

Pulse coding off-chip learning algorithm for memristive artificial neural network

Ming-Jian Guo(郭明健), Shu-Kai Duan(段书凯), and Li-Dan Wang(王丽丹)†

College of

Citation: Chin. Phys. B, 2022, 31 (7): 078702 DOI: 10.1088/1674-1056/ /ac4f4e

Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Relationship between manifold smoothness and adversarial vulnerability in deep learning with local errors

Zijian Jiang(蒋子健), Jianwen Zhou(周健文), and Haiping Huang(黄海平)

Chin. Phys. B, 2021, 30 (**4**): 048702. DOI: 10.1088/1674-1056/abd68e

Entrainment range affected by the difference in sensitivity to light-information between two groups of SCN neurons

Bao Zhu(朱宝), Jian Zhou(周建), Mengting Jia(贾梦婷), Huijie Yang(杨会杰), Changgui Gu(顾长贵)

Chin. Phys. B, 2020, 29 (**6**): 068702. DOI: 10.1088/1674-1056/ab8897

Effects of refractory period on dynamical range inexcitable networks

Ya-Qin Dong(董亚琴), Fan Wang(王帆), Sheng-Jun Wang(王圣军), Zi-Gang Huang(黄子罡)

Chin. Phys. B, 2019, 28 (**12**): 128701. DOI: 10.1088/1674-1056/ab4f60

Entrainment range affected by the heterogeneity in the amplitude relaxation rate of suprachiasmatic nucleus neurons

Chang-Gui Gu(顾长贵), Ping Wang(王萍), Hui-Jie Yang(杨会杰)

Chin. Phys. B, 2019, 28 (**1**): 018701. DOI: 10.1088/1674-1056/28/1/018701

Analytically determining frequency and amplitude of spontaneous alpha oscillation in Jansen's neural mass model using the describing function method

Yao Xu(徐瑶), Chun-Hui Zhang(张春会), Ernst Niebur, Jun-Song Wang(王俊松)

Chin. Phys. B, 2018, 27 (**4**): 048701. DOI: 10.1088/1674-1056/27/4/048701

Pulse coding off-chip learning algorithm for memristive artificial neural network

Ming-Jian Guo(郭明健), Shu-Kai Duan(段书凯), and Li-Dan Wang(王丽丹)†

College of Artificial Intelligence, Southwest University, Chongqing 400715, *China*

(Received 8 December 2021; revised manuscript received 18 January 2022; accepted manuscript online 27 January 2022)

Memristive neural network has attracted tremendous attention since the memristor array can perform parallel multiply-accumulate calculation (MAC) operations and memory-computation operations as compared with digital CMOS hardware systems. However, owing to the variability of the memristor, the implementation of high-precision neural network in memristive computation units is still difficult. Existing learning algorithms for memristive artificial neural network (ANN) is unable to achieve the performance comparable to high-precision by using CMOS-based system. Here, we propose an algorithm based on off-chip learning for memristive ANN in low precision. Training the ANN in the high-precision in digital CPUs and then quantifying the weight of the network to low precision, the quantified weights are mapped to the memristor arrays based on VTEAM model through using the pulse coding weight-mapping rule. In this work, we execute the inference of trained 5-layers convolution neural network on the memristor arrays and achieve an accuracy close to the inference in the case of high precision (64-bit). Compared with other algorithms-based off-chip learning, the algorithm proposed in the present study can easily implement the mapping process and less influence of the device variability. Our result provides an effective approach to implementing the ANN on the memristive hardware platform.

Keywords: off-chip learning, mapping, memristor array, artificial neural network

PACS: 87.18.Sn, 07.05.Rm, 07.05.Tp, 07.05.Mh **DOI:** 10.1088/1674-1056/ac4f4e

1. Introduction

In recent years, artificial neural networks (ANN) have achieved explosive development in many different fields, such as object detection,[1] natural language processing,[2] autonomous driving, and security. Forward propagation network is a vital network structure of ANN, whose information only goes forward. Generally, the forward propagation networks can be divided into two main categories according to network structures. One is the convolution neural network,[3] whose inputs are weighted by the convolution operations, and the other is the perceptron network. The calculations of both networks are based on the principle of parallel multiply-accumulate calculation (MAC) operations.[4] At the hardware level, the architecture that supports the MAC operation such as graphics processing units (GPUs)[5] and accelerators for specific applications[6,7] enables the implementing of deep neural networks such as CNN with lower power consumption, and the higher computational efficiency makes the practical application of ANN further expand to areas. However, the memory unit is separated from the computing unit in the von-Neumann architecture which greatly limits data communication between computing unit and memory unit, thus resulting in the von-Neumann bottleneck.[8] With the demand for data processing in the era of big data increasing, it is urgent to find an alternative way to the von-Neumann computing paradigm. In contrast, neuromorphic computing supported by the memristor provides a promising non-von Neumann architecture, in which memory computing eliminates the cost of data transfer.[9–11]

The memristor is a double-ended 'storage resistor' of which the weight is stored in the memristor in the form of conductance, and the weight can be updated by controlling the conductance.[12–15] Through tuning the resistance of the memristors, different patterns can be stored into the memristive Hopfield neural network.[16] Memristors are similar to biological synapses because of variable resistance and non-volatility and are used to design brain-like spike neurons model.[17] The parallel MAC operation can be realized on the memristor array through Ohm's law and Kirchhoff's law, which can be employed in feed-forward fully connected neural networks.[4,18–22] As far as the parallel MAC operation is concerned, its simple calculation principle and high performance in computation efficiency and energy consumption have received great attention.[23–25] Recent studies show that the neural networks implemented on memristive neuromorphic systems could achieve huge power savings.[26] Even the computing on memristor array shows excellent performance in terms of computation and power dissipation, it is a significant challenge to train a high precision neural network memristive computation units which are compared with digital CPUs or FPGA.[27]

There are two main programming methods to use memristor array in artificial neural networks, *i.e.* on-chip learning[28,29] and off-chip learning.[30,31] The former is to divide the conductance of memristor into multiple levels as a low-precision digital device, such as 4 bits, this means the

†Corresponding author. E-mail: ldwang@swu.edu.cn

© 2022 Chinese Physical Society and IOP Publishing Ltd

conductance of memristor is divided into 16 levels. The memristor conductance is updated by a back-propagation algorithm which operates in the memristive system. Although the method allows the device to possess the potential to have any conductance states so that it could adapt itself to the device variability,[32] the variation of conductance state in memristor array is limited in low precision. For example, if the number of levels of conductance in the memristor are designed to arrive at 256 , this means that the precision of the memristor is 8 bits, it will be a big challenge to an integrated memristor array.[33] It causes the neural network during on-chip learning performance to fluctuate greatly. The loss of the neural network is more difficult to converge. In addition, changing the conductance frequently reduce the durability of the memristor.[34] The latter, *i.e.*, off-chip learning, is to use the high-precision computation units such as digital CPUs or GPUs to train neural network,[29] then quantify the trained weights to low precision such as 4 bits, *etc*. Next, mapping the quantified weights to the memristor array in the form of conductance, in other words, each device in the memristor array is written as the target conductance state by the write–verify method, which verifies the conductance by using a pulse with small amplitude and reading out the output current.[18] Compared with on-chip learning, the off-chip learning greatly reduces the time of changing the conductance, which improves the durability of the memristor device. The neural network trained under the condition of the high precision in digital CPUs could converge to better performance, The weight is trained in digital CPUs, then programmed in memristor array. However, the write–verify method needs a large number of write and erase operations, which reduces the efficiency of weight update. The variations make it hard to write the memristor to correct conductance states because of the device gap.[33]

Because of the above reasons, we propose a new off-chip learning algorithm for the memristive artificial neural network. The algorithm does not need to extra change the conductance state except for the mapping process, which can mitigate the effect of some device characteristics, such as the device variation and asymmetry.[32] We design a 5-layer convolution neural network on MNIST data set to verify the validity of the algorithm. First of all, we train the convolution neural network under the precision of 64 bits, then quantify the trained weight to low precision by the difference quantification method. Next, we encode the weight into voltage pulses and then apply them to memristors. The flowchat is shown in Fig. 1. In addition, we discuss the effect of different precision of quantified weight in the quantification process and different conductance intervals in the mapping process on neural networks. Finally, we use the 4-bit quantified weight to map the memristor array, obtaining 97.36% classification accuracy comparable to 98.23% accuracy in 64 bits.

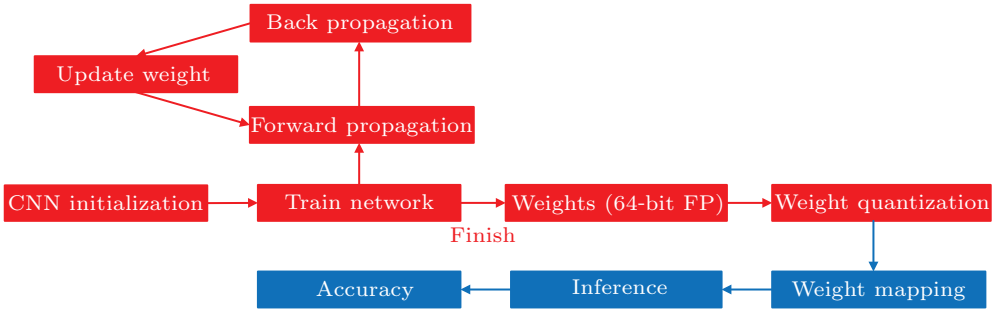


Fig. 1. Flowchart of memristive off-chip learning algorithm, where red part is implemented in high-precision computation unit, and blue part in memristive computation unit.

2. Methods

2.1. Memristor model and memristor array

The memristor is a new type of device used in various fields including storage, logic circuits, and neuromorphic computing.[10] The memristor model is established to simulate memristor phenomena in various memristive applications. The VTEAM model,[35] based on the expression of internal state variables, is widely used in the simulation of memristive applications due to its voltage threshold characteristics. Generally, the expressions of VTEAM during voltage control are given by

$$\frac{\mathrm{d}w}{\mathrm{d}t} = f(w,v), \tag{1}$$

$$i(t) = G(w,v) \times v(t), \tag{2}$$

where w is the internal state variable. $v(t)$ is the expression of the voltage applied across the memristor, $i(t)$ is the expression of the current flowing through the memristor under voltage control, $G(w,v)$ is the conductance of the memristor, and $f(w,v)$ is a function of the internal state variables of the memristor, and its expression can be written as follows:

$$\frac{\mathrm{d}w(t)}{\mathrm{d}t} = \begin{cases} k_{\mathrm{off}} \times \left(\frac{v(t)}{v_{\mathrm{off}}} - 1\right)^{a_{\mathrm{off}}} \times f_{\mathrm{off}}(w), & 0 < v_{\mathrm{off}} < v, \\ 0, & v_{\mathrm{on}} < v < v_{\mathrm{off}}, \\ k_{\mathrm{on}} \times \left(\frac{v(t)}{v_{\mathrm{on}}} - 1\right)^{a_{\mathrm{on}}} \times f_{\mathrm{on}}(w), & v < v_{\mathrm{on}} < 0, \end{cases} \tag{3}$$

where k_{off}, k_{on}, a_{off}, and a_{on} are constants; k_{off}, k_{on}, and a_{off} are positive numbers; k_{on} is a negative number; v_{on} and v_{off} repre-

sent the threshold of the positive voltage and negative voltage; $f_{\rm off}(w)$ and $f_{\rm on}(w)$ are window functions in which the state variable w has a different symbol. Window function contains state variable w in a fixed bound of$w \in [w_{\rm on}, w_{\rm off}]$, which is expressed as Eq. (4). Different window functions can be used in the VTEAM model. The ideal window function is 1 when $w \in (w_{\rm on}, w_{\rm off})$, otherwise it is 0.

$$f(x) = \begin{cases} 1, & w_{\rm on} \le w \le w_{\rm off}, \\ 0, & \text{else}. \end{cases} \tag{4}$$

The relationship between voltage and current $G(v,t)$ has no inherent definition.[33] Here, the original voltage-current relationship of VTEAM is modified, and the relationship between conductance and state variables is obtained. The relationship between current and voltage is

$$i(t) = \left[R_{\rm on} + \frac{R_{\rm off} - R_{\rm on}}{w_{\rm off} - w_{\rm on}} \cdot (w - w_{\rm on}) \right], \tag{5}$$

where $w_{\rm on}$ and $w_{\rm off}$ are the bounds of internal state variable w. $R_{\rm on}$ and $R_{\rm off}$ are both constants, which are the corresponding resistances of the device. Here, we set the parameters as follows: $w_{\rm on} = 0$, $w_{\rm off} = 10$, $R_{\rm on} = 100\ \Omega$, $R_{\rm off} = 2500\ \Omega$, $k_{\rm off} = 4.03 \times 10^{-4}$, $k_{\rm on} = -80$, $a_{\rm off} = 1$, and $a_{\rm on} = 3$. When the amplitude of the pulse is greater than 0.5 V, the conductance of the memristor changes, otherwise, the conductance of the memristor remain unchanged . When applying a series of pulses to the memristor, its conductance can be changed, which makes the memristor crucial for the memristive neural networks as synapses. The pulses with the same number but different amplitudes cause the conductance of VTEAM memristor model to change into different ranges, just as shown in Fig. 2. Here, we select four different amplitudes of the pulse, which are 0.8 V, 1 V, 1.5 V, 2 V. Figure 2(b) shows that under the condition of the same pulse number, the range of conductance changed is positively correlated with the amplitude of the pulse.

Figure 3 shows the cross-point structure that is built by the VTEAM model in this work, which is termed memristor array. The memristor array can realize the weighted combination and matrix-vector multiplication following simple physics laws such as Ohm law and Kirchhoff's law. The the input voltage and output currens can be related by the following equation:

$$I_{\rm n} = \sum_{m=1}^{M} V_m \cdot (G_{m,n} + G_w(i,j)), \tag{6}$$

where $V_{\rm m}$ is the input voltage on the j-th row, $I_{\rm n}$ is the output current in the n-th column, and $G_{m,n}$ is the conductance of the memristor, whose position is the m-th row and the n-th column in memristor array, $m = 1,2,\ldots M$, $n = 1,2,\ldots N$, where M is the number of memristor array rows and N is the number of memristor array columns, $G_{\rm w}(i,j)$ is the wire conductance along the per conductive path. The resistance of wire is much smaller than that of the memristor, and it has less effect on the result of computation of the memristor array. Therefore, $G_{\rm w}(i,j)$ is set to be zero in the simulation. In the memristor, the X_m is represented as V_m directly, and $A(m,n) = G_{m,n}$ without considering the wire resistance. The current output can regard as y, which is the product of output matrix-vector, *i.e.* $y = Ax$.

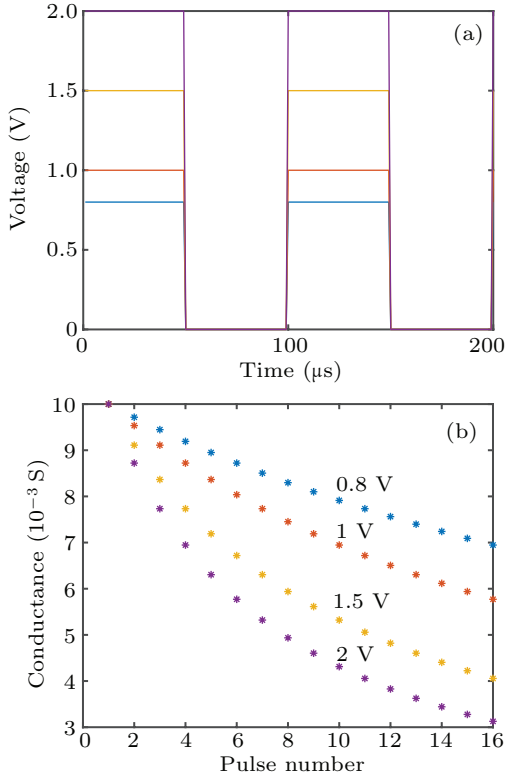


Fig. 2. Memristor model showing (a) curves of voltage pulse *versus* time with pulse amplitudes of 0.8 V, 1 V, 1.5 V, 2 V and width 50 μs, and (b) curves of conductance *versus* pulse number for different amplitudes of the pulses, each curve with 16 pulses of memristor model.

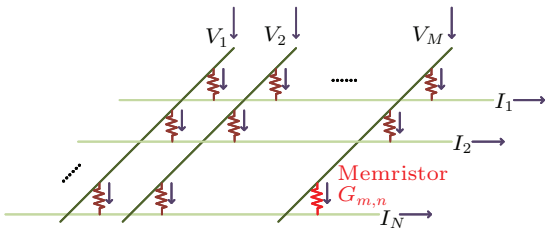


Fig. 3. Illustration of memristor array and operation process.

2.2. Weight quantification

The precision of weight trained in digital CPUs is 64 bits, which means that the weight will have 2^{64} values. However, it is very difficult to modulate 2^{64} different conductance states steadily. Therefore, the trained weight needs to be quantified into low precision such as 4 bits or others prior to the mapping process. Here, we use two quantification methods, named uniform quantification and Gaussian quantification, and dis-

cuss the influences of two quantification methods on the performance of the memristive neural network. For uniform quantification, the trained weights are assigned to the nearest simulation weights, and the quantified simulation weights are uniform distributed. Given the weight w in the interval (W_n, W_{n+1}), we define the weight w as follows:

$$W_{\mathrm{q}} = \begin{cases} W_n, & (w - W_n) < (W_{n+1} - w), \\ W_{n+1}, & (w - W_n) > (W_{n+1} - w), \end{cases} \tag{7}$$

where W_{q} is the quantified weight, W_n and W_{n+1} are the simulated weights in the interval $(-1, 1)$, and w is the trained weight.

When the weight is quantified, the Gaussian quantification is added to Gaussian noise to make weights closer to Gaussian distribution. The purpose of adding noise is to concentrate the quantified simulation weights around the average value. Our definition of Gaussian quantification is as follows:

$$E = \frac{1}{n}\sum_{i=1}^{n} w_i, \tag{8}$$

$$D = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(w_i - E)^2}, \tag{9}$$

$$N_i = \frac{1}{\sqrt{2\pi}} \exp\left[\frac{(w_i - E)^2}{2 \times D^2}\right], \tag{10}$$

where w_i represents the weight, n is the number of weights, and E and D are the mean and standard deviations of all weights, and N_i is the Gaussian parameter weight w_i.

The process expression of Gaussian quantification is as follows:

$$W^i_{\mathrm{gauss}} = N_i * W_i, \tag{11}$$

where W is the analog weight quantified into n bits in the range of $(-1, 1)$, the value of W is 2^{n-1}, W^i_{gauss} is the Gaussian simulation weight that is added to Gaussian noise. The weight is quantified according to the following formula:

$$W^i_{\mathrm{sim}} = \begin{cases} W^n_{\mathrm{gauss}}, & (w_i - W^n_{\mathrm{gauss}}) < (W^{n+1}_{\mathrm{gauss}} - w_i), \\ W^{n+1}_{\mathrm{gauss}}, & (w_i - W^n_{\mathrm{gauss}}) > (W^{n+1}_{\mathrm{gauss}} - w_i), \end{cases} \tag{12}$$

where W^i_{sim} is the quantification weight of w_i. Equation (11) means that the trained weight is assigned to the nearest simulation weight W_{gauss}.

2.3. Weight mapping

The weight is in the form of conductance of memristor when executing the inference of the memristive neural network. Mapping is an inevitable step for transferring the trained weight from the digital computing unit to the memristive computing unit. In the precious off-chip learning algorithm, performing multiple write–verify methods to make the memristor reach a target conductance state. However, the device variation makes it hard to write the memristor into the correct conductance. Here, we propose a new mapping method that adopts the specific pulses according to the position of the weight in the distribution. Using a pair of adjacent memristors to represent a signed weight, one represents positive weight, the other represents negative weight, the total weight is obtained by the difference between the positive weight and the negative weight. In addition, the trained weight is distributed in the interval $(-1, 1)$. In the quantification process, we set $2^{(n-1)}$ simulation weights in the positive direction and the negative direction respectively, which n means the precision of quantification is n bits. We label the weight after quantification from zero to positive direction and from positive direction to negative direction, the label of zero weight is 0 and the value of label increases when the weight is far from zero. If the label of the weight is P in the positive (negative) direction, then we apply P pulse to the corresponding positive (negative) weight in the memristor array, which is shown in Table 1.

Table 1. Mapping process in this work and write–verify operation.

Label of weight	Sign	Number of pulse to memristor
P	positive	P to positive weights (this work) $>$ 2P (write–verify operation)
	negative	P to negative weights (this work) $>$ 2P (write–verify operation)
0		nothing

2.4. Computational structure based on memristor array

In memristor array, by directly using Ohm's law and Kirchhoff's law, parallel multi-accumulate calculations (MAC) operations can be realized, which improves the calculation efficiency. The relationship between input and output in the neural network can be expressed as

$$Y = W \cdot X, \tag{13}$$

$$I = G \cdot V = G_+ \cdot V + G_- \cdot (-V) = V(G_+ - G_-), \tag{14}$$

where X represents the input of the neural network, W denotes the weight of the neural network, and Y refers to the output of the neural network. The input is converted into voltage V, and the weight is mapped into conductance G. In the memristor array, the product of weight and input is replaced with the Ohm's law of difference between conductance and voltage, which can be regarded as Eq. (13). Therefore, the memristor array can implement weighted calculations of inputs in neural networks, such as convolution and fully connected operations. In this work, we use the following method to implement convolution operations and fully connected operations of the memristor array.

We propose a structure of a fully connected layer of memristor, which is shown in Fig. 4. The input of the fully connected layer is M, the output is N, and the number of weights is $M \times N$. The input is converted into a voltage vector, and each of them is converted into a corresponding voltage and a reverse voltage according to the sign and magnitude. All weights are mapped to the memristor array in the form of a matrix, and the size and sign of each weight are represented by two adjacent memristors on the same line. The input voltage vector V is input from the WL line, and the output is obtained from the current of the SL line. The relationship between input and output can be expressed as Eqs. (14) and (15).

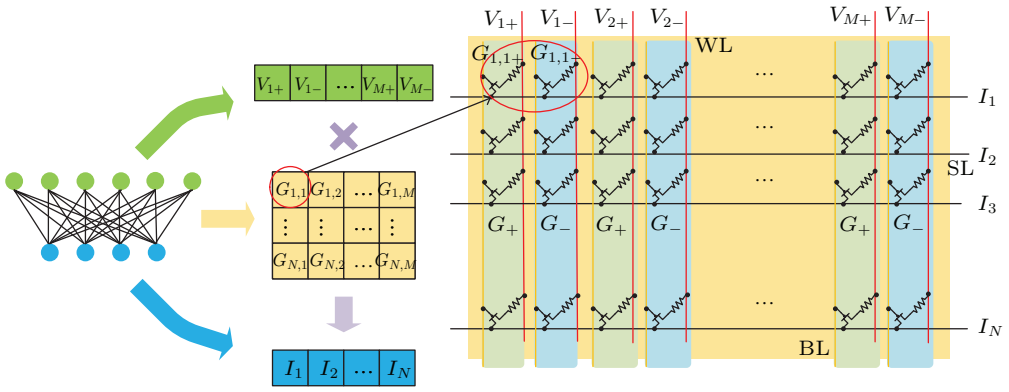


Fig. 4. Memristor array fully connected operation. The input is converted into a corresponding voltage value and a voltage in one direction, input from the WL line; the output result is read from the row current of the SL line. The row current vector is the resulting vector of the matrix operation.

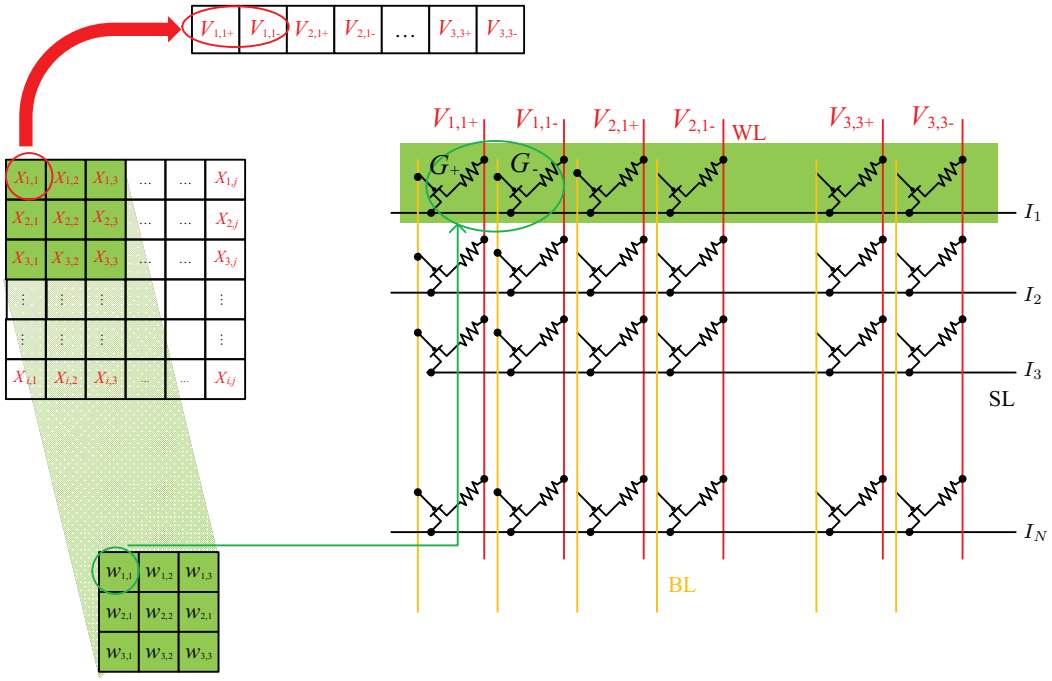


Fig. 5. Schematic diagram of convolution operation of memristor array. The input of the convolution is converted into the corresponding voltage value and reverse voltage value, input from the WL line to the memristor array, the result of the convolution is read from the current of each SL line, and each row current is a convolution result of the operation.

The $K \times K$ convolution kernel mapping and the equivalent convolution calculation of the memristor array are shown in Fig. 5. The convolution kernel is converted into a one-dimensional row vector, and two adjacent memristors in the same column represent a signed weight. We pre-process the input and rearrange the input pictures in the order of convolution. Then, the input is converted into corresponding positive voltage and negative voltage, respectively, and they are input to the two memristors, representing the difference between the positive equivalent memristor and the negative equivalent memristor. Finally, the output row current represents the result of convolution as given below.

$$V_i^- = -V_i^+, \tag{15}$$

$$I_j = \sum_{i=1}^{M} (V_{i,j}^+ \cdot G_{i,j}^+ + V_{i,j}^- \cdot G_{i,j}^-). \tag{16}$$

2.5. Dataset

Modified National Institute of Standards and Technology (MNIST) database is a large database of handwritten digits

that are usually used to train various image processing systems. Figure 6 shows part of images in MNIST dataset. There are handwritten digits 0 to 9 on the MNIST data set, and both the writing styles of the digits and their positions in different pictures are different.

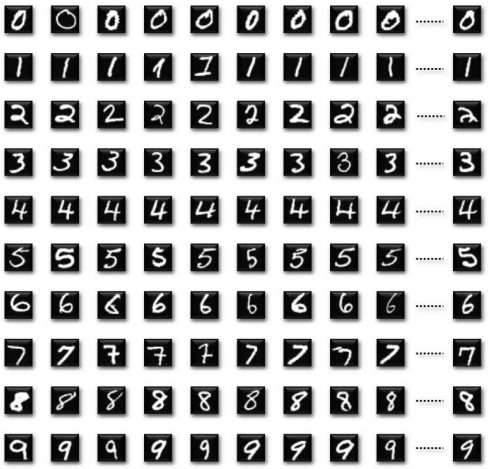


Fig. 6. Schematic diagram of some MNIST handwritten pictures, where the numbers 0–9 from top to bottom have different writing styles from each other.

In this work, we need to determine which number is in the picture. The database is also widely used for learning and testing in the field of machine learning, which contains 6×10^4 training images and 10^4 test images. Half of the training set and half of the test set come from the MNIST training data set, while the other half of the training set and the other half of the test set come from the NIST test data set. The size of each picture is 28×28. After conversion, each image is represented by a column vector composed of 784 elements. Each column vector is divided into 28 groups, each with 28 elements. To simplify the training, we binarize the gray value of the image. The gray value of the background part of the picture is 0, which denotes white, and the gray value of the digital part is 1, which refers to black.

3. Experiment

In this work, we demonstrated a five-layer memristive convolution neural network on the MNIST dataset. The size of each layer respectively was $1 \times 5 \times 5 \times 6$ for the first convolution layer, 3×3 for the first max-pooling layer, $6 \times 3 \times 3 \times 12$ for the second convolution layer, 2×2 for the second max-pooling layer, and 108×10 for the last fully connected layer as shown in Fig. 7. The parameters of each layer of memristor CNN are shown in Table 2. There are 6×10^4 training images of which the size is 28×28 and 10^4 test images for inference, all of them were divided into 10 classes from '0' to '9'. All training images were trained in the convolution neural network in MATLAB then benchmarked the network on the test dataset, with an accuracy of 98.23%. The distribution of trained weights is shown in Fig. 8, the weight shows a Gaussian-like distribution concentrated near the zero value. There are also weights distributed outside the range of $(-1, 1)$, we set these weights to be -1 or 1, which has less influence on the performance of the neural network.

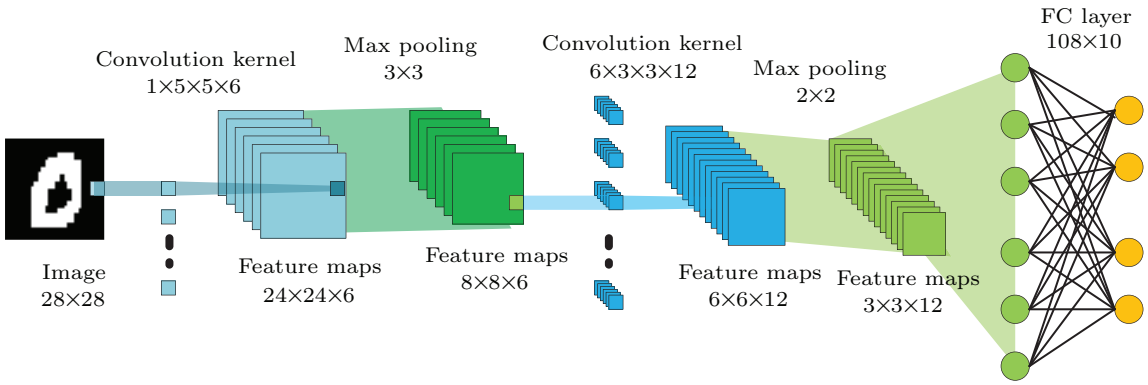


Fig. 7. Schematic diagram of CNN on MNIST dataset.

Table 2. Parameters of each layer of CNN.

	Size	Number of memristor for each weight	Number of memristor
The first convolution layer	$1 \times 5 \times 5 \times 6$	2	300
The second convolution layer	$6 \times 3 \times 3 \times 12$	2	1296
Fully-connected layer	108×10	2	2160
Sum			3756

Here, we first quantified the trained weight of the CNN to 3-bit using the uniform quantification and Gaussian quantification. The preliminary result in accordance with the above setting was not satisfactory as shown in Fig. 9. The quantified weight using Gaussian quantification in 3-bit made the accuracy dramatically decrease as compared with 64 bits. The

test accuracy of the weight used in uniform quantification in 3 bits also decreased a lot (by 31.93%), for which the reason is that the distribution of quantify weight has changed, both of the weight distributions are more concentrated than the distribution before the quantification as shown in Fig. 10(a). The concentrated part of the trained weight is quantified into a certain simulation weight, which results in weakening the ability of the network to extract features. We dispersed the part of concentrated date after having been quantified as shown in Fig. 10(b), the classification has been improved by 72.67%, which implies that compared with the value of the weight, the distribution of weight could determine the performance of the neural network more. The recognized accuracy of CNN using 3-bit uniform quantification was 90.3%, which means that the uniform quantification is the better way to quantify the weight to low bits as compared with Gaussian quantification. Because the Gaussian quantification makes the simulation weight more centralized than uniform quantification, the distribution of weight is changed.

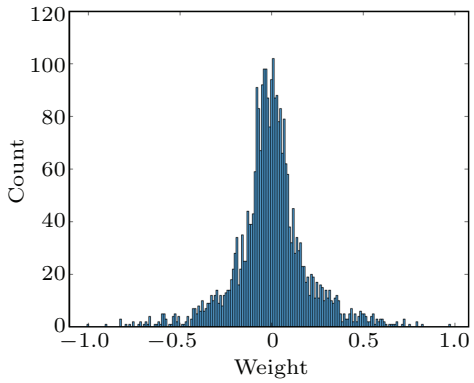


Fig. 8. CNN weight distribution map training on 64-bit precision.

3.1. Influence of weight quantification and mapping in CNN

The precision of quantification plays a key role in studying the neural network performance. With the proper precision, the neural network could attain a higher classification accuracy. We further investigate the influence of precision on network performance. The precision is increased gradually from 4 bits to 7 bits, and the classification accuracies of weights with different precision are shown in Fig. 9. With the count of simulation weights increasing, the most concentrated part of weights could be assigned to more simulation weights. The accuracy of inference with Gaussian quantification under the 4-bit precision increases obviously as compared with that under 3-bit precision. However, when the precision is higher than 4 bits, the accuracy is slightly increases for both quantification conditions. It means that it is enough for CNN to form the distribution which maintains a good performance under the 4-bit precision.

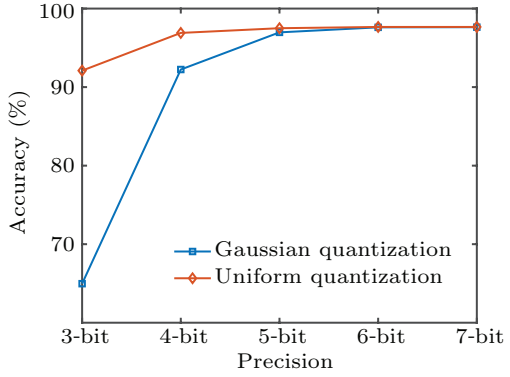


Fig. 9. Classification of different precisions (3 bits to 7 bits) using Gaussian quantification and uniform quantification.

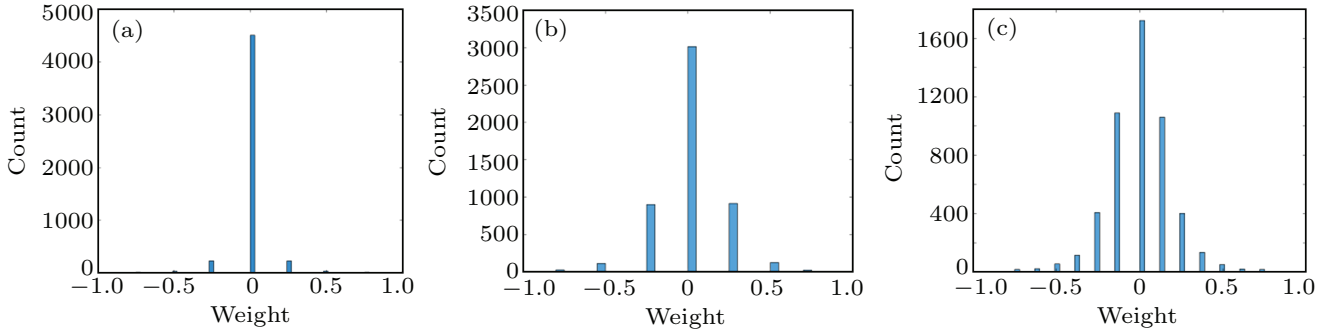


Fig. 10. Distribution of weight with 3-bit Gaussian quantification and uniform quantification: (a) Gaussian quantification, (b) depart some of weights after Gaussian quantification, (c) uniform quantification.

Besides, we also evaluated the influence of the conductance interval in the mapping process. Applying the pulses of different amplitudes to the VTEAM model, the intervals of conductance change will be different, which has been discussed in the above section. We mapped the 4-bit simulation weight into memristor array with pulse amplitudes of 0.8 V, 1 V, 1.5 V, and 2 V respectively. The different numbers of pulses could modulate the conductances of memristors to various values. The classification accuracy is shown in Fig. 11, in which the red line is the accuracy of inference under 64 bits in the CPUs. When the pulse amplitude increase, the classification of the CNN decreases. The accuracy drops to 90% when the pulse amplitude is 2 V. Besides the conductance interval in the mapping process, the device linearity may also be a key

element for the performance of the CNN, which needs further studying. In this work, we trained the CNN in MATLAB, and then quantified the weight with uniform quantification under 4 bits, the quantified weight was mapped into the memristor array with 0.8 V of pulse amplitude of which the distribution of conductance is shown in Fig. 12, We achieved the comparable recognition accuracy (97.36%) with inference in CPUs under 64 bits (98.23%).

In our experiments, we have used different ANNs to verify the availability of the algorithm. The MLP and Lenet-5 are tested by using the same methods, and the classification accuracies of both neural networks are shown in Table 3. The algorithm worked well when the symmetry in SET and in RESET were bad. It was because the conductance of the memristor in the mapping process had been modulated once. The experiment explained that the off-chip learning algorithm has the potential to operate various ANN. Furthermore, we added noise into the mapping process to simulate the memristor variability of the memristor, which can be described in the following form:

$$C_{\text{noise}} = \pm((C_{\text{initialize}} + C_{\text{max}})/N) \times \text{random}(0,1), \quad (17)$$

where C_{noise} is the noise of conductance, $C_{\text{initialize}}$ is the conductance of applying zero pulse, C_{max} is the conductance when using the maximum number of pulses, and random is the random interval (0,1). Table 3 shows that the algorithm is effective in consideration of the memristor variability.

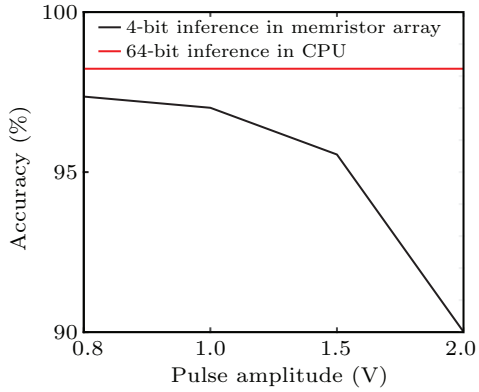


Fig. 11. Accuracy varying with pulse amplitude of memristor array for 4- and 64-bit inferences.

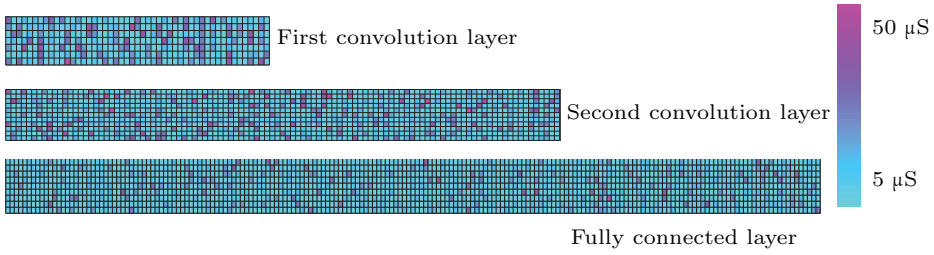


Fig. 12. Mapping 4-bit uniform quantification to the conductance distribution of the memristor when pulse amplitude is 0.8 V.

Table 3. Accuracies of various neural networks during training and inference implemented on memristor arrays.

	64-bit (%)	4 bits in memristor (%)	4 bits with noise in memristor (%)
MLP	97.8	96.44	95.50
LENET	97.82	95.83	95.40
CNN	98.23	97.36	94.49

4. Conclusions

In this paper, we proposed a memristor array deployment algorithm of ANN based on the off-chip learning. The simulation results show that the proposed algorithm can achieve a classification accuracy of 97.36% with a 5-layer CNN structure on the MNIST data set while the intermediate results significantly decrease to 4 bits. In addition, we discuss the influences of different quantification methods and precision on the performance of CNN. The uniform quantification makes CNN show better recognition accuracy, and it is enough for accuracy to quantify the weight into 4 bits. Then, we study the influence of conductance interval in the mapping process, conclude that the smaller the conductance interval, the better the CNN performance is. Besides, we use the algorithm to implement MLP and LENET on the memristor arrays and add the noise into the process of mapping. The results show that it is also applicable to other neural networks. Experimental results demonstrate that the algorithm can be implemented on a memristive computation platform and tolerate the device characteristics such as the device noise and low-precision.

Acknowledgements

Project supported by the National Natural Science Foundation of China (Grant Nos. 62076208, 62076207, and U20A20227) and the National Key Research and Development Program of China (Grant No. 2018YFB1306600).

References

[1] Uijlings J R R, van de Sande K E A, Gevers T and Smeulders A W M 2013 *International Journal of Computer Vision* **104** 154

[2] Richardson F, Reynolds D and Dehak N 2015 *IEEE Signal Processing Letters* **22** 1671

[3] LeCun Y, Bengio Y and Hinton G 2015 *Nature* **521** 436

[4] Li C, Hu M, Li Y, Jiang H, Ge N, Montgomery E, Zhang J, Song W, Davila N, Graves C E, Li Z, Strachan J P, Lin P, Wang Z, Barnell M, Wu Q, Williams R S, Yang J J and Xia Q 2018 *Nat. Electron.* **1** 52

[5] Coates A, Huval B, Wang T, Wu D J, Ng A Y and Catanzaro B 2013 30*th International Conference on Machine Learning*, (ICML 2013), June 16–21, 2013, Atlanta, USA, p. 2374

[6] Jouppi N P, Young C and Patil N 2017 44*th Annual International Symposium on Computer Architecture* (ISCA 2017), June 24–28, 2017, Toronto, Canada, p. 1

[7] Chen Y H, Krishna T, Emer J S and Sze V 2017 *IEEE Journal of Solid-State Circuits* **52** 127

[8] Zidan M A, Strachan J P and Lu W D 2018 *Nat. Electron.* **1** 22

[9] Boschker J E and Calarco R 2017 *Adv. Phys. X* **2** 675

[10] Boybat I, Le Gallo M, Nandakumar S R, Moraitis T, Parnell T, Tuma T, Rajendran B, Leblebici Y, Sebastian A and Eleftheriou E 2018 *Nat. Commun.* **9** 2514

[11] Indiveri G, Linares-Barranco B, Legenstein R, Deligeorgis G and Prodromakis T 2013 *Nanotechnology* **24** 384010

[12] Chua L O 1971 *IEEE Transactions on Circuit Theory* **CT18** 507

[13] Shang L, Duan S, Wang L and Huang T 2018 *IEEE Transactions on Very Large Scale Integration (Vlsi) Systems* **26** 2830

[14] Li Y, Li J, Li J, Duan S, Wang L and Guo M 2021 *Neurocomputing* **454** 382

[15] Strukov D B, Snider G S, Stewart D R and Williams R S 2008 *Nature* **453** 80

[16] Yu F, Zhang Z, Shen H, Huang Y, Cai S and Du S 2021 *Chin. Phys. B* **1088** 152

[17] Xu Q, Ju Z, Ding S, Feng C, Chen M and Bao B 2021 *Cognitive Neurodynamics* **1007** 64

[18] Le Gallo M, Sebastian A, Mathis R, Manica M, Giefers H, Tuma T, Bekas C, Curioni A and Eleftheriou E 2018 *Nat. Electron.* **1** 246

[19] Prezioso M, Merrikh-Bayat F, Hoskins B D, Adam G C, Likharev K K and Strukov D B 2015 *Nature* **521** 61

[20] Chen J, Wang L and Duan S 2021 *Neurocomputing* **461** 129

[21] Li C, Belkin D, Li Y, Yan P, Hu M, Ge N, Jiang H, Montgomery E, Lin P and Wang Z 2018 *Nat. Commun.* **9** 1

[22] Yao P, Wu H, Gao B, Eryilmaz S B, Huang X, Zhang W, Zhang Q, Deng N, Shi L and Wong H S P 2017 *Nat. Commun.* **8** 1

[23] Gao B, Bi Y, Chen H Y, Liu R, Huang P, Chen B, Liu L, Liu X, Yu S, Wong H S P and Kang J 2014 *Acs Nano* **8** 6998

[24] Jo S H, Chang T, Ebong I, Bhadviya B B, Mazumder P and Lu W 2010 *Nano Lett.* **10** 1297

[25] Hu M, Li H, Wu Q, Rose G S and Chen Y 2012 *The International Joint Conference on Neural Networks (IJCNN)*, 2012, p. 1

[26] Li B, Shan Y, Hu M, Wang Y, Chen Y and Yang H 2013 *International Symposium on Low Power Electronics and Design (ISLPED)*, 2013, p. 242

[27] Zhang Q, Wu H, Yao P, Zhang W, Gao B, Deng N and Qian H 2018 *Neural Networks* **108** 217

[28] Lim S, Bae J H, Eum J H, Lee S, Kim C H, Kwon D, Park B G and Lee J H 2019 *Neural Computing & Applications* **31** 8101

[29] Merrikh-Bayat F, Guo X, Klachko M, Prezioso M, Likharev K K and Strukov D B 2018 *IEEE Transactions on Neural Networks and Learning Systems* **29** 4782

[30] Hikawa H, Tamaki M and Ito H 2018 *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences* **E101A** 499

[31] Wong H S P, Lee H Y, Yu S, Chen Y S, Wu Y, Chen P S, Lee B, Chen F T and Tsai M J 2012 *Proc. IEEE* **100** 1951

[32] Zamanidoost E, Klachko M, Strukov D and Kataeva I 2015 *Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH'15)* 2015, p. 139

[33] Chi P, Li S, Xu C, Zhang T, Zhao J, Liu Y, Wang Y and Xie Y 2016 "PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory", in: 2016 *ACM/IEEE* 43*rd Annual International Symposium on Computer Architecture (ISCA)*, p. 27

[34] Wang Z, Wu H, Burr G W, Hwang C S, Wang K L, Xia Q and Yang J J 2020 *Nat. Rev. Mater.* **5** 173

[35] Kvatinsky S, Ramadan M, Friedman E G and Kolodny A 2015 *IEEE Transactions on Circuits and Systems II-Express Briefs* **62** 786