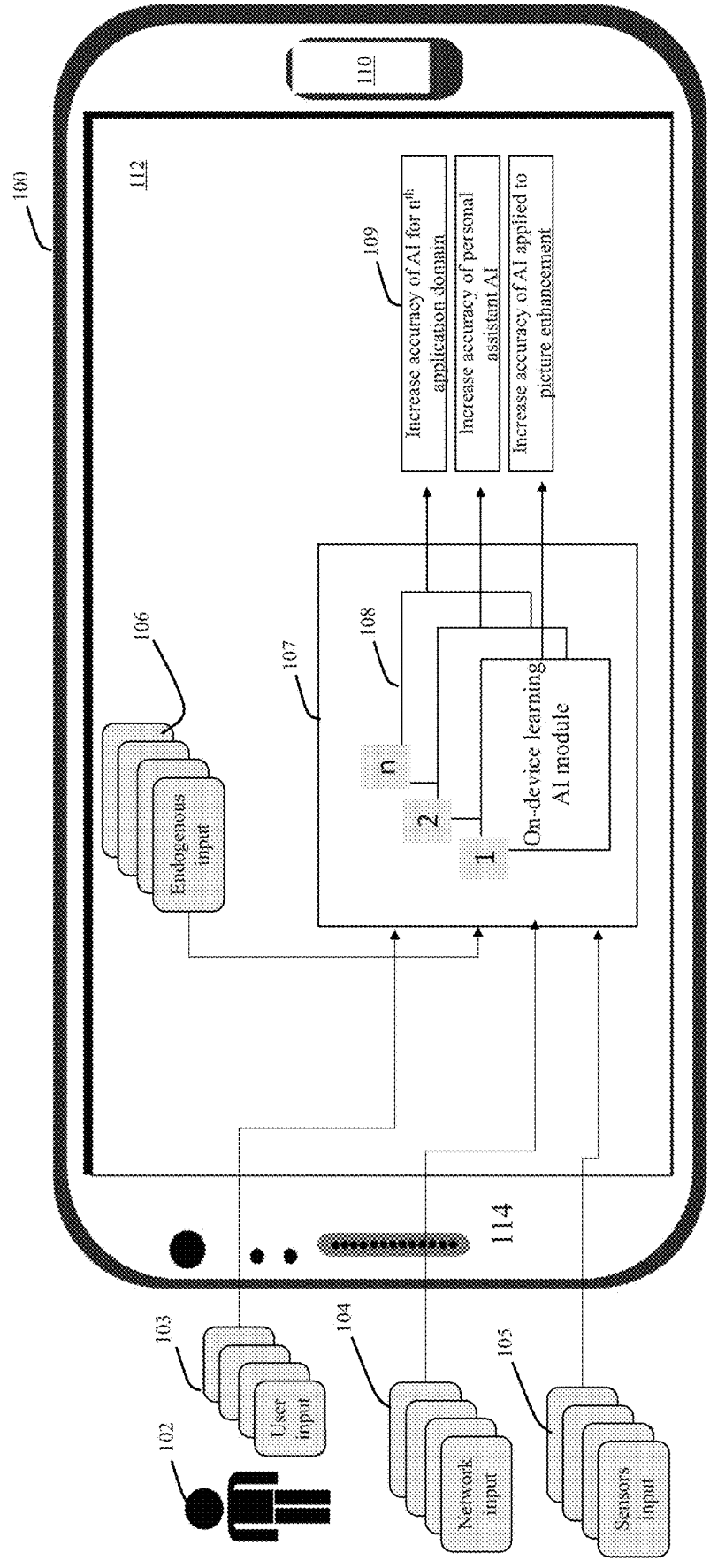

100
110
112
109
Increase accuracy of AI for n^th application domain
Increase accuracy of personal assistant AI
Increase accuracy of AI applied to picture enhancement
106
Endogenous input
107
108
n
2
1
On-device learning AI module
114
103
User input
102
104
Network input
105
Sensors input

Fig. 1

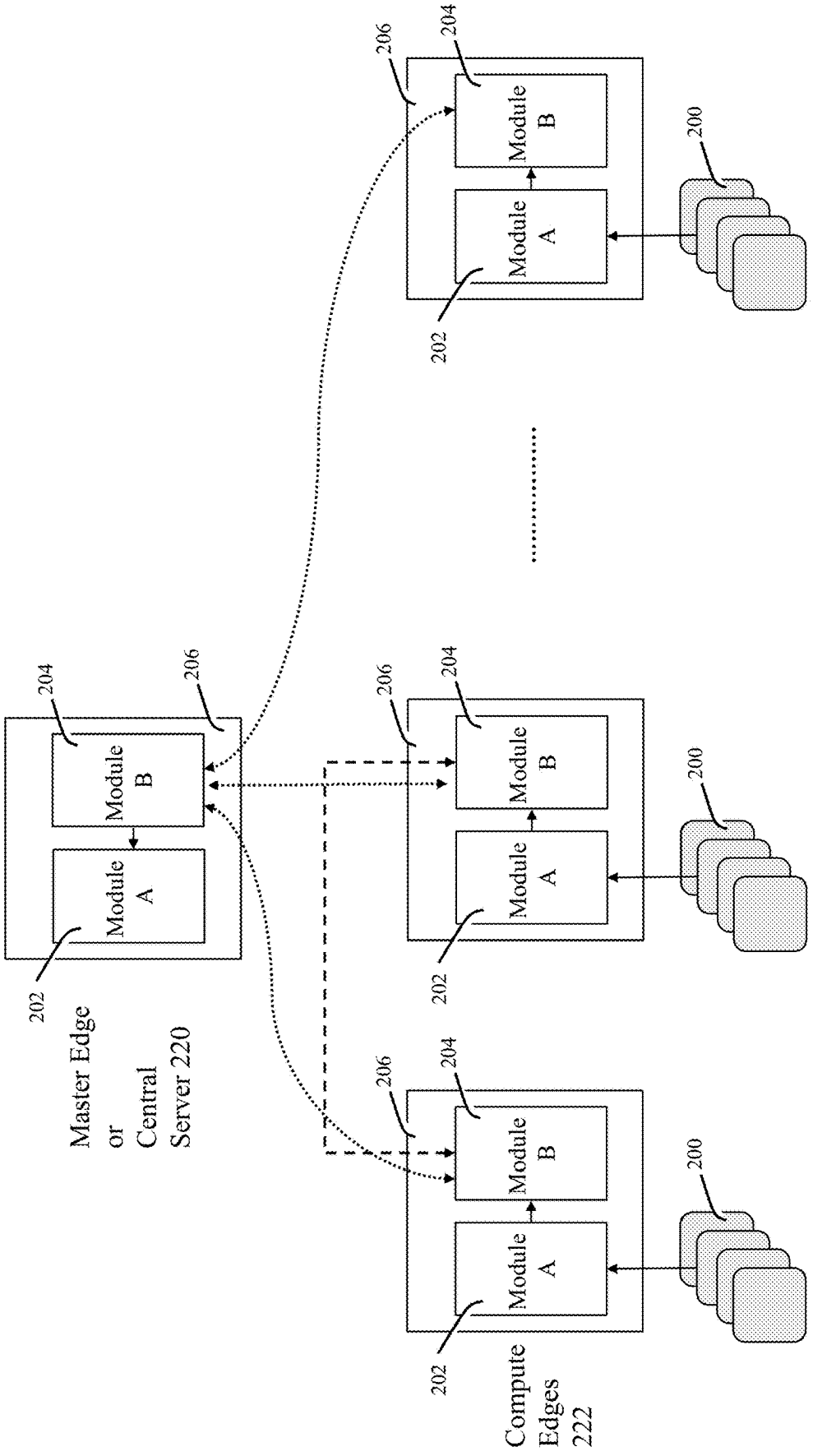

Master Edge
or
Central
Server 220
Compute
Edges
222
Module
A
Module
B
202
204
206
200

FIG. 2

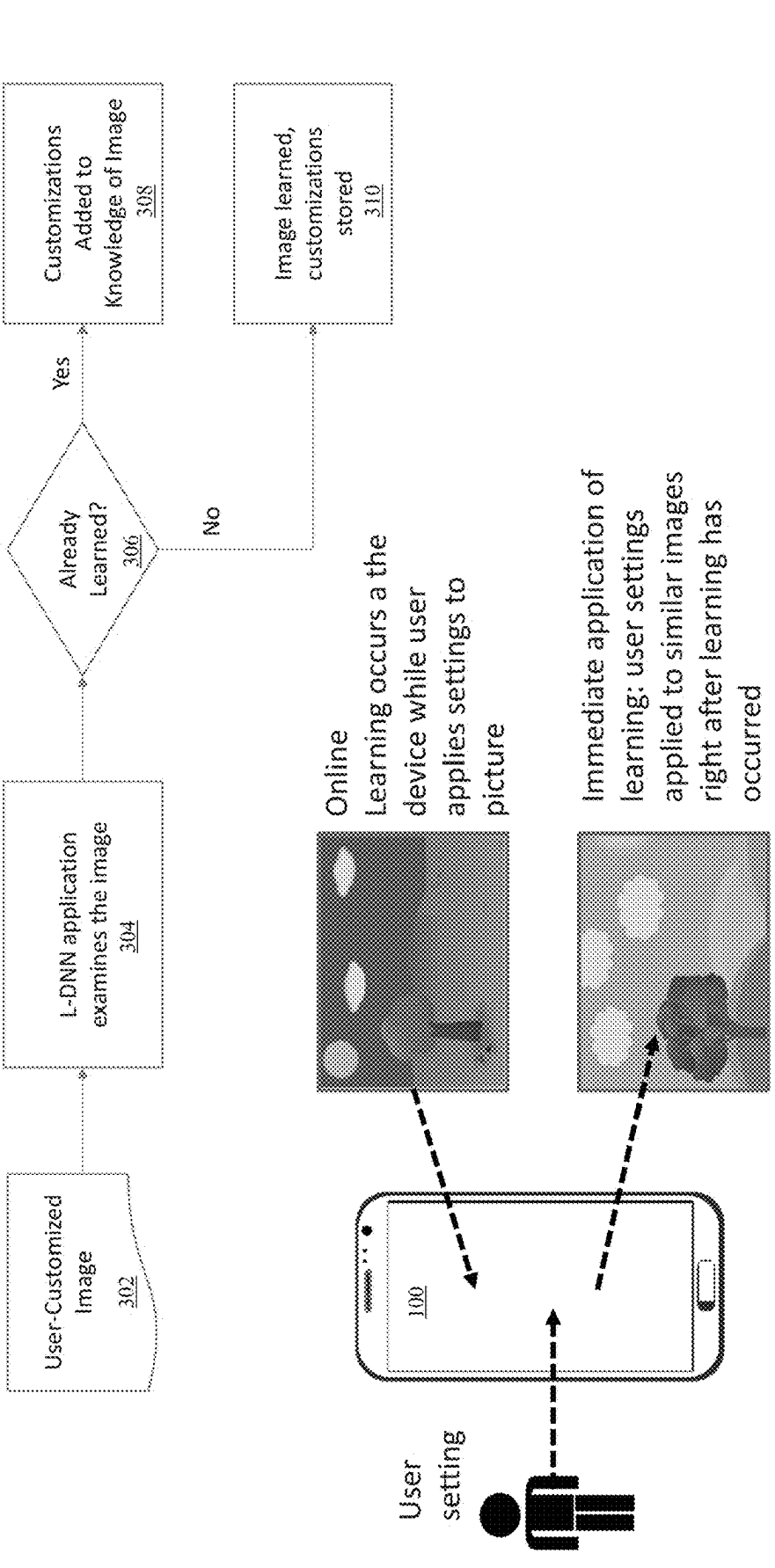

300
User-Customized Image 302
L-DNN application examines the image 304
Already Learned? 306
Yes
No
Customizations Added to Knowledge of Image 308
Image learned, customizations stored 310
Online Learning occurs a the device while user applies settings to picture
Immediate application of learning: user settings applied to similar images right after learning has occurred
100
User setting

Fig. 3

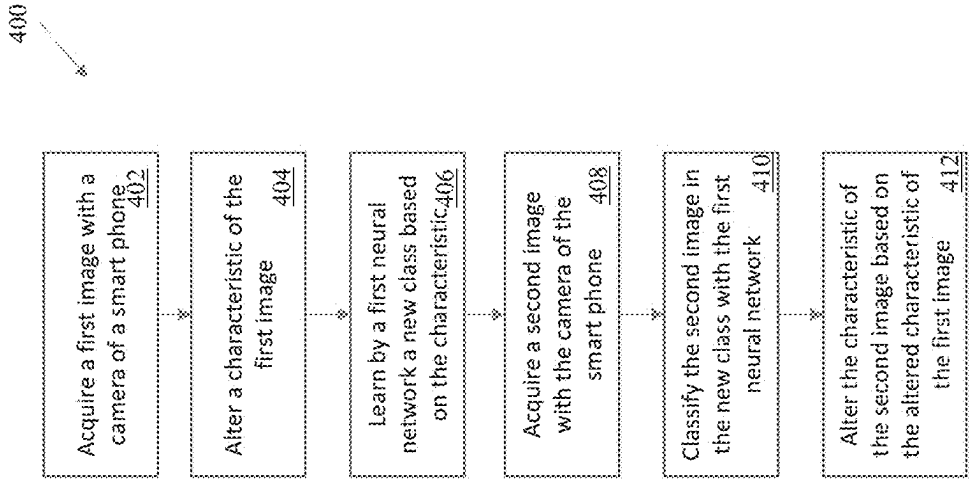

400
Acquire a first image with a camera of a smart phone 402
Alter a characteristic of the first image 404
Learn by a first neural network a new class based on the characteristic 406
Acquire a second image with the camera of the smart phone 408
Classify the second image in the new class with the first neural network 410
Alter the characteristic of the second image based on the altered characteristic of the first image 412

Fig. 4

SYSTEMS AND METHODS FOR DEEP NEURAL NETWORKS ON DEVICE LEARNING (ONLINE AND OFFLINE) WITH AND WITHOUT SUPERVISION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation of U.S. application Ser. No. 16/952,250, entitled "Systems and Methods for Deep Neural Networks on Device Learning (Online and Offline) with and without Supervision," filed on Nov. 19, 2020, which is a bypass continuation of International Application No. PCT/US2019/033345, entitled "Systems and Methods for Deep Neural Networks on Device Learning (Online and Offline) with and without Supervision," filed on May 21, 2019, which in turn claims the priority benefit, under 35 U.S.C. § 119(e), of U.S. Application No. 62/680,937, entitled "Systems and Methods to Enable Continual, Memory-Bounded Learning in Artificial Intelligence and Deep Learning Continuously Operating Applications with and Without Supervision," filed on Jun. 5, 2018, and of U.S. Application No. 62/674,346, entitled "Systems and Methods for Deep Neural Networks on Device Learning (Online and Offline) with Applications to Photography, Image Processing, and User Behavior Understanding," filed on May 21, 2018. Each of these applications is incorporated herein by reference in its entirety.

BACKGROUND

[0002] Traditional artificial neural networks (ANNs) and deep neural networks (DNNs), which often include many layers of neurons interposed between the input and output layers, often require thousands or millions of iteration cycles to train. These cycles are frequently performed in a high-performance computing sever. In fact, some traditional DNNs may take days or even weeks to be trained, depending on the size of the input dataset.

[0003] One technique for training a DNN involves the backpropagation algorithm. The backpropagation algorithm computes changes of all the weights in the DNN in proportion to the error gradient from a labeled dataset via application of the chain rule in order to backpropagate error gradients. Backpropagation makes small changes to the weights for each datum and runs over all data in the set for many epochs.

[0004] The larger the step size taken per iteration cycle, the less likely that the gradient of the loss function can lead to actual performance gains. Thus, DNNs make small changes to their weights every training sample. Furthermore, since the gradient of the loss function computed for any single training sample can affect all the weights in the network (due to the typically distributed representations), standard DNNs are vulnerable to forgetting previous knowledge when they learn new objects.

[0005] Repetitive presentations of the same inputs over multiple epochs mitigates this issue of forgetting previous knowledge, with the drawback of making it extremely difficult to add new knowledge to the system quickly. This is one reason why learning is impractical or altogether impossible on a computationally limited edge device (e.g., a cell phone, tablet, or small form-factor processor). Even if the problem of forgetting was solved, learning on edge devices would still be impractical due to the high computational load of the training, small training steps, and repetitive presentation of all inputs.

[0006] These limitations are true for a single compute Edge across its deployment lifespan, where the Edge may update its knowledge, and for distributed, multi-Edge systems (e.g., smart phones connected in a network, networked smart cameras, fleets of drones or self-driving vehicles, and the like), where quick sharing of newly acquired knowledge is desirable for an intelligent agent across the agent's deployment life cycle.

[0007] In order to learn knowledge, a real-time operating machine that uses a traditional DNN may have to accumulate a large amount of data to retrain the DNN. The accumulated data is transferred from the "Edge" of the real-time operating machine (i.e., the device itself, for example, a self-driving car, drone, robot, etc.) to a central server (e.g., a cloud-based server) to get the labels from the operator and then retrain the DNN executed on the Edge. The more accumulated data there is, the more expensive the transfer process in terms of time and network bandwidth. In addition, interleaved training on the central server has to combine the new data with the original data that is stored for the whole life cycle of the system. This creates severe transmission bandwidth and data storage limitations.

[0008] Hence, due to the data-and-compute intensive nature of this training process, it is extremely cumbersome to add new knowledge on-the-fly exploiting relatively low-powered devices, such as personal computers, smart phones, tablets, Internet of Things (IoT) devices, and the like.

SUMMARY

[0009] The present technology extends Artificial Intelligence (AI), Artificial Neural Networks (ANNs), Deep Neural Network (DNNs), and other machine vision processes so that they can be trained at the compute Edge (e.g., on a smartphone, drone, or robot) through application of Lifelong Deep Neural Network (L-DNN) technology. The term "compute Edge," (also referred to herein as "Edge'), in particular as it relates to remote (cloud) compute, refers to computing systems that perform data processing at a device located at the edge of the computer network, near the source where data is generated. This approach involves leveraging resources that may not be continuously connected to a network, such as smartphones, tablets, personal computing devices, IoT devices, cameras, and other sensors.

[0010] Whereas AI (e.g., ANNs, DNNs) can be performed on remote or cloud resources, there are several benefits of operating the AI at the device level (Edge), some of which include:

[0011] Reduction of AI latency: because it is located at the Edge, the AI can achieve real-time or near real-time data analysis, since data is processed on a processor located directly at the local device level, not on a cloud or data center resource. Put differently, in conventional approaches, data is transmitted from the edge to a remote or cloud resource for processing. Once the data is processed (i.e., AI performed), instructions are transmitted from the remote or cloud resource back to the edge. By processing data at the device level, this transmission back-and-forth to and from the remote or cloud resource can be eliminated, thereby reducing latency;

[0012] Reduction of data transfer and network traffic: since data is processed at the device level, raw data

does not need to be transmitted from the device via a computer network to a data center or cloud, thereby reducing network traffic;

[0013] Application of L-DNN technology allows one-shot learning, removing the need to store data for future use either on the edge or on the server.

[0014] Consequently, data transfer, data storage, and data manipulation costs related to remote AI applications are much lower for Edge computing than for cloud-based AI applications;

[0015] Data privacy: processing data (both inference and learning) at the device enhances user privacy and data security because user data is generated, processed, and discarded at the device level, possibly without being transmitted off the device and without being stored persistently on the device in raw form.

[0016] On-device learning techniques like L-DNN enable new sets of functionalities in devices that have limited compute power and/or connectivity to a central server, where these limitations on compute power and connectivity can be technical or related to data privacy. By learning directly on the device where the data is generated and/or used by the end user, the inventive technology unlocks capabilities previously only achieved with running AI (ANN, DNN) processes on a compute server.

[0017] Using unsupervised or semi-supervised L-DNN further simplifies the use of AI on the Edge by reducing the number and/or duration of AI-related user interactions with the device. This frees the user to pursue more creative tasks while retaining the ability of AI to learn both from data and from user actions that are performed normally during tasks that AI is designed to assist.

[0018] Edge devices capable of implementing the inventive methods include smart phones, tablets, and other devices (e.g., drones, robots, IoT devices, cameras, etc.). They can implement online and offline learning methods, including learning while the device is being used (online) and learning when the device is idle/offline (e.g., during battery recharge, overnight, or in other situations where the user is not actively interacting with the device). Learning can be executed locally in the device, namely, without broadcasting data to an external computing medium for learning, with all processing related to learning executed by an on-board processor. As an example, learning may occur on a smart phone or tablet or PC compute power, which may include a Central Processing Unit (CPU), a Graphic Processing Unit (GPU), a Neural Processing Unit (NPU), another co-processor available on-device (e.g., a vector co-processor or specialized ASIC or FPGA), or a combination of one or more of the above-mentioned processors.

[0019] On-device learning can be used to enhance engagement with the user device by intelligently augmenting, via learning directly on the user device, the knowledge of an ANN or DNN already implemented on the user device. Normally, the process of learning or enhancing the knowledge of an ANN or DNN is cumbersome and involves costly (in terms of time and compute power) training that normally occurs on a compute server or powerful workstation. Conversely, on-device learning, taking advantage of on-board processing power, can augment user experience in AI-powered applications. One example of this on-device learning is termed Lifelong Deep Neural Network (L-DNN) and disclosed in U.S. application Ser. No. 15/975,280, which was filed on May 9, 2018, and is incorporated herein by reference in its entirety. Other equivalent on-learning techniques may be used as well.

[0020] The method discussed herein can occur in two modes on the target device: (1) a fast, online learning mode; and (2) a slow, offline learning mode. In the fast learning mode/online mode, a real-time operating machine implementing L-DNN learns new knowledge and new experiences quickly so that it can respond to the new knowledge almost immediately. In this mode, the learning rate in the fast learning subsystem is high to favor new knowledge and the corresponding new experiences, while the learning rate of the fast subsystem in the slow learning mode/offline mode is set to a low value or zero to preserve old knowledge and the corresponding old experiences. In the slow learning mode/offline mode, the fast subsystem serves as a teacher to the slow subsystem. Therefore, the learning rate of the fast subsystem in slow learning mode/offline mode is set to a low value (e.g., zero) in order to reduce modification (e.g., low learning that is closer/similar to biology) of the fast subsystem.

[0021] Online: learning and adaptation of a fast learning module/fast learning subsystem in the L-DNN occurs while the user is actively using the device. In this case, learning occurs instantaneously either by using user-delivered supervision (e.g., an action performed by the user on the device) or using other events captured by the device. These other events may include endogenous events in the device (e.g., events triggered in the device OS, or by applications running on the device) or events captured in the network traffic of the device (e.g., incoming signals through wireless connectivity).

[0022] Offline: while learning still occurs entirely on-device, the update of the underlying AI algorithm occurs offline (e.g., when the device is idle), namely while the user is not actively engaged in using the device, or the AI is not actively engaged in performing a task. In the offline learning case, the L-DNN uses self-supervised learning to improve generalization and reduce the memory footprint. In this mode, the learning rate in the fast learning subsystem of L-DNN is low or zero and it serves as a supervisor to the slow learning system, while the learning rate in the slow learning subsystem is elevated to allow consolidation of learned features through an appropriate loss function (for example a triplet loss) driving a traditional back propagation of error.

[0023] Applications of this technology include, but are not limited to, enhancing AI, ANNs, and DNNs implemented in devices such as smart phones, IoT devices, drones, cameras, tablets, and use cases ranging from, but not limited to, photography, personal assistants, and other applications where AI is performing a task at the compute Edge.

[0024] Learning at the compute Edge enables AI embedded on a device at the compute Edge to adapt its output to the data immediately available at the device itself. Examples uses of this technology include: AI-powered photography; AI-powered personal assistants; and AI-powered data processing that occurs at the compute Edge in smart phones, IoT devices, cameras, and the like.

[0025] In summary, different variations of the inventive technology include:

[0026] On-device learning where training occurs on a continuous basis directly at the compute Edge, where

the performance of the AI changes, and user experience of the device is continuously improved, by continuous learning;

[0027] This on-device learning in some implementations can be performed in unsupervised fashion to reduce the load on the user of the device and free the user for more creative tasks other than supervision of AI.

[0028] Offline learning on the device, where the system uses self-supervision to further improve the quality of AI and reduce its memory footprint.

[0029] A method of image processing with a smart phone executing a neural network comprising a fast learning subsystem and a slow learning subsystem is disclosed herein. The method includes acquiring a first image with a camera of the smart phone. The method also includes altering at least one characteristic of the first image in response to input from a user. The method also includes learning a new class by the fast learning subsystem based on the at least one characteristic of the first image altered in response to the input from the user. The method also includes acquiring a second image with the camera of the smart phone, classifying the second image in the new class with the fast learning subsystem, and in response to classifying the second image in the new class, automatically altering at least one characteristic of the second image on the smart phone based on the at least one characteristic of the first image altered in response to the input from the user.

[0030] In some implementations, the neural network can be a Lifelong Deep Neural Network. In some implementations, the method also includes providing the first image and the second image to the slow learning subsystem, generating ground truth data by the slow learning subsystem based on the first image and the second image, and providing the ground truth data to the fast learning subsystem.

[0031] In some implementations, the slow learning subsystem can be a Deep Neural Network. In some implementations, the method includes providing the first image and the second image to the slow learning subsystem when the smart phone is idle. In some implementations, the method also includes generating at least one feature vector using the slow learning subsystem for the first image and the second image, determining an error via the fast learning subsystem for the at least one feature vector based on a triplet loss function, and adjusting at least one weight of the slow learning subsystem based on the error.

[0032] In some implementations, determining the error includes computing a first distance between the at least one feature vector and a correct class label, computing a second distance between the at least one feature vector and an incorrect class label, and determining the error based on the first distance and/or the second distance.

[0033] In some implementations, the method also includes training the fast learning subsystem based on the ground truth data. In some implementations, training the fast learning subsystem occurs when the smart phone is charging and/or is idle.

[0034] In some implementations, the method also includes learning by the slow learning subsystem an identity of the first image and the second image. In some implementations, the identity of the first image and the second image includes at least one label for the first image and the second image. In some implementations, the method also includes teaching by the slow learning subsystem the identity of the first image and the second image to the fast learning subsystem.

[0035] A method of image processing with a smart phone is disclosed herein. The method includes acquiring a first image with a camera of the smart phone, and while the smart phone is in an idle state: creating a first label for the first image with a first subsystem included in a neural network executed by a processor on a smart phone, and teaching, by the first subsystem, the first label to a second subsystem included in the neural network executed by the processor of the smart phone.

[0036] In some implementations, the method also includes acquiring a second image with the camera of the smart phone, and applying the first label to the second image with the second subsystem.

[0037] In some implementations, the neural network is a Lifelong Deep Neural Network. In some implementations, the second subsystem can enable real-time learning. In some implementations, the teaching can include teaching the first label to the second subsystem via backpropagation.

[0038] In some implementations, the method includes acquiring a second image with a camera of the smart phone, determining by the second subsystem an association between a feature vector representing an object in the second image and a second label, and applying the second label to the object in the second image. In some implementations, the second label is received from a user. In some implementations, the second label is generated by the neural network.

[0039] A smart phone is disclosed herein. The smart phone includes an image sensor to acquire a first image and a second image. The smart phone also includes at least one processor executing a neural network. The neural network can include a first subsystem and a second subsystem. The at least one processor to: alter at least one characteristic of the first image in response to input from a user, learn via the first subsystem a new class based on the at least one characteristic of the first image, classify via the first subsystem the second image in the new class, and alter at least one characteristic of the second image based on the at least one characteristic of the first image.

[0040] In some implementations, the at least one processor can be further configured to generate via the second subsystem ground truth data based on the first image and the second image, and provide the ground truth data to the first subsystem.

[0041] In some implementations, the neural network can be a Lifelong Deep Neural Network. In some implementations, the second subsystem is a Deep Neural Network.

[0042] In some implementations, the at least one processor can be configured to generate the ground truth data when the smart phone is idle. In some implementations, the at least one processor can be further configured to: generate via the second subsystem at least one feature vector for the first image and the second image, determine via the first subsystem an error for the at least one feature vector based on a triplet loss function, and adjust at least one weight of the second subsystem based on the error.

[0043] In some implementations, the at least one processor can be further configured to: compute a first distance between the at least one feature vector and a correct class label, compute a second distance between the at least one feature vector and an incorrect class label, and determine the error based on the first distance and/or the second distance.

[0044] In some implementations, the at least one processor is configured to train the first subsystem based on the ground truth data. In some implementations, the at least one processor is further configured to train the first subsystem when the smart phone is charging and/or is idle.

[0045] In some implementations, the at least one processor can be configured to learn via the second subsystem an identity of the first image and the second image. In some implementations, the identity of the first image and the second image can include at least one label for the first image and the second image. In some implementations, the at least one processor can be further configured to teach via the second subsystem the identity of the first image and the second image to the first subsystem.

[0046] A method for manipulating a first image taken by a user is disclosed herein. The method includes analyzing using a Lifelong Deep Neural Network (L-DNN) implemented by a processor the first image. The L-DNN can include a first module and a second module. The method also includes extracting a feature set for the first image using the first module. The method also include determining using the second module similarity between the feature set and at least one previously extracted feature set. The at least one previously extracted feature set can be extracted from a plurality of previous images taken by a user. The method also includes in response to determining that the feature set and the at least one previously extracted feature set are similar, adding the first image to a cluster comprising the plurality of previous images.

[0047] In some implementations, the method also include in response to determining that the feature set and the at least one previously extracted feature sets are not similar: learning using the first module the feature set, and creating a first class for the first image.

[0048] The method also includes automatically applying at least one user setting associated with the plurality of previous images to the first image.

[0049] All combinations of the foregoing concepts and additional concepts discussed in greater detail below (provided such concepts are not mutually inconsistent) are contemplated as being part of the inventive subject matter disclosed herein. In particular, all combinations of claimed subject matter appearing at the end of this disclosure are contemplated as being part of the inventive subject matter disclosed herein. It should also be appreciated that terminology explicitly employed herein that also may appear in any disclosure incorporated by reference should be accorded a meaning most consistent with the particular concepts disclosed herein.

[0050] Other systems, processes, and features will become apparent to those skilled in the art upon examination of the following drawings and detailed description. It is intended that all such additional systems, processes, and features be included within this description, be within the scope of the present invention, and be protected by the accompanying claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0051] The skilled artisan will understand that the drawings primarily are for illustrative purposes and are not intended to limit the scope of the inventive subject matter described herein. The drawings are not necessarily to scale; in some instances, various aspects of the inventive subject matter disclosed herein may be shown exaggerated or enlarged in the drawings to facilitate an understanding of different features. In the drawings, like reference characters generally refer to like features (e.g., functionally similar and/or structurally similar elements).

[0052] FIG. **1** illustrates an overview of various components of the system, using a smart phone as an example compute Edge device.

[0053] FIG. **2** illustrates an overview of a Lifelong Deep Neural Network (L-DNN) as it relates to multiple compute Edges, either acting individually over a data stream, or connected peer-to-peer or via an intermediary compute server.

[0054] FIG. **3** illustrates an example workflow of the method for picture enhancement in a smart phone.

[0055] FIG. **4** illustrates image processing in a smart phone using a Lifelong Deep Neural Network (L-DNN).

DETAILED DESCRIPTION

[0056] FIG. **1** provides an overview of a compute Edge device **100** (in this example, a smart phone **100**). Other examples of edge devices include robots, drones, and IoT devices. The smart phone **100** has an on-board AI module **107** with one or, typically, several separate AI sub-modules **108** that process information. Each of these submodules **108** can be implemented as an L-DNN that serves a particular application, such as image classification. The on-board AI module **107** and the AI sub-modules **108** can be implemented in one or more suitable computer processors, such as graphics processor units (GPUs), field-programmable gate arrays (FPGAs), or application-specific integrated circuits (ASICs), with appropriate volatile and non-volatile memory and appropriate input/output interfaces.

[0057] The smart phone **100** includes or is coupled to one or more input sources. The smart phone's on-board input sources may include one or more digital cameras **110** (e.g., a front-facing camera and a rear-facing camera), a touch screen **112**, a keypad, buttons, a microphone **114**, an inertial measurement unit (IMU), an accelerometer, a gyroscope, and a network interface, such as an antenna or connection for a data cord. These input sources may acquire input **103** generated by a user **102**, such as finger swipes, button pushes, spoken commands, and device movements, and/or information about the environment, including images, video data, structured light data, audio data, and location data (e.g., Global Positioning System data). The smart phone **100** may collect this information about the environment in the form of sensor input **105**, possibly in response to user commands, e.g., taking a picture when the user pushes a button or recording audio or video data in response to a button push.

[0058] The smart phone **100** may also acquire input from network-generated events and data **104**, user (**102**) generated input **103** and/or endogenous input **106** generated by the device. Endogenous input may be events generated by the phone operating system, including clock events, app usage, app events (e.g., 3 apps were opened between 2:00 PM and 2:45 PM), events intrinsic to a specific app (e.g., calendar), device power status, device movement, and the like. The exact nature of inputs used during learning is defined by the nature of the application that uses on-device learning. For instance, a personal assistant application might use calendar events, alarms, call logs, etc., while a photo manipulation application might use camera settings, user inputs, time of day inputs, etc. An on-device learning system, such as an L-DNN system, uses one or more of these

data sources to learn and change the behavior of the AI module **107** and/or AI sub-modules **108**, such as changing picture parameters based on the type of picture or other applications **109**, etc.

Lifelong Deep Neural Network (L-DNN)

[0059] L-DNN implements a heterogeneous neural network architecture to combine a fast learning mode and a slow learning mode. In the fast learning mode, a real-time operating machine implementing an L-DNN, such as the smart phone **100** of FIG. **1**, learns new knowledge and new experiences quickly so that it can respond to the new knowledge almost immediately. In this mode, the learning rate in the fast learning subsystem is high to favor new knowledge and the corresponding new experiences, while the learning rate in the slow learning subsystem is set to a low value or zero to preserve old knowledge and the corresponding old experiences.

[0060] FIG. **2** provides an overview of L-DNN architecture where multiple devices, including a master edge/central server **220** and several compute edges **222** (e.g., smartphones, drones, robots, or other IoT devices), running L-DNNs operate in concert. Each device **220**, **222** receives sensory input **200** and feeds it to a corresponding L-DNN **206** comprising a slow learning Module A **202** and a fast learning Module B **204**. Each Module A **202** is based on a pre-learned (fixed weight) DNN and serves as feature extractor. It receives the sensory input **200**, extracts the relevant features into compressed representations of objects and feeds these representations to the corresponding Module B **204**. Module B **204** is capable of fast learning of these object representations. During supervised mode, through the interactions with the user, Module B **204** receives correct labels for the unfamiliar objects, quickly learns the association between each feature vector and corresponding label, and as a result can recognize these new objects immediately. In unsupervised mode, the system uses feature similarities between vectors to assign these vectors into the same or different classes.

[0061] The L-DNN **206** takes advantage of the fact that weights in the DNN are excellent feature extractors. The Module B **204** continuously processes the features extracted by Module A **202** as the input source **200** provides data. The Module B neural network classifier **204** uses fast, one-shot learning to associate these features with object classes.

[0062] In the fast learning mode, when a novel set of features is presented as input **200**, Module B **204** associates these features with a class label that is given by the user in supervised mode or generated internally in unsupervised mode. In either case, Module B **204** is now familiar with this input and can recognize it on the next presentation. The result of Module B **204** serves as an output of L-DNN **206** either by itself or as a combination with an output from a specific DNN layer from Module A **202**, depending on the task that the L-DNN **206** is solving.

[0063] The slow learning mode uses a set of labeled training inputs from the user in the similar way to conventional DNN training, but this set can be much smaller and can contain for example only the inputs for which L-DNN has shown poor performance. The user provides the label for each input **200** to Module B **204**. Module B **204** converts this label into a prototype feature vector that Module B **204** expects to see for this particular label. At the same time, Module A **202** does a forward pass on the input **200** and computes the resulting feature vector. A contrastive loss function computes the distance between the prototype and computed feature vectors and backpropagates the error through Module A **202** to make the computed feature vector closer to the prototype that Module B **204** expects for this input. A contrastive loss function either reduces or minimizes the distance between the prototype and the computed feature vectors if they match or increases or maximizes the distance between the prototype and the computed feature vectors if they don't match. This results in features of an object, e.g., a dog, extracted by Module A **202** to be more like a prototypical dog with features expected by Module B **204**. In a complementary procedure, the user can provide an additional label with input **200** to serve as a negative example for a given class, e.g., an image of a cat, and then a triplet loss function generates an error to move the feature vector of a dog further away from a prototypical feature vector for a cat and closer to prototypical feature vector for a dog.

[0064] Triplet loss is used in conventional DNNs to improve the quality of the feature vectors by adjusting the weights of the DNN as following. For each presented input, the feature vector is computed through the usual forward pass through the DNN. The triplet loss function compares this feature vector with the prototypical feature vector for the classes known to DNN. For classes that are not correct for this input, the distance between their prototypes and the input feature vectors needs to increase as a result of backpropagation. For the correct classes, the distance between class prototype and the input feature vector shall decrease as a result of backpropagation.

[0065] As a result, an iterative process that cycles through stages of fast learning and slow learning as the L-DNN system operates in a dynamic environment becomes possible.

[0066] Fast learning of new objects happens as discussed above during real time operation (e.g., when the device is active) of the L-DNN system in the environment. Module B **204** weights are updated to contain this new knowledge, Module A **202** weights are unchanged. This stage continues for as long as new input information continues to flow into the system.

[0067] When the situation allows to disconnect the system from environmental input (e.g., when the device is idle or offline) for a substantial period of time, and if the examples of some, preferably confusing objects are still available, slow learning within Module A **202** can happen. Examples of objects are presented to the system together with the ground truth as in conventional DNN training. A forward pass through Module A **202** of the L-DNN **206** creates a feature vector for each input. This feature vector activates Module B **204**, which provides positive (correct class label) and negative (incorrect class label) examples in triplet loss computation. A triplet loss function computes the error, which is backpropagated and adjusts the weights in Module A **202**. In this stage only the weights of Module A **202** change. Module B **204** weights stay constant to fix the prototypes in place and reduce forgetting of previously learned information.

[0068] The result of this iterative loop is a continuously learning L-DNN **206** that operates very close to how the brain operates according to the sleep consolidation hypothesis. The L-DNN **206** also inherits the flexibility of the brain

in the sense that if certain exemplars of objects disappear from the world and are no longer parts of the input, the L-DNN **206** gradually adjusts the prototypes towards the remaining exemplars that are still relevant for performance.

[0069] Additional advantages can be gained by L2 normalization of the feature vectors during training of a DNN used in Module A of L-DNN. L2 normalization reduces feature comparison to simple geometric operations further improving ease of clustering of similar objects.

[0070] Working in unsupervised mode, L-DNN allows an additional set of use cases where the user's involvement during learning process is either undesirable or impossible. An example of such a use case is described below.

Example Use Cases of Unsupervised L-DNN

[0071] The following use cases are non-limiting examples of how an L-DNN operating in an unsupervised regime can address technical problems in several image search, edit, and manipulation applications.

Using Unsupervised L-DNN for Image Manipulation Based on Clustering

[0072] Consider a photo management or editing applications developer wanting to automate the priority settings for editing or management actions based on one or more similarities between the image currently in processing and other images previously processed for a particular user. For example, if the user consistently applies certain filters to all the beach scenes, or consistently places them in a certain album, then the unsupervised L-DNN system can determine the similarity of a new scene to the beach scenes and suggest these filters and album as first choices in the user action menus.

[0073] The following use case illustrates a possible application of the technology described herein. The use of a L-DNN on a camera-enabled mobile device, such as a smart phone, allows for fine-grained optimization of photos taken and edited on the device. Users can apply smart filters and customize the look and feel of the images they take before sharing, printing, or otherwise using the photos.

[0074] Traditionally, a DNN used on a mobile device is trained before being installed and is only capable of applying the learning on which it was trained. There is no capability for the DNN to learn based on the device user's behavior. In other words, the DNN is limited to its factory functionality.

[0075] Heavy mobile photo users may take and want to adjust the settings on hundreds of images in a very short timeframe. The need to adjust each picture manually severely limits the number of images that a user can customize and increases the amount of effort required to maintain a constant throughput of personally-optimized photos.

[0076] The application of an L-DNN to the photo editing workflow allows each mobile device to learn the preferences of its user as that person customizes their photos. This application is run on the device, keeping user data local and ensuring data privacy and security. FIG. **3** illustrates a process **300** of learning at the edge for photography with an Edge device like the smart phone **100** in FIG. **1** or another device with a digital camera. In this process **300**, a picture is automatically examined by an on-device learning system—in this case, using L-DNN. At **304**, the Edge device examines a user-customized image **302** using the L-DNN and determines if the user-customized image **302** matches any previously learned image (e.g., at **306**), namely, if the image is classified as belonging to a previously encountered image class.

[0077] The user-customized image **302** is fed as input (e.g., input **200** in FIG. **2**) to an L-DNN (e.g., L-DNN **206** in FIG. **2**) executed on the smart phone **100**. A slow learning module of the L-DNN (e.g., Module A **202** in FIG. **2**) extracts the features of the picture and a fast learning module of the L-DNN (e.g., Module B **204** in FIG. **2**) analyzes the similarity between the extracted feature set and the feature sets extracted from the previous pictures this user took and processed. If the new features are sufficiently close (defined by the neural network parameters of the fast learning module (e.g., Module B **204** in FIG. **2**) and can be exposed in the user settings) to one of the previously learned prototypes, then the new picture is added to the cluster of similar pictures **308** and the Edge device stores customizations of this image in local memory in association with that learned image class. If the user-customized image **302** does not match any previously learned images, then at **310**, the L-DNN on the Edge device (smart phone **100**) learns the image **302** and creates an additional class. The new class is then immediately usable by the Edge device user as a new class, and the Edge device stores custom user settings in local memory in association with that class. Then the history of user actions for this cluster is used to order the actions in the menu of available actions. This results in a menu personally tailored not only to the specific user, but also to a specific class of images. In this case, no user input is required, and the on-device learning system using the L-DNN uses its own output as a teaching signal to modify the classification outcome.

[0078] This process **300** in FIG. **3** enables the L-DNN on the mobile device to learn the types of photos taken by the user and the settings associated with each of those photo types. The L-DNN may apply those settings to additional photos based on the photos' shared characteristics. This reduces or eliminates the need for the user to apply the same custom settings to multiple similar images and increases the efficiency with which the user can optimize multiple images.

Image Processing on a Smart Phone with L-DNN

[0079] FIG. **4** illustrates image processing **400** in a smart phone (e.g., the smart phone **100** in FIG. **1**) that can learn at the edge in a supervised or unsupervised fashion. At **402**, the smart phone **100** acquires a first image with a camera **110** (e.g., the smart phone's camera or another digital camera communicatively coupled to the smart phone). At **404**, one or more characteristics of the first image can be altered based on user input. For instance, the user can customize the first image to change characteristics such as brightness, contrast, and/or the like. At **406**, a first neural network (e.g., an L-DNN) can learn the characteristics and can create a new class based on the characteristics. At **408**, the smart phone **100** acquires a second image with the camera **110**. The L-DNN then examines the second image and determines if the second image matches any previously-learned images. If the L-DNN determines that the second image matches the first image, at **410**, the L-DNN classifies the second image in the new class as the first image. At **412**, the smart phone applies the user-customization (e.g., change(s) in the image characteristic(s)) that was applied to the first image to the second image.

[0080] This process is described with an example below:

[0081] A user is at the beach and takes five photographs (#**1**, **2**, **3**, **4**, **5**). The photos are not exactly the same but feature similar elements, such as sand, water, sky, and people.

[0082] The user opens photo #**1** and uses the phone's photo application to change the photo settings. Contrast is increased, brightness is decreased, and the blue of the water and sky are made more vibrant.

[0083] Photo #**1** and its user-applied settings are submitted to the L-DNN.

[0084] The L-DNN learns the characteristics of photo #**1**, creates a new "class" of knowledge and assigns it a photo class ID (e.g., ABC123). The user's custom settings are stored in association with that photo class ID.

[0085] The user opens photo #**2**. Before the user makes any edits, photo #**2** is submitted to the L-DNN.

[0086] The L-DNN examines photo #**2** and determines that photo #**2** is a member of photo class ABC123 because photo #**2** has similar characteristics to photo #**1**.

[0087] The user's stored settings are applied to photo #**2**. Photo #**2** is adjusted with the same settings as the user applied to the photo #**1**, but without the user having to re-create those settings.

[0088] While L-DNN as described above works in this application in unsupervised mode, it still retains all the capabilities of a full L-DNN. For example, all the local knowledge from a particular device **100** for the user can be shared and updated through brain melding with all the other devices this user can utilize to take, manipulate and/or edit images. Furthermore, the effects of user actions can be used as partial supervision for L-DNN without the need to force the user to explicitly label the clusters. For example, if the user consistently applies the same actions to several clusters, L-DNN can unify them in a single cluster and treat them as a single entity from this point on. On the other hand, if the user follows more than one common scenario for different members of the same cluster, the system can automatically adjust internal parameters and split this cluster into multiple smaller clusters.

[0089] Another feature for such applications that unsupervised L-DNN enables is batch image manipulation based on similarity. In this case the user provides an example image to the L-DNN, which selects all similar images from either local storage, cloud or all the devices of this user and allows image manipulation on all of these images as a batch. This feature does not require the L-DNN to return only the images it has seen before; any previously unseen image undergo similarity evaluation by L-DNN and are either added to the batch or skipped. Settings changes or filters are applied to all images in a batch. Similarly, all images in a batch can be moved to the same album, shared, or deleted. This may save time and effort in use cases in which a user wishes to edit or manipulate multiple photos.

Using Unsupervised L-DNN for Image Search

[0090] The last feature discussed in the previous use case can be taken to the industrial level of a large-scale image search based on similarities between images instead of verbal tags. In this use case the user provides a sample image to the L-DNN system. L-DNN either selects an existing cluster prototype or creates a new prototype if none of the existing prototypes is close enough. Then the L-DNN goes through all available images that can be in the large database on a server or even scattered across a multitude of servers on the internet. Images that are in the same cluster are selected and ranked based on the similarity to the prototype, and then returned to the user as a result of the search. In some implementations, this feature can be performed on an edge device such as smart phones, tablets, and other edge devices (e.g., drones, robots, IoT devices, cameras, etc.). In some implementations, this feature can be performed on distributed, multi-Edge systems (e.g., smart phones connected in a network, networked smart cameras, fleets of drones or self-driving vehicles, and the like).

CONCLUSION

[0091] While various inventive embodiments have been described and illustrated herein, those of ordinary skill in the art will readily envision a variety of other means and/or structures for performing the function and/or obtaining the results and/or one or more of the advantages described herein, and each of such variations and/or modifications is deemed to be within the scope of the inventive embodiments described herein. More generally, those skilled in the art will readily appreciate that all parameters, dimensions, materials, and configurations described herein are meant to be exemplary and that the actual parameters, dimensions, materials, and/or configurations will depend upon the specific application or applications for which the inventive teachings is/are used. Those skilled in the art will recognize or be able to ascertain, using no more than routine experimentation, many equivalents to the specific inventive embodiments described herein. It is, therefore, to be understood that the foregoing embodiments are presented by way of example only and that, within the scope of the appended claims and equivalents thereto, inventive embodiments may be practiced otherwise than as specifically described and claimed. Inventive embodiments of the present disclosure are directed to each individual feature, system, article, material, kit, and/or method described herein. In addition, any combination of two or more such features, systems, articles, materials, kits, and/or methods, if such features, systems, articles, materials, kits, and/or methods are not mutually inconsistent, is included within the inventive scope of the present disclosure.

[0092] The above-described embodiments can be implemented in any of numerous ways. For example, embodiments may be implemented using hardware, software or a combination thereof. When implemented in software, the software code can be executed on any suitable processor or collection of processors, whether provided in a single computer or distributed among multiple computers.

[0093] Further, it should be appreciated that a computer may be embodied in any of a number of forms, such as a rack-mounted computer, a desktop computer, a laptop computer, or a tablet computer. Additionally, a computer may be embedded in a device not generally regarded as a computer but with suitable processing capabilities, including a Personal Digital Assistant (PDA), a smart phone or any other suitable portable or fixed electronic device.

[0094] Also, a computer may have one or more input and output devices. These devices can be used, among other things, to present a user interface. Examples of output devices that can be used to provide a user interface include printers or display screens for visual presentation of output

and speakers or other sound generating devices for audible presentation of output. Examples of input devices that can be used for a user interface include keyboards, and pointing devices, such as mice, touch pads, and digitizing tablets. As another example, a computer may receive input information through speech recognition or in other audible format.

[0095] Such computers may be interconnected by one or more networks in any suitable form, including a local area network or a wide area network, such as an enterprise network, and intelligent network (IN) or the Internet. Such networks may be based on any suitable technology and may operate according to any suitable protocol and may include wireless networks, wired networks or fiber optic networks.

[0096] The various methods or processes outlined herein may be coded as software that is executable on one or more processors that employ any one of a variety of operating systems or platforms. Additionally, such software may be written using any of a number of suitable programming languages and/or programming or scripting tools, and also may be compiled as executable machine language code or intermediate code that is executed on a framework or virtual machine.

[0097] Also, various inventive concepts may be embodied as one or more methods, of which an example has been provided. The acts performed as part of the method may be ordered in any suitable way. Accordingly, embodiments may be constructed in which acts are performed in an order different than illustrated, which may include performing some acts simultaneously, even though shown as sequential acts in illustrative embodiments.

[0098] All publications, patent applications, patents, and other references mentioned herein are incorporated by reference in their entirety.

[0099] All definitions, as defined and used herein, should be understood to control over dictionary definitions, definitions in documents incorporated by reference, and/or ordinary meanings of the defined terms.

[0100] The indefinite articles "a" and "an," as used herein in the specification and in the claims, unless clearly indicated to the contrary, should be understood to mean "at least one."

[0101] The phrase "and/or," as used herein in the specification and in the claims, should be understood to mean "either or both" of the elements so conjoined, i.e., elements that are conjunctively present in some cases and disjunctively present in other cases. Multiple elements listed with "and/or" should be construed in the same fashion, i.e., "one or more" of the elements so conjoined. Other elements may optionally be present other than the elements specifically identified by the "and/or" clause, whether related or unrelated to those elements specifically identified. Thus, as a non-limiting example, a reference to "A and/or B", when used in conjunction with open-ended language such as "comprising" can refer, in one embodiment, to A only (optionally including elements other than B); in another embodiment, to B only (optionally including elements other than A); in yet another embodiment, to both A and B (optionally including other elements); etc.

[0102] As used herein in the specification and in the claims, "or" should be understood to have the same meaning as "and/or" as defined above. For example, when separating items in a list, "or" or "and/or" shall be interpreted as being inclusive, i.e., the inclusion of at least one, but also including more than one, of a number or list of elements, and, optionally, additional unlisted items. Only terms clearly indicated to the contrary, such as "only one of" or "exactly one of," or, when used in the claims, "consisting of," will refer to the inclusion of exactly one element of a number or list of elements. In general, the term "or" as used herein shall only be interpreted as indicating exclusive alternatives (i.e. "one or the other but not both") when preceded by terms of exclusivity, such as "either," "one of," "only one of," or "exactly one of." "Consisting essentially of," when used in the claims, shall have its ordinary meaning as used in the field of patent law.

[0103] As used herein in the specification and in the claims, the phrase "at least one," in reference to a list of one or more elements, should be understood to mean at least one element selected from any one or more of the elements in the list of elements, but not necessarily including at least one of each and every element specifically listed within the list of elements and not excluding any combinations of elements in the list of elements. This definition also allows that elements may optionally be present other than the elements specifically identified within the list of elements to which the phrase "at least one" refers, whether related or unrelated to those elements specifically identified. Thus, as a non-limiting example, "at least one of A and B" (or, equivalently, "at least one of A or B," or, equivalently "at least one of A and/or B") can refer, in one embodiment, to at least one, optionally including more than one, A, with no B present (and optionally including elements other than B); in another embodiment, to at least one, optionally including more than one, B, with no A present (and optionally including elements other than A); in yet another embodiment, to at least one, optionally including more than one, A, and at least one, optionally including more than one, B (and optionally including other elements); etc.

[0104] In the claims, as well as in the specification above, all transitional phrases such as "comprising," "including," "carrying," "having," "containing," "involving," "holding," "composed of," and the like are to be understood to be open-ended, i.e., to mean including but not limited to. Only the transitional phrases "consisting of" and "consisting essentially of" shall be closed or semi-closed transitional phrases, respectively, as set forth in the United States Patent Office Manual of Patent Examining Procedures, Section 2111.03.

1. A method of image processing with a smart phone executing a neural network comprising a fast learning subsystem and a slow learning subsystem, the method comprising:

acquiring a first image with a camera of the smart phone;

altering at least one characteristic of the first image in response to input from a user;

learning, by the fast learning subsystem, a new class based on the at least one characteristic of the first image altered in response to the input from the user;

acquiring a second image with the camera of the smart phone;

classifying the second image in the new class with the fast learning subsystem;

in response to classifying the second image in the new class, automatically altering at least one characteristic of the second image on the smart phone based on the at least one characteristic of the first image altered in response to the input from the user;

providing the first image and the second image to the slow learning subsystem; and

generating, by the slow learning subsystem, ground truth data based on the first image and the second image.

2. The method of claim **1**, wherein the neural network is a Lifelong Deep Neural Network.

3. The method of claim **1**, further comprising:

providing the ground truth data to the fast learning subsystem.

4. The method of claim **3**, wherein the slow learning subsystem is a Deep Neural Network.

5. The method of claim **3**, wherein providing the first image and the second image to the slow learning subsystem when the smart phone is idle.

6. The method of claim **5**, further comprising:

generating, using the slow learning subsystem, at least one feature vector for the first image and the second image;

determining, via the fast learning subsystem, an error for the at least one feature vector based on a triplet loss function; and

adjusting at least one weight of the slow learning subsystem based on the error.

7. The method of claim **6**, wherein determining the error includes:

computing a first distance between the at least one feature vector and a correct class label;

computing a second distance between the at least one feature vector and an incorrect class label; and

determining the error based on the first distance and/or the second distance.

8. The method of claim **3**, further comprising:

training the fast learning subsystem based on the ground truth data.

9. The method of claim **8**, wherein training the fast learning subsystem occurs when the smart phone is charging and/or is idle.

10. The method of claim **3**, further comprising:

learning, by the slow learning subsystem, an identity of the first image and the second image.

11. The method of claim **10**, wherein the identity of the first image and the second image includes at least one label for the first image and the second image.

12. The method of claim **11**, further comprising:

teaching, by the slow learning subsystem, the identity of the first image and the second image to the fast learning subsystem.

13. A method of image processing with a smart phone, the method comprising:

acquiring a first image with a camera of the smart phone; and

while the smart phone is in an idle state:

creating a first label for the first image with a first subsystem included in a neural network executed by a processor of the smart phone; and

teaching, by the first subsystem, the first label to a second subsystem included in the neural network executed by the processor of the smart phone.

14. The method of claim **13**, further comprising:

acquiring a second image with the camera of the smart phone; and

applying the first label to the second image with the second subsystem.

15. The method of claim **13**, wherein the neural network is a Lifelong Deep Neural Network.

16. The method of claim **15**, wherein the second subsystem enables real-time learning.

17. The method of claim **13**, wherein teaching includes teaching the first label to the second subsystem via back-propagation.

18. The method of claim **13**, further comprising:

acquiring a second image with a camera of the smart phone;

determining, by the second subsystem, an association between a feature vector representing an object in the second image and a second label; and

applying the second label to the object in the second image.

19. The method of claim **18**, wherein the second label is received from a user.

20. The method of claim **18**, wherein the second label is generated by the neural network.

* * * * *