Adjusting Learning Rate of Memristor-Based Multilayer Neural Networks via Fuzzy Method

Shiping Wen, *Member, IEEE*, Shuixin Xiao, Yin Yang, *Member, IEEE*, Zheng Yan, *Member, IEEE*, Zhigang Zeng, *Senior Member, IEEE*, and Tingwen Huang, *Senior Member, IEEE*

***Abstract*—Back propagation (BP) based on stochastic gradient descent is the prevailing method to train multilayer neural networks (MNNs) with hidden layers. However, the existence of the physical separation between memory arrays and arithmetic module makes it inefficient and ineffective to implement BP in conventional digital hardware. Although CMOS may alleviate some problems of the hardware implementation of MNNs, synapses based on CMOS cost too much power and areas in very large scale integrated circuits. As a novel device, memristor shows promises to overcome this shortcoming due to its ability to closely integrate processing and memory. This paper proposes a novel circuit for implementing a synapse based on a memristor and two MOSFET tansistors (p-type and n-type). Compared with a CMOS-only circuit, the proposed one reduced the area consumption by 92%–98%. In addition, we develop a fuzzy method for the adjustment of the learning rates of MNNs, which increases the learning accuracy by 2%–3% compared with a constant learning rate. Meanwhile, the fuzzy adjustment method is robust and insensitive to parameter changes due to the approximate reasoning. Furthermore, the proposed methods can be extended to memristor-based multilayer convolutional neural network for complex tasks. The novel architecture behaves in a human-liking thinking process.**

Manuscript received October 2, 2017; revised January 7, 2018; accepted April 3, 2018. Date of publication May 9, 2018; date of current version May 20, 2019. This work was supported in part by the Natural Science Foundation of China under Grant 61673187 and Grant 61673188, in part by the Shenzhen Innovative from the Research Institute in Shenzhen, Huazhong University of Science and Technology under Grant JCYJ20170307160806070, and in part by NPRP through the Qatar National Research Fund (a member of Qatar Foundation) under Grant NPRP 8-274-2-107. This paper was recommended by Associate Editor Y. Chen. *(Corresponding author: Tingwen Huang.)*

S. Wen is with the School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China, also with the Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Research Institute in Shenzhen, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: wenshiping226@hust.edu.cn).

S. Xiao and Z. Zeng are with the School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, Huazhong University of Science and Technology, Wuhan 430074, China.

Y. Yang is with the College of Science, Engineering, and Technology, Hamad Bin Khalifa University, Doha, Qatar (e-mail: yyang@hbku.edu.qa).

Z. Yan is with the Centre for Artificial Intelligence, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: yan.zheng@uts.edu.au).

T. Huang is with the Science Program, Texas A&M University at Qatar, Doha, Qatar (e-mail: tingwen.huang@qatar.tamu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2018.2834436

***Index Terms*—Fuzzy adjustment, learning rate, memristor, multilayer neural network (MNN).**

I. Introduction

MULTILAYER neural networks (MNNs) are powerful and popular artificial neural networks (ANNs) that are competent in learning, computing, and intelligent signal processing. Therefore, MNNs have been used in numerous fields, such as cloud computing [1]–[3] and machine learning [4]. In MNNs, the direction of information processing is from the input layer to the hidden layer, and so on until to the output layer. Each neuron takes a weighted sum of the outputs of the pervious layer as an input, and it produces an output to feed to the next layer. It was difficult to train such neural networks due to the hidden layers. With an efficient implementation using stochastic gradient descent (SGD), back propagation (BP) algorithm [5] has facilitated the training of MNNs (especially in large-scale machine learning [6]).

The power of MNNs mainly stems from the BP learning algorithm that uses the chain rule to compute gradients and update the synaptic weights. However, it is ineffective to implement BP in conventional digital hardware due to the physical separation between the memory arrays and arithmetic module, which are, respectively, used to store the value of the synaptic weight and calculate the update rules. To overcome this shortcoming, a great number of CMOS-based hardware design technologies [7] have been proposed in the past few years. Most of them utilize parallel synaptic to store synaptic weights and update them for learning tasks in MNNs. However, CMOS synapses need much more power and areas in very large scale integrated (VLSI) circuits [8].

As a novel device, memristor makes it possible to implement ANNs and MNNs in hardware [9]–[20]. Memristor was predicted by Chua in 1971 [15] with the theory of memristor system [21]. In May 2008, Hewlett-Packard (HP) lab realized the first physical memristor using physics nano-technology. With the memory function, memristor becomes an ideal material for registering and updating the synaptic weights [22]. Moreover, the nano-scale size makes memristor form a compact and efficient architecture for learning algorithms and other neural-network-related applications. And memristor has been used in cellular nonlinear neural network [23], the design, analysis, and applications of which are novel. The application to image processing in memristive multilayer cellular neural network has also been a great success [24].

Most previous implementations of learning rules using memristor crossbar have been limited in spiking neurons and spike-timing-dependent plasticity which can be used to explain biological neuroscience results [25]–[32]. Other learning systems, which rely on memristor crossbar, are only used in single-layer neural network (SNN) [33]–[35]. Recently, memristor crossbar was designed to train SNNs with binary output on VLSI [36], [37]. However, it is difficult to implement MNNs for the scalable training of MNNs along with BP algorithm [38]. In the synaptic array circuit, all learning rules require a multiplicative term [39]. To solve this problem, a novel design was proposed with a memristor and two MOSFET transistor(p-type and n-type) in a synapse [40]. In this circuit, the memristor was used to store the weight in order to realize the multiplication operation. Compared to CMOS-only circuit, it only requires $2\%-8\%$ of the area.

Current research commonly uses a constant learning rate in BP [39]. However, the learning rate is a hyper-parameter whose design and tuning are significant for the training performance. To design this hyper-parameter, the conventional way is based on Rules of Thumb to statistical characteristics. Some optimization methods are considered, such as heuristic simulated annealing [41] and momentum factor [42]. In 1996, Hayjin reached the following conclusion: if learning rate satisfies $0 < \eta < (2/\lambda_{\max})$, the SGD algorithm is convergent by variance, where $\lambda_{\max}$ is the maximum eigenvalue of the input vector autocorrelation matrix. However, we cannot get this if the input vector is very large. Darken and Moody [43] proposed a search-then-converge scheme in which the learning rate decreased gradually as the number of iterations increased. However, it does not follow real neuronal learning process. Duchi *et al.* [44], [45] proposed AdaGrad, and Zeiler *et al.* [46], [47] proposed AdaDelta to adjust gradient rather than weight. If the initial value of weight is too large, the penalty of the initial gradient will be very little. The basic idea of AdaDelta is to approximate the two-order Newton method with a first-order method. From the case of multiple data sets, AdaDelta has very good acceleration in the early and middle stages of training. However, at the end of the training period, AdaDelta is prone to be dithered near the local minimum.

In this paper, we propose the fuzzy method to dynamically adjust the learning rate, which is different from algorithmic improvements mentioned above. It is more concerned about hardware implementation as fuzzy method has been realized in hardware, such as field-programmable gate array (FPGA) and PLC [48]–[50], which translate the natural language strategy of people into the algorithm language that computer can recognize. By simulating the way people think, it could adjust the objects, which could not be constructed by mathematical model. In 1965, Zadeh [51] first proposed membership to describe fuzziness of things. Mamdani and Assilian [52] successfully utilized fuzzy control in steam engine control. Recently, fuzzy control has been used in Web-based service [53] and fault-tolerant computing [54]. The advantages of fuzzy algorithm can be described as follows.

1) The fuzzy system is robust and insensitive to changes in process parameters [55], as well as suitable for solving the problems of nonlinearity, time variation and hysteresis which are difficult to be solved by conventional methods.
2) Fuzzy algorithm utilizes the approximate reasoning [56], which imitates human thinking process to deal with complex and even morbid system with involved human experience. Therefore, it is suitable to use fuzzy algorithm to adjust the learning rate in MNNs.

This paper is organized as follows. In Section II, we give a brief introduction about memristor and SGD. In Section III, a conventional fuzzy controller is described. We define its inputs and outputs. Then we analyze its structure and results. Section IV presents our proposed fuzzy adjustment algorithm and proof of the convergence of the algorithms. In Section V, we propose two-layer MNN to classify the Iris dataset and five-layer memristor-based convolutional neural network to classify the MNIST dataset with the proposed algorithm. Then we express the conclusion and analyze the result.

II. Memristor and Stochastic Gradient Descent

A. Memristor

As well-known, there are four basic circuit variables in the circuit domain. They are voltage $v(t)$, current $i(t)$, flux $\phi(t)$, and charge $q(t)$. The above four circuit variables should correspond to six mathematical relationships. $f(q, i)$ is the relationship between charge and electric current. $f(\phi, v)$ corresponds to Faraday's law of electromagnetic induction. The other three variable relationships correspond to three conventional two port circuit components resistance R, capacitance C, and inductance L. Until 1971, Chua predicted that the relationship between charge q and flux ϕ can be described as a conventional two port circuit components which was named as memristor. In 2008, the Information and Quantum Systems Laboratory in HP found that a variety of switching materials had a $v - i$ hysteresis curve. The leader Wiliams and other researchers carried on thorough research on bi-level titanium dioxide. They found its electrical property fills well with memristor. According to Chua's definition on memristor, the relationship between flux and charge can be described as

$$f(\phi, q) = 0 \tag{1}$$

and

$$\phi(t) = \int_{-\infty}^{t} v(\tau)d\tau$$
$$q(t) = \int_{-\infty}^{t} i(\tau)d\tau.$$

It can also be described as a relationship curve in $\phi-q$ zone. If the curve has no relationship with connection mode of outside network, the curve reflects essential characteristics of memristor. If flux is the single valued function of charge, memristor can be regarded as flux control. Thus, we can get

$$\phi = f(q). \tag{2}$$

Differentiating (2)

$$\frac{d\phi}{dt} = \frac{df(q)}{dq} \cdot \frac{dq}{dt}. \tag{3}$$

According to relationship between voltage and magnetic flux $v = (d\phi/dt)$ as well as the relationship between electricity and

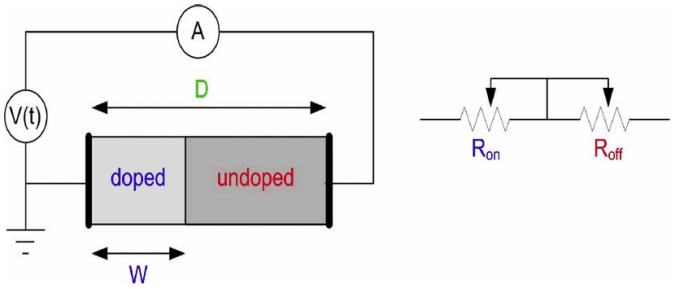


Fig. 1. Structure of TiO_2 memristor.

charge $i = (dq/dt)$, we get the relationship between voltage and current in memristor

$$v(t) = M(q)i(t) \tag{4}$$

where $M(q)$ reflects the slope of the tangent of the static working point on the $q-\phi$ relation line. As $M(q)$ has dimension of resistance, it is called small signal in the static working point in memristor with charge control and it can be described as

$$M(q) = \frac{df(q)}{dq}. \tag{5}$$

In view of the above discussion, we can find that at a fixed moment, the value of memristor depends on the integration of memristor current at a period time $-\infty < t \leq t^0$. Therefore, memristor can reflect the resistance characteristics at a fixed moment t_0. Meanwhile, memristor also has memory function as its resistance value has relationship with the current passes through it, which proves memristor as the fourth basic circuit component. In 2008, HP proposed memristor model TiO_2, which is the ideal material for memristor and the effect is similar to theoretical model. Therefore, we choose the HP model of memristor. HP's TiO_2 memristor is mainly composed of a metal platinum oxide electrode arranged between the metal platinum electrode and the electrode at both ends. We assume that the minimum value of memristor is R_{ON} and the maximum value of memristor is R_{OFF}. The value of memristor can be described as (Fig. 1)

$$M(t) = R_{\mathrm{ON}}\frac{w(t)}{D} + R_{\mathrm{OFF}}\left(1 - \frac{w(t)}{D}\right) \tag{6}$$

where $w(t)$ is thickness of the doped layer, D is the thickness of the entire semiconductor film, $0 \leq [w(t)/D] \leq 1$ is the internal state control variable of memristor.

B. Stochastic Gradient Descent

Assume $X_k = (X_1^T(k), X_2^T(k), \ldots, X_R^T(k))^T$ as the input of the network, $d_k = (d_1^T(k), d_2^T(k), \ldots, d_s^T(k))^T$ as the desired output of the network and $r_k = (r_1^T(k), r_2^T(k), \ldots, r_s^T(k))^T$ as the actual output of the network, where $k = 1, 2, \ldots, m$ are related to the sampled pair of input vector and desired output vector of the network. By calculating the error between actual output vector and desired output vector to adjust network weights and thresholds, we can gradually reduce the error. Meanwhile SGD usually adopts LMS learning rules, which try to reduce the mean square sum of these errors. Defining mean square sum of errors (MSE) as follows:

$$\mathrm{MSE} = \frac{1}{m}\sum_{k=1}^{m} e^2(k) = \frac{1}{m}\sum_{k=1}^{m}(d(k) - r(k))^2. \tag{7}$$

Taking the partial derivatives of the MSE function with respect to network weights and thresholds, we obtain

$$\frac{\partial e^2(k)}{\partial \omega_{i,j}} = 2e(k)\frac{\partial e(k)}{\partial \omega_{i,j}} \tag{8}$$

where $j = 1, 2, \ldots, s$ and

$$\frac{\partial e^2(k)}{\partial b} = 2e(k)\frac{\partial e(k)}{\partial b}. \tag{9}$$

Then we can calculate the partial derivatives of the error with respect to network weights and thresholds

$$\frac{\partial e(k)}{\partial \omega_{i,j}} = \frac{\partial [d(k) - r(k)]}{\partial \omega_{i,j}} = \frac{\partial e}{\partial \omega_{i,j}}[d(k) - (W_X(k) + b)] \tag{10}$$

or

$$\frac{\partial e(k)}{\partial \omega_{i,j}} = \frac{\partial [d(k) - r(k)]}{\partial \omega_{i,j}} = \frac{\partial e}{\partial \omega_{i,j}}\left[d(k) - \left(\sum_{i=1}^{R}\omega_{i,j}X_i(k) + b\right)\right] \tag{11}$$

where $p_i(k)$ expresses ith input vector in the kth circle. Then

$$\begin{aligned}\frac{\partial e}{\partial \omega_{i,j}} &= -X_i(k)\\ \frac{\partial e(k)}{\partial b} &= -1.\end{aligned} \tag{12}$$

According to the gradient descent principle, the changed values of network weights and thresholds are $2\eta e(k)p(k)$ and $2\eta e(k)$, respectively. Therefore, we can get the regulating formula

$$\begin{aligned}\omega(k+1) &= w(k) + 2\eta e(k)X^T(k)\\ b(k+1) &= b(k) + 2\eta e(k)\end{aligned} \tag{13}$$

where η is the learning rate. When η is suitable, it can accelerate training speed of the network. However, when η is too big or too small, it hinders convergence. Therefore, it is important to choose an appropriate value for η. In this paper, we proposes a fuzzy controller dynamically adjusting η to accelerate training speed of the network on the basis of decreasing the training error.

III. Fuzzy Adjusting

Assume an object with single input and single output and use 2-D fuzzy controller to control it. To simplify the description, we set system control variable $u(t)$, system output $y(t)$, reference input $s(t)$, discrete time variable t, and sampling period T. Assuming the inputs of the fuzzy controller are error $e(t)$ and difference error $r(t)$. So $e(t) = s(t) - y(t)$, $r(t) = e(t) - e(t-1)$ and output is $u(t)$.

First, we introduce normalization factor and scaling factor Ge, Gr, Gu to normalize the output and input of the fuzzy controller. Then

$$e^* = Ge \cdot e(t) \tag{14}$$

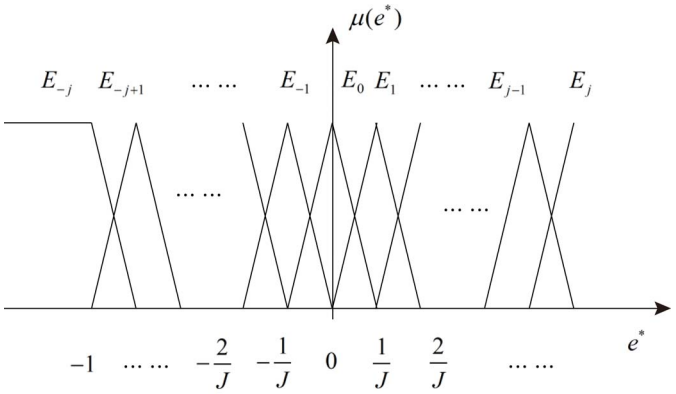


Fig. 2. Membership of e^*.

$$r^* = Gr \cdot r(t) \tag{15}$$

$$u(t) = Gu \cdot u^* \tag{16}$$

where $e^*, r^* \in [-1, 1], u^* \in [-4, 4]$.

A. Fuzzy Parameters

The fuzzy parameters of typical fuzzy controllers are described in the following.

1) Output and input variables $e^*(t), r^*(t), u^*(t)$ are symmetrical, uniform distribution, full fold, triangle membership functions. Uniform distribution means that the central points of each triangle are evenly distributed over the field. Full fold means that the end point of each triangle bottom is exactly the center point of the two adjacent triangles. Considering commonly Mamdani fuzzy controller, we assume the input variables e^*, r^* both have $N = 2 * J + 1$ fuzzy numbers, among of which J fuzzy numbers are negative numbers and J fuzzy numbers are positive numbers and one number is zero. We use E_i, R_j to express them. We assume that the output of the fuzzy controller has $2N - 1 = 4J + 1$ fuzzy numbers and we use U_k to express them. The expression of the input membership function is shown (in Fig. 2) as follows:

$$\mu(e^*) = \begin{cases} \frac{1}{S}\left[e^* - (i-1)S\right], & e^* \in [(i-1)S, iS] \\ -\frac{1}{S}\left[e^* - (i+1)S\right], & e^* \in (iS, (i+1)S] \\ 0, & e^* \notin [(i-1)S, (i+1)S] \end{cases} \tag{17}$$

$$\mu(r^*) = \begin{cases} \frac{1}{S}\left[r^* - (j-1)S\right], & r^* \in [(j-1)S, jS] \\ -\frac{1}{S}\left[r^* - (j+1)S\right], & r^* \in (jS, (j+1)S] \\ 0, & r^* \notin [(j-1)S, (j+1)S] \end{cases} \tag{18}$$

where $S = (1/J)$, $J = [(N-1)/2]$. The expression of the output membership function is presented as follows:

$$\mu\left(u^*\right) = \begin{cases} \frac{1}{S}\left[u^* - (k-1)S\right], & u^* \in [(k-1)S, kS] \\ -\frac{1}{S}\left[u^* - (k+1)S\right], & u^* \in (kS, (k+1)S] \\ 0, & u^* \notin [(j-1)S, (j+1)S]. \end{cases} \tag{19}$$

2) Linear fuzzy control rules. As e^* and r^* have N fuzzy numbers, respectively, we have $N * N$ fuzzy control rules as follows. IF e^* is E_i and r^* is R_j, THEN u^* is U_k. Linear fuzzy control rules mean that output fuzzy numbers have linear relationship as

$$U_k = E_i + R_j, k = i + j.$$

To express the linear relationship between conditions and conclusions, we get

$$U_k = U_{i+j} = U_{i,j}.$$

3) Fuzzy reasoning process. The reasoning process includes: AND operation among input variables adopts min operator; OR operation among different rules adopts max operator; fuzzy implication operation, which means the reasoning process achieves output fuzzy set via rules and input variables. We adopt operations of isosceles fuzzy numbers with weights to obtain the following results:

$$U_{i,j} = \frac{\sum_{i,j=1}^{N} \omega(i,j) \cdot U_k}{\sum_{i,j=1}^{N} \omega(i,j)} = [e, f], \tag{20}$$

$$\mu\left(u^*\right) = I^{\lambda}[e,f] = \begin{cases} \frac{1}{2S}\left[u^* - e\right], & u^* \in \left[e, \frac{e+f}{2}\right] \\ \frac{1}{2S}\left[u^* - f\right], & u^* \in \left(\frac{e+f}{2}, f\right] \end{cases} \tag{21}$$

where $\lambda = 1$, $\omega(i,j) = \min\{\mu_i(e^*), \mu_j(r^*)\}$ is the activation of rules, U_k is the output fuzzy number of linear rules.

4) The centroid method is used to solve the ambiguity as follows:

$$\frac{\int_e^f u^* \cdot \mu(u^*) du^*}{\int_e^f \mu(u^*) du^*}. \tag{22}$$

B. Structural Analysis and Analysis Process

At any time t, we get the output of system $y(t)$ as well as the error between output variables and setting value. The input variable is $e(t)$. Then, we can calculate the difference $r(t) = e(t) - e(t-1)$ and normalize them. Thus, we get e^*, r^*. After that, fuzzy transformation is utilized. Therefore, there exist i, j: $-J \le i, j \le J-1$, which satisfy $iS \le e^* \le (i+1)S$ and $e^* \in [E_i, E_{i+1}], r^* \in [R_i, R_{i+1}]$ as follows:

$$\begin{aligned} \mu_i\left(e^*\right) &= -\frac{1}{S}\left[e^* - (i+1)S\right]; \mu_{i+1}\left(e^*\right) = \frac{1}{S}\left[e^* - iS\right] \\ \mu_j\left(r^*\right) &= -\frac{1}{S}\left[r^* - (j+1)S\right]; \mu_{j+1}\left(r^*\right) = \frac{1}{S}\left[r^* - jS\right]. \end{aligned} \tag{23}$$

The membership of e^*, r^* are 0 for other fuzzy numbers.

Thus, $\mu_i(e^*) + \mu_{i+1}(e^*) = 1, \mu_j(r^*) + \mu_{j+1}(r^*) = 1$. At this time, e^*, r^* only activate the following four rules:

R_1: IF $e^* = E_{i+1}$ and $r^* = R_{j+1}$, THEN $u^* = U_{i+j+2}$

R_2: IF $e^* = E_{i+1}$ and $r^* = R_j$, THEN $u^* = U_{i+j+1}$

R_3: IF $e^* = E_i$ and $r^* = R_{j+1}$, THEN $u^* = U_{i+j+1}$

R_4: IF $e^* = E_i$ and $r^* = R_j$, THEN $u^* = U_{i+j}$.

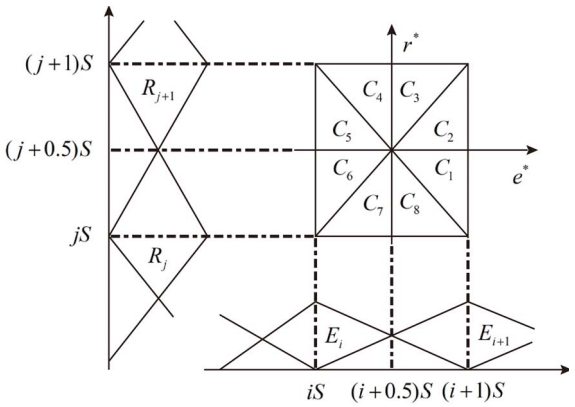


Fig. 3. Distribution of input variables e^*, r^*.

TABLE I
ACTIVATION DEGREES OF FOUR RULES

area	R1	R2	R3	R4	$\Sigma\omega(i,j)$
C_1, C_2	$\mu_{j+1}(r^*)$	$\mu_j(r^*)$	$\mu_i(e^*)$	$\mu_{i+1}(e^*)$	$\mu_{i+1}(e^*)$
C_3, C_4	$\mu_{j+1}(r^*)$	$\mu_{j+1}(r^*)$	$\mu_{j+1}(r^*)$	$\mu_{j+1}(r^*)$	$\mu_{i+1}(e^*)$
C_5, C_6	$\mu_{j+1}(r^*)$	$\mu_{j+1}(r^*)$	$\mu_{j+1}(r^*)$	$\mu_{j+1}(r^*)$	$\mu_{i+1}(e^*)$
C_7, C_8	$\mu_{j+1}(r^*)$	$\mu_{j+1}(r^*)$	$\mu_{j+1}(r^*)$	$\mu_{j+1}(r^*)$	$\mu_{i+1}(e^*)$

If we use a square area to express e^*, r^* and divide it into eight small areas (Fig. 3), the four rules' activation degrees $\omega(i,j) = \min\{\mu_i(e^*), \mu_j(r^*)\}$ are listed in Table I. As the results of R_2 and R_3 are both U_{i+j+1}, *OR* operator adopts *max* operation, we get the results in area C_1 C_8 as follows:

$$\begin{aligned}
&C_1, C_2\text{: } \max\{\mu_j(r^*), \mu_i(e^*)\} = \mu_j(r^*)\\
&C_3, C_4\text{: } \max\{\mu_j(r^*), \mu_i(e^*)\} = \mu_j(e^*)\\
&C_5, C_6\text{: } \max\{\mu_{j+1}(r^*), \mu_{i+1}(e^*)\} = \mu_{j+1}(r^*)\\
&C_7, C_8\text{: } \max\{\mu_{j+1}(r^*), \mu_{i+1}(e^*)\} = \mu_{i+1}(e^*).
\end{aligned}$$

After that, we get the sum activation degree of four rules $\sum\omega(i,j)$ in each area.

According to Table I, in areas C_1 and C_2, we use the addition of isosceles fuzzy numbers to lead to fuzzy reasoning as follows:

$$\begin{aligned}
U &= \frac{\sum_{i,j=1}^{N}\omega(i,j)\cdot U_k}{\sum_{i,j=1}^{N}\omega(i,j)}\\
&= \frac{\mu_{j+1}(r^*)U_{i+j+2} + \mu_j(r^*)U_{i+j+1} + \mu_i(e^*)U_{i+j}}{1+\mu_i(e^*)}. \qquad (24)
\end{aligned}$$

Then we get

$$\begin{aligned}
e = \frac{1}{1+\mu_i(e^*)}\Big\{&\mu_{j+1}(r^*)(i+j)S + \mu_j(r^*)\\
&\times (i+j-2)S + \mu_i(e^*)(i+j-2)S\Big\}. \qquad (25)
\end{aligned}$$

By substituting the membership function value

$$\begin{aligned}
e &= (i+j-1)S + S\cdot\frac{[r^*-(j+0.5)S] + [e^*-(i+0.5)S]}{\frac{3}{2}S - [e^*-(i+0.5)S]}\\
f &= e + 4S = (i+j+3)S\\
&\quad + S\cdot\frac{[r^*-(j+0.5)S] + [e^*-(i+0.5)S]}{\frac{3}{2}S - [e^*-(i+0.5)S]}.
\end{aligned}$$

Similarly, we get U in areas C_3 C_8. The forms are the same and the differences are in the form of e, f. So the output fuzzy numbers of the fuzzy controller can be calculated as follows:

$$\mu(u^*) = \begin{cases} \frac{1}{2S}[u^*-e], u^* \in \left[e, \frac{e+f}{2}\right] \\ \frac{1}{2S}[u^*-f], u^* \in \left(\frac{e+f}{2}, f\right]. \end{cases} \qquad (26)$$

We solve the fuzzy formula by substituting the center of gravity method

$$\begin{aligned}
u^* &= \frac{\int_e^f u^*\cdot\mu(u^*)du^*}{\int_e^f \mu(u^*)du^*}\\
&= \frac{\int_e^{\frac{e+f}{2}} t\cdot\frac{1}{2S}(t-e)dt + \int_{\frac{e+f}{2}}^f t\cdot\frac{-1}{2S}(t-f)dt}{\int_e^{\frac{e+f}{2}}\frac{1}{2S}(t-e)dt + \int_{\frac{e+f}{2}}^f\frac{-1}{2S}(t-f)dt}\\
&= \frac{\int_e^{\frac{e+f}{2}}(t^2-et)dt - \int_{\frac{e+f}{2}}^f(t^2-ft)dt}{\int_e^{\frac{e+f}{2}}(t-et)dt - \int_{\frac{e+f}{2}}^f(t-ft)dt}\\
&= \frac{e+f}{2}
\end{aligned}$$

which is equal to

$$u^* = \begin{cases} (i+j-1)S + S\cdot\frac{[r^*-(j+0.5)S]+[e^*-(i+0.5)S]}{\frac{3}{2}S-[e^*-(i+0.5)S]} \\ \qquad\times (C_1, C_2, C_5, C_6) \\ (i+j-1)S + S\cdot\frac{[r^*-(j+0.5)S]+[e^*-(i+0.5)S]}{\frac{3}{2}S-[e^*-(i+0.5)S]} \\ \qquad\times (C_3, C_4, C_7, C_8). \end{cases}$$

IV. PROPOSED STRUCTURE

The learning rate η is usually a constant ($0 < \eta < 1$). However, it is difficult to find the best learning rate at the initial time. We have to adjust η with 0.5, 0.1, 0.01,..., in order to find an appropriate learning rate. In this paper, memristor is employed to realize SGD. Meanwhile, the learning rate η is dynamically adjusted by the fuzzy controller.

A. Result Analysis

From the structural analysis, we get exact mathematical analytic relationship between the input variables of the fuzzy controller e^*, r^* and the output variable u^*. From above formula, we can find obvious mathematical laws among them. To be more exactly, u^* consists of two parts: the first part

$$\Phi_G(i,j) = (i+j+1)S = (i+0.5)S + (j+0.5)S$$

which is a 2-D multivalued relay. Its value has nothing to do with the change of the input variables e^*, r^*, but depends on i, j in their fuzzy subset. As the point $[(i+0.5)S, (j+0.5)S]$ is in the center of the area of input status (e^*, r^*), $\Phi_G(i,j)$ has global role and we call it global 2-D multivalued relay. When t is determined and i, j are constants, $\Phi_G(i,j)$is also a constant. The second part can be expressed as $\Phi_L(i,j) = K_P(e^*, r^*)[e^*-(i+0.5)S] + K_D(e^*, r^*)[r^*-(j+0.5)S]$, and

$$K_P(e^*, r^*) = K_D(e^*, r^*) = \begin{cases} \frac{S}{\frac{3}{2}-|e^*-(i+0.5)S|}(C_1, C_2, C_5, C_6) \\ \frac{S}{\frac{3}{2}-|r^*-(i+0.5)S|}(C_3, C_4, C_7, C_8). \end{cases}$$

Meanwhile, u^* also depends on the input (e^*, r^*) as well as its relative position with the center of the area $[(i + 0.5)S, (j + 0.5)S]$. This limits the influence on control variables. From its structure, it is a conventional PD controller. $K_P(e^*, r^*), K_D(e^*, r^*)$ are functions of input variables and change with (e^*, r^*). When the distance from input status (e^*, r^*) to central area $[(i + 0.5)S, (j + 0.5)S]$ is longer, the value of $K_P(e^*, r^*), K_D(e^*, r^*)$ will be bigger. Therefore, we regard it as a limited nonlinear time varying PD controller and it satisfies

$$\frac{1}{2} \le K_P(e^*, r^*) = K_D(e^*, r^*) \le 1.$$

From above analysis, we can conclude that if (e^*, r^*) are in the interval $[-1,1]$ to satisfy the designed values, the typical Mamdani fuzzy controllers analytic structure is consisted of a global 2-D multivalued relay and a limited nonlinear time-varying PD controller as follows:

$$\begin{aligned} u^* &= \Phi_G(i,j) + \Phi_L(i,j) \\ \Phi_G(i,j) &= \frac{2(i+j+1)}{N-1} \\ &= (i+j+1)S, (-J \le i,j \le J-1) \\ \Phi_L(i,j) &= K_P(e^*, r^*)\big[e^* - (i+0.5)S\big] \\ &\quad + K_D(e^*, r^*)\big[r^* - (j+0.5)S\big]. \end{aligned} \tag{27}$$

where $-J \le i, j \le J-1$.

B. Stability of Proposed Fuzzy Control

To control the learning rate, it is important to guarantee the stability of the fuzzy controlled system. No matter the objective of the controlled object is linear or nonlinear, the system balance point should satisfy $e^* = r^* = 0$. Therefore, there are four areas near the equilibrium point.

1) $e^* > 0, r^* > 0$, corresponding to $i = j = 0$, at this point

$$\begin{aligned} u^* &\approx S + \frac{S\big[(e^* - 0.5S) + (r^* - 0.5S)\big]}{2(S - 0.5S)} \\ &= S + \big[\big(e^* - 0.5S\big) + \big(r^* - 0.5S\big)\big] \\ &= e^* + r^*. \end{aligned}$$

2) $e^* > 0, r^* < 0$, corresponding to $i = 0, j = -1$, at this point

$$\begin{aligned} u^* &\approx 0 + \frac{S\big[(e^* - 0.5S) + (r^* + 0.5S)\big]}{2(S - 0.5S)} \\ &= \big[\big(e^* - 0.5S\big) + \big(r^* - 0.5S\big)\big] \\ &= e^* + r^*. \end{aligned}$$

3) $e^* < 0, r^* > 0$, corresponding to $i = -1, j = 0$, at this point

$$\begin{aligned} u^* &\approx 0 + \frac{S\big[(e^* - 0.5S) + (r^* - 0.5S)\big]}{2(S - 0.5S)} \\ &= \big[\big(e^* + 0.5S\big) + \big(r^* - 0.5S\big)\big] \\ &= e^* + r^*. \end{aligned}$$

4) $e^* < 0, r^* < 0$, corresponding to $i = j = -1$, at this point

$$\begin{aligned} u^* &\approx -S + \frac{S\big[(e^* - 0.5S) + (r^* - 0.5S)\big]}{2(S - 0.5S)} \\ &= -S + \big[\big(e^* + 0.5S\big) + \big(r^* + 0.5S\big)\big] \\ &= e^* + r^*. \end{aligned}$$

Then we can obtain that when $N \longrightarrow \infty$, $\Phi_L(i, j \longrightarrow 0)$. $\Phi_G(i,j)$ will be a linear PD controller $u^* = \Phi_G^\infty = e^* + r^*$, as

$$S = \frac{1}{J}, J = \frac{N-1}{2}$$

when $N \to \infty, S \to \infty, J \to \infty$, we have $\lim_{N \to \infty} J \to \infty, \lim_{N \to \infty} S = 0$ such that

$$\lim_{S \to 0} \Phi_G(i,j) = \lim_{S \to 0} \frac{S}{2} \times \frac{e^* + r^*}{-|e^*|} = 0.$$

Therefore,

$$\lim_{S \to 0} \Phi_G(i,j) = \lim_{J \to \infty} \frac{(i+j+1)}{J} = \lim_{J \to \infty} \left(\frac{i}{J} + \frac{j}{J}\right).$$

As $iS \le e^* \le (i+1)S, jS \le r^* \le (j+1)S$

$$\frac{i}{J} \le e^* \le \frac{i+1}{J}, \frac{i}{J} \le r^* \le \frac{j+1}{J}$$

and

$$\begin{aligned} \Phi_G^\infty &= \lim_{J \to \infty} \Phi_G(i,j) = e^* + r^* \\ \lim_{N \to \infty} u^* &= \lim_{N \to \infty} \big[\Phi_G(i,j) + \Phi_L(i,j)\big] = e^* + r^*. \end{aligned}$$

From the above discussion, when $N \longrightarrow \infty$, we can obtain that Mamdani fuzzy controller is a linear PD controller. Furthermore, we can also prove that the difference between the output of the fuzzy controller u^* and linear PD controller Φ_G^∞ is not greater than $\frac{S}{2}$ as follows:

$$\big|u^* - \Phi_G^\infty\big| \le \frac{S}{2}. \tag{28}$$

According to the Lyapunov stability theory, the asymptotic stability of the traditional Mamdani fuzzy control is necessary and sufficient condition for the asymptotic stability of the linear PD control system at the balance point $e^* = r^* = 0$.

From this result, it is obvious that fuzzy controller has relationship with the PID controller. When the number of the rules approaches to infinity, the fuzzy controller becomes a conventional PD control with the same Lyapunov stability principle. Therefore, we can draw the following conclusion: for the linear or nonlinear controlled object, which meets proposed design parameters, Mamdani fuzzy controller has the same stability properties as PD controller Φ_G^∞ at system balance point.

In this paper, we utilize fuzzy controller to adjust learning rate and it can be described as Fig. 4.

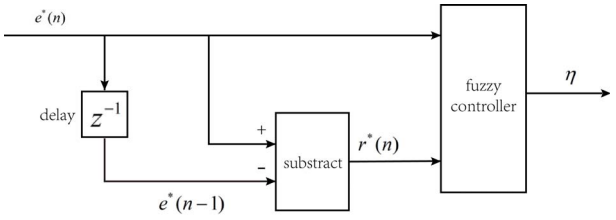


Fig. 4. Structure of fuzzy controller to control learning rate.

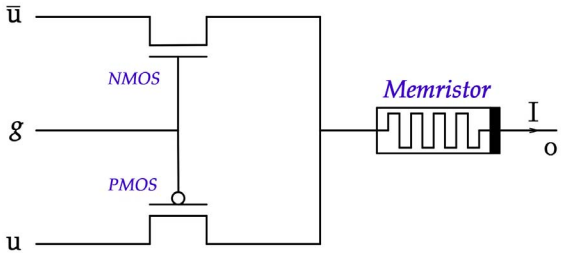


Fig. 5. Structure of artificial synapse.

C. Circuit Architecture

1) Artificial Synapse: This paper employs the structure which was proposed by Soudry *et al.* [40] with a single memristor and two MOSFET transistors (one p-type and the other n-type) as shown in Fig. 5.

When the n-type MOSFET transistor works in the linear region, we have

$$i_D = K_n\left[2(v_{\rm GS} - V_T)v_{\rm GS} - v_{\rm GS}^2\right] \tag{29}$$

where K_n is conductance constant of the n-type MOSFET transistor, V_T is the threshold voltage, $v_{\rm GS}$ is the drain-source voltage. When $v_{\rm GS} < V_T$, it works in cut-off area and $i_D = 0$.

Similarly, when the p-type MOSFET transistor works in the linear region, we have

$$i_D = -K_p\left[2(v_{\rm DS} - V_T)v_{\rm DS} - v_{\rm DS}^2\right] \tag{30}$$

where K_p is conductance constant of the p-type MOSFET transistor, V_T is the threshold voltage, $v_{\rm DS}$ is the drain-source voltage. When $v_{\rm DS} > -V_T$, it works in cut-off area and $i_D = 0$.

Then we assume that $K_n = K_p$ and they have the same threshold voltage V_T. We control the $u(t)$ that $-V_T < u(t) < V_T$. Thus, we have three possible cases.

1) When $g = 0$, the n-type MOSFET transistor and p-type MOSFET transistor both work in cut-off area and $i_D = 0$. The voltage does not pass memristor and the status of the single memristor is not changed.
2) When $g = V_{\rm DD}$, the n-type MOSFET transistor works in linear area while the p-type MOSFET transistor works in cut-off area. The voltage forward passes the memristor and the change value of the memristor positively increases.
3) When $g = -V_{\rm DD}$, the p-type MOSFET transistor works in linear area while the n-type MOSFET transistor works in cut-off area. The voltage reversely passes the memristor and the changed value of the memristor reversely increases.

Thus, we can use e to control the MOSFET transistors and the MOSFET transistors change the value of single memristor.

2) Cirurit Operation: In the proposed structure, memristor is utilized to store the synaptic weight. The sampled input x is converted to voltage u with $u = ax$, where a is a positive constant. To avoid that the MOSFET transistors work in cut-off area, at any time, $u < V_T$. So a should satisfy

$$a \cdot \max\{x\} < V_T. \tag{31}$$

In order to make the synaptic weight keep unchanged, the value of memristor need not to be changed (when the structure is under calculating). Therefore, we adopt that

$$g(t) = \begin{cases} V_{\rm DD} & \text{if} \quad 0 \le t \le 0.5T_R \\ -V_{\rm DD} & \text{if} \quad 0.5\ T_R < t \le T_R. \end{cases} \tag{32}$$

The changed value of memristor is

$$s = \int_0^{0.5T_R} (ax)dt + \int_{0.5T_R}^{T_R} (-ax)dt = 0 \tag{33}$$

where T_R is the length of the unchanged memristor's time.

In order to change the synaptic weight (when the structure is under updating), the value of memristor will be changed. We adopt

$$g(t) = \begin{cases} \text{sign}(e)V_{\rm DD} & \text{if} \quad 0 \le t - T_r \le b|e| \\ 0 & \text{if} \quad b|e| < t - T_r \le T_w \end{cases} \tag{34}$$

where T_w is the update time, b is a constant to convert error y to a time unit. Meanwhile

$$b \cdot \max\{e\} < T_w. \tag{35}$$

The changed value of memristor is

$$s = \int_{T_r}^{T_r + b|e|} (\text{asign}(e)x)dt = abxe. \tag{36}$$

Based on the above discussion, we can use the structure to achieve online learning based on LMS. Furthermore, if we use $N \times M$ artificial synapses, we can get multilayer neural networks (MNNs). However, we could not apply proposed circuit in DNN or CNN because they have different operation rules. But we might utilize memristor in other circuits to achieve DNN and CNN in the future.

V. Simulation and Result

A. Two-Layer MNN for Iris Classifying Task

In this section, we will use the data of the Iris classifying task, which is a well-known dataset in the pattern recognition literature with 3 classes of 50 instances in each class, where each class refers to a type of iris plant. One class is linearly separable from the other two; the latter are NOT linearly separable from each other. Attribute information includes sepal length(*cm*), sepal width(*cm*), petal length(*cm*), petal width(*cm*), and class. Then we predict the class of iris plant (Setosa Versicolour Virginica).

As the data is nonlinear, we use a two-layer MNN circuit. The neuronal activation function in the first hidden layer is given as follows:

$$f(x_i) = 1.7159 \tanh\left(\frac{2x_i}{3}\right). \tag{37}$$

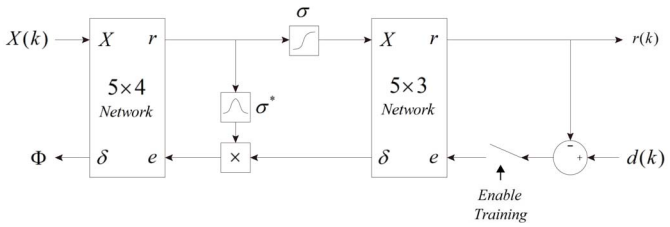


Fig. 6. Structure of two-layer MNN, where σ is the neuronal activation function.

TABLE II
ACCURACY RATE OF IRIS

number	$\eta = 0.1$	$\eta \in [0, 0.1]$
1	90.26	92.33
2	90.11	86.89
3	87.89	91.86
4	89.52	92.95
5	89.22	89.73
1	89.67	90.47
2	87.89	92.33
3	88.63	90.97
4	85.37	91.71
5	88.63	91.96
Maximum	90.26	92.33
Minimum	85.37	86.89
Average	88.95	91.50

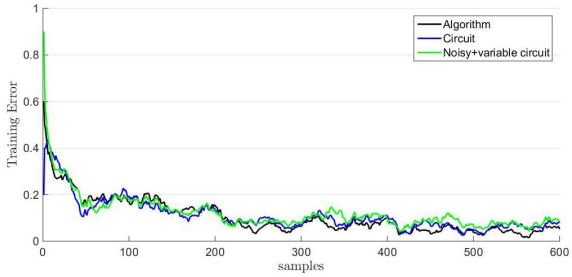


Fig. 7. Performances of the circuit without/with noise as well as the software implementation.

This function has been recommended in [5]. The two-layer MNN circuit is described with 5×4 layers and 5×3 layers as shown in Fig. 6. The results are presented as in Table II.

From Table II, it is obvious that the performances of the proposed circuit are better than that with constant learning rate in the maximum, minimum, average accuracy rate. The average accuracy rate is approximate 2.5% better than that with the constant learning rate. To illustrate the robustness of the proposed method, we also add noisy into the proposed circuit. In the training error, the performances of the circuits without/with noise and software implementation are shown in Fig. 7. It is obvious that similar final training errors are reached by these methods, which indicates that the circuit design can precisely implement MNNs with SGD with good robustness against noise.

B. Seven-Layer MNN for MNIST Dataset of Handwritten Digits

In this section, we will utilize the MNIST dataset of handwritten digits as shown in Fig. 8, which contains a training set of 60 000 examples, and a test set of 10 000 examples. Our utilized network consists of an input layer, two convolution layers, two max-pooling layers, and two fully connected layers with the proposed fuzzy adjusting learning rate $\eta \in$[0,1e–4].

Fig. 8. MNIST dataset of handwritten digits.

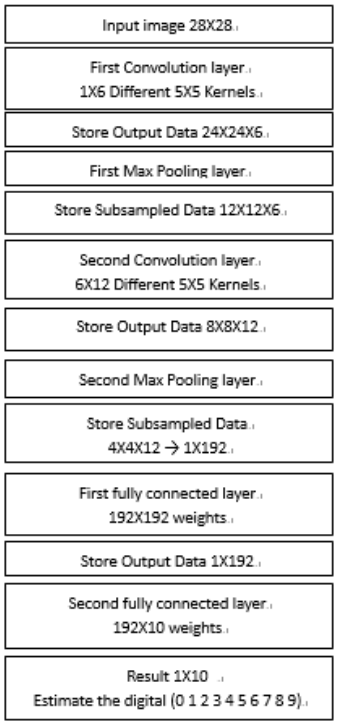


Fig. 9. Flowchart describing memristor-based convolutional neural network. The designed inputs and outputs of the proposed system are based on this flowchart.

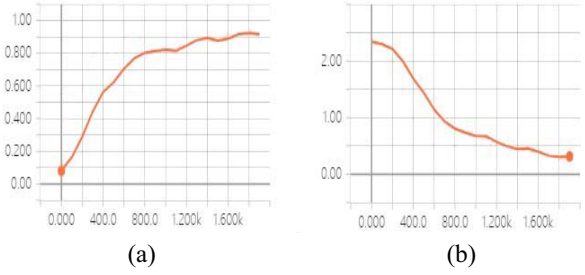


Fig. 10. (a) Training accuracy curve when training memristor-based convolutional neural network. The abscissa is the number of training steps, and the ordinate is the loss of training. The discrete points are errors of training, and the red line is the fitting curve. (b) Training loss curve when training memristor-based convolutional neural network. The abscissa is the number of training steps, and the ordinate is the loss of training. The discrete points are errors of training, and the red line is the fitting curve.

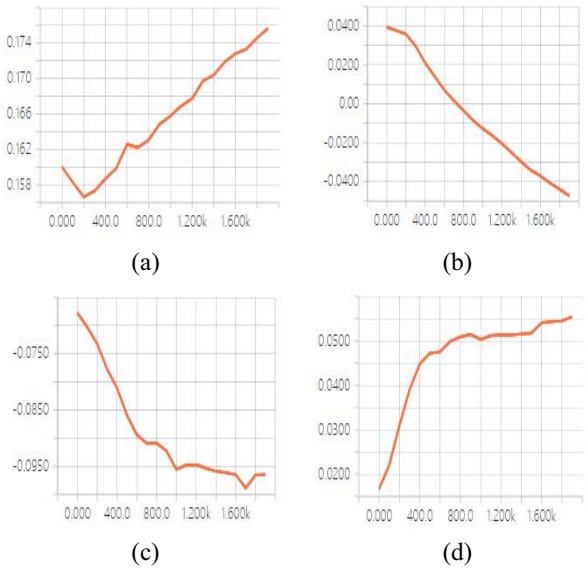


Fig. 11. Change curves of weights for convolution layers and fully connected layer in training. The horizontal coordinate represents the number of training steps, the ordinate represents the weight value. One of the weights in the (a) first convolution layer, (b) second convolution layer, (c) first fully connected layer, and (d) second fully connected layer.

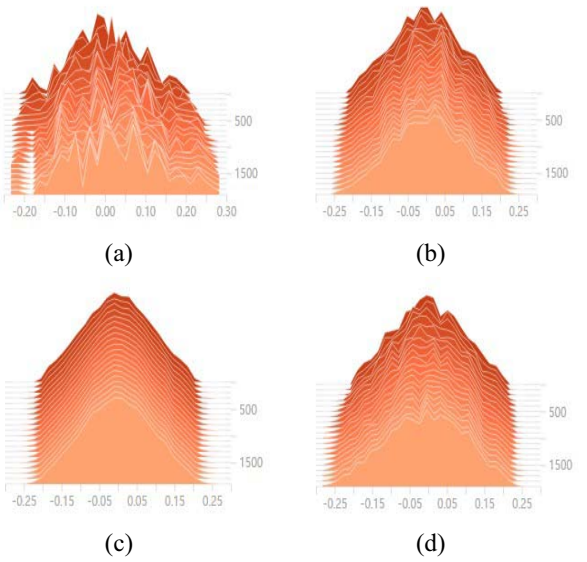


Fig. 12. Adjustment process of all the weights in each channel in training. The horizontal coordinate represents the weight value, the ordinate represents the number of training steps, and the height represents the number of corresponding weights. Weights of the (a) first convolution layer, (b) second convolution layer, (c) first fully connected layer, and (d) second fully connected layer.

To simply illustrate the structure of memristor-based convolutional neural network, a flowchart is presented to describe the memristor-based convolutional neural network in Fig. 9.

Fig. 10 shows the training accuracy curve and loss curve, respectively, from which we can see that when training with 2000 steps, the training loss has been very small, which indicates that memristor-based convolutional neural network is efficient in the MNIST database of handwritten digits. Fig. 11 shows the change curves of weights for convolution layers and fully connected layers in the training process, in which the values of memristors change to certain values to guarantee the convergence of the training loss as shown in Fig. 10 with 2000 steps. Fig. 12 shows the adjustment process of all the weights in each channel in training. From Fig. 10(a), we can see that the accuracy is about 92.46%. This implies the promising potential application of memristive deep learning networks with the nonvolatile properties of memristor in future. Future work will aim at reducing parameters to speed up the network computing and simplify the memristive hardware implementation by sparsifying CNN layers.

VI. Conclusion

In this paper, memristor-based artificial synapses were employed in the circuit design to implement MNNs. Compared with the CMOS-only circuit, the proposed scheme consumes only $2\% - 8\%$ area. To enable learning rate adjustment in MNNs, we utilized a fuzzy control method. Fuzzy control has been realized in hardware, such as FPGA and PLC. The hardware of fuzzy method has been utilized in many fields. Compared with a constant learning rate, our design results in $2\% - 3\%$ improvement in terms of accuracy. Meanwhile, the fuzzy approach was shown to be robust and insensitive to changes in parameters due to the approximate reasoning ability, which manifested some similarities with the human thinking process. Furthermore, the proposed methods have been extended to memristor-based multilayer convolutional neural network for complex tasks. For further work, we will focus on improving the accuracy and we need to find some memristive materials, which is easy to get and have the performance which is close to that of the ideal memristor model.

References

[1] T. Faure, H. Isaka, and B. Guillemet, "Neural network retrieval of cloud parameters from high-resolution multispectral radiometric data: A feasibility study," *Remote Sens. Environ.*, vol. 80, no. 2, pp. 285–296, 2002.

[2] X. Bi, M. Mao, D. Wang, and H. H. Li, "Cross-layer optimization for multilevel cell STT-RAM caches," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 6, pp. 1807–1820, Jun. 2017.

[3] J. Guo *et al.*, "FlexLevel NAND flash storage system design to reduce LDPC latency," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 7, pp. 1167–1180, Jul. 2017.

[4] S. B. Cho and J. H. Lee, "Learning neural network ensemble for practical text classification," in *Proc. Int. Conf. Intell. Data Eng. Autom. Learn.*, Heidelberg, Germany: Springer, Mar. 2003, pp. 1032–1036.

[5] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*. Heidelberg, Germany: Springer, 2012, pp. 9–48.

[6] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*. Heidelberg, Germany: Springer, 2012, pp. 421–436.

[7] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, nos. 1–3, pp. 239–255, 2010.

[8] A. R. Omondi, "Letter to the editor: Neurocomputers: A dead end?" *Int. J. Neural Syst.*, vol. 10, no. 6, pp. 475–481, 2000.

[9] X. Wang, Y. Chen, H. Xi, H. Li, and D. Dimitrov, "Spintronic memristor through spin-torque-induced magnetization motion," *IEEE Electron Device Lett.*, vol. 30, no. 3, pp. 294–297, Mar. 2009.

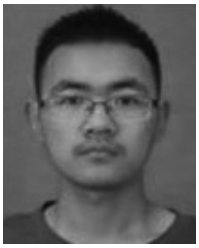
[10] B. Liu, Y. Chen, B. Wysocki, and T. Huang, "The circuit realization of a neuromorphic computing system with memristor-based synapse design," in *Proc. Int. Conf. Neural Inf. Process.*, Doha, Qatar, 2012, pp. 357–365.

[11] M. Hu, Y. Chen, J. J. Yang, Y. Wang, and H. H. Li, "A compact memristor-based dynamic synapse for spiking neural networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 8, pp. 1353–1366, Aug. 2017.

[12] B. Xu, Y. Shen, X. Wang, and L. Chen, "Efficient memristor model implementation for simulation and application," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 7, pp. 1226–1230, Jul. 2017, doi: 10.1109/TCAD.2017.2648844.
[13] Q. Chen, X. Wang, H. Wan, and R. Yang, "A logic circuit design for perfecting memristor-based material implication," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 2, pp. 279–284, Feb. 2017.
[14] Y. Guo, X. Wang, and Z. Zeng, "A compact memristor-CMOS hybrid Look-up-table design and potential application in FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 12, pp. 2144–2148, Dec. 2017.
[15] L. Chua, "Memristor-the missing circuit element," *IEEE Trans. Circuit Theory*, vol. CT-18, no. 5, pp. 507–519, Sep. 1971.
[16] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.
[17] F. Corinto and A. Ascoli, "A boundary condition-based approach to the modeling of memristor nanostructures," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 11, pp. 2713–2726, Nov. 2012.
[18] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "TEAM: ThrEshold adaptive memristor model," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 1, pp. 211–221, Jan. 2013.
[19] M. Versace and B. Chandler, "The brain of a new machine," *IEEE Spectr.*, vol. 47, no. 12, pp. 30–37, Dec. 2010.
[20] L. Deng *et al.*, "Complex learning in bio-plausible memristive," *Sci. Rep.*, vol. 5, Jun. 2015, Art. no. 10684.
[21] L. O. Chua and S. M. Kang, "Memristive devices and systems," *Proc. IEEE*, vol. 64, no. 2, pp. 209–223, Feb. 1976.
[22] S. Kvatinsky *et al.*, "Memristor-based multithreading," *IEEE Comput. Archit. Lett.*, vol. 13, no. 1, pp. 41–44, Jan./Jun. 2014.
[23] S. Duan, X. Hu, Z. Dong, L. Wang, P. Mazumder, "Memristor-based cellular nonlinear/neural network: Design, analysis, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 6, pp. 1202–1213, Jun. 2015.
[24] X. Hu, G. Feng, S. Duan, and L. Liu, "A memristive multilayer cellular neural network with applications to image processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 8, pp. 1889–1901, Aug. 2017.
[25] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat, "Immunity to device variations in a spiking neural network with memristive nanodevices," *IEEE Trans. Nanotechnol.*, vol. 12, no. 3, pp. 288–295, May 2013.
[26] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, "STDP and STDP variations with memristors for spiking neuromorphic learning systems," *Front. Neurosci.*, vol. 7, p. 2, Feb. 2013.
[27] W. Chan and J. Lohn, "Stochastic gradient descent tricks," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2012, pp. 1–6.
[28] Y. Kim, Y. Zhang, and P. Li, "A digital neuromorphic VLSI architecture with memristor crossbar synaptic array for machine learning," in *Proc. IEEE Int. SOC Conf. (SOCC)*, Sep. 2012, pp. 328–333.
[29] A. Nere, U. Olcese, D. Balduzzi, and G. Tononi, "A neuromorphic architecture for object recognition and motion anticipation using burst-STDP," *PLoS ONE*, vol. 7, no. 5, 2012, Art. no. e36958.
[30] O. Kavehei *et al.*, "Memristor-based synaptic networks and logical operations using in-situ computing," in *Proc. IEEE 7th Int. Conf. Intell. Sensors Sensor Netw. Inf. Process. (ISSNIP)*, Adelaide, SA, Australia, Dec. 2011, pp. 137–142.
[31] C. Zamarreño-Ramos *et al.*, "On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex," *Front. Neurosci.*, vol. 5, p. 26, Mar. 2011.
[32] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a memristor-based spiking neural network immune to device variations," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, San Jose, CA, USA, Jul. 2011, pp. 1775–1781.
[33] H. Manem, J. Rajendran, and G. S. Rose, "Stochastic gradient descent inspired training technique for a CMOS/nano memristive trainable threshold gate array," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 5, pp. 1051–1060, May 2012.
[34] D. Chabi, W. Zhao, D. Querlioz, and J.-O. Klein, "Robust neural logic block (NLB) based on memristor crossbar array," in *Proc. IEEE/ACM Int. Symp. Nanoscale Archit.*, San Diego, CA, USA, Jun. 2011, pp. 137–143.
[35] M. Soltiz, D. Kudithipudi, C. Merkel, G. S. Rose, and R. E. Pino, "Memristor-based neural logic blocks for nonlinearly separable functions," *IEEE Trans. Comput.*, vol. 62, no. 8, pp. 1597–1660, Aug. 2013.
[36] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 5, pp. 368–408, 2012.
[37] F. Alibart, E. Zamanidoost, and D. B. Strukov, "Pattern classification by memristive crossbar circuits using ex situ and in situ training," *Nat. Commun.*, vol. 4, p. 2072, Jun. 2013.
[38] O. Bousquet and L. Bottou, "The tradeoffs of large scale learning," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2008, pp. 161–168.
[39] Z. Vasilkoski *et al.*, "Review of stability properties of neural plasticity rules for implementation on memristive neuromorphic hardware," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, San Jose, CA, USA, Jul./Aug. 2011, pp. 2563–2569.
[40] D. Soudry, D. Di Castro, A. Gal, A. Kolodny, and S. Kvatinsky, "Memristor-based multilayer neural networks with online gradient descent training," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2408–2421, Oct. 2015.
[41] M. Waltz and K. Fu, "A heuristic approach to reinforcement learning control systems," *IEEE Trans. Autom. Control*, vol. AC-10, no. 4, pp. 390–398, Oct. 1965.
[42] X.-H. Yu and G.-A. Chen, "Efficient backpropagation learning using optimal learning rate and momentum," *Neural Netw.*, vol. 10, no. 3, pp. 517–527, 1997.
[43] C. Darken and J. Moody, "Towards faster stochastic gradient search," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 1991, pp. 272–279.
[44] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l_1-ball for learning in high dimensions," in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, Jul. 2008, pp. 272–279.
[45] J. Duchi and Y. Singer, "Efficient online and batch learning using forward backward splitting," *J. Mach. Learn. Res.*, vol. 10, pp. 2899–2934, Dec. 2009.
[46] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.
[47] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Barcelona, Spain, Nov. 2011, pp. 2018–2025.
[48] E. Monmasson and M. N. Cirstea, "FPGA design methodology for industrial control systems—A review," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1824–1842, Aug. 2007.
[49] D. Kim, "An implementation of fuzzy logic controller on the reconfigurable FPGA system," *IEEE Trans. Ind. Electron.*, vol. 47, no. 3, pp. 703–715, Jun. 2000.
[50] G. Wang, H. Song, and Q. Niu, "Mine water level fuzzy control system design based on PLC," in *Proc. 2nd Int. Conf. Intell. Comput. Technol. Autom.*, vol. 3. Changsha, China, Oct. 2009, pp. 130–133.
[51] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.
[52] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man Mach. Stud.*, vol. 7, no. 1, pp. 1–13, 1975.
[53] C. Zhou *et al.*, "An improved direct adaptive fuzzy controller of an uncertain PMSM for Web-based E-service systems," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 1, pp. 58–71, Feb. 2015.
[54] D. Ye, N.-N. Diao, and X.-G. Zhao, "Fault-tolerant controller design for general polynomial-fuzzy-model-based systems," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 2, pp. 1046–1051, Apr. 2018.
[55] J. Chen and R. J. Patton, *Robust Model-Based Fault Diagnosis for Dynamic Systems*, vol. 3. New York, NY, USA: Springer, 2012.
[56] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 20, no. 2, pp. 404–418, Mar./Apr. 1990.

Shiping Wen (M'16) received the M.Eng. degree in control science and engineering from the School of Automation, Wuhan University of Technology, Wuhan, China, in 2010, and the Ph.D. degree in control science and engineering from the School of Automation, Huazhong University of Science and Technology, Wuhan, in 2013.

He is currently an Associate Professor with the School of Automation, Huazhong University of Science and Technology, where he also with the Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China. His current research interests include memristor-based circuits and systems and machine learning.



Shuixin Xiao received the B.Eng. degree from Automation School, Huazhong University of Science and Technology, Wuhan, China, in 2018. He is currently pursuing the Ph.D. degree with Shanghai Jiaotong University, Shanghai, China.

He was with the School of Automation, Huazhong University of Science and Technology, Wuhan, China. His current research interests include memristor-based circuit, neuromorphic computation, and resistive memory and machine learning.

Yin Yang (M'16) received the B.Eng. degree in computer science from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2004, and the Ph.D. degree in computer science from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, in 2009.

He is currently an Assistant Professor with the College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar. He has published extensively in top venues on differentially private data publication and analysis, and on query authentication in outsourced databases. His current research interests include cloud computing, database security and privacy, and query optimization, cloud-based big-data analytics, with a focus on fast streaming data.

Zheng Yan (M'15) received the B.Eng. degree in automation and computer-aided engineering and the Ph.D. degree in mechanical and automation engineering from the Chinese University of Hong Kong, Hong Kong, in 2010 and 2014, respectively.

Dr. Yan was a recipient of the Graduate Research Grant from the IEEE Computational Intelligence Society in 2014.

Zhigang Zeng (SM'07) received the B.S. degree from Hubei Normal University, Huangshi, China, in 1993, and the M.S. degree from Hubei University, Wuhan, China, in 1996, and the Ph.D. degree from the Huazhong University of Science and Technology, Wuhan, in 2003.

He is a Professor with the School of Automation, Huazhong University of Science and Technology, where he also with the Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China. His current research interests include neural networks, switched systems, computational intelligence, stability analysis of dynamic systems, pattern recognition, and associative memories.

Tingwen Huang (SM'16) received the B.S. degree from Southwest Normal University (currently Southwest University), Chongqing, China, 1990, the M.S. degree from Sichuan University, Chengdu, China, 1993, and the Ph.D. degree from Texas A&M University, College Station, TX, USA, in 2002.

He is a Professor of mathematics, Texas A&M University at Qatar, Doha, Qatar. His current research interests include dynamical systems, neural networks, complex networks, optimization and control, and traveling wave phenomena.