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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 April 2020 (16.04.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/074989 A1

(51) International Patent Classification:
G06N 3/06 (2006.01)

(21) International Application Number:
PCT/IB2019/058120

(22) International Filing Date:
25 September 2019 (25.09.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
16/157,852 11 October 2018 (11.10.2018) US

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, New York 10504 (US).

(71) Applicants *(for MG only)*: **IBM UNITED KINGDOM LIMITED** [GB/GB]; PO Box 41, North Harbour, Portsmouth Hampshire PO6 3AU (GB). **IBM (CHINA) INVESTMENT COMPANY LIMITED** [CN/CN]; 25/F, Pangu Plaza, No.27, Central North 4th Ring Road, Chaoyang District, Beijing 100101 (CN).

(72) Inventors: **ARTHUR, John, Vernon**; IBM Corporation, Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099 (US). **CASSIDY, Andrew, Stephen**; IBM Corporation, 11501 Burnet Road, Austin, Texas 78758-3400 (US). **FLICKNER, Myron**; IBM Corporation, Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099 (US). **DATTA, Pallab**; IBM Corporation, Almaden Research Center, 650 Harry Road, San Jose, California 95120-609 (US). **PENNER, Hartmut**; IBM Corporation, Almaden Research Center, Intellectual Property Law C4TA/J2B, 650 Harry Road, San Jose, California 95120-6099 (US). **APPUSWAMY, Rathinakumar**; IBM Corporation, Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099 (US). **SAWADA,**

(54) Title: DATA REPRESENTATION FOR DYNAMIC PRECISION IN NEURAL NETWORK CORES

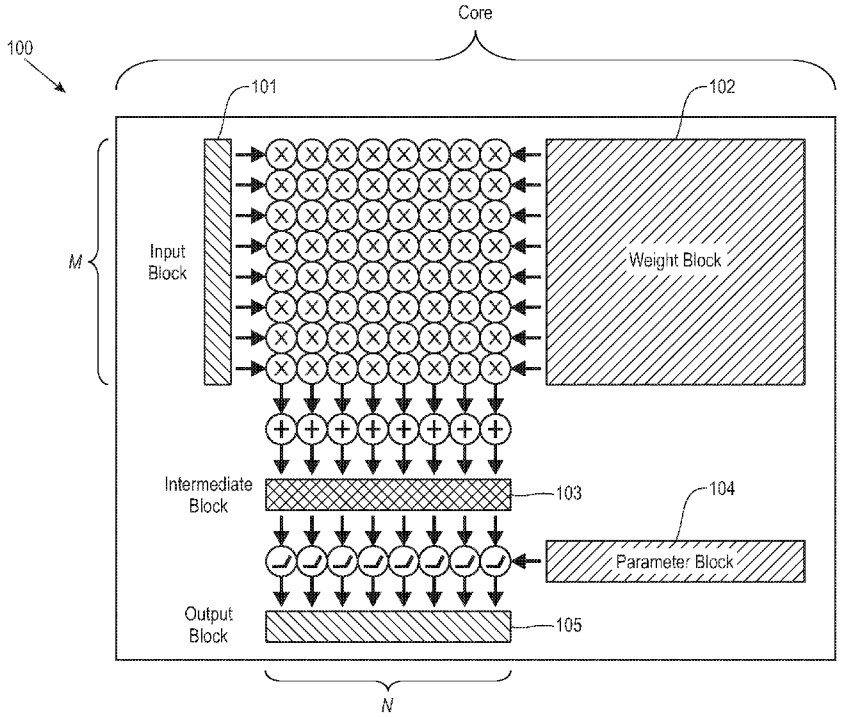

FIG. 1

(57) Abstract: Systems for neural network computation are provided. A neural network processor comprises a plurality of neural cores. The neural network processor has one or more processor precisions per activation. The processor is configured to accept data having a processor feature dimension. A transformation circuit is coupled to the neural network processor, and is adapted to:receive an input data tensor having an input precision per channel at one or more features; transform the input data tensor from the input precision to the processor precision; divide the input data into a plurality of blocks, each block conforming to one of the processor feature dimensions; provide each of the plurality of blocks to one of the plurality of neural cores. The neural network processor is adapted to compute, by the plurality of neural cores,output of one or more neural network layers.

WO 2020/074989 A1

[Continued on next page]

Jun; IBM Corporation, 11501 Burnet Road, Austin, Texas 78758-3400 (US). **MODHA, Dharmendra**; IBM Corporation, Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099 (US). **ESSER, Steven, Kyle**; IBM Corporation, Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099 (US). **TABA, Brian, Seisho**; IBM Corporation, Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099 (US). **KLAMO, Jennifer**; IBM Corporation, 650 Harry Road, Almaden Research Center, San Jose, California 95120-6099 (US).

(74) **Agent: LITHERLAND, David**; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, 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, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Published:

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

DATA REPRESENTATION FOR DYNAMIC PRECISION IN NEURAL NETWORK CORES

BACKGROUND

[0001] Embodiments of the present disclosure relate to multi-core hardware implementations of neural networks, and more specifically, to data representation for dynamic precision in neural network cores.

BRIEF SUMMARY

[0002] According to embodiments of the present disclosure, methods of and computer program products for operating a neural network are provided. In various embodiments, an input data tensor is received at a neural network processor comprising a plurality of neural cores. The input data tensor has feature dimensions at an input bit precision. The neural network processor is configured for one or more processor feature dimensions at one or more processor bit precisions. The input data is transformed from the input bit precision to one of the processor bit precisions. The input data tensor is divided into a plurality of blocks, each block conforming to one of the processor feature dimensions. Each of the plurality of blocks is provided to one of the plurality of neural cores. The plurality of neural cores computes output of one or more neural network layers.

[0003] According to embodiments of the present disclosure, systems for neural network computation are provided. A neural network processor comprises a plurality of neural cores. The neural network processor has one or more processor precisions per activation. The processor is configured to accept data having a processor feature dimension. A transformation circuit is coupled to the neural network processor, and is adapted to: receive an input data tensor having an input precision per channel at one or more features; transform the input data tensor from the input precision to the processor precision; divide the input data into a plurality of blocks, each block conforming to one of the processor feature dimensions; provide each of the plurality of blocks to one of the plurality of neural cores. The neural network processor is adapted to compute, by the plurality of neural cores, output of one or more neural network layers.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

[0004] **Fig. 1** illustrates a neural core according to embodiments of the present disclosure.

[0005] **Fig. 2** illustrates an exemplary Inference Processing Unit (IPU) according to embodiments of the present disclosure.

[0006] **Fig. 3** illustrates a multi-core Inference Processing Unit (IPU) according to embodiments of the present disclosure.

[0007] **Fig. 4** illustrates tranformtation according to embodiments of the present disclosure.

[0008] **Fig. 5** illustrates a method of operating a neural network according to embodiments of the present disclosure.

[0009] **Fig. 6** depicts a computing node according to an embodiment of the present disclosure.

DETAILED DESCRIPTION

[0010] An artificial neuron is a mathematical function whose output is a nonlinear function of a linear combination of its inputs. Two neurons are connected if the output of one is an input to the other. A weight is a scalar value encoding the strength of the connection between the output of one neuron and the input of another neuron.

[0011] A neuron computes its output, called an activation, by applying a nonlinear activation function to a weighted sum of its inputs. A weighted sum is an intermediate result computed by multiplying each input with the corresponding weight and accumulating the products. A partial sum is a weighted sum of a subset of inputs. A weighted sum of all inputs may be computed in stages by accumulating one or more partial sums.

[0012] A neural network is a collection of one or more neurons. A neural network is often divided into groups of neurons called layers. A layer is a collection of one or more neurons that all receive input from the same layers and all send output to the same layers, and typically perform a similar function. An input layer is a layer that receives input from a source outside the neural network. An output layer is a layer that sends output to a target outside the neural network. All other layers are intermediate processing layers. A multilayer neural network is a neural network with more than one layer. A deep neural network is a multilayer neural network with many layers.

[0013] A tensor is a multidimensional array of numerical values. A tensor block is a contiguous subarray of the elements in a tensor.

[0014] Each neural network layer is associated with a parameter tensor V, weight tensor W, input data tensor X, output data tensor Y, and intermediate data tensor Z. The parameter tensor contains all of the parameters that control neuron activation functions σ in the layer. The weight tensor contains all of the weights that connect inputs to the layer. The input data tensor contains all of the data that the layer consumes as input. The output data tensor contains all of the data that the layer computes as output. The intermediate data tensor contains any data that the layer produces as intermediate computations, such as partial sums.

[0015] The data tensors (input, output, and intermediate) for a layer may be 3-dimensional, where the first two dimensions may be interpreted as encoding spatial location and the third dimension as encoding different features. For example, when a data tensor represents a color image, the first two dimensions encode vertical and horizontal coordinates within the image, and the third dimension encodes the color at each location. Every element of the input data tensor X can be connected to every neuron by a separate weight, so the weight tensor W generally has 6 dimensions, concatenating the 3 dimensions of the input data tensor (input row a, input column b, input feature c) with the 3 dimensions of the output data tensor (output row i, output column j, output feature k). The intermediate

data tensor Z has the same shape as the output data tensor Y. The parameter tensor V concatenates the 3 output data tensor dimensions with an additional dimension o that indexes the parameters of the activation function σ.

[0016] An element of a layer's output data tensor Y can be computed as in Equation 1 where the neuron activation function σ is configured by the vector of activation function parameters $V[i,j,k,:]$, and the weighted sum $Z[i,j,k]$ can be computed as in Equation 2.

$$Y[i,j,k] = \sigma(V[i,j,k,:];\ Z[i,j,k])$$

Equation 1

$$Z[i,j,k] = \sum_{a=1}^{A}\sum_{b=1}^{B}\sum_{c=1}^{C} W[i,j,k,a,b,c] \cdot X[a,b,c]$$

Equation 2

[0017] For simplicity of notation, the weighted sum in Equation 2 may be referred to as the output, which is equivalent to using a linear activation function $Y[i,j,k] = \sigma(Z[i,j,k]) = Z[i,j,k]$, with the understanding that the same statements apply without loss of generality when a different activation function is used.

[0018] In various embodiments, computation of the output data tensor as described above is decomposed into smaller problems. Each problem may then be solved on one or more neural core, or on one or more core of a conventional multicore system in parallel.

[0019] A convolution layer exploits spatially local correlations in natural sensor data by enforcing a local connectivity pattern between neurons in adjacent layers: each neuron receives connections from only a small region of the input data tensor. The extent of this connectivity is called the receptive field of the neuron. All neurons that compute elements of the same output feature share the same set of weights and activation function parameters, called a filter, with a local receptive field. The size of a filter is fixed, irrespective of input size, so a convolution layer has far fewer free parameters than a fully connected layer with the same input and output dimensions, and a correspondingly smaller memory footprint.

[0020] For example, in a spatial convolution layer, each filter's receptive field covers only a fraction of the height and width of the input data tensor, but extends through the full feature depth. Such an architecture ensures that the filters produce the strongest response to a spatially local input pattern. The layer computes its output by convolving each filter across the width and height of the input data tensor, computing the dot product between the entries of the filter and the input data tensor at each location to produce a 2-dimensional activation map for each filter.

[0021] Stacking the feature maps for all filters forms the full output data tensor for the convolution layer. Every element in the output data tensor can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares filter parameters with neurons in the same output feature map.

[0022] It will be appreciated that the size of a convolution layer's output data tensor is a function of the size of the input data tensor, the size of the filters, and the stride with which the filters are applied to the input data tensor. The stride parameters $stride_a$, $stride_b$, and $stride_c$ define the distance between each filter application, a value of 1 signifying that each filter application is shifted one pixel along the relevant dimension of the input data tensor. Larger stride values reduce the computational load by computing only a subset of the possible output pixels.

[0023] It will further be appreciated that in various convolutions, zero padding may be applied to the input data tensor in order to vary the output data tensor size relative to the input data tensor size, for example to make those sizes equal. Padding may be omitted in various examples for simplicity of explanation, but it will be apparent that padding may be included in various embodiments without departing from the scope of the present disclosure.

[0024] In a convolution layer, the 6-dimensional weight tensor contains many repeated blocks, since all elements of the same output feature share the same filter weights that are replicated at each output location. The shared filter weights can be described more compactly by a dense 4-dimensional filter tensor F that contains all of the filters that compute output features of the layer, and is indexed by the output feature dimension (output feature k) and 3 filter input dimensions (filter row r, filter column s, filter feature t).

$$Z[i,j,k] = \sum_{r=1}^{R}\sum_{s=1}^{S}\sum_{t=1}^{T} F[k,r,s,t] \cdot X[\alpha(i,r;\mathrm{A}), \beta(j,s;\mathrm{B}), \gamma(r,t;\Gamma)]$$

Equation 3

[0025] In Equation 3, the range of the r, s, t summations may be reduced for i, j, k coordinates near the edges of the output data tensor to accommodate boundary conditions, which are omitted here for simplicity. The accessor function $\alpha(i,r;\mathrm{A})$ computes the row coordinate a of the input data tensor element to be multiplied by filter weight $F[k,r,s,t]$ in order to compute output data element $Y[i,j,k]$, where A is a set of constants that parameterize the row coordinate calculation, such as striding parameter $stride_a$ and padding parameter pad_a. Analogous accessor functions $\beta(j,s;\mathrm{B})$ and $\gamma(r,t;\Gamma)$ compute the column and feature coordinates b and c.

[0026] In various embodiments, convolution as described above is decomposed into smaller problems. Each problem may then be solved on one or more neuromorphic core or on one or more core of a conventional multicore system in parallel.

[0027] In some embodiments, the input data tensor is decomposed along its row and column dimensions into blocks that extend the entire length of its feature dimension. In such embodiments, a neural core receives an input data tensor block and the whole filter tensor F. To correctly compute convolutions for pixels at boundaries of the input data tensor blocks, input data tensor blocks may be generated such that they overlap. Alternatively, partial sums from multiple neural cores may be summed together to arrive at a final result.

[0028] Neural Networks are useful for a variety of tasks, such as image classification. Low precision and/or constrained neural network hardware accelerators can implement such tasks using fewer bits of precision for neuron activations (values sent between layers) and parameters (such as neuron-to-neuron weights) compared to conventional computers (*e.g.*, CPUs and GPUs). This saves silicon area as well as compute energy without sacrificing task performance (such as classification accuracy).

[0029] However, reduced precision substrates face challenges such as transformation of inputs, which may be high precision, into a compatible low-precision representation. Such low-precision representations may require specific formats, such as vectors of inputs rather than scalars. In addition, hardware precision is fixed is fabrication time, yet different neural networks and different layers within a single neural network may benefit from varied levels of precision.

[0030] To address these and other shortcomings of alternative approaches, the present disclosure provides systems and methods enabling a low-precision hardware neural network substrate to receive high-precision data in various formats, transduce it into a compatible low-precision vector representation, and implement flexible-precision computations.

[0031] With reference now to **Fig. 1**, a neural core according to embodiments of the present disclosure is depicted. A neural core **100** is a tileable computational unit that computes one block of an output tensor. A neural core **100** has M inputs and N outputs. In various embodiments, $M = N$. To compute an output tensor block, a neural core multiplies an $M \times 1$ input data tensor block **101** with an $M \times N$ weight tensor block **102** and accumulates the products into weighted sums that are stored in a $1 \times N$ intermediate tensor block **103**. A $O \times N$ parameter tensor block contains the O parameters that specify each of the N neuron activation functions that are applied to the intermediate tensor block **103** to produce a $1 \times N$ output tensor block **105**.

[0032] Multiple neural cores may be tiled in a neural core array. In some embodiments, the array is 2-dimensional.

[0033] A neural network model is a set of constants that collectively specify the entire computation performed by a neural network, including the graph of connections between neurons as well as the weights and activation function parameters for every neuron. Training is the process of modifying the neural network model to perform a

desired function. Inference is the process of applying a neural network to an input to produce an output, without modifying the neural network model.

[0034] An inference processing unit is a category of processors that perform neural network inference. A neural inference chip is a specific physical instance of an inference processing unit.

[0035] Referring to **Fig. 2**, an exemplary Inference Processing Unit (IPU) is illustrated according to embodiments of the present disclosure. IPU **200** includes a memory **201** for the neural network model. As described above, the neural network model may include the synapse weights for a neural network to be computed. IPU **200** includes an activation memory **202**, which may be transient. Activation memory **202** may be divided into input and output regions, and stores neuron activations for processing. IPU **200** includes a neural computation unit **203**, which is loaded with a neural network model from model memory **201**. Input activations are provided from activation memory **202** in advance of each computation step. Outputs from neural computation unit **203** are written back to activation memory **202** for processing on the same or another neural computation unit.

[0036] In various embodiments a scheduler **204** is included in IPU **200**. In such embodiments, all operations in the IPU are directed by the scheduler. As set out below, central and/or distributed schedulers may be provided in various embodiments. A global scheduler may be referred to as a chip microengine, while a local scheduler may be referred to as a core microengine or local controller. In various embodiments a scheduler comprises one or more microengines, microcontrollers, state machines, CPUs, or other controllers.

[0037] Referring to **Fig. 3**, a multi-core Inference Processing Unit (IPU) is illustrated according to embodiments of the present disclosure. IPU **300** includes a model memory **301** for the neural network model. As described above, the neural network model may include the synapse weights for a neural network to be computed. IPU **300** includes an activation memory **302**, which may be transient. Activation memory **302** may be divided into input and output regions, and stores neuron activations for processing. In some embodiments, IPU **300** includes an instruction memory **307** for storing chip-level instructions.

[0038] IPU **300** includes a plurality of cores **303** in an array **305**. Each core **303** includes a neural computation unit **333**, which is loaded with a neural network model from model memory **301**. Each core also include a local activation memory **332**. Input activations are provided from local activation memory **332** in advance of each computation step. Outputs from neural computation unit **333** are written back to activation memory **332** for processing on the same or another neural computation unit.

[0039] In various embodiments a global scheduler **304** is included in IPU **300**. In various embodiments, a local core controller **334** is included on each core **303**. In such embodiments, the direction of operations is shared between the global scheduler (chip microengine) and the local core controller (core microengine). In some embodiments, a chip-level instruction memory **306** is provided for storing chip-level instruction for execution by chip

microengine **304**. In some embodiments, core **303** includes a core-level instruction memory **336** for storing core-level instructions for execution by core microengine **334**.

[0040] At **311**, compute instructions are loaded from instruction memory **301** to each core **303** by global scheduler **304**. At **312**, parameters (*e.g.*, neural network/synaptic weights) are loaded from model memory **301** to each core **303** by global scheduler **304**. At **313**, neural network activation data are loaded from data memory **302** to each core **303** by global scheduler **304**. At **314**, the cores in matrix **305** perform computation to generate output neuron activations. In particular, the computation comprises applying the input synaptic weights to the input activations. It will be appreciated that various methods are available for performing such computations, including *in silico* dendrites, as well as vector multiplication units. At **315**, the results from computation are stored in data memory **302**. These stages may be pipelined, in order to provide efficient usage of the neural computation unit on each core.

[0041] It will be appreciated that inputs and outputs may be transferred from local activation memory **332** to global activation memory **302** according to the requirements of a given neural network. Likewise, the model memory and instruction memory may have both chip-level and core-level components. At the core level, at **341**, compute instructions are loaded to local instruction memory **336**. At **342**, parameters (*e.g.*, neural network/synaptic weights) are loaded to model memory **331**. At **343**, neural network activation data are loaded to data memory **332**. At **344**, the computation unit **332** computed output activations or partial sums. At **345**, the results from computation are outputted from data memory **332**. These stages may be pipelined, in order to provide efficient usage of each core.

[0042] With reference now to **Fig. 4**, transformation is illustrated according to embodiments of the present disclosure. In various embodiments, transformation from high precision to low precision and/or from scalar to vector is provided. In particular, a transformation block can receive high-precision data in various formats and transduce it into a hardware-compatible low-precision representation. Likewise, a transformation block can receive various formats of scalar or vector data and transduce it into a hardware-compatible vector representation.

[0043] An input image (*e.g.*, **401**) is often 3-color channels (*e.g.*, RGB), each with 8-bits of precision (coding 0-255). If, for example, the hardware uses R bits to represent each activation channel (where $R < 8$), various approaches may be used to cast to R-bit precision.

- Each 8-bit value may be cast to an R-bit value by removing the $8 - R$ least significant bits (for example, in cases where all 8 bits of precision are not required).
- Each 8-bit value can be cast to $8/R$ R-bit values (*e.g.*, for $R = 2$, $8/2 = 4$ 2-bit values are used).
- Each 8-bit value can be cast to less than $8/R$ R-bit values (*e.g.*, for $R = 2$, 2 2-bit values may be used, dropping the 4 least significant bits).

[0044] An input image (*e.g.*, **401**) of size $W \times H \times C$ may be represented as $W \times H$ C-element vectors or $W \times H \times C$ scalars, where W is input width, H is input height, and C is the input number of channels or colors. If, for example, the hardware uses T-element vectors as inputs, there are several ways to cast to T-element vectors.

- If $T > C$, each C-element vector can be directly mapped to a T-element vector with $T - C$ zero entries, resulting in $W \times H$ T-element vectors.
- If $T > C$, up to T/C pixels can be packed into each T-element vector. For example, if $C = 3$ and $T = 24$, 8 pixels can be packed, for example into a 4×2 patch with a stride of 4 in width and 2 in height, resulting in $W/4 \times H/2$ T-element vectors.
- If $T > C$, up to T/C pixels can be packed into each T-element vector in a manner that optimizes for computation in the next layer. For example, if $C = 3$ and $T = 24$ and the next layer implements a 8×8 convolution with a stride of 4, 8 pixels can be packed, such as a 8×1 patch with a stride of 4, resulting in $W/4 \times H$ T-element vectors. Now the next layer need only implement a 1×8 convolution across the resulting input, optimizing hardware utilization.
- If $T < C$, each C-element vector can be unpacked into C/T T-element vectors, resulting in $W \times H \times C/T$ T-element vectors. For example, if $C = 4$ and $T = 2$, each 4-element vector can be unpacked into 2 2-element vectors, resulting in $W \times H \times 2$ 2-element vectors.

[0045] In an illustrative example of transformation, input image **401** is $256 \times 256 \times 3$, and the underlying hardware uses 4-bit activations and 32-element vectors. Input image **401** is distributed among a 4×4 array of cores **402**. To complete a convolution, each core requires a $(32 + 3) \times (32 + 3) \times 3$ portion of original image **401** for a $7 \times 7 \times 3$ filter. Using a multiple-row scratch pad memory **402**, the $1 \times 7 \times 3(\times 4)$ most significant bits are packed into a 21-element vector (of 4 bits per element), which fits within the 32-element platform vector **404**. As pictured, a column stride of 2 is used, resulting in some data duplication, but sizing the data for the convolutional layers.

[0046] The result of this transformation is that a $7 \times 7 \times 3$ filter becomes a $7 \times 1 \times 21$ filter. Vector utilization is $21/32 \approx 66\%$. The packed image tensor **405, 406** is 256×128×32, so each core (in 4×4 array **402**) receives a $32 \times 16 \times 32$ portion of the tensor (a tensorlet).

[0047] As pictured, the input blocks may be double buffered, alternately processing one tensor **405** while sending the prior tensor **406** into core array **402**.

[0048] It will be appreciated that the example given above is but one potential configuration of the input data tensor. For example, a swatch size of 4×7×1 gives vector utilization of $24.5/32 \approx 77\%$.

[0049] In various embodiments, flexible precision transformation is provided. In an exemplary neural network, each neural network layer computes low precision activations, each represented with R bits, using low precision weights, each represented by S bits. As described above, neurons in a layer receive inputs, X (with elements x_i), from one (or more) preceding layers. Each neuron computes its state, Y, based on the input and a set of weights, W (with elements w_i), applied to the inputs, where the result is added to a bias and run through a nonlinear activation function, σ. For example, to compute a single neuron activation, $Y = \sigma(b + \sum x_i w_i)$.

[0050] In the underlying hardware, R bits of precision are used to represent each output and input activation value, given by Y and x_i. S bits of precision are used to represent each weight value, given by w_i. For a given network structure, each layer may need a different minimum precision to maintain high performance (*e.g.*, classification accuracy). One approach is to build hardware with the maximum precision needed and use that precision for all layers. However, such an approach is inefficient in energy and area. Accordingly, the present disclosure provides an iterative approach that can implement any necessary precision by trading off time for precision.

[0051] To increase activation precision beyond R bits, a modified nonlinear activation function is used to first compute the most-significant R bits, zero them (such as by shifting), then compute the next most significant R bits, and repeat. To compute $R * M$ bits of precision would require M steps (which may be pipelined with subsequent operations). The limit of precision is set by the precision of the partial sum before the nonlinearity: $b + \sum x_i w_i$.

[0052] To increase weight precision beyond S bits, an iterative approach is applied. To compute $S * N$ bits of weight precision, N steps are used. For each step, the partial sum update is computed with S bits, starting with the most significant. After each update, the partial sum, P, is multiplied by (up to) 2^S (which can be implemented with a bit shift) to credit the more significant bits appropriately. When P is updated with the least significant S bits it is not multiplied by 2^S.

[0053] For example if $S = 1$ and $N = 4$, the $S * N$ bit precision weight w_i is stored and used as 4 parts: $w_{i_1} = \lfloor w_i / 2^{S*(N-1)} \rfloor$; $w_{i_2} = \lfloor (w_i - w_{i_1}) / 2^{S*(N-2)} \rfloor$; $w_{i_3} = \lfloor (w_i - w_{i_1} - w_{i_2}) / 2^{S*(N-3)} \rfloor$; $w_{i_4} = \lfloor (w_i - w_{i_1} - w_{i_2} - w_{i_3}) / 2^{S*(N-4)} \rfloor$.

[0054] The partial sum P is then aggregated as follows across the multiple partial sum computations:

$P = 0$

For $k = 1 : N - 1$ (3 steps in the above example)

$$P = P + \sum x_i w_{i_k}$$

$$P = P * 2^S$$

$$P = P + \sum x_i w_{i_k}$$

[0055] Accordingly, the present disclosure provides a transcoding circuit that takes in various scalar, vector, matrix, or tensor formats and outputs a tensor format compatible with target underlying neural network hardware. In various embodiments, high-precision data is transduced into low-precision data, compatible with the underlying neural network hardware. In some embodiments, scalar, vector, matrix, or tensor formatted input is transformed into a specific vector, matrix, or tensor format, compatible with the underlying neural network hardware. In various embodiments, flexible precision transformation is provided. In some embodiments, flexible precision transformation is implemented by iterating sequentially.

[0056] The present disclosure provides flexible-precision neural network computation by using one or more low-precision compute elements. In some embodiments, flexible precision is achieved with low-precision elements by iterating sequentially.

[0057] Referring now to **Fig. 5**, a method of operating a neural network is illustrated according to embodiments of the present disclosure. At **501**, an input data tensor is received at a neural network processor comprising a plurality of neural cores. The input data tensor has feature dimensions at an input bit precision. The neural network processor is configured for one or more processor feature dimensions at one or more processor bit precisions. At **502**, the input data tensor is transformed from the input bit precision to one of the processor bit precisions. At **503**, the input data tensor is divided into a plurality of blocks, each block conforming to one of the processor feature dimensions. At **504**, each of the plurality of blocks is provided to one of the plurality of neural cores. At **505**, the plurality of neural cores computes output of one or more neural network layers.

[0058] Referring now to **Fig. 6**, a schematic of an example of a computing node is shown. Computing node **10** is only one example of a suitable computing node and is not intended to suggest any limitation as to the scope of use or functionality of embodiments described herein. Regardless, computing node **10** is capable of being implemented and/or performing any of the functionality set forth hereinabove.

[0059] In computing node **10** there is a computer system/server **12**, which is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well-known computing systems, environments, and/or configurations that may be suitable for use with computer system/server **12** include, but are not limited to, personal computer systems, server computer systems, thin clients, thick clients, handheld or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputer systems, mainframe computer systems, and distributed cloud computing environments that include any of the above systems or devices, and the like.

[0060] Computer system/server **12** may be described in the general context of computer system-executable instructions, such as program modules, being executed by a computer system. Generally, program modules may include routines, programs, objects, components, logic, data structures, and so on that perform particular tasks or implement particular abstract data types. Computer system/server **12** may be practiced in distributed cloud computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed cloud computing environment, program modules may be located in both local and remote computer system storage media including memory storage devices.

[0061] As shown in **Fig. 6**, computer system/server **12** in computing node **10** is shown in the form of a general-purpose computing device. The components of computer system/server **12** may include, but are not limited to, one or more processors or processing units **16**, a system memory **28**, and a bus **18** that couples various system components including system memory **28** to processor **16**.

[0062] Bus **18** represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, Peripheral Component Interconnect (PCI) bus, Peripheral Component Interconnect Express (PCIe), and Advanced Microcontroller Bus Architecture (AMBA).

[0063] Computer system/server **12** typically includes a variety of computer system readable media. Such media may be any available media that is accessible by computer system/server **12**, and it includes both volatile and non-volatile media, removable and non-removable media.

[0064] System memory **28** can include computer system readable media in the form of volatile memory, such as random access memory (RAM) **30** and/or cache memory **32**. Computer system/server **12** may further include other removable/non-removable, volatile/non-volatile computer system storage media. By way of example only, storage system **34** can be provided for reading from and writing to a non-removable, non-volatile magnetic media (not shown and typically called a "hard drive"). Although not shown, a magnetic disk drive for reading from and writing to a removable, non-volatile magnetic disk (*e.g.*, a "floppy disk"), and an optical disk drive for reading from or writing to a removable, non-volatile optical disk such as a CD-ROM, DVD-ROM or other optical media can be provided. In such instances, each can be connected to bus **18** by one or more data media interfaces. As will be further depicted and described below, memory **28** may include at least one program product having a set (*e.g.*, at least one) of program modules that are configured to carry out the functions of embodiments of the disclosure.

[0065] Program/utility **40**, having a set (at least one) of program modules **42**, may be stored in memory **28** by way of example, and not limitation, as well as an operating system, one or more application programs, other program

modules, and program data. Each of the operating system, one or more application programs, other program modules, and program data or some combination thereof, may include an implementation of a networking environment. Program modules **42** generally carry out the functions and/or methodologies of embodiments as described herein.

[0066] Computer system/server **12** may also communicate with one or more external devices **14** such as a keyboard, a pointing device, a display **24**, etc.; one or more devices that enable a user to interact with computer system/server **12**; and/or any devices (*e.g.*, network card, modem, etc.) that enable computer system/server **12** to communicate with one or more other computing devices. Such communication can occur via Input/Output (I/O) interfaces **22**. Still yet, computer system/server **12** can communicate with one or more networks such as a local area network (LAN), a general wide area network (WAN), and/or a public network (*e.g.*, the Internet) via network adapter **20**. As depicted, network adapter **20** communicates with the other components of computer system/server **12** via bus **18**. It should be understood that although not shown, other hardware and/or software components could be used in conjunction with computer system/server **12**. Examples, include, but are not limited to: microcode, device drivers, redundant processing units, external disk drive arrays, RAID systems, tape drives, and data archival storage systems, etc.

[0067] The present disclosure may be embodied as a system, a method, and/or a computer program product. The computer program product may include a computer readable storage medium (or media) having computer readable program instructions thereon for causing a processor to carry out aspects of the present disclosure.

[0068] The computer readable storage medium can be a tangible device that can retain and store instructions for use by an instruction execution device. The computer readable storage medium may be, for example, but is not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing. A non-exhaustive list of more specific examples of the computer readable storage medium includes the following: a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), a static random access memory (SRAM), a portable compact disc read-only memory (CD-ROM), a digital versatile disk (DVD), a memory stick, a floppy disk, a mechanically encoded device such as punch-cards or raised structures in a groove having instructions recorded thereon, and any suitable combination of the foregoing. A computer readable storage medium, as used herein, is not to be construed as being transitory signals per se, such as radio waves or other freely propagating electromagnetic waves, electromagnetic waves propagating through a waveguide or other transmission media (*e.g.*, light pulses passing through a fiber-optic cable), or electrical signals transmitted through a wire.

[0069] Computer readable program instructions described herein can be downloaded to respective computing/processing devices from a computer readable storage medium or to an external computer or external

storage device via a network, for example, the Internet, a local area network, a wide area network and/or a wireless network. The network may comprise copper transmission cables, optical transmission fibers, wireless transmission, routers, firewalls, switches, gateway computers and/or edge servers. A network adapter card or network interface in each computing/processing device receives computer readable program instructions from the network and forwards the computer readable program instructions for storage in a computer readable storage medium within the respective computing/processing device.

[0070] Computer readable program instructions for carrying out operations of the present disclosure may be assembler instructions, instruction-set-architecture (ISA) instructions, machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, or either source code or object code written in any combination of one or more programming languages, including an object oriented programming language such as Smalltalk, C++ or the like, and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The computer readable program instructions may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider). In some embodiments, electronic circuitry including, for example, programmable logic circuitry, field-programmable gate arrays (FPGA), or programmable logic arrays (PLA) may execute the computer readable program instructions by utilizing state information of the computer readable program instructions to personalize the electronic circuitry, in order to perform aspects of the present disclosure.

[0071] Aspects of the present disclosure are described herein with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems), and computer program products according to embodiments of the disclosure. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer readable program instructions.

[0072] These computer readable program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks. These computer readable program instructions may also be stored in a computer readable storage medium that can direct a computer, a programmable data processing apparatus, and/or other devices to function in a particular manner, such that the computer readable storage medium having instructions stored therein comprises

an article of manufacture including instructions which implement aspects of the function/act specified in the flowchart and/or block diagram block or blocks.

[0073] The computer readable program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other device to cause a series of operational steps to be performed on the computer, other programmable apparatus or other device to produce a computer implemented process, such that the instructions which execute on the computer, other programmable apparatus, or other device implement the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0074] The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods, and computer program products according to various embodiments of the present disclosure. In this regard, each block in the flowchart or block diagrams may represent a module, segment, tensorlet, block, or portion of instructions, which comprises one or more executable instructions for implementing the specified logical function(s). In some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts or carry out combinations of special purpose hardware and computer instructions.

[0075] 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 comprising:

receiving an input data tensor at a neural network processor comprising a plurality of neural cores, the input data tensor having feature dimensions at an input bit precision,

the neural network processor being configured for one or more processor feature dimensions at one or more processor bit precisions;

transforming the input data tensor from the input bit precision to one of the processor bit precisions; dividing the input data tensor into a plurality of blocks, each block conforming to one of the processor feature dimensions;

providing each of the plurality of blocks to one of the plurality of neural cores;

computing, by the plurality of neural cores, output of one or more neural network layers.

2. The method of claim 1, wherein the input data tensor comprises an image.

3. The method of claim 1, wherein transforming the input data tensor comprises removing least significant bits.

4. The method of claim 1, the neural network processor being configured for a predetermined number of features, wherein transforming the input data tensor comprises dividing input features into a plurality of feature sets, each having less than or equal to the predetermined number of features .

5. The method of claim 1, wherein dividing the input data tensor comprises zero-padding the plurality of blocks in one of the feature dimensions to conform with one of the processor feature dimensions.

6. The method of claim 1, wherein dividing the input data tensor comprises packing the input data tensor.

7. The method of claim 6, wherein packing the input data tensor comprises:

reorganizing input features to load unused processor feature dimensions with data from non-feature dimensions of the input features.

8. The method of claim 1, wherein the neural network processor is configured to:

compute a plurality of fixed precision partial sums; and

combine the plurality of fixed precision partial sums into complete sums.

9. The method of claim 8, wherein the plurality of fixed precision partial sums are intermediate results.

10. The method of claim 9, wherein the intermediate results are weighted sums of a subset of inputs.

11. The method of claim 8, wherein the neural network processor is configured to iteratively compute a partial sum from the plurality of fixed precision partial sums.

12. A system comprising:

a neural network processor comprising a plurality of neural cores, the neural network processor having one or more processor precisions per activation, the processor being configured to accept data having a processor feature dimension;

a transformation circuit coupled to the neural network processor, and adapted to:

receive an input data tensor having an input precision per channel at one or more features;

transform the input data tensor from the input precision to the processor precision;

divide the input data tensor into a plurality of blocks, each block conforming to one of the processor feature dimensions;

provide each of the plurality of blocks to one of the plurality of neural cores; and wherein

the neural network processor is adapted to compute, by the plurality of neural cores, output of one or more neural network layers.

13. The system of claim 12, wherein the input data tensor comprises an image.

14. The system of claim 12, wherein transforming the input data tensor comprises removing least significant bits.

15. The system of claim 12, wherein transforming the input data tensor comprises dividing each channel into a plurality of values having less than or equal to the processor precision.

16. The system of claim 12, wherein dividing the input data tensor comprises zero-padding the plurality of blocks in a feature dimension to conform with the processor feature dimension.

17. The system of claim 12, wherein dividing the input data tensor comprises packing the input data tensor.

18. The wherein of claim 17, wherein packing the input data tensor comprises:

reorganizing input features to load unused processor feature dimensions with data from non-feature dimensions of the input features.

19. The system of claim 12, wherein the neural network processor is configured to:

compute a plurality of fixed precision partial sums; and

combine the plurality of fixed precision partial sums into complete sums.

20. The system of claim 19, wherein the plurality of fixed precision partial sums are intermediate results.

21. The system of claim 20, wherein the intermediate results are weighted sums of a subset of inputs.

22. The system of claim 19, wherein the neural network processor is configured to iteratively compute a partial sum from the plurality of fixed precision partial sums.

23. A system comprising:

a neural network processor comprising a plurality of neural cores, the neural network processor being configured for one or more processor feature dimensions at one or more processor bit precisions; and

a transformation circuit coupled to the neural network processor, and adapted to:

receive an input data tensor having feature dimensions at an input bit precision;

transform the input data tensor from the input bit precision to one of the processor bit precisions;

divide the input data tensor into a plurality of blocks, each block conforming to one of the processor feature dimensions;

provide each of the plurality of blocks to one of the plurality of neural cores; and wherein

the plurality of neural cores is adapted to compute the output of one or more neural network layers.

24. A method comprising:

receiving an input data tensor,

the input data tensor having an input precision per channel at one or more features;

transforming the input data tensor from the input precision to a processor precision of a neural network processor,

the neural network processor being configured to accept data having a processor feature dimension;

dividing the input data tensor into a plurality of blocks, each block conforming to one of the processor feature dimensions;

providing each of the plurality of blocks to one of the plurality of neural cores, and wherein

the neural network processor is adapted to compute, by the plurality of neural cores, output of one or more neural network layers.

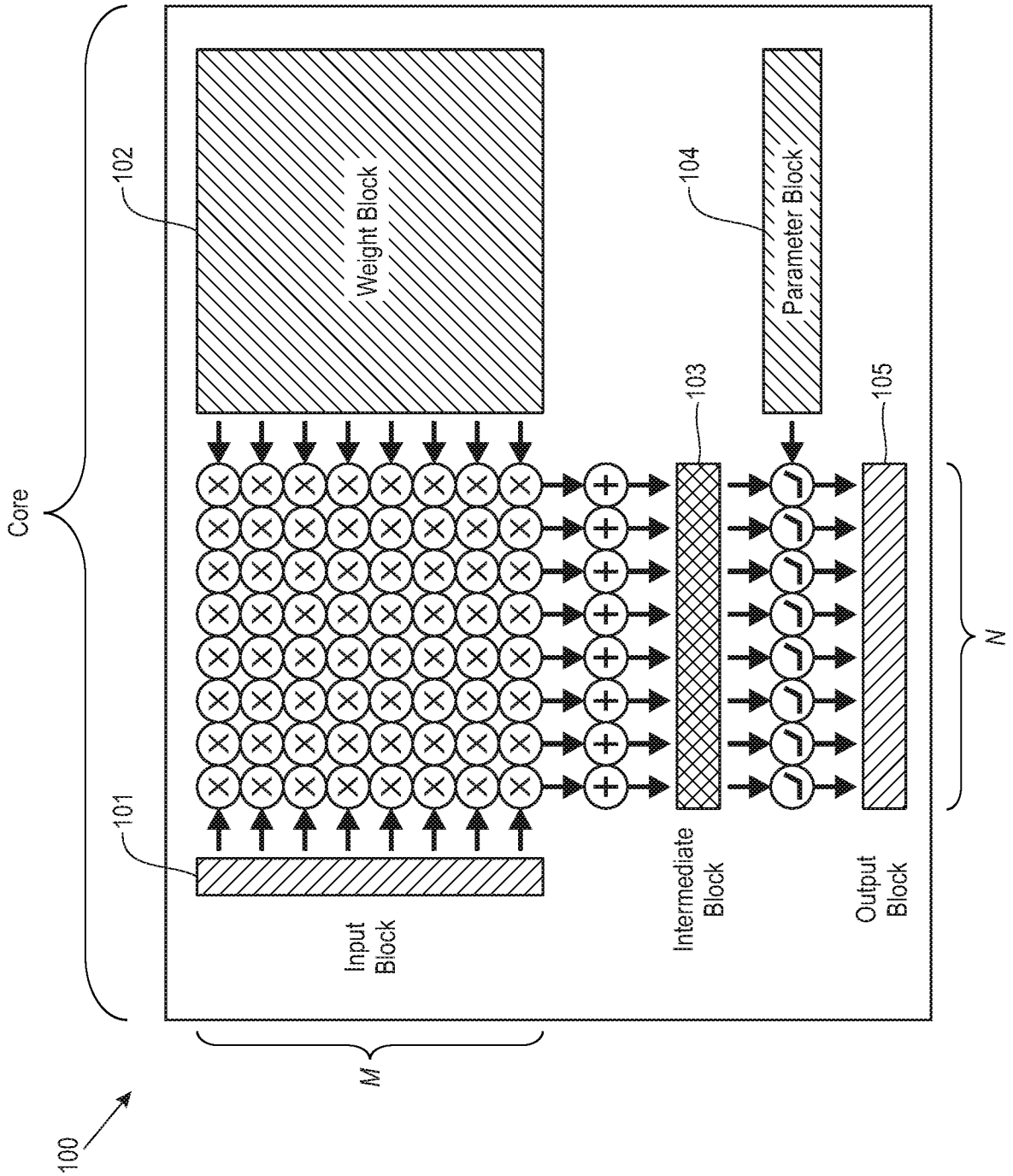
100
Core
101
Input Block
102
Weight Block
M
103
Intermediate Block
104
Parameter Block
105
Output Block
N

FIG. 1

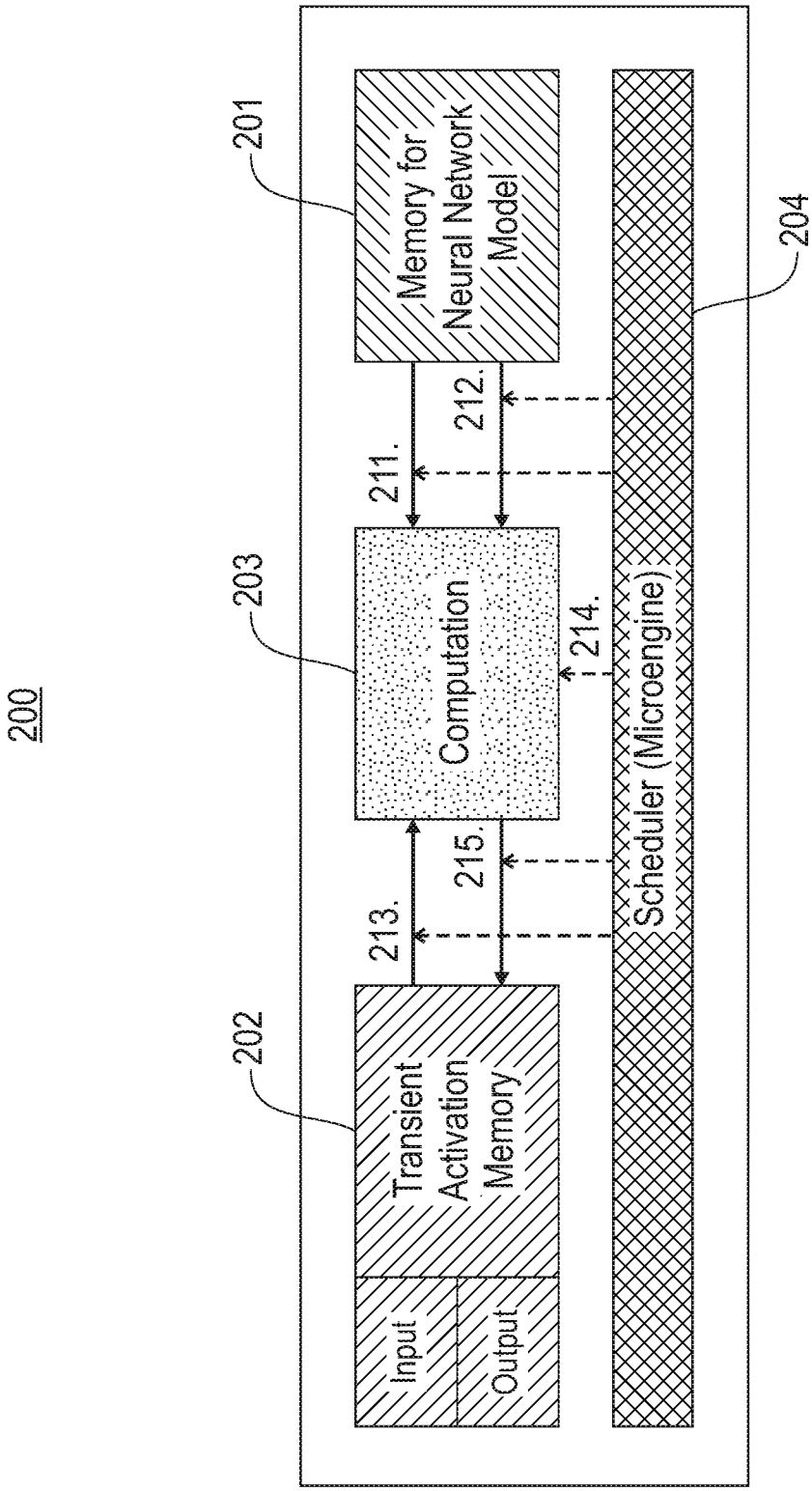
200
201
Memory for Neural Network Model
202
Transient Activation Memory
Input
Output
203
Computation
204
Scheduler (Microengine)
211.
212.
213.
214.
215.

FIG. 2

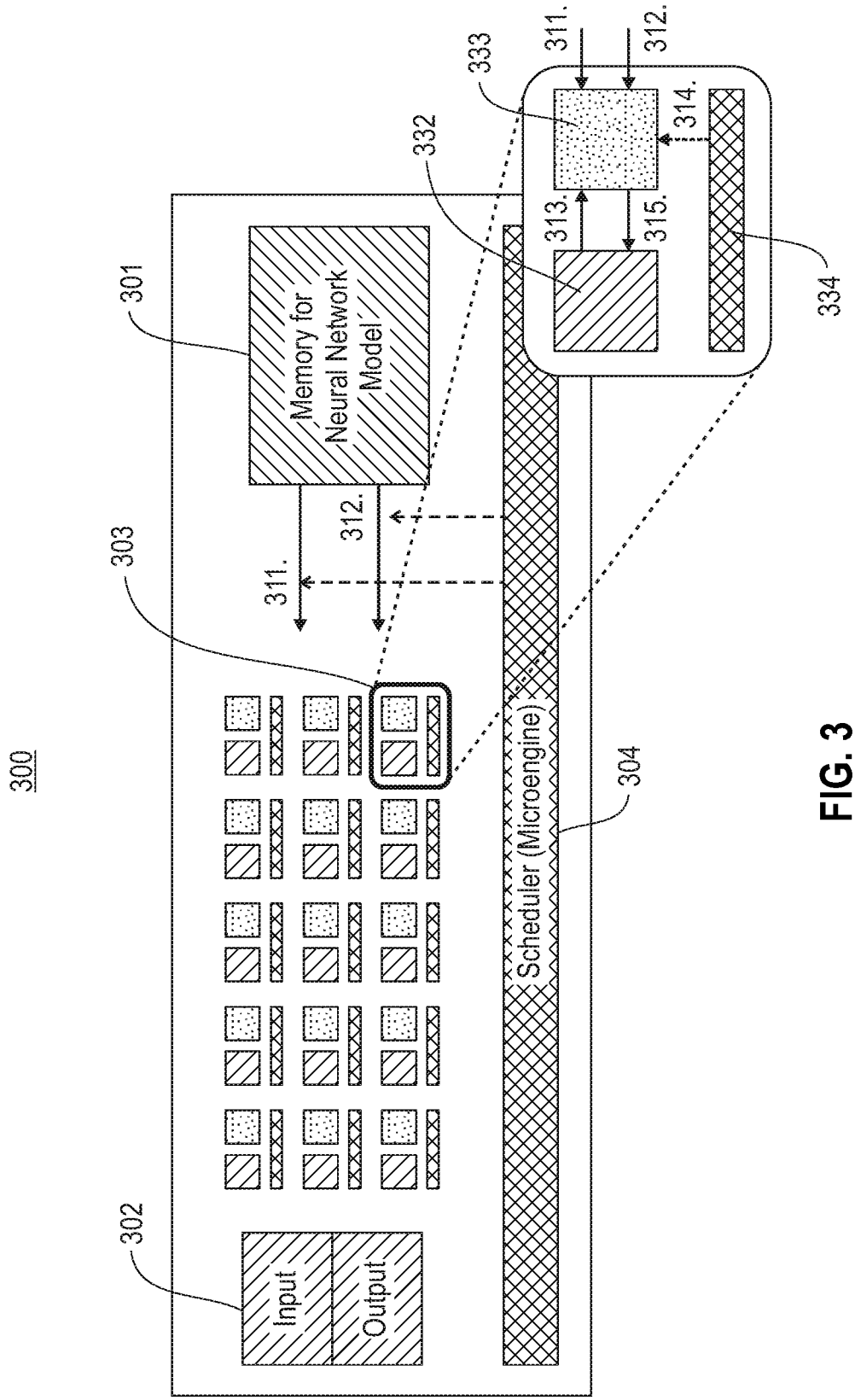
300
301
Memory for Neural Network Model
302
Input
Output
303
311.
312.
Scheduler (Microengine)
304
332
333
311.
312.
313.
314.
315.
334

FIG. 3

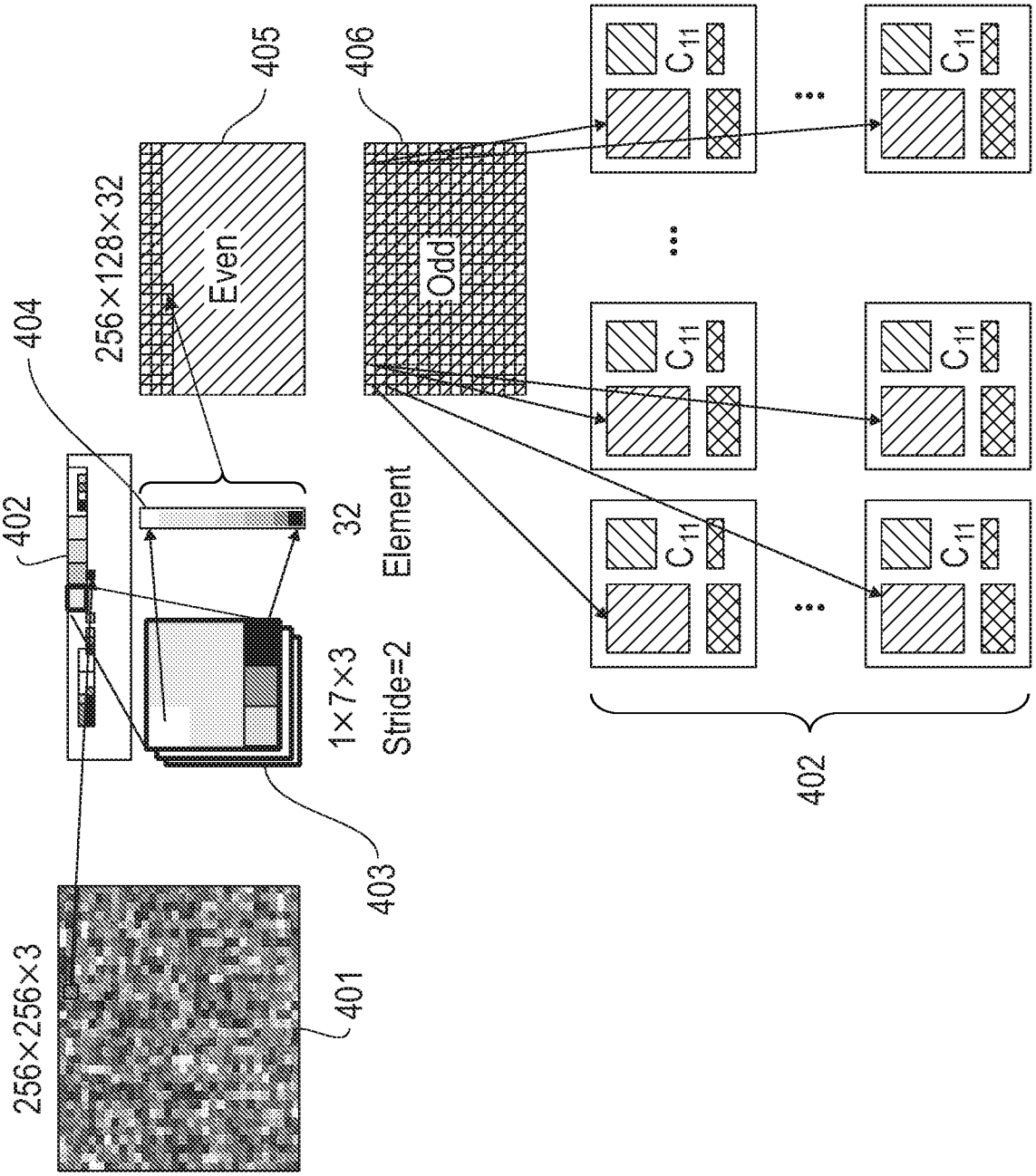
256×256×3
401
402
403
404
1×7×3
Stride=2
32
Element
256×128×32
Even
405
Odd
406
C11
402

FIG. 4

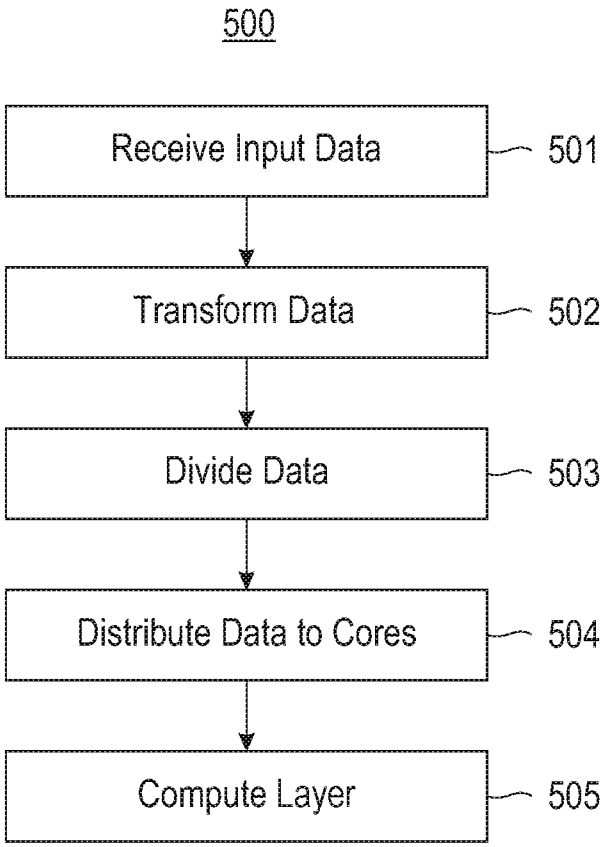
500
Receive Input Data
501
Transform Data
502
Divide Data
503
Distribute Data to Cores
504
Compute Layer
505

FIG. 5

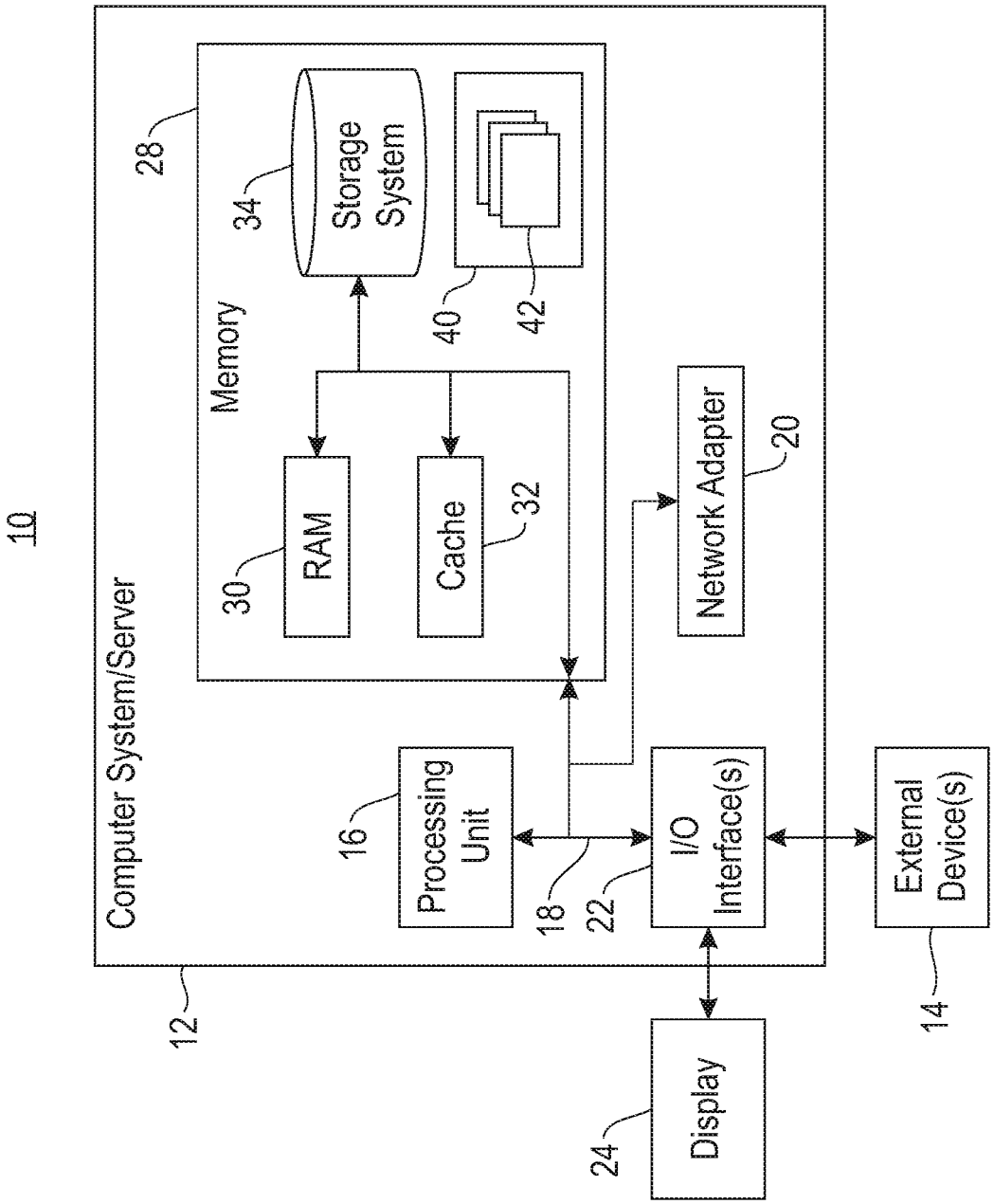
10
Computer System/Server
12
28
Memory
34
Storage System
40
42
30
RAM
Cache
32
Network Adapter
20
16
Processing Unit
18
22
I/O Interface(s)
External Device(s)
14
24
Display

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB2019/058120

A. CLASSIFICATION OF SUBJECT MATTER

G06N 3/06(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)

G06N,G06F,G06K

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)

WPI,EPODOC,CNPAT,CNKI,IEEE: neural, network, core, multi, data, representation, processor, tensor, image, picture, input, output, divide, block, dimension, transform, precision, layer

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 107633010 A (CENTRAL SOUTH UNIVERSITY) 26 January 2018 (2018-01-26) description, paragraphs [0039]-[0106], and figures 1-5	1-24
A	CN 104641385 A (INTERNATIONAL BUSINESS MACHINES CORPORATION) 20 May 2015 (2015-05-20) the whole document	1-24
A	US 2016155047 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 02 June 2016 (2016-06-02) the whole document	1-24
A	JP H09128428 A (NEC CORPORATION) 16 May 1997 (1997-05-16) the whole document	1-24
A	WO 2017168275 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION et al.) 05 October 2017 (2017-10-05) the whole document	1-24

☐ 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	Date of mailing of the international search report
29 November 2019	**30 December 2019**
Name and mailing address of the ISA/CN **National Intellectual Property Administration, PRC** **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing** **100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer **WANG,Yang** Telephone No. **(86-10)53961523**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/IB2019/058120

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	107633010	A	26 January 2018	None			
CN	104641385	A	20 May 2015	US	2015379393	A1	31 December 2015
				JP	2015534172	A	26 November 2015
				DE	112013003349	T5	19 March 2015
				US	2015254551	A1	10 September 2015
				WO	2014041443	A1	20 March 2014
US	2016155047	A1	02 June 2016	US	2015242743	A1	27 August 2015
				US	2015262059	A1	17 September 2015
				US	2014032464	A1	30 January 2014
JP	H09128428	A	16 May 1997	None			
WO	2017168275	A1	05 October 2017	GB	2557780	A	27 June 2018
				DE	112017000855	T5	25 October 2018
				JP	2019515367	A	06 June 2019
				US	2017286825	A1	05 October 2017