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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 February 2025 (20.02.2025)

WIPO | PCT

(10) International Publication Number
WO 2025/035403 A1

(51) International Patent Classification:
G06F 7/483 (2006.01)

(21) International Application Number:
PCT/CN2023/113202

(22) International Filing Date:
15 August 2023 (15.08.2023)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95054 (US).

(72) Inventors; and

(71) Applicants *(for BZ only)*: **GAO, Qun** [CN/US]; 3550 Old Airport Rd NW 1117, Albuquerque, New Mexico 87114 (US). **HE, Xin** [CN/CN]; No. 880 Zixing Road Zizhu Science Park, Minhang District, Shanghai 200241 (CN). **SHEN, Haihao** [CN/CN]; No. 880 Zixing Road Zizhu Science Park, Minhang District, Shanghai 200241 (CN). **MELLEMPUDI, Naveen** [IN/IN]; A5, Skanda Avani C99, Chikkabellandur, Carmelaram, Bangalore KA, 560035 (IN). **LIANG, Lv** [CN/CN]; Room 402, No. 94, Lane 1111, Donglan Road, Minhang District, Shanghai 201101 (CN). **CHANG, Hanwen** [CN/CN]; Room 202, No. 54, Lane 51, Guilin East Street, Changning District, Shanghai 200336 (CN).

(74) Agent: **BEIJING EAST IP LTD.** et al.; Suite 1601, Tower E2, The Towers, Oriental Plaza, No.1 East Chang An Ave., Dongcheng District, Beijing 100738 (CN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, 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))*

(54) **Title:** FLOATING POINT ACCURACY CONTROL VIA DYNAMIC EXPONENT AND MANTISSA BIT CONFIGURATIONS

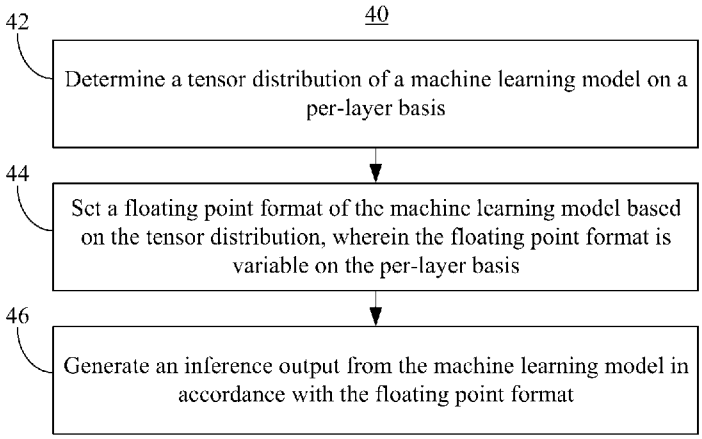


FIG. 2

(57) **Abstract:** Systems, apparatuses and methods may provide for technology that determines a tensor distribution of a machine learning model on a per-layer basis, sets a floating point format of the machine learning model based on the tensor distribution, wherein the floating point format is variable on the per-layer basis, and generates an inference output from the machine learning model in accordance with the floating point format.

WO 2025/035403 A1

FLOATING POINT ACCURACY CONTROL VIA DYNAMIC EXPONENT AND MANTISSA BIT CONFIGURATIONS

TECHNICAL FIELD

Embodiments generally relate to the use of floating point data formats in artificial intelligence (AI) compute operations. More particularly, embodiments relate to floating point accuracy control via dynamic exponent and mantissa bit configurations.

BACKGROUND

The complexity of artificial intelligence (AI) compute operations may be dependent on the format used for the underlying data. For example, the FP8 (8-bit floating point) data format may reduce compute complexity of neural network training computations relative to the BF16 (16-bit brain floating point) data format. Additionally, the non-linear sampling of real numbers in the FP8 data format may be advantageous over the INT8 (8-bit integer) data format for inference computations. Conventional solutions typically use a fixed data format for all layers in the neural network, which may result in sub-optimal accuracy.

BRIEF DESCRIPTION OF THE DRAWINGS

The various advantages of the embodiments will become apparent to one skilled in the art by reading the following specification and appended claims, and by referencing the following drawings, in which:

FIG. 1 is a comparative set of charts of an example of conventional relative layer-by-layer stacked mean squared error (MSE) of FP8 formats and enhanced relative layer-by-layer stacked MSE of FP8 formats according to an embodiment;

FIG. 2 is a flowchart of an example of a method of conducting inference computations according to an embodiment;

FIG. 3 is a flowchart of an example of a method of determining a tensor distribution of a machine learning model according to an embodiment;

FIG. 4 is an illustration of an example of an identification of an MSE of a tensor distribution according to an embodiment;

FIG. 5 is an illustration of an example of an outlier signature of a tensor distribution according to an embodiment;

FIG. 6 is a comparative set of charts of an example of relative layer-by-layer MSE of an FP8 format and the INT8 format for computer vision (CV) and natural language processing (NLP) according to an embodiment;

FIG. 7 is a scatter chart of an example of accuracy loss for a computer vision model according to an embodiment;

FIG. 8 is a scatter chart of an example of accuracy loss for a natural language processing model according to an embodiment;

FIG. 9 is a comparative set of plots of examples of tensor fingerprints for a natural language processing model and a computer vision model according to an embodiment;

FIG. 10 is a block diagram of an example of a performance-enhanced computing system according to an embodiment;

FIG. 11 is an illustration of an example of a semiconductor package apparatus according to an embodiment;

FIG. 12 is a block diagram of an example of a processor according to an embodiment; and

FIG. 13 is a block diagram of an example of a multi-processor based computing system according to an embodiment.

DETAILED DESCRIPTION

As already noted, using a fixed data format for all layers in a neural network may result in sub-optimal accuracy. The technology described herein sets the floating point format of a machine learning model based on the tensor distribution, which is determined on a per-layer (e.g., per-operator, per-tensor) basis. Varying the floating point format on the per-layer basis improves the model inference accuracy at the operator ("op") level with dynamic exponent and mantissa bit configurations, particularly given that machine learning (ML) model sizes are growing rapidly and there is a lack of a one-size-fits-all data format solution.

Turning now to FIG. 1, the FP8 (8-bit floating point) data format includes multiple data types based on the exponent and mantissa bit configuration. For example, the FP8_E3M4 datatype includes one sign bit, three exponent bits and four bits of mantissa, the FP8_E4M3 datatype includes one sign bit, four exponent bits and three bits of mantissa, and so forth. The technology described herein provides for a hybrid solution ("FP8_Hybrid") that selects the floating point format of the machine learning

model based on the tensor (e.g., weight tensors, input activation tensors, output activation tensors) distribution.

More particularly, a first weight chart 20 (20a, 20b) shows the relative mean squared error (MSE) percentage of the FP8_E3M4 datatype for a fixed floating point format solution and the FP8_E4M3 datatype for a fixed floating point format solution. Thus, the first weight chart 20 is a stacked bar plot to show the relative proportion of two MSEs: E3M4 versus E4M3 for each layer (e.g., x-axis) in which a bar is divided into two segments, E3M34 and E4M3, respectively. The two segments therefore add up to 100% totally. In the illustrated example, an E3M4 portion 20a yields significantly less error than an E4M3 portion 20b for weight tensors. By contrast, a second weight chart 22 (22a, 22b, e.g., stacked bar plot) shows the relative MSE percentage of the enhanced FP8_Hybrid solution (e.g., variable floating point format of the machine learning model based on the tensor distribution) and the FP8_E3M4 datatype for a fixed floating point format solution. In the illustrated example, a hybrid portion 22a yields similar error to an E3M4 portion 22b for weight tensors.

A first input activation chart 24 (24a, 24b, e.g., stacked bar plot) shows the relative MSE percentage of the FP8_E3M4 datatype for a fixed floating point format solution and the FP8_E4M3 datatype for a fixed floating point format solution. In the illustrated example, an E3M4 portion 24a yields significantly less error than an E4M3 portion 24b for most layers of the neural network. By contrast, a second input activation chart 26 (26a, 26b, e.g., stacked bar plot) shows the relative MSE percentage of the enhanced FP8_Hybrid solution (e.g., variable floating point format of the machine learning model based on the tensor distribution) and the FP8_E3M4 datatype for a fixed floating point format solution. In the illustrated example, a hybrid portion 26a yields less error than an E3M4 portion 26b for input activation tensors.

A first output activation chart 28 (28a, 28b, e.g., stacked bar plot) shows the relative MSE percentage of the FP8_E3M4 datatype for a fixed floating point format solution and the FP8_E4M3 datatype for a fixed floating point format solution. In the illustrated example, an E3M4 portion 28a yields significantly less error than an E4M3 portion 28b for most layers of the neural network. By contrast, a second output activation chart 30 (30a, 30b, e.g., stacked bar plot) shows the relative MSE percentage of the enhanced FP8_Hybrid solution (e.g., variable floating point format of the machine learning model based on the tensor distribution) and the FP8_E3M4 datatype

for a fixed floating point format solution. In the illustrated example, a hybrid portion 30a yields less error than an E3M4 portion 30b for output activation tensors.

FIG. 2 shows a method 40 of conducting inference computations. The method 40 may be implemented in one or more modules as a set of logic instructions stored in a machine- or computer-readable storage medium such as random access memory (RAM), read only memory (ROM), programmable ROM (PROM), firmware, flash memory, etc., in hardware, or any combination thereof. For example, hardware implementations may include configurable logic, fixed-functionality logic, or any combination thereof. Examples of configurable logic (e.g., configurable hardware) include suitably configured programmable logic arrays (PLAs), field programmable gate arrays (FPGAs), complex programmable logic devices (CPLDs), and general purpose microprocessors. Examples of fixed-functionality logic (e.g., fixed-functionality hardware) include suitably configured application specific integrated circuits (ASICs), combinational logic circuits, and sequential logic circuits. The configurable or fixed-functionality logic can be implemented with complementary metal oxide semiconductor (CMOS) logic circuits, transistor-transistor logic (TTL) logic circuits, or other circuits.

For example, computer program code to carry out operations shown in the method 40 can be written in any combination of one or more programming languages, including an object oriented programming language such as JAVA, SMALLTALK, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages. Additionally, logic instructions might include assembler instructions, instruction set architecture (ISA) instructions, machine instructions, machine dependent instructions, microcode, state-setting data, configuration data for integrated circuitry, state information that personalizes electronic circuitry and/or other structural components that are native to hardware (e.g., host processor, central processing unit/CPU, microcontroller, etc.).

Illustrated processing block 42 provides for determining a tensor distribution of a machine learning model on a per-layer basis (e.g., for each layer, operator and/or tensor of a neural network). The machine learning model may include a computer vision (CV) model, a natural language processing (NLP) model, etc., or any combination thereof. In this regard, CV models are generally mantissa-bounded (e.g., requiring more mantissa bits for acceptable accuracy) and NLP models are typically range-bounded (e.g., involving activation tensors with extreme input-channel-wise

outliers). As will be discussed in greater detail below, block 42 may include identifying an MSE of the tensor distribution.

Block 44 sets a floating point format (e.g., FP8_E3M4, FP8_E4M3, FP8_E5M2, etc.) of the machine learning model based on the tensor distribution, wherein the floating point format is variable on the per-layer basis. In an embodiment, block 44 includes recording the floating point format in the machine learning model as an exponent and mantissa bit configuration (e.g., attribution). Block 46 generates an inference output from the machine learning model in accordance with the floating point format. The method 40 therefore enhances performance at least to the extent that the variable floating point format increases the accuracy and/or efficiency of the inference output. The variable floating point format also enables hardware potential to be fully utilized (e.g., when the hardware supports mixed exponent/mantissa encodings).

FIG. 3 shows a method 50 of determining a tensor distribution of a machine learning model on a per-layer basis. The method 50 may generally be incorporated into block 42 (FIG. 2), already discussed. More particularly, the method 50 may be implemented in one or more modules as a set of logic instructions stored in a machine- or computer-readable storage medium such as RAM, ROM, PROM, firmware, flash memory, etc., in hardware, or any combination thereof. For example, hardware implementations may include configurable logic, fixed-functionality logic, or any combination thereof.

Illustrated processing block 52 identifies an MSE of the tensor distribution. With continuing reference to FIGs. 3 and 4, block 52 may include a plurality of "fake" FP8 determinations 54 that evaluate the per-tensor MSE for tensors under different encodings. In one example, the fake FP8 determinations 54 will indicate whether the tensor is range-bounded or mantissa-bounded, implicitly. Since the use of MSE can only provide implicit distribution information on the per-tensor scale, another technique can be used to "zoom in" and obtain the per-input-channel outliers to better select exponent/mantissa bit configurations when appropriate.

More particularly, block 56 determines whether the difference between the lowest MSE and the second lowest MSE is greater than a threshold (e.g., X%). If so, block 58 records the floating point format in the machine learning model as the exponent and mantissa bit configuration corresponding to the lowest MSE. Otherwise, block 60 identifies an outlier signature (e.g., fingerprint) of the tensor distribution on a per-input channel basis. With continuing reference to FIGs. 3 and 5, block 60 may

include a model-specific recommendation system 63 that defines channel outliers as a function of the 3rd quartile (Q3) of the tensor distribution, the interquartile range (IQR) and a programmable constant (N, e.g., ranging from four to infinity). Block 62 records the floating point format in the machine learning model as an exponent and mantissa bit configuration in accordance with the outlier signature.

FIG. 6 shows a first weight chart 70 (70a, 70b, e.g., stacked bar plot) demonstrating the relative MSE percentage of the FP8_E4M3 datatype for a fixed floating point format solution for a fixed floating point format solution and the INT8 datatype for a fixed floating point format solution in a CV model (e.g., ResNet50 dataset). In the illustrated example, an E4M3 portion 70a yields significantly greater error than an INT8 portion 70b for weight tensors. Additionally, a second weight chart 72 (72a, 72b, e.g., stacked bar plot) shows the relative MSE percentage of the FP8_E4M3 datatype for a fixed floating point format solution and the INT8 datatype for a fixed floating point format solution in an NLP model (e.g., Bert_Cola dataset). In the illustrated example, an E4M3 portion 72a yields significantly greater error than an INT8 portion 72b for weight tensors.

Additionally, a first activation chart 74 (74a, 74b, e.g., stacked bar plot) demonstrating the relative MSE percentage of the FP8_E4M3 datatype for a fixed floating point format solution and the INT8 datatype for a fixed floating point format solution in a CV model (e.g., ResNet50 dataset). In the illustrated example, an E4M3 portion 74a yields slightly greater error than an INT8 portion 74b for weight tensors. Additionally, a second activation chart 76 (76a, 76b, e.g., stacked bar plot) shows the relative MSE percentage of the FP8_E4M3 datatype for a fixed floating point format solution and the INT8 datatype for a fixed floating point format solution in an NLP model (e.g., Bert_Cola dataset). In the illustrated example, an E4M3 portion 76a yields significantly less error than an INT8 portion 76b for activation tensors.

The second activation chart 76 therefore demonstrates that Bert_Cola (NLP) activation tensors are clearly range-bounded as INT8 shows significantly worse MSE than FP8. By contrast, the first activation chart 74 demonstrates that Resnet50 (CV) activation tensors are not range-bounded. Therefore, for range-bounded NLP tensors, FP_E4M3 with four exponent digits may be a better choice to cover the range and obtain better accuracy.

Turning now to FIGs. 7 and 8, a scatter chart 80 for a CV model and a scatter chart 90 for an NLP model are shown, respectively. The charts 80, 90 demonstrate that

there is a clear trend of CV models being mantissa-bounded instead of range-bounded (e.g., benefiting from more mantissa digits). Therefore, E3M4 is more advantageous than E5M2/E4M3 in CV models. On the contrary, NLP models are range-bounded and not very sensitive to mantissa. Accordingly, FP8 with greater better range/mantissa balance may be more advantageous in NLP models.

FIG. 9 shows a first plot 100 of an activation tensor distribution for an NLP model (e.g., NLP activation signature/fingerprint), a second plot 102 of an activation tensor distribution for a CV model (e.g., CV activation signature/fingerprint), a third plot 104 of a weight tensor distribution for an NLP model (e.g., NLP weight signature/fingerprint), and a fourth plot 106 of a weight tensor distribution for a CV model (e.g., CV weight signature/fingerprint). In the illustrated example, a portion 108 of the first plot 100 shows extreme per-input-channel outliers that benefit from better exponent/mantissa balance. By contrast, the remaining plots 102, 104, 106 demonstrate that the weight tensors of the NLP model and both activation/weight tensors of the CV models are mantissa-bounded that benefit from high mantissa digits. Based on the number/range of the per-input-channel outliers, optimal exponent/mantissa may be elected using the signature technology described herein.

Recent trends indicate that developing larger and more complex deep learning models can achieve better performance. Additionally, lager models may be more successful in achieving state-of-the-art results in many domains, such as, for example, large language models (LLM, e.g., CHATGPT). Although "brute force" evaluations may work well for small models, it may be unrealistic to use such approaches to cover all large models, such as the popular generative pre-trained transformer (GPT) models. Therefore, more efficient solutions such as the technology described herein will become more valuable.

With regard to overhead, the technology described herein can decide the data format ahead of real inference by calibration. Therefore, minimal resources may be used to calculate and record the decision of the data format for each layer. The information gathered from previous operations can be attached to each layer and each tensor as an attribution. Therefore, compared with INT8 quantization, there is little to no additional data processing for FP8 format in inference computations.

Turning now to FIG. 10, a performance-enhanced computing system 280 is shown. The system 280 may generally be part of an electronic device/platform having computing functionality (e.g., personal digital assistant/PDA, notebook computer,

tablet computer, convertible tablet, server), communications functionality (e.g., smart phone), imaging functionality (e.g., camera, camcorder), media playing functionality (e.g., smart television/TV), wearable functionality (e.g., watch, eyewear, headwear, footwear, jewelry), vehicular functionality (e.g., car, truck, motorcycle), robotic functionality (e.g., autonomous robot), Internet of Things (IoT) functionality, drone functionality, etc., or any combination thereof.

In the illustrated example, the system 280 includes a host processor 282 (e.g., CPU) having an integrated memory controller (IMC) 284 that is coupled to a system memory 286 (e.g., dual inline memory module/DIMM). In an embodiment, an IO module 288 is coupled to the host processor 282. The illustrated IO module 288 communicates with, for example, a display 290 (e.g., touch screen, liquid crystal display/LCD, light emitting diode/LED display), and a network controller 292 (e.g., wired and/or wireless). The host processor 282 may be combined with the IO module 288, a graphics processor 294, and an AI accelerator 296 into a system on chip (SoC) 298.

In an embodiment, the AI accelerator 296, the host processor 282 and/or the SoC 298 executes a set of program instructions 300 retrieved from mass storage 302 and/or the system memory 286 to perform one or more aspects of the method 40 (FIG. 2) and/or the method 50 (FIG. 3), already discussed. Thus, execution of the instructions 300 causes the AI accelerator 296, the host processor 282 and/or the SoC 298 to determine a tensor distribution of a machine learning model (e.g., CV model, NLP model) on a per-layer basis and set a floating point format of the machine learning model based on the tensor distribution, wherein the floating point format is variable on the per-layer basis. Execution of the instructions 300 may also cause the AI accelerator 296, the host processor 282 and/or the SoC 298 to generate, via the display 290 and/or the network controller 292, an inference output from the machine learning model in accordance with the floating point format. The computing system 280 is therefore considered performance-enhanced at least to the extent that the variable floating point format increases the accuracy and/or efficiency of the inference output. The variable floating point format enables hardware potential to be fully utilized (e.g., when the hardware supports mixed exponent/mantissa encodings).

FIG. 11 shows a semiconductor apparatus 350 (e.g., chip, die, package). The illustrated apparatus 350 includes one or more substrates 352 (e.g., silicon, sapphire, gallium arsenide) and logic 354 (e.g., transistor array and other integrated circuit/IC

components) coupled to the substrate(s) 352. In an embodiment, the logic 354 implements one or more aspects of the method 40 (FIG. 2) and/or the method 50 (FIG. 3), already discussed. The semiconductor apparatus 350 may also be incorporated into the AI accelerator 296 (FIG. 10).

The logic 354 may be implemented at least partly in configurable or fixed-functionality hardware. In one example, the logic 354 includes transistor channel regions that are positioned (e.g., embedded) within the substrate(s) 352. Thus, the interface between the logic 354 and the substrate(s) 352 may not be an abrupt junction. The logic 354 may also be considered to include an epitaxial layer that is grown on an initial wafer of the substrate(s) 352.

FIG. 12 illustrates a processor core 400 according to one embodiment. The processor core 400 may be the core for any type of processor, such as a micro-processor, an embedded processor, a digital signal processor (DSP), a network processor, or other device to execute code. Although only one processor core 400 is illustrated in FIG. 12, a processing element may alternatively include more than one of the processor core 400 illustrated in FIG. 12. The processor core 400 may be a single-threaded core or, for at least one embodiment, the processor core 400 may be multithreaded in that it may include more than one hardware thread context (or "logical processor") per core.

FIG. 12 also illustrates a memory 470 coupled to the processor core 400. The memory 470 may be any of a wide variety of memories (including various layers of memory hierarchy) as are known or otherwise available to those of skill in the art. The memory 470 may include one or more code 413 instruction(s) to be executed by the processor core 400, wherein the code 413 may implement the method 40 (FIG. 2) and/or the method 50 (FIG. 3), already discussed. The processor core 400 follows a program sequence of instructions indicated by the code 413. Each instruction may enter a front end portion 410 and be processed by one or more decoders 420. The decoder 420 may generate as its output a micro operation such as a fixed width micro operation in a predefined format, or may generate other instructions, microinstructions, or control signals which reflect the original code instruction. The illustrated front end portion 410 also includes register renaming logic 425 and scheduling logic 430, which generally allocate resources and queue the operation corresponding to the convert instruction for execution.

The processor core 400 is shown including execution logic 450 having a set of execution units 455-1 through 455-N. Some embodiments may include a number of

execution units dedicated to specific functions or sets of functions. Other embodiments may include only one execution unit or one execution unit that can perform a particular function. The illustrated execution logic 450 performs the operations specified by code instructions.

After completion of execution of the operations specified by the code instructions, back end logic 460 retires the instructions of the code 413. In one embodiment, the processor core 400 allows out of order execution but requires in order retirement of instructions. Retirement logic 465 may take a variety of forms as known to those of skill in the art (e.g., re-order buffers or the like). In this manner, the processor core 400 is transformed during execution of the code 413, at least in terms of the output generated by the decoder, the hardware registers and tables utilized by the register renaming logic 425, and any registers (not shown) modified by the execution logic 450.

Although not illustrated in FIG. 12, a processing element may include other elements on chip with the processor core 400. For example, a processing element may include memory control logic along with the processor core 400. The processing element may include I/O control logic and/or may include I/O control logic integrated with memory control logic. The processing element may also include one or more caches.

Referring now to FIG. 13, shown is a block diagram of a computing system 1000 embodiment in accordance with an embodiment. Shown in FIG. 13 is a multiprocessor system 1000 that includes a first processing element 1070 and a second processing element 1080. While two processing elements 1070 and 1080 are shown, it is to be understood that an embodiment of the system 1000 may also include only one such processing element.

The system 1000 is illustrated as a point-to-point interconnect system, wherein the first processing element 1070 and the second processing element 1080 are coupled via a point-to-point interconnect 1050. It should be understood that any or all of the interconnects illustrated in FIG. 13 may be implemented as a multi-drop bus rather than point-to-point interconnect.

As shown in FIG. 13, each of processing elements 1070 and 1080 may be multicore processors, including first and second processor cores (i.e., processor cores 1074a and 1074b and processor cores 1084a and 1084b). Such cores 1074a, 1074b,

1084a, 1084b may be configured to execute instruction code in a manner similar to that discussed above in connection with FIG. 12.

Each processing element 1070, 1080 may include at least one shared cache 1896a, 1896b. The shared cache 1896a, 1896b may store data (e.g., instructions) that are utilized by one or more components of the processor, such as the cores 1074a, 1074b and 1084a, 1084b, respectively. For example, the shared cache 1896a, 1896b may locally cache data stored in a memory 1032, 1034 for faster access by components of the processor. In one or more embodiments, the shared cache 1896a, 1896b may include one or more mid-level caches, such as level 2 (L2), level 3 (L3), level 4 (L4), or other levels of cache, a last level cache (LLC), and/or combinations thereof.

While shown with only two processing elements 1070, 1080, it is to be understood that the scope of the embodiments are not so limited. In other embodiments, one or more additional processing elements may be present in a given processor. Alternatively, one or more of processing elements 1070, 1080 may be an element other than a processor, such as an accelerator or a field programmable gate array. For example, additional processing element(s) may include additional processors(s) that are the same as a first processor 1070, additional processor(s) that are heterogeneous or asymmetric to processor a first processor 1070, accelerators (such as, e.g., graphics accelerators or digital signal processing (DSP) units), field programmable gate arrays, or any other processing element. There can be a variety of differences between the processing elements 1070, 1080 in terms of a spectrum of metrics of merit including architectural, micro architectural, thermal, power consumption characteristics, and the like. These differences may effectively manifest themselves as asymmetry and heterogeneity amongst the processing elements 1070, 1080. For at least one embodiment, the various processing elements 1070, 1080 may reside in the same die package.

The first processing element 1070 may further include memory controller logic (MC) 1072 and point-to-point (P-P) interfaces 1076 and 1078. Similarly, the second processing element 1080 may include a MC 1082 and P-P interfaces 1086 and 1088. As shown in FIG. 13, MC's 1072 and 1082 couple the processors to respective memories, namely a memory 1032 and a memory 1034, which may be portions of main memory locally attached to the respective processors. While the MC 1072 and 1082 is illustrated as integrated into the processing elements 1070, 1080, for alternative

embodiments the MC logic may be discrete logic outside the processing elements 1070, 1080 rather than integrated therein.

The first processing element 1070 and the second processing element 1080 may be coupled to an I/O subsystem 1090 via P-P interconnects 1076 1086, respectively. As shown in FIG. 13, the I/O subsystem 1090 includes P-P interfaces 1094 and 1098. Furthermore, I/O subsystem 1090 includes an interface 1092 to couple I/O subsystem 1090 with a high performance graphics engine 1038. In one embodiment, bus 1049 may be used to couple the graphics engine 1038 to the I/O subsystem 1090. Alternately, a point-to-point interconnect may couple these components.

In turn, I/O subsystem 1090 may be coupled to a first bus 1016 via an interface 1096. In one embodiment, the first bus 1016 may be a Peripheral Component Interconnect (PCI) bus, or a bus such as a PCI Express bus or another third generation I/O interconnect bus, although the scope of the embodiments are not so limited.

As shown in FIG. 13, various I/O devices 1014 (e.g., biometric scanners, speakers, cameras, sensors) may be coupled to the first bus 1016, along with a bus bridge 1018 which may couple the first bus 1016 to a second bus 1020. In one embodiment, the second bus 1020 may be a low pin count (LPC) bus. Various devices may be coupled to the second bus 1020 including, for example, a keyboard/mouse 1012, communication device(s) 1026, and a data storage unit 1019 such as a disk drive or other mass storage device which may include code 1030, in one embodiment. The illustrated code 1030 may implement the method 40 (FIG. 2) and/or the method 50 (FIG. 3), already discussed. Further, an audio I/O 1024 may be coupled to second bus 1020 and a battery 1010 may supply power to the computing system 1000.

Note that other embodiments are contemplated. For example, instead of the point-to-point architecture of FIG. 13, a system may implement a multi-drop bus or another such communication topology. Also, the elements of FIG. 13 may alternatively be partitioned using more or fewer integrated chips than shown in FIG. 13.

Additional Notes and Examples:

Example 1 includes a performance-enhanced computing system comprising a network controller, a processor coupled to the network controller, and a memory coupled to the processor, wherein the memory includes a set of instructions, which when executed by the processor, cause the processor to determine a tensor distribution of a machine learning model on a per-layer basis, set a floating point format of the machine learning model based on the tensor distribution, wherein the floating point

format is variable on the per-layer basis, and generate an inference output from the machine learning model in accordance with the floating point format.

Example 2 includes the computing system of Example 1, wherein to set the floating point format, the instructions, when executed, further cause the computing system to record the floating point format in the machine learning model as an exponent and mantissa bit configuration.

Example 3 includes the computing system of Example 1, wherein the instructions, when executed, further cause the computing system to identify a mean squared error (MSE) of the tensor distribution.

Example 4 includes the computing system of Example 3, wherein the instructions, when executed, further cause the computing system to identify an outlier signature of the tensor distribution if a difference between a lowest MSE and a second lowest MSE is less than a threshold, wherein the outlier signature is identified on a per-input channel basis, and wherein the floating point format is to correspond to a lowest MSE if a difference between the lowest MSE and a second lowest MSE is greater than the threshold.

Example 5 includes the computing system of Example 1, wherein the machine learning model includes one of a computer vision model or a natural language processing model.

Example 6 includes at least one computer readable storage medium comprising a set of instructions, which when executed by a computing system, cause the computing system to determine a tensor distribution of a machine learning model on a per-layer basis, and set a floating point format of the machine learning model based on the tensor distribution, wherein the floating point format is variable on the per-layer basis.

Example 7 includes the at least one computer readable storage medium of Example 6, wherein to set the floating point format, the instructions, when executed, further cause the computing system to record the floating point format in the machine learning model as an exponent and mantissa bit configuration.

Example 8 includes the at least one computer readable storage medium of Example 6, wherein the instructions, when executed, further cause the computing system to identify a mean squared error (MSE) of the tensor distribution.

Example 9 includes the at least one computer readable storage medium of Example 8, wherein the floating point format is to correspond to a lowest MSE if a

difference between the lowest MSE and a second lowest MSE is greater than a threshold.

Example 10 includes the at least one computer readable storage medium of Example 8, wherein the instructions, when executed, further cause the computing system to identify an outlier signature of the tensor distribution if a difference between a lowest MSE and a second lowest MSE is less than a threshold, and wherein the outlier signature is identified on a per-input channel basis.

Example 11 includes the at least one computer readable storage medium of Example 6, wherein the machine learning model includes one of a computer vision model or a natural language processing model.

Example 12 includes the at least one computer readable storage medium of any one of Examples 6 to 11, wherein the instructions, when executed, further cause the computing system to generate an inference output from the machine learning model in accordance with the floating point format.

Example 13 includes a semiconductor apparatus comprising one or more substrates, and logic coupled to the one or more substrates, wherein the logic is implemented at least partly in one or more of configurable or fixed-functionality hardware, the logic to determine a tensor distribution of a machine learning model on a per-layer basis, and set a floating point format of the machine learning model based on the tensor distribution, wherein the floating point format is variable on the per-layer basis.

Example 14 includes the semiconductor apparatus of Example 13, wherein to set the floating point format, the logic is to record the floating point format in the machine learning model as an exponent and mantissa bit configuration.

Example 15 includes the semiconductor apparatus of Example 13, wherein the logic is further to identify a mean squared error (MSE) of the tensor distribution.

Example 16 includes the semiconductor apparatus of Example 15, wherein the floating point format is to correspond to a lowest MSE if a difference between the lowest MSE and a second lowest MSE is greater than a threshold.

Example 17 includes the semiconductor apparatus of Example 15, wherein the logic is to identify an outlier signature of the tensor distribution if a difference between a lowest MSE and a second lowest MSE is less than a threshold, and wherein the outlier signature is identified on a per-input channel basis.

Example 18 includes the semiconductor apparatus of Example 13, wherein the machine learning model includes one of a computer vision model or a natural language processing model.

Example 19 includes the semiconductor apparatus of any one of Examples 13 to 18, wherein the logic is to generate an inference output from the machine learning model in accordance with the floating point format.

Example 20 includes the semiconductor apparatus of any one of Examples 13 to 18, wherein the logic coupled to the one or more substrates includes transistor channel regions that are positioned within the one or more substrates.

Example 21 includes a method of operating a performance-enhanced computing system, the method comprising determining a tensor distribution of a machine learning model on a per-layer basis, setting a floating point format of the machine learning model based on the tensor distribution, wherein the floating point format is variable on the per-layer basis, and generating an inference output from the machine learning model in accordance with the floating point format.

Example 22 includes an apparatus comprising means for performing the method of Example 21.

Embodiments are applicable for use with all types of semiconductor integrated circuit ("IC") chips. Examples of these IC chips include but are not limited to processors, controllers, chipset components, programmable logic arrays (PLAs), memory chips, network chips, systems on chip (SoCs), SSD/NAND controller ASICs, and the like. In addition, in some of the drawings, signal conductor lines are represented with lines. Some may be different, to indicate more constituent signal paths, have a number label, to indicate a number of constituent signal paths, and/or have arrows at one or more ends, to indicate primary information flow direction. This, however, should not be construed in a limiting manner. Rather, such added detail may be used in connection with one or more exemplary embodiments to facilitate easier understanding of a circuit. Any represented signal lines, whether or not having additional information, may actually comprise one or more signals that may travel in multiple directions and may be implemented with any suitable type of signal scheme, e.g., digital or analog lines implemented with differential pairs, optical fiber lines, and/or single-ended lines.

Example sizes/models/values/ranges may have been given, although embodiments are not limited to the same. As manufacturing techniques (e.g.,

photolithography) mature over time, it is expected that devices of smaller size could be manufactured. In addition, well known power/ground connections to IC chips and other components may or may not be shown within the figures, for simplicity of illustration and discussion, and so as not to obscure certain aspects of the embodiments. Further, arrangements may be shown in block diagram form in order to avoid obscuring embodiments, and also in view of the fact that specifics with respect to implementation of such block diagram arrangements are highly dependent upon the computing system within which the embodiment is to be implemented, i.e., such specifics should be well within purview of one skilled in the art. Where specific details (e.g., circuits) are set forth in order to describe example embodiments, it should be apparent to one skilled in the art that embodiments can be practiced without, or with variation of, these specific details. The description is thus to be regarded as illustrative instead of limiting.

The term "coupled" may be used herein to refer to any type of relationship, direct or indirect, between the components in question, and may apply to electrical, mechanical, fluid, optical, electromagnetic, electromechanical or other connections. In addition, the terms "first", "second", etc. may be used herein only to facilitate discussion, and carry no particular temporal or chronological significance unless otherwise indicated.

As used in this application and in the claims, a list of items joined by the term "one or more of" may mean any combination of the listed terms. For example, the phrases "one or more of A, B or C" may mean A; B; C; A and B; A and C; B and C; or A, B and C.

Those skilled in the art will appreciate from the foregoing description that the broad techniques of the embodiments can be implemented in a variety of forms. Therefore, while the embodiments have been described in connection with particular examples thereof, the true scope of the embodiments should not be so limited since other modifications will become apparent to the skilled practitioner upon a study of the drawings, specification, and following claims.

CLAIMS

We claim:

1. A performance-enhanced computing system comprising:

a network controller;

a processor coupled to the network controller; and

a memory coupled to the processor, wherein the memory includes a set of instructions, which when executed by the processor, cause the processor to:

determine a tensor distribution of a machine learning model on a per-layer basis,

set a floating point format of the machine learning model based on the tensor distribution, wherein the floating point format is variable on the per-layer basis, and

generate an inference output from the machine learning model in accordance with the floating point format.

2. The computing system of claim 1, wherein to set the floating point format, the instructions, when executed, further cause the computing system to record the floating point format in the machine learning model as an exponent and mantissa bit configuration.

3. The computing system of claim 1, wherein the instructions, when executed, further cause the computing system to identify a mean squared error (MSE) of the tensor distribution.

4. The computing system of claim 3, wherein the instructions, when executed, further cause the computing system to identify an outlier signature of the tensor distribution if a difference between a lowest MSE and a second lowest MSE is less than a threshold, wherein the outlier signature is identified on a per-input channel basis, and wherein the floating point format is to correspond to a lowest MSE if a difference between the lowest MSE and a second lowest MSE is greater than the threshold.

5. The computing system of claim 1, wherein the machine learning model includes one of a computer vision model or a natural language processing model.

6. At least one computer readable storage medium comprising a set of instructions, which when executed by a computing system, cause the computing system to:

determine a tensor distribution of a machine learning model on a per-layer basis; and

set a floating point format of the machine learning model based on the tensor distribution, wherein the floating point format is variable on the per-layer basis.

7. The at least one computer readable storage medium of claim 6, wherein to set the floating point format, the instructions, when executed, further cause the computing system to record the floating point format in the machine learning model as an exponent and mantissa bit configuration.

8. The at least one computer readable storage medium of claim 6, wherein the instructions, when executed, further cause the computing system to identify a mean squared error (MSE) of the tensor distribution.

9. The at least one computer readable storage medium of claim 8, wherein the floating point format is to correspond to a lowest MSE if a difference between the lowest MSE and a second lowest MSE is greater than a threshold.

10. The at least one computer readable storage medium of claim 8, wherein the instructions, when executed, further cause the computing system to identify an outlier signature of the tensor distribution if a difference between a lowest MSE and a second lowest MSE is less than a threshold, and wherein the outlier signature is identified on a per-input channel basis.

11. The at least one computer readable storage medium of claim 6, wherein the machine learning model includes one of a computer vision model or a natural language processing model.

12. The at least one computer readable storage medium of any one of claims 6 to 11, wherein the instructions, when executed, further cause the computing system to generate an inference output from the machine learning model in accordance with the floating point format.

13. A semiconductor apparatus comprising:

one or more substrates; and

logic coupled to the one or more substrates, wherein the logic is implemented at least partly in one or more of configurable or fixed-functionality hardware, the logic to:

determine a tensor distribution of a machine learning model on a per-layer basis; and

set a floating point format of the machine learning model based on the tensor distribution, wherein the floating point format is variable on the per-layer basis.

14. The semiconductor apparatus of claim 13, wherein to set the floating point format, the logic is to record the floating point format in the machine learning model as an exponent and mantissa bit configuration.

15. The semiconductor apparatus of claim 13, wherein the logic is further to identify a mean squared error (MSE) of the tensor distribution.

16. The semiconductor apparatus of claim 15, wherein the floating point format is to correspond to a lowest MSE if a difference between the lowest MSE and a second lowest MSE is greater than a threshold.

17. The semiconductor apparatus of claim 15, wherein the logic is to identify an outlier signature of the tensor distribution if a difference between a lowest MSE and a second lowest MSE is less than a threshold, and wherein the outlier signature is identified on a per-input channel basis.

18. The semiconductor apparatus of claim 13, wherein the machine learning model includes one of a computer vision model or a natural language processing model.

19. The semiconductor apparatus of any one of claims 13 to 18, wherein the logic is to generate an inference output from the machine learning model in accordance with the floating point format.

5

20. The semiconductor apparatus of any one of claims 13 to 18, wherein the logic coupled to the one or more substrates includes transistor channel regions that are positioned within the one or more substrates.

10

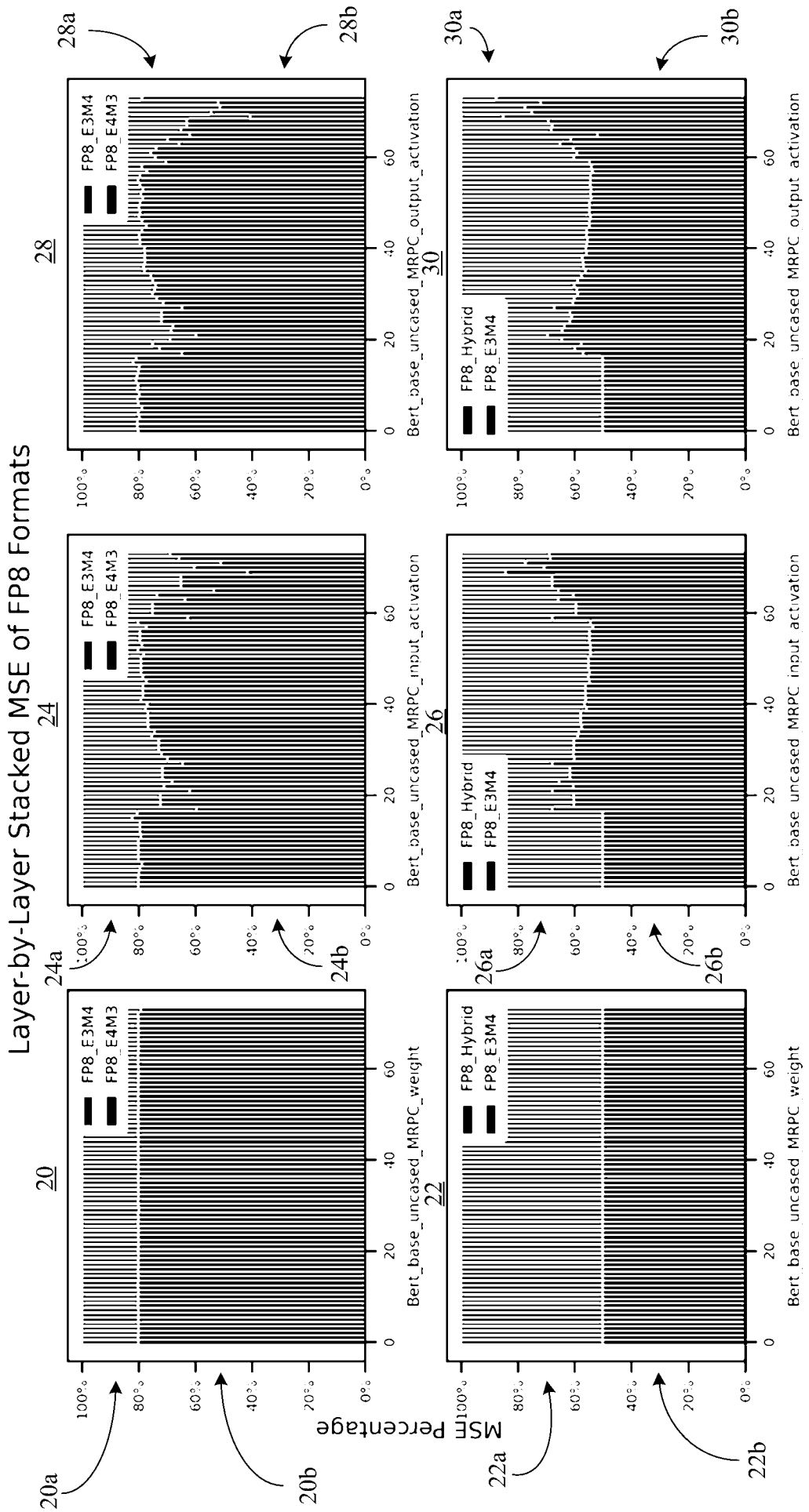

Layer-by-Layer Stacked MSE of FP8 Formats
MSE Percentage
20
20a
20b
Bert_base_uncased_MRPC_weight
22
22a
22b
Bert_base_uncased_MRPC_input_activation
24
24a
24b
Bert_base_uncased_MRPC_input_activation
26
26a
26b
Bert_base_uncased_MRPC_output_activation
28
28a
28b
Bert_base_uncased_MRPC_output_activation
30
30a
30b
FP8_E3M4
FP8_E4M3
FP8_Hybrid
0
20
40
60
80
100

FIG. 1

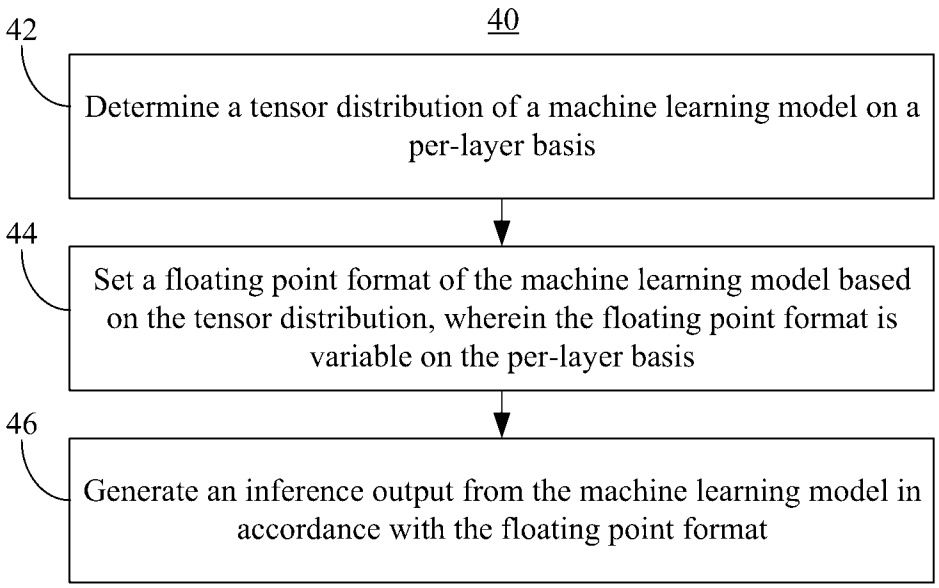

40
42
Determine a tensor distribution of a machine learning model on a per-layer basis
44
Set a floating point format of the machine learning model based on the tensor distribution, wherein the floating point format is variable on the per-layer basis
46
Generate an inference output from the machine learning model in accordance with the floating point format

FIG. 2

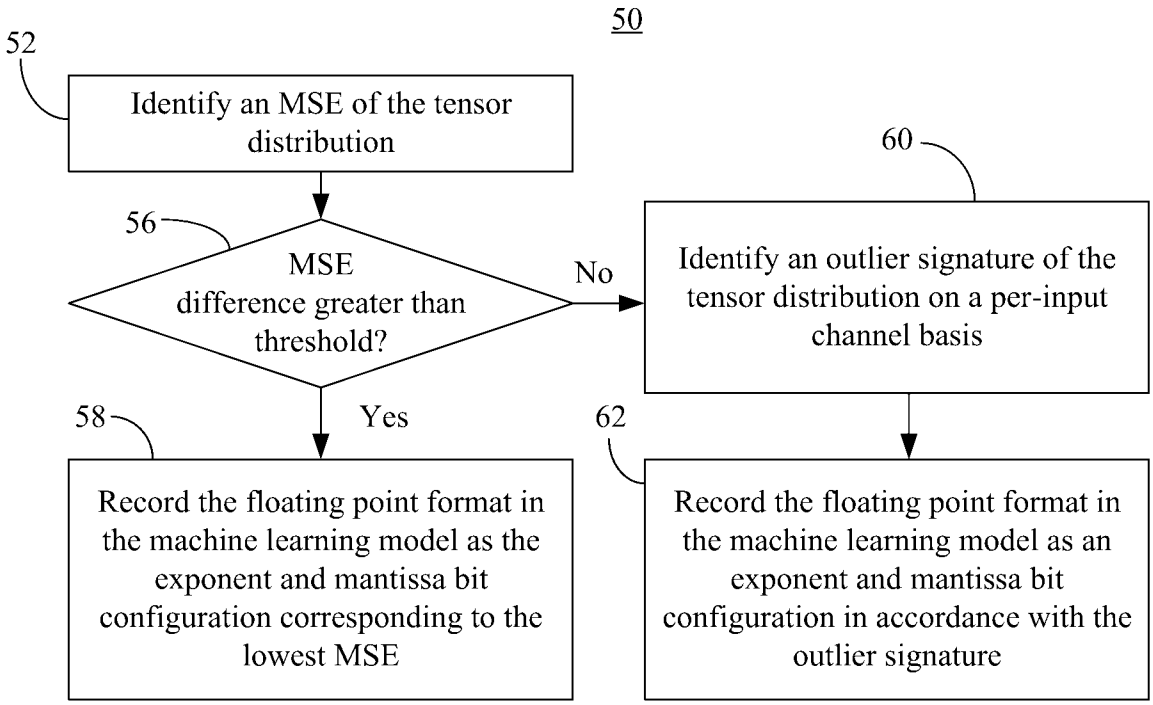

50
52
Identify an MSE of the tensor distribution
56
MSE difference greater than threshold?
No
Yes
60
Identify an outlier signature of the tensor distribution on a per-input channel basis
58
Record the floating point format in the machine learning model as the exponent and mantissa bit configuration corresponding to the lowest MSE
62
Record the floating point format in the machine learning model as an exponent and mantissa bit configuration in accordance with the outlier signature

FIG. 3

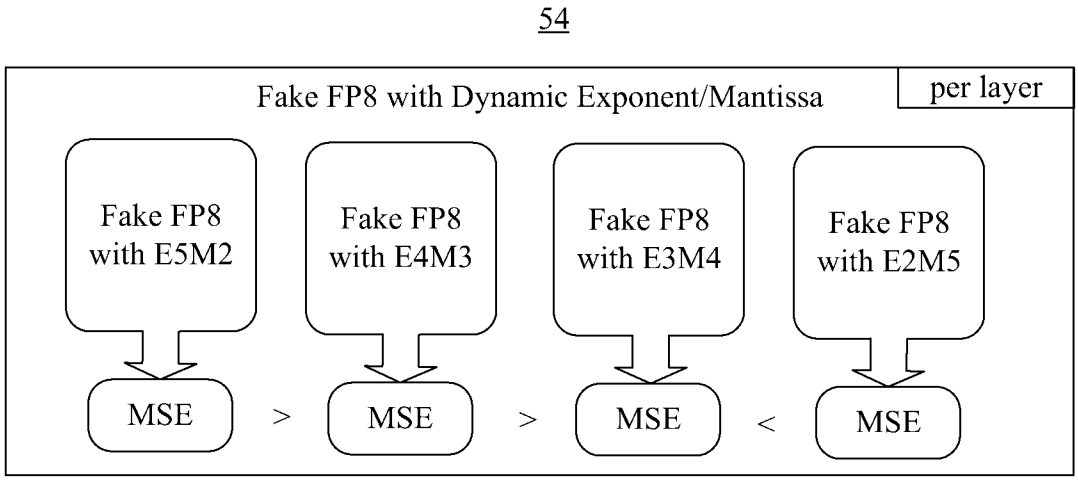

54
Fake FP8 with Dynamic Exponent/Mantissa
per layer
Fake FP8 with E5M2
Fake FP8 with E4M3
Fake FP8 with E3M4
Fake FP8 with E2M5
MSE
>
MSE
>
MSE
<
MSE

FIG. 4

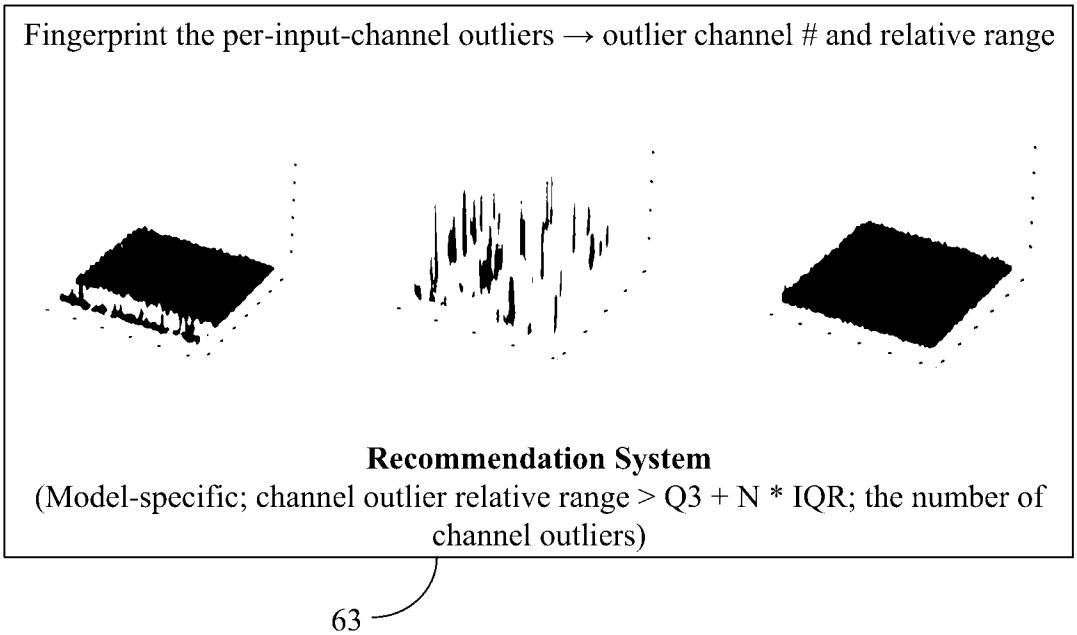

Fingerprint the per-input-channel outliers → outlier channel # and relative range
Recommendation System
(Model-specific; channel outlier relative range > Q3 + N * IQR; the number of channel outliers)
63

FIG. 5

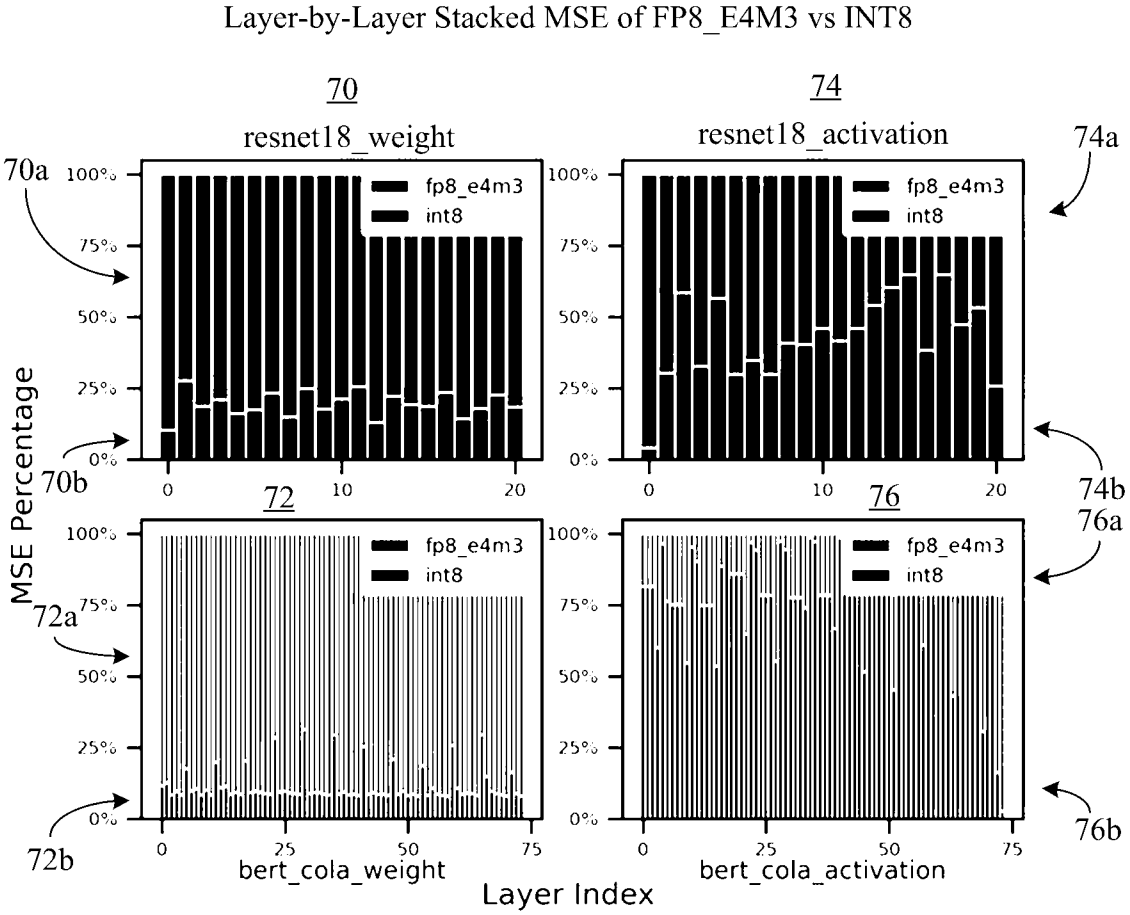

Layer-by-Layer Stacked MSE of FP8_E4M3 vs INT8
70
resnet18_weight
74
resnet18_activation
70a
74a
fp8_e4m3
int8
MSE Percentage
100%
75%
50%
25%
0%
70b
74b
0
10
20
72
76
72a
76a
0
25
50
75
72b
76b
bert_cola_weight
bert_cola_activation
Layer Index

FIG. 6

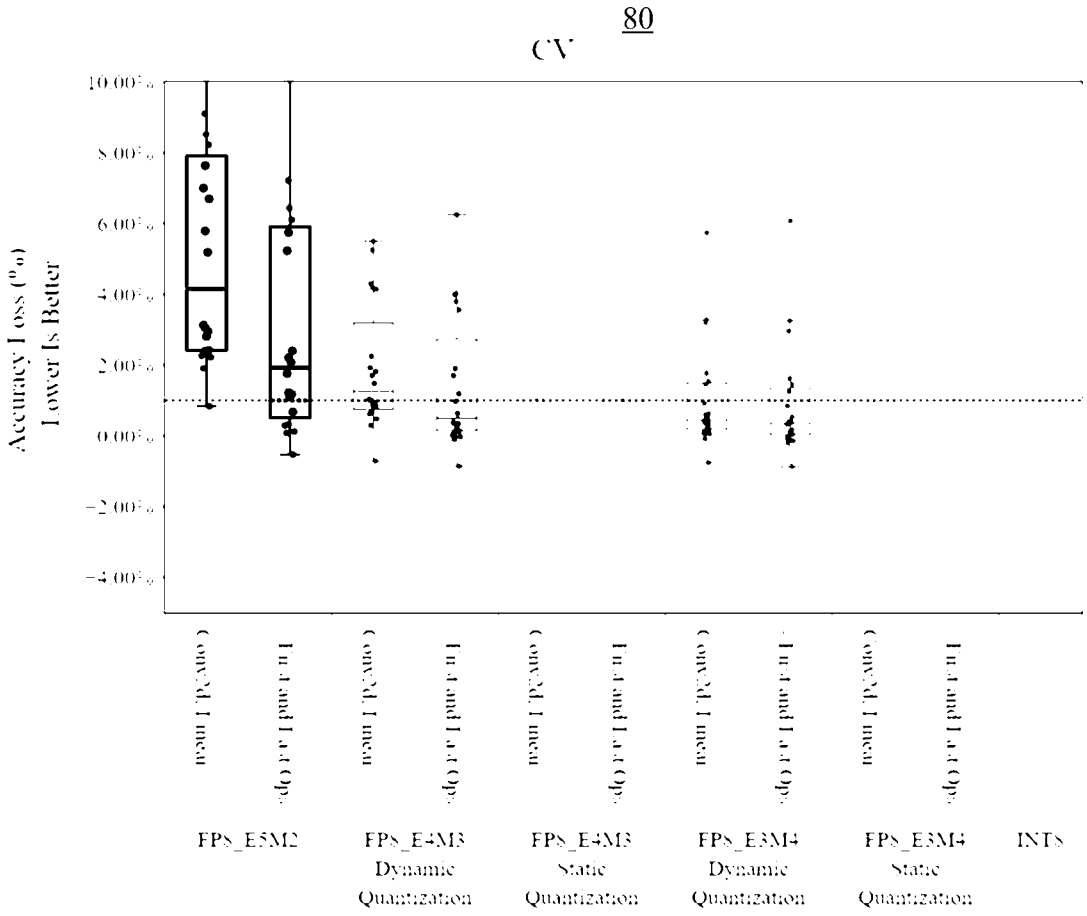

80
CV
Accuracy Loss (%)
Lower Is Better
10.00%
8.00%
6.00%
4.00%
2.00%
0.00%
-2.00%
-4.00%
ConvNeXt-Large
FP8_E5M2
FP8_E4M3
Dynamic
Quantization
FP8_E4M3
Static
Quantization
FP8_E3M4
Dynamic
Quantization
FP8_E3M4
Static
Quantization
INT8

FIG. 7

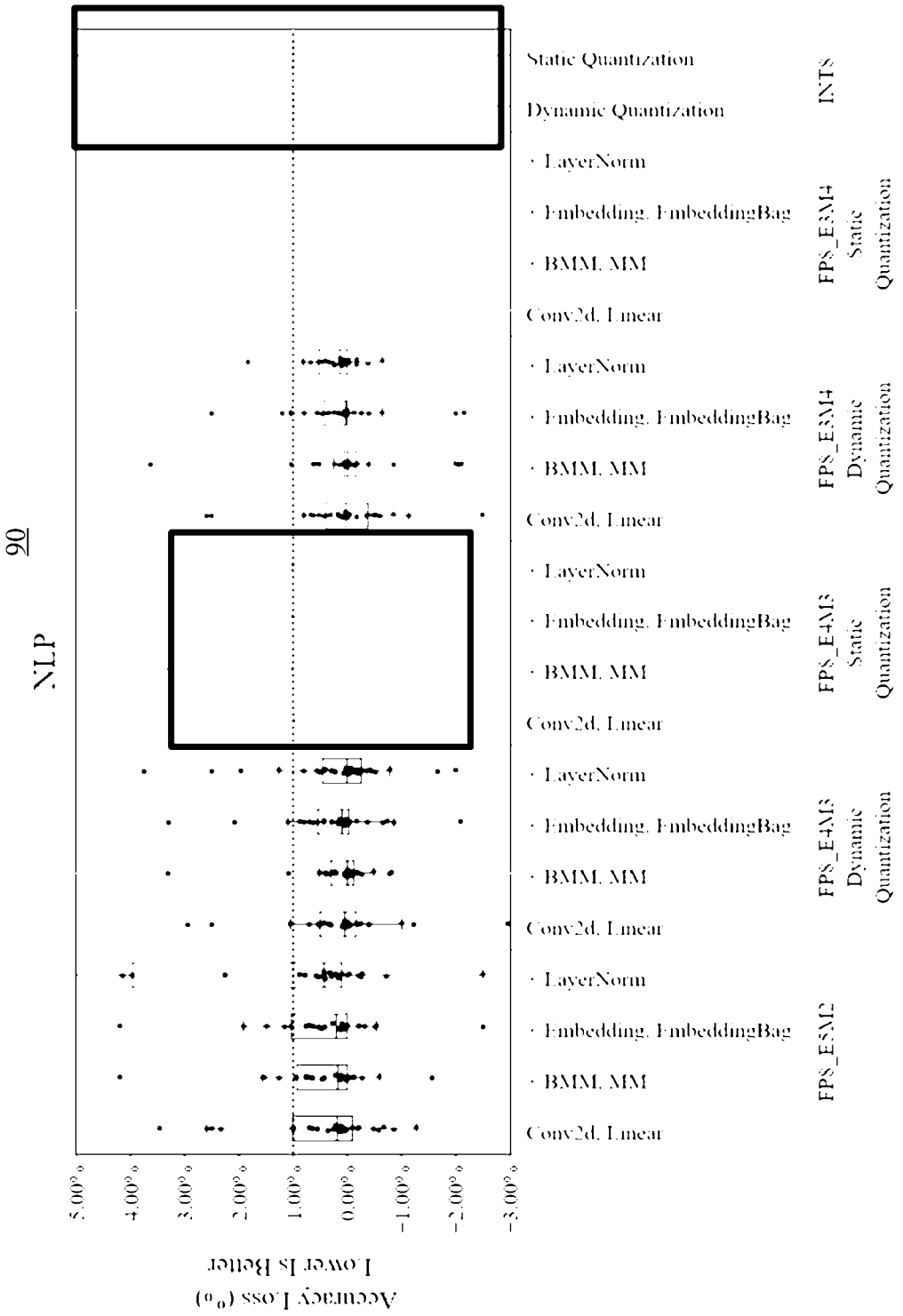

90
NLP
Accuracy Loss (%)
Lower Is Better
5.00%
4.00%
3.00%
2.00%
1.00%
0.00%
−1.00%
−2.00%
−3.00%
Conv2d, Linear
· BMM, MM
· Embedding, EmbeddingBag
· LayerNorm
FP8_E5M2
Conv2d, Linear
· BMM, MM
· Embedding, EmbeddingBag
· LayerNorm
FP8_E4M3 Dynamic Quantization
Conv2d, Linear
· BMM, MM
· Embedding, EmbeddingBag
· LayerNorm
FP8_E4M3 Static Quantization
Conv2d, Linear
· BMM, MM
· Embedding, EmbeddingBag
· LayerNorm
FP8_E3M4 Dynamic Quantization
Conv2d, Linear
· BMM, MM
· Embedding, EmbeddingBag
· LayerNorm
FP8_E3M4 Static Quantization
Conv2d, Linear
Dynamic Quantization
Static Quantization
INT8

FIG. 8

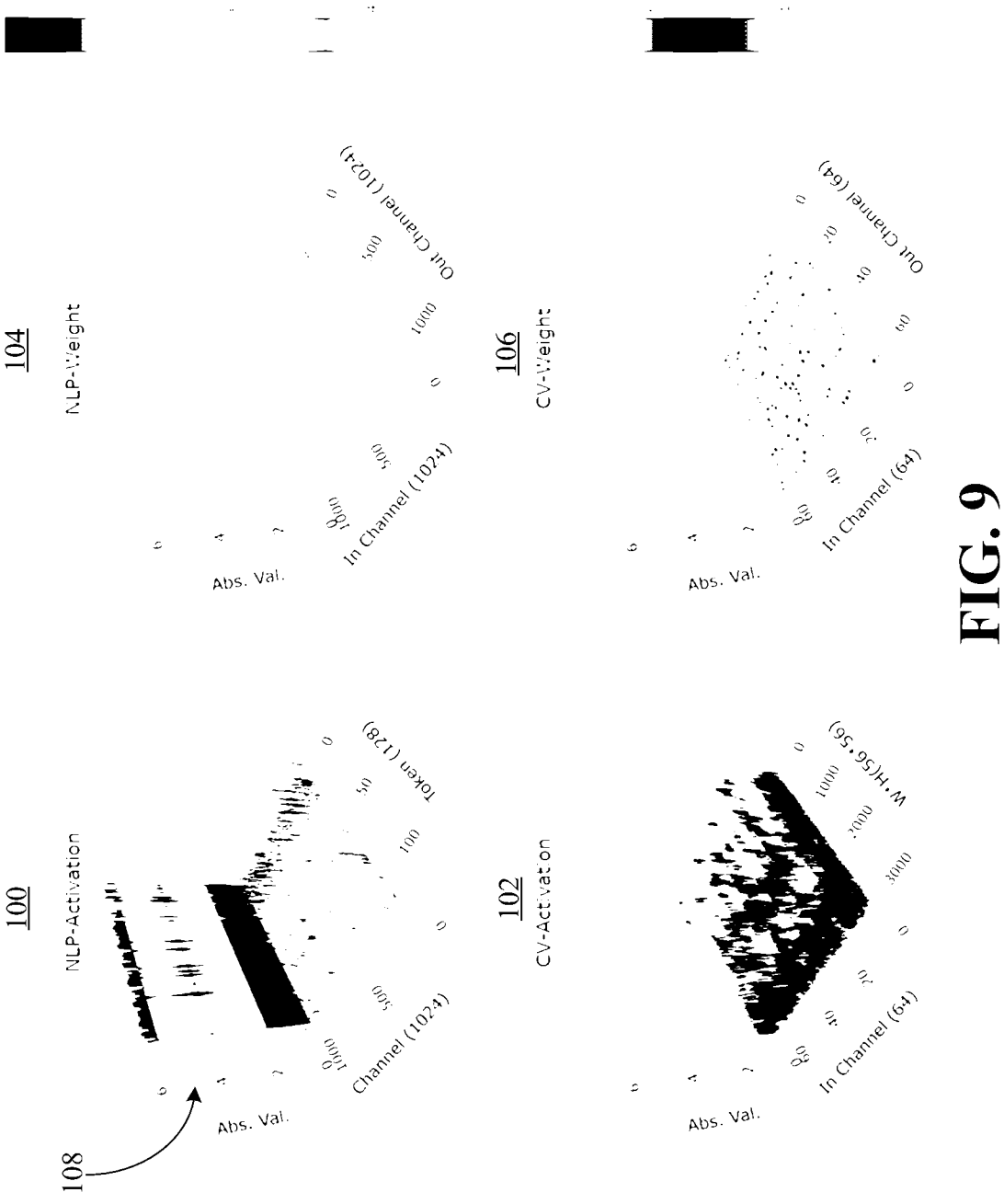

100
NLP-Activation
Token (128)
Channel (1024)
Abs. Val.
102
CV-Activation
W*H(56*56)
In Channel (64)
Abs. Val.
104
NLP-Weight
Out Channel (1024)
In Channel (1024)
Abs. Val.
106
CV-Weight
Out Channel (64)
In Channel (64)
Abs. Val.
108

FIG. 9

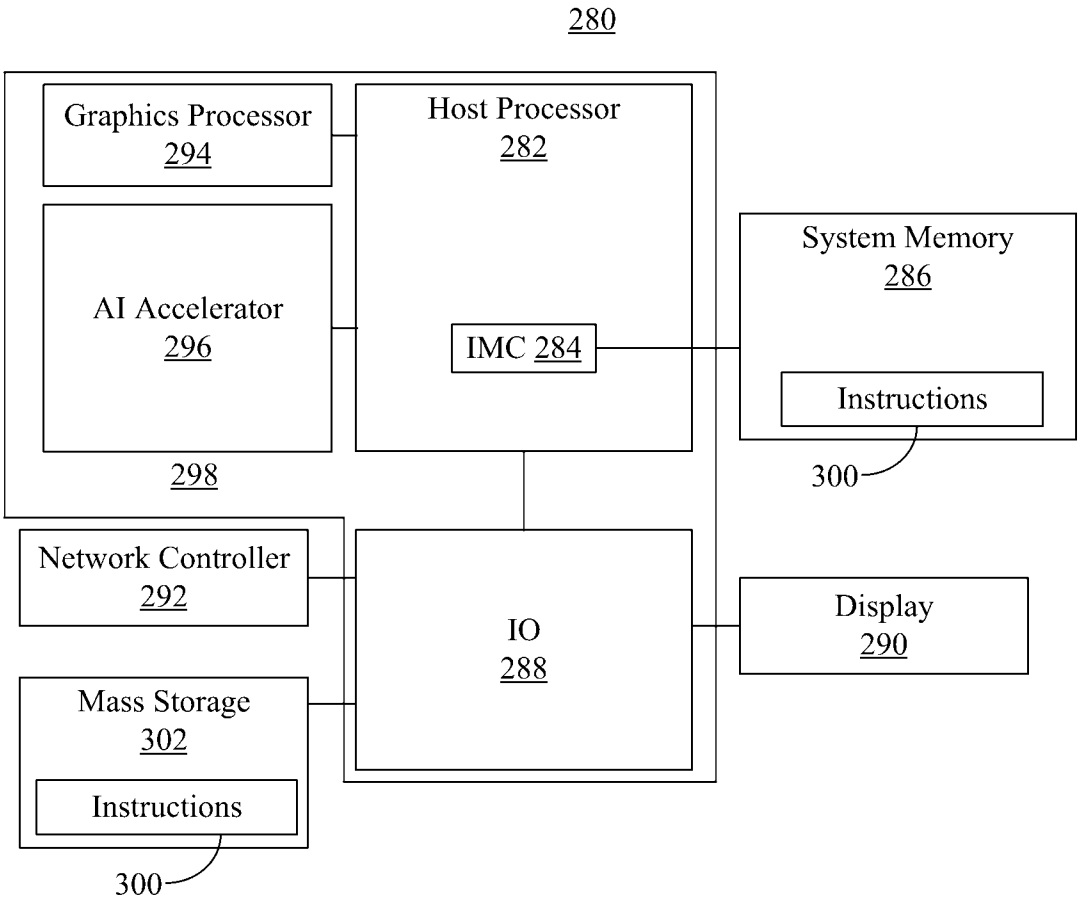

280
Graphics Processor
294
Host Processor
282
AI Accelerator
296
IMC 284
System Memory
286
Instructions
300
298
Network Controller
292
IO
288
Display
290
Mass Storage
302
Instructions
300

FIG. 10

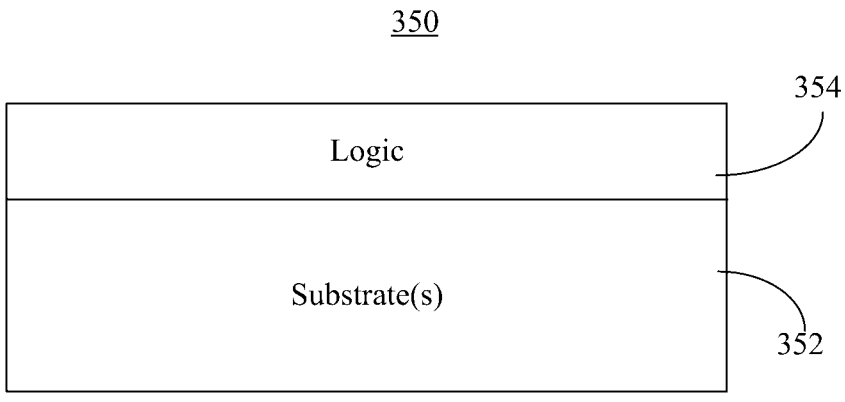

350
354
Logic
Substrate(s)
352

FIG. 11

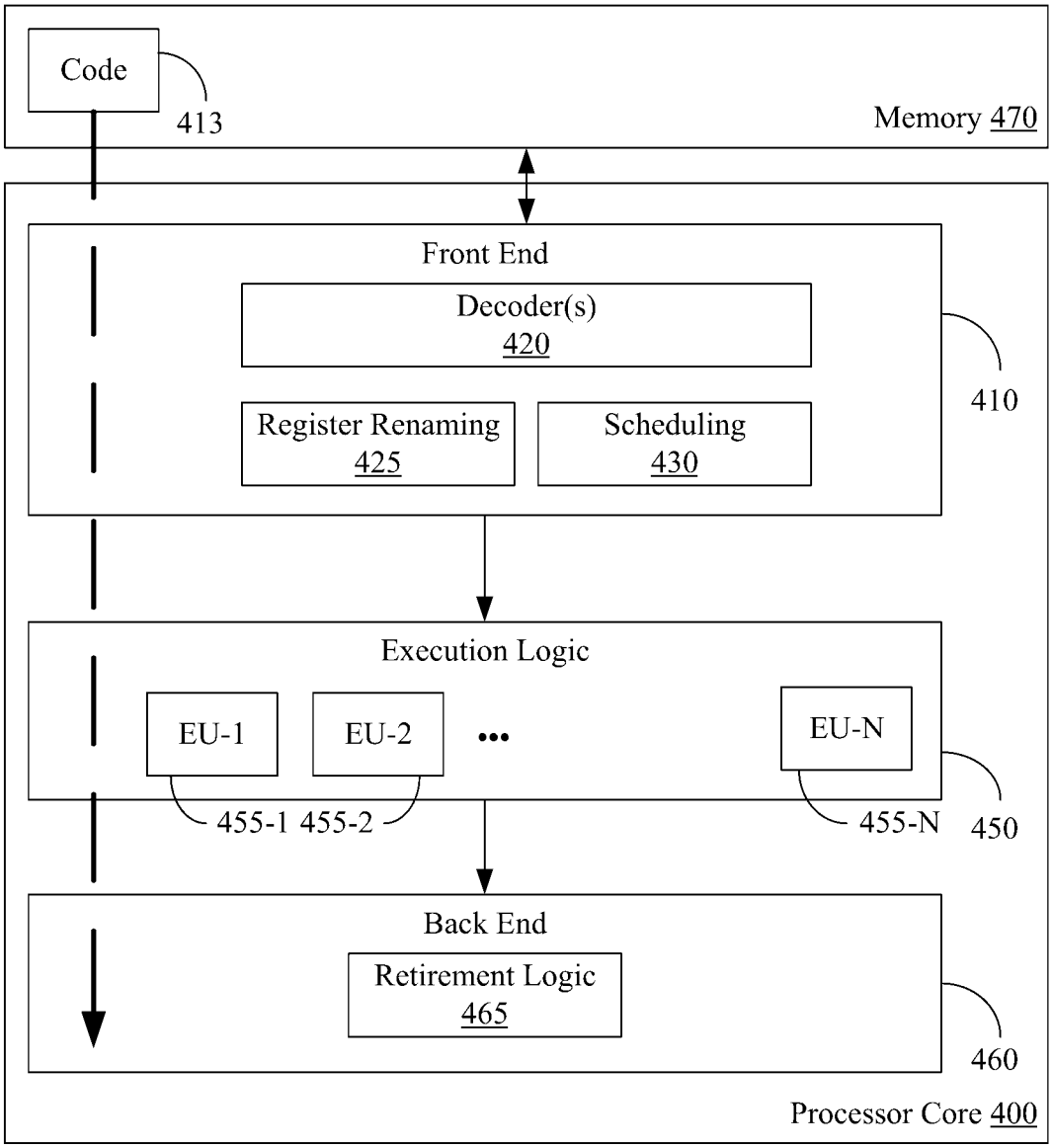
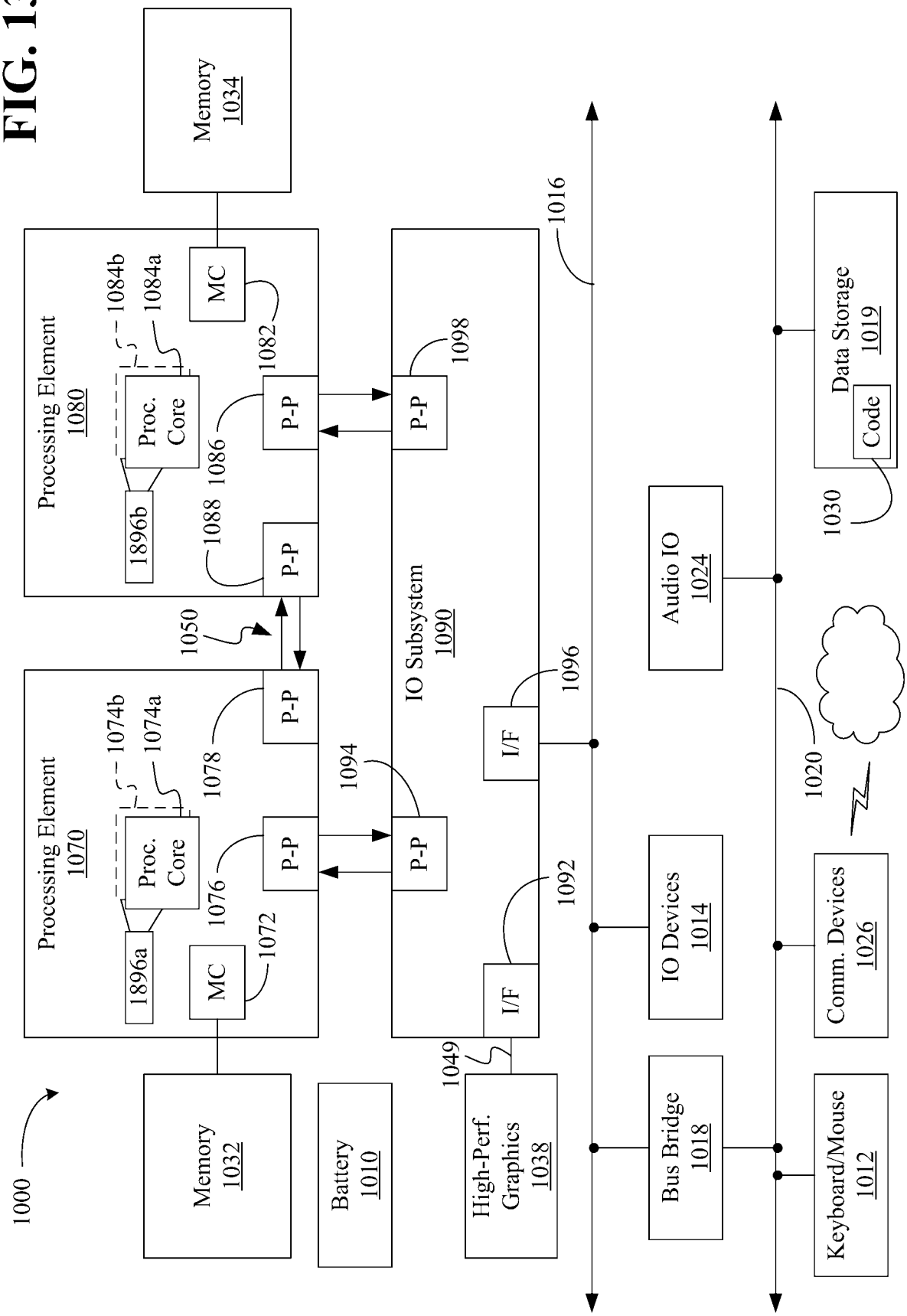

Code
413
Memory 470
Front End
Decoder(s)
420
Register Renaming
425
Scheduling
430
410
Execution Logic
EU-1
EU-2
EU-N
455-1
455-2
455-N
450
Back End
Retirement Logic
465
460
Processor Core 400

FIG. 12

FIG. 13


1000
Memory 1032
Battery 1010
High-Perf. Graphics 1038
1049
I/F
1092
IO Devices 1014
Bus Bridge 1018
Keyboard/Mouse 1012
Comm. Devices 1026
1020
Audio IO 1024
Data Storage 1019
Code
1030
1016
1096
I/F
IO Subsystem 1090
P-P
1098
1094
P-P
1078
1074b
1074a
Processing Element 1070
Proc. Core
1076
P-P
1072
MC
1896a
1050
1088
P-P
1086
1084b
1084a
Proc. Core
1082
P-P
MC
Processing Element 1080
1896b
Memory 1034

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2023/113202

A. CLASSIFICATION OF SUBJECT MATTER

G06F 7/483(2006.01)i

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)

IPC: G06N, G06F

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)

CNTXT, ENTXT, DWPI, BAIDU, CNKI, IEEE: tensor, distribution, machine learning, model, floating point, format, variable, layer, MSE, threshold

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2019042944 A1 (INTEL CORPORATION) 07 February 2019 (2019-02-07) description, paragraphs [0016]-[0099]	1-20
A	US 2020193274 A1 (MICROSOFT TECHNOLOGY LICENSING, LLC) 18 June 2020 (2020-06-18) the whole document	1-20
A	US 2019122100 A1 (SAMSUNG ELECTRONICS CO., LTD.) 25 April 2019 (2019-04-25) the whole document	1-20
A	US 2017372202 A1 (NVIDIA CORPORATION) 28 December 2017 (2017-12-28) the whole document	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
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 December 2023	**27 December 2023**

Name and mailing address of the ISA/CN	Authorized officer
CHINA NATIONAL INTELLECTUAL PROPERTY ADMINISTRATION **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088, China**	**WU,ShaoHong** Telephone No. **(+86) 010-53961533**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2023/113202

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2019042944 A1	07 February 2019	KR 20190139757 A	18 December 2019
		DE 102019111995 A1	12 December 2019
		JP 2019212295 A	12 December 2019
		JP 7331303 B2	23 August 2023
		CN 110580524 A	17 December 2019
		KR 20190139757 A	18 December 2019
US 2020193274 A1	18 June 2020	ES 2927404 T3	04 November 2022
		EP 3955173 A1	16 February 2022
		EP 3955173 B1	24 August 2022
		EP 3877913 A1	15 September 2021
		WO 2020131390 A1	25 June 2020
		US 11676003 B2	13 June 2023
		US 2023267319 A1	24 August 2023
US 2019122100 A1	25 April 2019	JP 2019079531 A	23 May 2019
		JP 7296709 B2	23 June 2023
		KR 20190043849 A	29 April 2019
		KR 102564456 B1	07 August 2023
		US 11593625 B2	28 February 2023
		EP 3474194 A1	24 April 2019
		EP 3474194 B1	20 September 2023
		CN 109685198 A	26 April 2019
US 2017372202 A1	28 December 2017	DE 102017113232 A1	21 December 2017
		CN 107526709 A	29 December 2017