Article

A Memristor-based Cascaded Neural Networks for Specific Target Recognition

Sheng-Yang Sun, Hui Xu, Jiwei Li, Yi Sun, Qingjiang Li, Zhiwei Li * and Haijun Liu *

College of Electronic Science, National University of Defense Technology, Changsha 410073, China; sunshengyang13@nudt.edu.cn (S.-Y. S.); xuhui@nudt.edu.cn (H. X.); lijiwei10@nudt.edu,cn (J. L.); sunyi12@nudt.edu.cn (Y. S.); liqingjiang@nudt.edu.cn (Q. L.)

* Correspondence: lizhiwei@nudt.edu.cn (Z. L.); liuhaijun@nudt.edu.cn (H. L.)

Abstract: Multiply-accumulate calculations using a memristor crossbar array is an important method to realize neuromorphic computing. However, the memristor array fabrication technology is still immature, and it is difficult to fabricate large-scale arrays with high-yield, which restricts the development of memristor-based neuromorphic computing technology. Therefore, cascading small-scale arrays to achieve the neuromorphic computational ability that can be achieved by large-scale arrays, which is of great significance for promoting the application of memristor-based neuromorphic computing. To address this issue, we present a memristor-based cascaded framework with some basic computation units, several neural network processing units can be cascaded by this means to improve the processing capability of the dataset. Besides, we introduce a split method to reduce pressure of input terminal. Compared with VGGNet and GoogLeNet, the proposed cascaded framework can achieve 93.54% Fashion-MNIST accuracy under the 4.15M parameters. Extensive experiments with Ti/AlOx/TaOx/Pt we fabricated are conducted to show that the circuit simulation results can still provide a high recognition accuracy, and the recognition accuracy loss after circuit simulation can be controlled at around 0.26%.

Keywords: cascaded neural networks; memristor crossbar; convolutional neural networks

1. Introduction

Convolutional neural networks (CNNs) have been widely used in computer vision tasks such as image classification [1][2][3], they are widely popular in industry for their superior accuracy on datasets. Some concepts about CNNs were proposed by Fukushima in 1980 [4]. In 1998, LeCun et al. proposed the LeNet-5 CNNs structure based on a gradient-based learning algorithm [5]. With the development of deep learning algorithms, the dimension and complexity of CNNs layers have grown significantly. However, state-of-the-art CNNs require too many parameters and compute at billions of FLOPs, which prevents them from being utilized in embedded applications. For instance, the AlexNet [1] proposed by Krizhevsky et al. in 2012, requires 60M parameters and 650K neurons; ResNet [6], which is broadly used in detection task [7], has a complexity of 7.8 GFLOPs and fails to achieve real-time applications even with a powerful GPU.

For the past few years, embedded neuromorphic processing systems have acquired significant advantages, such as the ability to solve the image classification problems while consuming very little area and power consumption [8]. In addition to these advantages, neuromorphic computing can also be used to simulate the functions of human brains [9]. With the rapid development of the VLSI industry, increasingly more devices are beginning to be miniaturized and integrated [10]. As mentioned above, CNNs lead to an exploding number of network synapses and neurons in which the relevant hardware costs will be huge [11], these complex computations and high memory requirements make it impractical for the application of embedded systems, robotics and real-time or mobile scenarios in general.

In recent years, the memristor [12] has received significant attention as synapses for neuromorphic systems. The memristor is a non-volatile device that can store the synaptic weights of neural networks.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 January 2019 doi:10.20944/preprints201901.0319.v1

Voltage pulses can be applied to memristors to change their conductivity and tune them to a specific resistance [13]. A memristor crossbar array can naturally carry out vector-matrix multiplication which is a computationally expensive operation for neural networks. It has been recently demonstrated that analog vector-matrix multiplication can be of orders of magnitude that are more efficient than ASIC, GPUs or FPGA based implementations [14].

Some neural network architectures based on the memristor crossbar have been proposed [15–17], these network architectures utilize memristor crossbar to perform convolution computation iterations which also cost too much time and require high memory. On the other hand, the fabrication technology of large-scale memristor crossbar array is still immature [18,19], however useful cascaded framework in real applications with memristor-based architectures have seldom been reported. To make full use of limited memristor resources and make the system work at high speed in real-time processing applications, we present a memristor-based cascaded framework with some neuromorphic processing chip. That means several neural network processing chips can be cascaded to improve the processing capability of the dataset. The basic computation unit in this work builds on our prior work devoloping memristor-based CNNs architecture [20], which has validated that the three-layer CNNs with Abs activation function can get desired recognition accuracy.

The rest of this paper is organized as follows, Section II presents the cascaded method based on the basic computation unit and a split method, including the circuits implemented based on memristor crossbar array. Section III exhibits the experimental results. The final Section IV concludes the paper.

2. Proposed Cascaded Method

To have a better understanding of cascaded method, we first introduce basic units that make up the cascaded network, and then a detailed description of our cascaded CNN framework is presented.

2.1. Basic Computation Unit

To take full advantage of limited memristor array, a three-layer simplified CNNs is used as the basic computation unit (BCU). The structure of this network is shown in Figure 1.

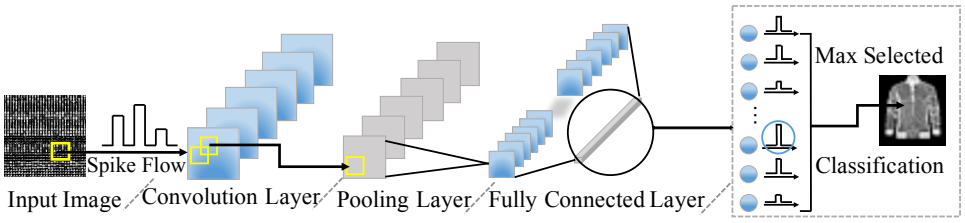

Figure 1. The basic computation unit architecture. It consists of three layers, behind the input layer is convolution layer which consists of k kernels, followed by a average-pooling layer and a fully connected layer.

The simplified CNNs includes three layers. The convolution layer includes k kernels with kernel size $K_s \times K_s$ followed by absolute nonlinearity function (Abs), which extracts the features from the input images and produce the feature maps. The average-pooling obtains spatial invariance while scaling feature maps with pooling size $P_s \times P_s$ from their preceding layers. A sub-sample is taken from each feature map, and it reduces the data redundancy. The fully connected layer (FCL) performs the final classification or image reconstruction, it takes the extracted feature maps and multiplies a weight matrix following a dense matrix vector multiplication pattern.

$$\begin{aligned} G = H(I^*) &= \sigma(W \cdot G^* + b) \\ &= \sigma(W \cdot (G^{**} * W^*) + b) \\ &= \sigma(W \cdot (\delta(W^{**} * I^* + b^*) * W^*) + b) \end{aligned} \tag{1}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 January 2019 doi:10.20944/preprints201901.0319.v1

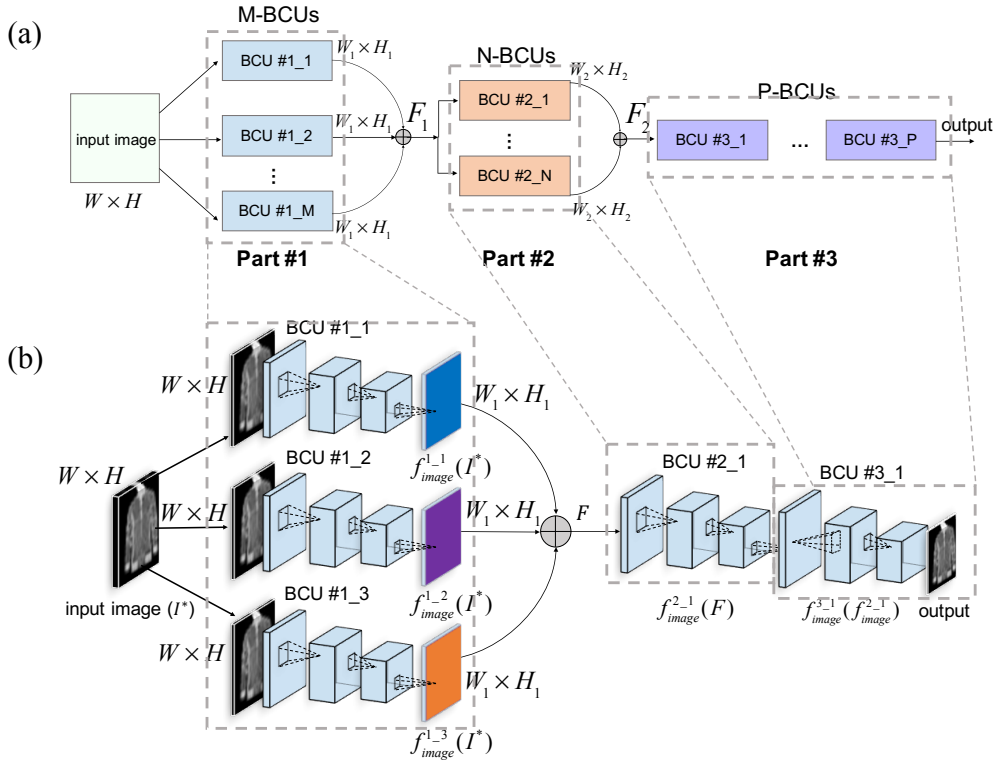

Figure 2. The diagram of the proposed cascaded framework. (**a**): Standard cascaded network framework "*M-N-P*". (**b**): The typical "3-1-1" cascaded type.

where $H : \mathbb{R}^{C\times W\times H} \rightarrow \mathbb{R}^{C^*\times W^*\times H^*}$ is the transformation in the BCU (in other words, BCU performs the image transformation), C is the number of channels of input image, σ indicates the hyperbolic tangent function, δ indicates the Abs function, $*$ represents the convolution operator, b, b^* are bias value, and W, W^*, W^{**} represent the weight matrix of each layer, respectively (refer to Figure 1).

2.2. The Proposed Cascaded Framework

Aim to combine several monolithic networks (BCUs) to obtain better performance, we propose a cascaded CNN network, whose specific design can be seen in Figure 2a.

The cascaded framework includes three parts. Given the output $G \in \mathbb{R}^{C\times W\times H}$ generated from a part, a reconstruction transformation $f : \mathbb{R}^{C\times W\times H} \rightarrow \mathbb{R}^{C\times W\times H}$ is applied to aggregate outputs over all BCUs of this part, where C is the number of channels of input image, W and H are the spatial dimensions. The output of k^{th} part is described as

$$F_k = G_{k\_1} \oplus G_{k\_2} \oplus ... \oplus G_{k\_n} \tag{2}$$

$$\begin{aligned} F_{k+1} &= G_{k+1\_1} \oplus G_{k+1\_2} \oplus ... \oplus G_{k+1\_m} \\ &= H_{k+1\_1}(F_k) \oplus ... \oplus H_{k+1\_m}(F_k) \end{aligned} \tag{3}$$

where G_{k_n} is the output of the n^{th} BCU of the k^{th} part, and $\oplus$ represents the reconstruction operator which combined with several original outputs using an element-wise addition to produce a new output. The new output F_k is treated as input to feed into the $k+1$ part. Equation (3) describes the calculation process of the next part output from the k^{th} part, here the $H(\cdot)$ is the transformation of BCU.

As shown in Figure 2a, the **Part #1** includes M BCUs, which extract features from the input image for the subsequent subnetworks. The output $F_1 \in \mathbb{R}^{C_1\times W_1\times H_1}$ is generated by reconstruction operation

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 January 2019 doi:10.20944/preprints201901.0319.v1

from the BCU outputs $G_{1\_n} \in \mathbb{R}^{C_1 \times W_1 \times H_1}$ ($n \in [1, M]$). The next **Part #2** includes N BCUs, which take the outputs of the previous part as the inputs to produce the $F_2 \in \mathbb{R}^{C_2 \times W_2 \times H_2}$. The final **Part #3** includes P BCUs, the first BCU takes the F_2 as the input to generate the $G_{3\_1} \in \mathbb{R}^{C_3 \times W_3 \times H_3}$, and the second BCU uses G_{3_1} to produce the G_{3_2}, and so on until $G_{3\_P} \in \mathbb{R}^{\widetilde{C}_3 \times \widetilde{W}_3 \times \widetilde{H}_3}$ is produced. This cascaded mode is called the "*M-N-P*" type.

The typical "3-1-1" cascaded framework is shown in Figure 2b, it includes five BCUs which three BCUs in parallel and two in series to produce the classification output.

2.3. Memristor-based Implementation

Based on BCU architecture, the computation unit can be treated as an image transformator. According to Equation (1), the output G can be described as a multiply-addition calculation so that it can be performed by several memristor.

As mentioned above, the BCU is a simplied CNN architecture. After the network training is finished, the weight matrix also has some negative weights. Therefore, the converted mapping method is needed to be applied. We assume that W represents a 2×2 weights matrix, and it includes positive and negative values. Then, the W is converted to the 2×4 matrix so that the memristor crossbar can easily calculate the weighted-sum with the amplifiers. Each original value is extended in two parts, W^+ and W^-. If one element is a positive value, then the value is defined in W^+, and the value of W^- is zero. In contrast, the negative element value is defined in W^- and the W^+ is zero. Similarly, if the arrangement of the memristor in the array corresponds to the converted matrix, the R_{off} takes the place of the zero element.

The BCU can produce entire output feature maps in one processing cycle. It no longer needs to wait for an output feature map to be completed before the next operation can be executed. This outcome means that it eliminates the requirement of a data storage device between each network layer. Figure 3a is a simple demo about performing a convolution computation in a memristor crossbar. A two-dimensional input image is converted into a one-dimensional electrical signal as an input, and the weighted-sum computation is completed by the memristor crossbar array.

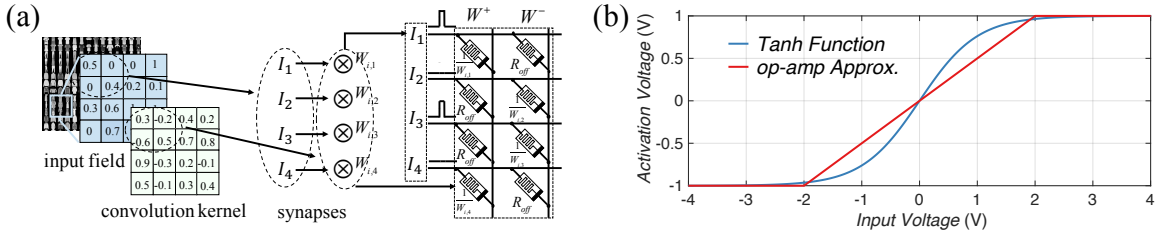

Figure 3. (**a**): Diagram displaying a memristor crossbar that is capable of completing a convolution computation. (**b**): Comparison of the tanh activation function and the amplifier alternative.

A memristor-based computation unit architecture is shown in Figure 4. A $K_s^2 \times P$ matrix $M_{(1)}(M_{ij})$ corresponds to all outputs of convolutional kernels (here $P = k \times 2$, two memristors represent one synapse), so the convolution layer consists of some kernels crossbar. The Abs activation module follows the kernels crossbar, it consists of two op-amps and two diodes. The activation module generates signals (like V_{O1} or V_{O2} in Figure 4) and sends these signals to the Average-Pooling modules. The pooling module consists some $P_s^2 \times 1$ crossbar and the matrix weight is $1/P_s^2$, the multiply-addition calculation is applied to complete the pooling operations. The fully connected layer receives the signals from pooling module, a $L \times N$ matrix $M_{(2)}(M'_{ij})$ corresponds to N neurons and L inputs. The Tanh activation circuit at the end of fully connected layer in Figure 4 is used to both scale the output voltage and implement the Tanh activation function. There is an alternative calculation to complete the Tanh activation function by referring the Equation (4). A linear amplifier activation function (with bounded upper and lower voltage rails) matches the Tanh relatively closely, and the simulation results show that this is an effective alternative (in Figure 3b). To obtain the optimal linear, $m = 1/2$ fits the Tanh

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 January 2019 doi:10.20944/preprints201901.0319.v1

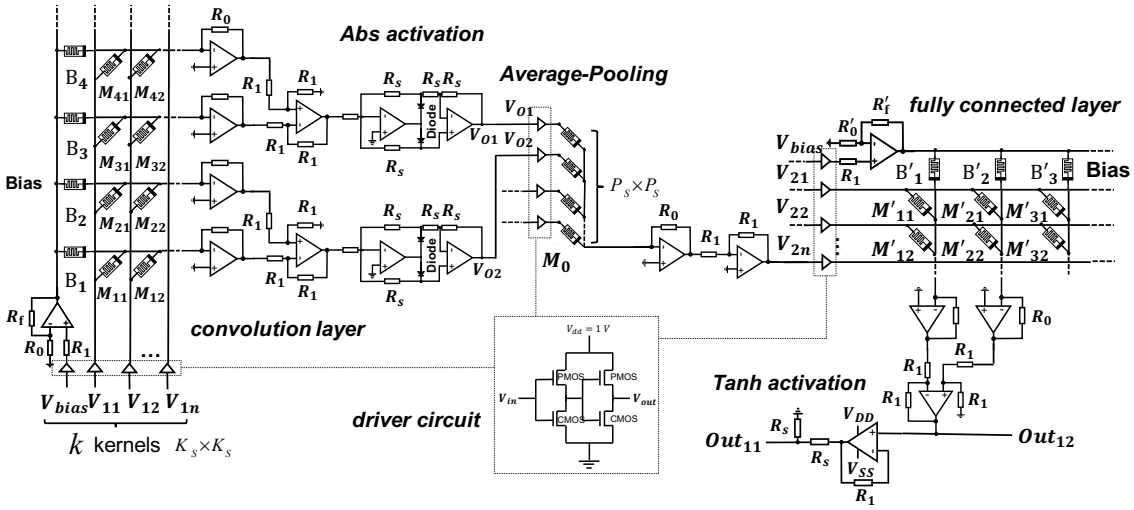

Figure 4. Memristor-based basic computaion unit architecture.

function in Equation (4). It should be noted that we use the Out_{11} to cascade the next network, the Out_{12} is treated as the output when we use the single BCU for classification.

$$Out_{11}(x) = \begin{cases} 1, & x > 2 \\ mx, & |x| \leqslant 2 \\ -1, & x < -2 \end{cases} \approx \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4}$$

To enhance the driving capability of crossbar arrays, driver circuits are needed so that low current communication signals can drive the large resistive crossbar structures, they are placed at each row input on the crossbar. These driver circuits have a finite impedance and errors will occur if the crossbar becomes too large [21]. Driver circuit is shown in Figure 4, and MOSFET based CMOS inverter circuits is used.

2.4. BCU Split Method

As we discussed in the previous section, the BCU can generate entire output feature maps in one processing cycle. It means that BCU is a highly parallel systems, it will improve the time efficiency of the classification. However, this calculation method will put pressure on the input terminal. Assumed the $W \times H$ input image, each of the images is converted to a $W \cdot H \times 1$ voltage output vector by DAC as the input dataflow. Each $K_s \times K_s$ field of the images will be sent to the K_i^{convX} module for the convolution computation.

K_i^{convX} indicates that when using the i^{th} convolution kernel to complete the X^{th} convolution computation, K_i^{convX} is the kernel crossbar discussed above. A $K_s \times K_s$ convolution computation of a $W \times H$ image requires a total of $(W - K_s + 1) \times (H - K_s + 1)$ operations. The modules of each convolution kernel are independent, so the K_i^{convX} does not need to be reconfigured.

In order to reduce the pressure on the input terminals, the BCU split method is need to be considered. According to Equation (1), the input image I^* will be sent to the BCU to complete the convolutional calculation. As we can see, the input I^* can be described as $I^* = I_1^* + I_2^* + ... + I_N^*$, so the Equation (1) can be rewritten as

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 January 2019 doi:10.20944/preprints201901.0319.v1

$$
\begin{aligned}
G &= \sigma(W \cdot (\delta(W^{**} * I^* + b^*) * W^*) + b) \\
&= \sigma(W \cdot (\delta(W^{**} * (I_1^* + ... + I_N^*) + b^*) * W^*) + b) \\
&= \sigma(W \cdot (\delta(W^{**} * I_1^* + b^*) * W^*) + b) \\
&+ \sigma(W \cdot (\delta(W^{**} * I_2^* + b^*) * W^*) + b) + ... \\
&+ \sigma(W \cdot (\delta(W^{**} * I_N^* + b^*) * W^*) + b) \\
&= H(I_1^*) \oplus H(I_2^*) \oplus ... \oplus H(I_N^*)
\end{aligned} \tag{5}
$$

where $H : \mathbb{R}^{C \times W \times H} \rightarrow \mathbb{R}^{C^* \times W^* \times H^*}$ is the transformation in the BCU, and $\oplus$ represents the reconstruction operator which combined with the output of different split chip using an element-wise addition, and N is the number of subimages.

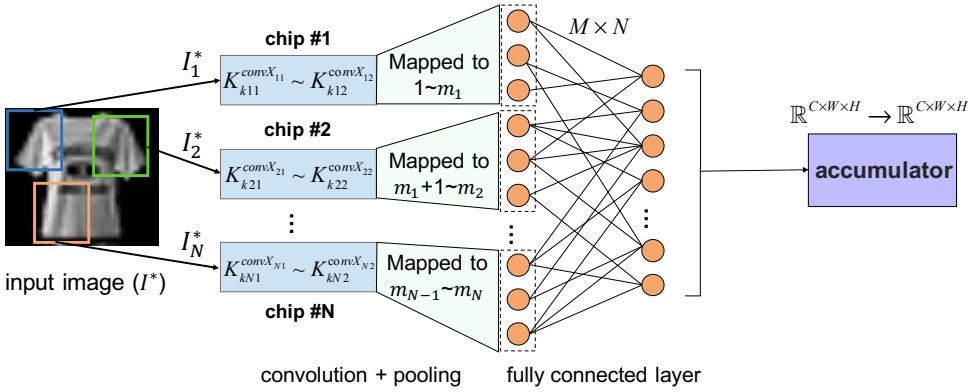

Figure 5. The diagram of BCU split method of one part.

Figure 5 demonstrates how a BCU in one part distributes input to multiple chips. Take the *chip #i* for example, this chip will perform the convolution calculation $K_{i11}^{convX_{i1}} \sim K_{i12}^{convX_{i2}}$ in the portion of I^*. As we mentioned above, the fully connected layer is a process that calculates $W \cdot G^*$ (here G^* can be regarded as $\mathbb{R}^{1 \times m_N}$), and the *chip #i* will map the output to the $m_{i-1} + 1 \sim m_i$ in G^*. Several $\mathbb{R}^{C \times W \times H}$ outputs are generated by these chips and accumulated by $\oplus$ operation.

3. Materials and Simulation Results

3.1. Simulation Settings

All experiments are conducted using an Intel Xeon E7 (2.6 GHz), 16GB DDR4 and a NVIDIA Titan XP graphics card. This work is implemented with the Theano python open-source library to train the neural network models for 100 epochs with a batch size of 128 and a learning rate of 0.05, and HSPICE is also used for some circuit simulations. To reflect the practical application of algorithm performance better, we do not use any data augmentation technologies. In the experiments, we chose the MNIST and Fashion-MNIST datasets for verification, and normalize the gray value of the image as the input voltage, whose range is 0-1 V.

We compare the performance of the cascaded network with some other state-of-the-art architectural units designed for classification on the MNIST and Fashion-MNIST dataset.

The MNIST dataset [5] consists of 60,000 training examples and 10,000 test examples of handwritten digits. Each input image is a 28×28 grey scale image with a corresponding label $l \in [0, 9]$, indicates different digits. The size and simplicity of the MNIST dataset make it convenient and quick to test models on.

Besides, the Fashion-MNIST [22] dataset is one choice for a more challenging classification task that is a drop-in replacement for the MNIST dataset. The Fashion-MNIST dataset consists of 60,000 training examples and 10,000 test examples of clothing articles, instead of digits. The input examples

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 January 2019 doi:10.20944/preprints201901.0319.v1

have the same format as the MNIST dataset and are 28×28 grey scale images with corresponding labels $l \in [0, 9]$. The Fashion-MNIST dataset is a step up from the MNIST dataset. It remains simple enough that complex architectures, learning algorithms, and models are not needed to view progress and soundly measure performance.

The weights of neural network are converted to conductivity values of memristor device by Equation (6). The C_{max} and C_{min} indicate the maximum and minimum conductances, respectively. The W is the original weights set, and the W_{max} represents the maximum absolute value of the weights set, and the C_i is the conductance of the memristor crossbar array.

$$C_i = \frac{C_{max} - C_{min}}{W_{max}} \cdot |W_i| + C_{min} \tag{6}$$

The 1T1R memristor crossbar array which stores the network weights is used for simulation. The weights obtained from training with the theano are programmed into a crossbar where each wire segment has a resistance of 2Ω, and 5% programmed erros were generated in memristor programming simulation process. The circuit structure in HSPICE is shown in Figure 4. The process of simulation implemented is illustrated as below.

1. Convert test image samples to voltage (0-1 V).
2. Execute software simulation (used Python) and circuit simulation (used HSPICE) based on step 1.
3. Obtain difference of the potential output (Difference = HSPICE potential - Python output) from Python and HSPICE.
4. Get the labels of test samples from dataset, and determine whether the results of circuit simulation are correct (the label that corresponds the maximum voltage is the output).

The Python (uses theano library) is applied to complete the software simulation, the model weights are converted to memristance, and the Python output is compared with the output of the HSPICE simulation. We select the number which has the maximum output voltage in the fully connected layer as the HSPICE output, the number is compared with the label of the dataset.

3.2. Memristor Materials

We then evaluated the cascaded neural network based on the Ti/AlOx/TaOx/Pt electronic synapse, which was proposed in our previous work[23]. The scanning electron microscopy (SEM) image and the cross-sectional transmission electron microscopy (TEM) is shown in Figure 6a.

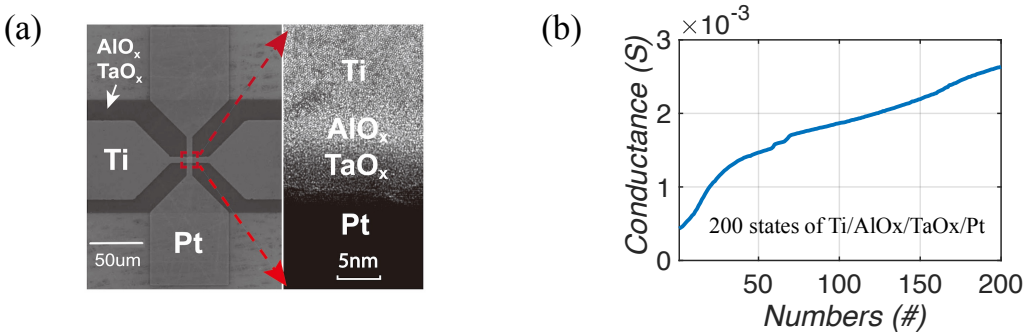

Figure 6. (**a**): SEM image and the Cross sectional TEM image of the Ti/AlOx/TaOx/Pt device. (**b**): 200 different states for our cascaded network simulation, achieved by applying triangular pulses.

During the fabrication process, the bottom electrode (Pt, 25nm) and the resistive switching layer (AlOx/TaOx, 5nm/3nm) were deposited by electron beam evaporation, respectively. And the top electrode (Ti, 30nm) was grown by magnetron sputtering. In all, three lithography processes were performed to form the patterns of the three different layers.

The Ti/AlOx/TaOx/Pt memristor proposed shows high uniformity, and over 200 states were achieved by applying triangular pulses (0→0.7 V, 28 KV/s for SET and 0→-1.2 V, 600 KV/s for RESET).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 January 2019 doi:10.20944/preprints201901.0319.v1

The device which has the multilevel characteristic can be repeatedly programmed to different target resistance states from 1KΩ to 12KΩ, indicating the great potential for neuromorphic computing applications.

Figure 6b demonstrates that 200 states achieved by applying triangular pulses are utilized for circuits simulations, the trained matrix weights are converted to conductivity values that fall within the bounded range of states. Equation (6) introduces the method of weight mapping. In the actual simulation, the closest value is selected from the 200 states.

3.3. Simulation Results

In this section, we compare the cascaded model with other state-of-the-art architectures to verify the classification performance. Circuit simulation experiments are used to verify the correctness of the circuit design and the impact of the device on the recognition accuracy.

3.3.1. Comparison With Other State-of-the-art Architectures

Table 1. Cascaded architecture for Fashion-MNIST under the 4.15 M parameters. The number before the "Cascaded" is the number of the BCUs. "×14" indicates the cascaded network uses 14 convolution kernels.

Stage	Output Size	**7-Cas.×14** (4-2-1)	**8-Cas.×14** (4-2-2)	**9-Cas.×14** (4-3-2)
	28 × 28	image		
Part #1	20 × 20	3×3, 5×5, 7×7, 9×9 conv. 2×2 average-pool		
Part #2	20 × 20	9×9, 9×9 conv, Abs 2×2 average-pool	9×9, 9×9 conv. 2×2 average-pool	3×3, 5×5, 7×7 conv. 2×2 average-pool
Part #3	6 × 6	9×9 conv. 2×2 average-pool	9×9, 9×9 conv. 2×2 average-pool	9×9, 9×9 conv. 2×2 average-pool
Classifier	1 × 1	Fully Connected Layer + softmax		
Parameters		3.45 M	3.46 M	4.15 M
Accuracy (%)		93.12	93.25	93.54

Based on basic computation unit, we test the cascaded network with different architecture on MNIST and Fashion-MNIST datasets, the configuration details and performance is shown in Table 1. The BCU which has 14 kernels with 9×9 kernel size and 2×2 pooling size, and Abs activation function applied can achieve 89.68% Fashion-MNIST accuracy, while Cascaded×14 achieves improvements of 3.18%, 3.44%, 3.57% and 3.86% under 4.15M parameters. The output sizes of the Part#1 and Part#2 are both 20×20. 7-Cascaded×14(4-1-1) provides 93.12% Fashion-MNIST accuracy which uses 9×9 kernel. 8-Cascaded×14(4-2-2) slightly outperforms "4-2-1" (∼0.13%) with 10,000 more parameters. It can be seen that cascaded models can achieve 93.54% accuracy under the 4.15M parameters.

Table 2. Accuracy rates (%) on the MNIST and Fashion-MNIST datasets.

Models	Parameters	MNIST Acc.	Fashion-MNIST Acc.
VGGNet-16 [2]	26 M	99.22	93.35
GoogLeNet [3]	6.8 M	99.45	93.52
ResNet-50 [6]	25.5 M	99.55	94.24
CapsuleNet [24]	11.3 M	99.35	93.55
This work	**4.15 M (<500K per single BCU)**	**99.55**	**93.54**

We further compare the cascaded architecture with the building structures of four state-of-the-art models, VGGNet[2], GoogLeNet[3], ResNet[6] and CapsuleNet[24]. The BCU architecture in Table 2 includes 14 kernels, 2×2 average-pool with Abs activation function. A "4-3-2" cascaded type is implemented, "3×3, 5×5, 7×7, 9×9 conv." is in Part #1, "3×3, 5×5, 7×7 conv." is in Part #2, and "9×9, 9×9 conv." in Part #3. VGGNet-16 provides 99.22% MNIST accuracy and 93.35% Fashion-MNIST

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 January 2019 doi:10.20944/preprints201901.0319.v1

accuracy, GoogLeNet achieves 99.45% MNIST accuracy and 93.52% Fashion-MNIST accuracy with 6.8M parameters, CapsuleNet provides 99.35% and 93.55% accuracy on MNIST and Fashion-MNIST datasets respectively, while ResNet-50 produces remarkably better Fashion-MNIST accuracy of 94.24%. However, they are all computationally intensive (approximately 26M parameters of VGGNet). In comparison, 9-Cascaded×14(4-3-2) achieves 99.55% MNIST accuracy and 93.54% Fashion-MNIST accuracy under 4.15M parameters. From the table it can be seen that the cascaded network significantly surpasses VGGNet-16 by 0.33% on MNIST accuracy with 6.26× fewer parameters, and slightly outperforms GoogLeNet 0.02% Fashion-MNIST accuracy with 1.64× fewer parameters.

3.3.2. Circuits Simulation Peformance

Plots in Figure 7 show the potential output of BCU obtained by the HSPICE simulation and the software simulation output (used Python) in 10,000 test samples.

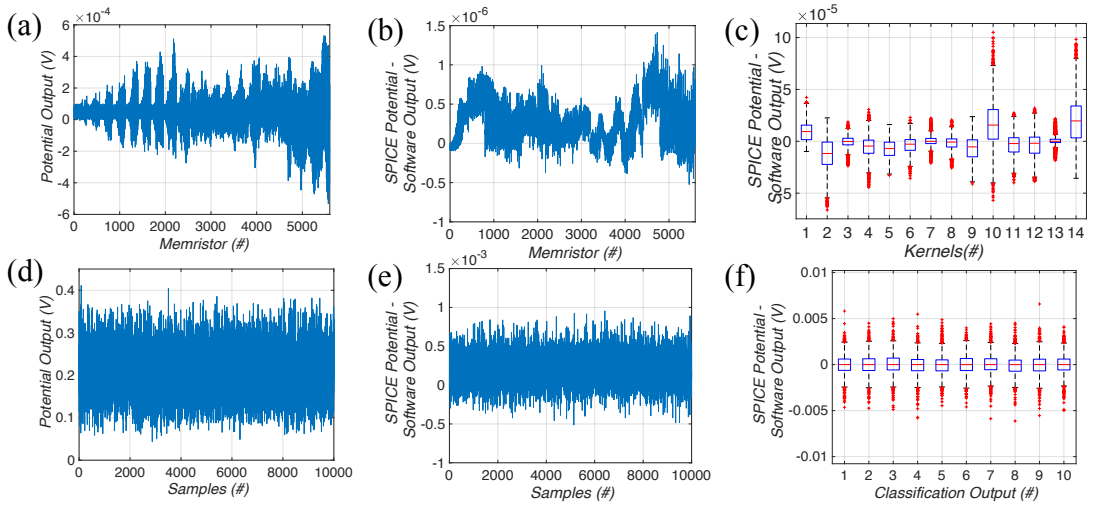

Figure 7. (**a**) and (**d**): Potential output of each kernel and fully connected layer. (**b**) and (**e**): Difference of the potential output of each kernel and fully connected layer. (**c**) and (**f**): Boxplots of simulation difference for the output of kernel and fully connected layers.

Figure 7a demonstrates all kernels potential output of one test sample, which includes 400 convolution operations applied by 14 kernels. Similarly, Figure 7d shows the potential output of 10,000 samples, each potential is the maximum voltage of 10 outputs in fully connected layer. It can be seen that the outputs of each samples are between 0.05 V to 0.45 V. Figure 7b and 7e depict the difference of the corresponding potential outputs in the crossbars, obtained from HSPICE and the Python software based simulations. From Figure 7e we can observe that the corresponding output result from HSPICE and Python simulations are close (absolute values of the difference of the corresponding values are less than 1×10^{-3}). Figure 7c and 7f depict the boxplot of simulation difference for the output of kernels and fully connected layers. As shown in Figure 7c, the differences are concentrated in the range of -5×10^{-6} to 10×10^{-6}. In Figure 7f, the differences are concentrated in the range of -5×10^{-3} to 5×10^{-3}. This is two orders of magnitude different for the kernels and fully connected layer output. Simulation experiments demonstrate that the output of the circuit is close to the software simulation, indicating that the circuit implementation is feasible.

Figure 8 illustrates the relationship between the amount of model parameters and Fashion-MNIST recognition accuracy. The amount of convolution kernels in the BCU is between 5 to 32, 9×9 kernel size and 2×2 average pooling are applied. As shown in figure, the BCU can achieve 89.60% accuracy under 0.5M parameters (when the amount of convolution kernels is 32). We tested the five cascaded type for their performance. It can be observed that as the number of parameters increases, the recognition accuracy of the network gradually increases. The “4-3-2" cascaded type provides 93.12% accuracy with 4.15M parameters. It is worth noting that the accuracy of the BCU is better than“1-1-1" cascaded type

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 January 2019 doi:10.20944/preprints201901.0319.v1

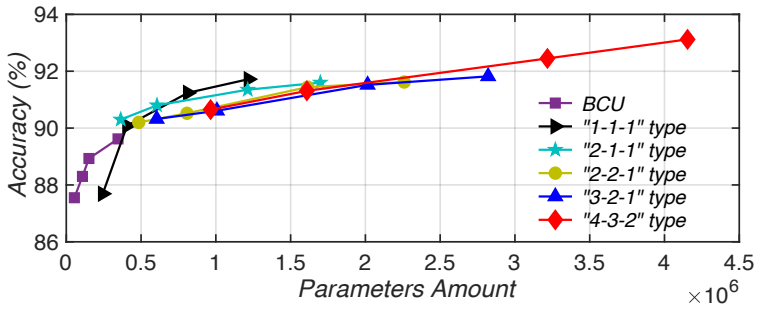

Figure 8. Comparison of different cascaded architecture with one BCU on Fashion-MNIST dataset based on HSPICE simulation.

under 0.5M parameters, the feature information may not be effectively transmitted due to the lack of the cooperation of the parallel structure. The specific reason remains to be studied in the future work.

4. Conclusions

In this paper, we propose a memristor-based cascaded framework, which can use several basic computation units in cascaded mode to improve the processing capability of the dataset. We demonstrate the architecture of the BCU and corresponding circuit structure. Based on the proposed BCU architecture, we present a “M-N-P" cascaded specially designed for improving recognition accuracy of the datasets, and we introduce a split method about BCU to reduce pressure of input terminal. Compared with VGGNet and GoogLeNet, the proposed cascaded framework can achieve desired accuracy with fewer parameters. Extensive circuits experiments are conducted show that the circuit simulation results can still provide a high recognition accuracy. For future work, we consider to further evaluate the cascaded framework on other tasks such as object detection.

Author Contributions: Conceptualization, S.-Y. S. and H. X.; Methodology, H. X.; Formal analysis, J. L. and H. L.; Materials fabricated, Y. S.; Investigation, Z. L. and J. L.; Software, H. L. and S.-Y. S.; Project administration, Q. L. and H. X.; Funding acquisition, Q. L. and H. L.; Writing–original draft, S.-Y. S.; Writing–review and editing, H. L., Q. L. and H. X.

Funding: This work is supported by National Natural Science Foundation of China under grant No.61471377, No.61604177 and No.61704191.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2012, pp. 1097–1105.
2. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* **2014**, *abs/1409.1556*, [1409.1556].
3. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.
4. Fukushima, K.; Miyake, S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*; Springer, 1982; pp. 267–285.
5. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **1998**, *86*, 2278–2324.
6. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
7. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **2017**, pp. 1137–1149.

8. Belhadj, B.; Joubert, A.; Li, Z.; Héliot, R.; Temam, O. Continuous real-world inputs can open up alternative accelerator designs. ACM SIGARCH Computer Architecture News. ACM, 2013, Vol. 41, pp. 1–12.
9. Park, S.; Noh, J.; Choo, M.l.; Sheri, A.M.; Chang, M.; Kim, Y.B.; Kim, C.J.; Jeon, M.; Lee, B.G.; Lee, B.H.; others. Nanoscale RRAM-based synaptic electronics: toward a neuromorphic computing device. *nanotechnology* **2013**, *24*, 384009.
10. Taur, Y.; Ning, T.H. *Fundamentals of modern VLSI devices*; Cambridge university press, 2013.
11. Kim, B.; Lee, M.; Kim, J.; Kim, J.; Lee, J. Hierarchical Compression of Deep Convolutional Neural Networks on Large Scale Visual Recognition for Mobile Applications **2016**.
12. Chua, L. Memristor-the missing circuit element. *IEEE Transactions on circuit theory* **1971**, *18*, 507–519.
13. Yakopcic, C.; Alom, M.Z.; Taha, T.M. Extremely Parallel Memristor Crossbar Architecture for Convolutional Neural Network Implementation. International Joint Conference on Neural Networks, 2017.
14. Hu, M.; Strachan, J.P.; Li, Z.; Grafals, E.M.; Davila, N.; Graves, C.; Lam, S.; Ge, N.; Yang, J.J.; Williams, R.S. Dot-product engine for neuromorphic computing: programming 1T1M crossbar to accelerate matrix-vector multiplication. Proceedings of the 53rd annual design automation conference. ACM, 2016, p. 19.
15. Shafiee, A.; Nag, A.; Muralimanohar, N.; Balasubramonian, R.; Strachan, J.P.; Hu, M.; Williams, R.S.; Srikumar, V. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News* **2016**, *44*, 14–26.
16. Chi, P.; Li, S.; Xu, C.; Zhang, T.; Zhao, J.; Liu, Y.; Wang, Y.; Xie, Y. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. ACM SIGARCH Computer Architecture News. IEEE Press, 2016, Vol. 44, pp. 27–39.
17. Li, B.; Wang, Y.; Wang, Y.; Chen, Y.; Yang, H. Training itself: Mixed-signal training acceleration for memristor-based neural network. Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific. IEEE, 2014, pp. 361–366.
18. Kim, S.; Ishii, M.; Lewis, S.; Perri, T.; BrightSky, M.; Kim, W.; Jordan, R.; Burr, G.; Sosa, N.; Ray, A.; others. NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous in-situ learning. Electron Devices Meeting (IEDM), 2015 IEEE International. IEEE, 2015, pp. 17–1.
19. Yu, S.; Chen, P.Y.; Cao, Y.; Xia, L.; Wang, Y.; Wu, H. Scaling-up resistive synaptic arrays for neuro-inspired architecture: Challenges and prospect. Electron Devices Meeting (IEDM), 2015 IEEE International. IEEE, 2015, pp. 17–3.
20. Sun, S.; Li, J.; Li, Z.; Liu, H.; Li, Q.; Xu, H. Low-Consumption Neuromorphic Memristor Architecture Based on Convolutional Neural Networks. 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–6. doi:10.1109/IJCNN.2018.8489441.
21. Uppala, R.; Yakopcic, C.; Taha, T.M. Methods for reducing memristor crossbar simulation time. 2015 National Aerospace and Electronics Conference (NAECON), 2015, pp. 312–319. doi:10.1109/NAECON.2015.7443089.
22. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* **2017**.
23. Sun, Y.; Xu, H.; Wang, C.; Song, B.; Liu, H.; Liu, Q.; Liu, S.; Li, Q. A Ti/AlO x/TaO x/Pt analog synapse for memristive neural network. *IEEE Electron Device Letters* **2018**, *39*, 1298–1301.
24. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic routing between capsules. Advances in Neural Information Processing Systems, 2017, pp. 3856–3866.