US 20200117986A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2020/0117986 A1**

BURR et al. (43) **Pub. Date: Apr. 16, 2020**

(54) **EFFICIENT PROCESSING OF CONVOLUTIONAL NEURAL NETWORK LAYERS USING ANALOG-MEMORY-BASED HARDWARE**

(71) Applicants: **INTERNATIONAL BUSINESS MACHINES CORPORATION**, Armonk, NY (US); **THE UNIVERSITY OF CHICAGO**, Chicago, IL (US)

(72) Inventors: **GEOFFREY BURR**, CUPERTINO, CA (US); **BENJAMIN KILLEEN**, ST. LOUIS, MO (US)

(21) Appl. No.: **16/363,463**

(22) Filed: **Mar. 25, 2019**

Related U.S. Application Data

(60) Provisional application No. 62/745,132, filed on Oct. 12, 2018.

Publication Classification

(51) **Int. Cl.**

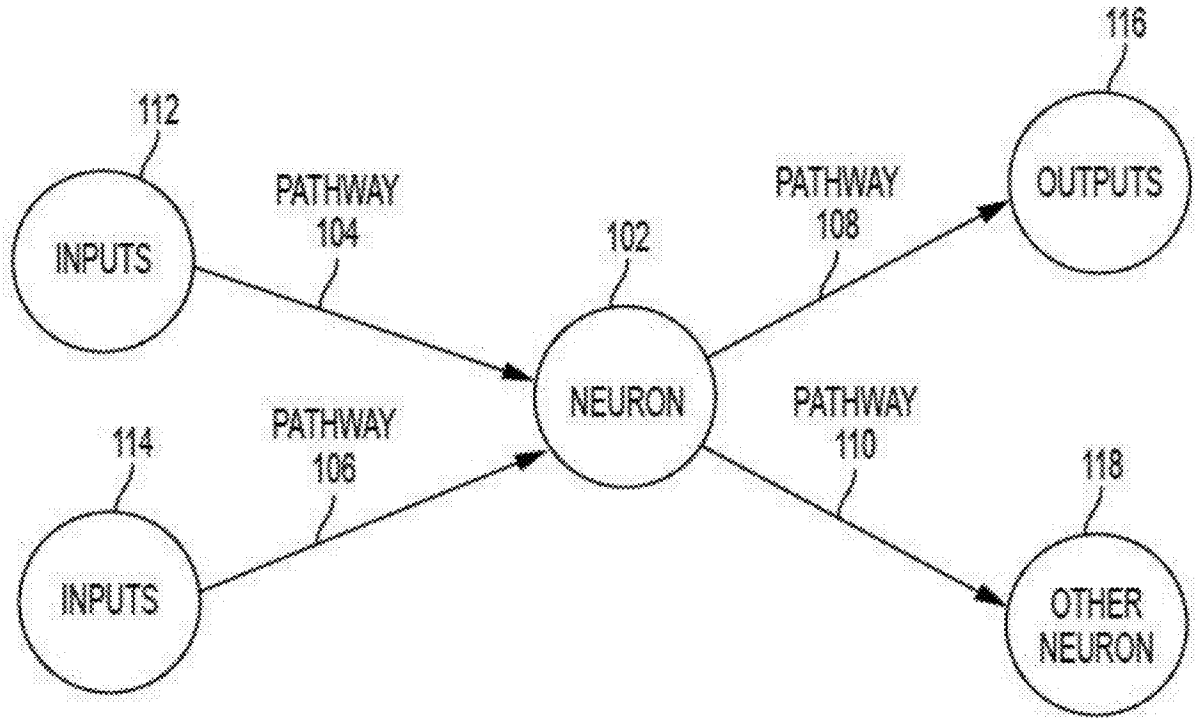
G06N 3/063	(2006.01)
G06F 17/18	(2006.01)
G06F 7/544	(2006.01)
G06F 17/15	(2006.01)

(52) **U.S. Cl.**

CPC ***G06N 3/0635*** (2013.01); ***G06F 17/15*** (2013.01); ***G06F 7/5443*** (2013.01); ***G06F 17/18*** (2013.01)

(57) **ABSTRACT**

According to one or more embodiments, a computer implemented method for implementing a convolutional neural network (CNN) using a crosspoint array includes configuring the crosspoint array corresponding to a convolution layer in the CNN by storing one or more convolution kernels of the convolution layer in one or more crosspoint devices of the crosspoint array. The method further includes performing computations for the CNN via the crosspoint array by transmitting voltage pulses corresponding to a vector of input data of the convolution layer to the crosspoint array. Performing the CNN computations further includes outputting an electric current representative of performing a multiplication operation at a crosspoint device in the crosspoint array based on a weight value stored by the crosspoint device and the voltage pulses from the input data. Performing the CNN computations further includes passing the output electric current from the crosspoint device to a selected integrator.



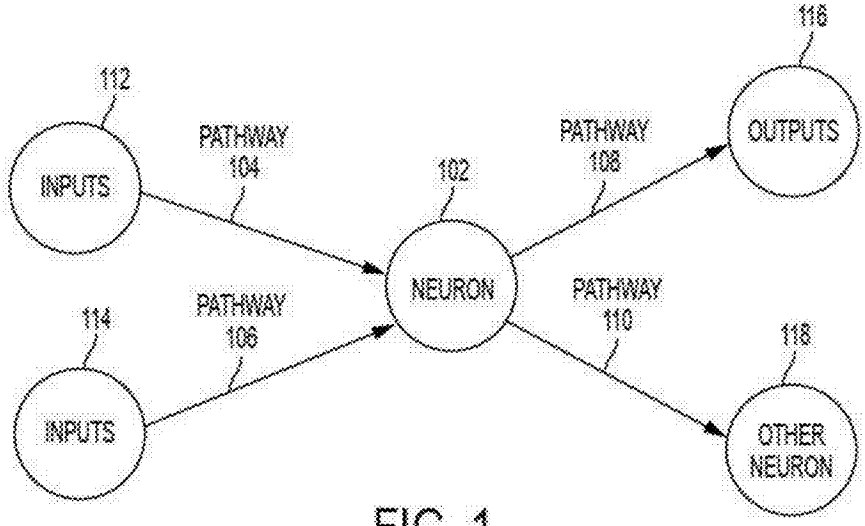
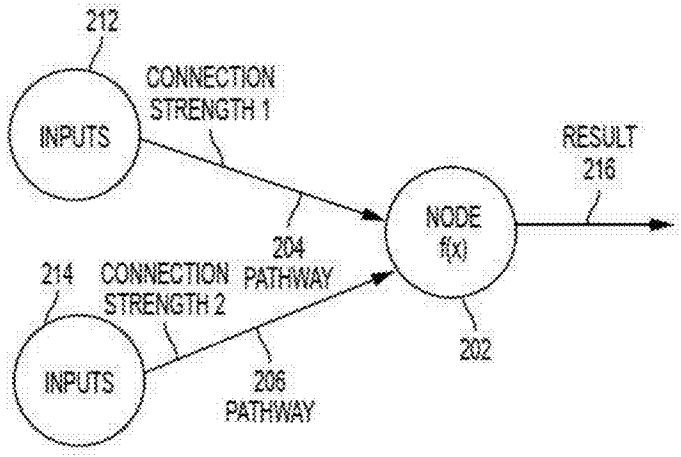
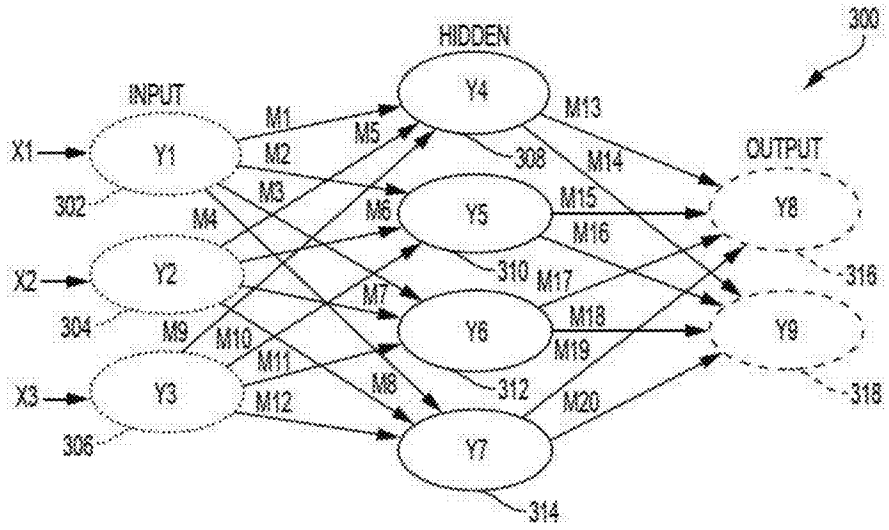


FIG. 1



f(x) = f (INPUT 1 * CONNECTION STRENGTH 1 + INPUT 2 * CONNECTION STRENGTH 2)

FIG. 2



Y1 = F(X1)
Y2 = F(X2)
Y3 = F(X3)

Y4 = F(M1*Y1 + M5*Y2 + M9*Y3)
Y5 = F(M2*Y1 + M6*Y2 + M10*Y3)
Y6 = F(M3*Y1 + M7*Y2 + M11*Y3)
Y7 = F(M4*Y1 + M8*Y2 + M12*Y3)

Y8 = F(M13*Y4 + M15*Y5 + M17*Y6 + M19*Y7)
Y9 = F(M14*Y4 + M16*Y5 + M18*Y6 + M20*Y7)

FIG. 3

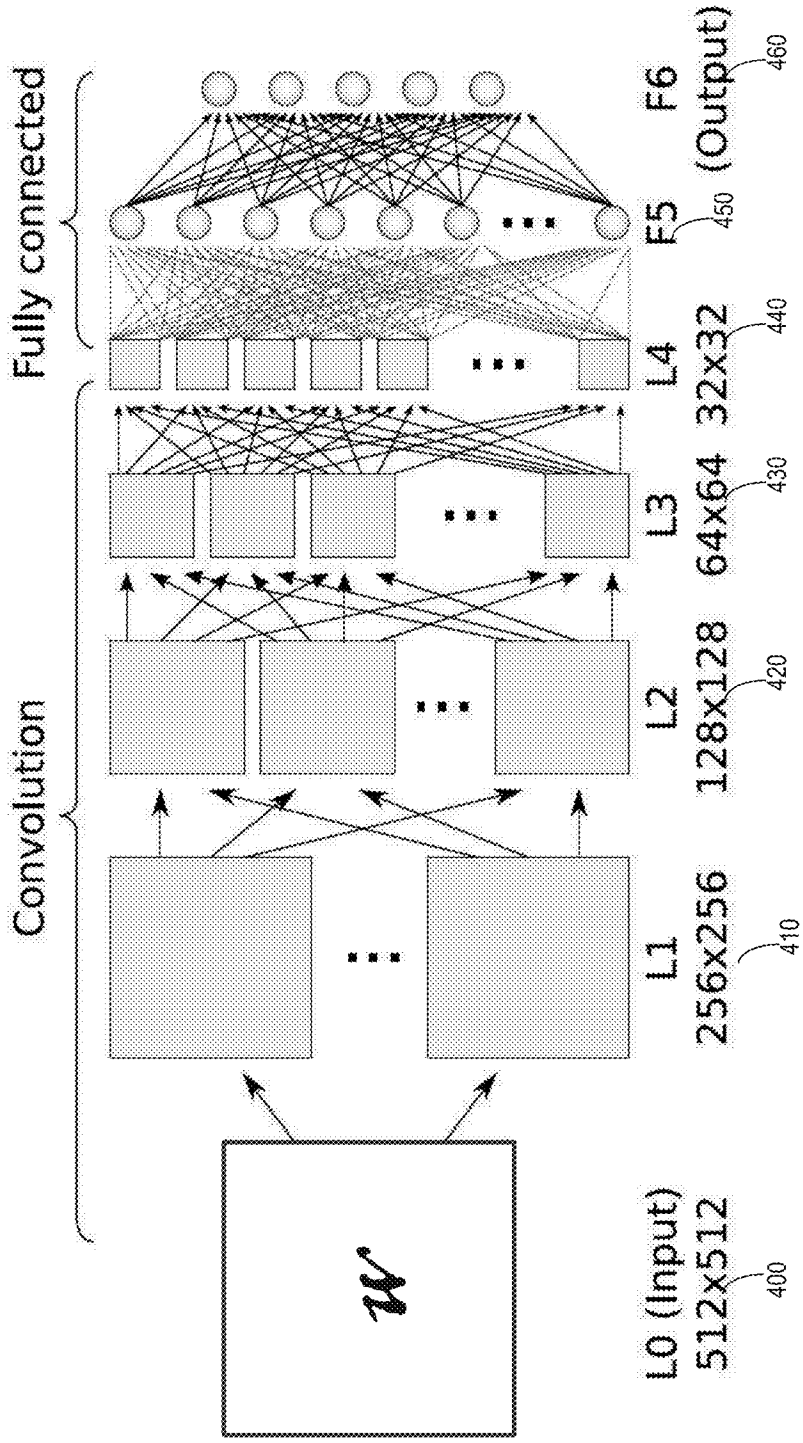

Fully connected
Convolution
F6 (Output)
460
F5
450
L4
32x32
440
L3
64x64
430
L2
128x128
420
L1
256x256
410
L0 (Input)
512x512
400
W

FIG. 4

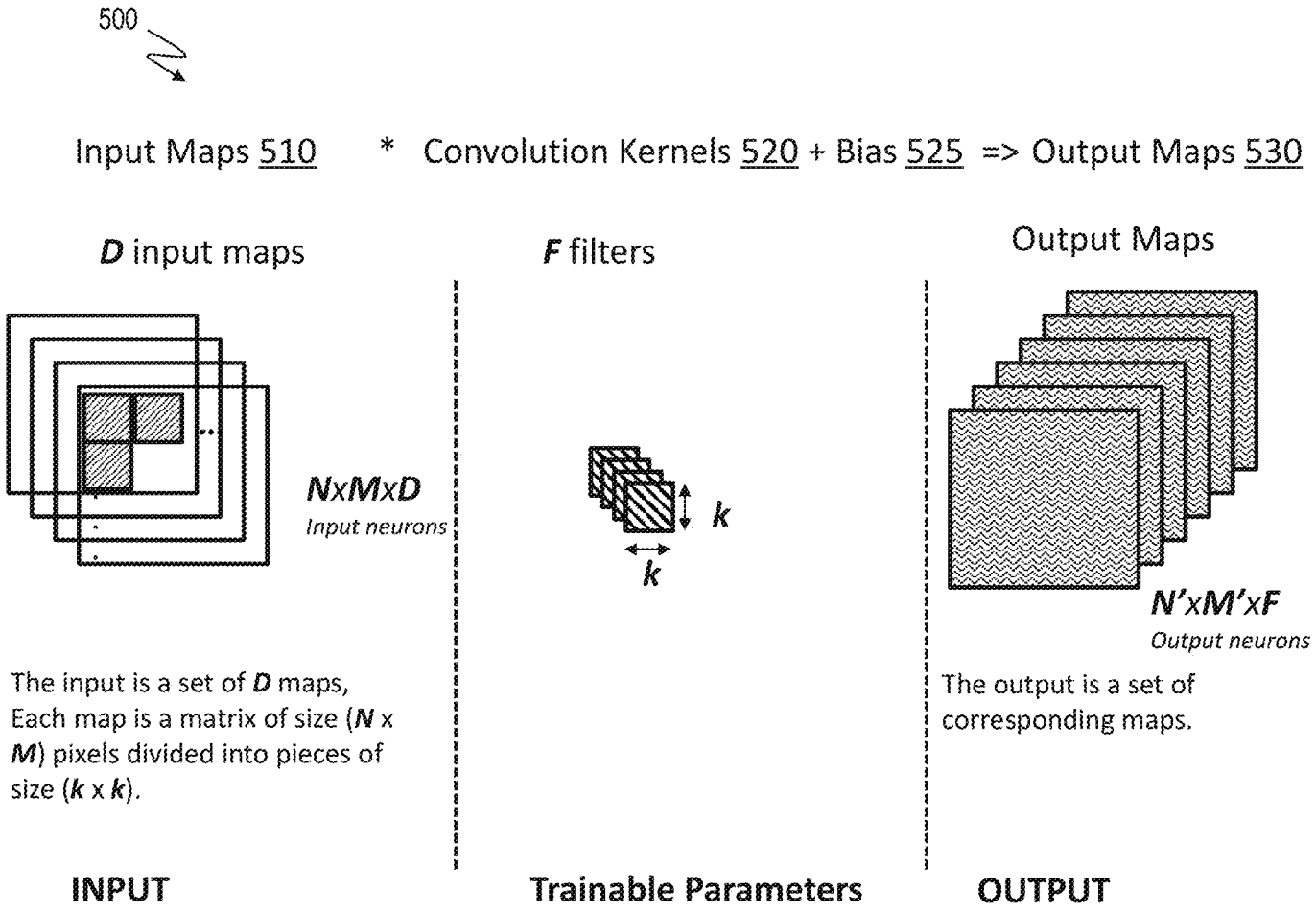

500
Input Maps 510 * Convolution Kernels 520 + Bias 525 => Output Maps 530
D input maps
F filters
Output Maps
NxMxD
Input neurons
k
k
N'xM'xF
Output neurons
The input is a set of D maps, Each map is a matrix of size (N x M) pixels divided into pieces of size (k x k).
The output is a set of corresponding maps.
INPUT
Trainable Parameters
OUTPUT

FIG. 5

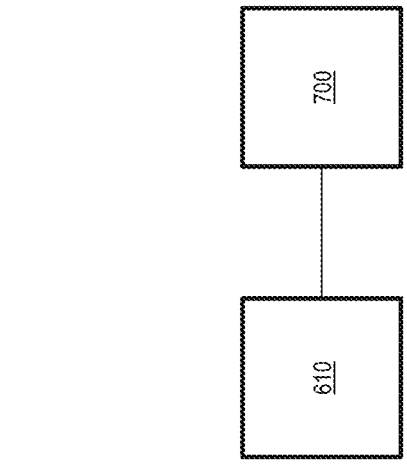


FIG. 6

600

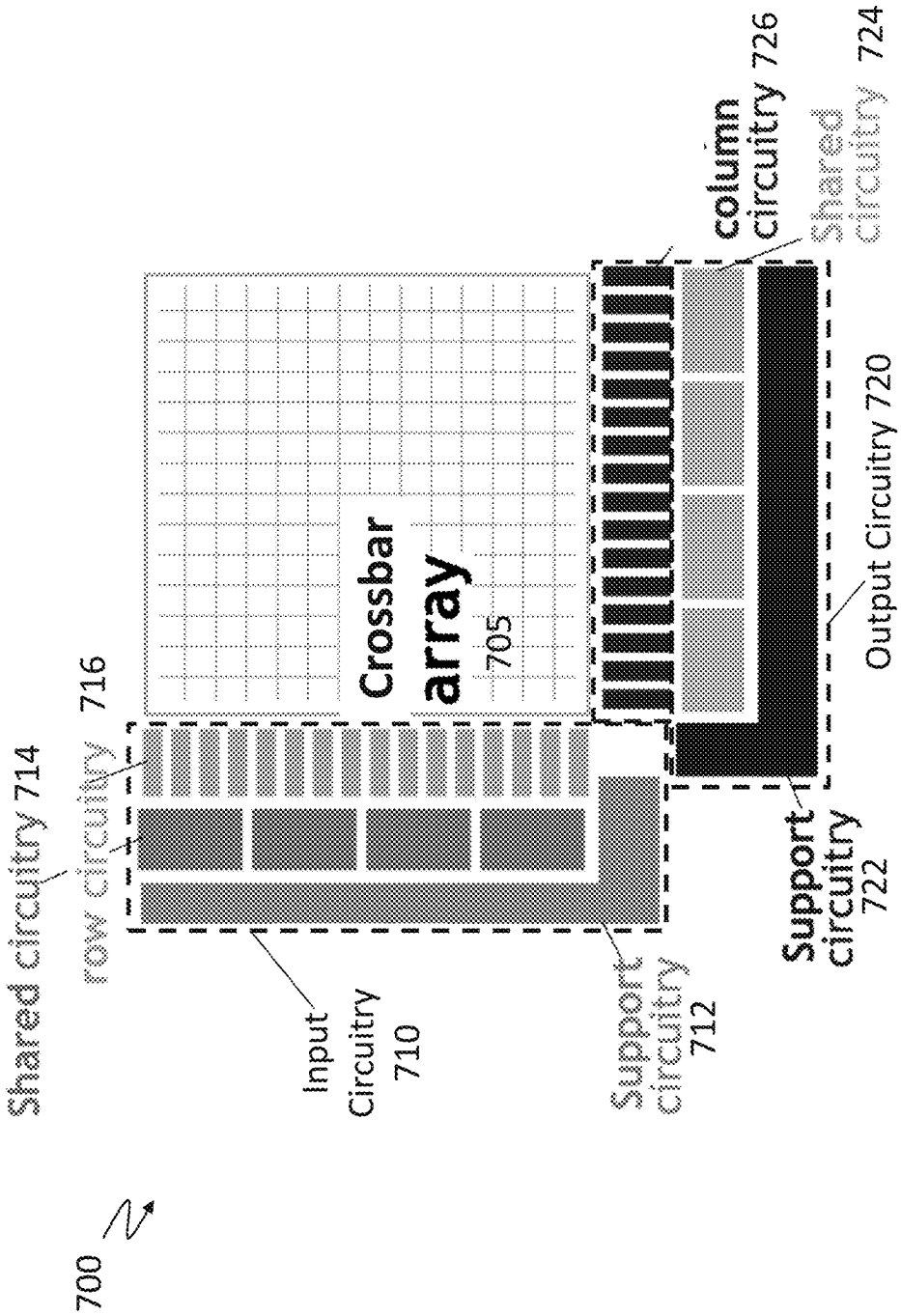

700
Shared circuitry 714
row circuitry 716
Crossbar array 705
Input Circuitry 710
Support circuitry 712
column circuitry 726
Shared circuitry 724
Output Circuitry 720
Support circuitry 722

FIG. 7

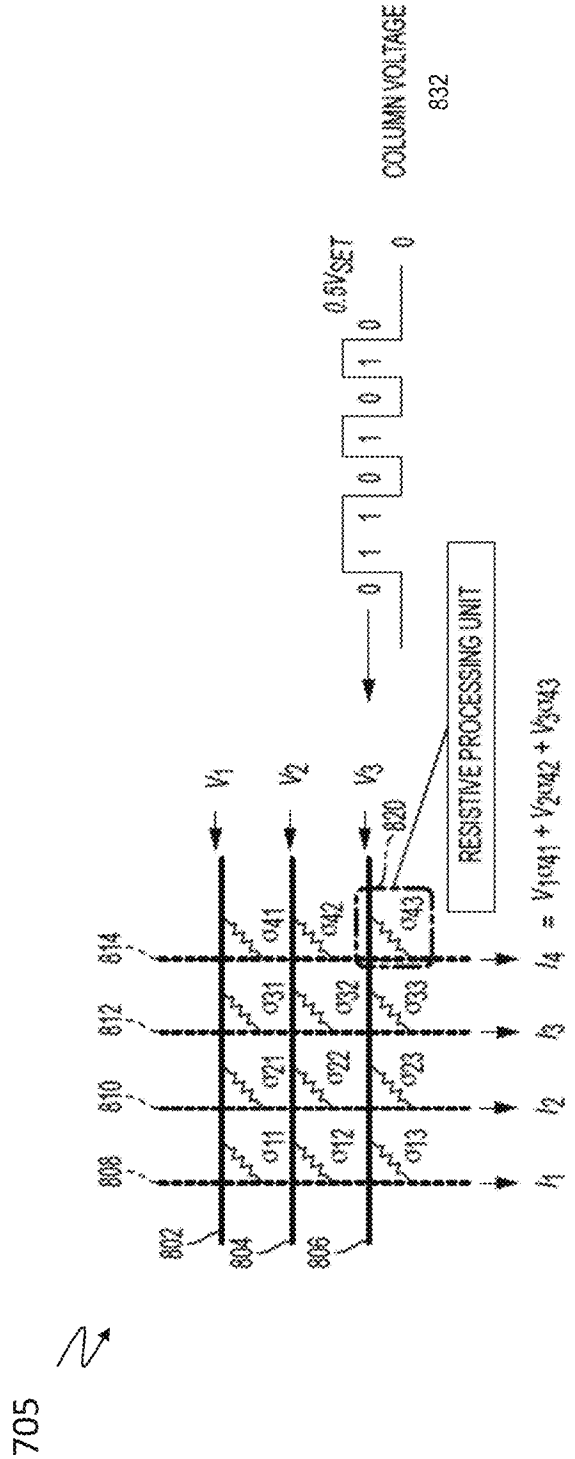

705
802
804
806
808
810
812
814
011
012
013
021
022
023
041
042
043
820
V_1
V_2
V_3
I_1
I_2
I_3
$I_4 = V_{1041} + V_{2042} + V_{2043}$
RESISTIVE PROCESSING UNIT
0.6V/SET
0
1
1
0
1
0
1
0
1
0
COLUMN VOLTAGE
832

FIG. 8

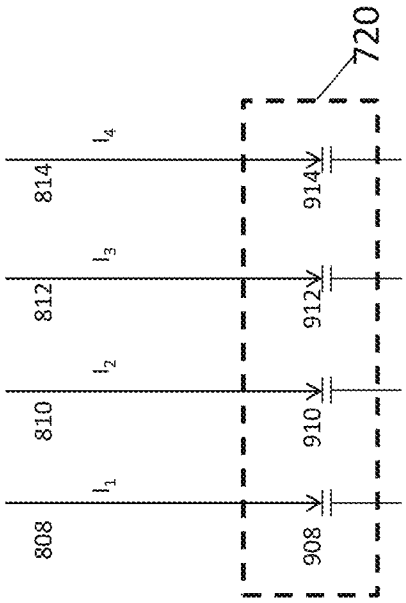


FIG. 9

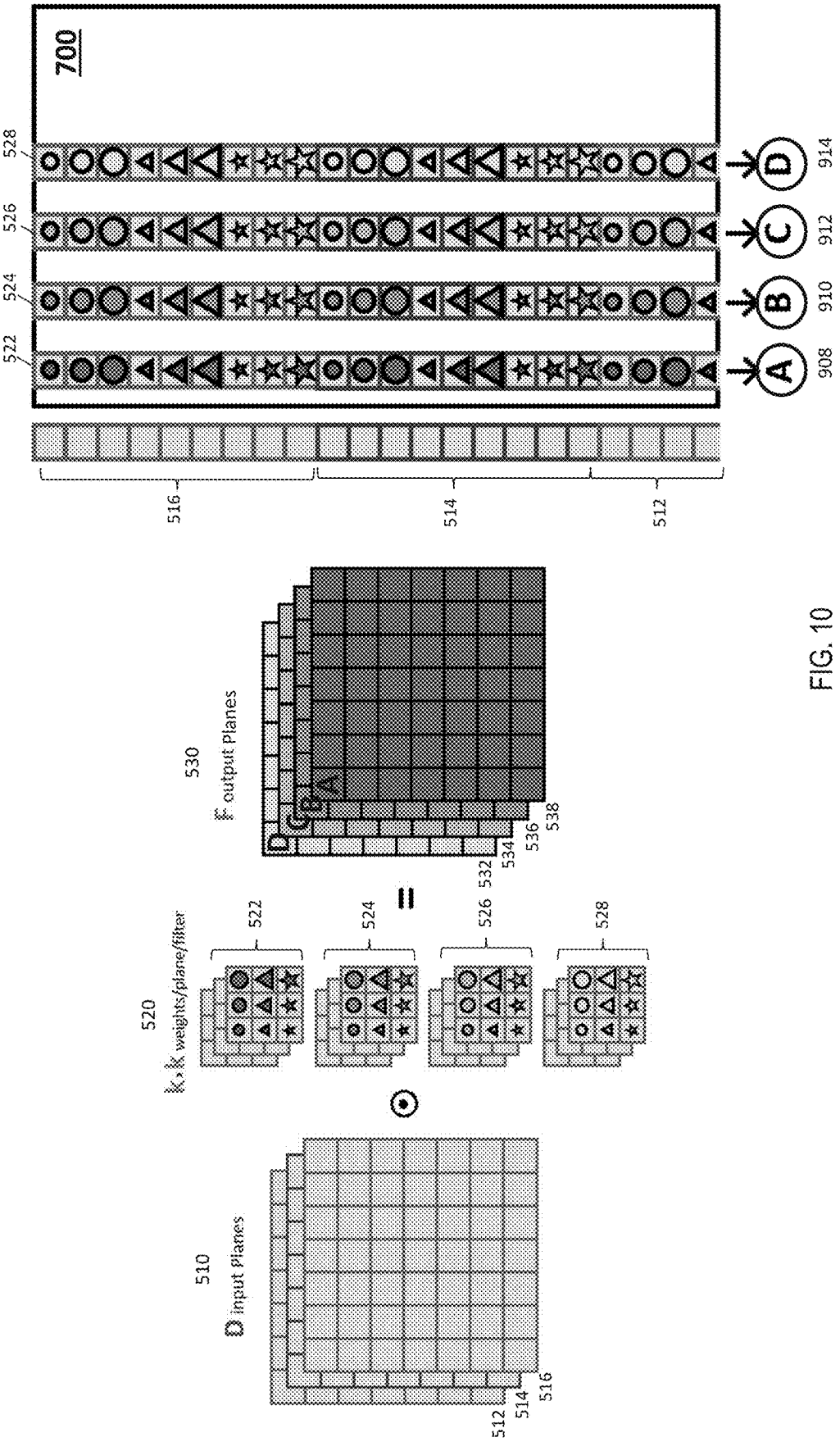

700
528
526
524
522
A
B
C
D
908
910
912
914
516
514
512
530
F output Planes
D
C
B
A
538
536
534
532
520
k x k weights/plane/filter
522
524
526
528
510
D input Planes
516
514
512

FIG. 10

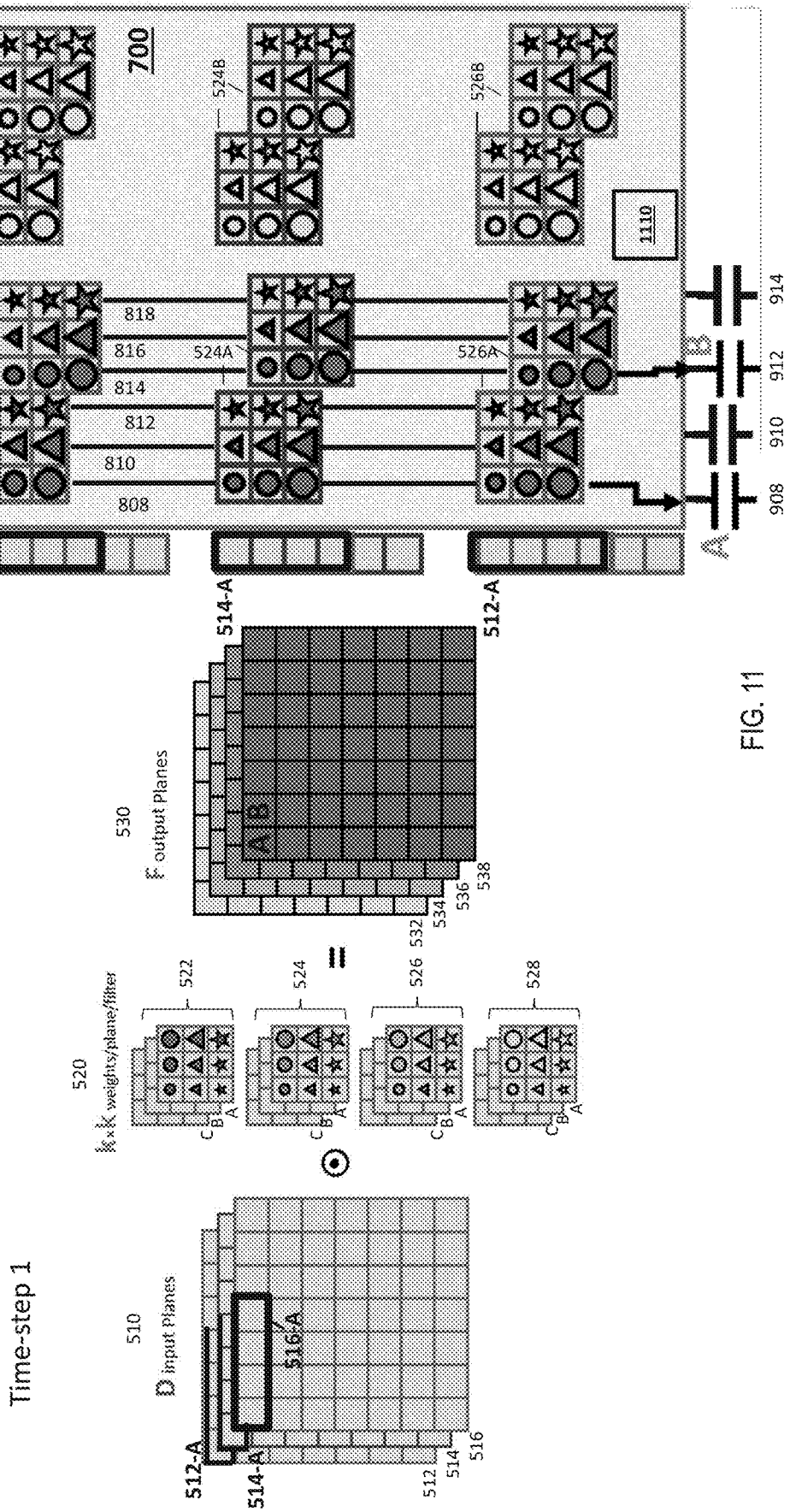

Time-step 1
510
D input Planes
512-A
514-A
516-A
512
514
516
520
K x K weights/plane/filter
522
524
526
528
530
F output Planes
532
534
536
538
700
522B
524B
526B
1110
522A
524A
526A
818
816
814
812
810
808
908
910
912
914
A
B

FIG. 11

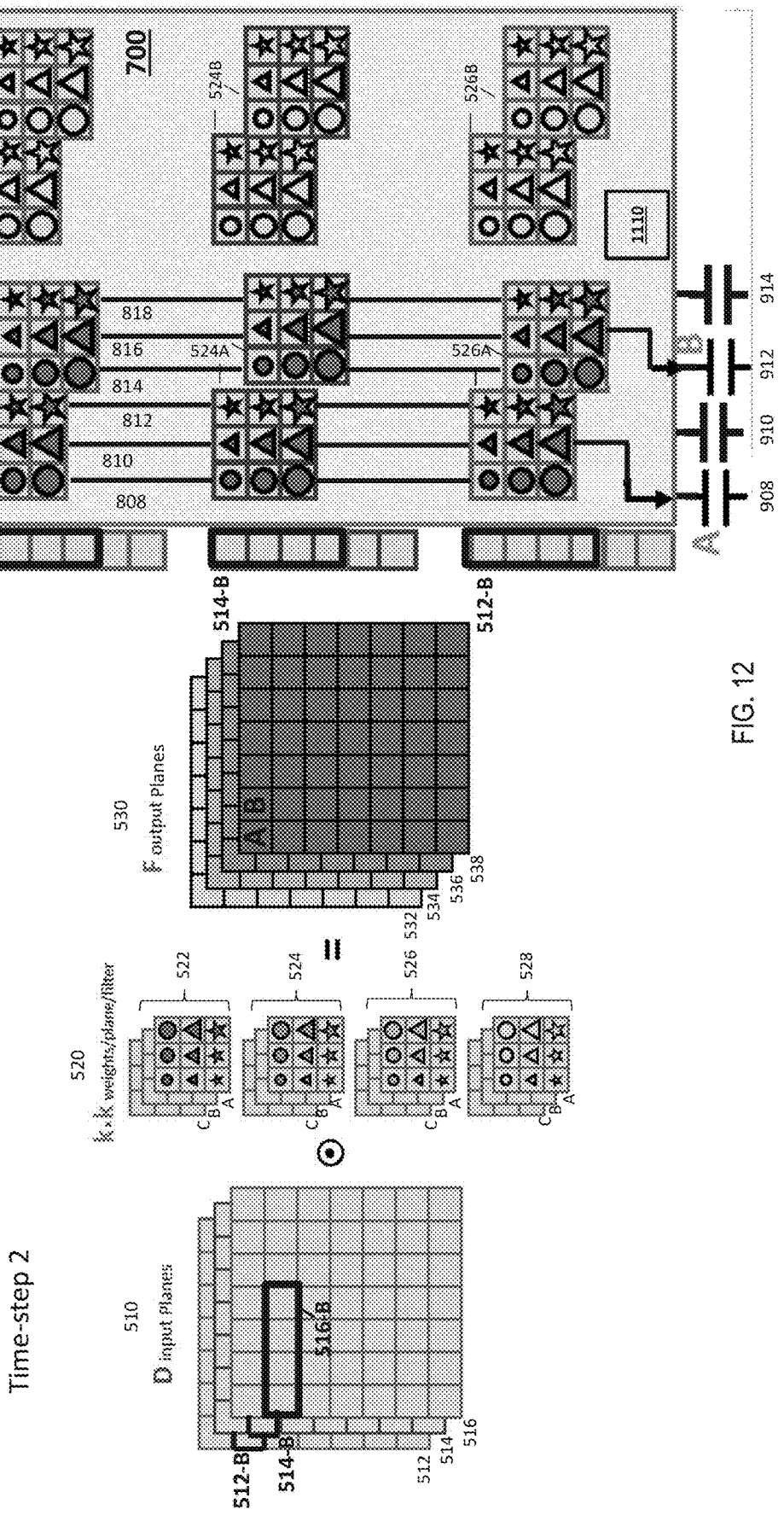

Time-step 2
510
D input planes
516-B
514-B
512-B
520
k × k weights/plane/filter
522
524
526
528
530
F output planes
700
1110
808
810
812
814
816
818
522A
524A
526A
522B
524B
526B
908
910
912
914

FIG. 12

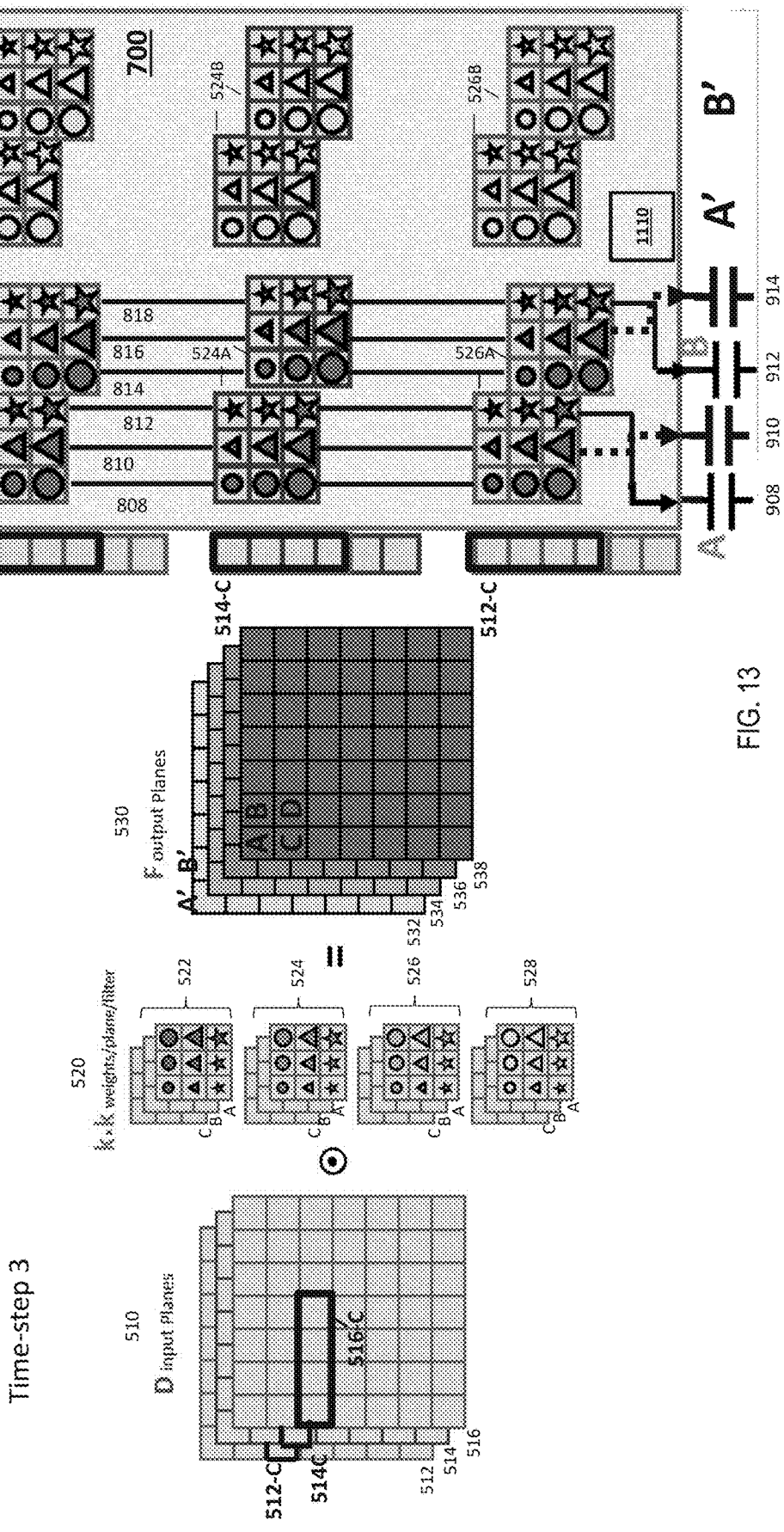

Time-step 3
510
D input planes
516C
512-C
514C
512
514
516
520
K x K weights/plane/filter
522
524
526
528
530
F output Planes
A' B'
532
534
536
538
700
522B
524B
526B
522A
524A
526A
818
816
814
812
810
808
516-C
514-C
512-C
1110
A
A'
B
B'
908
910
912
914

FIG. 13

EFFICIENT PROCESSING OF CONVOLUTIONAL NEURAL NETWORK LAYERS USING ANALOG-MEMORY-BASED HARDWARE

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This patent application claims priority to U.S. Provisional Patent Application Ser. No. 62/745,132, filed Oct. 12, 2018, which is incorporated herein by reference in its entirety.

BACKGROUND

[0002] The present invention relates in general to novel configurations of resistive crosspoint devices, which are referred to herein as resistive processing units (RPUs). More specifically, the present invention relates to performing operations of convolutional neural network layers using such crosspoint devices in crossbar arrays, such as in analog-memory-based hardware.

[0003] Technical problems such as character recognition and image recognition by a computer are known to be well handled by machine-learning techniques. "Machine learning" is used to broadly describe a primary function of electronic systems that learn from data. In machine learning and cognitive science, neural networks are a family of statistical learning models inspired by the biological neural networks of animals in particular, the brain. Neural networks can be used to estimate or approximate systems and functions that are generally unknown and depend on a large number of inputs. Neural networks use a class of algorithms based on a concept of inter-connected "neurons." In a typical neural network, neurons have a given activation function that operates on the inputs. By determining proper connection weights (a process also referred to as "training"), a neural network achieves efficient recognition of a desired patterns, such as images and characters. Oftentimes, these neurons are grouped into "layers" to make connections between groups more obvious and to organize the computation process. With these proper connection weights, other patterns of interest that have never been seen by the network during training can also be correctly recognized, a process known as "Forward Inference."

SUMMARY

[0004] According to one or more embodiments, a computer implemented method for implementing a convolutional neural network (CNN) using a crosspoint array or arrays includes configuring the crosspoint array(s) corresponding to a convolution layer in the CNN by storing one or more convolution kernels of the convolution layer in one or more crosspoint devices of each crosspoint array. The method further includes performing computations for the CNN via the crosspoint array by transmitting voltage pulses corresponding to a vector of input data of the convolution layer to the crosspoint array. Performing the CNN computations further include outputting an electric current representative of performing a multiplication operation at a crosspoint device in the crosspoint array based on a weight value stored by the crosspoint device and the voltage pulses from the input data. Performing the CNN computations further include passing the output electric current from the one or more crosspoint devices to a selected integrator.

[0005] According to one or more embodiments of the present invention, an electronic circuit for performing computations of a trained convolutional neural network (CNN) includes a crosspoint array, and an output circuit that includes one or more integrators. Performing the computations of the trained CNN comprises performing a method that includes configuring the crosspoint array(s) corresponding to a convolution layer in the CNN by storing one or more convolution kernels of the convolution layer in one or more crosspoint devices of each crosspoint array. The method further includes performing computations for the CNN via the crosspoint array by transmitting voltage pulses corresponding to a vector of input data of the convolution layer to the crosspoint array. Performing the CNN computations further include outputting an electric current representative of performing a multiplication operation at a crosspoint device in the crosspoint array based on a weight value stored by the crosspoint device and the voltage pulses from the input data. Performing the CNN computations further include passing the output electric current from the one or more crosspoint devices to a selected integrator.

[0006] According to one or more embodiments of the present invention, an electronic circuit includes an array of resistive memory elements. The array provides a vector of current outputs equal to an analog vector-matrix-product between (i) a vector of voltage inputs to the array encoding a vector of analog input values and (ii) a matrix of analog resistive weights within the array. The electronic circuit further includes accumulation wires and circuits aggregating a current from a dedicated subset of the resistive memory elements. The electronic circuit further includes integration capacitors, each of the integration capacitors being electrically switchable so as to aggregate current from one of a plurality of accumulation wires during a single integration step. The electronic circuit further includes data-output circuitry to allow an integrated charge from a subset of the integration capacitors, accumulated over a plurality of integration steps, to be suitably converted and transmitted either as an analog duration or as a digital representation using binary digits.

[0007] It is to be understood that the technical solutions are not limited in application to the details of construction and to the arrangements of the components set forth in the following description or illustrated in the drawings. The technical solutions are capable of embodiments in addition to those described and of being practiced and carried out in various ways. Also, it is to be understood that the phraseology and terminology employed herein, as well as the abstract, are for the purpose of description and should not be regarded as limiting. As such, those skilled in the art will appreciate that the conception upon which this disclosure is based may readily be utilized as a basis for the designing of other structures, methods and systems for carrying out the several purposes of the presently described technical solutions.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The examples described throughout the present document will be better understood with reference to the following drawings and description. The components in the figures are not necessarily to scale. Moreover, in the figures, like-referenced numerals designate corresponding parts throughout the different views.

[0009] FIG. **1** depicts a simplified diagram of input and output connections of a mathematical neuron;

[0010] FIG. **2** depicts a simplified model of the mathematical neuron shown in FIG. **1**;

[0011] FIG. **3** depicts a simplified model of an ANN incorporating the mathematical neuron model shown in FIG. **2**;

[0012] FIG. **4** illustrates a simplified block diagram of a representative CNN, which is interpreting a sample input map;

[0013] FIG. **5** illustrates an example convolutional layer in a CNN being trained using training data that include input maps and convolution kernels;

[0014] FIG. **6** depicts a system for performing a matrix-matrix multiplication using a crossbar array according to one or more embodiments of the present invention;

[0015] FIG. **7** depicts a two-dimensional (2D) crossbar system that performs forward matrix multiplication, backward matrix multiplication, and weight updates according to the present description;

[0016] FIG. **8** depicts an expanded view of the crossbar array according to one or more embodiments;

[0017] FIG. **9** depicts a typical output circuitry in a crossbar system;

[0018] FIG. **10** depicts existing operations to perform such operations using the crossbar array;

[0019] FIG. **11** depicts performing CNN operations using selective integrators according to one or more embodiments;

[0020] FIG. **12** depicts performing CNN operations using selective integrators according to one or more embodiments; and

[0021] FIG. **13** depicts performing CNN operations using selective integrators according to one or more embodiments.

DETAILED DESCRIPTION

[0022] The technical solutions described herein facilitate efficient implementation of deep learning techniques that use convolutional neural networks. Deep learning techniques are widely used in machine-based pattern recognition problems, such as image and speech recognition. Deep learning inherently leverages the availability of massive training datasets (that are enhanced with the use of Big Data) and computing power (that is expected to grow according to Moore's Law).

[0023] It is understood in advance that although one or more embodiments are described in the context of biological neural networks with a specific emphasis on modeling brain structures and functions, implementation of the teachings recited herein are not limited to modeling a particular environment. Rather, embodiments of the present invention are capable of modeling any type of environment, including for example, weather patterns, arbitrary data collected from the Internet, and the like, as long as the various inputs to the environment can be turned into a vector.

[0024] ANNs are often embodied as so-called "neuromorphic" systems of interconnected processor elements that act as simulated "neurons" and exchange "messages" between each other in the form of electronic signals. Similar to the so-called "plasticity" of synaptic neurotransmitter connections that carry messages between biological neurons, the connections in ANNs that carry electronic messages between simulated neurons are provided with numeric weights that correspond to the strength or weakness of a given connection. The weights can be adjusted and tuned based on experience, making ANNs adaptive to inputs and capable of learning. For example, an ANN for handwriting recognition is defined by a set of input neurons which can be activated by the pixels of an input image. After being weighted and transformed by a function determined by the network's designer, the activations of these input neurons are then passed to other downstream neurons, which are often referred to as "hidden" neurons. This process is repeated until an output neuron is activated. The activated output neuron determines which character was read.

[0025] Crossbar arrays, also known as crosspoint arrays, crosswire arrays, or resistive processing unit (RPU) arrays, are high density, low cost circuit architectures used to form a variety of electronic circuits and devices, including ANN architectures, neuromorphic microchips and ultra-high density nonvolatile memory. A basic crossbar array configuration includes a set of conductive row wires and a set of conductive column wires formed to intersect the set of conductive row wires. The intersections between the two sets of wires are separated by so-called crosspoint devices, which can be formed from thin film material.

[0026] Crosspoint devices, in effect, function as the ANN's weighted connections between neurons. Nanoscale two-terminal devices, for example memristors having "ideal" conduction state switching characteristics, are often used as the crosspoint devices in order to emulate synaptic plasticity with high energy efficiency. The conduction state (e.g., resistance) of the ideal memristor material can be altered by controlling the voltages applied between individual wires of the row and column wires. Digital data can be stored by alteration of the memristor material's conduction state at the intersection to achieve a high conduction state, a low conduction state, or any intermediate conductance state in between. The memristor material can also be programmed to maintain one of these distinct conduction states—high, low, or intermediate—by selectively setting the conduction state of the material. The conduction state of the memristor material can be read by applying a voltage across the material and measuring the current that passes through the target crosspoint device.

[0027] In order to limit power consumption, the crosspoint devices of ANN chip architectures are often designed to utilize offline learning techniques, wherein the approximation of the target function does not change once the initial training phase has been resolved. Offline learning allows the crosspoint devices of crossbar-type ANN architectures to be simplified such that they draw very little power.

[0028] Providing simple crosspoint devices that can implement Forward Inference of previously-trained ANN networks with low power consumption, high computational throughput, and low latency would improve overall ANN performance and allow a broader range of ANN applications.

[0029] Although the present invention is directed to an electronic system, for ease of reference and explanation various aspects of the described electronic system are described using neurological terminology such as neurons, plasticity and synapses, for example. It will be understood that for any discussion or illustration herein of an electronic system, the use of neurological terminology or neurological shorthand notations are for ease of reference and are meant to cover the neuromorphic, ANN equivalent(s) of the described neurological function or neurological component.

[0030] ANNs, also known as neuromorphic or synaptronic systems, are computational systems that can estimate or

approximate other functions or systems, including, for example, biological neural systems, the human brain and brain-like functionality such as image recognition, speech recognition, and the like. ANNs incorporate knowledge from a variety of disciplines, including neurophysiology, cognitive science/psychology, physics (statistical mechanics), control theory, computer science, artificial intelligence, statistics/mathematics, pattern recognition, computer vision, parallel processing and hardware (e.g., digital/analog/VLSI/optical).

[0031] Instead of utilizing the traditional digital model of manipulating zeros and ones, ANNs create connections between processing elements that are substantially the functional equivalent of the core system functionality that is being estimated or approximated. For example, a computer chip that is the central component of an electronic neuromorphic machine attempts to provide similar form, function, and architecture to the mammalian brain. Although the computer chip uses the same basic transistor components as conventional computer chips, its transistors are configured to mimic the behavior of neurons and their synapse connections. The computer chip processes information using a network of just over one million simulated "neurons," which communicate with one another using electrical spikes similar to the synaptic communications between biological neurons. The architecture of such a computer chip includes a configuration of processors (i.e., simulated "neurons") that read a memory (i.e., a simulated "synapse") and perform simple operations. The communications between these processors (pathways), which are typically located in different cores, are performed by on-chip network routers.

[0032] As background, a general description of how a typical ANN operates will now be provided with reference to FIGS. **1**, **2**, and **3**. As previously noted herein, a typical ANN is a mathematical model inspired by the human brain, which includes about one hundred billion interconnected cells called neurons. FIG. **1** depicts a simplified diagram of a mathematical neuron **102** having pathways **104**, **106**, **108**, **110** that connect it to upstream inputs **112**, **114**, downstream outputs **116**, and downstream "other" neurons **118**, configured and arranged as shown. Each mathematical neuron **102** sends and receives electrical impulses through pathways **104**, **106**, **108**, **110**. The nature of these electrical impulses and how they are processed in biological neurons (not shown) are primarily responsible for overall brain functionality. Mimicking this functionality is the intent of a mathematical ANN constructed from mathematical neurons **102** organized in a network. Just as the pathway connections between biological neurons can be strong or weak, so can the pathways between mathematical neurons. When a given neuron receives input impulses, the neuron processes the input according to the neuron's function and sends the result of the function to downstream outputs and/or downstream "other" neurons.

[0033] Mathematical neuron **102** is modeled in FIG. **2** as a node **202** having a mathematical function, f(x), depicted by the equation shown in FIG. **2**. Node **202** takes electrical signals from inputs **212**, **214**, multiplies each input **212**, **214** by the strength of its respective connection pathway **204**, **206**, takes a sum of the inputs, passes the sum through a function, f(x), and generates a result **216**, which can be a final output or an input to another node, or both. In the present description, an asterisk (*) is used to represent a multiplication, which can be a matrix multiplication. For example, the matrix multiplication can be used to perform convolution operations between input data and one or more convolution kernels to generate output maps. Weak input signals are multiplied by a very small connection strength number, so the impact of a weak input signal on the function is very low. Similarly, strong input signals are multiplied by a higher connection strength number, so the impact of a strong input signal on the function is larger. The function f(x) is a design choice, and a variety of functions can be used. A typical design choice for f(x) is the hyperbolic tangent function, which takes the function of the previous sum and outputs a number between minus one and plus one. An alternative design choice of f(x) is the ReLU or Rectified Linear Unit, a function in which the output matches the input for positive inputs and is zero otherwise.

[0034] FIG. **3** depicts a simplified ANN model **300** organized as a weighted directional graph, wherein the artificial neurons are nodes (e.g., **302**, **308**, **316**), and wherein weighted directed edges (e.g., m1 to m20) connect the nodes. ANN model **300** is organized such that nodes **302**, **304**, **306** are input layer nodes, nodes **308**, **310**, **312**, **314** are hidden layer nodes, and nodes **316**, **318** are output layer nodes. Each node is connected to every node in the adjacent layer by connection pathways, which are depicted in FIG. **3** as directional arrows having connection strengths m1 to m20. Although only one input layer, one hidden layer, and one output layer are shown, in practice, multiple input layers, hidden layers, and output layers can be provided.

[0035] In this attempt to mimic the functionality of a human brain, each input layer node **302**, **304**, **306** of ANN **300** receives inputs x1, x2, x3 directly from a source (not shown) with no connection strength adjustments and no node summations. Accordingly, y1=f(x1), y2=f(x2) and y3=f(x3), as shown by the equations listed at the bottom of FIG. **3**. Each hidden layer node **308**, **310**, **312**, **314** receives its inputs from all input layer nodes **302**, **304**, **306**, according to the connection strengths associated with the relevant connection pathways. Thus, in hidden layer node **308**, y4=f(m1*y1+m5*y2+m9*y3), wherein * represents a multiplication. In one or more examples, the multiplication can be a matrix multiplication used to perform a convolution operation. A similar connection strength multiplication and node summation is performed for hidden layer nodes **310**, **312**, **314** and output layer nodes **316**, **318**, as shown by the equations defining functions y5 to y9 depicted at the bottom of FIG. **3**.

[0036] ANN model **300** processes data records one at a time, and it "learns" by comparing an initially arbitrary classification of the record with the known actual classification of the record. Using a training methodology knows as "backpropagation" (i.e., "backward propagation of errors"), the errors from the initial classification of the first record are fed back into the network and used to modify the network's weighted connections the second time around, and this feedback process continues for many iterations. In the training phase of an ANN, the correct classification for each record is known, and the output nodes can therefore be assigned "correct" values, for example, a node value of "1" (or 0.9) for the node corresponding to the correct class, and a node value of "0" (or 0.1) for the others. It is thus possible to compare the network's calculated values for the output nodes to these "correct" values, and to calculate an error term for each node (i.e., the "delta" rule). These error terms

are then used to adjust the weights in the hidden layers so that in the next iteration the output values will be closer to the "correct" values.

[0037] There are many types of neural networks, but the two broadest categories are feed-forward and feedback/recurrent networks. ANN model **300** is a non-recurrent feed-forward network having inputs, outputs, and hidden layers. The signals used for forward-inference can only travel in one direction. Input data are passed onto a layer of processing elements that perform calculations. Each processing element makes its computation based upon a weighted sum of its inputs. The new calculated values then become the new input values that feed the next layer. This process continues until it has gone through all the layers and determined the output. A threshold transfer function is sometimes used to quantify the output of a neuron in the output layer.

[0038] A feedback/recurrent network includes feedback paths, which mean that the signals used for forward-inference can travel in both directions using loops. All possible connections between nodes are allowed. Because loops are present in this type of network, under certain operations, it can become a non-linear dynamical system that changes continuously until it reaches a state of equilibrium. Feedback networks are often used in associative memories and optimization problems, wherein the network looks for the best arrangement of interconnected factors, and in the learning of sequences of characters and/or words.

[0039] The speed and efficiency of machine learning in feed-forward and recurrent ANN architectures depend on how effectively the crosspoint devices of the ANN crossbar array perform the core operations of typical machine learning algorithms. Although a precise definition of machine learning is difficult to formulate, a learning process in the ANN context can be viewed as the problem of updating the crosspoint device connection weights so that a network can efficiently perform a specific task. The crosspoint devices typically learn the necessary connection weights from available training patterns. Performance is improved over time by iteratively updating the weights in the network. Instead of following a set of rules specified by human experts, ANNs "learn" underlying rules (like input-output relationships) from the given collection of representative examples. Accordingly, a learning algorithm can be generally defined as the procedure by which learning rules are used to update and/or adjust the relevant weights.

[0040] The three main learning algorithm paradigms are supervised, unsupervised, and hybrid. In supervised learning, or learning with a "teacher," the network is provided with a correct answer (output) for every input pattern. Weights are determined to allow the network to produce answers as close as possible to the known correct answers. Reinforcement learning is a variant of supervised learning in which the network is provided with only a critique on the correctness of network outputs, not the correct answers themselves. In contrast, unsupervised learning, or learning without a teacher, does not require a correct answer associated with each input pattern in the training data set. It explores the underlying structure in the data, or correlations between patterns in the data, and organizes patterns into categories from these correlations. Hybrid learning combines supervised and unsupervised learning. Parts of the weights are usually determined through supervised learning, while the others are obtained through unsupervised learning. Additional details of ANNs and learning rules are described in Artificial Neural Networks: A Tutorial, by Anil K. Jain, Jianchang Mao and K. M. Mohiuddin, IEEE, March 1996, the entire description of which is incorporated by reference herein.

[0041] Beyond the application of training ANNs, the Forward Inference of already trained networks includes applications ranging from implementations of cloud-based services built on ANNs to smartphone, Internet-Of-Things (IOT), and other battery-constrained applications which require extremely low power operation. In general, while training is an application that calls for high throughput (in order to learn from many training examples), Forward Inference is an application that calls for fast latency (so that any given new test example can be classified, recognized, or otherwise processed as rapidly as possible).

[0042] Described here are technical solutions for performing convolutional neural network computations using analog-memory-based hardware, such as crossbar arrays that include crosspoint devices. Deep Neural Network (DNN) accelerators based on crossbar arrays of non-volatile memories (NVMs)—such as Phase-Change Memory (PCM) or Resistive Memory (RRAM)—can implement multiply-accumulate operations that are extensively used in DNN acceleration in a parallelized manner. In such systems, computation occurs in the analog domain at the location of weight data encoded into the conductance (resistance) of the NVM devices. Such NVM devices are also referred to as RPU devices and crosspoint devices. The computation of multiply-accumulate operations can be mathematically described as vector-matrix multiplication between a vector of neuron excitations and a dense matrix of weights. The DNN computations for a Fully-Connected (FC) layer include such multiply-accumulate operations and, accordingly, using crossbar arrays to implement the FC layers of a DNN is computationally efficient.

[0043] In one or more examples, DNNs used for feature detection in input data include convolutional layers. Such DNNs are commonly referred to as convolutional neural networks (CNN). In a CNN, kernels convolute overlapping regions, such as those in a visual field, and accordingly emphasize the importance of spatial locality in feature detection. Computing the convolutional layers of the CNN typically encompasses more than 90% of computation time in neural network training and inference. Accelerating the forward-inference of CNN networks and reducing the amount of electrical power used, by performing the mathematical operations of the convolutional layers efficiently and with a minimum of extraneous data movement or computation, as described by the examples of the technical solutions herein, is a desirable improvement. As such the technical solutions are rooted in and/or tied to computer technology in order to overcome a problem specifically arising in the realm of computers, specifically neural networks, and more particularly convolutional neural networks.

[0044] However, in a convolutional layer as is used in many image-processing applications, multiple smaller vectors of neuron excitations (image patches) each are multiplied by smaller kernel matrices (filters). While this is advantageous for digital accelerators since there are fewer weights to retrieve from off-chip memory, the analog memory-based approach that increases efficiency for fully-connected layers is now at a disadvantage. If there is only one copy of the kernel matrices, then each vector of neuron

excitations must be computed in serial fashion, leading to computational performance that is not very interesting. Alternatively, multiple copies of the kernel matrices can be stored and operated simultaneously. However, the output excitations resulting from each copy of the kernel matrix must be organized, stored, duplicated, shuffled, and prepared to fill the neuron excitation vectors for the next convolutional layer. These operations significantly limit performance efficiency of the neural network by requiring digitization of the neuron excitation values and a significant amount of local digital storage and local digital processing, in order to convert raw output vectors into the next set of neuron excitation vectors.

[0045] The technical solutions described herein address such technical problems by facilitating the organization of the analog memory computations in such a way as to greatly simplify the processing and bookkeeping of the resulting computational outputs. In one or more examples, the analog memory computations are organized so that the neural network processes each set of inputs to a convolutional layer (an image with rows and columns, organized into multiple input "planes") one row (or column) at a time.

[0046] FIG. **4** illustrates a simplified block diagram of a CNN. In the depicted example, the CNN is being used for interpreting a sample input map **400**, and in this particular example uses a handwritten letter "w" as an input map. However, it is understood that other types of input maps are possible and also that the technical solutions described herein are applicable to a CNN performing other operations, such as other types of feature detections. In the illustrated example, the input map **100** is used to create a set of values for the input layer **410**, or "layer-**1**." For example, layer-**1** can be generated by direct mapping of a pixel of the sample input map **400** to a particular neuron in layer-**1**, such that the neuron shows a 1 or a 0 depending on whether the pixel exhibits a particular attribute. Another example method of assigning values to neurons is discussed below with reference to convolutional neural networks. Depending on the vagaries of the neural network and the problem it is created to solve, each layer of the network can have differing numbers of neurons, and these may or may not be related to particular qualities of the input data.

[0047] Referring to FIG. **4**, neurons in layer-**1 410** are connected to neurons in a next layer, layer-**2 420**, as described earlier (see FIG. **3**). The neurons in FIG. **4** are as described with reference to FIG. **1**. A neuron in layer-**2 420**, consequently, receives an input value from each of the neurons in layer-**1 410**. The input values are then summed and this sum compared to a bias. If the value exceeds the bias for a particular neuron, that neuron then holds a value, which can be used as input to neurons in the next layer of neurons. This computation continues through the various layers **430**-**450** of the CNN, which include at least one FC layer **450**, until it reaches a final layer **460**, referred to as "output" in FIG. **4**. In some CNN networks, "residual" results from earlier layers may be combined with the results of later layers, skipping over the layers in between. In an example of a CNN used for character recognition, each value in the layer is assigned to a particular character. When designed for classification tasks, the network is configured to end with the output layer having only one large positive value in one neuron, which then demonstrates which character the network has computed to be the most likely handwritten input character. In other scenarios, the network may have been designed such that output neuron values may be used to estimate probability (likelihood), confidence or other metrics of interest.

[0048] The data values for each layer in the CNN are typically represented using matrices (or tensors in some examples), and computations are performed as matrix computations. The indexes (and/or sizes) of the matrices vary from layer to layer and network to network, as illustrated in FIG. **4**. Different implementations orient the matrices or map the matrices to computer memory differently. Referring to FIG. **4**, in the example CNN illustrated, each level is a tensor of neuron values, as is illustrated by matrix dimensions for each layer of the neural network. At the input of the CNN, an example might be multiple input "planes," each a two-dimensional image. For instance, there might be a red plane, a green plane, and a blue plane, stemming from a full-color image. Deeper into the CNN, layers may take intermediate data in the form of many "planes" and produce for the next layer a large number of output planes. The values in an input tensor at a layer are multiplied by connection strengths, which are in a transformation tensor known as a filter. This matrix multiplication scales each value in the previous layer according to the connection strengths, with the aggregate total of these contributions then summed. This fundamental operation is known as a multiply-accumulate operation. A bias matrix may then be added to the resulting product matrix to account for the threshold of each neuron in the next level. Further, an activation function is applied to each resultant value, and the resulting values are placed in the output tensor to be applied to the next layer. In an example, the activation function can be rectified linear units, sigmoid, or tan h( ). Thus, as FIG. **4** shows, the connections between each layer, and thus an entire network, can be represented as a series of matrices. Training the CNN includes finding proper values for these matrices.

[0049] While fully-connected neural networks are able, when properly trained, to recognize input patterns, such as handwriting or photos of household pets, they do not exhibit shift-invariance. In order for the network to recognize the whiskers of a cat, it must be supplied with cat images with the whiskers located at numerous different 2-D locations within the image. Each different image location will lead to neuron values that interact with different weights in such a fully-connected network. In contrast, in a CNN, the connection strengths are convolution kernels. The convolution operation introduces shift-invariance. Thus, as multiple images are presented with cats with whiskers, as long as the scale, color, and rotation of the whiskers is unchanged from image to image, the 2-D position within the image no longer matters. Thus, during training, all examples of similar features work together to help learn this feature, independent of the feature location within the 2-D image. After training, a single or much smaller set of filters is sufficient to recognize such image features, allowing a bank of many filters (which is what a CNN layer is) to then recognize many different features that are useful for discriminating images (dogs from cats, or even subtleties that are representative of different breeds of cats).

[0050] FIG. **5** illustrates an example convolutional layer **500** in a CNN being trained using training data that include input maps **510** and convolution kernels **520**. For simplicity, FIG. **5** does not illustrate bias matrices **525**. The input maps **510** (also referred to as input planes) can include multiple input patterns, for example, D input maps. Each input map

is a matrix, such as a matrix of size N×M. Accordingly, a total number of input neurons in this case is N×M×D. The input maps are convolved with F convolution kernels **520** of size k×k as illustrated to produce corresponding output maps **530**. Each output map can have a dimension N'×M'. In case the input maps are square matrices of size n, the output maps are of size n−k+1×n−k+1. Each convolution is a 3D convolution involving the D input maps. A CNN can include multiple such layers, where the output maps **530** from a previous layer are used as input maps **510** for a subsequent layer. The backpropagation algorithm can be used to learn the k×k×D×F weight values of the filters.

[0051] For example, the input maps **510** are convolved with each filter bank to generate a corresponding output map. For example, in case the CNN is being trained to identify handwriting, the input maps **510** are combined with a filter bank that includes convolution kernels representing a vertical line. The resulting output map identifies vertical lines that are present in the input maps **510**. Further, another filter bank can include convolution kernels representing a diagonal line, such as going up and to the right. An output map resulting from a convolution of the input maps **510** with the second filter bank identifies samples of the training data that contain diagonal lines. The two output maps show different information for the character, while preserving pixel adjacency. This can result in more efficient character recognition.

[0052] FIG. **6** depicts a system **600** in which the crossbar array **700** is controlled using a controller **610** for performing the matrix-matrix multiplication among other operations according to one or more embodiments of the present invention. For example, the controller **610** sends the input data **510** to be multiplied by the crossbar array **700**. In one or more examples, the controller **610** stores the weight values, such as from convolution kernels **520**, in the crossbar array **700** and sends the input vectors. In one or more examples, the controller **610** and the crossbar array **700** are coupled in a wired or a wireless manner, or a combination thereof. The controller **610** further sends and instruction/command to the crossbar array **700** to initiate the operations for one or more layers in the CNN. The controller **610** further can read the output data **530** from the crossbar array **700** after receiving a notification that the computations have been performed. The controller **610** can be a processing unit, or a computing system, such as a server, a desktop computer, a tablet computer, a phone, and the like. The controller **610** can include a memory device that has computer executable instructions stored therein, the instructions when executed by the controller cause the matrix-matrix computation.

[0053] Turning now to an overview of the present description, one or more embodiments are directed to a two-terminal programmable resistive crosspoint component referred to herein as a resistive processing unit (RPU), which provides local data storage functionality and local data processing functionality. In other words, when performing data processing, the weighted contribution represented by each crosspoint device is contributed into a massively-parallel multiply-accumulate operation that is performed at the stored location of data. This eliminates the need to move relevant data in and out of a processor and a separate storage element. Accordingly, implementing a machine learning CNN architecture having the described crosspoint device enables the implementation of online machine learning capabilities that optimize the speed, efficiency, and power consumption when performing Forward-Inference of previously trained CNN models. The described crosspoint device and resulting CNN architecture improve overall CNN performance and enable a broader range of practical CNN applications.

[0054] The described crosspoint device can be implemented as a two-terminal resistive crosspoint device. For example, the described crosspoint device can be implemented with resistive random access memory (RRAM), phase change memory (PCM), programmable metallization cell (PMC) memory, non-linear memristive systems, or any other two-terminal device that offers a wide range to analog-tunable non-volatile resistive memory states that are sufficiently stable over time.

[0055] FIG. **7** depicts a two-dimensional (2D) crossbar system **700** that performs forward inference according to the present description. While such a crossbar system can be used to implement simple matrix multiplication, backward matrix-multiplication, and even in-situ weight-update according to the backpropagation algorithm, the present invention concerns the efficient implementation of convolutional layers for previously-trained networks. The crossbar system **700** includes a crossbar array **705**, an input circuitry **710**, and an output circuitry **720**, among other components. The crossbar system **700** can be a computer chip in one or more examples.

[0056] FIG. **8** depicts an expanded view of the crossbar array **705** according to one or more embodiments. The crossbar array **705** is formed from a set of conductive row wires **802**, **804**, **806** and a set of conductive column wires **808**, **810**, **812**, **814** that intersect the set of conductive row wires **802**, **804**, **806**. The intersections between the set of row wires and the set of column wires are separated by crosspoint devices, which are shown in FIG. **8** as resistive elements each having its own adjustable/updateable resistive weight, depicted as σ_{11}, σ_{21}, σ_{31}, σ_{41}, σ_{12}, σ_{22}, σ_{32}, σ_{42}, σ_{13}, σ_{23}, σ_{33} and σ_{43}, respectively. For ease of illustration, only one crosspoint device **820** is labeled with a reference number in FIG. **8**. In forward matrix multiplication, the conduction state (i.e., the stored weights) of the crosspoint device can be read by applying a voltage across the crosspoint device and measuring the current that passes through the crosspoint device.

[0057] Input voltages V_1, V_2, V_3 are applied to row wires **802**, **804**, **806**, respectively. Each column wire **808**, **810**, **812**, **814** sums the currents I_1, I_2, I_3, I_4 generated by each crosspoint device along the particular column wire using an integrator, such as a capacitor. For example, as shown in FIG. **8**, the current **14** generated by column wire **814** is given by the equation $I_4=V_{1}\sigma_{41}+V_{2}\sigma_{42}+V_{3}\sigma_{43}$. Thus, array **700** computes the forward matrix multiplication by multiplying the values stored in the crosspoint devices by the row wire inputs, which are defined by voltages V_1, V_2, V_3.

[0058] Referring to FIG. **7**, the input circuitry **710** includes, in one or more examples, at least a support circuitry **712**, a shared circuitry **714**, and a row circuitry **716**. The row circuitry includes hardware components associated with each row wire **802**, **804**, and **806**. The input circuitry **710** facilitates providing the input voltages to the crossbar array **705**.

[0059] FIG. **9** depicts a typical output circuitry **720**. The output circuitry includes integrators **908**, **910**, **912**, and **914** corresponding to the column wires **808**, **810**, **812**, and **814**, respectively. The integrators **908**, **910**, **912**, and **914**, in one

or more examples, are capacitors. The output currents along each column wire are accumulated in the integrators and passed on to a next layer of the CNN. As described earlier, such an arrangement of the integrators makes the computations of the FC layers very efficient; however, for the convolution operations, to use such an arrangement of the integrators incurs significant additional overhead in terms of data transport, storage, organization and subsequent data transport. Such operations require additional resources such as time, power, and additional circuit-area, thus making the overall system inefficient.

[0060] FIG. **10** depicts existing operations to perform such operations using the crossbar array. It should be noted that the dimensions of the matrices shown in the figures herein are just examples, and in one or more examples different dimensions can be used.

[0061] As depicted in FIG. **10**, one image-row (**512**, **514**, and **516**) of all input planes **510** is presented concurrently as a column of inputs to the array-rows (**802**, **804**, and **806**) of the crossbar array **705** of the crossbar system **700**. The crosspoint devices **820** at each crosspoint contains weight-elements from the filters **525**, each leading to a multiplication between the array-row excitation, x_i, and the stored weight, w_{ij} by Ohm's law (voltage times conductance equals current). The integration of all such read current contributions is summed along each array-column and stored in the corresponding integrators (**908**, **910**, **912**, and **914**) of the array-columns (**808**, **810**, **812**, and **814**). The computation can be expressed as: the current I_1 on column #1 (**808**) is stored on capacitor C_1 (**908**), I_2 is stored on capacitor C_2, I_3 on C_3, and so on. In the existing technical solutions that use such crossbar arrays **705**, the integrated charge on the capacitors (**908**, **910**, **912**, and **914**) is treated as the output of the multiply-accumulate operation and is either converted to a digital number or to pulse-duration for shipment to a next array **705**.

[0062] In this manner, at each time-step (i.e., each computation performed by the array **705**), values across all input planes **510** are integrated producing an output for all output planes **530**. However, this results only in one output pixel per time-step.

[0063] Further, every output from convolutional layer i has to be combined with outputs from other convolutional layers as part of pooling. The other convolutional layers from which the outputs that are to be pooled depend on the number of elements in the filter kernels **520**. Alternatively, or in addition, every output from layer i has to be positioned at different spots in the input planes **510** for the convolutional layer i+1. Such organization of the output values for the purpose of pooling can also require additional computing resources, such as read-write access, power and the like.

[0064] The technical solutions described herein address technical challenges of existing technical solutions by facilitating, after the multiply-accumulate operations are performed, the steering of the aggregate current to a selected integrator, from any of the integrators in the output circuitry **720**. For instance, current I_1 might now be steered to capacitor C_2, I_2 to capacitor C_3, and I_3 to capacitor C_1, instead of retaining the charges in the same columns, with the next image-row of the input planes to this convolutional layer being similarly presented to the same array **705**. The purpose of this is to allow each capacitor to integrate the total current contributions for different columns of the k-by-k weight kernel substantially simultaneously (each driven into the array by various array-row excitations) and for different rows of the weight kernel in time (added to any given capacitor over k different time-steps by steering the aggregate current from the array-column corresponding to the appropriate weight kernel coefficients).

[0065] "Pooling" as used in neural network operations can include determining results such as the maximum, sum, or average of the output excitations. The technical solutions described herein facilitate such pooled results being computed locally and then transmitted, only after all relevant weight kernels are fully integrated. In an alternative embodiment, the unpooled results are computed locally, and are only pooled after transmission.

[0066] FIGS. **11-14** depict the operations performed by the array **705** with the modified output circuit **720**, according to one or more embodiments. At each time-step, each of the integrators (**908**, **910**, **912**, and **914**) receives contributions from k*D multiply-accumulate terms, where D is the number of input planes **510**. After k time-steps, the total charge on an integrator contains all k*k*D terms and is ready to be output to the next convolutional layer. Except for during the first k or last k time-steps, after each integration step, every kth integrator from the output circuit **720** reaches this status, and accordingly, is ready to generate all the output pixels of one image-row (**512**-A, **514**-A, and **516**-A) of the convolutional-layer output. All other jth integrators have a different phase in their respective integration phase, depending the value of j.

[0067] For example, as shown in FIG. **11**, the first rows of each input plane **512**-A, **514**-A, **516**-A are input to the convolutional layer. The crosspoint devices **820** of the crossbar array **705** are loaded with the filters **520** as shown. Particularly, filter kernels **522**-A and **522**-B are loaded in the crosspoint devices **820** to perform a convolution with the first rows of the first input plane **516**-A. Similarly, filter kernels **524**-A and **524**-B from a second bank of filter kernels **520** are convolved with the first row of a second input plane **514**-A, and so on. The results of the respective convolutions are forwarded to one or more of the integrators (**908**, **910**, **912**, **914**) from the output circuitry **720** by output controller **1110**.

[0068] The output controller **1110** can be part of the output circuitry **720** or an external controller that is coupled with the output circuitry **720**. The output controller **1110** steers the output of the multiply-accumulate operations from each column in the array **705** to a particular integrator in the output circuitry **720**. In one or more examples, the output controller **1110** receives a mode signal that provides a selection of the integrators for each column at each time-step. Alternatively, the output controller **1110** is provided a mode signal that indicates the selection of the integrator for each column until all convolutional layers are executed. The mode signal, in one or more examples, can be a bit pattern that is indicative of the selected integrators for each column.

[0069] In the example of FIG. **11**, the outputs from the columns **808** and **814** are stored in the integrators **908** and **912**, respectively, at time-step #1. FIG. **12** depicts the operations performed in time-step #2. Here, second rows **512**-B, **514**-B, and **516**-B from the input planes **510** are used as input to the crosspoint array **705**. The crosspoint devices **820** are still loaded with the kernel filters **520** as in time-step #1 (FIG. **11**). In the time-step #2, the output controller **1110** selects the same integrators **908** and **912** for the outputs of the columns **810** and **816** (different from time-step #1).

Accordingly, the integrators **908** and **912**, in this case, receive outputs from different columns in different time-steps.

[0070] FIG. **13** depicts the operations performed in time-step #3. In a manner similar to the first two time-steps, in time-step #3, a third row **512**-C, **514**-C, and **516**-C from the input planes **510** is used as input to the crosspoint array **705**. In the time-step #3, the output controller **1110** selects the same integrators **908** and **912** for the outputs of the columns **812** and **818** (different from time-step #1). Accordingly, the integrators **908** and **912**, in this case, receive outputs from different columns in different time-steps. In this manner, in general, after k time-steps, an entire row in the output planes **530** is computed (compared to a single output pixel in the existing solution).

[0071] It should be noted that, while the only the computations of the first two entries (A and B) from the first output row in the output plane **530** is described above, in a similar manner, the other portions of the output planes **530** are computed in parallel by other portions of the crosspoint array **705**. Further yet, the crosspoint array **705** can be accumulating computation outputs for other output rows (C and D) at each time-step using the other integrators (**910** and **914**) as shown in FIG. **13**.

[0072] Accordingly, as a result of the output controller **1110** steering the output of the crosspoint array **705**, all input is in the form of a complete and contiguous image-row over all input planes. Further, after the first k time-steps before any output is available, (that is, from the $k+1^{th}$ time-step), a complete and contiguous image-row over all the output planes is produced at each time-step. Accordingly, the output maps **530** produced by such operations can be pipelined to a subsequent convolutional layer without any intermediate storage of the neuron excitations. Because pooling operations such as sum, average, and maximum can be performed incrementally on data as they arrive, any pooling operation only requires temporary storage sufficient for the output image-row. These intermediate results are stored and updated as each set of neuron excitations arrive until the R-by-R pooling operation is complete, at which point the buffer of intermediate results is effectively the output of the pooling layer.

[0073] It should be noted that although in the examples used in the above description to explain the technical solutions, a single image-row is used for calculations, in one or more examples, more than a single image-row can be used. For example, in an alternative embodiment, two image-rows of the output planes **530** are output simultaneously, and so on. The output rows are further supplied as the data for pooling operations, for example, a 2×2 pooling operation can be performed simultaneously using the two output rows. In such examples with additional output rows, the need to organize, store, or even transmit the output data elsewhere is eliminated by steering the output to the integrators in the output circuitry **720** of the crossbar system **700** itself.

[0074] The examples herein use k=3 in most cases, however, it is understood that k can be any other value in other examples.

[0075] The technical solutions described herein accordingly facilitate improving performance efficiency in terms of speed, computing resources, and power used when implementing a CNN. Empirical data for the inventors suggest the improvements are at least an order of magnitude in some cases. The technical solutions described herein are rooted in computer technology, particularly implementing CNN using a neural network computing chip that is typically configured to increase efficiency of fully connected layers in the CNN by performing multiply-accumulate operations along a column of the crossbar array. The technical solutions described herein allow the computer chip to maintain those efficiencies, and in addition, to be configured during convolutional layer computations to steer output of the columns to particular integrators in the crossbar array, and to maintain the output in the integrators and directly provide that output to subsequent convolutional layers. Such operations reduce, if not eliminate, read-write operations and digitization operations of outputs of each convolutional layer.

[0076] It should also be noted that although the examples described herein use rows of the input planes **510** to perform the computations of the CNN, in one or more examples, the columns can be used with corresponding adjustments to the matrices in the operations, as will be obvious to a person skilled in the art.

[0077] The technical solutions described herein accordingly provide a circuit that includes an array of resistive memory elements, the array providing a vector of current outputs equal to the analog vector-matrix-product between (i) a vector of voltage inputs to the array encoding a vector of analog input values and (ii) a matrix of analog resistive weights within the array. The circuit further includes accumulation wires and circuits aggregating the current from a dedicated subset of the resistive elements. Further, the circuit includes integration capacitors, each of the integration capacitors being electrically switchable (selectable) so as to aggregate current from at least one of the accumulation wires during a single integration step. The circuit also includes data-output circuitry to allow the integrated charge from a subset of the integration capacitors, accumulated over multiple integration steps, to be converted and transmitted either as an analog duration or as a digital representation using binary digits.

[0078] The subset of resistive elements can include one or more column of the array. Alternatively, the subset of resistive elements can include one or more rows of the array. In one or more examples, the resistive elements are non-volatile memory devices. In one or more examples, the resistive elements store synaptic weights of a neural network.

[0079] In one or more examples, the resistive memory elements are arranged so as to implement the columns of the weight kernels of a given layer of a convolutional neural network. The accumulation over the integration steps implements the multiply-accumulate operations across multiple rows of said weight kernels, as the input neuron excitations to the said layer of the convolutional neural network are presented one row at a time. Further, the integrated charge representing an output excitation is suitably converted and transmitted only after all rows of said weight kernel are fully integrated.

[0080] Further, in one or more examples, the integrated charge stored by multiple capacitors representing respective output excitations are suitably converted and a suitable pooled result such as the maximum, sum, or average of the said plurality of output excitations is computed locally and then transmitted, only after all relevant weight kernels are fully integrated.

[0081] In one or more examples, the resistive memory elements are arranged so as to implement the rows of the weight kernels of a given layer of a convolutional neural network. The accumulation over the integration steps implements the multiply-accumulate operations across multiple columns of said weight kernels, as the input neuron excitations to the said layer of the convolutional neural network are presented one column at a time. Further, the integrated charge representing an output excitation is suitably converted and transmitted only after all columns of said weight kernel are fully integrated. Further, in one or more examples, the integrated charge stored by multiple capacitors representing respective output excitations are suitably converted and a suitable pooled result such as the maximum, sum, or average of the said plurality of output excitations are computed locally and then transmitted, only after all relevant weight kernels are fully integrated.

[0082] The present technical solutions may be a system, a method, and/or a computer program product at any possible technical detail level of integration. The computer program product may include a computer readable storage medium (or media) having computer readable program instructions thereon for causing a processor to carry out aspects of the present technical solutions.

[0083] The computer readable storage medium can be a tangible device that can retain and store instructions for use by an instruction execution device. The computer readable storage medium may be, for example, but is not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing. A non-exhaustive list of more specific examples of the computer readable storage medium includes the following: a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), a static random access memory (SRAM), a portable compact disc read-only memory (CD-ROM), a digital versatile disk (DVD), a memory stick, a floppy disk, a mechanically encoded device such as punch-cards or raised structures in a groove having instructions recorded thereon, and any suitable combination of the foregoing. A computer readable storage medium, as used herein, is not to be construed as being transitory signals per se, such as radio waves or other freely propagating electromagnetic waves, electromagnetic waves propagating through a waveguide or other transmission media (e.g., light pulses passing through a fiber-optic cable), or electrical signals transmitted through a wire.

[0084] Computer readable program instructions described herein can be downloaded to respective computing/processing devices from a computer readable storage medium or to an external computer or external storage device via a network, for example, the Internet, a local area network, a wide area network and/or a wireless network. The network may comprise copper transmission cables, optical transmission fibers, wireless transmission, routers, firewalls, switches, gateway computers and/or edge servers. A network adapter card or network interface in each computing/processing device receives computer readable program instructions from the network and forwards the computer readable program instructions for storage in a computer readable storage medium within the respective computing/processing device.

[0085] Computer readable program instructions for carrying out operations of the present technical solutions may be assembler instructions, instruction-set-architecture (ISA) instructions, machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, configuration data for integrated circuitry, or either source code or object code written in any combination of one or more programming languages, including an object oriented programming language such as Smalltalk, C++, or the like, and procedural programming languages, such as the "C" programming language or similar programming languages. The computer readable program instructions may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider). In some embodiments, electronic circuitry including, for example, programmable logic circuitry, field-programmable gate arrays (FPGA), or programmable logic arrays (PLA) may execute the computer readable program instructions by utilizing state information of the computer readable program instructions to personalize the electronic circuitry, in order to perform aspects of the present technical solutions.

[0086] Aspects of the present technical solutions are described herein with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems), and computer program products according to embodiments of the technical solutions. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer readable program instructions.

[0087] These computer readable program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks. These computer readable program instructions may also be stored in a computer readable storage medium that can direct a computer, a programmable data processing apparatus, and/or other devices to function in a particular manner, such that the computer readable storage medium having instructions stored therein comprises an article of manufacture including instructions which implement aspects of the function/act specified in the flowchart and/or block diagram block or blocks.

[0088] The computer readable program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other device to cause a series of operational steps to be performed on the computer, other programmable apparatus or other device to produce a computer implemented process, such that the instructions which execute on the computer, other programmable apparatus, or other device implement the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0089] The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods, and computer program products according to various embodiments of the present technical solutions. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of instructions, which comprises one or more executable instructions for implementing the specified logical function(s). In some alternative implementations, the functions noted in the blocks may occur out of the order noted in the Figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts or carry out combinations of special purpose hardware and computer instructions.

[0090] A second action may be said to be "in response to" a first action independent of whether the second action results directly or indirectly from the first action. The second action may occur at a substantially later time than the first action and still be in response to the first action. Similarly, the second action may be said to be in response to the first action even if intervening actions take place between the first action and the second action, and even if one or more of the intervening actions directly cause the second action to be performed. For example, a second action may be in response to a first action if the first action sets a flag and a third action later initiates the second action whenever the flag is set.

[0091] To clarify the use of and to hereby provide notice to the public, the phrases "at least one of <A>, <B>, . . . and <N>" or "at least one of <A>, <B>, <N>, or combinations thereof" or "<A>, <B>, . . . and/or <N>" are to be construed in the broadest sense, superseding any other implied definitions hereinbefore or hereinafter unless expressly asserted to the contrary, to mean one or more elements selected from the group comprising A, B, . . . and N. In other words, the phrases mean any combination of one or more of the elements A, B, . . . or N including any one element alone or the one element in combination with one or more of the other elements which may also include, in combination, additional elements not listed.

[0092] It will also be appreciated that any module, unit, component, server, computer, terminal or device exemplified herein that executes instructions may include or otherwise have access to computer readable media such as storage media, computer storage media, or data storage devices (removable and/or non-removable) such as, for example, magnetic disks, optical disks, or tape. Computer storage media may include volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data. Such computer storage media may be part of the device or accessible or connectable thereto. Any application or module herein described may be implemented using computer readable/executable instructions that may be stored or otherwise held by such computer readable media.

[0093] The descriptions of the various embodiments of the technical features herein have been presented for purposes of illustration but are not intended to be exhaustive or limited to the embodiments disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope and spirit of the described embodiments. The terminology used herein was chosen to best explain the principles of the embodiments, the practical application or technical improvement over technologies found in the marketplace, or to enable others of ordinary skill in the art to understand the embodiments disclosed herein.

What is claimed is:

1. A computer implemented method for implementing a convolutional neural network (CNN) using a crosspoint array, the method comprising:

configuring the crosspoint array, the crosspoint array corresponding to a convolution layer in the CNN, by storing one or more convolution kernels of the convolution layer in one or more crosspoint devices of the crosspoint array; and

performing computations for the CNN via the crosspoint array by:

transmitting voltage pulses corresponding to a vector of input data of the convolution layer to the crosspoint array;

outputting an electric current representative of performing a multiplication operation at a crosspoint device in the crosspoint array, the electric current based on a weight value stored by the crosspoint device and the voltage pulses from the input data; and

passing the output electric current from the crosspoint device to a selected integrator.

2. The computer implemented method of claim **1**, wherein the integrator is a capacitor.

3. The computer implemented method of claim **1**, wherein the output electric current is generated by the crosspoint device, the crosspoint device being at an intersection of a first row wire of the crosspoint array and a first column wire of the crosspoint array, and said integrator is physically proximate to a second column wire of the crosspoint array, and is electrically coupled to said first column wire of the crosspoint array in order to receive said output electric current.

4. The computer implemented method of claim **1**, wherein the output electric current is a first output electric current, the crosspoint device being a first crosspoint device that is at an intersection of a first row wire of the crosspoint array and a first column wire of the crosspoint array, the method comprising:

outputting a second electric current by a second crosspoint device in the crosspoint array, based on a weight value stored by the second crosspoint device and the voltage pulses from the input data, the second crosspoint device being at an intersection of a second row wire of the crosspoint array and a second column wire of the crosspoint array; and

passing the second output electric current from the crosspoint device to said selected integrator, where said integrator is physically proximate to said first column wire of the crosspoint array, and is electrically coupled to said second column wire of the crosspoint array in order to receive said second electric current.

5. The computer implemented method of claim 1, wherein the selected integrator is selected by an output controller, based on a mode signal that maps the output electric current from the crosspoint device to the selected integrator.

6. The computer implemented method of claim 1, wherein the crosspoint devices are arranged to implement one or more columns of the convolution kernels of a given layer of the CNN, and wherein the vector of input data represents neuron excitations to the given layer of the CNN presented from the input data, one row at a time.

7. The computer implemented method of claim 6, wherein a charge held by the selected integrator represents an output excitation according to the given layer of the CNN, the output excitation being converted and transmitted only after all rows of said convolution kernel are integrated.

8. The computer implemented method of claim 1, wherein the crosspoint devices are arranged to implement one or more rows of the convolution kernels of a given layer of the CNN, and wherein the vector of input data represents neuron excitations to the given layer of the CNN presented one column at a time.

9. The computer implemented method of claim 8, wherein a charge held by the selected integrator represents an output excitation according to the given layer of the CNN, the output excitation being converted and transmitted only after all columns of said convolution kernel are integrated.

10. An electronic circuit for performing computations of a trained convolutional neural network (CNN), the electronic circuit comprising:

a crosspoint array; and

an output circuit comprising one or more integrators;

wherein performing the computations of the trained CNN comprises performing a method that comprises:

configuring the crosspoint array corresponding to a convolution layer in the CNN by storing one or more convolution kernels of the convolution layer in one or more crosspoint devices of the crosspoint array; and

performing computations for the CNN via the crosspoint array by:

transmitting voltage pulses corresponding to a vector of input data of the convolution layer to the crosspoint array;

outputting an electric current representative of performing a multiplication operation at a crosspoint device in the crosspoint array, the electric current based on a weight value stored by the crosspoint device and the voltage pulses from the input data; and

passing the output electric current from the crosspoint device to a selected integrator from the output circuit.

11. The electronic circuit of claim 10, wherein the integrator is a capacitor.

12. The electronic circuit of claim 10, wherein the output electric current is generated by the crosspoint device, the crosspoint device being at an intersection of a first row wire of the crosspoint array and a first column wire of the crosspoint array, and said integrator is physically proximate to a second column wire of the crosspoint array, yet is electrically coupled to said first column wire of the crosspoint array in order to receive said output electric current.

13. The electronic circuit of claim 10, wherein the output electric current is a first output electric current, the crosspoint device is a first crosspoint device that is at an intersection of a first row wire of the crosspoint array and a first column wire of the crosspoint array, the method further comprising:

outputting a second electric current by a second crosspoint device in the crosspoint array based on a weight value stored by the second crosspoint device and the voltage pulses from the input data, the second crosspoint device being at an intersection of a second row wire of the crosspoint array and a second column wire of the crosspoint array; and

passing the second output electric current from the crosspoint device to said selected integrator, where said integrator is physically proximate to said first column wire of the crosspoint array, yet is electrically coupled to said second column wire of the crosspoint array in order to receive said second electric current.

14. The electronic circuit of claim 10, wherein the selected integrator is selected by an output controller based on a mode signal that maps the output electric current from the crosspoint device to the selected integrator.

15. The electronic circuit of claim 10, wherein the crosspoint devices are arranged to implement one or more columns of the convolution kernels of a given layer of the CNN, and wherein the vector of input data represents neuron excitations to the given layer of the CNN presented from the input data, one row at a time.

16. The electronic circuit of claim 15, wherein a charge held by the selected integrator represents an output excitation according to the given layer of the CNN, the output excitation is converted and transmitted only after all rows of said convolution kernel are integrated.

17. The electronic circuit of claim 10, wherein the crosspoint devices are arranged to implement one or more rows of the convolution kernels of a given layer of the CNN, and wherein the vector of input data represents neuron excitations to the given layer of the CNN presented from the input data, one column at a time.

18. The electronic circuit of claim 17, wherein a charge held by the selected integrator represents an output excitation according to the given layer of the CNN, the output excitation is converted and transmitted only after all columns of said convolution kernel are integrated.

19. An electronic circuit comprising:

an array of resistive memory elements, the array providing a vector of current outputs equal to an analog vector-matrix-product between (i) a vector of voltage inputs to the array encoding a vector of analog input values and (ii) a matrix of analog resistive weights within the array;

accumulation wires and circuits aggregating a current from a dedicated subset of the resistive memory elements;

integration capacitors, each of the integration capacitors being electrically switchable so as to aggregate current from one of a plurality of accumulation wires during a single integration step;

data-output circuitry to allow an integrated charge from a subset of the integration capacitors, accumulated over a plurality of integration steps, to be suitably converted and transmitted either as an analog duration or as a digital representation using binary digits.

20. The electronic circuit of claim **19**, wherein the subset of the resistive memory elements corresponds to one or more column(s) of the array.

21. The electronic circuit of claim **19**, wherein the subset of the resistive memory elements corresponds to one or more row(s) of the array.

22. The electronic circuit of claim **19**, wherein the resistive memory elements are non-volatile memory devices.

23. The electronic circuit of claim **19**, wherein the resistive memory elements store synaptic weights of a neural network.

24. The electronic circuit of claim **19**,

wherein the resistive memory elements are arranged so as to implement columns of weight kernels of a given layer of a convolutional neural network;

wherein accumulation over a plurality of integration steps implements multiply-accumulate operations across multiple rows of said weight kernels, as the input neuron excitations to the said layer of the convolutional neural network are presented one row at a time; wherein the integrated charge representing an output excitation is suitably converted and transmitted only after all rows of said weight kernel are fully integrated; and

wherein the integrated charge on a plurality of capacitors representing a plurality of output excitations is suitably converted and a suitable pooled result such as the maximum, sum, or average of the said plurality of output excitations is computed locally and then transmitted, only after all relevant weight kernels are fully integrated.

25. The electronic circuit of claim **19**,

wherein the resistive memory elements are arranged so as to implement rows of weight kernels of one layer of a convolutional neural network;

wherein accumulation over a plurality of integration steps implement the integration across multiple columns of said weight kernels as the input data to said layer of the convolutional neural network are presented one column at a time;

wherein the integrated charge is suitably converted and transmitted only after all columns of said weight kernel are fully integrated; and

wherein the integrated charge on a plurality of capacitors representing a plurality of output excitations is suitably converted and a suitable pooled result such as the maximum, sum, or average of the said plurality of output excitations is computed locally and then transmitted, only after all relevant weight kernels are fully integrated.

* * * * *