



- (51) International Patent Classification:
G06N 3/098 (2023.01) *G06N 3/086* (2023.01)
- (21) International Application Number:
PCT/EP2024/074118
- (22) International Filing Date:
29 August 2024 (29.08.2024)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
18/531,983 07 December 2023 (07.12.2023) US
- (71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, New York, 10504 (US).
- (72) Inventors: **HICKS, Andrew, Christian, Morrell**; IBM Corporation, 2455 South Road, Poughkeepsie, New York 12601-5400 (US). **WERNER, John, Stephen**; IBM Corporation, 2455 South Road, Poughkeepsie, New York 12601-5400 (US). **FROTA, Rafaela**; IBM Corporation, 2455 South Road, Poughkeepsie, New York 12601-5400 (US). **GHOLAMI, Faezeh**; IBM Corporation, 2455 South Road, Poughkeepsie, New York 12601-5400 (US).
- (74) Agent: **STEVEN-FOUNTAIN, Jessica**; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Hursley, Winchester Hampshire SO21 2JN (GB).
- (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,

(54) Title: SELECTIVE BREEDING FOR DIVERGENT NEURAL NETWORKS IN AN EDGE COMPUTING ENVIRONMENT

500

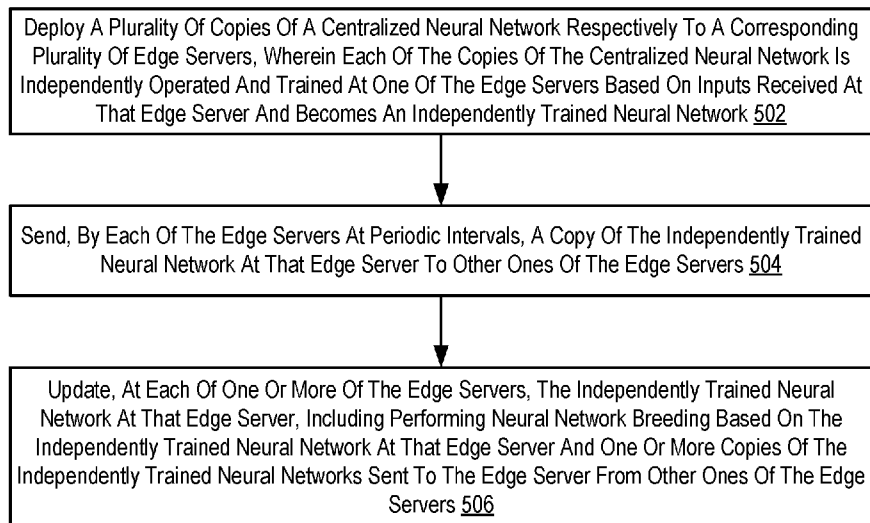


FIG. 5

(57) Abstract: Selective breeding for divergent neural networks in an edge computing environment includes deploying a plurality of copies of a centralized neural network respectively to a corresponding plurality of edge servers, wherein each of the copies of the centralized neural network is independently operated and trained at one of the edge servers based on inputs received at that edge server and becomes an independently trained neural network. Each of the edge servers at periodic intervals sends a copy of the independently trained neural network at that edge server to other ones of the edge servers. At each of one or more of the edge servers, the independently trained neural network at that edge server is updated, including performing neural network breeding based on the independently trained neural network at that edge server and one or more copies of the independently trained neural networks sent to the edge server.

CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, 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, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, 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, ME, 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))

SELECTIVE BREEDING FOR DIVERGENT NEURAL NETWORKS IN AN EDGE COMPUTING ENVIRONMENT**BACKGROUND**

[0001] The present disclosure relates to methods, apparatus, and products for selective breeding for divergent neural networks in an edge computing environment.

SUMMARY

[0002] According to embodiments of the present disclosure, various methods, apparatus and products for selective breeding for divergent neural networks in an edge computing environment are described herein. In some aspects, selective breeding for divergent neural networks in an edge computing environment includes deploying a plurality of copies of a centralized neural network respectively to a corresponding plurality of edge servers, wherein each of the copies of the centralized neural network is independently operated and trained at one of the edge servers based on inputs received at that edge server and becomes an independently trained neural network. Each of the edge servers at periodic intervals sends a copy of the independently trained neural network at that edge server to other ones of the edge servers. At each of one or more of the edge servers, the independently trained neural network at that edge server is updated, including performing neural network breeding based on the independently trained neural network at that edge server and one or more copies of the independently trained neural networks sent to the edge server from other ones of the edge servers.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] FIG. 1 sets forth an example computing environment according to aspects of the present disclosure.

[0004] FIG. 2 sets forth an example implementation of a system for selective breeding for divergent neural networks in an edge computing environment according to aspects of the present disclosure.

FIG. 3 sets forth a flowchart of an example method for selective breeding for divergent neural networks in an edge computing environment according to aspects of the present disclosure.

[0005] FIG. 4 sets forth a flowchart of an example method for updating edge neural networks according to aspects of the present disclosure.

[0006] FIG. 5 sets forth a flowchart of an example method for selective breeding for divergent neural networks in an edge computing environment according to another aspect of the present disclosure.

DETAILED DESCRIPTION

[0007] Neural networks may be deployed in edge computing environments to increase bandwidth and reduce latency for end users. It may be useful to continuously train the neural network such that it keeps improving over time. When neural networks are individually trained at edge systems, one or more edge systems may drift and diverge from other edge systems. In some instances, it may be useful to allow edge nodes to train differently and drift from one another to adapt to the incoming data within a specific physical location (e.g., near the edge node). Although it may be beneficial to allow the edge nodes to drift from one another, some neural networks may not perform as well with certain inputs since those inputs have not been seen at specific edge nodes.

[0008] In some examples, an initial centralized neural network is created in a cloud data center using training data at the cloud data center. In some examples, the cloud data center deploys individual neural networks (e.g., copies of the centralized neural network) to edge servers (also referred to herein as edge nodes), and allows them to train differently and drift from one another to adapt to the incoming data within their specific physical locations. As the individual neural networks are operated at the edge servers, each edge server captures and stores all or a subset of data points associated with the neural network at that edge server. At periodic intervals (e.g., when a time threshold is reached, such as 1 week, 1 month, etc.), neural networks from the edge nodes are shared with each other. In some examples, each edge node shares its drifted neural network with other edge nodes. In some examples, an edge neural network and an associated set or subset of data points from each edge node may also be sent to the cloud data center or other fog nodes for comparison, testing, and in some cases, breeding.

[0009] In some examples, each edge node performs a test run on the incoming neural networks using data from that edge node to determine a fitness score (also referred to herein as a fitness measure) for each of the neural networks. The test run can use recent stored inputs seen at the edge node to run on the neural networks from other edge nodes. In embodiments where no input data is saved, new input data can be run on the neural network specific to that edge node to generate outputs for new incoming data and run the same inputs on other neural networks shared from other edge nodes either simultaneously or in the background when processing power is available to generate fitness scores. If a high fitness score is observed at an edge node for a given incoming neural network (e.g., the fitness score exceeds a score threshold indicating that it performed well), breeding may be performed between that incoming neural network and the neural network that was previously operating on that edge node where the data points were captured to generate an updated or optimized neural network that now accounts for additional inputs that may be observed. Other incoming neural networks that do not meet a fitness threshold may be discarded and not used in the breeding, and may continue to be used as is on the original edge nodes that they came from. After breeding, each edge node may discard the neural networks received from other edge nodes, and each edge node is left with only one neural network that was either bred with one or more other neural networks or left alone because no other neural networks met the fitness score threshold. In some examples, the

stored data points at each edge node may be discarded after updating to make room for new data points that are captured and stored until the next time threshold is reached. This process may be periodically repeated to provide continual updating of the neural networks at the edge nodes.

[0010] Some examples operate individual edge neural networks on one or more edge servers and allow those individual edge neural networks to train themselves at the edge independent of other edge servers using only the inputs seen at their edge server. The individual edge neural networks may begin to differ from one another. At periodic intervals (e.g., once per day, once per week, etc.), the individual edge neural network at each edge server are sent to the other edge servers. Each edge server then determines a fitness measurement for each of the edge neural networks received from the other edge servers using data points seen at the receiving edge server. Each edge server then performs neural network genetic breeding based on the fitness measurements to potentially update its edge neural network by merging its edge neural network with one or more other edge neural networks. This method may then be repeated such that continuous training and updating of the edge neural networks occur without sending any input data to the cloud data center or other edge servers which reduces network bandwidth. Some examples disclosed herein use NeuroEvolution of Augmenting Topologies (NEAT) for updating neural networks running on separate edge servers. In some examples, the neural network breeding may be performed using a hypercube-based NEAT (hyperNEAT) calculation which performs a weighted average, based on the fitness measure, of the parameters extracted from each neuron. Some examples may be implemented as a service (aaS) for deployment of artificial intelligence (AI) in edge computing environments.

[0011] An example of the present disclosure is directed to a method for selective breeding for divergent neural networks in an edge computing environment, which includes deploying a plurality of copies of a centralized neural network respectively to a corresponding plurality of edge servers, wherein each of the copies of the centralized neural network is independently operated and trained at one of the edge servers based on inputs received at that edge server and becomes an independently trained neural network. The method includes sending, by each of the edge servers at periodic intervals, a copy of the independently trained neural network at that edge server to other ones of the edge servers. The method includes updating, at each of one or more of the edge servers, the independently trained neural network at that edge server, including performing neural network breeding based on the independently trained neural network at that edge server and one or more copies of the independently trained neural networks sent to the edge server from other ones of the edge servers.

[0012] Examples of the method include various technical features that yield technical effects that provide various improvements to computer technology. For instance, some examples include the technical feature of deploying a plurality of copies of a centralized neural network respectively to a corresponding plurality of edge servers, wherein each of the copies of the centralized neural network is independently operated and trained at one of the edge servers based on inputs received at that edge server. This technical feature yields the technical effect

of continually training the neural networks such that they keep improving over time. Some examples include the technical features of sending, by each of the edge servers at periodic intervals, a copy of the independently trained neural network at that edge server to other ones of the edge servers; and updating, at each of one or more of the edge servers, the independently trained neural network at that edge server, including performing neural network breeding based on the independently trained neural network at that edge server and one or more copies of the independently trained neural networks sent to the edge server from other ones of the edge servers. These technical features yield the technical effect of improving performance of edge neural networks and reducing drift between edge servers, while also reducing network traffic, improving network performance, and reducing power consumption.

[0013] The method may also include storing, at each of the edge servers, a set of data points associated with the independently trained neural network at that edge server. This has the technical effect of maintaining information that may be used to test the performance of other edge neural networks and identify edge neural networks that are good candidates for breeding.

[0014] The method may also include calculating, at each of the edge servers based on the set of data points stored at the edge server, a fitness measure for each of the one or more copies of the independently trained neural networks sent to the edge server from the other ones of the edge servers. This has the technical effect of calculating measures that may be used to compare the performance of edge neural networks to identify edge neural networks that are good candidates for breeding.

[0015] In some examples of the method, the neural network breeding includes discarding, at each of the edge servers, any of the one or more copies of the independently trained neural networks sent to the edge server that have a corresponding fitness measure below a threshold. This has the technical effect of improving the performance of the updating by discarding any of the copies of the independently trained neural networks that have a relatively low fitness measure.

[0016] In some examples of the method, the neural network breeding includes selecting, at each of the edge servers based on the fitness measures, top performing ones of the one or more copies of the independently trained neural networks sent to the edge server. This has the technical effect of improving the performance of the updating by selecting and using top performing ones of the copies of the independently trained neural networks.

[0017] In some examples of the method, the neural network breeding includes performing, at each of the edge servers, a hyperNEAT calculation that includes determining a weighted average, based on the fitness measures, of parameters extracted from neurons in the one or more copies of the independently trained neural networks sent to

the edge server. This has the technical effect of improving the performance of the updating by using a weighted average based on the fitness measures so that higher fitness measures may provide a greater influence.

[0018] In some examples of the method, the periodic interval is defined by a preset period of time. This has the technical effect of reducing drift and improving neural network performance using a preset period of time for updates that may be set by an administrator, for example. This provides flexibility for the timing of updates and network usage.

[0019] In some examples of the method, the periodic interval is defined by a preset amount of drift occurring in one or more of the independently trained neural networks. This has the technical effect of reducing drift and improving neural network performance using a preset amount of drift for updates that may be set by an administrator, for example. This helps to ensure that there is not significant drift between updates.

[0020] Another example of the present disclosure is directed to an apparatus for selective breeding for divergent neural networks in an edge computing environment, which includes a processing device, and a memory operatively coupled to the processing device. The memory stores computer program instructions that, when executed, cause the processing device to deploy a plurality of copies of a centralized neural network respectively to a corresponding plurality of edge servers, wherein each of the copies of the centralized neural network is independently operated and trained at one of the edge servers based on inputs received at that edge server and becomes an independently trained neural network. The memory also stores computer program instructions that, when executed, cause the processing device to send, by each of the edge servers at periodic intervals, a copy of the independently trained neural network at that edge server to other ones of the edge servers. The memory also stores computer program instructions that, when executed, cause the processing device to update, at each of one or more of the edge servers, the independently trained neural network at that edge server, including performing neural network breeding based on the independently trained neural network at that edge server and one or more copies of the independently trained neural networks sent to the edge server from other ones of the edge servers.

[0021] Examples of the apparatus include various technical features that yield technical effects that provide various improvements to computer technology. For instance, some examples include the technical feature of deploying a plurality of copies of a centralized neural network respectively to a corresponding plurality of edge servers, wherein each of the copies of the centralized neural network is independently operated and trained at one of the edge servers based on inputs received at that edge server. This technical feature yields the technical effect of continually training the neural networks such that they keep improving over time. Some examples include the technical features of sending, by each of the edge servers at periodic intervals, a copy of the independently trained neural network at that edge server to other ones of the edge servers; and updating, at each of one or more of the edge servers, the independently trained neural network at that edge server, including performing neural network

breeding based on the independently trained neural network at that edge server and one or more copies of the independently trained neural networks sent to the edge server from other ones of the edge servers. These technical features yield the technical effect of improving performance of edge neural networks and reducing drift between edge servers, while also reducing network traffic, improving network performance, and reducing power consumption.

[0022] The memory of the apparatus may also store computer program instructions that, when executed, cause the processing device to store, at each of the edge servers, a set of data points associated with the independently trained neural network at that edge server. This has the technical effect of maintaining information that may be used to test the performance of other edge neural networks and identify edge neural networks that are good candidates for breeding.

[0023] The memory of the apparatus may also store computer program instructions that, when executed, cause the processing device to calculate, at each of the edge servers based on the set of data points stored at the edge server, a fitness measure for each of the one or more copies of the independently trained neural networks sent to the edge server from the other ones of the edge servers. This has the technical effect of calculating measures that may be used to compare the performance of edge neural networks to identify edge neural networks that are good candidates for breeding.

[0024] In some examples of the apparatus, the neural network breeding includes discarding, at each of the edge servers, any of the one or more copies of the independently trained neural network sent to the edge server that have a corresponding fitness measure below a threshold. This has the technical effect of improving the performance of the updating by discarding any of the copies of the independently trained neural networks that have a relatively low fitness measure.

[0025] In some examples of the apparatus, the neural network breeding includes selecting, at each of the edge servers based on the fitness measures, top performing ones of the one or more copies of the independently trained neural networks sent to the edge server. This has the technical effect of improving the performance of the updating by selecting and using top performing ones of the copies of the independently trained neural networks.

[0026] In some examples of the apparatus, the neural network breeding includes performing, at each of the edge servers, a hyperNEAT calculation that includes determining a weighted average, based on the fitness measures, of parameters extracted from neurons in the one or more copies of the independently trained neural networks sent to the edge server. This has the technical effect of improving the performance of the updating by using a weighted average based on the fitness measures so that higher fitness measures may provide a greater influence.

[0027] In some examples of the apparatus, the periodic interval is defined by a preset period of time. This has the technical effect of reducing drift and improving neural network performance using a preset period of time for updates that may be set by an administrator, for example. This provides flexibility for the timing of updates and network usage.

[0028] In some examples of the apparatus, the periodic interval is defined by a preset amount of drift occurring in one or more of the independently trained neural networks. This has the technical effect of reducing drift and improving neural network performance using a preset amount of drift for updates that may be set by an administrator, for example. This helps to ensure that there is not significant drift between updates.

[0029] Another example of the present disclosure is directed to a computer program product for selective breeding for divergent neural networks in an edge computing environment, comprising a computer readable storage medium, wherein the computer readable storage medium comprises computer program instructions that, when executed deploy a plurality of copies of a centralized neural network respectively to a corresponding plurality of edge servers, wherein each of the copies of the centralized neural network is independently operated and trained at one of the edge servers based on inputs received at that edge server and becomes an independently trained neural network; send, by each of the edge servers at periodic intervals, a copy of the independently trained neural network at that edge server to other ones of the edge servers; and update, at each of one or more of the edge servers, the independently trained neural network at that edge server, including performing neural network breeding based on the independently trained neural network at that edge server and one or more copies of the independently trained neural networks sent to the edge server from other ones of the edge servers.

[0030] Examples of the computer program product include various technical features that yield technical effects that provide various improvements to computer technology. For instance, some examples include the technical feature of deploying a plurality of copies of a centralized neural network respectively to a corresponding plurality of edge servers, wherein each of the copies of the centralized neural network is independently operated and trained at one of the edge servers based on inputs received at that edge server. This technical feature yields the technical effect of continually training the neural networks such that they keep improving over time. Some examples include the technical features of sending, by each of the edge servers at periodic intervals, a copy of the independently trained neural network at that edge server to other ones of the edge servers; and updating, at each of one or more of the edge servers, the independently trained neural network at that edge server, including performing neural network breeding based on the independently trained neural network at that edge server and one or more copies of the independently trained neural networks sent to the edge server from other ones of the edge servers. These technical features yield the technical effect of improving performance of edge neural networks and reducing drift between edge servers, while also reducing network traffic, improving network performance, and reducing power consumption.

[0031] In some examples of the computer program product, the computer readable storage medium may further comprise computer program instructions that, when executed, store, at each of the edge servers, a set of data points associated with the independently trained neural network at that edge server. This has the technical effect of maintaining information that may be used to test the performance of other edge neural networks and identify edge neural networks that are good candidates for breeding.

[0032] In some examples of the computer program product, the computer readable storage medium may further comprise computer program instructions that, when executed, calculate, at each of the edge servers based on the set of data points stored at the edge server, a fitness measure for each of the one or more copies of the independently trained neural networks sent to the edge server from the other ones of the edge servers. This has the technical effect of calculating measures that may be used to compare the performance of edge neural networks to identify edge neural networks that are good candidates for breeding.

[0033] In some examples of the computer program product, the neural network breeding includes performing, at each of the edge servers, a hyperNEAT calculation that includes determining a weighted average, based on the fitness measures, of parameters extracted from neurons in the one or more copies of the independently trained neural networks sent to the edge server. This has the technical effect of improving the performance of the updating by using a weighted average based on the fitness measures so that higher fitness measures may provide a greater influence.

[0034] In some examples, each edge server may contain one individual neural network. In some examples, no input data seen at any of the one or more edge servers is sent to the cloud data center. The edge neural networks are periodically shared amongst the edge servers to update the edge neural networks using neural network breeding. The periodic interval may be set by a network administrator. The periodic interval may be triggered when a preset amount of time has expired or a preset amount of drift has occurred in one or more of the individual edge neural networks. The neural network breeding may discard edge neural networks with fitness measures below a threshold. The neural network breeding may select only the top performing edge neural networks based on their fitness measures.

[0035] Some examples of the present disclosure use unsupervised models and unsupervised learning. Some examples reduce data congestion within an edge computing environment to improve network performance. Some examples use fitness measurements at edge compute nodes to breed edge neural networks and update the edge neural network at each edge server. Some examples update edge neural networks without sending all training data seen at the edge nodes back to a centralized server which could put significant strain on a network. This strain could increase latency and increase power of all nodes back to the centralized location. Some examples save on power of systems used to support an edge network.

[0036] Some examples of the present disclosure optimize deployment of a neural network across multiple edge nodes where it is desired to allow those neural networks to train independent of each other based on the physical location of the edge nodes. Some examples also allow for improvement of the neural networks at each node with selective breeding. This allows neural networks to diverge from one another which can help reduce the overall size and performance of neural networks because a single neural network does not have to be able to handle all types of different input data from the different edge nodes.

[0037] FIG. 1 sets forth an example computing environment according to aspects of the present disclosure. Computing environment 100 contains an example of an environment for the execution of at least some of the computer code involved in performing the various methods described herein, such as neural network updating code 107. In addition to neural network updating code 107, computing environment 100 includes, for example, computer 101, wide area network (WAN) 102, end user device (EUD) 103, remote server 104, public cloud 105, and private cloud 106. In this embodiment, computer 101 includes processor set 110 (including processing circuitry 120 and cache 121), communication fabric 111, volatile memory 112, persistent storage 113 (including operating system 122 and neural network updating code 107, as identified above), peripheral device set 114 (including user interface (UI) device set 123, storage 124, and Internet of Things (IoT) sensor set 125), and network module 115. Remote server 104 includes remote database 130. Public cloud 105 includes gateway 140, cloud orchestration module 141, host physical machine set 142, virtual machine set 143, and container set 144.

[0038] Computer 101 may take the form of a desktop computer, laptop computer, tablet computer, smart phone, smart watch or other wearable computer, mainframe computer, quantum computer or any other form of computer or mobile device now known or to be developed in the future that is capable of running a program, accessing a network or querying a database, such as remote database 130. As is well understood in the art of computer technology, and depending upon the technology, performance of a computer-implemented method may be distributed among multiple computers and/or between multiple locations. On the other hand, in this presentation of computing environment 100, detailed discussion is focused on a single computer, specifically computer 101, to keep the presentation as simple as possible. Computer 101 may be located in a cloud, even though it is not shown in a cloud in FIG. 1. On the other hand, computer 101 is not required to be in a cloud except to any extent as may be affirmatively indicated.

[0039] Processor set 110 includes one, or more, computer processors of any type now known or to be developed in the future. Processing circuitry 120 may be distributed over multiple packages, for example, multiple, coordinated integrated circuit chips. Processing circuitry 120 may implement multiple processor threads and/or multiple processor cores. Cache 121 is memory that is located in the processor chip package(s) and is typically used for data or code that should be available for rapid access by the threads or cores running on processor set 110. Cache memories are typically organized into multiple levels depending upon relative proximity to the

processing circuitry. Alternatively, some, or all, of the cache for the processor set may be located "off chip." In some computing environments, processor set 110 may be designed for working with qubits and performing quantum computing.

[0040] Computer readable program instructions are typically loaded onto computer 101 to cause a series of operational steps to be performed by processor set 110 of computer 101 and thereby effect a computer-implemented method, such that the instructions thus executed will instantiate the methods specified in flowcharts and/or narrative descriptions of computer-implemented methods included in this document. These computer readable program instructions are stored in various types of computer readable storage media, such as cache 121 and the other storage media discussed below. The program instructions, and associated data, are accessed by processor set 110 to control and direct performance of the computer-implemented methods. In computing environment 100, at least some of the instructions for performing the computer-implemented methods may be stored in neural network updating code 107 in persistent storage 113.

[0041] Communication fabric 111 is the signal conduction path that allows the various components of computer 101 to communicate with each other. Typically, this fabric is made of switches and electrically conductive paths, such as the switches and electrically conductive paths that make up buses, bridges, physical input / output ports and the like. Other types of signal communication paths may be used, such as fiber optic communication paths and/or wireless communication paths.

[0042] Volatile memory 112 is any type of volatile memory now known or to be developed in the future. Examples include dynamic type random access memory (RAM) or static type RAM. Typically, volatile memory 112 is characterized by random access, but this is not required unless affirmatively indicated. In computer 101, the volatile memory 112 is located in a single package and is internal to computer 101, but, alternatively or additionally, the volatile memory may be distributed over multiple packages and/or located externally with respect to computer 101.

[0043] Persistent storage 113 is any form of non-volatile storage for computers that is now known or to be developed in the future. The non-volatility of this storage means that the stored data is maintained regardless of whether power is being supplied to computer 101 and/or directly to persistent storage 113. Persistent storage 113 may be a read only memory (ROM), but typically at least a portion of the persistent storage allows writing of data, deletion of data and re-writing of data. Some familiar forms of persistent storage include magnetic disks and solid state storage devices. Operating system 122 may take several forms, such as various known proprietary operating systems or open source Portable Operating System Interface-type operating systems that employ a kernel. The code included in neural network updating code 107 typically includes at least some of the computer code involved in performing the computer-implemented methods described herein.

[0044] Peripheral device set 114 includes the set of peripheral devices of computer 101. Data communication connections between the peripheral devices and the other components of computer 101 may be implemented in various ways, such as Bluetooth connections, Near-Field Communication (NFC) connections, connections made by cables (such as universal serial bus (USB) type cables), insertion-type connections (for example, secure digital (SD) card), connections made through local area communication networks and even connections made through wide area networks such as the internet. In various embodiments, UI device set 123 may include components such as a display screen, speaker, microphone, wearable devices (such as goggles and smart watches), keyboard, mouse, printer, touchpad, game controllers, and haptic devices. Storage 124 is external storage, such as an external hard drive, or insertable storage, such as an SD card. Storage 124 may be persistent and/or volatile. In some embodiments, storage 124 may take the form of a quantum computing storage device for storing data in the form of qubits. In embodiments where computer 101 is required to have a large amount of storage (for example, where computer 101 locally stores and manages a large database), this storage may be provided by peripheral storage devices designed for storing very large amounts of data, such as a storage area network (SAN) that is shared by multiple, geographically distributed computers. IoT sensor set 125 is made up of sensors that can be used in Internet of Things applications. For example, one sensor may be a thermometer and another sensor may be a motion detector.

[0045] Network module 115 is the collection of computer software, hardware, and firmware that allows computer 101 to communicate with other computers through WAN 102. Network module 115 may include hardware, such as modems or Wi-Fi signal transceivers, software for packetizing and/or de-packetizing data for communication network transmission, and/or web browser software for communicating data over the internet. In some embodiments, network control functions and network forwarding functions of network module 115 are performed on the same physical hardware device. In other embodiments (for example, embodiments that utilize software-defined networking (SDN)), the control functions and the forwarding functions of network module 115 are performed on physically separate devices, such that the control functions manage several different network hardware devices. Computer readable program instructions for performing the computer-implemented methods can typically be downloaded to computer 101 from an external computer or external storage device through a network adapter card or network interface included in network module 115.

[0046] WAN 102 is any wide area network (for example, the internet) capable of communicating computer data over non-local distances by any technology for communicating computer data, now known or to be developed in the future. In some embodiments, the WAN 102 may be replaced and/or supplemented by local area networks (LANs) designed to communicate data between devices located in a local area, such as a Wi-Fi network. The WAN and/or LANs typically include computer hardware such as copper transmission cables, optical transmission fibers, wireless transmission, routers, firewalls, switches, gateway computers and edge servers.

[0047] End user device (EUD) 103 is any computer system that is used and controlled by an end user (for example, a customer of an enterprise that operates computer 101), and may take any of the forms discussed above in connection with computer 101. EUD 103 typically receives helpful and useful data from the operations of computer 101. For example, in a hypothetical case where computer 101 is designed to provide a recommendation to an end user, this recommendation would typically be communicated from network module 115 of computer 101 through WAN 102 to EUD 103. In this way, EUD 103 can display, or otherwise present, the recommendation to an end user. In some embodiments, EUD 103 may be a client device, such as thin client, heavy client, mainframe computer, desktop computer and so on.

[0048] Remote server 104 is any computer system that serves at least some data and/or functionality to computer 101. Remote server 104 may be controlled and used by the same entity that operates computer 101. Remote server 104 represents the machine(s) that collect and store helpful and useful data for use by other computers, such as computer 101. For example, in a hypothetical case where computer 101 is designed and programmed to provide a recommendation based on historical data, then this historical data may be provided to computer 101 from remote database 130 of remote server 104.

[0049] Public cloud 105 is any computer system available for use by multiple entities that provides on-demand availability of computer system resources and/or other computer capabilities, especially data storage (cloud storage) and computing power, without direct active management by the user. Cloud computing typically leverages sharing of resources to achieve coherence and economies of scale. The direct and active management of the computing resources of public cloud 105 is performed by the computer hardware and/or software of cloud orchestration module 141. The computing resources provided by public cloud 105 are typically implemented by virtual computing environments that run on various computers making up the computers of host physical machine set 142, which is the universe of physical computers in and/or available to public cloud 105. The virtual computing environments (VCEs) typically take the form of virtual machines from virtual machine set 143 and/or containers from container set 144. It is understood that these VCEs may be stored as images and may be transferred among and between the various physical machine hosts, either as images or after instantiation of the VCE. Cloud orchestration module 141 manages the transfer and storage of images, deploys new instantiations of VCEs and manages active instantiations of VCE deployments. Gateway 140 is the collection of computer software, hardware, and firmware that allows public cloud 105 to communicate through WAN 102.

[0050] Some further explanation of virtualized computing environments (VCEs) will now be provided. VCEs can be stored as "images." A new active instance of the VCE can be instantiated from the image. Two familiar types of VCEs are virtual machines and containers. A container is a VCE that uses operating-system-level virtualization. This refers to an operating system feature in which the kernel allows the existence of multiple isolated user-space instances, called containers. These isolated user-space instances typically behave as real computers from the point

of view of programs running in them. A computer program running on an ordinary operating system can utilize all resources of that computer, such as connected devices, files and folders, network shares, CPU power, and quantifiable hardware capabilities. However, programs running inside a container can only use the contents of the container and devices assigned to the container, a feature which is known as containerization.

[0051] Private cloud 106 is similar to public cloud 105, except that the computing resources are only available for use by a single enterprise. While private cloud 106 is depicted as being in communication with WAN 102, in other embodiments a private cloud may be disconnected from the internet entirely and only accessible through a local/private network. A hybrid cloud is a composition of multiple clouds of different types (for example, private, community or public cloud types), often respectively implemented by different vendors. Each of the multiple clouds remains a separate and discrete entity, but the larger hybrid cloud architecture is bound together by standardized or proprietary technology that enables orchestration, management, and/or data/application portability between the multiple constituent clouds. In this embodiment, public cloud 105 and private cloud 106 are both part of a larger hybrid cloud.

[0052] FIG. 2 sets forth an example implementation of a system 200 for selective breeding for divergent neural networks according to aspects of the present disclosure. System 200 includes cloud data center 202, and a plurality of edge servers 212(1)-212(n) (collectively referred to as edge servers 212). Cloud data center 202 is communicatively coupled to the edge servers 212 via network 210. Network 210 may be a wired or wireless network, and may use any communication protocol that allows data to be transferred between components of the system 200, including, for example, PCIe, I²C, Bluetooth, Wi-Fi, Cellular (e.g., 3G, 4G, 5G), Ethernet, fiber optics, as well as others. In some examples, network 210 may include other fog nodes.

[0053] Cloud data center 202 includes centralized neural network 204, neural network breeding module 206, and training data 208. In examples in which network 210 includes fog nodes, neural network breeding module 206 may be located in any of those fog nodes. Cloud data center 202 may include one or more computing devices (e.g., servers, compute nodes, etc.), which may be scattered among one or more different physical locations. Edge servers 212(1)-212(n) respectively include edge neural networks 214(1)-214(n) (collectively referred to as edge neural networks 214), neural network training modules 216(1)-216(n) (collectively referred to as neural network training modules 216), fitness measures 218(1)-218(n) (collectively referred to as fitness measures 218), neural network breeding modules 220(1)-220(n) (collectively referred to as neural network breeding modules 220), and data points 222(1)-222(n) (collectively referred to as data points 222). In an example, neural network updating code 107 (FIG. 1) includes neural network breeding module 206, neural network training modules 216, and neural network breeding modules 220. Edge servers 212 may be deployed in different physical locations such that the operations can be performed closer to end users of the system. Edge neural networks 214 receive inputs from end users (e.g., edge devices) and supply corresponding outputs to the end users.

[0054] Cloud data center 202 deploys copies of the centralized neural network 204 to the edge servers 212, which store their copies as edge neural networks 214. In an example, the centralized neural network 204 is initially built and trained at the cloud data center 202 using training data 208 during development, and, in some examples, may be updated using neural network breeding module 206 and redeployed to the edge servers 212. In other examples, cloud data center 202 is not involved in the updating of the edge neural networks 214, and the updating of the edge neural networks 214 is performed by the edge servers 212.

[0055] After deployment of the centralized neural network 204 to the edge servers 212 by the cloud data center 202, the edge neural networks 214 operate independently from each other and are trained and updated by their respective neural network training modules 216 based on the inputs seen at their respective edge servers 212. In some embodiments, each of the edge servers 212 stores data points 222 associated with the edge neural network 214 operating on that edge server 212. The data points 222 may include, for example, inputs and outputs of its associated edge neural network. In other embodiments, data points 222 are not stored and new input data may be used to test other neural networks from other edge servers to determine a fitness measure before performing neural network breeding.

[0056] In some examples, at periodic intervals, each of the edge servers 212 sends its edge neural network 214 to the other ones of the edge servers 212 for potential updating of the edge neural networks 214. The periodicity of the potential updates to the edge neural networks 214 may be dependent on use case (e.g., once per day, once per week, etc.). Each of the edge servers 212 then calculates fitness measures 218, which includes a fitness measure for each of the received edge neural networks 214, and is calculated based on the data points 222 stored at that edge server 212 or new inputs seen at that edge server 212. Each of the fitness measures 218 provides a measure of performance of the other edge neural networks 214 on different edge servers 212. In some examples, the fitness measures 218 may include memory allocation, input/output (I/O) through-put, score in a game, obtaining a system error, as well as others. The neural network breeding module 220 at each of the edge servers 212 then performs, based on the fitness measures 218, a breeding operation between the edge neural network 214 at that edge server 212 and all of the edge neural networks 214 received by that edge server 212 that have a fitness measure above a threshold to update the edge neural network 214 at that edge server 212. The system 200 shown in FIG. 2 is described in further detail below with reference to FIG. 3.

[0057] FIG. 3 sets forth a flowchart of an example method 300 for selective breeding for divergent neural networks in an edge computing environment according to aspects of the present disclosure. In a particular embodiment, the method 300 of FIG. 3 is performed by system 200 (FIG. 2) utilizing the neural network updating code 107, which includes neural network breeding module 206, neural network training modules 216, and neural network breeding modules 220. The method 300 of FIG. 3 starts at 302, and includes creating 304 a centralized neural network. In some examples, the centralized neural network in method 300 may be centralized neural

network 204 that is initially created within cloud data center 202 using training data 208. The method 300 further includes deploying 306 the centralized neural network to a plurality of edge servers. In some examples, cloud data center 202 deploys the most updated version of the centralized neural network 204 at 306 in method 300, and the edge servers in method 300 may be edge servers 212, which may each store a received copy of the centralized neural network as the initial edge neural network 214 before local training is performed. The method 300 further includes starting 308 a timer. In some examples, cloud data center 202 starts the timer in method 300, and the timer sets the duration before the edge neural networks 214 are shared between the edge servers 212 for potential updating. The timer limit or duration may be set by a system administrator based on how often they want the edge neural networks 214 updated (e.g., 1 day, 1 week, 1 month, etc.). In other embodiments, a timer is not used and a drift threshold may be set at 308 (e.g., by a system administrator) which sets the maximum amount of drift allowable on any of the edge neural networks 214 before the edge neural networks 214 are shared between the edge servers 212 for potential updating.

[0058] The method 300 further includes determining, at 310, whether input has been received at any of the edge servers. In some examples, each of the edge servers 212 operates independently and waits for input from edge devices (e.g., Internet of Things (IoT) devices, autonomous vehicles, mobile devices, etc.) to be processed by their edge neural networks 214. If it is determined at 310 that input has been received at the edge servers, the method 300 moves to 312, which includes generating 312 output using the edge neural network on the edge server where the input was received. In some examples, the output is provided to the edge device that provided the input to the edge server. The method 300 further includes using 314 the input as training data for the edge neural network on the edge server where the input was received, and the method 300 moves to 316. In some examples, the neural network training module 216 on the edge server 212 where the input was received uses that input as training data for the edge neural network 214 on that edge server 212 in method 300. The neural network training module 216 may use reinforcement learning, back-propagation, or other methods for the training. Because only the edge neural network that received the input is being trained and updated based on that input, the edge neural networks may begin to drift and differ from one another.

[0059] If it is determined at 310 that input has not been received at the edge servers, the method 300 moves to 316, which includes determining whether a specified time limit has been reached. In some examples, cloud data center 202 monitors the timer started at 308 and makes the determination at 316. In other examples, each of the edge servers 212 may start and monitor their own individual timers. If it is determined at 316 that the time limit has not been reached, the method 300 returns to 310 to again determine whether input has been received at the edge servers. Thus, the edge neural networks are allowed to continue to operate, train, and update independently.

[0060] If it is determined at 316 that the time limit has been reached, the method 300 moves to 318, which includes each edge server sending its edge neural network to the other edge servers. In some examples, cloud

data center 202 notifies the edge servers 212 at 318 when the time limit has been reached, and in response, each of the edge servers 212 shares their respective edge neural network 214 with the other edge servers 212. In some examples, instead of using a preset time limit, a preset amount of drift in one or more of the edge neural networks may be used.

[0061] The method 300 further includes performing 320 neural network breeding at one or more of the edge servers to update the edge neural networks at those edge servers, and then the method 300 returns to 308 to restart the timer. In some examples, for each of the edge servers 212 that performs neural network breeding, the neural network breeding module 220 at that edge server 212 performs the breeding in method 300 to update the edge neural network 214 at that edge server 212. In some examples, during any given iteration of updating of the edge neural networks 214, one or more of the edge neural networks 214 may not be updated because, for example, the fitness measures 218 do not exceed a threshold.

[0062] FIG. 4 sets forth a flowchart of an example method 400 for updating edge neural networks according to aspects of the present disclosure. In a particular embodiment, the method 400 of FIG. 4 is performed utilizing the neural network breeding modules 220, which are part of the neural network updating code 107, and which, in some examples, performs breeding of edge neural networks 214 to update one or more of the edge neural networks 214. The method 400 may be implemented as part of the breeding at 320 in method 300, and may be performed by each of the edge servers 212. The method 400 of FIG. 4 starts at 402, and includes receiving 404 edge neural networks from other edge servers. In some examples, for any given edge server 212, the neural network breeding module 220 for that edge server 212 receives edge neural networks 214 from all of the other edge servers 212.

[0063] The method 400 further includes calculating 405 a fitness measure for each of the received edge neural networks. In some examples, for any given edge server 212, the neural network breeding module 220 at that edge server 212 applies the data points 222 at that edge server 212 to each of the received edge neural networks 214 to calculate a fitness measure for each of the received edge neural networks 214, and stores the calculated fitness measures as fitness measures 218. In some embodiments where data points 222 are not stored, new input data may be used to test other neural networks from other edge servers to determine a fitness measure before performing neural network breeding. In these embodiments, the current edge neural network 214 at that edge server may still process the new inputs to generate outputs, but those inputs are also used on the received edge neural networks from other edge servers to generate fitness measures 218.

[0064] The method 400 further includes discarding 406 edge neural networks with an associated fitness measure that is below a threshold fitness measure. In examples where none of the other edge neural networks meet the threshold fitness measure, the method may jump to 428 to skip unnecessary processing steps and continue using the edge neural network at that edge server with no changes. In some examples, rather than using

a threshold at 406 in method 400, the method 400 may select the top “m” (e.g., top 3, top 5, etc.) fitness measures for breeding of their associated edge neural networks.

[0065] The method 400 further includes deconstructing 408 edge neural networks by layer. In some examples, the method 400 may alternatively decouple the entire set of layers into an array and then iterate through the list, rather than using nested loops as shown in FIG. 4. The method 400 further includes selecting 410 a first/next layer in the edge neural networks being analyzed. During the first pass through the selecting at 410, a first layer in the edge neural networks is selected, and during subsequent passes, other layers in the edge neural networks are selected. An iteration loop begins/continues at 410 in method 400 by selecting the first/next layer that will be analyzed. For example, if the edge neural networks 214 have 10 layers, then the method 400 will iterate through all 10 layers, one at a time. In some examples, all of the edge neural networks 214 have the same structure (i.e., same number of layers and same number of neurons within each layer). From layer to layer, the number of neurons may differ (e.g., all the edge neural networks 214 may have 5 neurons in layer 1, 7 neurons in layer 2, etc.). In some examples, one or more of the edge neural networks 214 have a different structure (i.e., different number of layers and/or different number of neurons within each layer). In these examples, the HyperNEAT calculation at 420 may include performing a weighted average of only similar neurons between the neural networks being breded and those new neurons may be added to the updated neural network at that edge server.

[0066] The method 400 further includes selecting 412 a first/next neuron in the selected layer of the edge neural networks. During the first pass through the selecting at 412, a first neuron in the selected layer is selected, and during subsequent passes, other neurons in the selected layer are selected. A nested loop begins/continues at 412 in method 400 where the first/next neuron within the currently selected layer is selected for analysis. For example, if layer 1 contains 5 neurons in all of the edge neural networks 214, the method 400 will iterate through all 5, one at a time.

[0067] The method 400 further includes selecting 414 a first/next edge neural network. Another nested loop begins/continues at 414 in method 400 where the first/next edge neural network is selected for analysis. For example, if there are 5 edge neural networks 214 being analyzed, the method 400 will iterate through all 5, one at a time.

[0068] The method 400 further includes extracting and storing 416 characteristics of the currently selected neuron. In some examples, the characteristics of the currently selected neuron within the currently selected layer within the currently selected edge neural network 214 may be extracted and stored in a data structure. The characteristics at 416 in method 400 may include weights, biases, activation functions, dependencies, etc.

[0069] At 418 in method 400, it is determined whether more edge neural networks are to be analyzed. If it is determined at 418 that there are more edge neural networks to be analyzed, the method 400 returns to 414 to select the next edge neural network to be analyzed. If it is determined at 418 that there are no more edge neural networks to be analyzed, the method 400 moves to 420.

[0070] The method 400 further includes performing 420 a HyperNEAT calculation to get parameters for the currently selected neuron in the currently selected layer within what will become an updated edge neural network. In some examples, the HyperNEAT calculation at 420 involves performing a weighted average (based on fitness measure) of the parameters extracted from each neuron.

[0071] At 422 in method 400, it is determined whether there are more neurons in the currently selected layer to be analyzed. If it is determined at 422 that there are more neurons in the currently selected layer to be analyzed, the method 400 returns to 412 to select the next neuron. If it is determined at 422 that there are not more neurons in the currently selected layer to be analyzed, the method 400 moves to 424.

[0072] At 424 in method 400, it is determined whether there are more layers in the edge neural networks to be analyzed. If it is determined at 424 that there are more layers to be analyzed, the method 400 returns to 410 to select the next layer. If it is determined at 424 that there are not more layers to be analyzed, the method 400 moves to 426. The method 400 further includes updating 426 the edge neural network. At 428, the method 400 may return to method 300 at 308, for example, to restart the timer.

[0073] FIG. 5 sets forth a flowchart of an example method 500 for selective breeding for divergent neural networks in an edge computing environment according to another aspect of the present disclosure. In a particular embodiment, the method 500 of FIG. 5 is performed by system 200 (FIG. 2) utilizing the neural network updating code 107, which includes neural network breeding module 206, neural network training modules 216, and neural network breeding modules 220. The method 500 includes deploying 502 a plurality of copies of a centralized neural network respectively to a corresponding plurality of edge servers, wherein each of the copies of the centralized neural network is independently operated and trained at one of the edge servers based on inputs received at that edge server and becomes an independently trained neural network. The method 500 further includes sending 504, by each of the edge servers at periodic intervals, a copy of the independently trained neural network at that edge server to other ones of the edge servers. The method 500 further includes updating 506, at each of one or more of the edge servers, the independently trained neural network at that edge server, including performing neural network breeding based on the independently trained neural network at that edge server and one or more copies of the independently trained neural networks sent to the edge server from other ones of the edge servers. In some examples, the method 500 may also include storing, at each of the edge servers, a set of data points associated with the copy of the centralized neural network at that edge server; and calculating, at each of the edge servers

based on the set of data points stored at the edge server, a fitness measure for each of the one or more copies of the independently trained neural networks sent to the edge server from the other ones of the edge servers.

[0074] Various aspects of the present disclosure are described by narrative text, flowcharts, block diagrams of computer systems and/or block diagrams of the machine logic included in computer program product (CPP) embodiments. With respect to any flowcharts, depending upon the technology involved, the operations can be performed in a different order than what is shown in a given flowchart. For example, again depending upon the technology involved, two operations shown in successive flowchart blocks may be performed in reverse order, as a single integrated step, concurrently, or in a manner at least partially overlapping in time.

[0075] A computer program product embodiment ("CPP embodiment" or "CPP") is a term used in the present disclosure to describe any set of one, or more, storage media (also called "mediums") collectively included in a set of one, or more, storage devices that collectively include machine readable code corresponding to instructions and/or data for performing computer operations specified in a given CPP claim. A "storage device" is any tangible device that can retain and store instructions for use by a computer processor. Without limitation, the computer readable storage medium may be an electronic storage medium, a magnetic storage medium, an optical storage medium, an electromagnetic storage medium, a semiconductor storage medium, a mechanical storage medium, or any suitable combination of the foregoing. Some known types of storage devices that include these mediums include: diskette, hard disk, random access memory (RAM), read-only memory (ROM), erasable programmable read-only memory (EPROM or Flash memory), static random access memory (SRAM), compact disc read-only memory (CD-ROM), digital versatile disk (DVD), memory stick, floppy disk, mechanically encoded device (such as punch cards or pits / lands formed in a major surface of a disc) or any suitable combination of the foregoing. A computer readable storage medium, as that term is used in the present disclosure, is not to be construed as storage in the form of transitory signals per se, such as radio waves or other freely propagating electromagnetic waves, electromagnetic waves propagating through a waveguide, light pulses passing through a fiber optic cable, electrical signals communicated through a wire, and/or other transmission media. As will be understood by those of skill in the art, data is typically moved at some occasional points in time during normal operations of a storage device, such as during access, de-fragmentation or garbage collection, but this does not render the storage device as transitory because the data is not transitory while it is stored.

[0076] The descriptions of the various embodiments of the present disclosure have been presented for purposes of illustration, but are not intended to be exhaustive or limited to the embodiments disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope of the described embodiments. The terminology used herein was chosen to best explain the principles of the embodiments, the practical application or technical improvement over technologies found in the marketplace, or to enable others of ordinary skill in the art to understand the embodiments disclosed herein.

CLAIMS

1. A method for selective breeding for divergent neural networks in an edge computing environment, comprising:
 - deploying a plurality of copies of a centralized neural network respectively to a corresponding plurality of edge servers, wherein each of the copies of the centralized neural network is independently operated and trained at one of the edge servers based on inputs received at that edge server and becomes an independently trained neural network;
 - sending, by each of the edge servers at periodic intervals, a copy of the independently trained neural network at that edge server to other ones of the edge servers; and
 - updating, at each of one or more of the edge servers, the independently trained neural network at that edge server, including performing neural network breeding based on the independently trained neural network at that edge server and one or more copies of the independently trained neural networks sent to the edge server from other ones of the edge servers.
2. The method of claim 1, and further comprising:
 - storing, at each of the edge servers, a set of data points associated with the independently trained neural network at that edge server.
3. The method of claim 2, and further comprising:
 - calculating, at each of the edge servers based on the set of data points stored at the edge server, a fitness measure for each of the one or more copies of the independently trained neural networks sent to the edge server from the other ones of the edge servers.
4. The method of claim 3, wherein the neural network breeding includes discarding, at each of the edge servers, any of the one or more copies of the independently trained neural networks sent to the edge server that have a corresponding fitness measure below a threshold.
5. The method of claim 3 or claim 4, wherein the neural network breeding includes selecting, at each of the edge servers based on the fitness measures, top performing ones of the one or more copies of the independently trained neural networks sent to the edge server.
6. The method of any of claims 3 to 5, wherein the neural network breeding includes performing, at each of the edge servers, a hyperNEAT calculation that includes determining a weighted average, based on the fitness measures, of parameters extracted from neurons in the one or more copies of the independently trained neural networks sent to the edge server.

7. The method of any preceding claim, wherein the periodic interval is defined by a preset period of time.
8. The method of any of claims 1 to 6, wherein the periodic interval is defined by a preset amount of drift occurring in one or more of the copies of the independently trained neural networks.
9. An apparatus for selective breeding for divergent neural networks in an edge computing environment, comprising:
 - a processing device; and
 - memory operatively coupled to the processing device, wherein the memory stores computer program instructions that, when executed, cause the processing device to:
 - deploy a plurality of copies of a centralized neural network respectively to a corresponding plurality of edge servers, wherein each of the copies of the centralized neural network is independently operated and trained at one of the edge servers based on inputs received at that edge server and becomes an independently trained neural network;
 - send, by each of the edge servers at periodic intervals, a copy of the independently trained neural network at that edge server to other ones of the edge servers; and
 - update, at each of one or more of the edge servers, the independently trained neural network at that edge server, including performing neural network breeding based on the independently trained neural network at that edge server and one or more copies of the independently trained neural networks sent to the edge server from other ones of the edge servers.
10. The apparatus of claim 9, wherein the memory stores computer program instructions that, when executed, cause the processing device to:
 - store, at each of the edge servers, a set of data points associated with the independently trained neural network at that edge server.
11. The apparatus of claim 10, wherein the memory stores computer program instructions that, when executed, cause the processing device to:
 - calculate, at each of the edge servers based on the set of data points stored at the edge server, a fitness measure for each of the one or more copies of the independently trained neural networks sent to the edge server from the other ones of the edge servers.
12. The apparatus of claim 11, wherein the neural network breeding includes discarding, at each of the edge servers, any of the one or more copies of the independently trained neural networks sent to the edge server that have a corresponding fitness measure below a threshold.

13. The apparatus of claim 11 or claim 12, wherein the neural network breeding includes selecting, at each of the edge servers based on the fitness measures, top performing ones of the one or more copies of the independently trained neural networks sent to the edge server.
14. The apparatus of claim 11 or claim 12, wherein the neural network breeding includes performing, at each of the edge servers, a hyperNEAT calculation that includes determining a weighted average, based on the fitness measures, of parameters extracted from neurons in the one or more copies of the independently trained neural networks sent to the edge server.
15. The apparatus of any of claims 9 to 14, wherein the periodic interval is defined by a preset period of time.
16. The apparatus of any of claims 9 to 14, wherein the periodic interval is defined by a preset amount of drift occurring in one or more of the copies of the independently trained neural networks.
17. A computer program product for selective breeding for divergent neural networks in an edge computing environment, comprising a computer readable storage medium, wherein the computer readable storage medium comprises computer program instructions that, when executed:
- deploy a plurality of copies of a centralized neural network respectively to a corresponding plurality of edge servers, wherein each of the copies of the centralized neural network is independently operated and trained at one of the edge servers based on inputs received at that edge server and becomes an independently trained neural network;
 - send, by each of the edge servers at periodic intervals, a copy of the independently trained neural network at that edge server to other ones of the edge servers; and
 - update, at each of one or more of the edge servers, the independently trained neural network at that edge server, including performing neural network breeding based on the independently trained neural network at that edge server and one or more copies of the independently trained neural networks sent to the edge server from other ones of the edge servers.
18. The computer program product of claim 17, wherein the computer readable storage medium further comprises computer program instructions that, when executed:
- store, at each of the edge servers, a set of data points associated with the independently trained neural network at that edge server.
19. The computer program product of claim 18, wherein the computer readable storage medium further comprises computer program instructions that, when executed:

calculate, at each of the edge servers based on the set of data points stored at the edge server, a fitness measure for each of the one or more copies of the independently trained neural networks sent to the edge server from the other ones of the edge servers.

20. The computer program product of claim 19, wherein the neural network breeding includes performing, at each of the edge servers, a hyperNEAT calculation that includes determining a weighted average, based on the fitness measures, of parameters extracted from neurons in the one or more copies of the independently trained neural networks sent to the edge server.

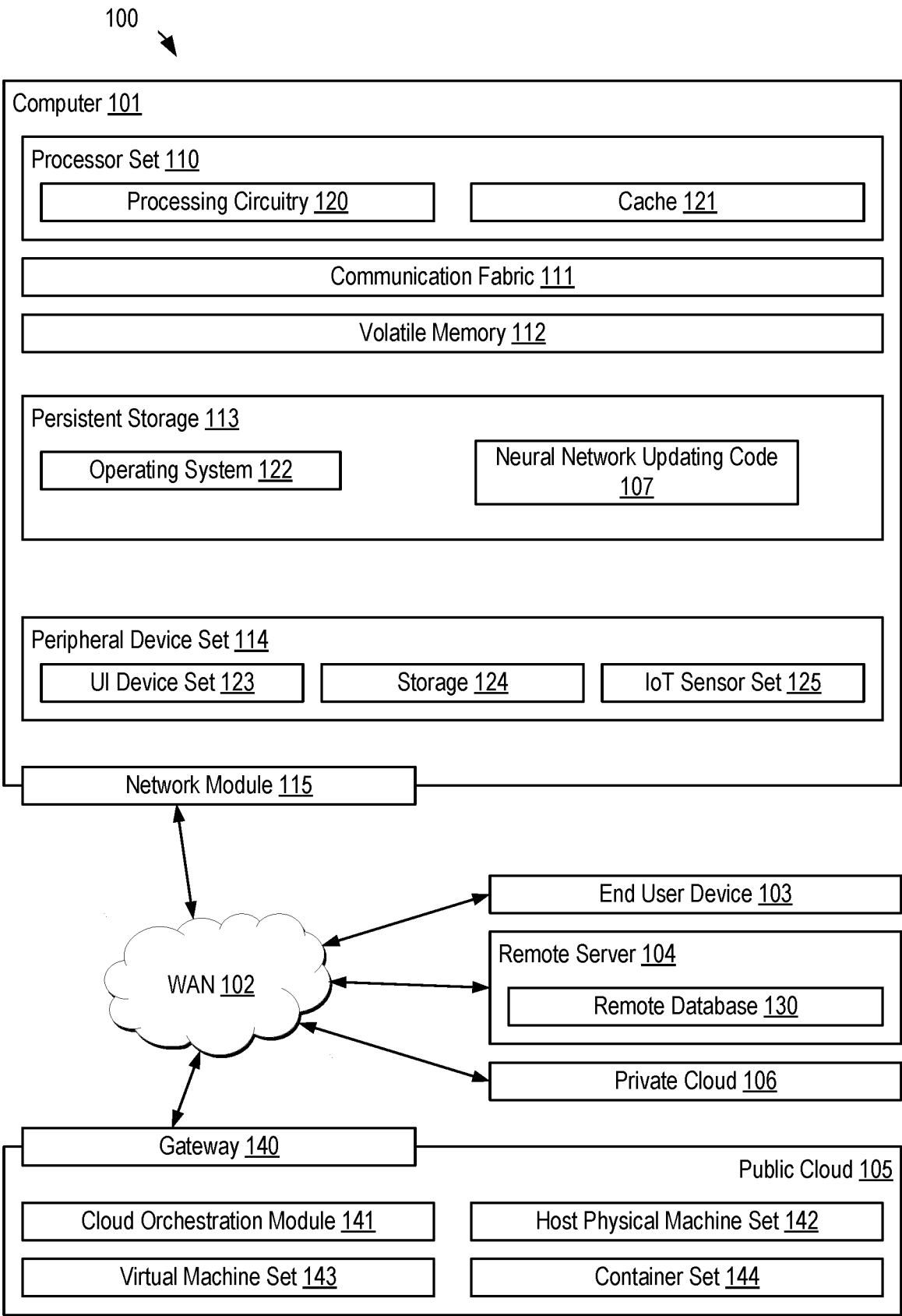


FIG. 1

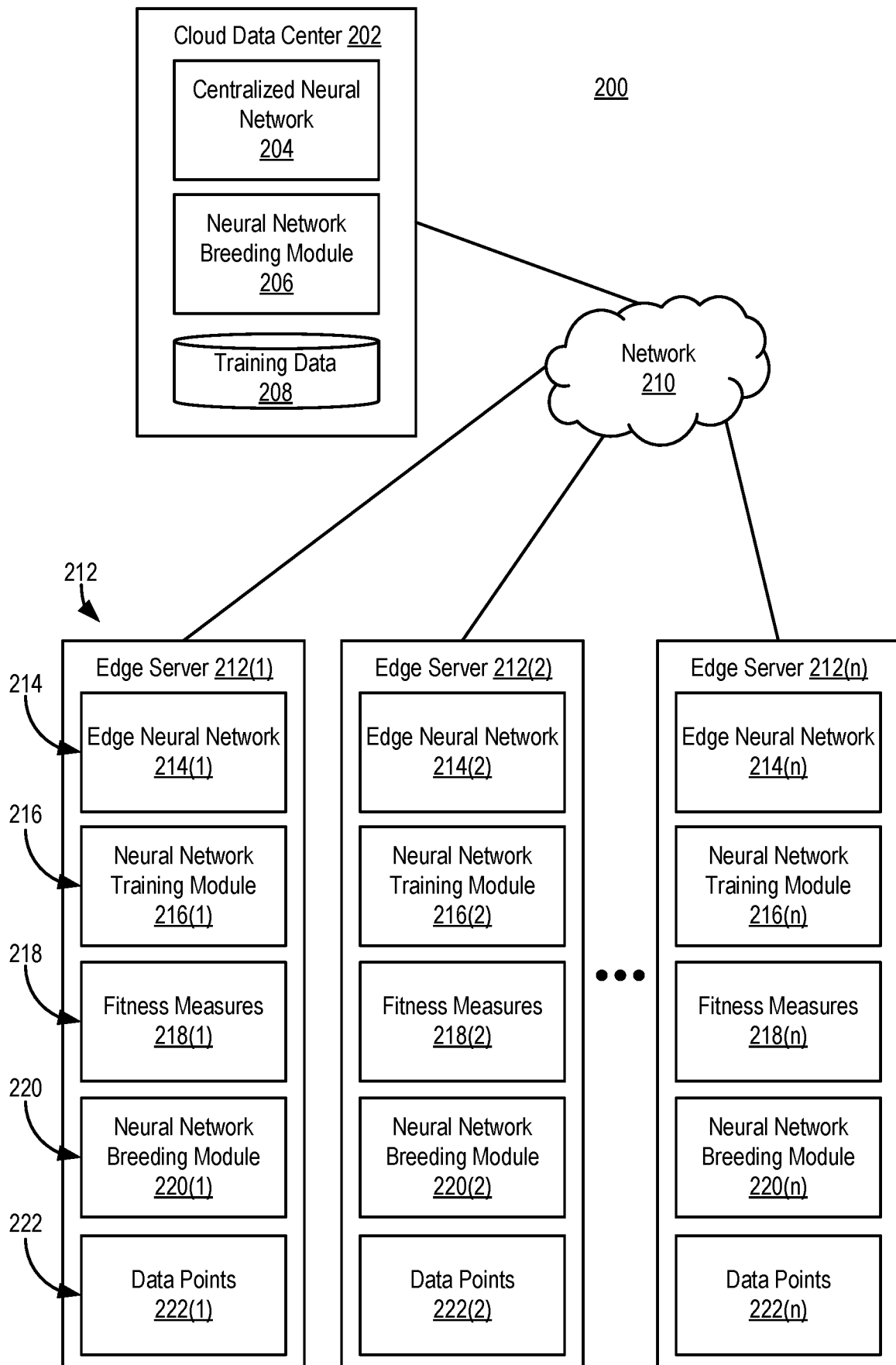


FIG. 2

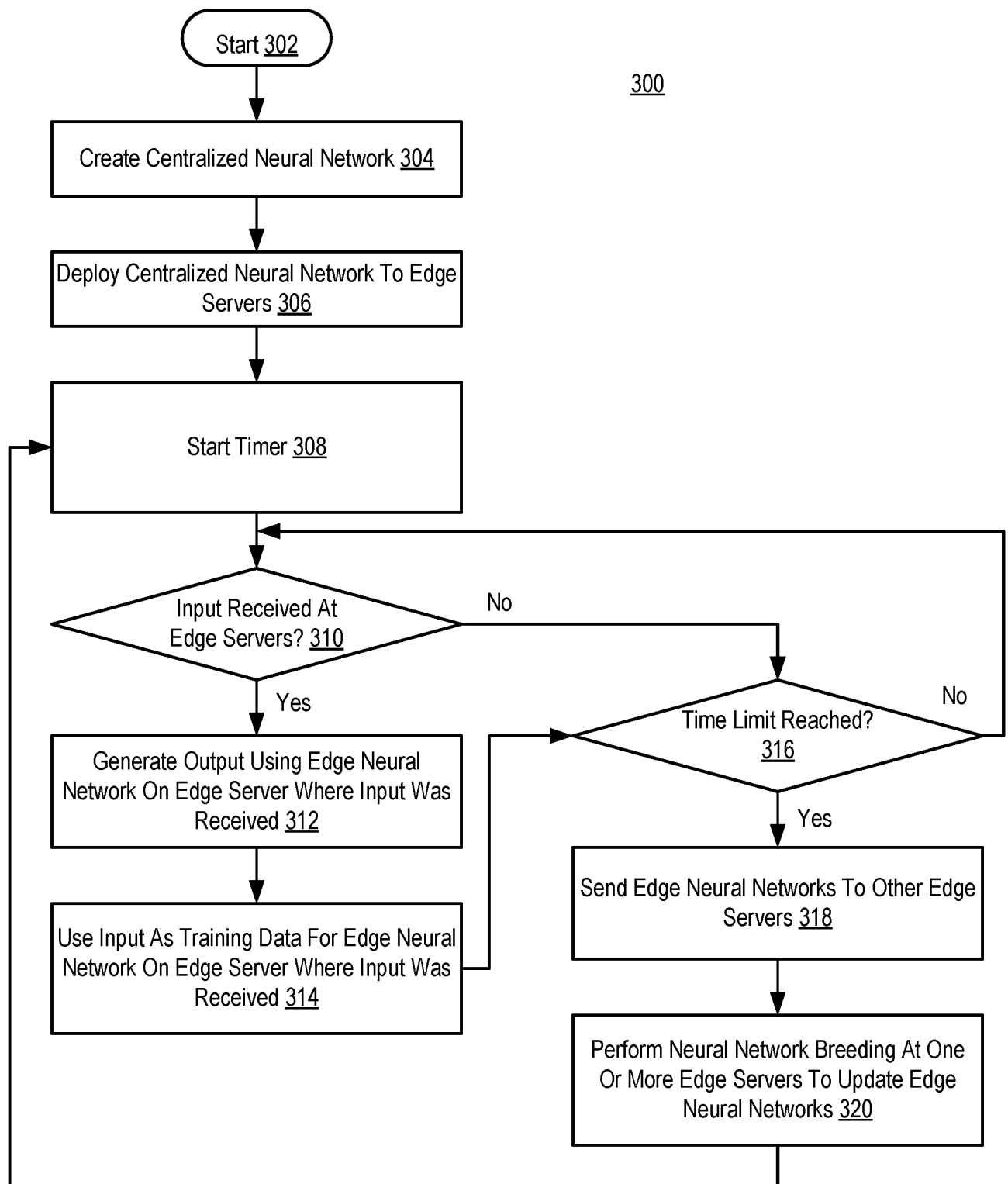


FIG. 3

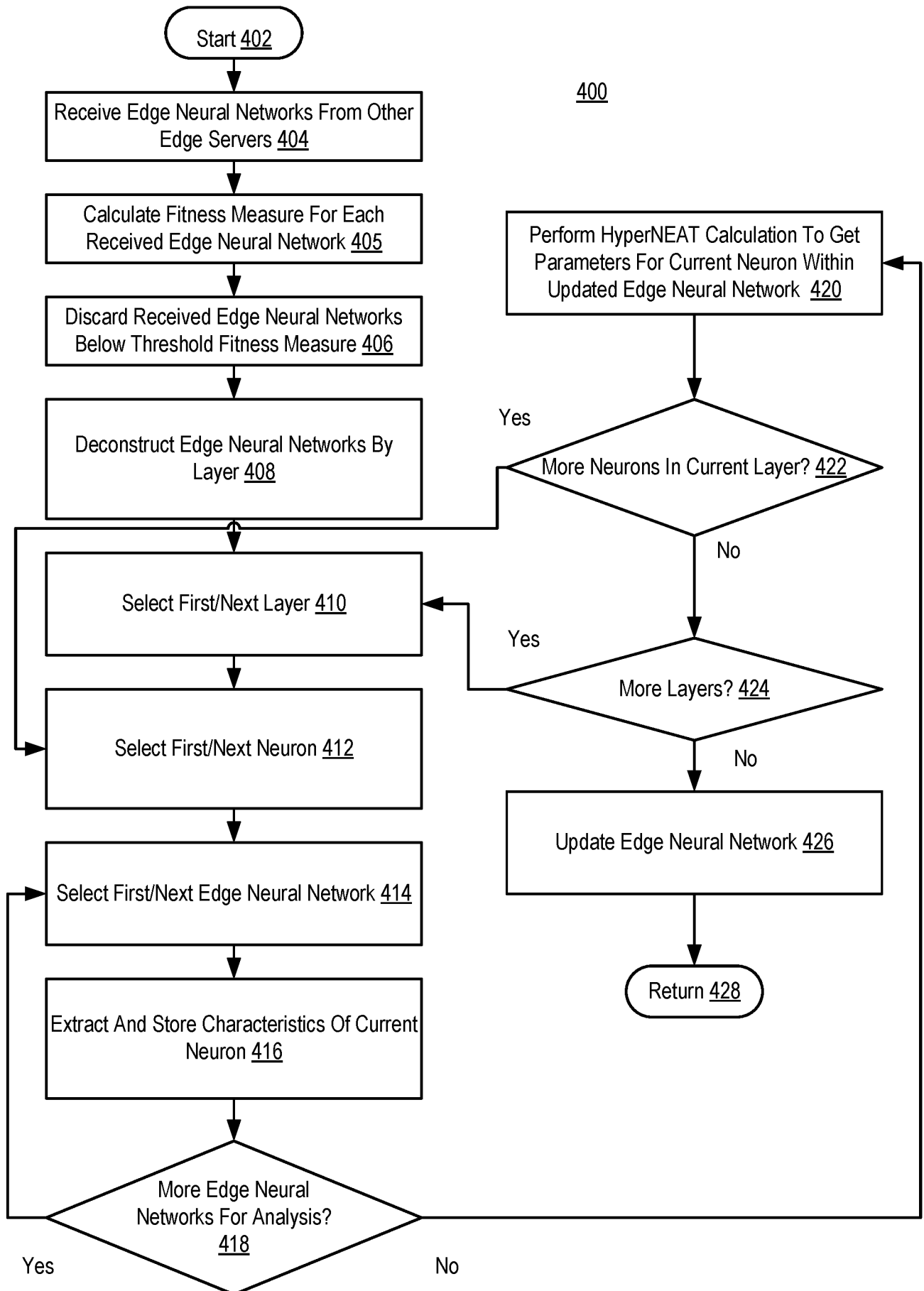


FIG. 4

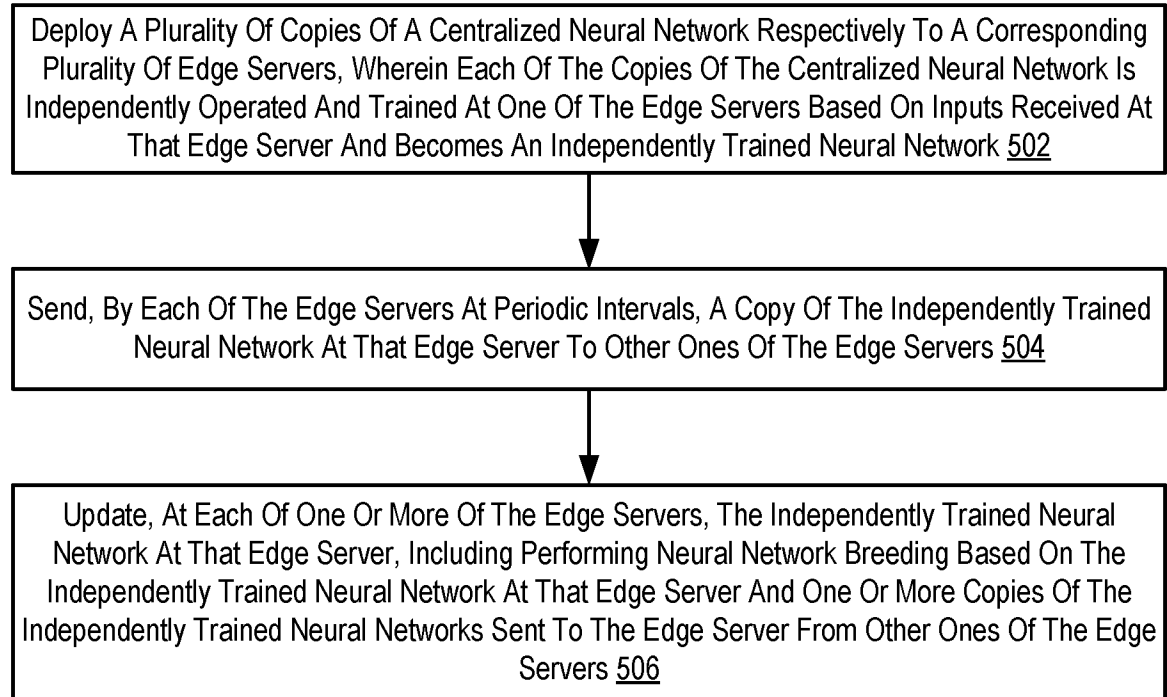
500

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2024/074118

A. CLASSIFICATION OF SUBJECT MATTER INV. G06N3/098 G06N3/086 ADD.		
According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED		
Minimum documentation searched (classification system followed by classification symbols) G06N		
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched		
Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) EPO-Internal		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	MERTENS J S ET AL: "MGM-4-FL: Combining federated learning and model gossiping in WSNs", COMPUTER NETWORKS, ELSEVIER, AMSTERDAM, NL, vol. 214, 14 July 2022 (2022-07-14), XP087152775, ISSN: 1389-1286, DOI: 10.1016/J.COMNET.2022.109144 [retrieved on 2022-07-14] sections 1, 4.1, 4.2 and 5.1 ----- - / - -	1 - 20
☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.		
* Special categories of cited documents : "A" document defining the general state of the art which is not considered to be of particular relevance "E" earlier application or patent but published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance;; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance;; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "&" document member of the same patent family		
Date of the actual completion of the international search 1 December 2024		Date of mailing of the international search report 17/12/2024
Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016		Authorized officer Millet, Guillaume

INTERNATIONAL SEARCH REPORT

International application No

PCT/EP2024/074118

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	SAVAZZI STEFANO ET AL: "Federated Learning With Cooperating Devices: A Consensus Approach for Massive IoT Networks", IEEE INTERNET OF THINGS JOURNAL, IEEE, USA, vol. 7, no. 5, 6 January 2020 (2020-01-06) , pages 4641-4654, XP011788101, DOI: 10.1109/JIOT.2020.2964162 [retrieved on 2020-05-12] section III -----	1 - 20
A	KENNETH O STANLEY ET AL: "A Hypercube-Based Indirect Encoding for Evolving Large-Scale Neural Networks", ARTIFICIAL LIFE JOURNAL, vol. 15, no. 2, 5 March 2009 (2009-03-05), pages 185-212, XP055264947, DOI: 10.1162/artl.2009.15.2.15202 the whole document -----	6, 14, 20