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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 November 2016 (24.11.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/186823 A1

(51) International Patent Classification:
G06N 3/08 (2006.01) *G06F 15/80* (2006.01)

(21) International Application Number:
PCT/US2016/030515

(22) International Filing Date:
3 May 2016 (03.05.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/165,020 21 May 2015 (21.05.2015) US
14/844,431 3 September 2015 (03.09.2015) US

(71) Applicant: **GOOGLE INC.** [US/US]; 1600 Amphitheatre Parkway, Mountain View, California 94043 (US).

(72) Inventor: **YOUNG, Reginald Clifford**; 1600 Amphitheatre Parkway, Mountain View, California 94043 (US).

(74) Agents: **HSIEH, I-Wei** et al.; Fish & Richardson P.C., P.O. Box 1022, Minneapolis, Minnesota 55440-1022 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, 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).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

WO 2016/186823 A1

(54) **Title**: BATCH PROCESSING IN A NEURAL NETWORK PROCESSOR

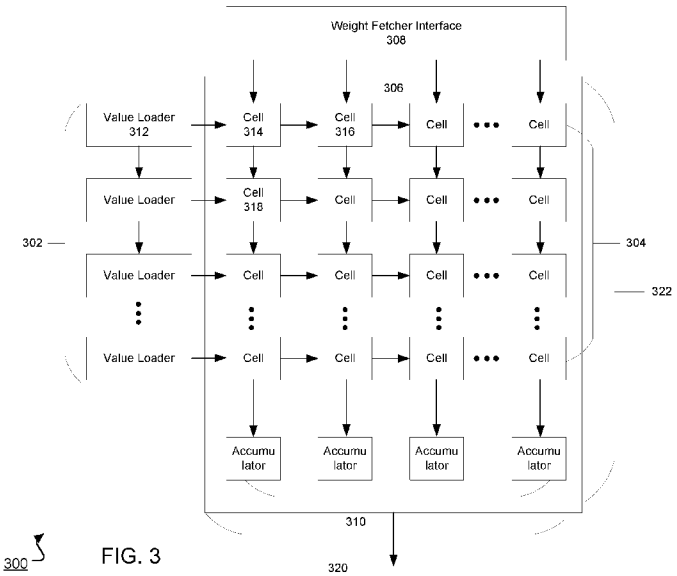

FIG. 3

(57) **Abstract**: Methods, systems, and apparatus, including computer programs encoded on computer storage media, for generating a respective neural network output for each of a plurality of inputs, the method comprising, for each of the neural network layers: receiving a plurality of inputs to be processed at the neural network layer; forming one or more batches of inputs from the plurality of inputs, each batch having a number of inputs up to the respective batch size for the neural network layer; selecting a number of the one or more batches of inputs to process, where a count of the inputs in the number of the one or more batches is greater than or equal to the respective associated batch size of a subsequent layer in the sequence; and processing the number of the one or more batches of inputs to generate the respective neural network layer output.

BATCH PROCESSING IN A NEURAL NETWORK PROCESSOR

BACKGROUND

This specification relates to computing neural network inferences in hardware.

Neural networks are machine learning models that employ one or more layers of neurons to generate an output, e.g., a classification, for a received input. Some neural networks include one or more hidden layers in addition to an output layer. The output of each hidden layer is used as input to the next layer in the network, i.e., the next hidden layer or the output layer of the network. Each layer of the network generates an output from a received input in accordance with current values of a respective set of parameters.

Traditionally, some neural network systems compute inferences serially. That is, when computing inferences for multiple inputs, the neural network system can process each input through each of the layers of the neural network to generate the output for the input before processing the next input.

SUMMARY

In general, this specification describes a special-purpose hardware circuit that computes neural network inferences.

In general, one innovative aspect of the subject matter described in this specification can be embodied in methods that include the actions of generating a respective neural network output for each of a plurality of inputs, wherein the generating comprises processing each input through each of a plurality of neural network layers to generate the respective neural network output for the input, wherein the neural network layers are arranged in a sequence, and wherein each neural network layer has a respective batch size, the method comprising, for each of the neural network layers: receiving a plurality of inputs to be processed at the neural network layer; forming one or more batches of inputs from the plurality of inputs, each batch having a number of inputs up to the respective batch size for the neural network layer; selecting a number of the one or more batches of inputs to process, where a count of the inputs in the number of the one or more batches is greater than or equal to the respective associated batch size of a subsequent layer in the sequence; and processing the number of the one or more batches of inputs to generate the respective neural network layer output.

Implementations can include one or more of the following features. The respective batch size is based at least on a weight reuse value, the weight reuse value

representing a number of times that weight inputs need to be reused for a compute time of output values using the weight inputs at a matrix computation unit to be longer than a load time of the weight inputs from memory. Where the weight reuse value is based at least on a clock rate of the memory storing the weight inputs. Each batch size is based at least on the weight reuse value divided by a number of times that weight inputs for the respective layer are reused. The plurality of neural network layers is processed at a matrix processing unit, where processing the number of the one or more batches of inputs comprises computing accumulated values for each input using the matrix computation unit. The weight reuse value is based on a number of arithmetic units inside the matrix computation unit. Each input corresponds to a distinct image resource. Forming a batch from the one or more layer outputs for processing at the subsequent layer. Generating, for each output, a corresponding inference.

Particular embodiments of the subject matter described in this specification can be implemented so as to realize one or more of the following advantages. The special purpose hardware circuit can efficiently perform the computation for a neural network having multiple layers by reusing weight inputs for a given layer on multiple independent inputs. In particular, the weight inputs are reused a number of times such that a compute time of reusing the weight inputs multiple times is greater than a fetch time of accessing new weight inputs from memory, thereby maximizing throughput in the circuit and avoiding stalling of the circuit. The circuit can efficiently perform the computation even if weight inputs are reused a different number of times at each layer.

The details of one or more embodiments of the subject matter of this specification are set forth in the accompanying drawings and the description below. Other features, aspects, and advantages of the subject matter will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a flow diagram of an example method for performing a computation for a given layer of a neural network.

FIG. 2 shows an example neural network processing system.

FIG. 3 shows an example architecture including a matrix computation unit.

FIG. 4 shows an example architecture of a cell inside a systolic array.

FIG. 5 is a flow diagram of an example method for performing neural network computations for multiple layers.

FIG. 6 shows an example neural network having multiple layers and batch sizes for each layer.

Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

A neural network having multiple layers can be used to compute inferences. For example, given an input, the neural network can compute an inference for the input. The neural network computes this inference by processing the input through each of the layers of the neural network. In particular, the layers of the neural network can be arranged in a sequence, each with a respective set of weights. Each layer receives an input and processes the input in accordance with the set of weights for the layer to generate an output. The output can be used as an input at the next neural network layer.

Therefore, in order to compute an inference from a received input, the neural network receives the input and processes it through each of the neural network layers in the sequence to generate the inference, with the output from one neural network layer being provided as input to the next neural network layer. Data inputs to a neural network layer, e.g., either the input to the neural network or the outputs of the layer below the layer in the sequence, to a neural network layer can be referred to as activation inputs to the layer.

In some implementations, the layers of the neural network are arranged in a directed graph structure. That is, any particular layer can receive multiple inputs, multiple outputs, or both. The layers of the neural network can also be arranged such that an output of a layer can be sent back as an input to a previous layer.

FIG. 1 is a flow diagram of an example process 100 for performing a computation for a given layer of a neural network using a special-purpose hardware circuit. For convenience, the method 100 will be described with respect to a system having one or more circuits that performs the method 100.The method 100 can be performed for each layer of the neural network in order to compute an inference from a received input.

The system receives sets of weight inputs (step 102) and sets of activation inputs (step 104) for the given layer. The sets of weight inputs and the sets of activation inputs can be received from dynamic memory and a unified buffer, respectively, of the special-purpose hardware circuit. In some implementations, both the sets of weight inputs and the sets of activation inputs can be received from the unified buffer.

The system generates accumulated values from the weight inputs and the activation inputs using a matrix multiplication unit of the special-purpose hardware circuit (step 106). In some implementations, the accumulated values are dot products of the sets of weight inputs and the sets of activation inputs. That is, for one set of weights, the system can multiply each weight input with each activation input and sum the products together to form an accumulated value. The system can then compute dot products of other set of weights with other sets of activation inputs.

The system can generate a layer output from the accumulation values (step 108) using a vector computation unit of the special-purpose hardware circuit. In some implementations, the vector computation unit applies an activation function to the accumulated values. The output of the layer can be stored in the unified buffer for use as an input to a subsequent layer in the neural network or can be used to determine the inference. The system finishes processing the neural network when a received input has been processed through each layer of the neural network to generate the inference for the received input.

FIG. 2 shows an example special-purpose integrated circuit 200 for performing neural network computations. The system 200 includes a host interface 202. The host interface 202 can receive instructions that include configuration information for a neural network computation. The configuration information can include at least one or more of the following: how many layers should be processed, corresponding sets of weight inputs for each layer of the layer, an initial set of activation inputs, i.e., the input to the neural network from which the inference is to be computed, corresponding input and output sizes of each layer, a stride value for the neural network computation, and a type of layer to be processed, e.g., a convolutional layer or a fully connected layer.

The host interface 202 can send the instructions to a sequencer 206, which converts the instructions into low level control signals that control the circuit to perform the neural network computations. In some implementations, the control signals regulate dataflow in the circuit, e.g., how the sets of weight inputs and the sets of activation inputs flow through the circuit. The sequencer 206 can send the control signals to a unified buffer 208, a matrix computation unit 212, and a vector computation unit 214. In some implementations, the sequencer 206 also sends control signals to a direct memory access engine 204 and dynamic memory 210. In some implementations, the sequencer 206 is a processor that generates clock signals. The sequencer 206 can use timing of the clock signals to, at appropriate times, send the control signals to each component of the circuit

200. In some other implementations, the host interface 202 passes in a clock signal from an external processor.

The host interface 202 can send the sets of weight inputs and the initial set of activation inputs to the direct memory access engine 204. The direct memory access engine 204 can store the sets of activation inputs at the unified buffer 208. In some implementations, the direct memory access stores the sets of weights to dynamic memory 210, which can be a memory unit. In some implementations, the dynamic memory is located off of the circuit.

The unified buffer 208 is a memory buffer. It can be used to store the set of activation inputs from the direct memory access engine 204 and outputs of the vector computation unit 214. The direct memory access engine 204 can also read the outputs of the vector computation unit 214 from the unified buffer 208.

The dynamic memory 210 and the unified buffer 208 can send the sets of weight inputs and the sets of activation inputs, respectively, to the matrix computation unit 212. In some implementations, the matrix computation unit 212 is a two-dimensional systolic array. The matrix computation unit 212 can also be a one-dimensional systolic array or other circuitry that can perform mathematical operations, e.g., multiplication and addition. In some implementations, the matrix computation unit 212 is a general purpose matrix processor.

The matrix computation unit 212 can process the weight inputs and the activation inputs and provide a vector of outputs to the vector computation unit 214. In some implementations, the matrix computation unit sends the vector of outputs to the unified buffer 208, which sends the vector of outputs to the vector computation unit 214. The vector computation unit can process the vector of outputs and store a vector of processed outputs to the unified buffer 208. For example, the vector computation unit 214 can apply a non-linear function to outputs of the matrix computation unit, e.g., a vector of accumulated values, to generate activated values. In some implementations, the vector computation unit 214 generates normalized values, pooled values, or both. The vector of processed outputs can be used as activation inputs to the matrix computation unit 212, e.g., for use in a subsequent layer in the neural network. The matrix computation unit 212 will be described in more detail below with reference to FIG. 3 and FIG. 4.

FIG. 3 shows an example architecture 300 including a matrix computation unit. The matrix computation unit is a two-dimensional systolic array 306. The array 306 includes multiple cells 304. In some implementations, a first dimension 320 of the

systolic array 306 corresponds to columns of cells and a second dimension 322 of the systolic array 306 corresponds to rows of cells. The systolic array can have more rows than columns, more columns than rows, or an equal number of columns and rows.

In the illustrated example, value loaders 302 send activation inputs to rows of the array 306 and a weight fetcher interface 308 sends weight inputs to columns of the array 306. In some other implementations, however, activation inputs are transferred to the columns and weight inputs are transferred to the rows of the array 306.

The value loaders 302 can receive the activation inputs from a unified buffer, e.g., the unified buffer 208 of FIG. 2. Each value loader can send a corresponding activation input to a distinct left-most cell of the array 306. The left-most cell can be a cell along a left-most column of the array 306. For example, value loader 312 can send an activation input to cell 314. The value loader can also send the activation input to an adjacent value loader, and the activation input can be used at another left-most cell of the array 306. This allows activation inputs to be shifted for use in another particular cell of the array 306.

The weight fetcher interface 308 can receive the weight input from a memory unit, e.g., the dynamic memory 210 of FIG. 2. The weight fetcher interface 308 can send a corresponding weight input to a distinct top-most cell of the array 306. The top-most cell can be a cell along a top-most row of the array 306. For example, the weight fetcher interface 308 can send weight inputs to cells 314 and 316.

In some implementations, a host interface, e.g., the host interface 202 of FIG. 2, shifts activation inputs throughout the array 306 along one dimension, e.g., to the right, while shifting weight inputs throughout the array 306 along another dimension, e.g., to the bottom. For example, over one clock cycle, the activation input at cell 314 can shift to an activation register in cell 316, which is to the right of cell 314. Similarly, the weight input at cell 316 can shift to a weight register at cell 318, which is below cell 314.

On each clock cycle, each cell can process a given weight input and a given activation input to generate an accumulated output. The accumulated output can also be passed to an adjacent cell along the same dimension as the given weight input. An individual cell is described further below with reference FIG. 4.

The accumulated output can be passed along the same column as the weight input, e.g., towards the bottom of the column in the array 306. In some implementations, at the bottom of each column, the array 306 can include accumulator units 310 that store and accumulate each accumulated output from each column when performing calculations

with layers having more weight inputs than columns or layers having more activation inputs than rows. In some implementations, each accumulator unit stores multiple parallel accumulations. This will be described further below with reference to FIG. 6. The accumulator units 310 can accumulate each accumulated output to generate a final accumulated value. The final accumulated value can be transferred to a vector computation unit. In some other implementations, the accumulator units 310 passes the accumulated values to the vector computation unit without performing any accumulations when processing layers with fewer weight inputs than columns or layers having fewer activating inputs than rows.

FIG. 4 shows an example architecture 400 of a cell inside a systolic array, e.g., the systolic array 306 of FIG. 3.

The cell can include an activation register 406 that stores an activation input. The activation register can receive the activation input from a left adjacent cell, i.e., an adjacent cell located to the left of the given cell, or from a unified buffer, depending on the position of the cell within the systolic array. The cell can include a weight register 402 that stores a weight input. The weight input can be transferred from a top adjacent cell or from a weight fetcher interface, depending on the position of the cell within the systolic array. The cell can also include a sum in register 404. The sum in register 404 can store an accumulated value from the top adjacent cell. Multiplication circuitry 408 can be used to multiply the weight input from the weight register 402 with the activation input from the activation register 406. The multiplication circuitry 408 can output the product to summation circuitry 410.

The summation circuitry can sum the product and the accumulated value from the sum in register 404 to generate a new accumulated value. The summation circuitry 410 can then send the new accumulated value to another sum in register located in a bottom adjacent cell. The new accumulated value can be used as an operand for a summation in the bottom adjacent cell.

The cell can also shift the weight input and the activation input to adjacent cells for processing. For example, the weight register 402 can send the weight input to another weight register in the bottom adjacent cell. The activation register 406 can send the activation input to another activation register in the right adjacent cell. Both the weight input and the activation input can therefore be reused by other cells in the array at a subsequent clock cycle.

In some implementations, the cell also includes a control register. The control register can store a control signal that determines whether the cell should shift either the weight input or the activation input to adjacent cells. In some implementations, shifting the weight input or the activation input takes one or more clock cycles. The control signal can also determine whether the activation input or weight inputs are transferred to the multiplication circuitry 408, or can determine whether the multiplication circuitry 408 operates on the activation and weight inputs. The control signal can also be passed to one or more adjacent cells, e.g., using a wire.

In some implementations, weights are pre-shifted into a weight path register 412. The weight path register 412 can receive the weight input, e.g., from a top adjacent cell, and transfer the weight input to the weight register 402 based on the control signal. The weight register 402 can statically store the weight input such that as activation inputs are transferred to the cell, e.g., through the activation register 406, over multiple clock cycles, the weight input remains within the cell and is not transferred to an adjacent cell. Therefore, the weight input can be applied to multiple activation inputs, e.g., using the multiplication circuitry 408, and respective accumulated values can be transferred to an adjacent cell.

In some implementations, one neural network input is processed through each of the layers of the neural network to generate a neural network output for the neural network input. in some implementations, however, the circuit processes multiple neural network inputs through the layers of the neural network at once in order to generate neural network outputs for the inputs, thereby increasing the utilization of the circuit.

An input to a particular layer of the neural network can be referred to as a set of activation inputs. Therefore, a neural network input can be a set of activation inputs to a first layer. The first layer can generate an output to be processed by a second layer, and the output can be referred to as a set of activation inputs to the second layer.

In some implementations, instead of the circuit processing one set of activation inputs by sending the activation inputs from a first layer through to a last layer of the neural network to generate an inference, the circuit can process multiple sets of activation inputs at a first layer, e.g., generate accumulated values for each set of activation inputs as described above in FIG. 3, then process multiple sets of activation inputs at a second layer, and so forth. The multiple sets of activation inputs at a given layer can make up one or more batches of inputs. In some implementations, batches for a given layer are processed before the circuit processes batches of a subsequent layer. The circuit can

process a batch by generating accumulated values for each input, which can be an independent set of activation inputs, in the batch.

In some implementations, in order to maximize throughput while performing computations for a given neural network layer, the circuit can be caused to process multiple independent activation inputs while the weight inputs for the given neural network layer are loaded in the systolic array. That is, the weight inputs can be reused with multiple activation inputs. In particular, the circuit reuses weight inputs by computing accumulated values using the weight inputs and different activation inputs, e.g., using matrix computation unit 212 of FIG. 2. By way of illustration, the activation inputs can be from multiple different image resources or multiple audio samples. This will be described further below in reference to FIG. 6.

The circuit has a processing speed and a speed at which it can access memory. Generally, the processing speed is faster than the memory access speed. By reusing weights, and thereby utilizing the circuit's processing speed, the circuit can utilize the cells of the systolic array while, in parallel, accessing weight inputs from memory to be used in subsequent computations.

The circuit can determine, for each layer of the neural network, how many times to reuse weights from a weight reuse value for the circuit. This can be determined by software on the circuit that is compiling for the hardware or by hardware. In some implementations, the weight reuse value is a number of times the circuit can reuse weight inputs before a compute time of reusing the weight inputs the number of times with different activation inputs is greater than or equal to a fetch time of accessing new weight inputs from memory. Therefore, the weight reuse value can indicate how many times the circuit should reuse weight inputs to maximize utilization and minimize memory access wait time.

The weight reuse value can be based on one or more of the following: a clock rate of the memory storing the weight inputs, e.g., dynamic memory 210 of FIG. 2, a number of arithmetic units inside the circuit, e.g., inside matrix computation unit 212 of FIG. 2, and a number of channels in memory, e.g., unified buffer 208 and dynamic memory 210 of FIG. 2. The weight reuse value can be used to determine a respective batch size for each layer of the neural network. The batch size can be referred to as an effective batch size, i.e., a batch size that allows a layer to operate in a way that produces the most efficient amount of weight reuse. The batch size is a number of inputs to be processed at a layer such that the systolic array in the circuit is maximized. In some implementations,

a batch size for a given layer is the ceiling of the weight reuse value divided by the number of times that weight inputs are reused at the layer. The number of times that weight inputs are reused can directly correspond to the number of activation inputs stored in cells of a systolic array of the circuit when performing an accumulation calculation. This is because the weight inputs can remain in the cells, i.e., "paused", while new sets of activation inputs are loaded. When new sets of activation inputs are loaded with the weight inputs being frozen in place, the circuit can reuse the weight inputs again to calculate convolutions with the new sets of activation inputs. An example will be described further below in reference to FIG. 6.

FIG. 5 is a flow diagram 500 of an example method for performing neural network computations for multiple layers. The method can be performed by a special purpose hardware circuit, e.g., using host 202 of FIG. 2 or by one or more processors, firmware, off-chip processes, or by some other software process that is configured to control the circuit.

The circuit can, during processing at each neural network layer, receive a set of layer inputs for processing by the layer (step 502). The set of layer inputs can be from memory, e.g., unified buffer 208 of FIG. 2, or from a previous layer, i.e., the set of inputs are outputs generated from the previous layer in the network, and using the outputs as the set of inputs can be managed by the host 202 as described above in reference to FIG. 2. Each input can be a set of activation inputs, and can be generated from an independent neural network input.

The system can, for a given neural network layer, form one or more batches of inputs from the set of inputs (step 504). As described above, each neural network layer has an associated batch size. Each formed batch at the layer includes a number of inputs that does not exceed the layer's batch size. The batch size for each neural network layer can be determined as a configuration of the neural network. In some implementations, the batch size for each neural network layer is computed when compiling a neural network model for execution on the circuit. Because the batch size for a given layer can be computed before processing begins at a layer, the number of inputs within a batch can be formed based on the batch size of the given layer.

As an example, if 24 images need to be processed by a neural network, and a first layer of the neural network has a batch size of 8, the circuit can form 3 batches of 8 images, i.e., divide the 24 images into 3 batches of 8 inputs. For each batch, the circuit can process the 8 distinct images in the batch by reusing a particular set of weight inputs

for the layer. The circuit can then either (1) process one or more batches at a subsequent layer or (2) process another batch of 8 distinct image resources at the layer using the particular set of weight inputs. The circuit determines whether to proceed with (1) or (2) based on batch sizes of other layers in the neural network, which will be described further below in step 506 and in reference to FIG. 6.

The circuit can, for a given neural network layer, select a number of the one or more batches of inputs to process (step 506). In some implementations, the number is selected such that a count of the inputs in the number of the one or more batches is greater than or equal to the corresponding batch size of a subsequent layer. For example, if a current layer has to process 3 batches each having 5 inputs, and a subsequent layer as a batch size of 10, the circuit can select 2 batches of the 3 to process. The 2 selected batches have a total of 10 inputs, i.e., 2 * 5, which is equal to the corresponding batch size of the subsequent layer, i.e., 10. In some implementations, the remaining third batch will be processed on a subsequent pass of the systolic array in the circuit. Yet another example will be described further below in reference to FIG. 6.

The circuit can process, e.g., using a systolic array, the selected number of batches of inputs to generate a respective layer output for each input in the selected batches (step 508). The circuit can process each selected batch of inputs by reusing weight inputs when computing accumulated values for each input. As described above, the weight inputs are applied to multiple independent inputs since the weight inputs can remain in the cells, i.e., "paused", while new sets of activation inputs are loaded.

By way of illustration, a batch can have two inputs - a set of activation inputs A1 and a set of activation inputs A2. The circuit can reuse a set of weight inputs W1 by applying the set W1 to both sets A1 and A2 before fetching a set W2 of weight inputs from memory, where the set W2 is a set of weights for a subsequent layer, or, if the layer has more weights to be processed by a systolic array of the circuit, a next subset of weights for the layer. In another example, if there are 8 batches of 5 inputs each to be processed, i.e., a total of 40 inputs, and the circuit selected 4 batches to process, the circuit can process the inputs within the 4 batches, i.e., a total of 20 inputs, to generate respective layer outputs, i.e., a total of 20 layer outputs. This will be described further below in reference to FIG. 6.

After processing the selected number of batches, the circuit can either process a second number of batches at the layer or proceed to process one or more batches of inputs at a subsequent layer. This will be described further below in reference to FIG. 6.

Although this method has been described to be implemented on a circuit processing a neural network, this method can also be implemented on a processor, e.g., a Central Processing Unit (CPU) or a Graphics Processing Unit (GPU). For example, the processor can reuse a weight value some number of times while retrieving weights from memory, e.g., Dynamic Random Access Memory (DRAM).

FIG. 6 shows an example neural network 600 having multiple layers and a batch size for each layer. As described above, a batch size for a given layer is the ceiling of the weight reuse value divided by the number of times that weight inputs are reused at the layer. The ceiling function can process a number and return the smallest integer that is not less than the number. The number of times that weight inputs are reused can directly correspond to the number of activation inputs stored in cells of a systolic array of the circuit. For example, a weight reuse value for the circuit can be 1500. Layer 1 602 processes a 170 x 170 input. Each of the 170 x 170 activation inputs can be stored in a cell of the systolic array. Therefore, the batch size for Layer 1 602 is CEIL (1500 / (170*170)) = 1. The batch size for Layer 2 604 is CEIL (1500 / (28 * 28)) = 2. Using the same calculation, Layers 3-5 606-610 have a batch size of 8 and Layer 6 612 has a batch size of 32. In some implementations, the batch size is rounded to a nearest power of 2.

By way of illustration, Layer 1 has a batch size of 1 input and Layer 2 has a batch size of 2 inputs. The circuit can receive, for example, ten inputs for processing at Layer 1, e.g., 10 total distinct images.

The circuit can determine that two batches out of ten at Layer 1, i.e., for a total of 2 inputs, should be processed. In some implementations, the circuit determines this number by dividing the batch size at Layer 2 with the batch size of Layer 1, i.e., 2/1 = 2. By processing two batches of single inputs, the circuit processes two inputs, which is equal to a batch size of Layer 2, i.e., 2. The circuit can process the two batches to generate two layer outputs from the two inputs, e.g., by computing two vectors of accumulated values – one from each image. Each layer output can be an input for input to a subsequent layer. The circuit can form a batch of two inputs, e.g., the two vectors of accumulated values, for processing at Layer 2. The batches can be stored in the unified buffer.

The circuit can determine that four batches at Layer 2 should be processed. By processing four batches of size two, the circuit processes eight inputs, which is equal to a batch size of Layer 3, i.e., 8. The circuit can process the four batches to generate eight

layer outputs from the eight inputs. Then, the circuit can form a batch of eight inputs for processing at Layer 3.

The circuit can continue until the inputs have been processed at all layers of the neural network. For example, because Layers 3-5 have a batch size of 8, the circuit can determine that one batch of eight inputs at each of the Layers 3-5 should be processed. Similarly, the circuit can determine that four batches having eight inputs each at Layer 5 should be processed, thereby forming a batch of 32 inputs, which can be processed at Layer 6. For each input, Layer 6 can generate one or more layer outputs which can be used for inference.

In some implementations, the circuit calculates, e.g., using arithmetic circuitry, a least common multiple of batch sizes across all layers in the neural network. The circuit can then process a minimum number of inputs equal to or greater than the least common multiple at each layer before processing subsequent layers. In other words, the circuit can determine whether to (1) process a batch at a subsequent layer or (2) process another batch at a current layer based on the least common multiple. For example, the least common multiple of the batch sizes for Layers 1-6 602-612 is 32. After processing one batch at Layer 1, the circuit can determine that Layer 1 generated only 1 output, which is less than the least common multiple of 32. Therefore, the circuit can determine to process another batch at Layer 1, at which point a total of 2 outputs have been generated. The circuit can continue generating outputs until 32 outputs, i.e., the least common multiple, have been generated at Layer 1. Then, the circuit can proceed to process a batch at a subsequent layer. In some implementations, each layer output is stored in a unified buffer, e.g., unified buffer 208 of FIG. 2.

By way of illustration, the circuit can select 32 batches for processing, which totals 32 inputs at Layer 1. The circuit can then select 16 batches totaling 32 inputs at Layer 2, then select 4 batches for processing totaling 32 inputs at Layer 3, then select 4 batches for processing totaling 32 inputs at Layer 4, then select 4 batches for processing totaling 32 inputs at Layer 5, then finally select 1 batch for processing totaling 32 inputs at layer 6.

In some implementations, at a particular layer, the circuit processes a number of inputs greater than the least common multiple based on a system parameter, e.g., a parameter provided by a user. In some implementations, the circuit processes a number of inputs that is divisible by the subsequent layer's batch size. In some other implementations, the circuit receives instructions to process a large number of inputs, and

the circuit processes batches of inputs so as to maximize a number of generated outputs at a particular layer before proceeding to generating outputs at a subsequent number. The circuit can constrain the number of outputs that are generated at the particular layer such that the number of outputs generated is divisible by a subsequent layer's batch size.

Embodiments of the subject matter and the functional operations described in this specification can be implemented in digital electronic circuitry, in tangibly-embodied computer software or firmware, in computer hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments of the subject matter described in this specification can be implemented as one or more computer programs, i.e., one or more modules of computer program instructions encoded on a tangible non transitory program carrier for execution by, or to control the operation of, data processing apparatus. Alternatively or in addition, the program instructions can be encoded on an artificially generated propagated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus for execution by a data processing apparatus. The computer storage medium can be a machine-readable storage device, a machine-readable storage substrate, a random or serial access memory device, or a combination of one or more of them.

The term "data processing apparatus" encompasses all kinds of apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit). The apparatus can also include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

A computer program (which may also be referred to or described as a program, software, a software application, a module, a software module, a script, or code) can be written in any form of programming language, including compiled or interpreted languages, or declarative or procedural languages, and it can be deployed in any form, including as a standalone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program may, but need not, correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data, e.g., one or more scripts stored in a markup language

document, in a single file dedicated to the program in question, or in multiple coordinated files, e.g., files that store one or more modules, sub programs, or portions of code. A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

The processes and logic flows described in this specification can be performed by one or more programmable computers executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

Computers suitable for the execution of a computer program include, by way of example, can be based on general or special purpose microprocessors or both, or any other kind of central processing unit. Generally, a central processing unit will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a central processing unit for performing or executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Moreover, a computer can be embedded in another device, e.g., a mobile telephone, a personal digital assistant (PDA), a mobile audio or video player, a game console, a Global Positioning System (GPS) receiver, or a portable storage device, e.g., a universal serial bus (USB) flash drive, to name just a few.

Computer readable media suitable for storing computer program instructions and data include all forms of nonvolatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

To send for interaction with a user, embodiments of the subject matter described in this specification can be implemented on a computer having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball,

by which the user can send input to the computer. Other kinds of devices can be used to send for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input. In addition, a computer can interact with a user by sending documents to and receiving documents from a device that is used by the user; for example, by sending web pages to a web browser on a user's client device in response to requests received from the web browser.

Embodiments of the subject matter described in this specification can be implemented in a computing system that includes a back end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front end component, e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation of the subject matter described in this specification, or any combination of one or more such back end, middleware, or front end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN") and a wide area network ("WAN"), e.g., the Internet.

The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

While this specification contains many specific implementation details, these should not be construed as limitations on the scope of any invention or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this specification in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system modules and components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

Particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In certain implementations, multitasking and parallel processing may be advantageous.

CLAIMS

1. A method for generating a respective neural network output for each of a plurality of inputs, wherein the generating comprises processing each input through each of a plurality of neural network layers to generate the respective neural network output for the input, wherein the neural network layers are arranged in a directed graph structure, and wherein each neural network layer has a respective batch size, the method comprising, for each of the neural network layers:

receiving a plurality of inputs to be processed at the neural network layer;

forming one or more batches of inputs from the plurality of inputs, each batch having a number of inputs equal to the respective batch size for the neural network layer;

selecting a number of the one or more batches of inputs to process, where a count of the inputs in the number of the one or more batches is greater than, less than, or equal to the respective associated batch size of a subsequent layer in the directed graph structure; and

processing the number of the one or more batches of inputs to generate the respective neural network layer output.

2. The method of claim 1, where the respective batch size is based at least on a weight reuse value, the weight reuse value representing a number of times that weight inputs need to be reused for a compute time of output values using the weight inputs at a matrix computation unit to be longer than a load time of the weight inputs from memory.

3. The method of claim 2, where the weight reuse value is based at least on a clock rate of the memory storing the weight inputs.

4. The method of claim 2 or 3, where each batch size is based at least on the weight reuse value divided by a number of times that weight inputs for the respective layer are reused.

5. The method of any of claims 2 to 4, where the plurality of neural network layers is processed at a matrix processing unit, where processing the number of the one or more batches of inputs comprises computing accumulated values for each input using the matrix computation unit.

6. The method of any preceding claim, where each input corresponds to a distinct image resource.

7. The method of any of claims 1 to 5, where each input corresponds to an audio sample.

8. The method of any preceding claim, further comprising forming a batch from the one or more layer outputs for processing at the subsequent layer.

9. The method of any preceding claim, further comprising generating, for each output, a corresponding inference.

10. A system for generating a respective neural network output for each of a plurality of inputs, wherein the generating comprises processing each input through each of a plurality of neural network layers to generate the respective neural network output for the input, wherein the neural network layers are arranged in a directed graph structure, and wherein each neural network layer has a respective batch size, the system comprising:

one or more computers; and

computer-readable medium coupled to the one or more computers and having instructions stored thereon, which, when executed by the one or more computers, cause the one or more computers to, for each of the neural network layers, perform operations comprising:

receiving a plurality of inputs to be processed at the neural network layer;

forming one or more batches of inputs from the plurality of inputs, each batch having a number of inputs equal to the respective batch size for the neural network layer;

selecting a number of the one or more batches of inputs to process, where a count of the inputs in the number of the one or more batches is greater than, less than, or equal to the respective associated batch size of a subsequent layer in the directed graph structure; and

processing the number of the one or more batches of inputs to generate the respective neural network layer output.

11. The system of claim 10, where the respective batch size is based at least on a weight reuse value, the weight reuse value representing a number of times that weight inputs need to be reused for a compute time of output values using the weight inputs at a matrix computation unit to be longer than a load time of the weight inputs from memory.

12. The system of claim 11, where the weight reuse value is based at least on a clock rate of the memory storing the weight inputs.

13. The system of claim 11 or 12, where each batch size is based at least on the weight reuse value divided by a number of times that weight inputs for the respective layer are reused.

14. The system of any of claims 11 to 13, where the plurality of neural network layers is processed at a matrix processing unit, where processing the number of the one or more batches of inputs comprises computing accumulated values for each input using the matrix computation unit.

15. The system of any of claims 10 to 14, where each input corresponds to a distinct image resource.

16. The system of any of claims 10 to 14, where each input corresponds to an audio sample.

17. The system of any of claims 10 to 16, further comprising forming a batch from the one or more layer outputs for processing at the subsequent layer.

18. The system of any of claims 10 to 17, further comprising generating, for each output, a corresponding inference.

19. A computer-readable medium having instructions stored thereon, which, when executed by one or more computers, cause the one or more computers to perform operations for generating a respective neural network output for each of a plurality of inputs, wherein the generating comprises processing each input through each of a plurality of neural network layers to generate the respective neural network output for the input, wherein the neural network layers are arranged in a directed graph structure, and wherein each neural network layer has a respective batch size, the operations comprising, for each of the neural network layers:

receiving a plurality of inputs to be processed at the neural network layer;

forming one or more batches of inputs from the plurality of inputs, each batch having a number of inputs equal to the respective batch size for the neural network layer;

selecting a number of the one or more batches of inputs to process, where a count

of the inputs in the number of the one or more batches is greater than, less than, or equal to the respective associated batch size of a subsequent layer in the directed graph structure; and

processing the number of the one or more batches of inputs to generate the respective neural network layer output.

20. The computer-readable medium of claim 19, where the respective batch size is based at least on a weight reuse value, the weight reuse value representing a number of times that weight inputs need to be reused for a compute time of output values using the weight inputs at a matrix computation unit to be longer than a load time of the weight inputs from memory.

21. The computer-readable medium of claim 20, where the weight reuse value is based at least on a clock rate of the memory storing the weight inputs.

22. The computer-readable medium of claim 20 or 21, where each batch size is based at least on the weight reuse value divided by a number of times that weight inputs for the respective layer are reused.

23. The computer-readable medium of any of claims 20 to 22, where the plurality of neural network layers is processed at a matrix processing unit, where processing the number of the one or more batches of inputs comprises computing accumulated values for each input using the matrix computation unit.

24. The computer-readable medium of any of claims 19 to 23, where each input corresponds to a distinct image resource.

25. The computer-readable medium of any of claims 19 to 23, where each input corresponds to an audio sample.

26. The computer-readable medium of any of claims 19 to 25, further comprising forming a batch from the one or more layer outputs for processing at the subsequent layer.

27. The computer-readable medium of any of claims 19 to 26, further comprising generating, for each output, a corresponding inference.

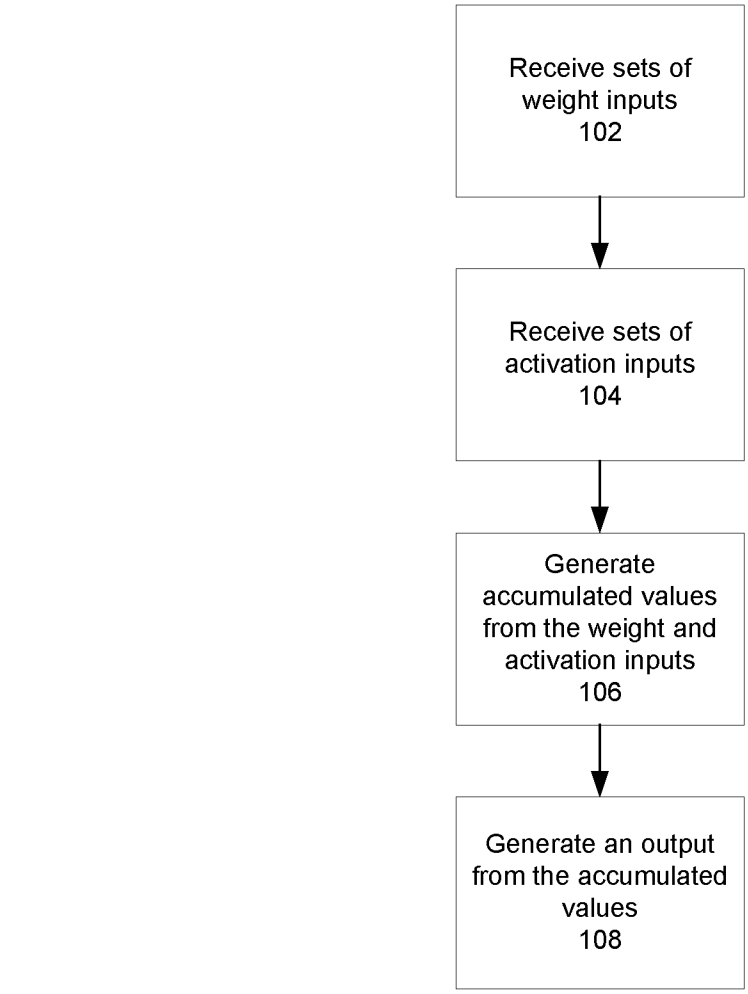
Receive sets of weight inputs 102
Receive sets of activation inputs 104
Generate accumulated values from the weight and activation inputs 106
Generate an output from the accumulated values 108
100

FIG. 1

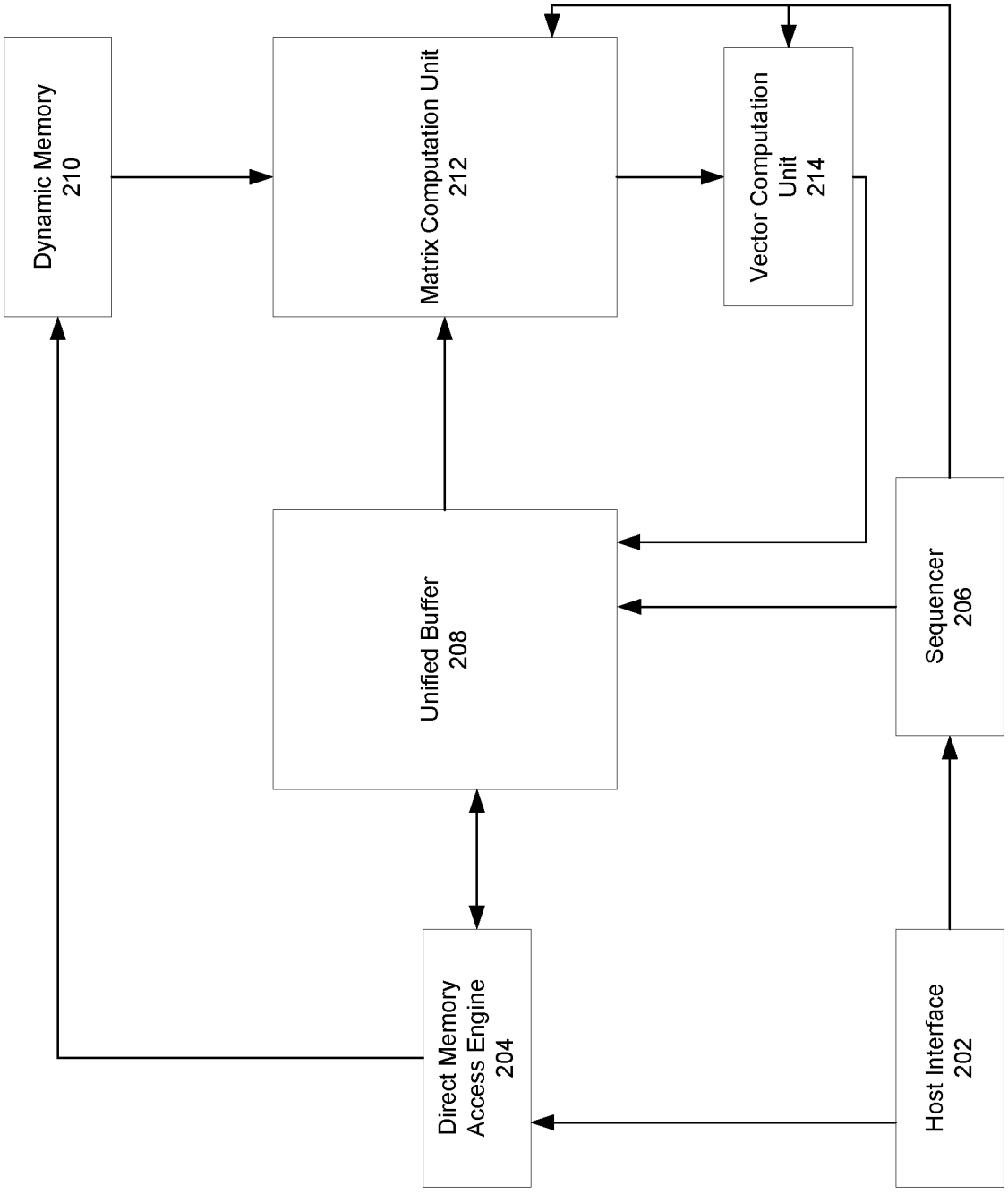
Dynamic Memory
210
Matrix Computation Unit
212
Vector Computation
Unit
214
Unified Buffer
208
Sequencer
206
Direct Memory
Access Engine
204
Host Interface
202
200

FIG. 2

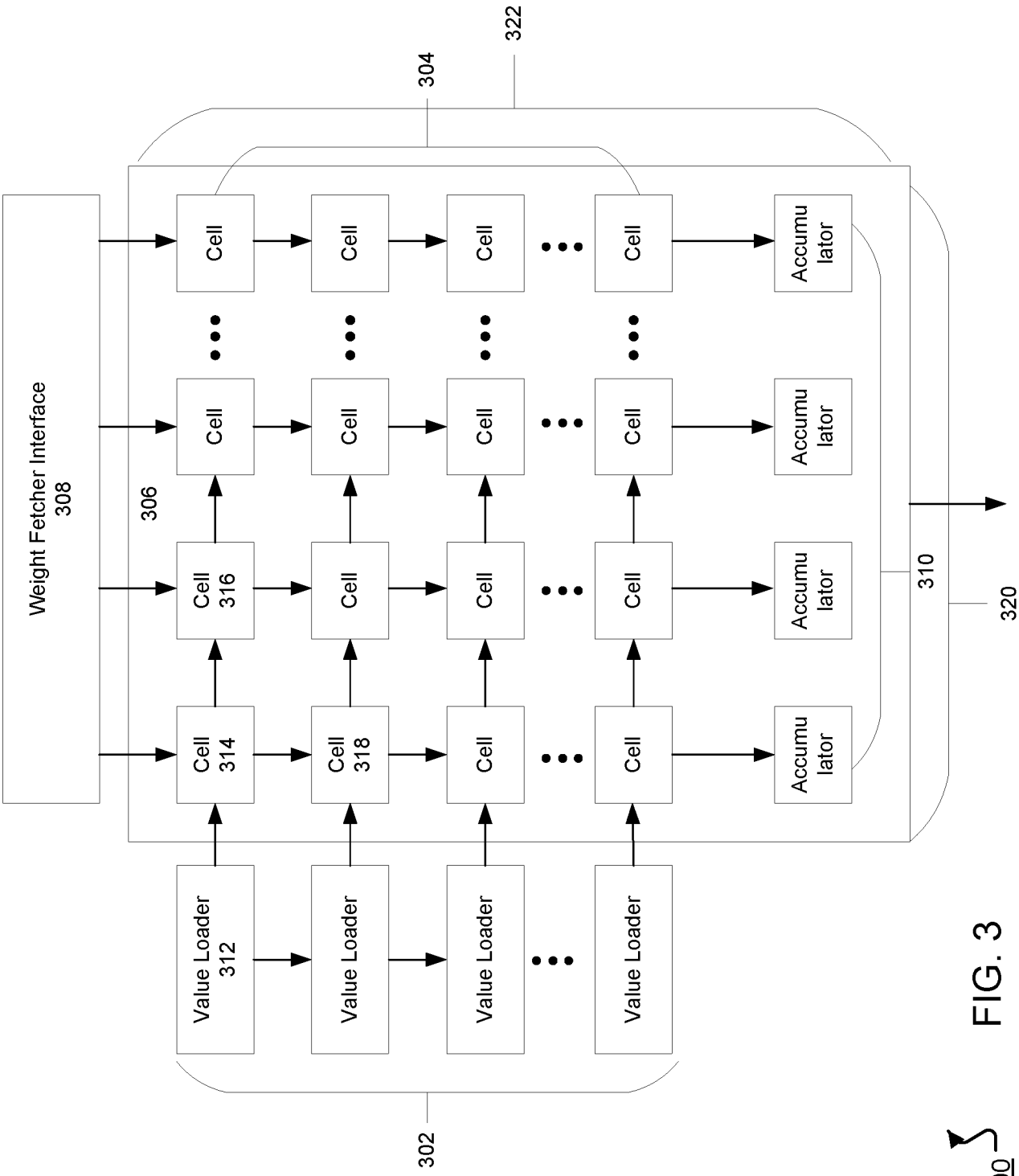
Weight Fetcher Interface
308
306
Value Loader
312
Cell
314
Cell
316
Cell
318
Cell
Accumu
lator
302
304
310
320
322
300

FIG. 3

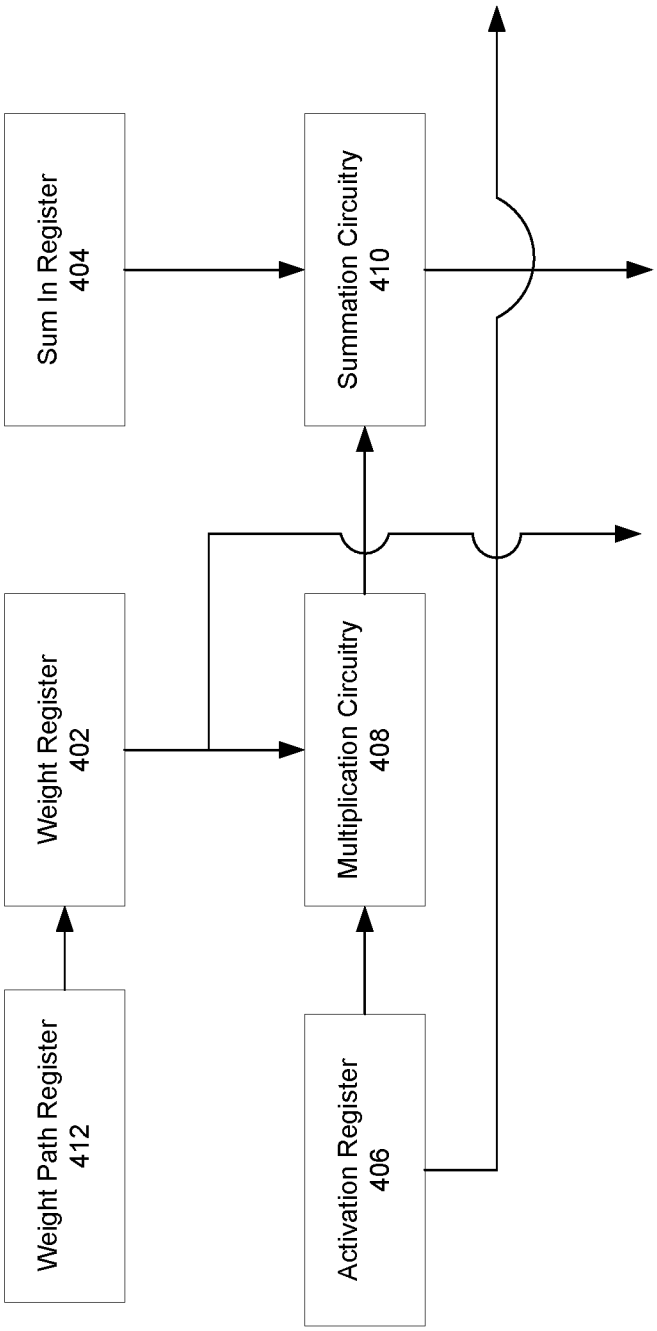
Sum In Register
404
Summation Circuitry
410
Weight Register
402
Multiplication Circuitry
408
Weight Path Register
412
Activation Register
406
400

FIG. 4

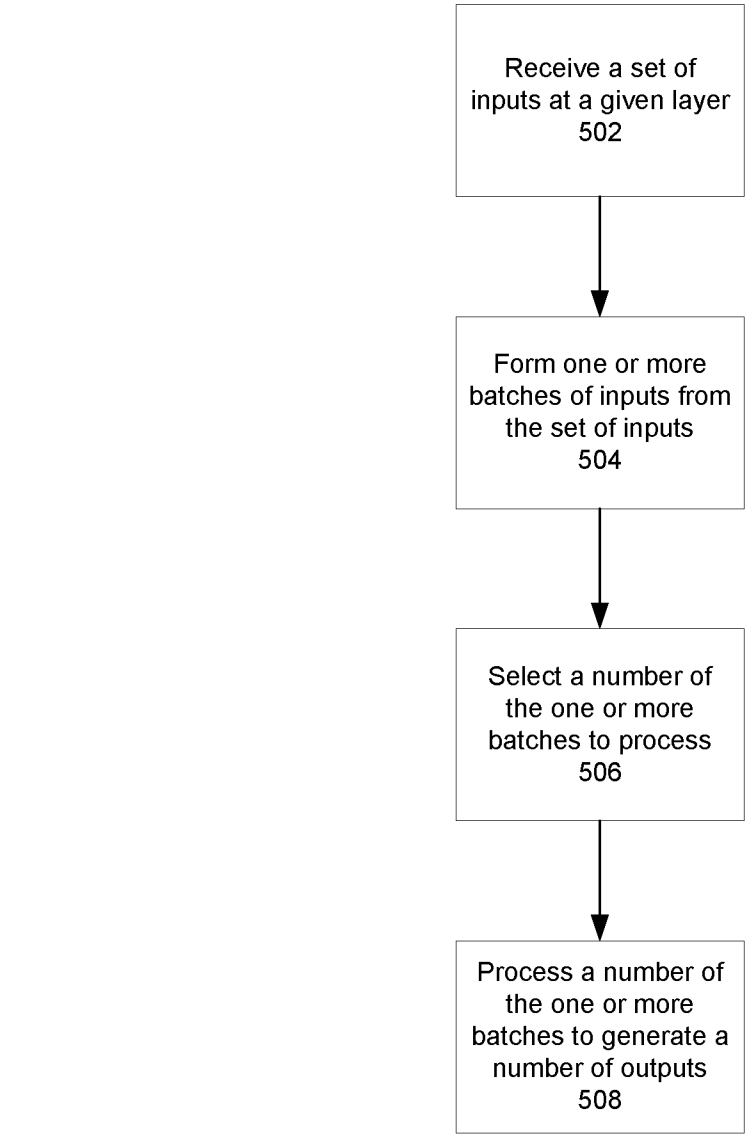
Receive a set of inputs at a given layer 502
Form one or more batches of inputs from the set of inputs 504
Select a number of the one or more batches to process 506
Process a number of the one or more batches to generate a number of outputs 508
500

FIG. 5

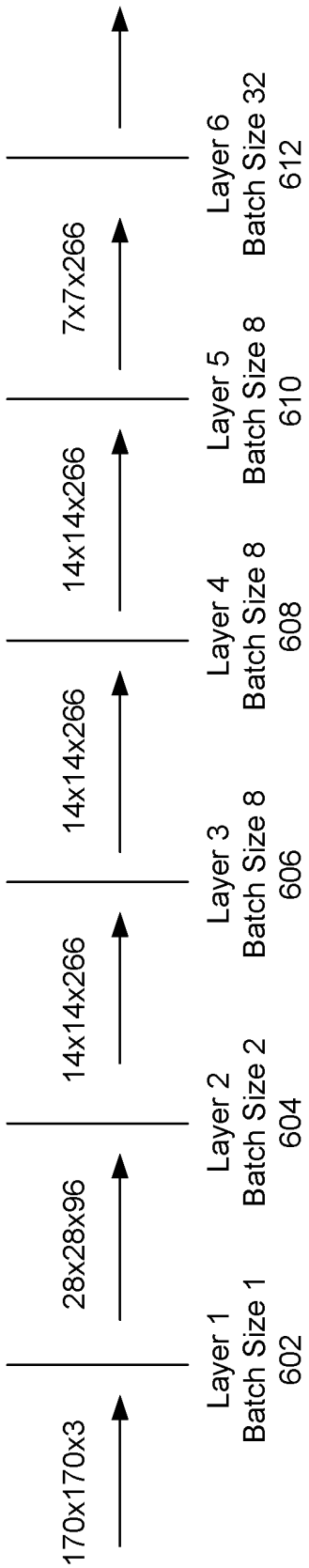
170x170x3
Layer 1
Batch Size 1
602
28x28x96
Layer 2
Batch Size 2
604
14x14x266
Layer 3
Batch Size 8
606
14x14x266
Layer 4
Batch Size 8
608
14x14x266
Layer 5
Batch Size 8
610
7x7x266
Layer 6
Batch Size 32
612
600

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2016/030515

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06N3/08 G06F15/80
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06N 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)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2014/142929 A1 (SEIDE FRANK TORSTEN BERND [CN] ET AL) 22 May 2014 (2014-05-22) paragraphs [0015] - [0022]; figure 1 paragraphs [0034] - [0036] paragraphs [0059] - [0060]; figure 4 ----- -/--	2-5, 10-27

[X] Further documents are listed in the continuation of Box C.

[X] 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
18 August 2016	25/08/2016

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Goy, Sébastien

3

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/030515

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. [X] Claims Nos.: 1, 6-9
because they relate to subject matter not required to be searched by this Authority, namely:

see FURTHER INFORMATION sheet PCT/ISA/210

2. [ ] Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. [ ] Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. [ ] As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. [ ] As all searchable claims could be searched without effort justifying an additional fees, this Authority did not invite payment of additional fees.

3. [ ] As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. [ ] No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

[ ] The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

[ ] The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

[ ] No protest accompanied the payment of additional search fees.

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

Continuation of Box II.1

Claims Nos.: 1, 6-9

No search and preliminary examination will be carried out for claims 1, and 6-9, as they fall under the exceptions enumerated in Rules 39 and 67 PCT. 1.1 The method of claim 1, the system of claim 10 and the computer-readable medium of claim 19 all aim at generating the output of an artificial neural network, i.e. a model created to simulate cognitive functions. 1.2 This model could be used for a concrete purpose, to e.g. predict the behavior of the entity it simulates, but this purpose cannot be derived from the wording of this claim. Thus this model can only be regarded as a purely abstract, i.e. mathematical, model. 1.3 "Scientific and mathematical theories" are covered by the provisions of Rules 39.1(i) and 67.1(i) PCT, which govern the exclusion of this specific type of subject-matter from search and preliminary examination. The International Searching and Preliminary Examining Authorities have divergent practice with regard to the search and examination of this type of subject-matter. The following illustrates the policy of the EPO when acting as International Authority (see also PCT Guidelines, 9.01-9.05). 1.4 The only outcome of the methods of claims 1, 8 and 9 are numerical values, namely: the output of the network's layers. No tangible effect these methods could produce can be derived from their wording. Thus, these methods can only be regarded as calculation procedures using a mathematical model. 1.5 Claims 6 and 7 stipulate that each input to be processed "corresponds to a distinct image resource" or "an audio sample", which are physical entities. However, this wording only establishes a cognitive relation between the input and the physical entities, from which no functional feature can be derived for the input. It follows that, even if the physical origin of the input is stipulated, it is akin to a mere sequence of numbers, and the purpose of processing this sequence of numbers according to these methods remains unspecified. 1.6 Therefore, the subject-matter of claims 1, and 6-9, belongs entirely to the theoretical field of mathematics. It follows that this Authority is not required to search and carry out a preliminary examination for the subject-matter of claims 1, and 6-9. Therefore, given their current wording, no search or preliminary examination will be carried by this Authority for claims 1, and 6-9.

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2016/030515

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	SANG KYUN KIM ET AL: "A Large-Scale Architecture for Restricted Boltzmann Machines", FIELD-PROGRAMMABLE CUSTOM COMPUTING MACHINES (FCCM), 2010 18TH IEEE ANNUAL INTERNATIONAL SYMPOSIUM ON, IEEE, PISCATAWAY, NJ, USA, 2 May 2010 (2010-05-02), pages 201-208, XP031681940, ISBN: 978-0-7695-4056-6 page 201 page 204 - page 205 page 207	2-5, 11-14, 20-23

A	THIERRY CORNU ET AL: "Design, Implementation, and Test of a Multi-Model Systolic Neural-Network Accelerator", SCIENTIFIC PROGRAMMING - PARALLEL COMPUTING PROJECTS OF THE SWISS PRIORITY PROGRAMME, vol. 5, no. 1, 1 January 1996 (1996-01-01) , pages 47-61, XP055294242, DOI: 10.1155/1996/189626 page 52, left-hand column page 55, left-hand column page 56, paragraph right - page 57, last paragraph	2,5,11, 14,20,23

3

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2016/030515

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014142929 A1	22-05-2014	NONE	