US Patent & Trademark Office
Patent Public Search | Text View

PARAMETER CACHING FOR NEURAL NETWORK ACCELERATORS

Abstract

Methods and systems, including computer programs encoded on a computer storage medium. In one aspect, a method includes obtaining data specifying one or more neural networks to be deployed on a neural network hardware accelerator, each of the one or more neural networks having a respective set of parameters, and the neural network hardware accelerator having one or more memories having a memory capacity; determining a maximum amount of the memory capacity that will be in use at any one time during a processing of any of the one or more neural networks by the neural network hardware accelerator; identifying a subset of the parameters of the one or more neural networks that consumes an amount of memory that is less than a difference between the memory capacity and the determined maximum amount of the memory capacity; and storing the identified subset of the parameters.

Inventors: Liu; Jack (Saratoga, CA), Woo; Dong Hyuk (San Jose, CA), Park; Jason Jong Kyu (San Jose, CA), Ashok; Raksit (Campbell, CA)

Applicant: Google LLC (Mountain View, CA)

Family ID: 1000008578361

Appl. No.: 19/025142

Filed: January 16, 2025

Related U.S. Application Data

parent US continuation 16971595 20200820 PENDING US continuation PCT/US2019/067289 20191218 child US 19025142

Publication Classification

Int. Cl.: **G06N3/063** (20230101); **G06F9/50** (20060101); **G06F12/0802** (20160101)

Background/Summary

CROSS-REFERENCE TO RELATED APPLICATIONS [0001] This application is a continuation, and claims priority, of U.S. application Ser. No. 16/971,595, filed Aug. 20, 2020, which is a National Stage Application under 35 U.S.C. § 371 and claims the benefit of International Application No. PCT/US2019/067289, filed on Dec. 18, 2019, the contents of which are incorporated herein by reference in their entirety.

BACKGROUND
[0002] This specification relates to a parameter caching technique for neural network layer computation on neural network hardware accelerators.
[0003] Neural networks are machine learning models that employ one or more layers 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 another layer in the network, e.g., 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.
[0004] A respective set of parameters are stored in the memory of neural network hardware accelerators. Upon a completion of the parameters transition to the computing units of the neural network hardware accelerator, a neural network layer computation, e.g., a convolutional layer computation, executes through multiplication of an input tensor and the stored set of parameters on the computing units of the neural network hardware accelerator.
SUMMARY
[0005] This specification describes a technique of parameter caching for neural network computation in a neural network hardware accelerator. The neural network processing system caches neural network parameters to the neural network accelerator before the execution of the neural network layers. Specifically, the neural network processing system pre-caches the neural network's parameters to available wide memories of computing units in the neural network accelerator, to utilize the abundant memory of the neural network accelerator and thus improve the neural network system efficiency.
[0006] More particularly, this technique includes determining an available memory capacity of computing units in the neural network accelerator. The system identifies a subset of the neural network parameter that is less than an available memory capacity of the computing units. The identified subset of the neural network parameter is then transferred from the data memory to the computing units and stored in the wide memories of the computing units in the hardware accelerator for neural network layer computation. In this technique, the parameter pre-caching is differentiated according to the attributes of the neural network layers. For fully connected neural network layers, this technique splits the parameters based on an amount of corresponding computing units and the available memory capacity of the corresponding computing units. For convolutional neural network layers, this technique differentiates the parameters based on the correspondence to the convolution layers, and transmits the differentiated parameters one-by-one from a data memory to the wide memories of the computing units in the hardware accelerator.
[0007] One innovative aspect of the subject matter described in this specification can be embodied in a method comprising: obtaining data specifying one or more neural networks to be deployed on a neural network hardware accelerator, each of the one or more neural networks having a respective

set of parameters, and the neural network hardware accelerator having one or more memories having a memory capacity; determining a maximum amount of the memory capacity that will be in use at any one time during a processing of any of the one or more neural networks by the neural network hardware accelerator; identifying a subset of the parameters of the one or more neural networks that consumes an amount of memory that is less than a difference between the memory capacity and the determined maximum amount of the memory capacity; and storing the identified subset of the parameters in at least one of the one or more memories of the neural network hardware accelerator.

[0008] Another innovative aspect of the subject matter described in this specification can be embodied in a neural network hardware accelerator comprises a plurality of hardware computing units, each of which is a processing element that has the one or more memories and is configured to perform neural network computations. The hardware computing unit is configured to perform operations comprise: executing, by a processor of the hardware computing unit, a plurality of instructions from the neural network hardware accelerator; accessing, one or more memories of the hardware computing unit, to retrieve the identified subset of the parameters; sending the identified subset of the parameters to one or more memories of other hardware computing units of the neural network hardware accelerator through a ring bus; and storing the identified subset of the parameters from other hardware computing units of the neural network hardware to the one or more memories of the hardware computing unit through the ring bus.

[0009] Another innovative aspect of the subject matter described in this specification can be that the one or more neural networks comprise a fully connected layer, wherein the parameters of the fully connected layer are in the identified subset, wherein the accelerator is configured to distribute the processing of the fully connected layer across a first subset of the hardware computing units, and wherein storing the identified subset comprises: storing a respective partition of the parameters of the fully connected layer in the one or more memories of each hardware computing unit in the first subset.

[0010] Another innovative aspect of the subject matter described in this specification can be that the one or more neural networks comprise a convolutional layer, wherein the parameters of the convolutional layer are in the identified subset, wherein the accelerator is configured to distribute the processing of the convolutional layer across a second subset of the hardware computing units, and wherein storing the identified subset comprises: storing a respective copy of the parameters of the convolutional layer in the one or more memories of each hardware computing unit in the second subset.

[0011] 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.

Description

BRIEF DESCRIPTION OF THE DRAWINGS
[0012] FIG. **1** is a block diagram of an example neural network processing system.
[0013] FIG. **2** is a diagram of an example neural network hardware accelerator for neural network computation.
[0014] FIG. **3** is an example flow chart of process for performing parameter caching on the neural network hardware accelerator.
[0015] FIG. **4** is an example flow chart of process for performing parameter caching for fully connected neural network layer computation on the neural network hardware accelerator.
[0016] FIG. **5** is an example flow chart of process for performing parameter caching for

convolutional neural network layer computation on the neural network hardware accelerator.

[0017] Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0018] Neural network hardware accelerators are typically used to perform operations or computations more efficiently for neural network layer computation.

[0019] A high-performance hardware accelerator, e.g., a Tensor Processing Unit (TPU), has one or more wide memories located on the computing units of the accelerator. Each wide memory may have local storage capacity, e.g., an 8 MB memory, to cache the instructions and parameter data for neural network layer computation. In general, a small portion of the capacity of wide memory on the computing unit is firstly reserved for the model's inference executables, and then the parameter data is stored in whatever space remains. For example, neural network model parameters need to be stored in the computing units' wide memories before a neural network computation executes. In many circumstances, the model parameters are loaded from a host interface of the neural network processing system to the local memory of corresponding computing units, e.g., computing units. Upon the execution of a neural network layer that requires the corresponding model parameters, the neural network processing system will issue an instruction to transmit the model parameters from the parameter memory to a receiver computing unit of the hardware accelerator through a ring bus. The receiver computing unit, based on an issued operation instruction, will then copy the model parameters from the ring bus to its wide memory.

[0020] The aforementioned parameter transition may not provide the best neural network processing system efficiency and neural network hardware accelerator memory utilization. There may be abundant wide memory capacity of the computing units of the accelerator that is underutilized during the neural network layer computation. Furthermore, the external parameter memory may have lower bandwidth compared to the bandwidth of the wide memory on the computing units, which may cause a computation halt while waiting for data transition.

[0021] In this specification, a parameter caching technique is described for high-performance neural network layer computation on hardware accelerators. This parameter caching technique includes obtaining data specifying one or more neural network layers to be deployed on one or more computing units of the neural network accelerator, determining a maximum amount of the wide memory capacity that will be in use at any one time during a processing of any of the one or more neural network layers, and identifying a subset of the parameters that consumes less memory compared to the available wide memory capacity on the computing units. The model parameters are stored in the wide memories of the hardware computing units of the neural network hardware accelerator before the neural network computation executes. More particularly, the wide memory located on the computing units of the accelerator may be compiler-allocated scratchpad memory. The accelerator compiler may be configured to add a small amount of executables inside the model that writes a specific amount of parameter data to the wide memories of the computing units on the accelerator before running an inference. This parameter caching technique enables faster inferencing speed compared to fetching the parameter data from external parameter memory, which was described earlier.

[0022] FIG. **1** shows a block diagram of an example neural network processing system **100**. As shown in FIG. **1**, the neural network processing system **100** includes a neural network accelerator **110**, a host interface **120**, a main memory **130**, a neural network implementation engine **140**, and an output **150**. The neural network processing system **100** may include two or more memories, to store data and instructions for neural network computation.

[0023] In the neural network processing system **100**, the host interface **120** receives and delivers parameters and instructions to the neural network hardware accelerator **110** and the main memory **130**. The parameters may include the weights and biases of the neural network layers. The host interface **120** may also send the instructions to the neural network hardware accelerator **110**, which executes the instructions that control the neural network hardware accelerator **110** to perform the

neural network computations. In some implementations, the transferred instructions regulate dataflow in the neural network hardware accelerator, e.g., how the sets of feature inputs and the sets of filter inputs flow through the circuit.

[0024] The neural network processing system **100** is a system that performs neural network computations using a neural network hardware accelerator **110**. The neural network hardware accelerator **110** is an integrated circuit for performing neural network computations and includes one or more computing units that perform matrix multiplications in the hardware accelerator. The computing units of the neural network hardware accelerator **110**, e.g., the hardware processing units, may also be configured to perform vector multiplications. In some implementations, the neural network accelerator **110** receives task instructions and performs multiply operations that can include multiplying an activation with a weight or other parameters to produce outputs, for example, a partial sum or an output activation, to the output **150**. In some other implementations, the neural network hardware accelerator **110** processes at least one of a tensor operation instruction or a direct memory access operation instruction. In this example, the neural network hardware accelerator **110** may be configured to execute at least one tensor operation instruction and at least one direct memory access operation instruction to perform tensor computations that include multiple linear algebra computations. An example neural network hardware accelerator **110** and the hardware computing units are described in more detail below with reference to FIG. **2**.

[0025] In the neural network processing system **100**, the neural network accelerator **110** is configured to process neural network computations, e.g., matrix multiplication or vector computation, by executing the instructions or parameters that transferred from the host interface **120** or stored in the main memory **130**. The main memory **130** may include one or more memory banks or units, for storing parameters, activations and weights used to process neural network computations for a given neural network layer. In some implementations, the main memory **130** is a volatile memory unit or units. In some other implementations, the main memory **130** is a non-volatile unit memory unit or units such as, for example, read-only memory (ROM) and/or electrically erasable programmable read-only memory (EEPROM). The main memory **130** may also be another form of computer-readable media, such as a floppy disk device, a hard disk device, an optical disk device, a tape device, a flash memory or other similar solid state memory device, or an array of devices including devices in a storage area network or other configurations. In the main memory **130**, parameters or activations can be pre-loaded to respective memory address locations that correspond to elements of an input tensor, to perform computations for a neural network layer. Likewise, weights and other parameters can be pre-loaded to the main memory **130** in respective memory address locations that correspond to elements of a weight tensor or others.

[0026] In general, when the neural network accelerator **110** executes one or more instructions for accessing a particular element of a tensor, it determines the memory address of the element, such that the neural network hardware accelerator **110** may access main memory **130** to read data representing the value of the particular element. For example, parameters stored in main memory **130** may include parameters for a fully connected neural network layer computation or parameter for a convolutional neural network layer computation. To implement a neural network on the neural network hardware accelerator **110**, the neural network processing system **100** includes a neural network implementation engine **140** that is implemented as one or more computer programs on one or more computers in one or more physical locations. The neural network implementation engine **140** generates instructions that, when executed by the neural network hardware accelerator **110**, cause the neural network hardware accelerator **110** to perform the operations specified by the neural network to generate a neural network output **150**.

[0027] FIG. **2** shows a block diagram of an example neural network hardware accelerator **110** for neural network computations. The neural network hardware accelerator **110** generally includes distributed hardware computing units, for the neural network computations. Each of the distributed computing units in the computing unit sets **208** has one or more wide memory **210**. The neural

network hardware accelerator also includes instruction bus **212**, mesh bus **214** and ring bus **216** for the transitions of instructions and parameters to the hardware computing unit sets **208**, respectively. As shown in FIG. **2**, the main memory **130** includes a data memory **204** and an instruction memory **206** to store the parameters and instructions that are transmitted from the host interface **120**. The main memory **130** is configured to execute one or more instructions relating to neural network layer computations in the neural network hardware accelerator **110**, including instructions stored in instruction memory **206**. The instruction memory **206** may store one or more machine readable instructions that are executable by the one or more processors of main memory **130**. The data memory **204** may be any of a variety of data storage media for storing and subsequently accessing a variety of data relating to computations that occur within the neural network hardware accelerator **110**.

[0028] As described earlier on FIG. **1**, the host interface **120** is connected to the main memory **130**, and configured to provide computing unit instructions and parameters to the main memory **130**. In some implementations, computing instructions can be provided to one or more hardware computing units in neural network computing hardware accelerator **110** through the instruction bus **212**, and model parameters can be provided to one or more hardware computing units in neural network hardware accelerator **110** through the ring bus **216**. In some implementations, the computing instructions and model parameters are received by the main memory **130** from the host interface **120** at an initial time and stored in instruction memory **206** and data memory **204**, respectively, for execution by the neural network hardware accelerator **110** at a later time.

[0029] In general, a hardware computing unit in the computing unit sets **208** is a distributed computation core within the neural network hardware accelerator **110**, and is the focal point for neural network computations, for example, tensor computation. Each computing unit is an individual computing hardware that cooperates with other computing units in the computing unit sets **208** to accelerate neural network computations across one or more neural network layers. Although the computing units can share execution of tensor computations associated with a given computing unit instruction, an individual computing unit is a self-contained computational component configured to execute a subset of tensor computations independently relative to other corresponding computing units in the computing unit sets **208**. FIG. **2** presents an example N×2 computing unit layout, with 2N−4 edge computing units are computing units with only three neighbor computing units and 4 corner computing units are computing units with two neighbor computing units. Regarding data flow methodology via mesh bus **214**, in general, every input activation that arrives via mesh bus **214** for a particular computing unit must be committed to one or more wide memory **210** of the computing unit.

[0030] Instruction bus **212** originates from the main memory **130** and provides communications coupling through a bus data path that connects computing units in computing unit sets **208** in a ring back to the main memory **130**. In some implementations, the main memory **130** broadcasts one or more computing instructions via the instruction bus **212**.

[0031] Within the computing unit sets **208**, there may be a bus stop associated with instruction bus **212** configured to examine a header bitmap to determine the instruction types. The computing instruction may be received by the computing unit and subsequently written to an instruction buffer of the computing unit prior to execution of the computing instruction by the computing unit. The instruction buffers may include a first-in first-out (FIFO) control scheme that prioritizes consumption of one or more related computing instructions. Thus, under this FIFO control scheme, computing instructions of the same type/subtype will always be executed in the order in which the computing unit instruction arrived on the instruction bus.

[0032] The mesh bus **214** provides a data communication path that between the adjacent computing units in the computing unit sets **208** in both lateral and vertical dimensions. In various implementations, mesh bus **214** may be used to transport input parameters, for example, activation quantities, between one or more wide memories **210** in adjacent computing units. As shown in FIG.

2, mesh bus **214** is not configured to allow direct forwarding of input data to non-adjacent computing units.

[0033] The ring bus **216** originates from main memory **130** and provides communications coupling through a bus data path that connects computing unit sets **208** in a ring back to the main memory **130**. In various implementations, the ring bus **216** generally connects or couples wide memory **210** units of all computing units in the computing unit sets **208**. Thus, a payload width of ring bus **216** corresponds to the width of the wide memory **210** disposed within each computing units. Furthermore, the ring bus **216** may include a bitmap header indicating the computing units that need to consume the computing instructions and model parameters communicated via ring bus **128**.

[0034] In some implementations, the main memory **130** may be used to pop the data from ring bus stops and forward the data to a ring bus stop in a next computing unit in the ring. In various implementations, the wide memory **210** of each computing units in the neural network hardware accelerator **110** is a single-ported single-banked memory within each computing unit of the computing unit sets **208**. The main memory **130** may cause the data to be committed to one or more wide memory **210** units of the computing unit if such action is required by instructions in the bitmap header. The address of the one or more wide memory **210** units to which the data needs to be written may be generated by direct memory access operation instructions within the particular computing unit.

[0035] In various implementations, each computing unit in the computing unit sets **208** can either be a producer of data or a consumer of data. When a computing unit is a producer of data, the computing unit reads the data from one or more of its wide memory **210** and multicasts the data over ring bus **216** for consumption by one or more adjacent computing units. When a computing unit is a consumer of data, the computing unit receives and writes the data to one or more wide memory **210** of the computing units and forwards the data for consumption by one or more other computing units. Regarding the movement of data via ring bus **216**, there typically will only be one producer/master data transition on ring bus **216** at any given time. The direct memory access operation instruction execution order (e.g., FIFO control scheme) in all computing units will ensure there is only one producer/master data transition on ring bus **216** at a given time to avoid data transition conflicts.

[0036] In various implementations, it is required to ensure there is only one producer/master of data on ring bus **216** at a given time when there are local multicast groups connected via ring bus **216** that do not have an overlapping region on the ring bus **216**. An important requirement of this dual master multicast methodology is that different multicast groups must not be allowed to see each other's data packets because packet overlapping may occur and lead to one or more data computation errors.

[0037] In various implementations, the neural network processing system **100** performs operations as follows. The host interface **120** provides one or more instructions to the main memory **130** and defines one or more direct memory access operations that occur for a given neural network computation. Descriptors associated with instructions fed to the main memory **130** include information required to facilitate large scale neural network computations associated with the hardware computing unit sets **208**. In general, the main memory **130** receives, from the host interface **120**, computing unit instructions or model parameters (i.e., input activations, computing unit instructions, and model weights) for executing neural network computations for a particular layer of a neural network. The main memory **130** may then cause the instructions or model parameters to be multicast to the computing units in a data flow manner defined by the instructions. As discussed above, the computing units consuming an instruction may then initiate a broadcast of a new/subsequent instruction to another computing unit based on bitmap data in the instruction header.

[0038] In various implementations, the model parameters may be stored in one or more wide memories **210** of the computing units, before a neural network computation can operate in the

neural network hardware accelerator **110**. In some implementations, when the host interface **120** transfers the parameters to the neural network hardware accelerator **110**, the main memory **130** receives and writes the parameters to its data memory **204**. Upon the execution of neural network computation for a particular neural network layer, the host interface **120** will provide an issued instruction, e.g., a RingInfeed instruction. The issued instruction initiates the operations to transfer the parameters from the data memory **204** to particular computing units within the computing unit sets **208**, via the ring bus **216** and the mesh bus **214**. The particular receiver computing units then issue a direct memory access operation instruction to receive and write the transferred parameters from the ring bus **216** to its one or more wide memories of the receiver computing units.

[0039] Regarding the data storage in the neural network hardware accelerator **110**, each computing units of the computing unit sets **208** may store a subset of the input instructions and parameters needed to compute a subset of output activations that are assigned to the particular computing unit. A neural network computation begins when required input activations, parameters/weights and computation instructions are available in the computing units. The computation ends in the computing units when all the computation operations defined by the instruction set are completed and pre-activation functions are applied to the results (i.e., output activations) of the neural network computation.

[0040] In general, as described earlier, the parameters need to be transmitted and stored in one or more wide memories **210** of the computing units, before executing a neural network computation instruction. The parameters may be loaded through host interface **120** to the data memory **204** within the main memory **130**. Upon an execution of a neural network layer that required the parameters, the neural network layer issues an instruction to transmit model parameters from the data memory **204** to the one or more particular computing units through the ring bus **216**.

[0041] This application presents a method to cache the parameters to the neural network hardware accelerator **110** before executing any input neural network layers. This parameter caching utilizes abundant bandwidth of wide memory **210** to improve the system efficiency and utilization for high-performance neural network computations, and is introduced more in details in following figures.

[0042] FIG. **3** is an example flow chart of process for performing parameter caching on the neural network hardware accelerator. This example may be performed using the neural network hardware accelerator **110** of FIGS. **1** and **2**.

[0043] The neural network hardware accelerator **110** obtains data specifying one or more neural network layers to be deployed on one or more computing units of the neural network hardware accelerator **110** (step **302**). Each of the one or more neural network layers have a respective set of parameters, and the neural network hardware accelerator **110** has one or more memories having a memory capacity. The set of parameters may be weights and biases of the neural network. The memories of the neural network hardware accelerator may be the one or more wide memories **210** on the computing units of the computing unit sets **208**. In some implementations, the host interface **120** receives the instructions of executing a particular neural network layer and the model parameters on the neural network hardware accelerator **110**. The computing instruction and model parameters are transmitted to the main memory **130**, and stored in the instruction memory **206** and data memory **204**, respectively.

[0044] In the next step, the neural network processing system **100** determines a maximum amount of the wide memory **210** capacity that will be in use at any one time during a processing of any of the one or more neural network layers by the neural network hardware accelerator **110** (step **304**). In particular, the main memory **130** may determine the amount of available capacity of wide memory **210** of each computing unit in the computing unit sets **208** during a processing of one or more neural network layers. In various implementations, the computing units may have abundant wide memory **210**, e.g., an 8 MB memory, which is underutilized during the neural network computation. In addition, the wide memory **210** of the computing units may provide higher bandwidth than that of the main memory **130**.

[0045] In some implementations, the neural network processing system **100** identifies a subset of the parameters of the one or more neural network layers that consumes an amount of memory less than a difference between the memory capacity and the determined maximum amount of the memory capacity (step **306**). In this illustration, the neural network processing system **100** identifies a subset of the parameters for neural network computations in one or more computing units in the computing unit sets **208**, by splitting the parameters based on the amount of one or more computing units and the available capacity of wide memory **210** of the one or more computing units. the neural network processing system **100** may also identify a subset of the parameters that correspond to one or more particular neural network layers, and send the subset of the parameters to one or more computing units of the computing unit sets **208** to perform neural network computations, e.g., convolutional computations.

[0046] In the next step, the neural network hardware accelerator **110** stores the identified subset of the parameters in at least one of the one or more wide memories **210** of the hardware computing units (step **308**). In some implementations, the parameters are transmitted from the data memory **204** to one or more particular computing units of the computing unit sets **208** via the ring bus **216**. As described earlier, the ring bus **216** may include a bitmap header indicating the one or more computing unit that need to consume the data comprising computing unit instructions and model parameters. Upon receiving the instructions, the one or more receiving computing units will zero (i.e., clear out) position data indicated in the bitmap header that is unique to the receiving computing unit, before forwarding the parameter data on to other adjacent computing units. In various implementations, the bandwidth of the ring bus **216** corresponds to the bandwidth of the wide memory **210** disposed on computing unit sets **208**.

[0047] In some implementations, when the neural network computation instructions are executing, the neural network processing system **100** may issue direct memory access operation instructions to one or more host computing units in the computing unit sets **208**, so as to send the parameters stored in the host computing units to other receiver computing units through the ring bus **216** or mesh bus **214**. As a response, the receiver computing units may issue the direct memory access operation instruction to copy the parameter transmitted from ring bus **216** to its wide memory **210**. After storing the identified subset of parameters to the corresponding computing units, the neural network hardware accelerator starts to execute neural network computations corresponding to the particular neural network layers (step **310**).

[0048] In this illustration, the neural network may be made of multiple convolutional layers consisting of filters, non-linear threshold, and pooling operations. Once desired number of convolutional layers are configured, the output is then vectorized and passed through a fully connected layer, after which the desired output is produced. A convolutional layer usually has fewer parameters as compared to that of a fully connected layer. A fully connected layer produces a same output size and does not use a bias for the neural network computation. In various implementations, a neural network may contain one or more fully connected layers sharing an amount of parameters for computation at one or more computing units of the neural network hardware accelerator **110**.

[0049] In some implementations, a parameter caching class may be created and deployed to support the described parameter caching task in the neural network hardware accelerator system **110**. For example, given an available wide memory **210** in each computing unit, the parameter caching class may traverse each neural network layer, and determine the wide memory **210** usage in each computing units for parameters caching. The implementation of parameter caching is in a first-come-first-serve manner. In some other examples, the implementation of parameter caching is fine-tuned for tasks that are more complicated.

[0050] For a given neural network layer, it is preferable to allocate its model parameters in a single computing unit. This configuration will reduce the instruction count on performing the corresponding neural network computation and simplify the management of the synchronization

flag of the neural network hardware accelerator **110**.

[0051] In neural network processing system **100**, compiler support is enabled for parameters caching. For example, a code generator may generate two programs: one program to load parameters from host interface **120** to the wide memory **210** of the computing units, and another program to execute an existing model of a neural network layer. The run-time may use various functions to get the corresponding binaries of these two programs. In some implementations, the required parameter caching are not ordered contiguously. For example, there may be five neural network layers with corresponding model parameters for neural network computation. The execution order of these five layers are: [0052] A.fwdarw.B.fwdarw.C.fwdarw.D.fwdarw.E

If layers A, C, and E have parameters cached, then there is a need to generate five direct memory access descriptors to cache the parameters. In an alternative way, the parameter packing can be ordered as [0053] A.fwdarw.C.fwdarw.E.fwdarw.B.fwdarw.D

wherein only one direct memory access descriptor is needed for parameter packing in a sequence of A.fwdarw.C.fwdarw.E, and another direct memory access descriptor is used for parameter packing of non-cached parameters in a sequence of B.fwdarw.D.

[0054] As descripted earlier, there is only one host computing unit that is allowed to send out parameters through the ring bus **216**. In a case of two or more host computing units are sending parameters simultaneously to a same receiver computing unit through the ring bus **216**, the receiver computing unit could get wrong parameters, and data packages with no active receiver computing units will float in the ring bus **216** and cause system hang.

[0055] FIG. **4** is an example flow chart of process for performing parameter caching for fully connected neural network layer computation on the neural network hardware accelerator **110**.

[0056] The neural network hardware accelerator **110** obtains data specifying one or more fully connected layers of the neural network to be deployed on one or more computing units of the neural network hardware accelerator **110**, wherein each of the fully connected neural network layers share a respective set of parameters. The host interface **120** receives the instructions of executing one or more fully connected neural network layers and the model parameters. The instructions and model parameters are transmitted to the main memory **130**, and stored in the instruction memory **206** and data memory **204**, respectively.

[0057] The neural network processing system **100** divides the set of parameters for the computation of fully connected layers on the neural network hardware accelerator **110** (step **402**). In various implementations, the neural network processing system **100** identifies a subset of the set parameters for fully connected layer computations that correspond to a computing unit in the computing unit sets **208**, by splitting the parameters based on the amount of corresponding computing units and the available wide memory **210** capacity of the corresponding computing units. The neural network processing system **100** may have a global view on the parameters corresponding to the neural network layers and models. It will determine the available capacity of wide memory **210** in each computing unit, and invoke parameter caching method to allocate wide memory **210** for the parameter caching.

[0058] In some implementations, based on no single computing unit being available to store the set of parameters into its wide memory **210**, the parameter may be split in Zout block granularity and distributed across various computing units. An ultimate optimization may be achieved by maximizing the parameter caching efficiency in terms of the capacity of wide memory **210**. For example, a specific model may have a fully connected layer with 1 MB parameters. However, the available wide memory **210** of each computing unit is smaller than 1 MB, thus the model parameter does not fit in a single computing unit of the neural network hardware accelerator **110**. As a result, the fully connected layer's parameter will be divided and cached to different computing units. In some implementations, the divided parameters are distributed to particular computing units to improve the tensors computation efficiency, and there is no need to unicast the divided parameters across the hardware computing unit set **208**.

[0059] In the next step, the neural network processing system **100** assigns the divided subsets of fully connected layers parameters to the one or more wide memories **210** of the one or more computing units on the neural network hardware accelerator **110** (step **404**).

[0060] The neural network hardware accelerator **110** sends the divided parameters one-by-one from the data memory **204** to the one or more wide memories **210** of the identified one or more computing units via ring bus **216** (step **406**). In various implementations, the fully connected neural network layers issue a RingInfeed instruction to transmit the subset of the parameters from the data memory **204** to the one or more particular computing units for executing of the corresponding neural network computation, through the ring bus **216**.

[0061] In the next step, the receiver computing units of the computing unit sets **208** receive the subset of parameters and write the parameters to the one or more wide memories **210** of the computing units (step **408**). The neural network processing system **100** may issue direct memory access operation instructions to one or more host computing units for storing parameters from the host computing units to other adjacent receiver computing units, through the ring bus **216** or the mesh bus **214**. As a response, the receiver computing units may issue a direct memory access operation instruction to copy the parameter transmitted from host computing units to its wide memory **210** (step **410**).

[0062] After storing the identified subsets of parameters to the corresponding computing units, the neural network hardware accelerator **110** starts to execute neural network computations corresponding to the one or more fully connected neural network layers. In various implementations, a particular neural network layer may have more than one models pending for parameter caching. Instructions indicating a list of the models in descending priority order may be sent through the host interface **120** to the main memory **130** of the hardware computing units for execution. In response to the instructions, models that are more important may be processed earlier to have the corresponding parameters cached in the wide memory **210** of one or more computing units.

[0063] FIG. **5** is an example flow chart of process for performing parameter caching for convolutional neural network layer computation on the neural network hardware accelerator **110**. Convolutional layers are a class of feed-forward neural network layers that are suited for operations and computations on two-dimensional data such as images. A convolution is a linear operation that involves multiplication of model parameters, for example, an array of input data and a two-dimensional array of weights. The neural network hardware accelerator **110** delivers the model parameters to one or more wide memories **210** of one or more computing units for a particular convolutional neural network layer computation.

[0064] Upon receiving the instructions, the main memory **130** specifies a set of model parameters for one or more convolutional neural network layers (step **502**). The model parameters may be transmitted from the host interface **120** and stored in the data memory **204**.

[0065] The neural network processing system **100** may differentiate the parameters to subset parameters based on the correspondence to one or more convolutional layers (step **504**). In some implementations, the subset parameters that corresponds to a particular convolutional neural network layer do not fit in the wide memories of the corresponding computing unit. As a result, the subset parameters are further split and transmitted to more computing units for the convolutional computation.

[0066] In the next step, the differentiated subset parameters are transmitted one-by-one from the data memory **204** to the wide memory **210** of the computing units via ring bus **216** (step **506**). In some implementations, the parameter caching implementation is in a simple first-come-first-serve manner. In some other implementations, the parameter caching implementation algorithm is fine-tuned for tasks that are more complicated.

[0067] The computing units may copy the transmitted subsets parameters and write the subsets of parameters to its wide memory **210** (step **508**). In some implementations, based on the subset

parameters being not fit to the wide memory **210** of the computing unit, the computing unit receives parameters from adjacent host computing units and stores the parameters to its wide memory **210** for completing a portion of the convolution computation.

[0068] After storing the differentiated subset parameters to the corresponding computing units, upon receiving the tensor operation instruction, the computing units start to execute neural network computations corresponding to the one or more convolutional neural network layers (step **510**).

[0069] 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.

[0070] 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.

[0071] 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.

[0072] 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).

[0073] 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.

[0074] 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.

[0075] 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.

[0076] 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.

[0077] 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.

[0078] 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.

[0079] 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.

[0080] 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. (canceled)

2. A method implemented using a special-purpose hardware integrated circuit of a neural network hardware accelerator configured to implement a neural network comprising a plurality of neural network layers, the method comprising: dividing a set of parameters for the plurality of neural network layers into a plurality of subsets of parameters, wherein the plurality of subset of parameters comprises a first subset of parameters and a second subset of parameters; generating an assignment that maps the plurality of subset of parameters to a respective plurality of compute units of the neural network hardware accelerator, wherein the first subset of parameters is assigned to a first compute unit and the second subset of parameters is assigned to a second compute unit; and based on the assignment, transmitting, from a data memory where the plurality of subsets of parameters are stored, (**1**) the first subset of the parameters for storage within a wide memory associated with the first compute unit and (**2**) the second subset of parameters for storage within a wide memory associated with the second compute unit.

3. The method of claim 2, further comprising: receiving, over a ring bus, (**1**) the first subset of the parameters for storage within a wide memory associated with the first compute unit and (**2**) the second subset of parameters for storage within a wide memory associated with the second compute unit; and writing the first subset of parameters to the wide memory associated with the first compute unit and writing the second subset of parameters to the wide memory associated with the second compute unit.

4. The method of claim 2, further comprising: subsequent to writing the first and second subset of parameters to their respective wide memories, causing the neural network hardware accelerator to perform processing of two or more neural network layers.

5. The method of claim 2, further comprising: comparing a memory capacity of the data memory with a memory capacity of the wide memories of the first and second compute units.

6. The method of claim 3, further comprising: accessing the first subset of parameters from the wide memory of the first compute unit; sending the first subset of parameters to another compute unit of the neural network hardware accelerator using the ring bus; and storing the first subset of parameters in the wide memory of the respective compute unit.

7. The method of claim 2, further comprising: ranking layers of the plurality of neural network layers according to execution priority; and identifying a respective set of parameters of a highest-ranked neural network layer as the first subset of parameters.

8. The method of claim 2, wherein one or more of the plurality of neural network layers comprise a fully connected layer, wherein parameters of the fully connected layer are included among the first subset of parameters, wherein the neural network hardware accelerator is configured to distribute processing of the fully connected layer across a first subset of the compute units, and wherein storing the first subset of parameters comprises: storing a respective partition of the first subset of parameters of the fully connected layer in the wide memories of each computing unit in the first subset.

9. The method of claim 2, wherein one or more of the plurality of neural network layers comprise a

convolutional layer, wherein parameters of the convolutional layer are included among the second subset of parameters, wherein the neural network hardware accelerator is configured to distribute processing of the convolutional layer across a second subset of the compute units, and wherein storing the second subset of parameters comprises: storing a respective copy of the parameters of the convolutional layer in the wide memories of each computing unit in the second subset.

10. A system comprising: a special-purpose hardware integrated circuit of a neural network hardware accelerator configured to implement a neural network comprising a plurality of neural network layers; a processing unit and one or more storage devices storing instructions that are executable by the processing unit to cause performance of operations comprising: dividing a set of parameters for the plurality of neural network layers into a plurality of subsets of parameters, wherein the plurality of subset of parameters comprises a first subset of parameters and a second subset of parameters; generating an assignment that maps the plurality of subset of parameters to a respective plurality of compute units of the neural network hardware accelerator, wherein the first subset of parameters is assigned to a first compute unit and the second subset of parameters is assigned to a second compute unit; based on the assignment, transmitting, from a data memory where the plurality of subsets of parameters are stored, (**1**) the first subset of the parameters for storage within a wide memory associated with the first compute unit and (**2**) the second subset of parameters for storage within a wide memory associated with the second compute unit.

11. The system of claim 10, further comprising: receiving, over a ring bus, (**1**) the first subset of the parameters for storage within a wide memory associated with the first compute unit and (**2**) the second subset of parameters for storage within a wide memory associated with the second compute unit; and writing the first subset of parameters to the wide memory associated with the first compute unit and writing the second subset of parameters to the wide memory associated with the second compute unit.

12. The system of claim 10, further comprising: subsequent to writing the first and second subset of parameters to their respective wide memories, causing the neural network hardware accelerator to perform processing of two or more neural network layers.

13. The system of claim 10, further comprising: comparing a memory capacity of the data memory with a memory capacity of the wide memories of the first and second compute units.

14. The system of claim 11, further comprising: accessing the first subset of parameters from the wide memory of the first compute unit; sending the first subset of parameters to another compute unit of the neural network hardware accelerator using the ring bus; and storing the first subset of parameters in the wide memory of the respective compute unit.

15. The system of claim 10, further comprising: ranking layers of the plurality of neural network layers according to execution priority; and identifying a respective set of parameters of a highest-ranked neural network layer as the first subset of parameters.

16. The system of claim 10, wherein one or more of the plurality of neural network layers comprise a fully connected layer, wherein parameters of the fully connected layer are included among the first subset of parameters, wherein the neural network hardware accelerator is configured to distribute processing of the fully connected layer across a first subset of the compute units, and wherein storing the first subset of parameters comprises: storing a respective partition of the first subset of parameters of the fully connected layer in the wide memories of each compute unit in the first subset.

17. The system of claim 10, wherein one or more of the plurality of neural network layers comprise a convolutional layer, wherein parameters of the convolutional layer are included among the second subset of parameters, wherein the neural network hardware accelerator is configured to distribute processing of the convolutional layer across a second subset of the compute units, and wherein storing the second subset of parameters comprises: storing a respective copy of the parameters of the convolutional layer in the wide memories of each compute unit in the second subset.

18. A special-purpose hardware integrated circuit of a neural network hardware accelerator configured to implement a neural network comprising a plurality of neural network layers, the hardware integrated circuit comprising: a processing unit; a memory having a maximum memory capacity; and a machine-readable storage device storing instructions that are executable by the processing unit to cause performance of operations comprising: dividing a set of parameters for the plurality of neural network layers into a plurality of subsets of parameters, wherein the plurality of subset of parameters comprises a first subset of parameters and a second subset of parameters; generating an assignment that maps the plurality of subset of parameters to a respective plurality of compute units of the neural network hardware accelerator, wherein the first subset of parameters is assigned to a first compute unit and the second subset of parameters is assigned to a second compute unit; based on the assignment, transmitting, from a data memory where the plurality of subsets of parameters are stored, (**1**) the first subset of the parameters for storage within a wide memory associated with the first compute unit and (**2**) the second subset of parameters for storage within a wide memory associated with the second compute unit.

19. The circuit of claim 18, the operations further comprising: receiving, over a ring bus, (**1**) the first subset of the parameters for storage within a wide memory associated with the first compute unit and (**2**) the second subset of parameters for storage within a wide memory associated with the second compute unit; and writing the first subset of parameters to the wide memory associated with the first compute unit and writing the second subset of parameters to the wide memory associated with the second compute unit.

20. The circuit of claim 18, the operations further comprising: subsequent to writing the first and second subset of parameters to their respective wide memories, causing the neural network hardware accelerator to perform processing of two or more neural network layers.

21. The circuit of claim 18, the operations further comprising: comparing a memory capacity of the data memory with a memory capacity of the wide memories of the first and second compute units.