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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 September 2018 (13.09.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/165209 A1

(51) International Patent Classification:
G06N 3/063 (2006.01) *G06F 17/16* (2006.01)

(21) International Application Number:
PCT/US2018/021221

(22) International Filing Date:
06 March 2018 (06.03.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15/452,624 07 March 2017 (07.03.2017) US
15/624,629 15 June 2017 (15.06.2017) US

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

(72) Inventors: **GULLAND, William John**; 1600 Amphitheatre Parkway, Mountain View, California 94043 (US). **YOUNG, Reginald Clifford**; 1600 Amphitheatre Parkway, Mountain View, California 94043 (US).

(74) Agent: **PORTNOV, Michael**; FISH & RICHARDSON P.C., P.O. BOX 1022, 3300 RBC PLAZA, 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, 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.

(54) Title: DEPTH CONCATENATION USING A MATRIX COMPUTATION UNIT

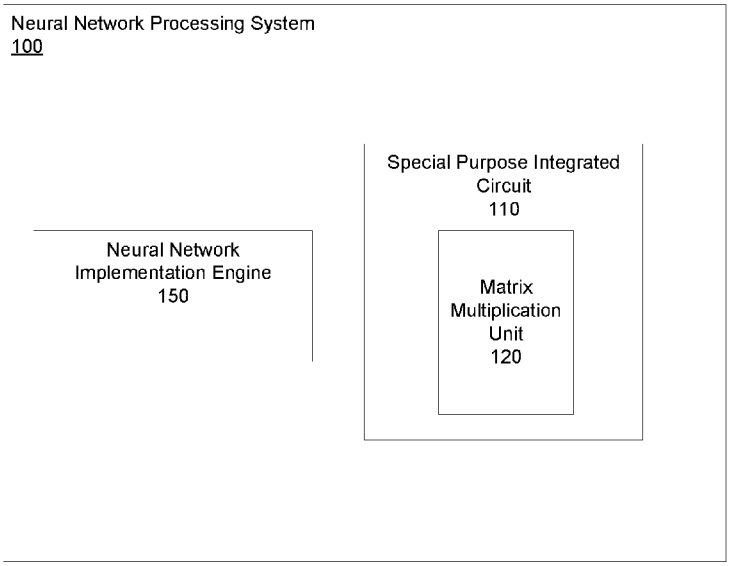


FIG. 1

(57) Abstract: Methods, systems, and apparatus, including computer programs encoded on computer storage media, for depth concatenation using a matrix computation unit. One of the methods includes: receiving a request to process network inputs to a neural network using an integrated circuit, the neural network comprising a depth concatenation neural network layer; and generating instructions that, when executed by the integrated circuit, cause the integrated circuit to performing operations comprising: for each spatial location in a first input tensor to the depth concatenation layer and a second input tensor to the depth concatenation layer: multiplying, using the matrix computation unit, a second depth vector for the spatial location by a shift weight matrix for the depth concatenation layer to generate a shifted second depth vector; and adding the shifted second depth vector and a first input depth vector for the spatial location to generate a concatenated depth vector.

WO 2018/165209 A1

[Continued on next page]

(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))*

DEPTH CONCATENATION USING A MATRIX COMPUTATION UNIT

BACKGROUND

[0001] This specification relates to performing neural network computations in hardware.

[0002] Neural networks are machine learning models that employ one or more layers of models 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 one or more other layers in the network, i.e., one or more other hidden layers 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.

[0003] Some neural networks include a depth concatenation layer that receives as input two or more tensors, i.e., multi-dimensional matrices, that are the outputs of other layers in the neural network and concatenates the input tensors along the depth dimension. In particular, each input tensor has two spatial dimensions x and y and a depth dimension z. By concatenating two input tensors, one having dimensions x_1 by y_1 by z_1 and the other having dimensions x_1 by y_1 by z_2, along the depth dimension z, the depth concatenation layer generates an output tensor that has dimensions x_1 by y_1 by $(z_1 + z_2)$. The output of the depth concatenation layer can then be provided as input to one or more other layers in the neural network or used as the output of the neural network.

SUMMARY

[0004] This specification describes technologies for performing a concatenation of two tensors along the depth dimension using a matrix computation unit. These technologies generally involve receiving a request to process network inputs to a neural network that includes a depth concatenation layer on an integrated circuit for performing neural network computations. The integrated circuit includes a matrix computation unit that performs vector - matrix multiplications in hardware, but cannot directly perform a depth concatenation operation in hardware. Instead, a neural network processing system generates instructions that, when executed by the integrated circuit, cause the integrated circuit to perform operations in hardware that generate an output that satisfies the specification of the depth concatenation layer using the matrix computation unit.

[0005] The subject matter described in this specification can be implemented in particular embodiments so as to realize one or more of the following advantages. An output that satisfies the specification of a depth concatenation layer can be generated in hardware by a special-purpose integrated circuit even though the integrated circuit cannot directly perform a depth concatenation operation in hardware. By generating the satisfactory output in hardware on the integrated circuit, the processing of an inference for a neural network that includes a depth concatenation layer can be performed without passing data back to the host, i.e., without performing part of the computation off-chip, even though the integrated circuit does not directly support the depth concatenation operation. In other words, all computations related to computing the output of the depth concatenation layer occurs on the special-purpose integrated circuit. In particular, by performing matrix multiplies on depth vectors from the two input tensors using shift matrices as described in this specification, the integrated circuit is able to compute the output of the depth concatenation layer in hardware on-chip. This allows the processing of an inference for such a neural network to be performed efficiently without modifying the hardware architecture of the integrated circuit. In particular, the system can efficiently process a neural network inference without adding depth-concatenating hardware to the special-purpose circuit or adding shift support to the vector unit of the special-purpose circuit. That is, processing delays due to needing to perform part of the computation off-chip, in software, or both are avoided.

[0006] 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

[0007] FIG. 1 shows an example neural network processing system.

[0008] FIG. 2 shows an example special-purpose integrated circuit.

[0009] FIG. 3 is a flowchart of an example process for generating instructions that cause a special-purpose integrated circuit to generate an output tensor for a depth concatenation layer.

[0010] FIG. 4 is a flowchart of an example process for concatenating two tensors along the depth dimension.

[0011] FIG. 5 is a flowchart of another example process for concatenating two tensors along the depth dimension.

[0012] FIG. 6 is a flowchart of yet another example process for concatenating two tensors along the depth dimension.

[0013] FIG. 7 shows an example of a depth concatenation computation that requires a single shift matrix.

[0014] FIG. 8 shows an example of a depth concatenation computation that requires two shift matrices.

[0015] Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0016] FIG. 1 shows an example neural network processing system 100.

[0017] The neural network processing system 100 is an example of a system implemented as one or more computers in one or more locations in which the systems, components, and techniques described below can be implemented.

[0018] The neural network processing system 100 is a system that performs neural network computations using a special-purpose integrated circuit 110. The integrated circuit 110 is a special-purpose integrated circuit for performing neural network computations and includes a matrix computation unit 120 that performs vector - matrix multiplications in hardware. An example of such a special-purpose integrated circuit is described in more detail below with reference to FIG. 2.

[0019] In particular, the neural network processing system 100 receives requests to implement neural networks on the special-purpose integrated circuit 110, implements the neural networks on the special-purpose integrated circuit 110, and, once a given neural network is implemented, processes inputs to the neural network using the special-purpose integrated circuit 110 to generate neural network inferences.

[0020] That is, the neural network processing system 100 can receive a request that specifies a neural network architecture for a neural network that is to be used to process inputs. The neural network architecture defines the number and configuration of layers in the neural network and values of the parameters for each of the layers that has parameters.

[0021] To implement a neural network on the special-purpose integrated circuit 110, the neural network processing system 100 includes a neural network implementation

engine 150 that is implemented as one or more computer programs on one or more computers in one or more physical locations.

[0022] The neural network implementation engine 150 generates instructions that, when executed by the integrated circuit 110, cause the integrated circuit 110 to perform the operations specified by the neural network to generate a neural network output from a received neural network input.

[0023] Once the instructions have been generated by the neural network implementation engine 150 and provided to the integrated circuit 110, the neural network processing system 100 can receive neural network inputs and can process the neural network inputs using the neural network by causing the integrated circuit 110 to execute the generated instructions.

[0024] Some neural networks, however, include one or more incompatible neural network layers. The term incompatible neural network layer as used in this specification refers to a neural network layer that specifies an operation that cannot be directly performed in hardware by the integrated circuit 110. To implement these neural networks on the integrated circuit, the neural network implementation engine 150 generates instructions that, when executed by the integrated circuit 110, cause the integrated circuit 110 to generate an output for the incompatible neural network layer by performing operations in hardware that are different from those specified by the neural network layer but that result in a layer output being generated that satisfies the specification of the incompatible neural network layer, i.e., a layer output that is the same as an output that would have been generated by directly performing the operations specified by the layer.

[0025] In particular, some neural networks include a depth concatenation layer. A depth concatenation layer is a layer that receives as input two or more tensors, i.e., two or more multi-dimensional matrices, that are the outputs of other layers in the neural network and concatenates the input tensors along the depth dimension. In particular, each input tensor has two spatial dimensions x and y and a depth dimension z. By concatenating two input tensors, one having dimensions x_1 by y_1 by z_1 and the other having dimensions x_1 by y_1 by z_2, along the depth dimension z, the depth concatenation layer generates an output tensor that has dimensions x_1 by y_1 by $(z_1 + z_2)$. The output tensor can then be used as input by another layer of the neural network.

[0026] Examples of neural networks that may be implemented on integrated circuit 110 and that include one or more depth concatenation layers are the image recognition neural networks described in Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, “Inception-

v4, Inception-ResNet and the Impact of Residual Connections on Learning," available at https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45169.pdf.

[0027] Other examples of neural networks that include depth concatenation layers are long-short term memory (LSTM) neural networks, like the ones described in Hasim Sak, Andrew Senior, Francoise Beaufays, Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling, available at http://193.6.4.39/~czap/letoltes/is14/is2014/pdf/author/is141304.pdf.

[0028] Because the main hardware unit that performs matrix operations on the integrated circuit 110 is the matrix computation unit 120, the integrated circuit cannot directly perform a depth concatenation operation in hardware.

[0029] To implement a neural network that includes a depth concatenation layer on the integrated circuit 110, the neural network implementation engine 150 generates instructions that, when executed by the integrated circuit 110 during processing of a neural network input by the neural network, cause the integrated circuit 110 to perform other operations in hardware to generate an output tensor that satisfies the specification of the depth concatenation neural network layer using the matrix computation unit 120. These instructions and other operations are described in more detail below with reference to FIGS. 3-8.

[0030] While this specification describes the circuit used to perform neural network as being a special-purpose integrated circuit, the techniques described in this specification can be performed on any circuit, e.g., an FPGA, an ASIC, a GPU, and so on, that is augmented with a matrix computation unit.

[0031] FIG. 2 shows an example special-purpose integrated circuit 200 for performing neural network computations.

[0032] The integrated circuit 200 includes a host interface 202. The host interface 202 can receive instructions that include parameters for a neural network computation. The parameters can include at least one or more of the following: how many layers should be processed, data identifying corresponding sets of weight inputs for each 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, and so on. When the neural network being processed includes a depth concatenation layer, the parameters include one or more shift weight matrices for the depth concatenation layer and one or more modified identity weight matrices for the depth concatenation layer. Shift weight

matrices and modified identity weight matrices are described in more detail below with reference to FIGS. 3-5.

[0033] 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 200, e.g., how the sets of weight inputs and the sets of activation inputs flow through the circuit 200. The sequencer 206 can send the control signals to a unified buffer 208, a matrix multiplication 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 a dynamic memory 210.

[0034] 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.

[0035] In some implementations, the direct memory access engine 204 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. When the neural network being processed includes a depth concatenation layer, the direct memory access engine 204 stores the one or more shift weight matrices for the depth concatenation layer and, in some implementations, the modified identity weight matrix for the depth concatenation layer in the dynamic memory 210.

[0036] 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.

[0037] When instructed to do so, 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 multiplication unit 212.

[0038] Generally, the matrix multiplication unit 212 can be any unit that performs multiplication of a vector by a matrix in hardware. In some implementations, the matrix multiplication unit 212 is a two-dimensional systolic array. In this case, the matrix multiplication unit 212 can perform multiple vector-matrix multiplication in parallel or perform matrix-matrix multiplications. The matrix multiplication unit 212 can also be a one-dimensional systolic array or other circuitry that can perform mathematical operations, e.g., multiplication and addition.

[0039] The matrix multiplication 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 cases, the matrix multiplication unit 212 sends the vector of outputs to the unified buffer 208, which sends the vector of outputs to the vector computation unit 214 or back to the matrix multiplication unit 212 if the current neural network layer does not apply an activation function. For example, once the depth concatenated output of a depth concatenation layer has been generated, the matrix multiplication unit 212 can send the output to the unified buffer 208 rather than to the vector computation unit 214 because the depth concatenation layer does not apply an activation function. In some other cases, even though the depth concatenation layer does not apply an activation function, the matrix multiplication unit 212 sends the output to the vector computation unit 214, which applies an identity activation function to the output, i.e., does not modify the output, before routing the output back to the unified buffer 208.

[0040] The vector computation unit 214 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 multiplication unit 212, e.g., for use in a subsequent layer in the neural network.

[0041] The integrated circuit 200 and, in particular, an example implementation of the matrix multiplication unit 212 that allows the matrix multiplication unit 212 to perform vector-matrix multiplication in hardware, is described in more detail in U.S. Patent Application No. 14/844,524, titled “Neural Network Processor,” filed on September 3, 2015, the entire contents of which are hereby incorporated by reference herein in their entirety.

[0042] FIG. 3 is a flowchart of an example process 300 for generating instructions that cause a special-purpose integrated circuit to generate an output tensor for a depth concatenation layer. For convenience, the process 300 will be described as being performed by a system of one or more computers, located in one or more locations, and programmed appropriately in accordance with this specification. For example, a neural network processing system, e.g., the neural network processing system 100 of FIG. 1, appropriately programmed, can perform the process 300.

[0043] The system receives a request to implement a neural network on a special-purpose integrated circuit, i.e., to process inputs to the neural network using the special-purpose integrated circuit to generate outputs (step 302).

[0044] In particular, the neural network to be implemented on the integrated circuit includes a depth concatenation layer that specifies a concatenation of two tensors along the depth dimension. For example, the depth concatenation layer may specify a concatenation along the depth dimension of a first input tensor having dimensions x_1 by y_1 by z_1 and a second input tensor having dimensions x_1 by y_1 by z_2 to generate an output tensor that has dimensions x_1 by y_1 by $(z_1 + z_2)$.

[0045] The system generates one or more shift weight matrices for use in performing the concatenation specified by the depth concatenation layer (step 304).

[0046] A shift matrix is a matrix that, when multiplied with an input vector, generates an output vector in which the position of one or more of the entries of the input vector is shifted while one or more other entries of the input vector have been replaced by zeroes. Generally, the shift matrices described in this specification are matrices for use as the matrices on the right side of a matrix multiplication operation to move values of input vectors to desired positions in an output vector. However, in some other implementations, differently constructed shift matrices can be used as the matrices on the left side of the matrix multiplication operation to achieve the same results.

[0047] In particular, because the matrix computation unit of the integrated circuit performs matrix multiplication in hardware, the matrix computation unit has a maximum vector length (*max*). The maximum vector length is the maximum length of a vector that can be multiplied by a matrix by the matrix computation unit in one pass, i.e., without dividing the vector into multiple inputs to the matrix computation unit. For example, if the matrix computation unit is a one-dimensional or two-dimensional systolic array, the maximum vector length is equal to the number of columns in the unit or to the number of rows in the unit.

[0048] For tensors having a depth of less than or equal to *max*, the system stores the tensor as a collection of respective depth vectors having length *max* for every spatial location in the input tensor.

[0049] A spatial location is a pair of (x,y) spatial coordinates, i.e., so that all entries at all depth dimensions that share the same (x,y) spatial coordinates in a tensor are at the same spatial location. The depth vector for a given spatial location is a vector that includes all of the entries in the tensor that are at the given spatial location. If the tensor

has a depth z of less than *max*, the last *max* - z entries of each depth vector are padded with zeroes or garbage values, i.e., values that may be used for other purposes but are not relevant to and should not influence the depth concatenation operation.

[0050] For tensors having a depth of more than *max*, the system represents each depth vector as multiple *max* length vectors. Each of these *max* length vectors that makes up part of a depth vector will be referred to in this specification as a chunk vector.

[0051] For a given spatial location in a tensor having a depth z, the depth vector for the location is represented as ceiling(z/max) chunk vectors, wherein ceiling(x) is the least integer greater than or equal to x. The first floor(z/max) chunk vectors, with floor(x) being the greatest integer less than or equal to x, each store values from the corresponding dimensions in the depth vector for the spatial location, i.e., with the first chunk vector storing the entries at the spatial location in the first *max* depth dimensions, the second chunk vector storing the entries at the spatial location in the second *max* depth dimensions, and so on. The first floor(z/max) * *max* entries of the last chunk vector are the entries at the last floor(z/max) * *max* depth dimensions and any remaining entries are padded with zeroes or garbage values.

[0052] In some cases, the number of depth dimensions (z_1+ z_2) of the output tensor will not exceed the maximum vector length and the depth vectors of the output tensors are single chunk vectors. In these cases, the system generates a single shift weight matrix for the depth concatenation operation.

[0053] If the number of depth dimensions of the output tensor does exceed the maximum vector length and the depth vectors are represented by multiple chunk vectors, the system may need to generate more than one shift weight matrix for the depth concatenation operation.

[0054] The system also generates one or more modified identity weight matrices for the depth concatenation layer. A modified identity weight matrix is a matrix that has ones along a portion of the main diagonal and zeros for all other entries.

[0055] In general, a shift matrix that shifts entries that start at the j-th position of an input vector to starting at the i-th position of the input vector is a *max* by *max* matrix that is all zeros other than a diagonal row of ones starting at the j-th value of the i-th column of the matrix.

[0056] Shift matrices and modified identity weight matrices will be described in more detail below with reference to FIGS 4-8.

[0057] The system stores the one or more shift weight matrices and the one or more modified identity weight matrices for the depth concatenation layer in a memory that is accessible to the special-purpose integrated circuit (step 306). For example, the system can send the generated matrix or matrices to a host interface of the special-purpose integrated circuit for storage in a dynamic memory accessible to the circuit.

[0058] The system generates instructions that, when executed by the integrated circuit during processing of a neural network input by the neural network, cause the integrated circuit to generate an output tensor that satisfies the specification of the depth concatenation neural network layer using the generated matrices (step 308). In particular, the system generates instructions that, when executed, cause the integrated circuit to perform the processes 400, 500, or 600 described below with reference to FIGS. 4-6, respectively.

[0059] FIG. 4 is a flowchart of another example process 400 for concatenating two input vectors. The process 400 is performed in hardware by a special-purpose integrated circuit that includes a hardware matrix computation unit, e.g., the special-purpose integrated circuit 110 of FIG. 1.

[0060] The integrated circuit receives two inputs to be concatenated (step 402). Each input is a depth vector at a given spatial location from a respective tensor to be depth concatenated and each input is made up of one or more *max*-sized chunk vectors. That is, one input is the depth vector at the spatial location from one input tensor and the other input is the depth vector at the spatial location from the other input tensor.

[0061] In some cases, i.e., when the number of depth dimensions in either or both tensors is not a multiple of *max*, the last chunk vector for each spatial location in either or both tensors includes padded values, i.e., zeros or garbage values that have been added to the chunk vector but are not part of the input tensor. The entries in a chunk vector other than the padded entries will be referred to as non-padded entries.

[0062] The integrated circuit identifies, e.g., based on a control signal, the chunk vector in the first or second input that is the first chunk vector that needs to be modified as part of the depth concatenation (referred to as "the first modified chunk vector") (step 404). The first chunk vector that needs to be modified is the first chunk vector that includes one or more padded entries.

[0063] For each chunk vector before the first chunk vector that needs to be modified, the integrated circuit moves the chunk vector to the output of the matrix computation unit unmodified and then moves the chunk vectors out of the output as output chunk vectors

of the concatenation operation (step 406). Moving vectors to the output of the matrix computation unit will be described below with reference to FIGS. 5-8.

[0064] The integrated circuit moves the non-padded entries of the first chunk vector that needs to be modified into the output of the matrix computation unit (step 408).

[0065] That is, the integrated circuit multiplies the first modified chunk vector by a modified identity matrix to move a chunk vector that has the non-padded entries of the first chunk vector and zeroes for the remaining entries to the output of the matrix computation unit. Multiplying vectors by partial identity matrices will be described below with reference to FIGS. 5-8.

[0066] The integrated circuit moves, using an appropriate shift matrix, a shifted chunk vector into the output to sum the shifted chunk vector and the chunk vector currently in the output and then moves the sum of the chunk vectors out of the output as an output chunk vector of the concatenation operation (step 410).

[0067] For the first iteration of step 410 that is performed during a depth concatenation, the chunk vector currently in the output is the first chunk vector that needs to be modified.

[0068] The shifted chunk vector is a chunk vector that has zeroes as its first (*max* - *n*) entries and the first *n* entries of the next chunk vector as its remaining entries, where *n* is the number of padded entries in the chunk vector that is currently in the output and the next chunk vector is the next chunk vector to be operated on when the chunk vectors in the inputs are arranged in sequence starting from the first chunk vector in the first input and ending with the last chunk vector in the second input.

[0069] Generating shifted chunk vectors using shift matrices and summing vectors is described in more detail below with reference to FIGS. 5-8.

[0070] The integrated circuit moves, using another appropriate shift matrix, another shifted chunk vector to the output (step 412).

[0071] The other shifted chunk vector is a chunk vector that has any additional non-padded entries in the next chunk vector as its first entries and padded entries as its remaining entries.

[0072] The integrated circuit continues performing steps 410 and 412 until there are no remaining next chunk vectors, i.e., until after all the input chunk vectors have been operated on.

[0073] The system can perform the process 400 for each spatial location in the input tensors to depth concatenate the two input tensors.

[0074] FIG. 5 is a flowchart of another example process 500 for concatenating two tensors along the depth dimension. The process 500 is performed in hardware by a special-purpose integrated circuit that includes a hardware matrix computation unit, e.g., the special-purpose integrated circuit 110 of FIG. 1.

[0075] In particular, the process 500 is an example of a process to be performed to concatenate two tensors when the concatenated tensor has a number of depth dimensions that does not exceed the maximum vector length of the matrix computation unit, i.e., each output depth vector can be stored as a single chunk.

[0076] The integrated circuit accesses a shift weight matrix for the depth concatenation (step 502) from a memory accessible to the integrated circuit, e.g., from the dynamic memory 210 of FIG. 2. In some implementations, the integrated circuit also accesses a modified identity weight matrix for the depth concatenation from the memory.

[0077] The integrated circuit moves a first depth vector for a given spatial location in the first input tensor to the output of the matrix computation unit (step 504). The first depth vector for the given spatial location is a vector that includes all of the entries in the first input tensor that are at the given spatial location and has padded values as any remaining values of the first depth vector.

[0078] For example, the integrated circuit can move each entry of the first depth vector to a respective sum-in register of a set of registers that stores the outputs of multiplications performed by the matrix computation unit.

[0079] To move the first depth vector to the output, the integrated circuit can multiply the first depth vector by a modified identity weight matrix for the depth concatenation using the matrix computation unit, resulting in the first depth vector being stored in the output of the matrix computation unit. The modified identity weight matrix is a *max* x *max* matrix that is all zeroes other than having ones in the first z_1 entries of the main diagonal.

[0080] The integrated circuit multiplies a second depth vector for the given spatial location in the second input tensor by the shift weight matrix for the depth concatenation (step 506) to generate a shifted second depth vector. The second depth vector for the given spatial location is a vector that includes all of the entries in the second input tensor that are at the given spatial location and has padded values as any remaining values of the second depth vector.

[0081] By virtue of the structure of the shift weight matrix, the resulting shifted second depth vector is a vector that has *max* entries, with the first z_1 entries being zero, the next z_2

entries being the entries of the second depth vector for the spatial location, and any remaining entries being zero.

[0082] The integrated circuit sums the first depth vector and the shifted second depth vector to generate a concatenated depth vector (step 508). For example, the system can add each entry of the second depth vector to the corresponding entry of the first depth vector by moving the entry of the second depth vector into the sum-in register that stores the corresponding entry of the first depth vector.

[0083] The integrated circuit can perform steps 504-508 for each spatial location in the input tensors to generate the output of the depth concatenation layer.

[0084] FIG. 6 is a flowchart of another example process 600 for concatenating two tensors along the depth dimension. The process 600 is performed in hardware by a special-purpose integrated circuit that includes a hardware matrix computation unit, e.g., the special-purpose integrated circuit 110 of FIG. 1.

[0085] In particular, the process 600 is an example of a process to be performed to concatenate two tensors when the first tensor has a number of depth dimensions that is less than the maximum vector length of the matrix computation unit but the concatenated tensor has a number of depth dimensions that exceeds the maximum vector length. That is, the first depth vector is the first chunk vector that needs to be modified in the process 400 of FIG. 4.

[0086] The integrated circuit accesses shift weight matrices for the depth concatenation (step 602) from a memory accessible to the integrated circuit, e.g., from the unified buffer 208 of FIG. 2. In some implementations, the integrated circuit also accesses a modified identity weight matrix for the depth concatenation from the memory.

[0087] The integrated circuit moves a first depth vector for a given spatial location in the first input tensor to the output of the matrix computation unit (step 604). In this example, the first depth vector is a vector that has *max* entries, with the first z_1 entries being the entries of the first depth vector for the spatial location and the remaining entries being zeros.

[0088] For example, the integrated circuit can move each entry of the first depth vector to a respective sum-in register of a set of registers that stores the outputs of multiplications performed by the matrix computation unit by multiplying the first depth vector by a modified identity weight matrix that has dimensions (*max*) by (*max*) and that includes zeros along the main diagonal until the z_1-th entry of the z_1-th column inclusive.

[0089] The integrated circuit multiplies the first chunk vector of the second depth vector for the given spatial location in the second input tensor by the first shift matrix for the depth concatenation (step 606) to generate a first partially shifted chunk.

[0090] By virtue of the structure of the first shift matrix, the first partially shifted depth vector for a given spatial location is a vector that has (*max*) entries, with the first z_1 entries being zero and the next (*max* - z_1) entries being the first (*max* - z_1) entries of the first chunk vector of the second depth vector for the spatial location.

[0091] The integrated circuit sums the first depth vector and the first partially shifted depth vector to generate an intermediate concatenated depth vector (step 608). For example, the system can add each entry of the first partially shifted depth vector to the corresponding entry of the first depth vector by moving the entry of the first partially shifted depth vector into the sum-in register that stores the corresponding entry of the first depth vector. The intermediate concatenated depth vector is a vector that has (max) entries, with the first z_1 entries being the entries of the first depth vector and the next (max - z_1) entries being the first (max - z_1) entries of the second depth vector for the spatial location.

[0092] The integrated circuit multiplies the depth vector for the given spatial location in the second input tensor by the second shift matrix for the depth concatenation (step 610) to generate a second partially shifted depth vector.

[0093] By virtue of the structure of the second shift matrix, the second partially shifted depth vector for a given spatial location is a vector that has (max) entries, with the first (z_1+ z_2 - max) entries being the last (z_1+ z_2 - max) of the second depth vector for the spatial location and the remaining entries being zero or garbage values.

[0094] The integrated circuit stores the second partially shifted depth vector and the intermediate concatenated depth vector as a representation of a concatenated depth vector for the spatial location (step 612). For example, the integrated circuit can store the second partially shifted depth vector and the intermediate concatenated depth vector in predetermined locations in the unified buffer that are identified in the instructions as locations where the two vectors that represent the concatenated depth vector for the spatial location are to be stored.

[0095] The integrated circuit can perform the process 600 for each spatial location in the input tensors to generate the output of the depth concatenation layer.

[0096] FIG. 7 shows an example of a depth concatenation computation 700 that requires a single shift matrix.

[0097] In the simplified example of FIG. 7, a first input tensor that has dimensions 3 x 3 x 3 is to be depth concatenated with a second input tensor that has dimensions 3 x 3 x 4 to generate a 3 x 3 x 7 output tensor and the maximum vector length that can be processed in one pass by the matrix computation unit is eight, so that the depth of the concatenated output tensor is less than the maximum vector length.

[0098] In part (a) of the computation, the integrated circuit operates on a first depth vector from the first tensor and a corresponding second depth vector from the second tensor. In particular, the first depth vector has as its first 3 entries the 3 entries at a given spatial location in the first input tensor and zeroes as its remaining entries, while the second depth vector has as its first 4 entries the 4 entries at the given spatial location in the second input tensor and zeroes as its remaining entries.

[0099] In the example of FIG. 7, the first depth vector and the second depth vector have been padded with zeroes, but in other examples one or both of the depth vectors can instead be padded with garbage data.

[0100] To perform part (a) of the computation, the integrated circuit multiplies the first depth vector by a modified identity weight matrix using the matrix computation unit to generate another instance of the first depth vector in the output of the matrix computation unit, i.e., to move the first depth vector to the output. In the example of FIG. 7, the modified identity matrix has 1s along the first three entries of the main diagonal and zeroes for the last four entries of the main diagonal. However, because the depth vectors are padded with zeros rather than garbage values, the modified identity weight matrix may instead have other values for the last four entries of the main diagonal and for other entries that are only multiplied by the padded entries of the first depth vector during the multiplication.

[0101] The integrated circuit then multiplies the second depth vector by a shift matrix to generate a shifted depth vector. The shift matrix is a 7 x 7 matrix that has entries that are all zero except for a diagonal row of ones starting at the first entry of the fourth column and ending at the fourth entry of the seventh column. By virtue of being multiplied by the shift matrix, the shifted depth vector has zeroes as its first 3 entries and the 4 entries of the second depth vector as its following 4 entries. As with the modified identity matrix, because the depth vectors are padded with zeros rather than garbage values, the shift matrix may instead have other values than 0 for entries that are only multiplied by the padded entries of the second depth vector during the multiplication.

[0102] In part (b) of the computation, the integrated circuit adds the first depth vector and the shifted second depth vector, i.e., by moving the shifted second depth vector into the output while the first depth vector is in the output, to generate a concatenated depth vector that has the entries of the first depth vector as its first 3 entries and the entries of the second depth vector as its last 4 entries.

[0103] The integrated can perform the example computation for each spatial location in the input tensors to generate a respective concatenated depth vector for each spatial location.

[0104] FIG. 8 shows an example of a depth concatenation computation 800 that requires two shift matrices.

[0105] In the simplified example of FIG. 8, a first input tensor that has dimensions 3 x 3 x 3 is to be depth concatenated with a second input tensor that has dimensions 3 x 3 x 4 to generate a 3 x 3 x 7 output tensor, but the maximum vector length that can be processed in one pass by the matrix computation unit is 5, so that the depth of the depth concatenated output tensor is greater than the maximum vector length.

[0106] In parts (a) and (c) of the computation, the integrated circuit operates on a first depth vector from the first input tensor and two instances of a corresponding second depth vector from the second input tensor. In particular, the first depth vector has as its first 3 entries the 3 entries at a given spatial location in the first input tensor and zeroes as its remaining entries, while the second depth vector has as its first 4 entries the 4 entries at the given spatial location in the second input tensor and a zero as its remaining entry. In the example of FIG. 8, the first depth vector and the second depth vector have been padded with zeroes, but in other examples some or all of the depth vectors can instead be padded with garbage data.

[0107] To perform part (a) of the computation, the integrated circuit multiplies the first depth vector by a modified identity weight matrix using the matrix computation unit to generate another instance of the first depth vector in the output of the matrix computation unit, i.e., to move the first depth vector to the output.

[0108] The integrated circuit then multiplies the second depth vector by a first shift matrix to generate a first partially shifted depth vector. The first shift matrix is a 5 x 5 matrix that has entries that are all zero except for a diagonal row of ones starting at the first entry of the fourth column and ending at the second entry of the fifth column. By virtue of being multiplied by the shift matrix, the first partially shifted depth vector has

zeroes as its first 3 entries and the first 2 entries of the second depth vector as the following 2 entries.

[0109] To perform part (b) of the computation, the integrated circuit then adds the first depth vector and the first partially shifted depth vector, i.e., by moving the first partially shifted depth vector into the output while the first depth vector is in the output, to generate the first chunk of the output depth vector that has as its first 3 entries the entries of the first depth vector and as its last 2 entries the first 2 entries of the second depth vector.

[0110] To perform part (c) of the computation, i.e., to generate the second chunk of the output depth vector, the integrated circuit multiplies the second depth vector by a second shift matrix to generate a second partially shifted depth vector. The second shift matrix is a 5 x 5 matrix that has entries that are all zero except for a diagonal row of ones starting at the fourth entry of the first column and ending at the fifth entry of the second column. By virtue of being multiplied by the shift matrix, the second partially shifted depth vector has the last two entries of the second depth vector as its first two entries and zeroes as the remaining entries.

[0111] To perform part (d) of the computation, the integrated circuit stores the first chunk vector and the second chunk vector as a representation of the concatenated depth vector for the spatial location, e.g., by storing the two vectors in predetermined locations in the unified buffer that are identified in the instructions as locations where the two vectors that represent the concatenated depth vector for the spatial location are to be stored.

[0112] The integrated circuit can perform the example computation for each spatial location in the input tensors to generate a respective concatenated depth vector for each spatial location.

[0113] 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 storage medium for execution by, or to control the operation of, 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. 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.

[0114] The term "data processing apparatus" refers to data processing hardware and 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 also be, or further include, special-purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application-specific integrated circuit). The apparatus can optionally include, in addition to hardware, code that creates an execution environment for computer programs, 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.

[0115] A computer program, which may also be referred to or described as a program, software, a software application, an app, 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 stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A 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 data communication network.

[0116] 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 special-purpose logic circuitry, e.g., an FPGA or an ASIC, or by a combination of special-purpose logic circuitry and one or more programmed computers.

[0117] Computers suitable for the execution of a computer program 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. The central processing unit and the memory can be supplemented by, or incorporated in, special-purpose logic circuitry. 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.

[0118] Computer-readable media suitable for storing computer program instructions and data include all forms of non-volatile 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.

[0119] To provide 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 provide input to the computer. Other kinds of devices can be used to provide 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 device in response to requests received from the web browser. Also, a computer can interact with a user by sending text messages or other forms of message to a personal device, e.g., a smartphone,

running a messaging application, and receiving responsive messages from the user in return.

[0120] 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, a web browser, or an app 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.

[0121] 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. In some embodiments, a server transmits data, e.g., an HTML page, to a user device, e.g., for purposes of displaying data to and receiving user input from a user interacting with the device, which acts as a client. Data generated at the user device, e.g., a result of the user interaction, can be received at the server from the device.

[0122] While this specification contains many specific implementation details, these should not be construed as limitations on the scope of any invention or on the scope 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 be 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.

[0123] 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.

Further examples include methods, systems, and apparatus, including computer programs encoded on computer storage media, for depth concatenation using a matrix computation unit, wherein one of the methods includes: receiving a request to process network inputs to a neural network using an integrated circuit, the neural network comprising a depth concatenation neural network layer; and generating instructions that, when executed by the integrated circuit, cause the integrated circuit to performing operations comprising: for each spatial location in a first input tensor to the depth concatenation layer and a second input tensor to the depth concatenation layer: multiplying, using the matrix computation unit, a second depth vector for the spatial location by a shift weight matrix for the depth concatenation layer to generate a shifted second depth vector; and adding the shifted second depth vector and a first input depth vector for the spatial location to generate a concatenated depth vector.

[0124] 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 some cases, multitasking and parallel processing may be advantageous.

WHAT IS CLAIMED IS:

1. A method comprising:

receiving a request to process network inputs to a neural network using an integrated circuit that performs neural network computations in hardware using a matrix computation unit, the neural network comprising a depth concatenation neural network layer that specifies a concatenation of an input tensor having dimensions x_1 by y_1 by z_1 and an input tensor having dimensions x_1 by y_1 by z_2 along a depth dimension to generate an output tensor having dimensions x_1 by y_1 by (z_1+z_2); and

generating instructions that, when executed by the integrated circuit, cause the integrated circuit to, during processing of a network input by the neural network, generate a layer output tensor that satisfies the specification of the depth concatenation neural network layer by performing operations comprising:

for each spatial location in a first input tensor to the depth concatenation layer and a second input tensor to the depth concatenation layer:

multiplying, using the matrix computation unit, a second depth vector for the spatial location in the second input tensor by a shift weight matrix for the depth concatenation layer to generate a shifted second depth vector that has zeroes as the first z_1 entries and entries of the second depth vector as the last z_2 entries; and

adding the shifted second depth vector and a first input depth vector for the spatial location in the first input tensor to generate a concatenated depth vector, the first input depth vector having entries of the first input depth vector as the first z_1 entries of the first input depth vector and zeroes as the last z_2 entries of the first input depth vector.

2. The method of claim 1, the operations further comprising:

moving the first input depth vector to a set of output sum-in registers of the matrix computation unit; and

wherein adding the shifted second depth vector and the first input depth vector comprises:

moving the shifted second depth vector into the set of output sum-in registers of the matrix computation unit while the first input depth vector is stored in the set of output sum-in registers of the matrix computation unit.

3. The method of claim 2, wherein moving the first input depth vector comprises:

multiplying the first input depth vector by a modified identity weight matrix for the depth concatenation layer using the matrix computation unit.

4. The method of claim 3, further comprising:

generating the modified identity weight matrix for the depth concatenation layer; and

storing the modified identity weight matrix for the depth concatenation layer in a memory accessible to the special-purpose integrated circuit.

5. The method of claim 1, further comprising:

generating the shift weight matrix for the depth concatenation layer; and

storing the shift weight matrix for the depth concatenation layer in a memory accessible to the special-purpose integrated circuit.

6. The method of claim 5, further comprising:

determining that the number of depth dimensions in the output tensor does not exceed a maximum vector length for the matrix computation unit; and

generating the shift weight matrix for the depth concatenation in response to determining that the number of depth dimensions in the output tensor does not exceed the maximum vector length for the matrix computation unit.

7. The method of claim 1, wherein the shift weight matrix for the depth concatenation layer is a $(z_1 + z_2)$ by $(z_1 + z_2)$ matrix having all entries be zero except for a diagonal row of ones starting at the first entry of the z_2-th column of the matrix.

8. A system comprising one or more computers and one or more storage devices storing first instructions that when executed by the one or more computers cause the one or more computers to perform first operations comprising:

receiving a request to process network inputs to a neural network using an integrated circuit that performs neural network computations in hardware using a matrix computation unit, the neural network comprising a depth concatenation neural network layer that specifies a concatenation of an input tensor having dimensions x_1 by y_1 by z_1 and an input tensor having dimensions x_1 by y_1 by z_2 along a depth dimension to generate an output tensor having dimensions x_1 by y_1 by (z_1+z_2); and

generating second instructions that, when executed by the integrated circuit, cause the integrated circuit to, during processing of a network input by the neural network, generate a layer output tensor that satisfies the specification of the depth concatenation neural network layer by performing second operations comprising:

for each spatial location in a first input tensor to the depth concatenation layer and a second input tensor to the depth concatenation layer:

multiplying, using the matrix computation unit, a second depth vector for the spatial location in the second input tensor by a shift weight matrix for the depth concatenation layer to generate a shifted second depth vector that has zeroes as the first z_1 entries followed by entries of the second depth vector; and

adding the shifted second depth vector and a first input depth vector for the spatial location in the first input tensor to generate a concatenated depth vector, the first input depth vector having entries of the first input depth vector as the first z_1 entries of the first input depth vector.

9. The system of claim 8, the second operations further comprising:

moving the first input depth vector to a set of output sum-in registers of the matrix computation unit; and

wherein adding the shifted second depth vector and the first input depth vector comprises:

moving the shifted second depth vector into the set of output sum-in registers of the matrix computation unit while the first input depth vector is stored in the set of output sum-in registers of the matrix computation unit.

10. The system of claim 9, wherein moving the first input depth vector comprises:

multiplying the first input depth vector by a modified identity weight matrix for the depth concatenation layer using the matrix computation unit.

11. The system of claim 10, the first operations further comprising:

generating the modified identity weight matrix for the depth concatenation layer; and

storing the modified identity weight matrix for the depth concatenation layer in a memory accessible to the special-purpose integrated circuit.

12. The system of claim 8, the first operations further comprising:

generating the shift weight matrix for the depth concatenation layer; and

storing the shift weight matrix for the depth concatenation layer in a memory accessible to the special-purpose integrated circuit.

13. The system of claim 12, the first operations further comprising:

determining that the number of depth dimensions in the output tensor does not exceed a maximum vector length for the matrix computation unit; and

generating the shift weight matrix for the depth concatenation in response to determining that the number of depth dimensions in the output tensor does not exceed the maximum vector length for the matrix computation unit.

14. The system of claim 8, wherein the shift weight matrix for the depth concatenation layer is a matrix having all entries be zero except for a diagonal row of ones starting at the first entry of the z_2-th column of the matrix.

15. One or more computer storage media encoded with first instructions that when executed by one or more computers cause the one or more computers to perform first operations comprising:

receiving a request to process network inputs to a neural network using an integrated circuit that performs neural network computations in hardware using a matrix computation unit, the neural network comprising a depth concatenation neural network layer that specifies a concatenation of an input tensor having dimensions x_1 by y_1 by z_1 and an input tensor having dimensions x_1 by y_1 by z_2 along a depth dimension to generate an output tensor having dimensions x_1 by y_1 by (z_1+z_2); and

generating second instructions that, when executed by the integrated circuit, cause the integrated circuit to, during processing of a network input by the neural network, generate a layer output tensor that satisfies the specification of the depth concatenation neural network layer by performing second operations comprising:

for each spatial location in a first input tensor to the depth concatenation layer and a second input tensor to the depth concatenation layer:

multiplying, using the matrix computation unit, a second depth vector for the spatial location in the second input tensor by a shift weight matrix for the depth concatenation layer to generate a shifted second depth vector that has zeroes as the first z_1 entries followed by entries of the second depth vector; and

adding the shifted second depth vector and a first input depth vector for the spatial location in the first input tensor to generate a concatenated depth vector, the first input depth vector having entries of the first input depth vector as the first z_1 entries of the first input depth vector.

16. The computer storage media of claim 15, the second operations further comprising:

moving the first input depth vector to a set of output sum-in registers of the matrix computation unit; and

wherein adding the shifted second depth vector and the first input depth vector comprises:

moving the shifted second depth vector into the set of output sum-in registers of the matrix computation unit while the first input depth vector is stored in the set of output sum-in registers of the matrix computation unit.

17. The computer storage media of claim 16, wherein moving the first input depth vector comprises:

multiplying the first input depth vector by a modified identity weight matrix for the depth concatenation layer using the matrix computation unit.

18. The computer storage media of claim 17, the first operations further comprising:

generating the modified identity weight matrix for the depth concatenation layer; and

storing the modified identity weight matrix for the depth concatenation layer in a memory accessible to the special-purpose integrated circuit.

19. The computer storage media of claim 15, the first operations further comprising:

generating the shift weight matrix for the depth concatenation layer; and

storing the shift weight matrix for the depth concatenation layer in a memory accessible to the special-purpose integrated circuit.

20. The computer storage media of claim 15, wherein the shift weight matrix for the depth concatenation layer is a matrix having all entries be zero except for a diagonal row of ones starting at the first entry of the z_2-th column of the matrix.

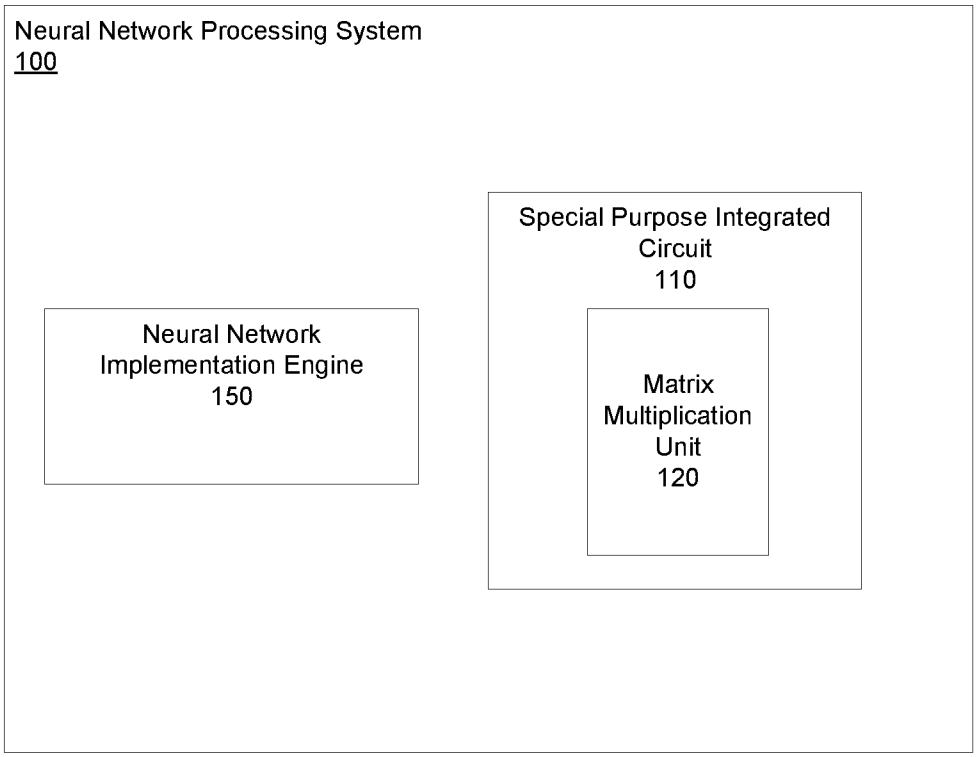

Neural Network Processing System
100
Special Purpose Integrated Circuit
110
Matrix Multiplication Unit
120
Neural Network Implementation Engine
150

FIG. 1

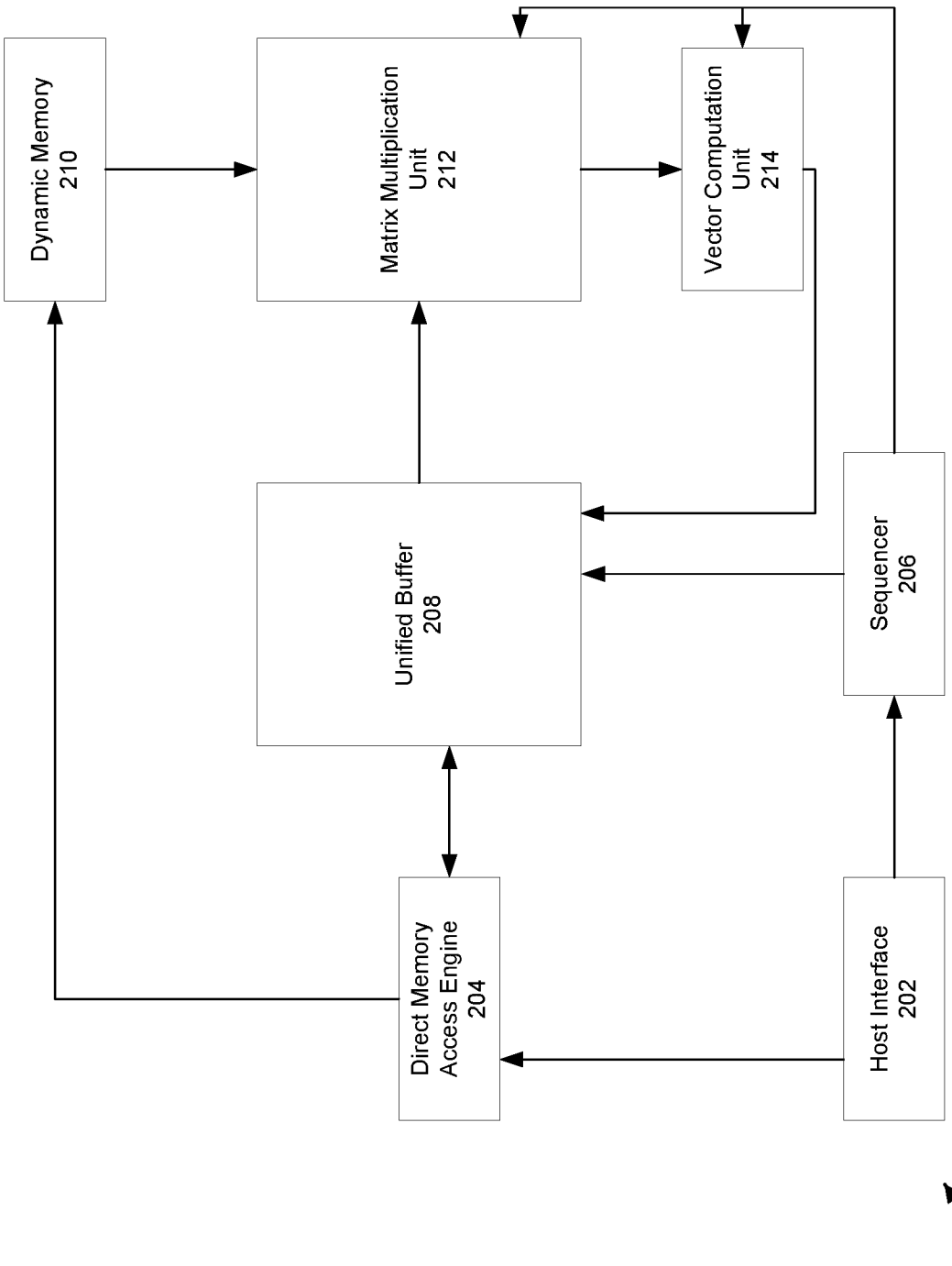

Dynamic Memory
210
Matrix Multiplication
Unit
212
Vector Computation
Unit
214
Unified Buffer
208
Sequencer
206
Direct Memory
Access Engine
204
Host Interface
202
200

FIG. 2

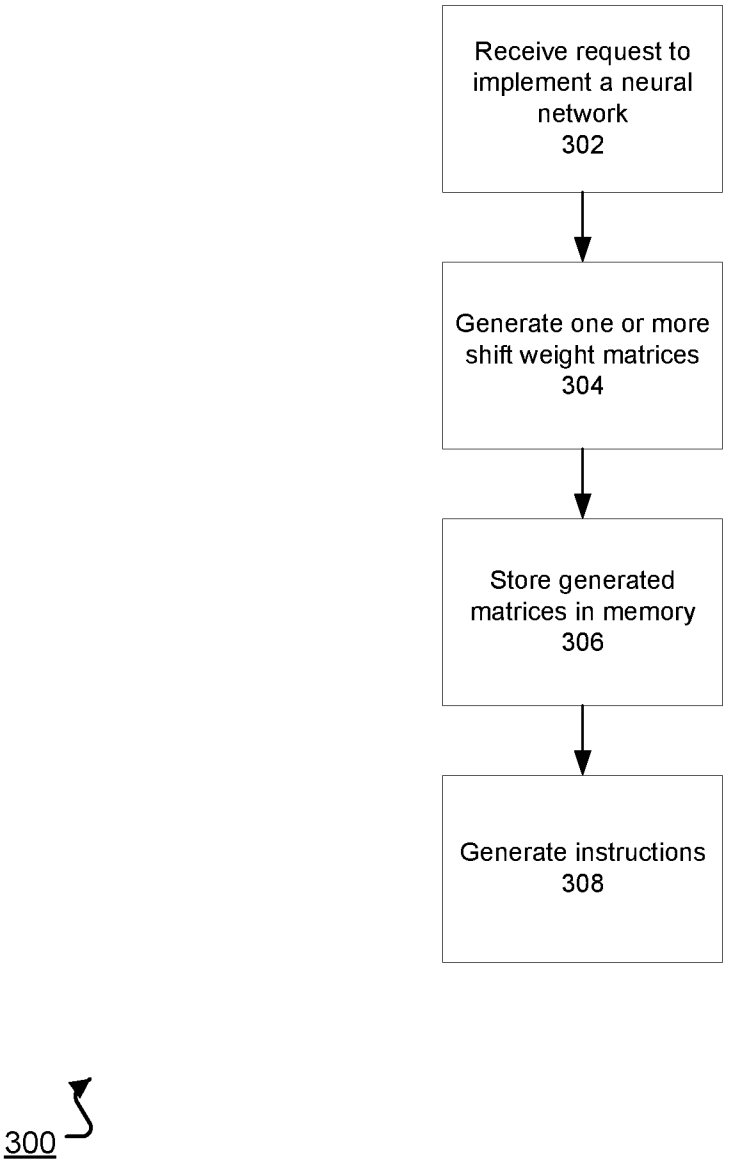

Receive request to implement a neural network
302
Generate one or more shift weight matrices
304
Store generated matrices in memory
306
Generate instructions
308
300

FIG. 3

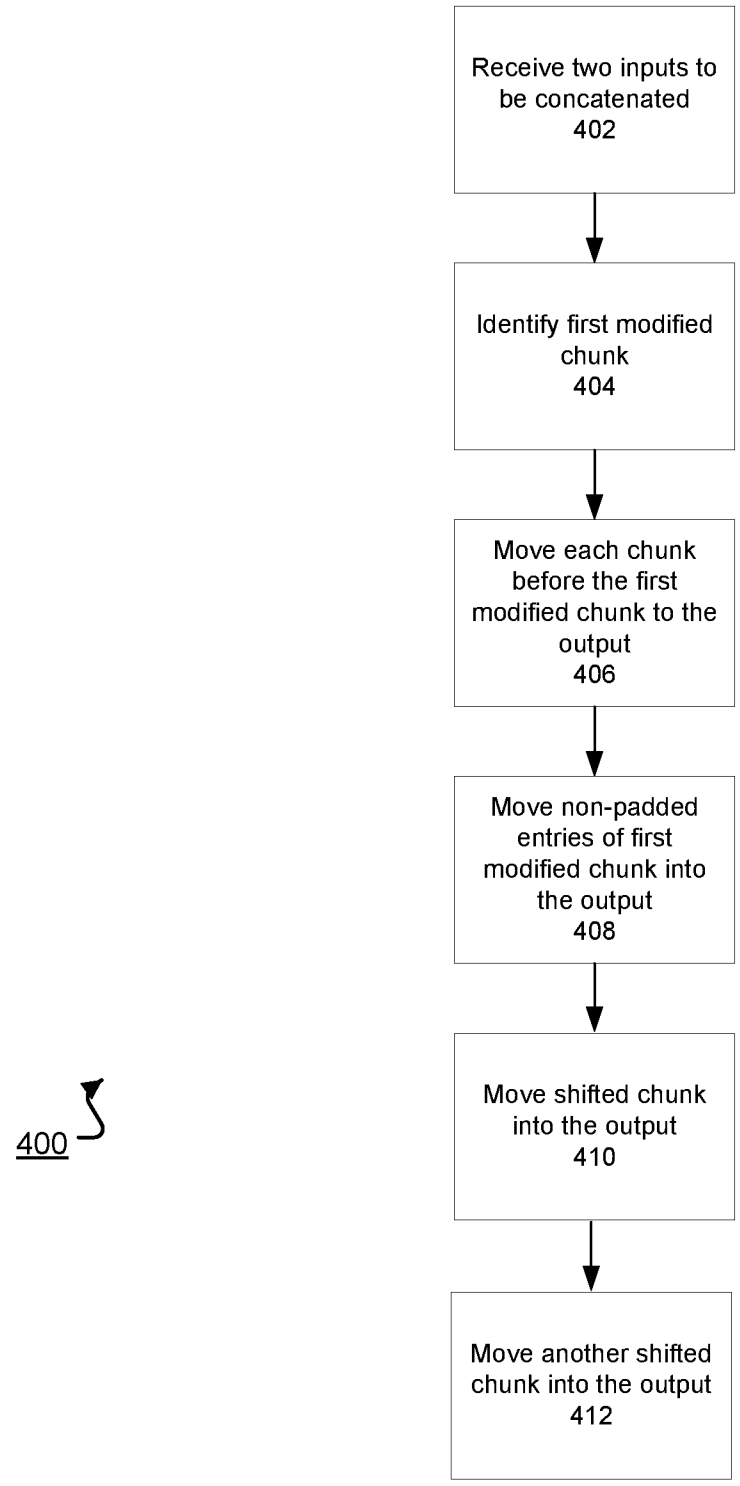

Receive two inputs to be concatenated
402
Identify first modified chunk
404
Move each chunk before the first modified chunk to the output
406
Move non-padded entries of first modified chunk into the output
408
Move shifted chunk into the output
410
Move another shifted chunk into the output
412
400

FIG. 4

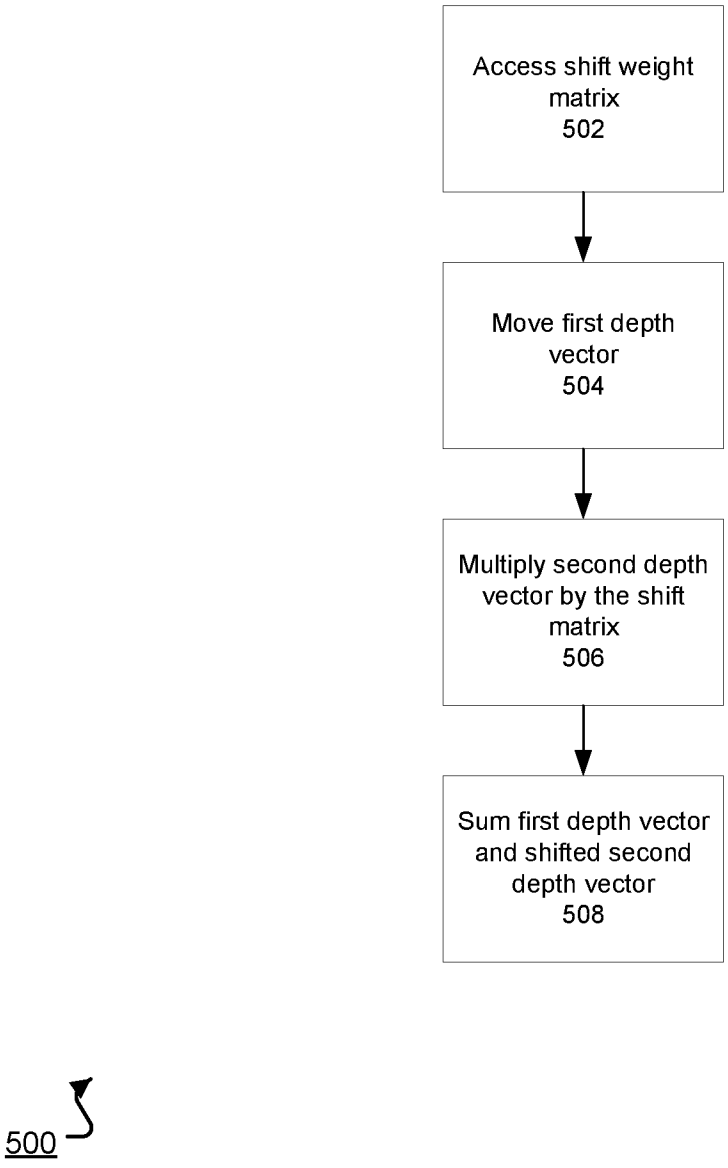

Access shift weight matrix
502
Move first depth vector
504
Multiply second depth vector by the shift matrix
506
Sum first depth vector and shifted second depth vector
508
500

FIG. 5

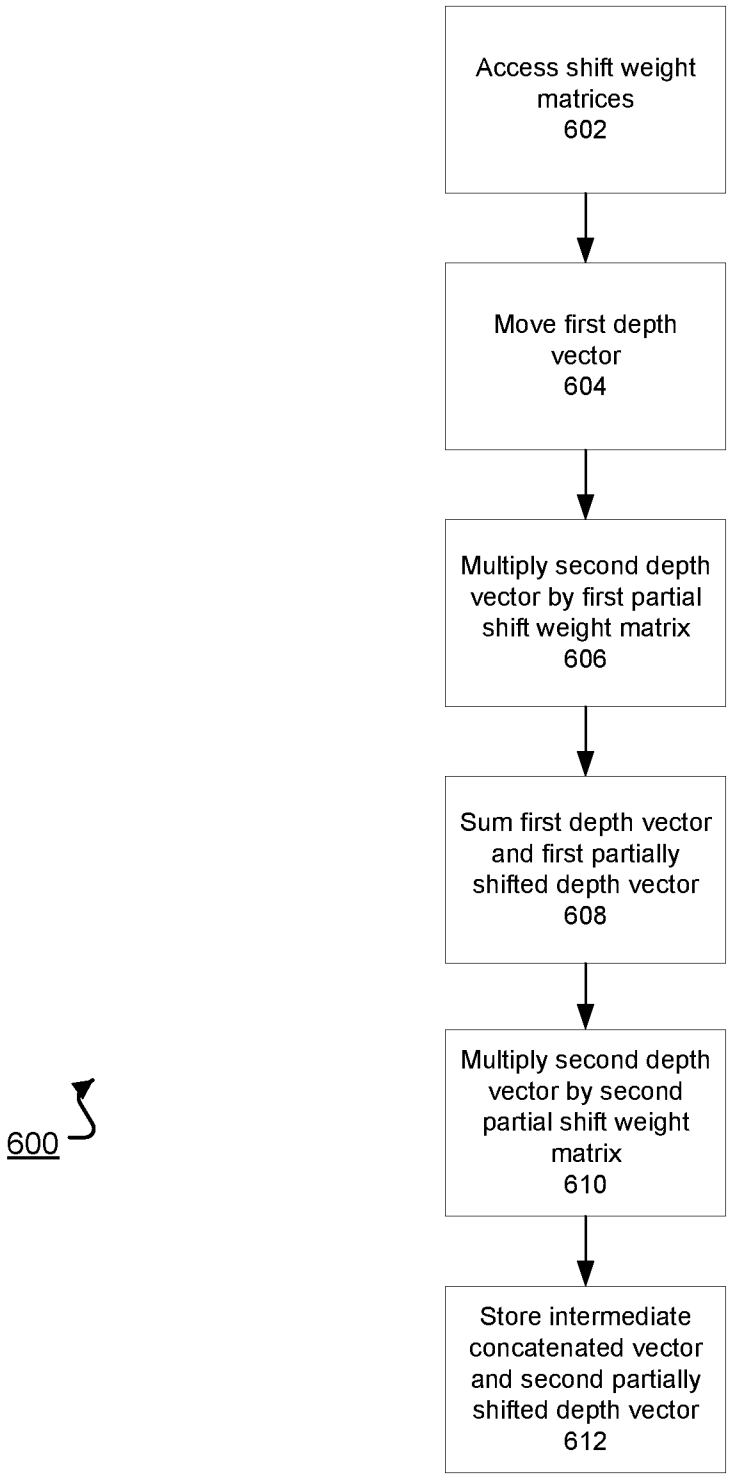
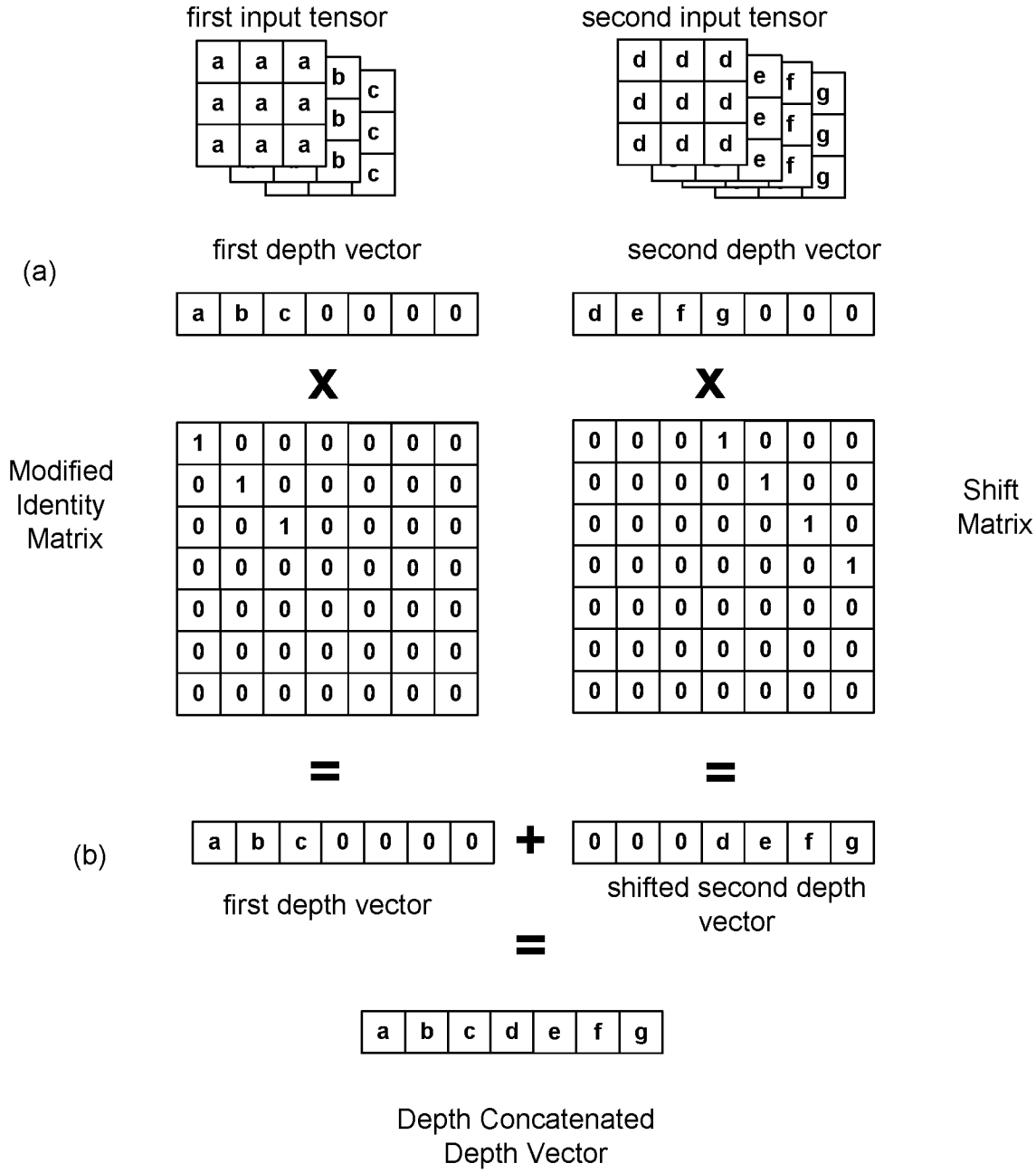

Access shift weight matrices
602
Move first depth vector
604
Multiply second depth vector by first partial shift weight matrix
606
Sum first depth vector and first partially shifted depth vector
608
Multiply second depth vector by second partial shift weight matrix
610
Store intermediate concatenated vector and second partially shifted depth vector
612
600

FIG. 6


first input tensor
second input tensor
(a)
first depth vector
second depth vector
a b c 0 0 0 0
d e f g 0 0 0
X
X
Modified Identity Matrix
Shift Matrix
=
=
(b)
a b c 0 0 0 0
+
0 0 0 d e f g
first depth vector
shifted second depth vector
=
a b c d e f g
Depth Concatenated Depth Vector

700

FIG. 7

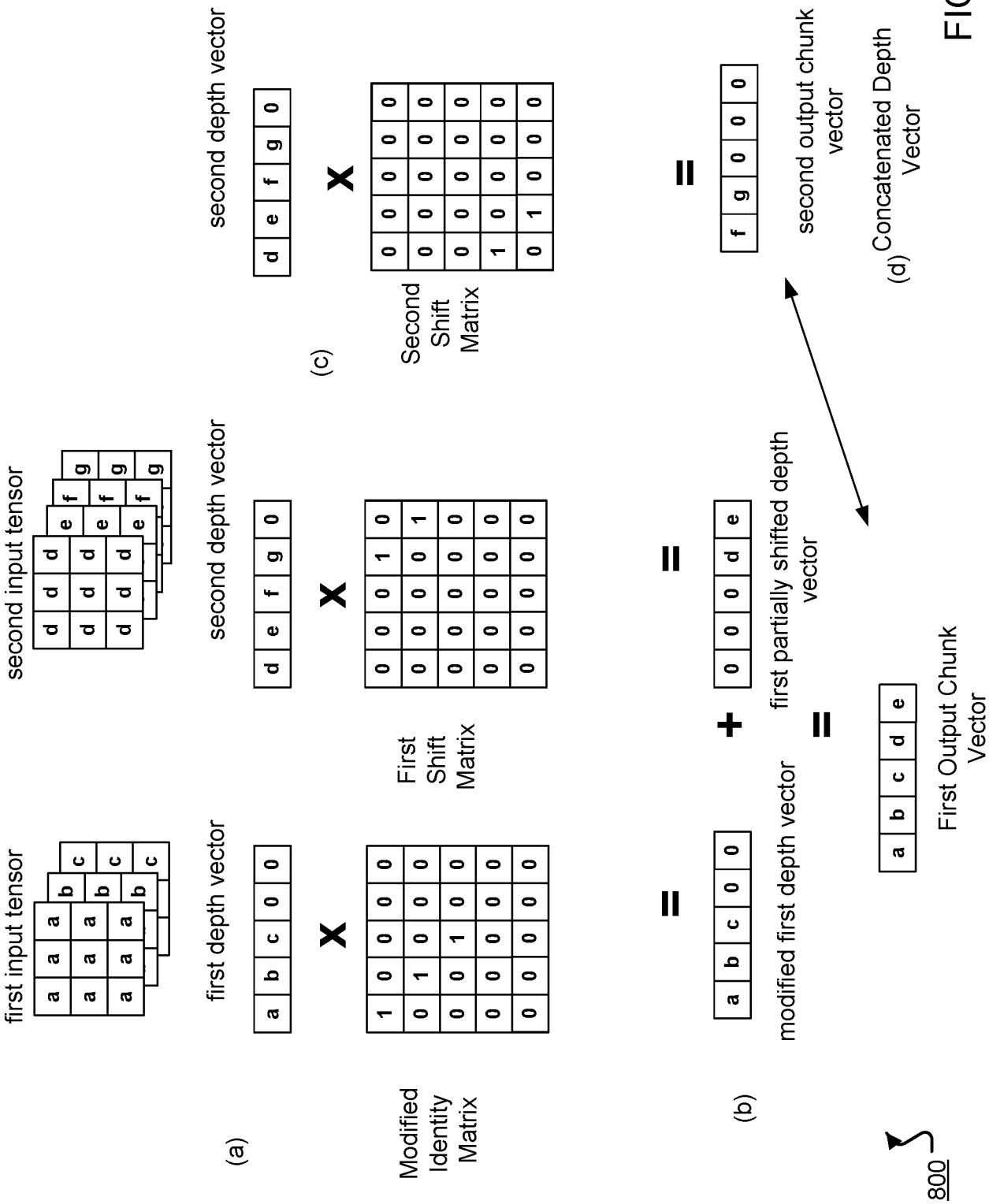

first input tensor
second input tensor
(a)
first depth vector
second depth vector
Modified Identity Matrix
First Shift Matrix
(b)
modified first depth vector
first partially shifted depth vector
(c)
second depth vector
Second Shift Matrix
(d) Concatenated Depth Vector
second output chunk vector
First Output Chunk Vector
800

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2018/021221

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06N3/063
ADD. G06F17/16

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	WO 2016/186810 A1 (GOOGLE INC [US]) 24 November 2016 (2016-11-24) abstract; figures 1-6 page 1 - page 9 page 11 - page 12 -----	1-20
X	Anonymous: "How does the DepthConcat operation in 'Going deeper with convolutions' work?", Cross Validated, 20 July 2016 (2016-07-20), pages 1-3, XP055482575, Retrieved from the Internet: URL:https://stats.stackexchange.com/questions/184823/how-does-the-depthconcat-operation-in-going-deeper-with-convolutions-work [retrieved on 2018-06-08] page 1 - page 3 -----	1-20
	-/--	

[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
11 June 2018	18/06/2018

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	Cilia, Elisa

1

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

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2018/021221

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 2016/033506 A1 (GOOGLE INC [US]) 3 March 2016 (2016-03-03) the whole document -----	1-20

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2018/021221

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2016186810 A1	24-11-2016	CN 107454966 A	08-12-2017
		DE 112016002298 T5	15-02-2018
		DE 202016107439 U1	16-01-2017
		EP 3298546 A1	28-03-2018
		GB 2553052 A	21-02-2018
		KR 20170126999 A	20-11-2017
		TW 201706872 A	16-02-2017
		US 2016342892 A1	24-11-2016
		US 2017103314 A1	13-04-2017
		WO 2016186810 A1	24-11-2016
WO 2016033506 A1	03-03-2016	CN 106471526 A	01-03-2017
		EP 3186753 A1	05-07-2017
		US 9911069 B1	06-03-2018
		US 2016063359 A1	03-03-2016
		US 2017316286 A1	02-11-2017
		US 2018137396 A1	17-05-2018
		WO 2016033506 A1	03-03-2016