Hardware-Friendly Stochastic and Adaptive Learning in Memristor Convolutional Neural Networks

Wei Zhang, Lunshuai Pan, Xuelong Yan, Guangchao Zhao, Hong Chen, Xingli Wang, Beng Kang Tay, Gaokuo Zhong,* Jiangyu Li,* and Mingqiang Huang* 

Memristors offer great advantages as a new hardware solution for neuromorphic computing due to their fast and energy-efficient matrix vector multiplication. However, the nonlinear weight updating property of memristors makes it difficult to be trained in a neural network learning process. Several compensation schemes have been proposed to mitigate the updating error caused by nonlinearity; nevertheless, they usually involve complex peripheral circuits design. Herein, stochastic and adaptive learning methods for weight updating are developed, in which the inaccuracy caused by the memristor nonlinearity can be effectively suppressed. In addition, compared with the traditional nonlinear stochastic gradient descent (SGD) updating algorithm or the piecewise linear (PL) method, which are most often used in memristor neural network, the design is more hardware friendly and energy efficient without the consideration of pulse numbers, duration, and directions. Effectiveness of the proposed method is investigated on the training of LeNet-5 convolutional neural network. High accuracy, about 93.88%, on the Modified National Institute of Standards and Technology handwriting digits datasets is achieved (with typical memristor nonlinearity as ±1), which is close to the network with complex PL method (94.7%) and is higher than the original nonlinear SGD method (90.14%).

1. Introduction

Neural networks have been widely explored in artificial intelligence (AI)-based applications such as computer vision and speech recognition. However, the AI algorithms running on traditional complementary metal–oxide–semiconductor (CMOS) digital platforms have been largely constrained by either the approaching end of Moore’s law or the limitation of von Neumann architecture. Novel neuromorphic computing architectures beyond CMOS technology are strongly desired. Recently, memristors (defined as memristive resistors) and memristor-based neuromorphic computing have attracted enormous attention due to their great energy efficiency. As new types of nanoscale devices, memristors provide many advantageous features such as excellent scalability (<10 nm), fast write/read speed (~ns), low energy consumption (~fJ/bit per operation), and nonvolatility. Furthermore, memristors are particularly appealing for realizing artificial neural networks. For the network inference process (forward propagation), memristors can be treated as a variable resistor with tunable synaptic weight (conductance), and the vector multiplication can be easily implemented using Ohm’s law and Kirchhoff’s law. For the network training process (back propagation), the synaptic weight can be updated by applying a certain number of electrical pulses (including the information on pulse direction, duration time, and amplitude) on the device terminals determined by the stochastic gradient descent (SGD) updating algorithm. To date, many experimental demonstrations of memristor networks have been featured. For example, Yao et al demonstrated the CMOS compatible network for gray-scale face
Li et al. developed multilayer memristor neural networks for image processing. Recently, Yao et al. reported the fully hardware implementation of memristor-based convolutional neural networks (CNNs). However, most of the previous works only focused on the inference process, and the training process is rarely addressed. As stated in Yao’s work, “In contrast to the pure in-situ training solution, the ex-situ training method appears to be a shortcut that takes advantage of existing high-performing parameters.” This is because there still exist several bottlenecks that need to be solved for the training process. First, the training process is difficult to be implemented by memristor hardware itself and is usually conducted by a host computer. Second, the online learning process demands great computation resources to calculate the parameters in weight updating, which would increase not only the complexity in peripheral circuits design, but also the power consumption. Third, the nonlinear and asymmetric conductance updating properties have largely degraded the network performance.

Li developed the two-pulse conductance programming scheme using SGD algorithm to address the challenges in memristor conductance programming, which enables the network to continuously adapt its knowledge and significantly improves its accuracy and defect tolerance. However, the nonlinear property remains a great challenge. Wang proposed a piecewise linear (PL) method to mitigate the weight update error caused by the nonlinearity of memristors. Compared with the classical nonlinear SGD method, it provides higher overall accuracy. However, it involves complex peripheral circuits to calculate and store the updating parameters, which is actually not energy efficient.

Generally, there are approaches to improve the performance of memristor networks by ex situ training, but most of them require that the parameters be tuned based on specific knowledge of the hardware (e.g., peripheral circuitry) and memristor array (e.g., device defects). In this work, we propose a concise and effective learning method for the memristor network weight updating. The in situ training adapts the weights and compensates them automatically and thus is more powerful and energy efficient in training neural networks.

The proposed method in this work is inspired by the traditional SGD method but no learning-rate parameter (an important parameter in SGD algorithm) is specifically needed here. In each updating process, only the updating direction (to be either increased or decreased) is provided. Therefore, it is self-adaptive and is free of calculating the specific numbers of electrical pulses. In addition, the probability updating algorithm has been intrinsically embedded in our method, and the nonlinear and stochastic updating can be fully implemented by the memristor network itself. Compared with the traditional nonlinear SGD updating algorithm and the PL updating method used in memristor neural network, our method is fully hardware friendly without complex peripheral circuits. Effectiveness of the proposed method has been carefully investigated on training of a classical LeNet-5 CNN and the demonstrated result shows that our stochastic-adaptive learning method exhibits good performance compared with the PL method and is better than the nonlinear SGD method.

2. Memristor CNN

The CNN was first proposed in the 1980s. However, it has not been widely used because the training algorithm was difficult. LeCun et al. applied a gradient-based learning algorithm to train a CNN and obtained successful results. After that, researchers further improved the CNN and reported strong results in the field of image recognition. To date, CNN is the top research subject for AI applications due to its great performance. The overall architecture of CNN is structured as a series of convolution layers. The convolution layer exploits the property that many natural signals are compositional hierarchies, in which higher-level features can be obtained by composing lower-level ones. For example, edges form motifs, motifs assemble into parts, and parts form objects. In deep CNNs, these specific features are detected by a series of convolution kernels (filters), of which the vector multiplication is used to measure the similarity between input feature and convolution filters.

LeNet-5 is the first successful CNN, which was designed for handwritten and machine-printed character recognition in the 1990s. As shown in Figure 1a, the LeNet-5 contains two convolutional layers and three fully connected (FC) layers. The FC layer can also be treated as a convolution operation with 1 * 1 filter; therefore, all the calculations in CNN are equivalent to the matrix vector multiplication (MVM) of Feature(H) / W / CH_in and Weight(K) / [CH_in] / [CH_out].

\[ F_{out}[h][w][ch_{out}] = \sum_{k_x=0}^{K_x} \sum_{k_y=0}^{K_y} \sum_{ch_{in}=0}^{CH_{in}} (F_{in}[h+k_x][w+k_y][ch_{in}][ch_{out}]) \]

in which the kernel size of convolution weight is \( K_x \times K_y \), input channel is \( CH_{in} \), output channel is \( CH_{out} \), and height and width of the input feature is \( H \) and \( W \), respectively.

The matrix multiplication can be directly implemented using the memristors network as shown in Figure 1b. For the forward propagation, the multiplication operation can be directly obtained using Ohm’s law in terms of \( I_j = V_{in} \times G_{ij} \), whereas the accumulation operation can be done by Kirchhoff’s law, \( I_{out} = \sum V_{in} \times G_{ij} \) at the output neuron, where \( G_{ij} \) represents the corresponding conductance of the memristor.

For the back propagation and weight updating, the desired conductance of each memristor can be translated into a certain number of electrical pulses for each device determined by the standard SGD algorithm. Figure 1c shows the typical performance of an electronic synaptic device (such as memristor and synaptic transistor). Conductance of the device can be well modulated by the input electrical pulse (including the pulse number, pulse duration time, pulse directions, and pulse amplitudes). When the electrical pulse is positive, the conductance shows a trend of continuous increase, regarding long-term potentiation (LTP, strengthening of synapses based on recent patterns of activity). However, with negative pulse, the device conductance will be gradually depressed, corresponding to the long-term depression process (LTD, decreasing in synaptic strength). LTP and LTD are the core parts of synaptic plasticity.
meaning that the synaptic weight can be modulated and thus can be applied in artificial neural networks.

An ideal behavior of a synaptic device desires a “linear” relationship between the conductance and the number of programmed voltage pulses, namely, \( G_{\text{ideal}} = k + N_{\text{pulse}} \). However, the practical device reported in literature does not follow such ideal trajectory, exhibiting “exponential-type” updating characteristics.\(^{[10]}\) What is worse, the precision of the conductance is not high and usually limited to be 4–8 bit (corresponding to 16–256 conductance states). The random conductance variation is inevitable due to the physical limitations, including the inherent drift and diffusion dynamics of the ions/vacancies in the device.\(^{[7]}\) Thus, it is difficult to train a memristor neural network in situ.

To analyze the impact of nonlinear weight updating during network training, a general behavior model\(^{[10]}\) is developed. The device properties can be well described with five parameters: \( G_{\text{max}} \) (the maximum conductance), \( G_{\text{min}} \) (the minimum conductance), \( P_{\text{max}} \) (the number of conductance states), \( A_p \) (nonlinearity of the LTP updating), and \( A_d \) (nonlinearity of the LTD updating). The conductance can then be deduced by

\[
G(x) = \frac{(G_{\text{max}} - G_{\text{min}})}{1 - \exp(-A)} \left[ 1 - \exp\left(-\frac{A}{P_{\text{max}} \cdot x}\right) \right] + G_{\text{min}} \tag{2}
\]

in which \( A \) is \( A_p \) (for the LTP updating) or \( A_d \) (for the LTD updating) and \( x \) represents the current state of the pulse, and it ranges from 0 to \( P_{\text{max}} \).

\( G_{\text{max}}, G_{\text{min}}, \) and \( P_{\text{max}} \) can be directly measured in the electrical characterization process. \( A_p \) and \( A_d \) can be extracted from the conductance–pulse curve. To get the further numerical analysis result, the normalized conductance–pulse curve of the mode device is shown in Figure 2, in which different \( A_p \) and \( A_d \) are plotted. It is worth mentioning that the \( A_p \) and \( A_d \) of one device are usually unrelated, and for most practical devices, the \( A_p \) is empirically better than that of \( A_d \). Take the device in Figure 1c as an example, the \( G_{\text{max}}, G_{\text{min}}, P_{\text{max}}, A_p, \) and \( A_d \) is 20, 3.5, 32, 0.1, and −3.5, respectively.

### 3. Stochastic and Adaptive Learning in Memristor Networks

Next, we will tend to the memristor network learning process. It should be first pointed out that the synaptic weight in a neural network can be either positive or negative. However, the conductance of the device is always positive. To map the positive conductance into both positive and negative ones, many previous

![Figure 1](image1.png)

**Figure 1.** a) Structure of LeNet-5 used in this work, of which the total number of parameters is about 44 000. b) Memristor network with the function of MVM. c) Typical electrical characterization of a memristor device with nonlinear LTP and LTD.

![Figure 2](image2.png)

**Figure 2.** Normalized nonlinear conductance–pulse curve of memristor. The device exhibits "exponential-type" of updating characteristics in both LTP and LTD.
works\(^\text{[9,11–13]}\) have taken advantage of the differential-pair method (Figure 3a,b). Generally, the output current of the original memristor network will be first translated into voltage by a current–voltage converter, namely, \(V_{\text{out}}^{+} = -R*(\sum V_{\text{in}}*G_{ij}^{+})\) and \(V_{\text{out}}^{-} = -R*(\sum V_{\text{in}}*G_{ij}^{-})\). Then, the final output can be obtained using the integrated operational amplifier circuit \(V_{\text{out}} = V_{\text{out}}^{+} - V_{\text{out}}^{-} = R*\sum V_{\text{in}}*(w_{1} - w_{2})\).

For the weight updating process, if the weight is calculated (according to the SGD algorithm) to be increased, several positive electrical pulses will be applied to the device \(w_{1}\); meanwhile, the device \(w_{2}\) is untouched. If the weight has to be decreased, several negative electrical pulses will be applied to the device \(w_{1}\) and the device \(w_{2}\) is still untouched. The above process shows the standard weight updating method used in today’s memristor network (which is named as “mode-0”). However, there exist several shortcomings here. First, the number of electrical pulses should be calculated before updating, which will largely increase the computing complexity.\(^{[14,15,25,26]}\) What is worse, in some cases, is that the forms of the pulses are variable that the direction or amplitude or duration time of the pulse will have to be changed for a higher accuracy, which will increase the design complexity.\(^{[10,12,16]}\) Secondly, the second memristor (device \(w_{2}\)) in the differential-pair is never updated.\(^{[12]}\) Third, the nonlinear characteristic of the device is inextricable.

Adaptive learning system is an automatic control system that preserves its operational capability under conditions of unforeseen change in the properties of the controlled system. For a neural network, an adaptive system desires self-optimization capabilities\(^{[12]}\) and is able to continuously adjust the parameters. Standing on this point, here, we propose new learning algorithms to address the earlier issues. Four different kinds of updating modes are developed, as shown in Table 1.

To improve the network accuracy with an elaborate updating algorithm, “mode-1” and “mode-2” (as the normal control group) are proposed with the 1-transistor-1-memristor plus 1-transistor-1-memristor (1T1R-IT1R) structure, which is the same as the classical nonlinear SGD algorithm (“mode-0”). The difference is that, in “mode-0,” the two memristors in the differential-pair are exactly the same, whereas in our design (both “mode-1” and “mode-2”), these two are with different device size (Note that this will not introduce additional issues during the device/chip design and fabrication process).

To reduce the circuit area, “mode-3” (as the normal control group) and “mode-4” are designed using the 1-transistor-1-memristor plus 1-resistor (1T1R-1R) structure. This is because the 1T1R-1IR involves twice the number of required devices, whereas the 1T1R-1R structure is able to share the resistor for different differential-pairs.\(^{[24]}\) In this way, about 50% circuit area in the memristor network can be saved theoretically. The difference between “mode-3” and “mode-4” is the initialization method of the resistor’s resistance. In “mode-3,” all the resistance (weights of \(w_{2}\)) will be randomly initialized at the very beginning and then be fixed during the whole updating process, which is exactly the same as “mode-0.” In mode-4, the resistance is always fixed at \((G_{\text{max}}/2 + G_{\text{min}}/2)\).

The weight updating method in the earlier four different modes is different from that of “mode-0” mainly because it is not necessary to calculate the specific number of voltage pulses, and the form of the pulse is also fixed without the change in duration time, amplitude, and directions. What we need is just to figure out the sign of updating \(\Delta w\) to be either positive or negative for the training. The simple learning rules make the memristor network capable of updating its knowledge adaptively, thus getting the adaptation and self-optimization capabilities for more conditions.

Figure 3. a) Differential-pair design of the typical 1T1R-1T1R memristor network, in which 1T1R means 1-transistor-1-memristor structure. b) Weight updating in the differential-pair.
The updating algorithm can be easily explained in “mode-2,” “mode-3,” and “mode-4.” In “mode-2,” namely, \( w = w_1 - w_2 \). If the weight of differential-pair \( w \) needs to be increased, we send one positive pulse to device \( w_1 \) and one negative pulse to device \( w_2 \). If the weight \( w \) needs to be decreased, we send one negative pulse to \( w_1 \) and one positive pulse to \( w_2 \). For “mode-3” and “mode-4,” namely, \( w = w_1 - R \). If the weight \( w \) needs to be increased, we should just send one positive electrical pulse to the first memristor (\( w_1 \)), and vice versa. These three updating methods are deterministic, because the practical weight updating directions are always the same as the algorithm suggested ones.

For “mode-1,” the updating algorithm is stochastic and can be described in the following steps.

**Step 0:** Randomly set the network weights into two parts: “default positive” and “default negative.” The “default positive” means that the differential-pair in the hardware is designed to be \( w = w_1 - w_2 \), and the “default negative” means that the differential-pair in the hardware is designed to be \( w = w_2 - w_1 \), where the devices \( w_1 \) and \( w_2 \) are with different area, and the area of device \( w_1 \) is larger than that of device \( w_2 \).

**Step 1:** Random initialization. Though the size of \( w_1 \) is larger than \( w_2 \), it does not mean that the value of \( w = w_1 - w_2 \) is always positive due to random initialization. The probability can be calculated as the following. Suppose \( w_1 \) and \( w_2 \) are randomly initialized with a distribution of \( f(x) \), the range of \( w_1 \) is \([G_{\text{min}1}, G_{\text{max}1}] \), and the range of \( w_2 \) is \([G_{\text{min}2}, G_{\text{max}2}] \); then, the probability of \( w = w_1 - w_2 > 0 \) can be expressed as

\[
P_{w_1 > w_2} = \int_{G_{\text{min}2}}^{G_{\text{max}2}} f(w_1) f(w_2) dw_1 dw_2
\]

For example, if \( f(x) \) represents uniform distribution, and \( G_{\text{min}1} = G_{\text{min}2} = 0 \), \( G_{\text{max}1} = 2 \), and \( G_{\text{max}2} = 1 \), the probability is \( P_{w_1 > w_2} = 73\% \).

**Step 2:** One-pulse updating. Using the standard SGD algorithm, we get the updating directions and determine whether \( \Delta w \) is positive or negative. If \( \Delta w > 0 \), one positive pulse will be sent to both \( w_1 \) and \( w_2 \) for the “default positive” differential-pair devices (see step 0). One negative pulse will be sent to both \( w_1 \) and \( w_2 \) for the “default negative” differential-pair devices. If \( \Delta w < 0 \), we need to send one negative pulse to both \( w_1 \) and \( w_2 \) for the “default positive” devices and one positive pulse to both \( w_1 \) and \( w_2 \) for the “default negative” devices.

**Step 3:** Stop the learning process until all the cycling loops (including the batch-number loop, layer-index loop, and device-number loop) are finished.

The aforementioned updating algorithm can be easily conducted in both software simulation and hardware implementation (Table 2). However, there is still one uncertain point to be emphasized—that the one-pulse updating may have failed due to the random initialization of \( w_1 \) and \( w_2 \). For instance, how can we guarantee that the conductance of \( w_{\text{new}} \) is really larger than \( w_{\text{old}} \) in the “default positive” devices (\( w = w_1 - w_2 \)) after one-pulse positive updating?

**Figure 4a** shows the typical updating process for \( \Delta w > 0 \) in “default positive” device as an example. The conductance can be written as

\[
W_{\text{old}} = w_{1 \text{ old}} - w_{2 \text{ old}} = G(x_1) - G(x_2)
\]

\[
W_{\text{new}} = w_{1 \text{ new}} - w_{2 \text{ new}} = G(x_1 + 1) - G(x_2 + 1)
\]

\[
\Delta W = W_{\text{new}} - W_{\text{old}} > 0
\]

where \( x \) is the memristor pulse state.

Equation (6) represents the successful learning after one-pulse updating, and it can be further simplified using Taylor’s expansion with the low-order approximation. For the first-order linear approximation, it is equivalent to

\[
\Delta W = [G(x_1 + 1) - G(x_1)] - [G(x_2 + 1) - G(x_2)]
\]

\[
\approx G'(x_1) - G'(x_2) > 0
\]

The probability of \( G'(x_1) - G'(x_2) > 0 \) is then calculated to be

\[
P_{G'(x_1) > G'(x_2)} = \int_{0}^{\infty} \frac{A_2}{A_1} G_{\text{max}1} - G_{\text{min}1} \frac{1}{1 - e^{-A_1 x}} dx
\]

\[
\ln \frac{A_1 G_{\text{max}1} - G_{\text{min}1}}{A_2 G_{\text{max}2} - G_{\text{min}2}} \frac{1}{1 - e^{-A_1 x}} \]

---

**Table 1.** Different weight updating methods. (Note 1: “\( \uparrow \)" means increase and “\( \downarrow \)" means decrease. Note 2: In “mode-0,” “mode-3,” and “mode-4,” the weight is written as \( w = w_1 - R \) because \( w_2 \) is always fixed to a constant value of \( R \).)

| Reference | Purpose | Differential-pair structure | Weight | \( \Delta w \) update | \( w_{\text{new}} \) update | \( w_{\text{old}} \) update |
|-----------|---------|----------------------------|--------|-----------------------|---------------------------|--------------------------|
| Nonlinear | Mode-0  | Baseline                   | 1T1R-1T1R | if \( \Delta w > 0 \) | \( w_1 \)                 | Fixed after initialization |
| SGD Algorithm | Mode-1 (nonlinearity & stochasticity) | For high performance | 1T1R-1T1R | if \( \Delta w < 0 \) | \( w_1 \)                 | Fixed after initialization |
|           | Mode-2 (nonlinearity) | Normal control group | 1T1R-1T1R | if \( \Delta w > 0 \) | \( w_1 \)                 | Fixed after initialization |
|           | Mode-3 (nonlinearity) | Normal control group | 1T1R-1R | if \( \Delta w < 0 \) | \( w_1 \)                 | Fixed after initialization |
|           | Mode-4 (nonlinearity) | For less chip area | 1T1R-1R | if \( \Delta w > 0 \) | \( w_1 \)                 | Fixed after initialization |
|           |         |                           |         | if \( \Delta w < 0 \) | \( w_1 \)                 | Fixed after initialization |
where \( P_{\text{max}} A_1, G_{\text{max}1}, G_{\text{min}1} \) and \( A_2, G_{\text{max}2}, G_{\text{min}2} \) are the attributes of device \( w_1 \) and \( w_2 \), respectively. The temporary variables are defined as

\[
Z = \frac{A_1}{A_2} \frac{P_{\text{max}} - C}{P_{\text{max}}}
\]

(9)

\[
C = \frac{P_{\text{max}}}{A_1} \ln \left( \frac{A_1 G_{\text{max}1} - G_{\text{min}1}}{A_2 G_{\text{max}2} - G_{\text{min}2}} \right) \left( 1 - e^{-A_1} \right)
\]

(10)

The probability of \( \mathcal{G}(x_1) - \mathcal{G}(x_2) > 0 \) shows a strong relationship with the ratio of \( G_{\text{max}1}/G_{\text{max}2} \). For example, if \( G_{\text{max}1}/G_{\text{max}2} = 1 \) (meanwhile \( G_{\text{max}1} = G_{\text{max}2} = 0 \)), the result is 50%, which is consistent with the intuitive knowledge. If \( G_{\text{max}1}/G_{\text{max}2} = 2 \), the probability can reach as high as 95.4%, as shown in Figure 4c (black dots), which means that the conductance of \( w_{\text{new}} \) shows a significant high probability to be practically larger than \( w_{\text{old}} \) after the one-pulse updating. Similar analysis on \( \Delta w < 0 \) can be obtained, as shown in Figure 4b.

**Figure 4.** a) Analyses of the updating process for \( \Delta w > 0 \) in "default positive" devices. b) Updating process for \( \Delta w < 0 \). c) Black dots: the probability of a successful one-pulse updating at different \( G_{\text{max}1}/G_{\text{max}2} \). Blue dots: the effective weight precision at different \( G_{\text{max}1} \). Note: \( G_{\text{min}1} = G_{\text{min}2} = 0.01, G_{\text{max}2} = 1 \), and \( G_{\text{min}1}/G_{\text{max}1} = G_{\text{min}2}/G_{\text{max}2} \).
However, it should be noted that with the increasing $G_{\text{max}}/G_{\text{max}}$ ratio, the effective weight precision (defined as $1/(G_{\text{max}}-G_{\text{max}})$) will be decreased because the number of conductance states is fixed, and this will degrade the network performance, which will be discussed later.

4. Network Performance

Before training the network using our stochastic-adaptive method, the linear scaling factor $k$ of each layer should be determined first, because the distribution range of the weight is largely dependent on the $k$-value.\(^\text{11-13}\) It is better to make the network weight parameter’s range consistent with the memristor’s conductance.

In software, $k$ is the factor of Rectified Linear Unit (ReLU) activation function.

$$Y_{\text{out}} = \begin{cases} k* (\sum F_{\text{in}}* W) & @ \sum F_{\text{in}}* W \geq 0 \\ 0 & @ \sum F_{\text{in}}* W < 0 \end{cases} \quad (11)$$

However in hardware, $k$ is mainly the value of the resistance $R$ in the current-voltage converter, as shown in Figure 3a. The activation in circuit is designed using

$$V_{\text{out}} = \begin{cases} R* (\sum V_{\text{in}}* (w_1 - w_2)) & @ \sum V_{\text{in}}* (w_1 - w_2) \geq 0 \\ 0 & @ \sum V_{\text{in}}* (w_1 - w_2) < 0 \end{cases} \quad (12)$$

The effectiveness of the proposed method has been investigated on the classical LeNet-5 network. Figure 5 shows the comparison of our method (this work) with the standard linear SGD algorithm (ideal performance) and nonlinear SGD algorithm (baseline), in which the $k$-value of conv1, conv2, FC1, FC2, and FC3 layer is 1, 1, 0.01, 0.1, and 0.5, respectively, and $G_{\text{max}} = 2.2$, $G_{\text{max}} = 1$, $G_{\text{min}} = G_{\text{min}} = 0$, $P_{\text{max}} = 100$, $A_{p1} = A_{p2} = 1$, $A_{d1} = A_{d2} = -1$, and batch_size = 128. Our method (red dots) shows comparable high accuracy with the standard linear SGD algorithm (gray dots) and meanwhile is higher than the nonlinear SGD algorithm (blue dots).

An adaptive learning system is an automatic control system that preserves its operational capability under conditions of unforeseen change in the properties of the controlled system. Based on the extremely simple updating algorithm, our network is able to continuously adjust the parameters and exhibit great self-optimization capabilities. Remarkably, it exhibits particular good performance at small training epoch. For example, the training accuracy can reach as high as about 85% at iteration = 50 (corresponding to only 6400 training images), which means the system is able to learn useful knowledge at small numbers of training samples, confirming the high performance of our stochastic-adaptive learning system.\(^{112}\)

Next, more detailed information on network training has been studied using “mode-1.” Figure 6 shows the network accuracy on different $G_{\text{max}}$, $G_{\text{min}}$, nonlinearity of LTP, and non-linearity of LTD. When the $G_{\text{max}}$ of $w_2$ is fixed at a normalized value of 1, the best $G_{\text{max}}$ of $w_1$ is checked to be around 2, and best $G_{\text{min}}$ ranges from 0 to 0.1, which means the on/off ratio of the original memristor is about 20 and can be easily achieved in a practical device. In addition, when the nonlinearity of [LTP] and [LTD] is less than 2, the recognition accuracy can reach about 90%, indicating good network performance.\(^{10,12,19,25}\)

The results on the other modes are shown in Figure 7. “Mode-1” exhibits the best performance, and “mode-2” exhibits worse performance than “mode-1” because of the excessive amount of compensation in updating, which is similar to the large learning rate in SGD algorithm. “Mode-3” and “mode-4” are designed to reduce the circuit area; however, the second weight ($w_2$) in the differential-pair is untouched in weight updating. Therefore, their performance is also less than that of “mode-1.” It should be noticed that mode-4 has a higher accuracy than mode-3. This is mainly because mode-4 has uniform and symmetric weight distribution than that of mode-3. Set $G_{\text{max}} = 1$ and $G_{\text{min}} = 0$, and the weight states number is 11. In mode-4, $w_2 = (G_{\text{max}} + G_{\text{min}})/2 = 0.5$; thus, the weight value of each differential-pair ($W = w_1 - w_2$) can be either negative or positive (weight set: $W = -0.5$, $-0.4$, $-0.3$, $-0.2$, $-0.1$, $0.1$, $0.2$, $0.3$, $0.4$, $0.5$). However in mode-3, the distribution of $W$ is usually asymmetric. For example, if $w_2$ is (randomly) initialized to be $w_2 = G_{\text{max}}$ (or $G_{\text{min}}$), the weight value of the differential-pair ($W = w_1 - w_2$) will be always negative (or positive), which is to the disadvantage of high-accuracy on-line learning.

Finally, the network performance with different training methods is shown in Table 3, where all the listed data represent the average result obtained from ten times of duplicated tests with random weight initialization conditions for each mode. The proposed stochastic-adaptive learning method exhibits much better performance than that of nonlinear SGD algorithm and is close to the network with complex updating method, such as the PL method. The comparison of hardware implementation methods for memristor network training using different updating algorithms is shown in Table 4. Our method is the best one with great hardware-friendly features.
and the on the network performance trained with $G = 6$. The best performance occurs at an on/off ratio of about 20 or more, corresponding to the nonlinearity method, in which $p$ is the variable on the vertical axis for the CNNs, and four different modes have been proposed to either improve the performance or to reduce the chip area. In the proposed learning method, only the updating direction provided by the learning algorithm is required, and the complex calculation of the specific conductance variations, fine-tuning of the pulse amplitude and pulse duration, specific numbers of electrical pulse are all exempted, thus

5. Conclusion

In summary, we developed a stochastic and adaptive learning method to train the memristor CNNs, and four different modes have been proposed to either improve the performance or to

Figure 6. The influence of $G_{max}$ and $G_{min}$ on the network performance trained with “mode-1” method, in which $G_{max1}$ is the variable on the horizontal axis and $G_{min2}$ is the variable on the vertical axis for the figure. In each panel, there are 49 results corresponding to the 49 combinations of $A_p$ (nonlinearity varies from 0 to 6) and $A_d$ (nonlinearity varies from 0 to −6). The best performance occurs at an on/off ratio of about 20 or more, corresponding to the panel a2, a3, b2, and b3. Note, $G_{max1}:G_{min1} = G_{max2}:G_{min2}$, and $G_{max2} = 1$. 

5. Conclusion

In summary, we developed a stochastic and adaptive learning method to train the memristor CNNs, and four different modes have been proposed to either improve the performance or to reduce the chip area. In the proposed learning method, only the updating direction provided by the learning algorithm is required, and the complex calculation of the specific conductance variations, fine-tuning of the pulse amplitude and pulse duration, specific numbers of electrical pulse are all exempted, thus
Figure 7. a–d) Accuracy of LeNet-5 network trained with different modes at various nonlinearities of $A_p$ and $A_d$. For mode-1 and mode-2, $G_{max1} = 2.2$, $G_{min1} = 0$, $G_{max2} = 1$, and $G_{min2} = 0$. For mode-3 and mode-4, $G_{max} = 1$ and $G_{min} = 0$. “Mode-1” (to improve the performance) and “Mode-4” (to reduce the area) exhibit good performance at all $A_p/A_d$ conditions. e) The comparison table for different modes at the typical nonlinear conditions.

Table 3. Comparison of the accuracy (average of ten duplicated tests) with different updating algorithms.

| algorithm          | Memristor network | Accuracy (%) at different nonlinearities ($A_p$, $A_d$) |
|--------------------|-------------------|-------------------------------------------------------|
|                    |                   | (1,−1) | (2,−2) | (3,−3) | (4,−4) | (5,−5) | (6,−6) |
| Nonlinear SGD method | LeNet-5           | 90.14  | 84.30  | 81.19  | 78.50  | 77.75  | 77.22  |
| This work          | LeNet-5 mode-1    | 93.88% | 91.16% | 89.01% | 87.87% | 85.43% | 85.94% |
| Nonlinear PL method | ANN 784-100-10    | 94.7   | 92.7   | 90.7   | 88.9   | 88.0   | 88.3   |

Table 4. Comparison of the hardware implementation with different updating methods.

| Reference                                      | [10] | [12] | [16] | [19] | [25] | This work |
|------------------------------------------------|------|------|------|------|------|-----------|
| Without precise read before writing           | ✓    | ✓    | ×    | ✓    | ✓    | ✓         |
| Without always change in pulse amplitude      | ✓    | ×    | ✓    | ✓    | ✓    | ✓         |
| Without always change in pulse duration       | ×    | ✓    | ×    | ✓    | ✓    | ✓         |
| Without calculation of pulse number           | ✓    | ✓    | ×    | ×    | ×    | ✓         |
| Nonlinearity almost disappears                 | ✓    | ✓    | ✓    | ✓    | ×    | ✓         |
making the method highly hardware friendly and energy efficient. In addition, the probability updating algorithm has been intrinsically embedded into our method, in which all the nonlinearity and stochasticity updating can also be conducted by the memristor network itself; therefore, it is fully hardware friendly without complex peripheral circuits when compared with the traditional nonlinear SGD algorithm or the PL algorithm. The effectiveness of the proposed method has been carefully investigated on training of a classical LeNet-5 CNN. The demonstrated network exhibits high accuracy, about 93.88% (statistical data), on the MNIST test, which is close to the network with a complex updating method, such as the PL method (94.7%), and is higher than the original nonlinear SGD algorithm (90.14%).

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

W.Z. and L.P. contributed equally to this work. This work was supported by Shenzhen Science and Technology Innovation Committee, JCYJ20200109115210307, KQTD20170810160424889, RCYX2020071414733204, and JCYJ20170818155813437, Guangdong Basic and Applied Basic Research Foundation, 2019A1515111142; Guangdong Provincial Key Laboratory Program (2021B1212040001); and the Open Research Fund of Key Laboratory of Polar Materials and Devices, Ministry of Education. The Supporting Information of this article can be found here: https://authorea.com doi/10.22541/au.161522093.39767392/v1.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

Research data are not shared.

Keywords

convolutional neural networks, memristors, weight updating

Received: March 1, 2021
Revised: May 31, 2021
Published online: July 5, 2021

[1] Y. LeCun, Y. Bengio, G. Hinton, Nature 2015, 521, 436.
[2] A. Krizhevsky, I. Sutskever, G. E. Hinton, Adv. Neural Inform. Proc. Syst. 2012, 25, 1097.
[3] A. Krizhevsky, I. Sutskever, G. E. Hinton, Commun. ACM 2017, 60, 84.
[4] M. Horowitz, in IEEE Int. Solid-State Circuits Conf. Digest of Technical Papers (ISSCC), IEEE, Piscataway, NJ 2014, pp. 10–14.
[5] L. Chua, IEEE Trans. Circuit Theory 1971, 18, 507.
[6] D. B. Strukov, G. S. Snider, D. R. Stewart, R. S. Williams, Nature 2008, 453, 80.
[7] J. Tang, F. Yuan, X. Shen, Z. Wang, M. Rao, Y. He, Y. Sun, X. Li, W. Zhang, Y. Li, B. Gao, H. Qian, G. Bi, S. Song, J. Yang, H. Wu, Adv. Mater. 2019, 31, 1902761.
[8] F. Albart, E. Zamanidoost, D. B. Strukov, Nat. Commun. 2013, 4, 1.
[9] M. Prezioso, F. Merrikk-Bayat, B. Hoskins, G. C. Adam, K. K. Likharev, D. B. Strukov, Nature 2015, 521, 61.
[10] P. Chen, B. Lin, J. Wang, T. Hou, J. Ye, S. Vrudhula, J. Seo, Y. Cao, S. Yu, in IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD), IEEE, Piscataway, NJ 2015, pp. 194–199.
[11] P. Yao, H. Wu, B. Gao, S. Eryilmaz, X. Huang, W. Zhang, Q. Zhang, N. Deng, L. Shi, H.-S. Philip Wong, H. Qian, Nat. Commun. 2017, 8, 1.
[12] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, W. Song, J. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. Yang, X. Qia, Nat. Commun. 2018, 9, 1.
[13] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, Y. Jiang, H. Qian, Nature 2020, 577, 641.
[14] Y. Zhang, X. Wang, E. G. Friedman, IEEE Trans. Circuits Syst. I: Reg. Papers 2017, 65, 677.
[15] Z. Wang, C. Li, P. Lin, M. Rao, Y. Nie, W. Song, Q. Qiu, Y. Li, P. Yan, J. Strachan, N. Ge, N. McDonald, Q. Wu, M. Hu, H. Wu, R. S. Williams, X. Qia, J. Yang, Nat. Mach. Intell. 2019, 1, 434.
[16] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Williams, J. Yang, X. Qia, Nat. Electron. 2018, 1, 52.
[17] C. Li, Z. Wang, M. Rao, D. Belkin, W. Song, H. Jiang, P. Yan, Y. Li, P. Lin, M. Hu, N. Ge, J. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. Yang, X. Qia, Nat. Mach. Intell. 2019, 1, 49.
[18] M. Huang, G. Zhao, X. Wang, W. Zhang, P. Coquet, B. K. Tay, G. Zhong, J. Li, IEEE Electron Device Lett. 2020, 42, 106.
[19] J. Fu, Z. Liao, N. Gong, J. Wang, IEEE J. Emerg. Sel. Top. Circuits Syst. 2019, 9, 377.
[20] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Proc. IEEE 1998, 86, 2278.
[21] A. Krizhevsky, G. Hinton, Handbook of Systemic Autoimmune Diseases 2009, Vol. 1, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.9220&rep=rep1&type=pdf.
[22] K. He, X. Zhang, S. Ren, J. Sun, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, IEEE, Piscataway, NJ 2016, pp. 770–778.
[23] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, P. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Nature 2016, 529, 484.
[24] G. Zhong, M. Zi, C. Ren, Q. Xiao, M. Tang, L. Wei, F. An, S. Xie, J. Wang, X. Zhong, M. Huang, J. Li, Appl. Phys. Lett. 2020, 117, 092903.
[25] I.-T. Wang, C.-C. Chang, L.-W. Chiu, T. Chou, T.-H. Hou, Nanotechnology 2016, 27, 365204.
[26] Y. Zhang, M. Cui, Y. Liu, L. Shen, Neurocomputing 2019, 355, 48.