US 20190205738A1

(19) United States
(12) Patent Application Publication
BANNON et al.

(10) Pub. No.: US 2019/0205738 A1
(43) Pub. Date: Jul. 4, 2019

(54) SYSTEMS AND METHODS FOR HARDWARE-BASED POOLING

(71) Applicant: **Tesla, Inc.**, Palo Alto, CA (US)

(72) Inventors: **Peter Joseph BANNON**, Woodside, CA (US); **Kevin Altair Hurd**, Redwood City, CA (US)

(73) Assignee: **Tesla, Inc.**, Palo Alto, CA (US)

(21) Appl. No.: **15/862,369**

(22) Filed: **Jan. 4, 2018**

Publication Classification

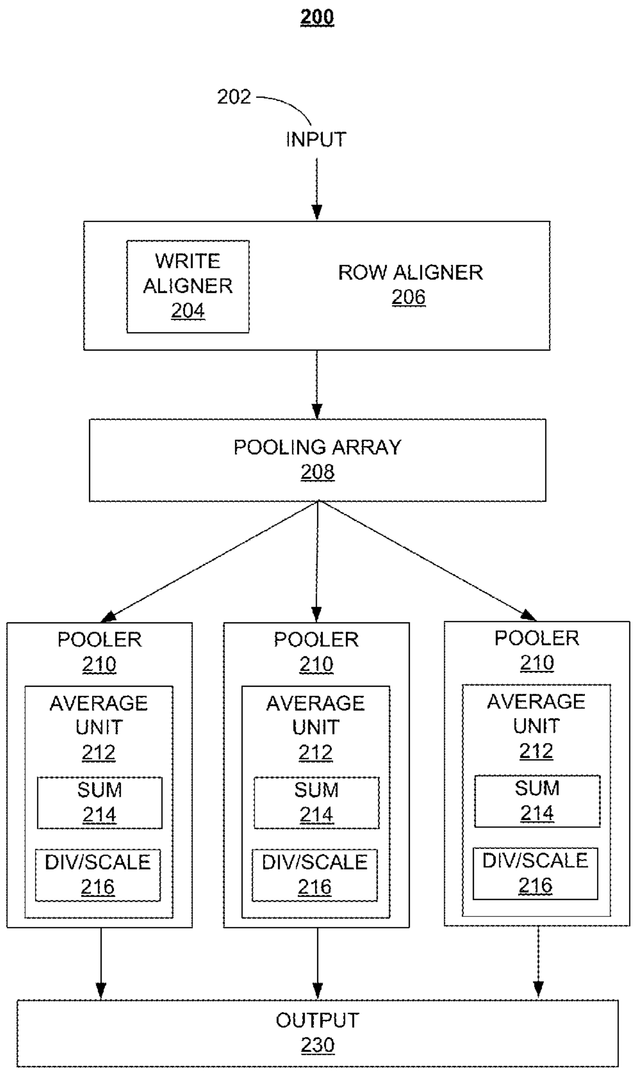
(51) **Int. Cl.**

G06N 3/063	(2006.01)
G06K 9/66	(2006.01)
G06F 15/18	(2006.01)
G06K 9/34	(2006.01)
G06K 9/46	(2006.01)

(52) **U.S. Cl.**
CPC ***G06N 3/063*** (2013.01); ***G06K 9/66*** (2013.01); ***G06K 9/4628*** (2013.01); ***G06K 9/342*** (2013.01); ***G06F 15/18*** (2013.01)

(57) **ABSTRACT**

Described herein are systems and methods that utilize a novel hardware-based pooling architecture to process the output of a convolution engine representing an output channel of a convolution layer in a convolutional neural network (CNN). The pooling system converts the output into a set of arrays and aligns them according to a pooling operation to generate a pooling result. In certain embodiments, this is accomplished by using an aligner that aligns, e.g., over a number of arithmetic cycles, an array of data in the output into rows and shifts the rows relative to each other. A pooler applies a pooling operation to a combination of a subset of data from each row to generate the pooling result.



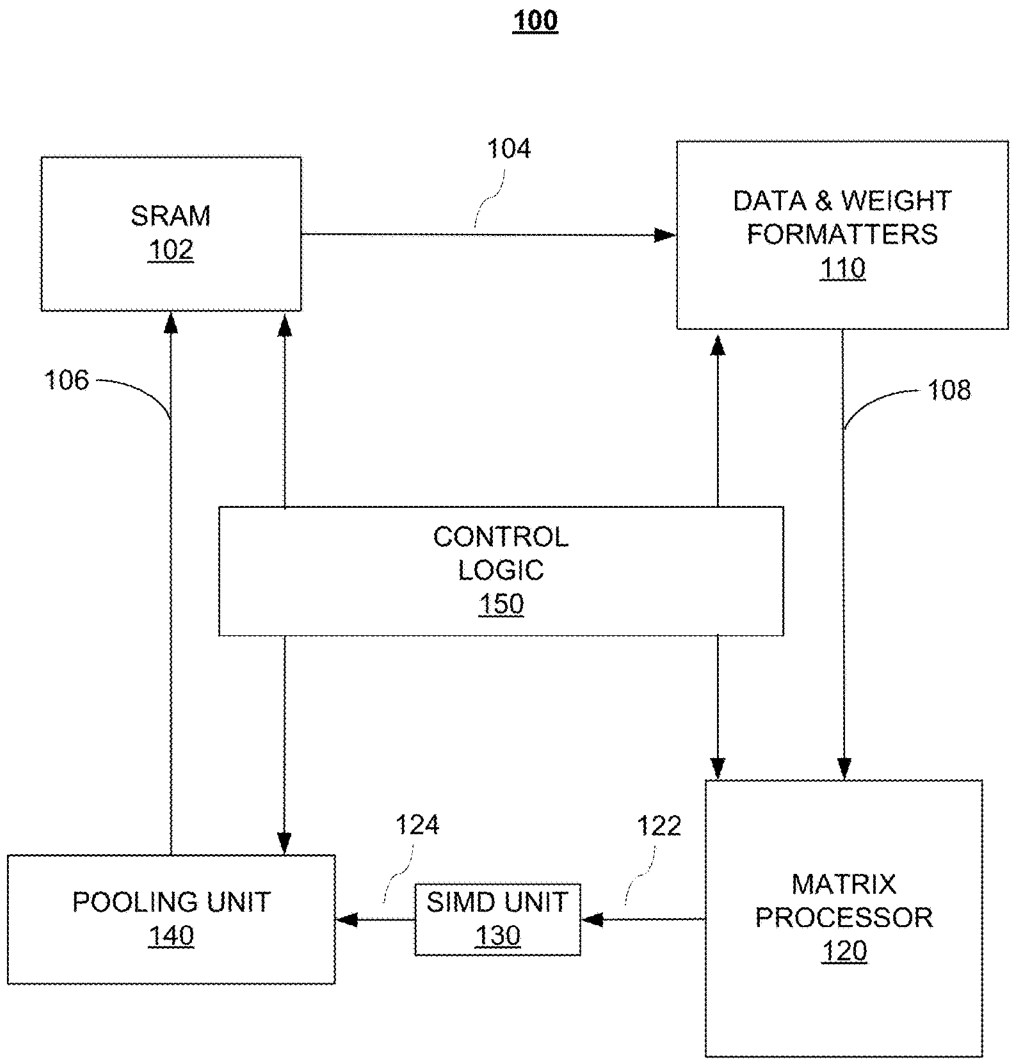

100
SRAM
102
104
DATA & WEIGHT
FORMATTERS
110
106
108
CONTROL
LOGIC
150
124
122
POOLING UNIT
140
SIMD UNIT
130
MATRIX
PROCESSOR
120

FIGURE 1

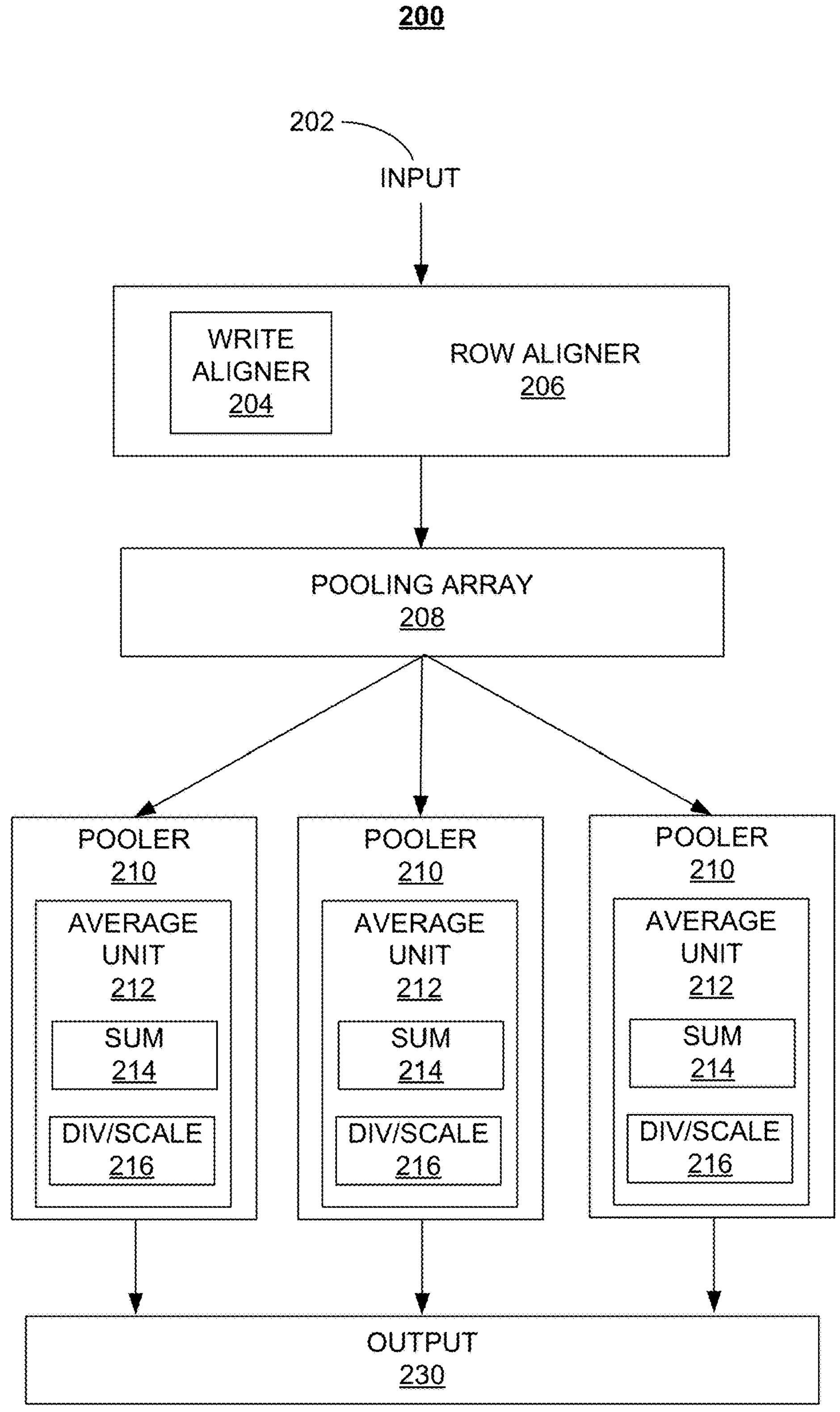

200
202
INPUT
WRITE ALIGNER 204
ROW ALIGNER 206
POOLING ARRAY 208
POOLER 210
AVERAGE UNIT 212
SUM 214
DIV/SCALE 216
POOLER 210
AVERAGE UNIT 212
SUM 214
DIV/SCALE 216
POOLER 210
AVERAGE UNIT 212
SUM 214
DIV/SCALE 216
OUTPUT 230

FIGURE 2

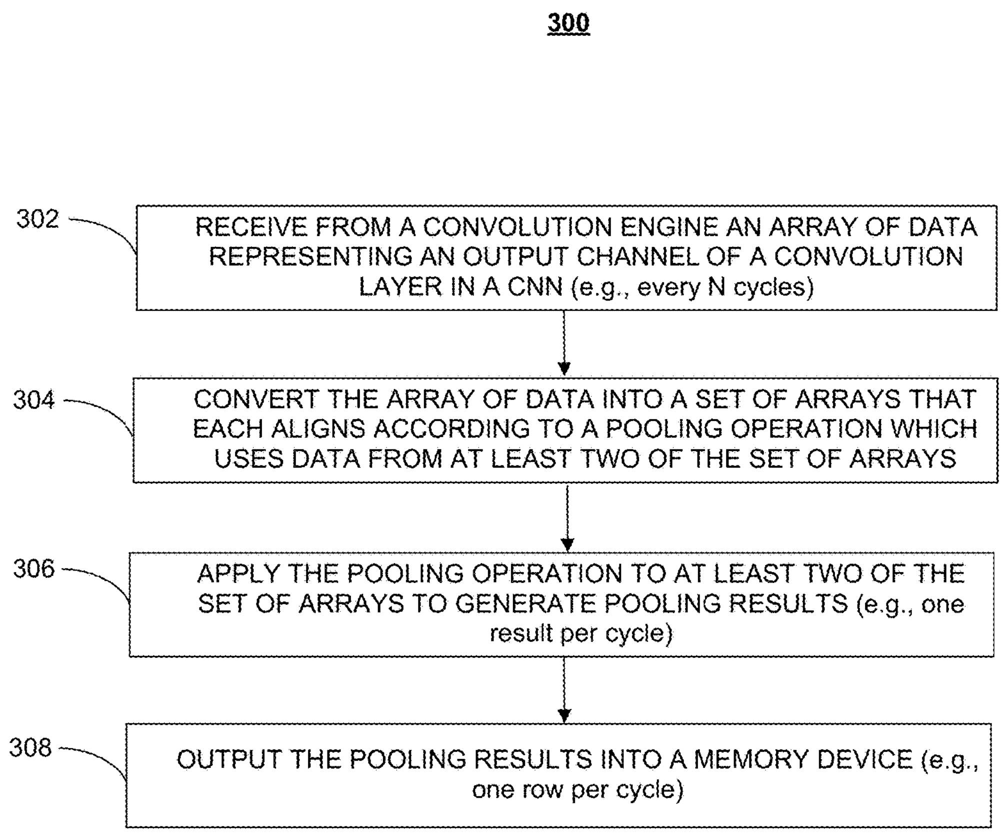

300
302
RECEIVE FROM A CONVOLUTION ENGINE AN ARRAY OF DATA REPRESENTING AN OUTPUT CHANNEL OF A CONVOLUTION LAYER IN A CNN (e.g., every N cycles)
304
CONVERT THE ARRAY OF DATA INTO A SET OF ARRAYS THAT EACH ALIGNS ACCORDING TO A POOLING OPERATION WHICH USES DATA FROM AT LEAST TWO OF THE SET OF ARRAYS
306
APPLY THE POOLING OPERATION TO AT LEAST TWO OF THE SET OF ARRAYS TO GENERATE POOLING RESULTS (e.g., one result per cycle)
308
OUTPUT THE POOLING RESULTS INTO A MEMORY DEVICE (e.g., one row per cycle)

FIGURE 3

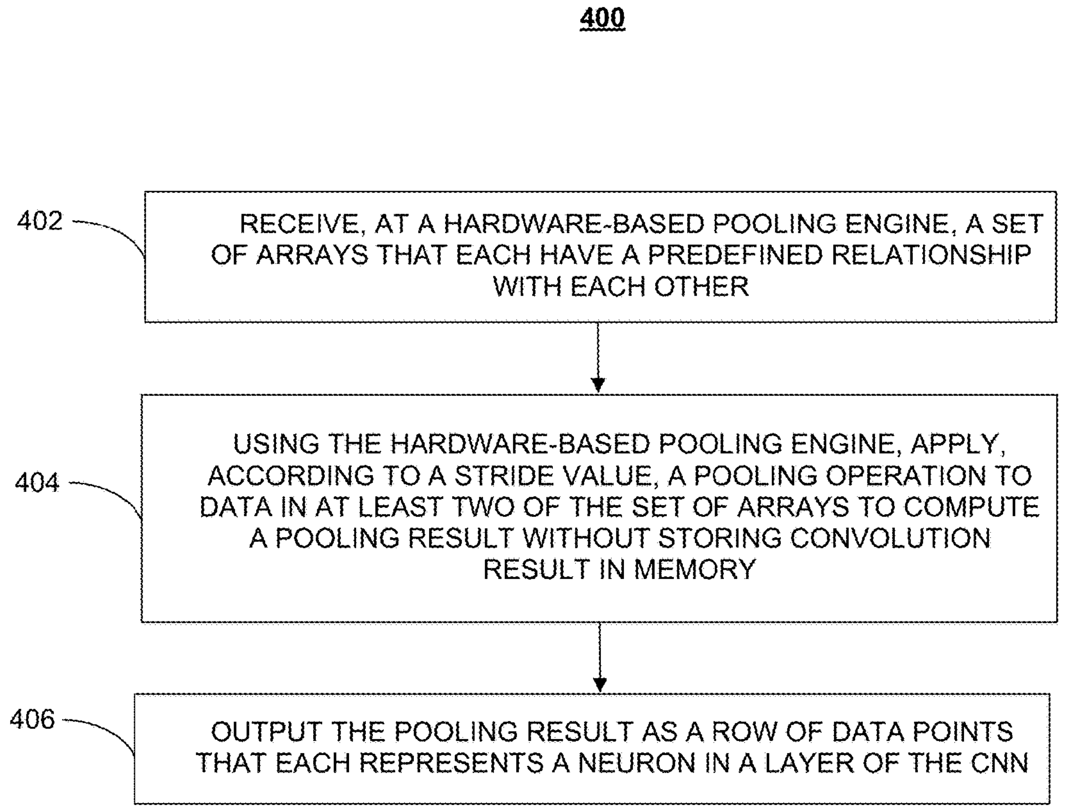

400
402
RECEIVE, AT A HARDWARE-BASED POOLING ENGINE, A SET OF ARRAYS THAT EACH HAVE A PREDEFINED RELATIONSHIP WITH EACH OTHER
404
USING THE HARDWARE-BASED POOLING ENGINE, APPLY, ACCORDING TO A STRIDE VALUE, A POOLING OPERATION TO DATA IN AT LEAST TWO OF THE SET OF ARRAYS TO COMPUTE A POOLING RESULT WITHOUT STORING CONVOLUTION RESULT IN MEMORY
406
OUTPUT THE POOLING RESULT AS A ROW OF DATA POINTS THAT EACH REPRESENTS A NEURON IN A LAYER OF THE CNN

FIGURE 4

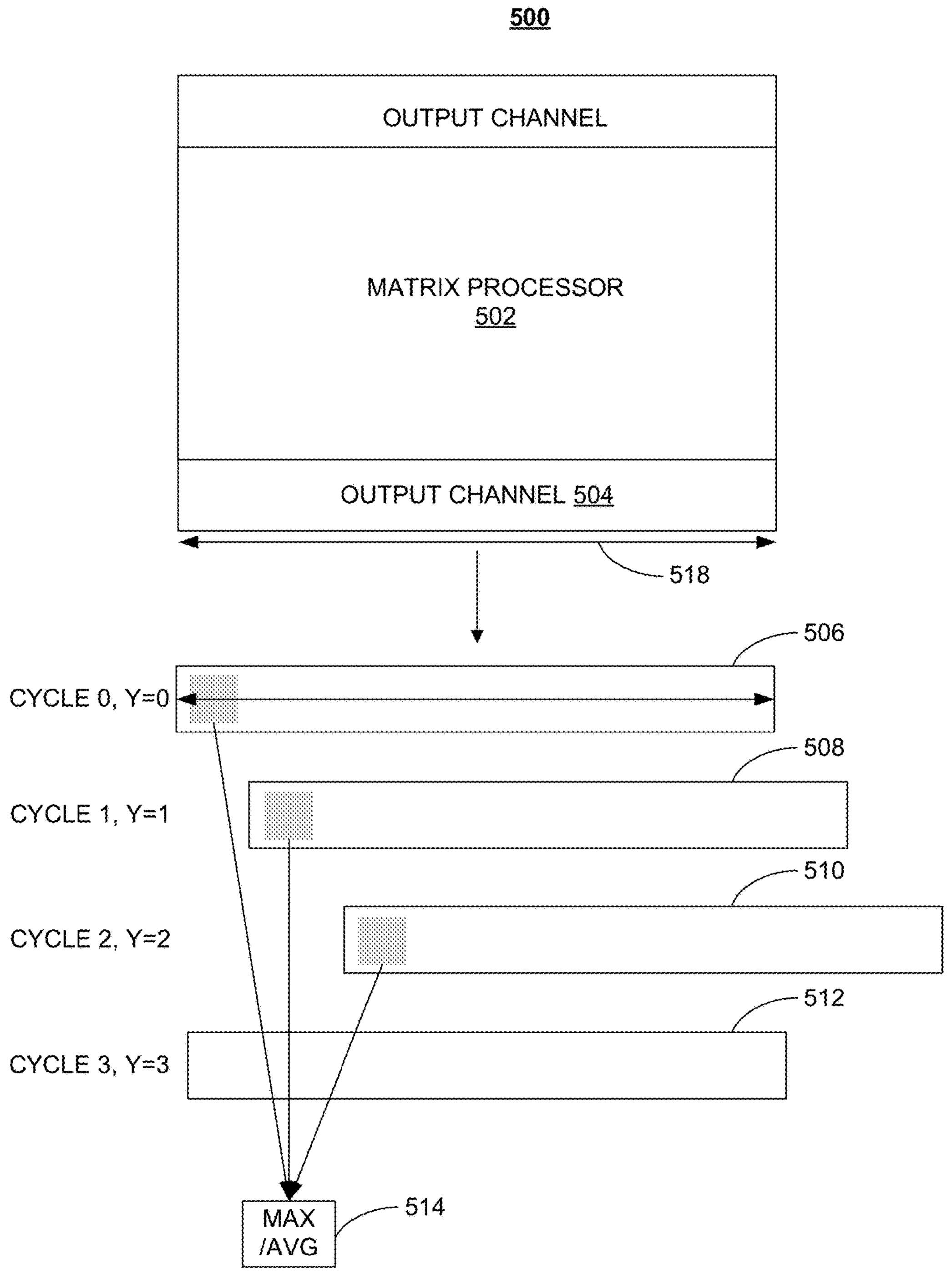

500
OUTPUT CHANNEL
MATRIX PROCESSOR
502
OUTPUT CHANNEL 504
518
506
CYCLE 0, Y=0
508
CYCLE 1, Y=1
510
CYCLE 2, Y=2
512
CYCLE 3, Y=3
MAX
/AVG
514

FIGURE 5

SYSTEMS AND METHODS FOR HARDWARE-BASED POOLING

A. TECHNICAL FIELD

[0001] The present disclosure relates generally to systems and methods for improving utilization of computing resources, such as computational power and storage requirements. More particularly, the present disclosure is related to systems and methods for improving efficiency of arithmetic processes in computer vision applications that use convolutional neural network (CNN) architectures to generate convolutional and pooling data.

B. BACKGROUND

[0002] Neural network-based image classifiers are achieving significant improvements in automatically learning complex features for classification and object recognition. For example, a Convolutional Neural Network (CNN) model may be used to automatically determine whether an image can be categorized as comprising a person or animal. The CNN applies a number of hierarchical network layers and sub-layers to an input image when making a determination or prediction. One characteristic of CNNs is that each network layer serves as an output of a previous layer, typically starting at a first convolutional layer and ending with one or more final layers, e.g., a fully connected layer that includes nodes whose activation values deliver scores that indicate a likelihood that the input image can indeed be classified as comprising a certain object.

[0003] A convolution layer may use several filters known as kernels or activation functions that apply to the pixels of a convolution window of an image a set of weights. The weights have been learned by the CNN during a training phase to generate an activation value associated with that window. For each filter, the convolution layer may have, for each pixel, one node, i.e., neuron, that outputs an activation value that is calculated based on the set of weights. The activation value for the convolution window identifies a feature or characteristic, such as an edge that can be used to identify the feature at other locations within the image. Since all nodes for a filter can share the same set of weights, reusing weights is a common technique to increase utilization of both storage space and computation time.

[0004] Among the most important types of layers of a CNN is the pooling layer—a basic, independent building block that is typically placed after a convolutional layer. As applied to images, a pooling layer allows the network to determine a feature map and learn a set of features for the image. Pooling is viewed as a form of nonlinear sub-sampling or down-sampling that uses a nonlinear function, such as max-pooling or average-pooling, to reduce the number of neurons when progressing from layer to layer through the network; thereby, reducing the amount of computation and further increasing computational performance.

[0005] Pooling generally involves sliding a pooling window, e.g., a two-dimensional square of multiple pixels in width and multiple pixels in height, stepwise across small, non-overlapping areas (i.e., receptive field) of the output of a preceding convolution layer. Aggregating the values of the group of neurons in that area provides single output values (e.g., integers) for each group in a local neighborhood. These output values assigned to each group are passed to a subsequent layer without performing a convolution and depend on the type of pooling function (e.g., average or max) that is used in the pooled area. The size and location of the pooling window depends on the pooling stride (i.e., interval or step size) and the location of the output pixel. Oftentimes, the last pooling layer is followed by the final output layer (e.g., a fully connected layer with a soft-max nonlinearity) of the CNN architecture that outputs the final prediction, e.g., as an estimate of a conditional probability, for each particular class.

[0006] While great progress has been achieved in improving the performance of convolutional layers by sharing of weights and improving arithmetic logic unit utilization, pooling layers, which are similarly computationally intensive, have been neglected mainly due to constraints inherent to existing neural network architectures.

[0007] Accordingly, it would be desirable to have systems and methods that improve the performance of pooling layers in neural networks to further increase the utilization end performance of available computational resources to reduce overall computational cost.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] References will be made to embodiments of the invention, examples of which may be illustrated in the accompanying figures. These figures are intended to be illustrative, not limiting. Although the invention is generally described in the context of these embodiments, it should be understood that it is not intended to limit the scope of the invention to these particular embodiments.

[0009] FIG. **1** is an exemplary block diagram of a system that uses a pooling unit for performing pooling operations according to various embodiments of the present disclosure.

[0010] FIG. **2** is an exemplary block diagram of a pooling unit architecture according to various embodiments of the present disclosure.

[0011] FIG. **3** is a flowchart of an illustrative process for using a pooling system shown in FIG. **1**.

[0012] FIG. **4** is a flowchart of an illustrative process for using the pooling unit architecture shown in FIG. **2**.

[0013] FIG. **5** is a flowchart of an illustrative process for performing pooling operations according to various embodiments of the present disclosure.

DETAILED DESCRIPTION OF EMBODIMENTS

[0014] In the following description, for purposes of explanation, specific details are set forth in order to provide an understanding of the invention. It will be apparent, however, to one skilled in the art that the invention can be practiced without these details. Furthermore, one skilled in the art will recognize that embodiments of the present invention, described below, may be implemented in a variety of ways, such as a process, an apparatus, a system, a device, or a method on a tangible computer-readable medium.

[0015] Components, or modules, shown in diagrams are illustrative of exemplary embodiments of the invention and are meant to avoid obscuring the invention. It shall also be understood that throughout this discussion that components may be described as separate functional units, which may comprise sub-units, but those skilled in the art will recognize that various components, or portions thereof, may be divided into separate components or may be integrated together, including integrated within a single system or component. It should be noted that functions or operations discussed herein may be implemented as components. Components may be implemented in software, hardware, or a combination thereof.

[0016] Furthermore, connections between components or systems within the figures are not intended to be limited to direct connections. Rather, data between these components may be modified, re-formatted, or otherwise changed by intermediary components. Also, additional or fewer connections may be used. It shall also be noted that the terms "coupled," "connected," or "communicatively coupled" shall be understood to include direct connections, indirect connections through one or more intermediary devices, and wireless connections.

[0017] Reference in the specification to "one embodiment," "preferred embodiment," "an embodiment," or "embodiments" means that a particular feature, structure, characteristic, or function described in connection with the embodiment is included in at least one embodiment of the invention and may be in more than one embodiment. Also, the appearances of the above-noted phrases in various places in the specification are not necessarily all referring to the same embodiment or embodiments.

[0018] The use of certain terms in various places in the specification is for illustration and should not be construed as limiting. A service, function, or resource is not limited to a single service, function, or resource; usage of these terms may refer to a grouping of related services, functions, or resources, which may be distributed or aggregated. Furthermore, the use of memory, database, information base, data store, tables, hardware, and the like may be used herein to refer to system component or components into which information may be entered or otherwise recorded.

[0019] Furthermore, it shall be noted that: (1) certain steps may optionally be performed; (2) steps may not be limited to the specific order set forth herein; (3) certain steps may be performed in different orders; and (4) certain steps may be done concurrently.

[0020] FIG. **1** is an exemplary block diagram of a system that uses a pooling unit for performing pooling operations according to various embodiments of the present disclosure. System **100** comprises SRAM **102**, data/weight formatter **110**, matrix processor **120**, post-processing unit **130**, pooling unit **140**, control logic **150**. It is understood that system **100** may comprise additional circuits and sub-circuits, such as logic circuitry and/or control circuitry, caches, local buffers, comparators, state machines, additional post processing units, and auxiliary devices that perform management functions.

[0021] In embodiments, any component in system **100** may be partially or entirely controlled by control logic **150** that may monitor the status and operations of system **100**, e.g., when performing an operation such as a convolution or other mathematical calculation, and calculate locations from which to retrieve data that will be used in a subsequent step of the operation. Similarly, control logic **150** may manage other components, e.g., components that are not shown in FIG. **1** and/or outside of system **100**.

[0022] In embodiments, SRAM **102** stores and makes accessible input image data, e.g., in a data input matrix and a weight input matrix **104**. One skilled in the art will recognize that other types of storage devices may be used.

[0023] In embodiments, based on the weight input matrix and data input matrix **104**, data/weight formatter **110** produces two outputs **108**, e.g., each 96-columns wide, for matrix processor **120**, which may process a very large number of elements of a matrix in parallel to efficiently map data into a matrix operation. Data/weight formatter **110** may be implemented as any number of in-line formatters that convert, e.g., data input matrices and weight input matrices **104** into a suitable format for further processing by matrix processor **120**, e.g., according to specific hardware requirements of matrix processor **120**. In embodiments, formatter **110** converts two-dimensional or three-dimensional matrices into a single vector or string that may be represented by a row or column before making the so linearized or vectorized data available as input **108** to matrix processor **120**. As a result, matrix processor **120** can be efficiently utilized to execute a matrix multiply operation as part of a convolution computation in system **100** to generate output array **122** that then may be reassembled, e.g., into an image.

[0024] A neural network model using the embodiments of the present disclosure may comprise a pooling network that uses max-pooling layers, averaging pooling layers, and other neural network layers. The pooling network may be followed or preceded by, e.g., (by a processing module that uses a fully-connected layer and), in embodiments, an activation layer that uses a known function, such as a non-linear function, e.g., a Rectified Linear Unit (ReLU), logistic sigmoid function, and the like.

[0025] In embodiments, matrix processor **120** performs a convolution operation by applying individual filters (e.g., weights) to input image data to detect small features within an input image. By analyzing a sequence of different features in a different order, macro features may so be identified in the input image. Matrix processor **120** may use a different set of weights for each input channel, as each input channel may contain a different set of information, and each weight matrix may be used detect a different feature. In embodiments, matrix processor **120** multiplies a rectangular input matrix with a rectangular weight matrix to obtain partial dot products that may then be summed to generate an accumulated dot product, i.e., an integer, which represents an output pixel in an output image. In embodiments, output array **122** may correspond to the dot product of two matrices **108** that have been processed by formatter **110**.

[0026] In embodiments, matrix processor **120** may perform convolution operations that convolve an input with a filter to generate output **122** by converting a convolution operation into a matrix multiplication (e.g., a 96×96 matrix multiplication). Matrix processor **120** may comprise circuitry, such as arithmetic logic units, registers, encoders and may be implemented as having an arbitrary number of columns and rows to perform mathematical accelerated operations across a large set of data and weights. These large-scale operations may be timed according to the specific hardware requirements of matrix processor **120** to accelerate convolution operations, e.g., by reducing redundant operations within system **100** and by implementing hardware specific logic.

[0027] In embodiments, matrix processor **120** outputs **122** a linearized vector or array representing an output channel that may be stored in storage within post-processing unit **130**. In embodiments, pooling unit **140** operates on a single output channel of matrix processor **120**, such that output **122** or post-processed output **124** is an array that may otherwise not conveniently map into a matrix operation. Therefore, in embodiments, output array **122** may be reformatted into a suitable format for pooling unit **140** to increase the efficiency of system **100**.

[0028] In contrast, conventional implementations that employ a vector engine that performs vector operations on a stored convolution would lead to rather complex and inefficient pooling operations the output of a highly efficiency matrix processor, such as matrix processor **120**, in part, because some values in output array **122** may be adjacent while others may not. In short, a pooling algorithm following a convolution operation by matrix processor **120** would have to process a combination of values in output array **122** that are not presented in a convenient shape or format for common pooling methods. Therefore, in embodiments, output array **122** is reformatted in order to allow for the application of improved pooling methods to a high-efficiency matrix processor **120**.

[0029] To achieve this, in embodiments, hardware pooling unit **140**, in response to receiving output array **122**, e.g., as processed by post-processing unit **130**, reformats the received data into a grid format, such that some elements of output array **122** may be aligned in a vertical direction and others may be aligned in a horizontal direction, such that pooling can be directly applied without the need to perform cumbersome, computational-intensive intermediate steps and data storage operations. In embodiments, formatter **110** may reformat different shapes of input matrix data into columns and rows suitable for matrix processor **120**. In embodiments, formatting may be performed dynamically to accommodate processing of matrices that have differing input sizes.

[0030] In embodiments, pooling unit **140** applies a pooling function, e.g., average pooling and max pooling, to the reformatted data in order to generate and output pooled data **106** that may then be written and stored in SRAM **102**, e.g., as a feature map. The internal operation of pooling unit **140** will be described in more detail with respect to FIG. **2**.

[0031] In embodiments, matrix processor **120** outputs a set of convolution data, e.g., output array **122**, while accumulating and computing the next set of convolution data. Similarly, pooling unit **140** generates output **106** on-the-fly from data shifted out of matrix processor **120**, thus, covering the cost of pooling and reducing computation time when compared to software-based pooling methods, which require that a convolution be stored in intermediate storage prior to being passed through a pooling layer.

[0032] In embodiments, post-processing unit **130** receives data, e.g., a dot product result that corresponds to an output channel, from the bottom row of matrix processor **120**, e.g., via output flip-flops (not shown) that form a shift register. Post-processing unit **130** may apply, e.g., a non-linear ReLU function to output array **122**.

[0033] It is noted that padding, e.g., zero-padding, may be performed at the edges of a matrix prior to a convolution layer operation in order to obtain a predetermined output feature map size. In embodiments, padding may be enabled if the stride is set to a value greater than 1. If padding is enabled, control logic **150** may treat certain columns as zeros, such that the divisor in an average pooling operation is adjusted to equal the sum of the non-zero pooling values involved in the average calculation.

[0034] FIG. **2** is an exemplary block diagram of a pooling unit architecture according to various embodiments of the present disclosure. Pooling unit **200** may comprise row aligner **206**, write aligner **204**, pooling array **208**, pooler **210**. In embodiments, pooler **210** may comprise a max unit (not shown), averaging unit **212**, or any other unit that may perform pooling operations to generate output **230**. In embodiments, averaging unit **212** performs and averaging function by using summing element **214** followed by divide and or scale unit **216**.

[0035] Input **202** may correspond to a set of feature maps. In embodiments, input **202** constitutes an output channel that has been produced according to the requirements of a high-efficiency matrix processor, for example, a matrix processor disclosed U.S. patent application Ser. No. 15/710,433 entitled "Accelerated Mathematical Engine," which reference is incorporated herein in its entirety.

[0036] In embodiments, pooling unit **200**, in response to receiving input **202**, reformats the data therein into the equivalent of a grid pattern to which conventional pooling methods may be applied, for example, to reduce the height and width of the feature maps by a factor of two. In embodiments, pooling unit **200** accomplishes reformatting by arranging and storing input **202** (e.g., in row aligner **206**) in a number of rows that have the same width as input **202**, such that each row comprises sections of data that correspond to a group of neighborhood values in a matrix to which a pooling operation may be applied to obtain a pooling result. In embodiments, once the rows are aligned such that those sections that belong to the same neighborhood can be extracted, pooling may be easily performed, e.g., by pooler **210**. In embodiments, the combination of sections pooled in this manner represents a pooling result of an entire pooled output channel of a convolution.

[0037] In embodiments, row aligner **206** stores input **202** in such a way that it can be accessed and read by pooler **210** as to-be-pooled data. In other words, the output channel of the matrix processor may be reformatted to a format that can be read easily pooled by pooler **210** while maintaining a stream of input data **102**. In embodiments, row aligner **206** is controlled by a controller (not shown) to shift incoming input **202** prior to writing the result into a number of pooling arrays **208**, e.g., **3** arrays that comprise the to-be-pooled data.

[0038] In embodiments, pooler **210** identifies suitable values in row aligner **206** for use in a particular pooling calculation and extracts from pooling arrays **208** a number of values to compute a pooling result. The pooling result depends on the type of pooling function used and may be an average value, a maximum value, or an intermediate value (e.g., a sum) that may be converted into a suitable pooling result. In embodiments, divide and or scale unit **216** may follow averaging unit **212** and may be implemented as a multiply-and-shift circuit that generates output **230**. In embodiments, pooler **210** may access pooling array **208** to process any subsection of pooling array **208** that comprises a number of to-be-pooled values. For example, e.g., pooler **210** may pool 9 values corresponding to a 3×3 pooling window to generate an average pooling value. It is understood that the pooling window may assume any arbitrary size and shape depending on parameters settings.

[0039] In embodiments, input **202** is read, and reformatting is applied over a period of n arithmetic cycles, e.g., using a method for aligning rows of data (further discussed with respect to FIG. **4**) to generate pooling results **230** in every cycle, e.g., one row at a time. In embodiments, once an output channel is read, e.g., as input **202**, the next output channel may be read and reformatting may be applied, for example, by using a different set of memory that stores the rows of data in a different pooler **212**, until all output channels provided by the matrix processor are processed and the results **230** can be output. It is understood that portions of an output channel and, in general, different output channels may be processed at different times using other methods and other circuit configurations than those depicted in FIG. **2** and accompanying text. As those skilled in the art will appreciate, additional pooling layers may be used to output higher level or refined feature maps.

[0040] In embodiments, pooling unit **200** computes pooling results as fast as matrix processor **120** to generate output **122**. Pooling unit **140** may apply a stride of, e.g., n=2 or n=3, to control the amount of elements the sliding window crosses between calculations. A person of skill in the art will appreciate that the sliding mechanism for pooling layers operates in a similar manner as that in a convolution layer that, for example, uses a common kernel size of 2 or 3, with the difference that the average or the largest value is selected in the pooling window.

[0041] In embodiments, pooling unit **200** receives the processed data and performs a computation on a set of arrays that may be spatially shifted relative to each other. In embodiments, pooling result **124** is pulled or shifted by a state machine (not shown) into an output array, e.g., one per clock cycle. The state machine may perform additional operations on pooling result **124** prior to sending data to SRAM **102** or some other post-processing unit (not shown).

[0042] It is understood that pooling unit **200** may further comprise components and sub-circuit circuits not shown in FIG. **2**, such as a control unit that coordinates the sequence of operations of any number of components coupled with pooling unit **200**. For example, the control unit may determine the number and location of data points that are involved in a given operation without modifying the sequence of the operation itself.

[0043] FIG. **3** is a flowchart of an illustrative process for using a pooling system shown in FIG. **1**. Process **300** begins step **302** when data from a convolution engine is received, e.g., at a pooling unit and at every n cycles. In embodiments, the data is received in the form of a data array and represents an output channel of a convolution layer in a CNN.

[0044] At step **304**, the array is converted into a set of arrays that are aligned according to a pooling operation. In embodiments, the pooling operation uses at least two arrays from the set of arrays to apply a pooling operation, at step **306**, to generate pooling results, e.g., one result per cycle.

[0045] Finally, at step **308**, the pooling result is output, e.g., as one row per arithmetic cycle, into a memory device.

[0046] FIG. **4** is a flowchart of an illustrative process for using the pooling unit architecture shown in FIG. **2**. Process **400** begins step **402** when a hardware-based pooling unit receives from a convolution engine a set of data arrays that each have a predefined relationship with each other.

[0047] At step **404**, using the hardware-based pooling unit, a pooling operation is applied to data in at least two arrays from the set of data arrays to obtain a pooling result, e.g., an average or max pooling result. The pooling operation may be applied according to a stride value. In addition, this hardware-based pooling method takes advantage of a 1:1 output channel to input channel relationship that, advantageously eliminates the need to write a convolution result into intermediate memory.

[0048] At step **406**, the pooling result is output, e.g., as one row of data points per cycle that each represent a neuron in a layer of the CNN.

[0049] FIG. **5** is an exemplary block diagram illustrating a process for performing pooling using the pooling unit architecture shown in FIG. **2**. In embodiments, the matrix processor **502** of the pooling unit architecture outputs output channel **504**. Since a pooling operation may be treated as a convolution with fixed weights a matrix processor could be used to perform the pooling operation. However, since there is typically only a single output channel in pooling, operating only one output channel of multi-output channel matrix processor at a time is a rather inefficient undertaking that unnecessarily ties up computing resources. Therefore, to increase computing efficiency, in embodiments, output channel **504** may be written into a number of rows **506-510** that are aligned, e.g., by a row aligner as shown in FIG. **2**, such that each row **506-510** is shifted against another in subsequent cycles. In embodiments, rows Y=0, Y=1, and Y=2 in FIG. **5** may hold output channel **504** and may have been written and stored in respective cycles 0 through 2.

[0050] For example, in a cycle 0, at least a first section of input **202** is stored, e.g., left aligned, into row Y=0. In the following cycle, cycle 1, the same section is stored into row Y=1, and so on, such that it takes three reading cycles to fill rows **506-510**. Once rows **506-510** are populated, data from rows **506-510** can be combined to perform pooling calculations. For example, 3 values from each of row **506-510** may be combined to 9 values that generate pooling value **514** as a result.

[0051] It is noted that of pooling calculations may be performed in parallel. For example, to maintain a stream of incoming output channels **504**, the number of pooling calculations may be equal to the total number of output channels in matrix processor **502**, such that regardless of kernel size, pooling data corresponding to the entire width **518** of matrix processor **502** may be output.

[0052] In embodiments, the shift from one row to another corresponds to a shift of a pooling window when convolving across a matrix to generate pooling results. In embodiments, the shift that is attributable to the pooling window is defined by the number of cycles and may correspond to a stride having a value that is defined by the same number of cycles. In short, the stride dictates how often pooling data is output. For example, for a stride of 2, pooling values may be output every other cycle, thereby, skipping a row (or column) between outputs.

[0053] In embodiments, to create a sliding window of three rows of storage that slide one at a time, in a third cycle **512**, the values of the first row **506** may be overwritten, such that the cycles use the set of three rows **506-510** and, based on pooling parameters, output a pooling calculation result.

[0054] It is understood that the number of rows of storage corresponds to the size of the kernel that is supported and that parameters such as window size, stride size, type of pooling used, etc., may be determined and controlled independent from the pooling process itself.

[0055] One skilled in the art will recognize no computing system or programming language is critical to the practice of the present invention. One skilled in the art will also recognize that a number of the elements described above may be physically and/or functionally separated into sub-modules or combined together.

[0056] It will be appreciated to those skilled in the art that the preceding examples and embodiments are exemplary and not limiting to the scope of the present disclosure. It is intended that all permutations, enhancements, equivalents, combinations, and improvements thereto that are apparent to those skilled in the art upon a reading of the specification and a study of the drawings are included within the true spirit and scope of the present disclosure. It shall also be noted that elements of any claims may be arranged differently including having multiple dependencies, configurations, and combinations.

What is claimed is:

1. A pooling unit architecture comprising:

a controller;

an aligner coupled to the controller, the aligner, in response to receiving input data, aligns the input data into rows to generate a pooling array and, over a number of arithmetic cycles, shift the rows relative to each other to reformat the input data into reformatted data; and

a pooler coupled to the aligner, the pooler applies, in subsequent arithmetic cycles, a pooling operation to at least some of the reformatted data to obtain a pooling output that comprises a pooling value, wherein a subset of data from each row is combined to a set of data from which the pooling value is generated.

2. The pooling unit according to claim **1**, wherein the input data has been generated by a matrix processor.

3. The pooling unit according to claim **2**, wherein, to maintain a stream of the input data, the pooling output is generated at a same rate as a rate at which the matrix processor generates the input data.

4. The pooling unit according to claim **2**, wherein the pooler performs one or more pooling calculations in parallel, and wherein the number of pooling calculations equals a number of output channels in the matrix processor, such that, independent of a kernel size, the pooling output corresponds to a width of the matrix processor.

5. The pooling unit according to claim **1**, further comprising a multiply-and-shift circuit coupled to the pooler, the multiply-and-shift circuit generates the pooling output based on the pooling operation.

6. The pooling unit according to claim **1**, wherein the input data corresponds to a set of feature maps, and wherein the pooler uses the reformatted input data to reduce, by a predetermined factor, at least one of a height and a width of the set of feature maps.

7. The pooling unit according to claim **1**, wherein the rows have the same width as the input data, each row comprising sections of data that correspond to a set of neighborhood values in a matrix.

8. The pooling unit according to claim **1**, further comprising a state machine that shifts the pooling output into an output array.

9. The pooling unit according to claim **1**, wherein the controller determines, without modifying the sequence of the pooling operation itself, a number and a location of data points involved in a pooling operation.

10. The pooling unit according to claim **1**, wherein a shift from one row to another row corresponds to a shift of a pooling window that convolves across a matrix at a stride value, the shift being defined by the number of arithmetic cycles.

11. A method for using a hardware-based pooling system, the method comprising:

receiving from a convolution engine an array of data that represents an output channel of a convolution layer in a convolutional neural network (CNN);

converting the array of data into a set of arrays that are aligned according to a pooling operation that applies data to at least two arrays of the set of arrays to generate a pooling result; and

outputting the pooling result into a memory device.

12. The method according to claim **11**, wherein the array of data is received at a hardware-based pooling unit.

13. The method according to claim **11**, wherein arrays of data are received at intervals of a number of arithmetic cycles.

14. The method according to claim **11**, wherein pooling results are generated at each interval.

15. The method according to claim **14**, wherein pooling results are output at each interval.

16. The method according to claim **11**, wherein the array of data corresponds to a set of feature maps.

17. A method for using a pooling unit architecture, the method comprising:

receiving, at a hardware-based pooling engine, a set of data arrays that each have a predefined relationship with each other;

using the hardware-based pooling unit, applying, according to a stride value, a pooling operation to data in at least two arrays from the set of data arrays to obtain a pooling result without having to satisfy a requirement of writing a convolution result into memory; and

outputting the pooling result as a row of data points that each represent a neuron in a layer of a convolutional neural network (CNN).

18. The method according to claim **17**, wherein the set of data arrays are received from a convolution engine.

19. The method according to claim **17**, wherein obtaining the pooling result utilizes a one-to-one relationship between an output channel and an input channel.

20. The method according to claim **17**, wherein pooling result comprises one of an average pooling result and a max pooling result.

* * * * *