# Memristive LSTM Network for Sentiment Analysis

**Article** *in* IEEE Transactions on Systems, Man, and Cybernetics: Systems · April 2019

DOI: 10.1109/TSMC.2019.2906098



**7 authors**, including:

Zhigang Zeng
Huazhong University of Science and Technology
**262** PUBLICATIONS **6,539** CITATIONS

SEE PROFILE

Tingwen Huang
Texas A&M University at Qatar
**507** PUBLICATIONS **10,310** CITATIONS

SEE PROFILE

# Memristive LSTM Network for Sentiment Analysis

Shiping Wen, Huaqiang Wei, Yin Yang, Zhenyuan Guo, Zhigang Zeng, *Senior Member, IEEE*, Tingwen Huang, *Fellow, IEEE*, and Yiran Chen, *Fellow, IEEE*

***Abstract*—This paper presents a complete solution for the hardware design of a memristor-based Long Short Term Memory (MLSTM) network. Throughout the design process, we fully consider the external and internal structures of the LSTM, both of which are efficiently implemented by memristor crossbars. In the specific design of the internal structure, the parameter sharing mechanism is used between the LSTM cells to minimize the hardware design scale. In particular, we designed a circuit that requires only one memristor crossbar for each unit in the LSTM cell. The activation function, including sigmoid and tanh (hyperbolic tangent function), involved in each unit is approximated by a piecewise function, which is designed with the corresponding hardware. To verify the effectiveness of the system we designed, we test it on IMDB and SemEval datasets. Considering the huge impact of the dimensions of the input data on the scale of the hardware design, we use word2vector instead of one-hot encoding for the input data encoding. With the parameter sharing mechanism, the transformed vectors are input in different periods, so only 65 memristive crossbars are needed in the entire system to complete the sentiment analysis of the input text. The experimental results verify the effectiveness of our proposed MLSTM system.**

***Index Terms*—Memristor, Long short term memory, Sentiment analysis, Deep learning**

## I. Introduction

TODAY, the development of deep neural networks such as convolutional neural network (CNN), fully convolutional network (FCN), and long short term memory (LSTM) has promoted the research of deep neural network hardware design. However, using hardware to build deep neural networks, we cannot avoid these problems, such as the large amount of input data (whether it is image or text), or the complexity of the network structure and numerous parameters. All of these require huge power consumption and hardware area. Memristor [1], [2], [3], [4], consuming very little power and area [5], [6], [7], [8], shows its great potential in deep network fields. Hence, memristor is expected to drive revolution in neuromorphic systems due to its nanoscale size, non-volatility, high density, low power consumption as well as compatibility with CMOS technology [9].

S. Wen, H. Wei, and Z. Zeng are with School of Automation, Huazhong University of Science and Technology, and Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, Wuhan, Hubei, 430074, China. (email: {wenshiping226; huaqiangwei; zgzeng}@hust.edu.cn). Y. Yang is with College of Science, Engineering and Technology, Hamad Bin Khalifa University, 5855, Doha, Qatar. (email: yyang@hbku.edu.qa). Z. Guo is with College of Mathematics and Econometrics, Hunan University, Changsha, Hunan, China (zyguo@hnu.edu.cn). T. Huang is with Texas A & M University at Qatar, Doha 23874, Qatar. (email: tingwen.huang@qatar.tamu.edu). Y. Chen is with Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (email: yiran.chen@duke.edu).

Memristor is usually in the form of crossbars [10], [11], [12] in the deep neural network circuit design. Using this layout, a high-density, extremely low-power parallel computing circuit can be generated. For example, numerous neuromorphic systems are developed by using memristive crossbar circuits in [13], [14] which illustrate the potential of the memristive crossbar structure in massive neural network implementation. Related works have proposed the implementation of memristive crossbar for multi-layer neural network with ex-situ method as described in [13], [15]. Also using memristive crossbar, Yakopcic et al. [16] presented a complete circuit implementation of a CNN based memristive crossbar. Therefore, this paper is also based on the memristive crossbar, in which we first propose a complete hardware circuit implementation scheme for the memristive LSTM network.

In our proposed MLSTM system, we first introduce its overall structure. It is well known that the simplest way to implement LSTM is to use a series approach. However, the problem caused by this method is obvious. The number of serially connected LSTM cells is related to the maximum length of the text we set. In this design, the maximum length of the text is M, so M cells need to be connected in series. However, the internal parameters of these cells, including weights and biases, are exactly the same, which would cause our hardware system to be inefficient. Therefore, we design the entire system as a circulatory system so that we only need one cell in the LSTM network of one layer, and this cell is used M times. The output and status of each intermediate process are written to memory to overwrite the original data. These data are read from memory in the next cycle. In this way, the hardware scale is greatly reduced.

The internal structure of the LSTM cell mentioned above is in the form of subunit, and its number is artificially set. The structure of each unit is the same while the parameters are completely different. Its internal structure is much more complex than that of RNN, the hardware design of the unit will be described in detail in the paper. While numerous LSTM variants have been described, here we describe the version used by Zaremba and Sutskever [17]. For the LSTM internal unit, we design a special memristive crossbar, which was implemented efficiently and succinctly. Only one memristor crossbar is required for each unit, meanwhile, the weights and biases are written into the memristors of the crossbars through ex-situ training method [18], [19]. The key benefit of ex-situ method is that any learning algorithm can be used for training, which is difficult in the in-situ method [20], [21], [22].

The hardware implementation of the internal activation function is a challenging part in our design. When designing the hardware of the ReLU activation function, a diode with a

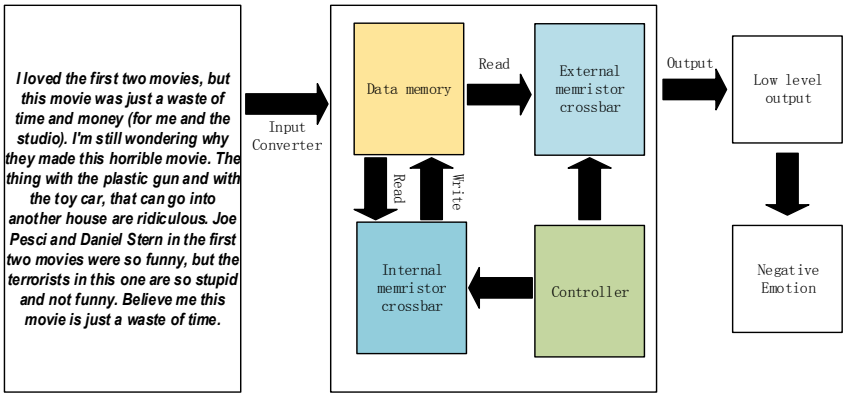



Fig. 1. The memristor-based LSTM neural network circuit system is applied to sentiment classification task. To the best of our knowledge, this is the first hardware design for a complete LSTM neural network based on memristor crossbars. The key point of this paper is the system design in the middle of the figure, which is divided into four modules. The core circuits are in the internal and external crossbars. In the example of figure, before the text is input to the circuit system, it is necessary to convert the input text into a matrix form and store it in the data memory. The textual sentiment that the system judges is obtained from the system output level.

bias voltage can be used to approximate the equivalent. For the sigmoid activation function, an approximate approach [23] is also used in the design of the CNN network. The hardware design of the tanh activation function is not mentioned before. In this MLSTM system, we derive the tanh expression from the sigmoid function and use a piecewise function to approximate the equivalent to simplify the designing the circuit of the piecewise function with memristor rather than the CMOS technology in [24]. In this case, the problem of activation function is solved in the case of acceptable accuracy. The parameters of the LSTM network are divided into internal hidden parameters and external parameters. The previous design is about the internal structure, while the external structure is relatively simple. We only need one external memristive crossbar in the entire system.

LSTM neural networks are currently widely used in natural language processing, especially in sentiment analysis tasks [25], [26], [27], [28]. So we take the well-known film review text sentiment analysis task in IMDB dataset [29] (the Internet Movie Database) and twitter sentiment analysis task in SemEval dataset [30] as examples to introduce proposed memristor-based LSTM circuit system. The next section introduces the overall structure, followed by the section III detailing the circuit. Section IV introduces the neural network implementation circuit scheme from the internal to external module. Section V performs simulations to verify the proposed MLSTM system.

## II. MLSTM Neural Network Structure

Recurrent neural networks (RNNs) have obvious advantages in text sentiment analysis tasks due to their ability to process arbitrary-length sequences. However, RNN has a serious problem during its training process. When it processes long sequences [31], [32], its gradient components may grow exponentially or decay. This problem can easily lead to the gradient explosion or disappearance of the training process, which makes it harder to learn the correlation of long sequence of RNN models. Fortunately, the Long Short-Term Memory (LSTM) architecture [33] addresses this problem of learning long-term dependencies, which is solved by introducing memory cells that can hold state for long periods of time. At present, the LSTM network has been successfully used for machine translation [34], [35], speech recognition [36], program execution [37], image title generation [38], and emotional analysis [39].

The wide applications of LSTM benefit from its special structure. In recent years, many LSTM-based variants have been proposed in deep learning, such as bi-directional LSTM, popular attention-based methods (such as global and local attention [40], self-attention [41], multi-attention [42]) and other optimization methods [43]. However, the LSTM structure of the backbone has not changed. Therefore, the implementation of the memristor-based LSTM infrastructure will greatly promote the hardware implementation of other LSTM variants. Taking the text sentiment classification for film reviews as an example, Fig. 1 shows the complete structure of the MLSTM system we designed. The main purpose of this paper is to introduce the methods of our memristor-based LSTM network, so one layer LSTM network is used. To increase the number of layers, only additional one LSTM cell with the same structure needed. After we test one layer of LSTM, we can still achieve a satisfactory sentiment classification result. The system is divided into two parts: internal and external. The inside part is a loop control system, in which an LSTM cell exists; in the outside part, the final classification task is completed by a memristor crossbar with a voltage comparator. The detailed structure is introduced in Section III and IV.

## III. Activation function and MLSTM internal unit implementation

This section focuses on the hardware implementation of the MLSTM internal unit and activation functions. Memristor crossbars are used in the hardware design of the LSTM internal unit. As the circuits of the activation functions are involved, we

firstly introduce the implementation of activation functions and then introduce the detailed implementation circuit of internal unit.

### A. Activation function implementation

In deep learning, we will add activation functions to increase the nonlinearity of neural network models, which map a large range of real numbers to a very small range. The activation functions we often use are sigmoid, tanh, relu, and so on. If the activation functions are not used, the output of each layer is a linear combination of the inputs. It is easy to verify that, no matter how many layers of your neural network, the output is a linear combination of input, which is equivalent to the effect of no hidden layers, such as the original perceptron. On the basis of the above reasons, the deep neural network is meaningful after introducing the nonlinear function as the activation function. It is no longer a linear combination of input, but can approximate any function. The following is mainly about the sigmoid and tanh activation functions used in our design, while we don not elaborate on hardware implementations of other activation functions that are not used.

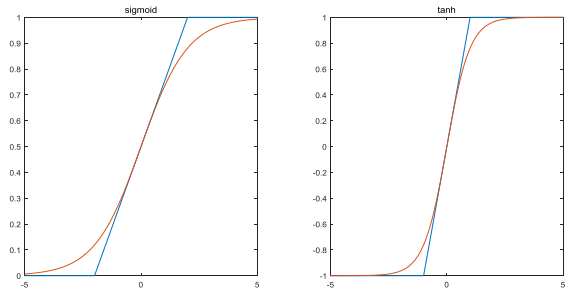



Fig. 2. Comparison of the sigmoid and tanh activation function with their approximated piecewise function in the form of curves.

*1) Sigmoid activation function implementation:* This paper mainly proposes the implementation method of tanh activation function, as the implementation of sigmoid function in [23]. In order to facilitate the introduction of tanh implementation, we give a brief explanation in this section. It is our core idea to use a piecewise function to approximate the sigmoid function. In this design, the piecewise function in Eq. (1) is used. In Fig. 2, the curves are respectively sigmoid function and approximate piecewise function. It can be seen from the graph that the piecewise function has obtained a good approximation effect.

$$sigmoid(x) \approx G1(x) = \begin{cases} 1 & x > 2 \\ 0.25x + 0.5 & |x| <= 2 \\ 0 & x < -2 \end{cases} \tag{1}$$

The hardware implementation of this piecewise function is described in detail below. We first give a concrete implementation circuit diagram, as shown in Fig. 3. In the left side of the circuit, $x$ is the value of the input feature. We all know that memristor cannot express negative weights, so we need to express a weight with two memristors and then divide the

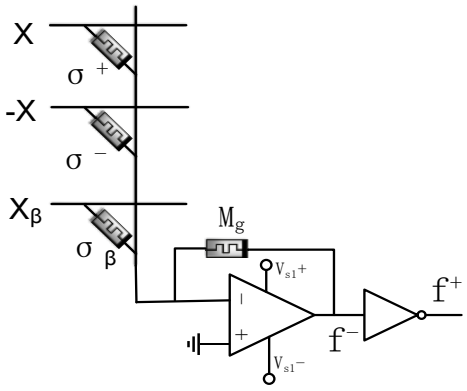



Fig. 3. The realization of the hardware circuit schematic of the sigmoid function. $X$ is the value of the input feature, $V_{s1}^{+}$=0 V, $V_{s1}^{-}$=-1 V. From top to bottom, the conductance values of the memristor are $\sigma^{+}$, $\sigma^{+}$ and $\sigma_{\beta}$, and the resistance value of $M_g$ is $R_g$. Since the input signal is input from the inverting terminal of the operational amplifier, a voltage inverter is finally required to convert the voltage direction.

input voltage into $x$ and $-x$. This method is described in detail in [13], [23]. Here $x_{\beta}$ is 1 V, $\sigma_{\beta}$ is $(1/2+\beta/4)/R_g$, $\beta$ is offset, and $R_g$ is

$$R_g = \frac{1}{4} * \frac{\max(|W|)}{(\sigma_{\max} - \sigma_{\min})} \tag{2}$$

Conductance values are considered as weight values in the memristor based neural network circuits. As mentioned above, we need two memristors to express a single weight. We have to convert the weights to $W^{+}$ and $W^{-}$, and then convert to the conductance values that can be programmed into the memristor crossbar according to the following formula:

$$\sigma^{+} = \frac{(\sigma_{\max} - \sigma_{\min})}{\max(|W|)} W^{+} + \sigma_{\min} \tag{3}$$

$$\sigma^{-} = \frac{(\sigma_{\max} - \sigma_{\min})}{\max(|W|)} W^{-} + \sigma_{\min} \tag{4}$$

The circuit specific parameters are given above. Next we verify the correctness of the proposed circuit by deriving calculations. According to the virtual-off characteristic of an ideal operational amplifier, we can have that $i_l$ is equal to $i_g$, from which we can derive the following:

$$f^{-} = -[R_g(x\sigma^{+} - x\sigma^{-}) + R_g x_{\beta}\sigma_{\beta}] \tag{5}$$

By taking the above setting parameters into Eq. (5), we can get the output voltage of the operational amplifier:

$$f^{-} = -\frac{1}{4}[x(W^{+} - W^{-}) + \beta] - \frac{1}{2} \tag{6}$$

After passing through a voltage inverter, the final output voltage of our circuit is as follows:

$$f^{+} = \frac{1}{4}[x(W^{+} - W^{-}) + \beta] + \frac{1}{2} \tag{7}$$

*2) Tanh activation function implementation:* Tanh is an abbreviation of the hyperbolic tangent function. In the implementation of the tanh activation function, we also use a piecewise function for replacement. We know that $sigmoid(x) = 1/(1+e^{-x})$. We make the following derivation:

$$\begin{aligned}\tanh(x) &= \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \\ &= \frac{1-e^{-2x}}{1+e^{-2x}} \\ &= \frac{2-(1+e^{-2x})}{1+e^{-2x}} \\ &= 2\frac{1}{1+e^{-2x}} - 1 \\ &= 2sigmoid(2x) - 1 \end{aligned} \tag{8}$$

Combining Eq. (1) and Eq. (8), we can get the tanh equivalent piecewise function as:

$$\tanh(x) \approx G2(x) = \begin{cases} 1 & x > 1 \\ x & |x| <= 1 \\ -1 & x < -1 \end{cases} \tag{9}$$

The relationship between the two activation functions and the corresponding segment functions is shown in Fig. 2. The sigmoid activation function is on the left and the tanh function is on the right. Comparing the two graphs in the left and right of the figure, we can see that the piecewise function approximation effect we adopt is better than that of the sigmoid function.

The circuit in Fig. 3 is versatile, and it can be extended to other three piecewise functions. Compared with Eq. (8) and Eq. (1), they are both three segment piecewise functions, and the upper and lower limits are all constants. Therefore, we can define the range of piecewise functions through the voltage of positive and negative voltage sources of operational amplifiers. The linear functions in the middle of piecewise functions are adjusted by controlling the values of $\sigma_\beta$ and $R_g$. Therefore, we still use the Fig. 3 circuit to implement the tanh activation function, and what is different is that the parameters in the circuit need to be changed. $\sigma_\beta$ is $\beta/R_g$ and

$$R_g = \frac{\max(|W|)}{(\sigma_{\max} - \sigma_{\min})} \tag{10}$$

Other parameters remain unchanged. According to Eq. (3), Eq. (5), Eq. (10) we can get the output of the circuit as:

$$f^{+} = -f^{-} = x(W^{+} - W^{-}) + \beta \tag{11}$$

Finally, the upper and lower limits of the function are defined by setting the positive and negative voltage source voltages of the operational amplifiers, so that a piecewise function is completely realized. Here $V_{s1}^{+}$=-1 V, $V_{s1}^{-}$=1 V. The following LSTM cell internal unit circuit design is based on the theory of these two sections.

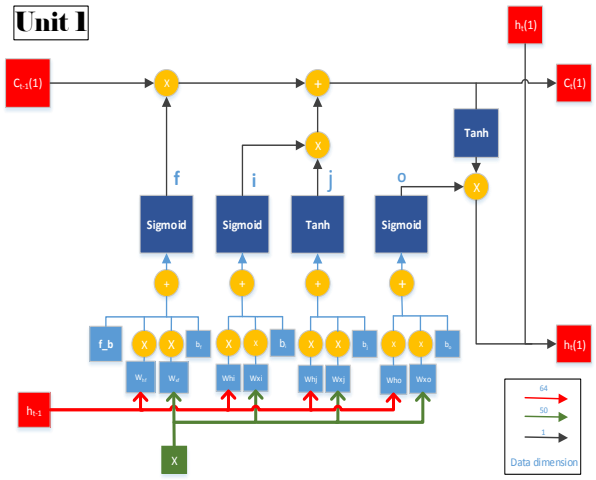



Fig. 4. A single unit structure diagram inside the LSTM cell. The box in the lower right corner of the figure indicates that three different colors represent different data dimensions. $W$ stands for weight and $b$ stands for bias. Next to the lines, $f$ is forget gate, $i$ is input gate, $j$ is input update, and $o$ is output gate. $c_t$ is cell state and $h_t$ is hidden state (output).

### *B. The units in MLSTM cell implementation*

In this section we introduce the complete memristor-based LSTM cell internal units implementation principle and circuit. In Fig. 4, we provide a detailed parameter calculation process for the internal unit of the LSTM cell, where the different colors of the data lines represent the dimensions of the data. To the best of our knowledge, this figure is the most detailed and intuitive representation of the internal LSTM. In [45], detailed formulas and internal structure diagrams are provided, however it gives the structure of the entire cell, so no specific parameter positions are provided, and the units are not split, and the weights are in the format of matrix instead of vector. In order to facilitate the hardware circuit design, we split the entire cell into 64 independent units. Below we provide a detailed calculation formula for the f, i, j, o in Fig. 4:

$$\begin{aligned} f &= sigmoid(x \cdot W_{xf} + h_{t-1} \cdot W_{hf} + b_f + f\_b) \\ i &= sigmoid(x \cdot W_{xi} + h_{t-1} \cdot W_{hi} + b_i) \\ j &= \tanh(x \cdot W_{xj} + h_{t-1} \cdot W_{hj} + b_j) \\ o &= sigmoid(x \cdot W_{xo} + h_{t-1} \cdot W_{ho} + b_o) \end{aligned} \tag{12}$$

where $f$ is forget gate, $i$ is input gate, $j$ is input update, and $o$ is output gate. It is worth noting that the forgetting bias $f_b$ is added to the $f$ calculation, which is constant 1 because the value is added by default in the calculation of TensorFlow. The weights in Eq. (12) are shown in Fig. 5.

$$\begin{aligned} c_t(1) &= c_{t-1}(1) \cdot f + i \cdot j \\ h_t(1) &= o \cdot \tanh(c_t(1)) \end{aligned} \tag{13}$$

The outputs $c_t(1)$ and $h_t(1)$ of the LSTM unit can be obtained from Eq. (13), which is also easily calculated from Fig. 4. It is worth emphasizing that since the figure is only a single unit of 64 units, both $c_t(1)$ and $h_t(1)$ dimensions are 1. The $c_t$

and $h_t$ are formed by combining the calculation results of 64 units.


Fig. 5. Internal hidden weight matrix of LSTM cell, where $m$ is the dimension of the input data, and $n$ is the number of LSTM cell internal units. They determine the size of the internal hidden weight matrix together.

We call the weights inside the LSTM cell as hidden weights because the weights in the depth learning framework such as TensorFlow does not need to be defined, nor can they be viewed directly, but can only be viewed by reading the ckpt file. And we only need to define the external weights and biases. However, the proportion of hidden weights and biases is significantly greater than the external weights, and the hidden weights and biases are constantly adjusted and updated in the learning process.

In Fig. 5, $m$ represents the input data dimension and $n$ refers to the number of internal units. Therefore, the total weight of each unit is $4(m+n)$, the number of offsets (excluding constant $f_b$) is 4, and the total parameter of a unit is:

$$N = 4 * (m + n + 1) \tag{14}$$

The number of parameters contained in the entire cell is $n \times N$. In our design, the dimension of the input data is $m$, and the number of elements is $n$, so the total number of parameters for a cell is N. The number of these parameters determines the number of memristors in our designed memristive crossbar.

$$\begin{aligned} V_f =& G1\Big(\sum_{i=1}^{m}(x_i \cdot {\sigma_{xf,i}}^{+} - x_i \cdot {\sigma_{xf,i}}^{-}) \\ &+ \sum_{i=1}^{n}(h_{t-1,i} \cdot {\sigma_{hf,i}}^{+} - h_{t-1,i} \cdot {\sigma_{hf,i}}^{-}) + x_b\sigma_{bf} + \sigma_{fb}\Big) \end{aligned} \tag{15}$$

$$\begin{aligned} V_i =& G1\Big(\sum_{i=1}^{m}(x_i \cdot {\sigma_{xf,i}}^{+} - x_i \cdot {\sigma_{xi,i}}^{-}) \\ &+ \sum_{i=1}^{n}(h_{t-1,i} \cdot {\sigma_{hi,i}}^{+} - h_{t-1,i} \cdot {\sigma_{hi,i}}^{-}) + x_b\sigma_{bi}\Big) \end{aligned} \tag{16}$$

$$\begin{aligned} V_j =& G2\Big(\sum_{i=1}^{m}(x_i \cdot {\sigma_{xj,i}}^{+} - x_i \cdot {\sigma_{xj,i}}^{-}) \\ &+ \sum_{i=1}^{n}(h_{t-1,i} \cdot {\sigma_{hj,i}}^{+} - h_{t-1,i} \cdot {\sigma_{hj,i}}^{-}) + x_b\sigma_{bj}\Big) \end{aligned} \tag{17}$$

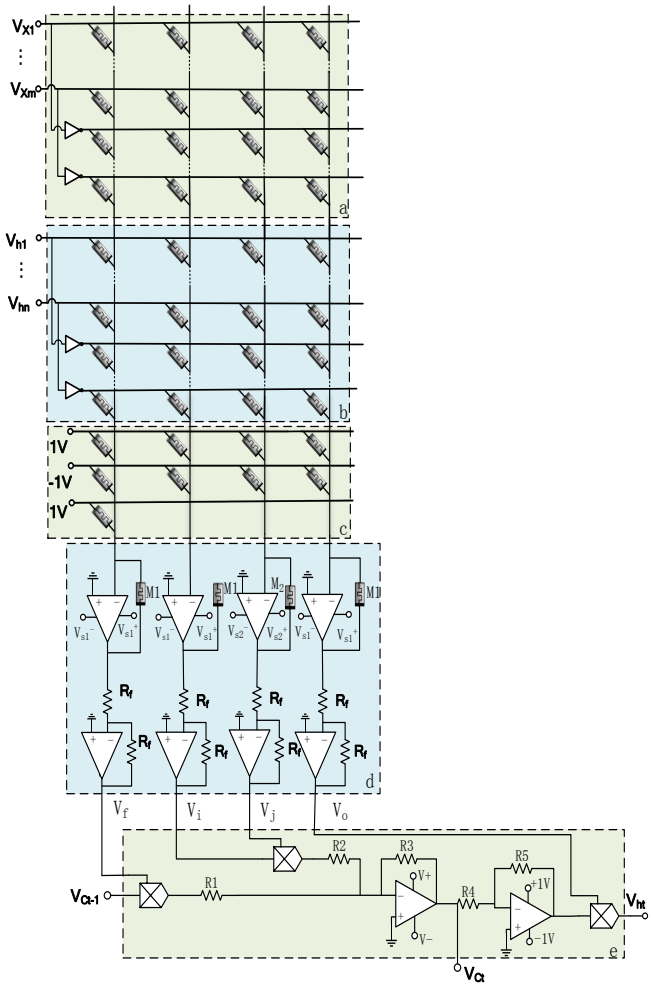



Fig. 6. The circuit diagram of the LSTM cell internal unit based on the memristive crossbar. For ease of explanation, we have marked five different areas in the circuit diagram. Memristors in part (a) of the crossbar store the $W_x$ of the weight matrix in Fig. 5, part (b) stores $W_h$, part (c) stores internal hidden offsets and oblivion offsets, part (d) of the circuit implements the activation function with the previous bias and and the operational amplifier with two $R_f$ in its lower part acts as a voltage inversion, part (e) is an auxiliary circuit for the memristive crossbar. This circuit completely implements the functions of the single unit of the LSTM neural network in Fig. 4. Thus the final output voltages of this circuit $V_{ht}, V_{ct}$ corresponds to $c_t(1), h_t(1)$ while the output voltages of memristor crossbar $V_f, V_i, V_j, V_o$ corresponds to $f, i, j, o$ in Fig. 4.

$$\begin{aligned} V_o =& G1\Big(\sum_{i=1}^{m}(x_i \cdot {\sigma_{xo,i}}^{+} - x_i \cdot {\sigma_{xo,i}}^{-}) \\ &+ \sum_{i=1}^{n}(h_{t-1,i} \cdot {\sigma_{ho,i}}^{+} - h_{t-1,i} \cdot {\sigma_{ho,i}}^{-}) + x_b\sigma_{bo}\Big) \end{aligned} \tag{18}$$

The circuit in Fig. 6 is an important part of our overall hardware design. The memristor crossbar is the core framework of the hardware implementation of Fig. 4. The output voltages of memristor crossbar $V_f, V_i, V_j, V_o$ can be calculated by Eqs. (15-18). We show the parts of the circuit that contain the parameters while the multiplication and addition circuits

for one-dimensional data are not shown. Since these circuits have been widely used, we do not explain in detail. The upper right tanh function implementation method of Fig. 4 has already been described in the previous section, and also uses the piecewise function to approximate substitution. The whole graph is divided into 5 different parts, and the function of each part is detailed in the annotations below the diagram. In our MLSTM system, there is only one LSTM cell, which contains 64 units. The implementation circuit of each unit is the same, however memristors in different units store different weights.

## IV. Memristive LSTM sentiment analysis system

The entire structure of the system proposed in this paper is provided in Section II. The previous section discussed how to use memristive crossbars to implement the circuit design of the LSTM cell internal units and the activation functions involved. This section shows the internal structure of the LSTM cell and the external classification structure.

### A. Internal LSTM cell structure

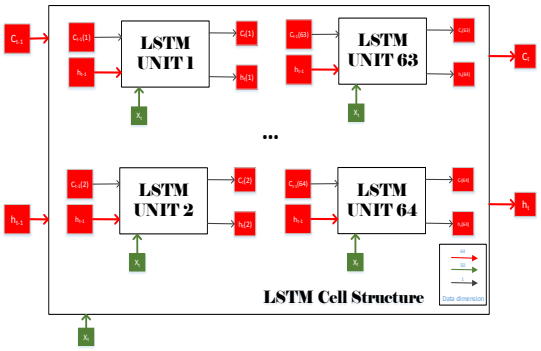
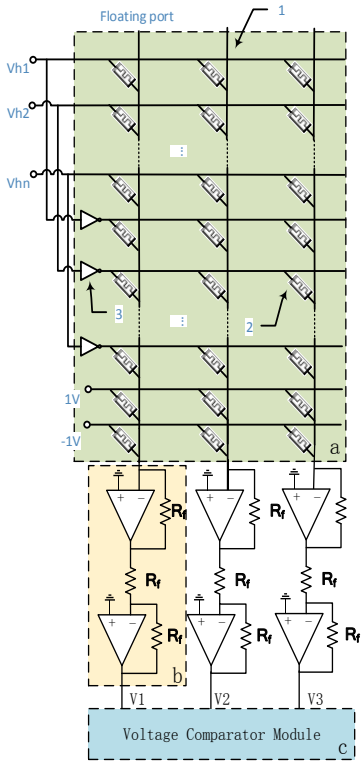



Fig. 7. LSTM cell structure. Different colors of data lines represent different dimensions. The same names in the red box are electrically connected. $c_t$ is cell state and $h_t$ is hidden state (output). $c_{t-1}$ and $h_{t-1}$ are the results of the previous loop.

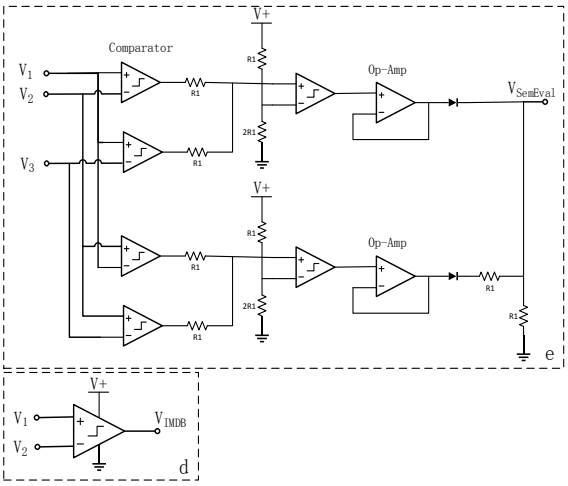
Fig. 7 shows the whole structure of LSTM cell. This graph corresponds to the LSTM cell in the loop control system in Fig. 1, and the LSTM unit module in Fig. 7 corresponds to the previous Fig. 4, whose specific implementation is based on the circuit in Fig. 6. The number of units set in our design is 64, so an LSTM cell contains 64 units, as shown in Fig. 7. $c_{t-1}$ and $h_{t-1}$ entered are both 64 in length. After entering the LSTM cell, $h_{t-1}$ is divided into 64 voltage input memristor crossbars, while $c_{t-1}$ is first split into 64 and then input into 64 cells respectively. The final 64 outputs are combined into a 64-length vector, so both $c_t$ and $h_t$ are 64 in length.

### B. External emotion classification circuit based on a memristive crossbar

After cycling $M$ times from the loop control system, its output voltages enter the external classification layer. The output dimension of the internal LSTM cell circuit is 64, so the size of our external memristor crossbar is $(64\times2+1)\times C$, where $C$ represents the number of categories for sentiment classification tasks ($C$ is 2 for IMDB and 3 for SemEval). Fig.




(a)




(b)

Fig. 8. (a) External emotion classification ciucuit based on a memristive crossbar. The key points of Fig. 8 (a) are marked with three parts a, b, and c, where part a is a memristive crossbar (note that there is no electrical connection at mark 1; mark 2 is a threshold memristor whose characteristics are detailed in the simulation section; mark 3 is the voltage inverter symbol, which is built with an op amp and resistors in analog circuits). (b) The detailed circuits for the voltage comparator module. The circuit of part d corresponds to the two-class task of IMDB, while part e implements the three-class task of SemEval.

8 shows the details of the external classification layer. Fig. 8 (a) shows the overall structure, while subgraph b supplements the details of the voltage comparator module implementation in (a).

In Fig. 8 (a), the parameters stored in this memristor crossbar are called external weights and biases. The number of input voltage ports is $n$, which is determined by the number of internal units in LSTM (where $n$ is 64). The key points in the circuit are marked with three parts a, b, and c. The function of part b is to convert the current signal into a voltage signal. The current is input from the inverting terminal, so the output voltage is reversed. Therefore, the second operational amplifier circuit acts to correct the voltage direction. Part c is a voltage comparator module and its final output level indicates the emotion of input text.

For two-class tasks, the voltage comparator module uses the circuit of part d in Fig. 8 (b), while for three-class tasks, it uses the circuit of part e. $V_{IMDB}$ has two levels and $V_{SemEval}$ has three levels. Their relationship with input voltage $V_1$, $V_2$ and $V_3$ can be expressed by the following formula:

$$V_{IMDB} = \begin{cases} V^+ & V_1 > V_2 \\ 0 & V_1 <= V_2 \end{cases} \tag{19}$$

$$V_{SemEval} = \begin{cases} V^r & V_1 > V_2 > V_3 \\ 0.5V^r & V_2 \geq V_1 > V_3 \\ 0 & V_3 \geq V_2 \geq V_1 \end{cases} \tag{20}$$

where $V^r = V^+ - V_D$, $V_D$ is the threshold voltage of a diode.

The system utilizes this circuit module to complete the sentiment analysis task. In the external emotion classification circuit module, only one crossbar is required here, so the entire system only needs 65 memristor crossbar. Thus the designed MLSTM system can be efficient and concise.

## V. Experimental simulation

The previous sections detail the principles and structure of our proposed MLSTM system. In this section, we will test it utilizing the popular IMDB and SemEval datasets. The whole application can be divided into the following four steps:

### A. Data preparation

*1) IMDB dataset:* The IMDB dataset contains 50,000 movie reviews, of which the number of training and testing is equal, and half are positive, the other half are negative. However, we only used 25,000 of these reviews, which include 2000 as test data and 23,000 as training data. Before we import the data, we first converted the text data into a word vector [46] because text data could not be input to our MLSTM system. We first get word vectors on the software and download it into our hardware. Here we use a pretrained word2vector model [47]. At present, there is a word2vector model that Google has trained on a massive Google News dataset. From this model, Google was able to create 3 million word vectors, each with a dimensionality of 300. However, this model is too large. We will use a more manageable matrix that is trained by Glove [48]. According to the statistics in IMDB dataset, the maximum length of the input (parameter M) is set to 250, and the converted vector of each word is 50-dimensional, so 250 vectors of length 50 are obtained and written to the memory.

TABLE I
THE NUMBER OF MEMRISTIVE CROSSBARS, CROSSBAR SIZE, AND MEMRISTOR QUANTITIES REQUIRED FOR EACH PART OF THE ENTIRE SYSTEM.

| Part of MLSTM | Required crossbars | Crossbar Size (row×col) | | Guantity of Memristors | |
|---|---|---|---|---|---|
| | | IMDB | SE | IMDB | SE |
| Unit | 1 | 230×4 | 530×4 | 921 | 2121 |
| Cell | 64 | 230×4 | 530×4 | 58944 | 135680 |
| External | 1 | 132×2 | 132×3 | 264 | 396 |

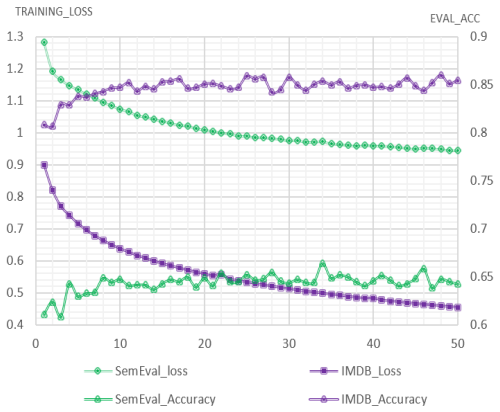
*2) SemEval dataset:* The Sentiment Analysis in Twitter task at SemEval dataset has been run since 2013. The SemEval dataset has been developed to include both Arabic and English and five subtasks. In this paper, our system completes the first subtask: Given an English tweet, decide whether it expresses POSITIVE, NEGATIVE or NEUTRAL sentiment. The dataset is divided into 90% for training and 10% for evaluation. Specifically, the training set contains 44,613 examples, and the other 4,957 examples are used for verification. During the experiment, we found that the twitter text in SemEval dataset is much shorter than the long text in IMDB, while the content of Twitter text is more complex. Thus we adjust some parameters of our network: the maximum reserved length of text (parameter $M$) is adjusted from 250 to 50, and the dimension of word vector (parameter $m$) is adjusted from 50 to 200. The other system parameters remain unchanged.

### B. Writing for memristors in MLSTM

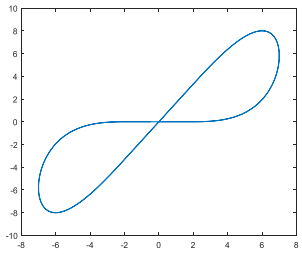
After the input data is ready, the next step is to download the weights and biases from software. We train the LSTM network in tensorflow [49]. For both IMDB and SemEval datasets, the training loss and accuracy on evaluation set during the training process can be seen from Fig. 9 (a), and the curves in Fig. 10 of the weights clearly show the entire training process.

It is then written into our memristors in crossbar, proposed by Lu et al at University of Michigan [50]. The V-I curve of threshold memristor is shown in Fig. 9 (b). When the voltage across the memristor is greater than the threshold voltage, its resistance changes back, otherwise the resistance remains unchanged. In this way, the memristor can be divided into two phases, one is a write phase and the other is a normal calculation phase. After writing phase, it switches to the read circuit to keep the voltage below the threshold voltage, and the value of the memristor does not change any more.

Fig. 11 shows the change process of the whole external weight and bias. We can see that in the last stage of training, the parameter change is decreasing and tends to be stable. According to our statistics, the number of memristor crossbar and memristors used are shown in Table I. The main circuit is to focus on the LSTM cell section. There is only one external memristive crossbar and it is relatively small.




(a)



(b)

Fig. 9. (a) The training loss curve and evaluating accuracy when training the LSTM network on software. The abscissa is the number of training epoch. The main vertical axis represents the training loss value, and the secondary vertical axis represents the accuracy of the evaluation set during training. (b) The V-I characteristic curve of the threshold memristor.

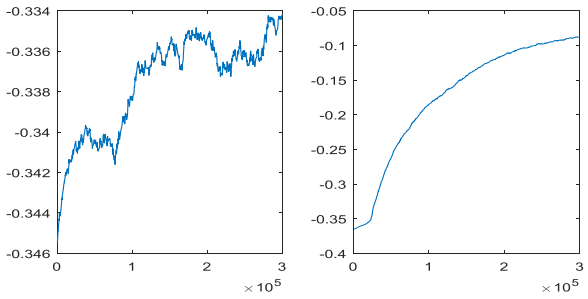
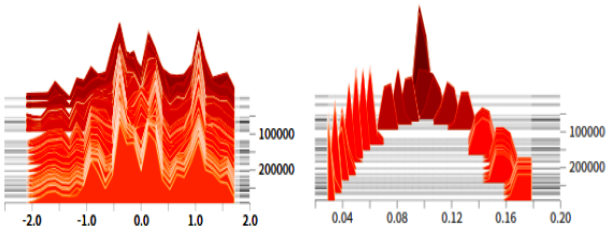


Fig. 10. The change curves of two external weights in training. The horizontal coordinate represents the number of training steps, the ordinate represents the weight value.

### C. Running the MLSTM system

After the input data is ready and the memristor values are set, the MLSTM system can be run for sentiment analysis. The operation process is shown in Fig. 12. The whole MLSTM circuit system runs in two stages:

*1) Internal memristive crossbars operation phase (Switch S1 is closed and S2 is open.):* Read input data from the data memory and enter a vector at a time. Each vector contains m values, each of which is converted to an analog voltage by a DA converter, where $m$ is the dimension of the input. The voltage is then input from the left port of the crossbars,




(a) (b)

Fig. 11. The adjustment process of all the external weights and biases. The horizontal coordinate represents the weight and bias value, the ordinate represents the number of training steps, and the height represents the number of corresponding weights.

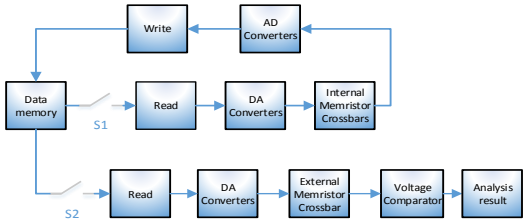



Fig. 12. A detailed block diagram of the operation of the proposed MLSTM system.

and finally the results of the calculation are read from the circuit output port through AD converters and written into the memory. The loop is operated n times, where n is the max length of text. And the final output is still stored in memory.

*2) External memristive crossbar operation phase (Switch S1 open and S2 is closed):* The h (its dimension is 64) is read from the data memory and converted to analog voltages by AD converters. Then input from the left segment of the external crossbar and get high or low voltage from the voltage comparator output port.

### D. Experiment results

At present, the methods based on hardware implementation of LSTM are mainly divided into CMOS [51], [52] and FPGA based [53], [54]. Until now, the FPGA-based method has not been able to implement text sentiment analysis, and the main problem of CMOS-based LSTM is that this device costs large areas and power consumption. Memristor is a nanoscale device, which has obvious advantages in size compared to the previous two methods. In terms of accuracy, we finally achieved 84.3% accuracy in the IMDB test set, which is significantly higher than the CMOS-based LSTM system [51], [52], and this accuracy is even higher than some software-based implementations [55], [56]. On the SemEval dataset, we also achieved a competitive accuracy of 65.8%. As far as we know, there are no other hardware-based LSTMs tested on this dataset. The results demonstrate that the threshold memristor has excellent characteristics as a neural synapse.

In order to verify the superiority of our system, we compared our MLSTM system for IMDB with CMOS-based neural network system in terms of power consumption and

area. To facilitate the performance comparison with CMOS, without calculating the auxiliary circuits in the system, we compare the power consumption of individual neurons, as shown in Table II. The calculation of power consumption is inseparable from the operating mechanism of the system, so the power consumption calculation can also be divided into the write phase and the calculation phase:

$$P = W/t = (\int_{t_0}^{t_i} U_1^2 G(t) d_t + \int_{t_i}^{t_r} U_2^2 G d_t)/(t_r - t_0) \quad (21)$$

The power consumption is closely related to the weights. Fig. 13 shows the weight and bias distribution of the entire system. We get the maximum value of all weights is 1.91, the maximum value of the internal weights is 0.38, and the average value is 0.065. The maximum value of the bias is 0.11, and the average value is 0.023. In addition, the maximum value of all input word vectors is 5.31, with an average value of 0.504. Finally, the maximum power consumption of a single memristor synapse is calculated to be $5.6\mu W$. The power consumption of a single synapse is much smaller than that of a CMOS setup, and our average power consumption is far below this value. For hardware complexity, the CMOS synapse using this design requires 16 CMOS tubes, and we only need a memristor to build a synapse.

TABLE II
SYSTEM LEVEL COMPARISON.

| System | Number of Devices for A Synapse | Max Power of A Synapse($\mu W$) |
|---|---|---|
| CMOS based | 16 | ≈50 |
| Memristor based | 1 | 5.6 |

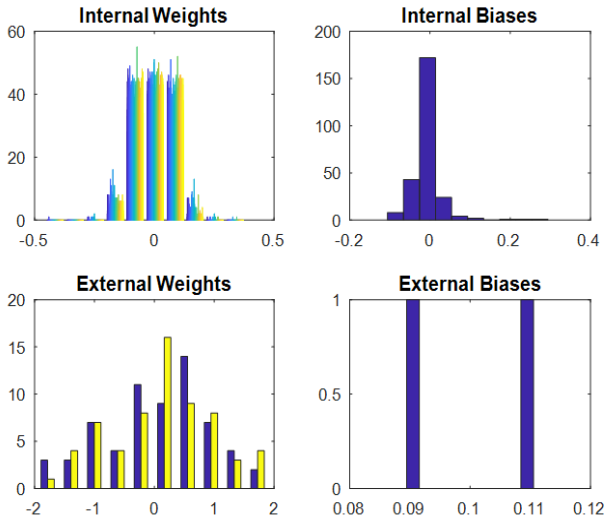



Fig. 13. The entire system parameter distribution. The abscissa is the parameter specific value, and the ordinate is the number of occurrences of this value.

The experimental results show that memristors are effective in building a deep neural network on the hardware, and we can also see the simplicity and low power consumption of the memristor crossbar. The structure is simplified by memristor for a general floating gate or capacitor in [57], [58]. Similarly, memristor has been verified with the lower power consumption than CMOS in [59], [60], [61]. Earlier, some researchers [62] also demonstrated that the power consumption of a low-pass filter implemented with memristor is $2\mu W$, while the power consumption for CMOS requires $17\mu W$. Memristors have become a research hotspot for hardware design of large-scale neural networks due to the above-mentioned reasons. At the same time, it provides better area efficiency than CMOS technology due to its nanoscale size [57], [63].

## VI. CONCLUSION

This paper presents a complete solution of LSTM network hardware implementation based on memristor crossbar. Hyperbolic tangent and sigmoid function hardware implementation are provided in this paper by utilizing piecewise functions to approximate. The MLSTM system was introduced in detail from internal and external structures. The main bodies of the two structures are both high-density memristor crossbars. Combined with the auxiliary circuits we designed, the whole circuit system is simple and efficient. Experimental results show that the proposed memristor-based LTSM system achieves higher accuracy than CMOS-based methods, as well as greater performance than CMOS and FPGA-based methods in terms of power consumption and area. The final simulation results demonstrate the validity of our proposed memristive system, and our system is easier to expand to other Natural Language Processing task because of its simplicity.

*Acknowledgements*

This work was supported by the Natural Science Foundation of China under Grants 61673187 and 61673188. This publication was made possible by NPRP grant: NPRP 8-274-2-107 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the author[s].

## REFERENCES

[1] 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.
[2] Z. Guo, J. Wang, and Z. Yan, "Global exponential synchronization of two memristor-based recurrent neural networks with time delays via static or dynamic coupling," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 2, pp. 235–249, 2015.
[3] S. Wen, R. Hu, Y. Yang, T. Huang, Z. Zeng, and Y.-D. Song, "Memristor-based echo state network with online least mean square," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–10, 2018.
[4] Z. Dong, C. Lai, D. Qi, Z. Xu, C. Li, and S. Duan, "A general memristor-based pulse coupled neural network with variable linking coefficient for multi-focus image fusion," *Neurocomputing*, vol. 308, pp. 172–183, 2018.
[5] P. Liu, Z. Zeng, and J. Wang, "Multistability of recurrent neural networks with nonmonotonic activation functions and mixed time delays," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 4, pp. 512-523, 2016.
[6] S. Wen, S. Xiao, Y. Yang, Z. Yan, Z. Zeng, and T. Huang, "Adjusting the learning rate of memristor-based multilayer neural networks via fuzzy method," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems,* DOI: 10.1109/TCAD.2018.2834436.

[7] T. M. Taha, R. Hasan, C. Yakopcic, and M. R. McLean, "Exploring the design space of specialized multicore neural processors," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–8.

[8] B. Belhadj, A. Joubert, Z. Li, R. Héliot, and O. Temam, "Continuous real-world inputs can open up alternative accelerator designs," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 1–12.

[9] A. Bala, A. Adeyemo, X. Yang, and A. Jabir, "Learning method for ex-situ training of memristor crossbar based multi-layer neural network," in *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2017 9th International Congress on*. IEEE, 2017, pp. 305–310.

[10] S. H. Jo, K.-H. Kim, and W. Lu, "High-density crossbar arrays based on a si memristive system," *Nano letters*, vol. 9, no. 2, pp. 870–874, 2009.

[11] A. M. Hassan, H. H. Li, and Y. Chen, "Hardware implementation of echo state networks using memristor double crossbar arrays," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 2171–2177.

[12] L. Xie, H. A. Du Nguyen, M. Taouil, S. Hamdioui, and K. Bertels, "A mapping methodology of boolean logic circuits on memristor crossbar," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 2, pp. 311–323, 2018.

[13] C. Yakopcic, R. Hasan, and T. M. Taha, "Memristor based neuromorphic circuit for ex-situ training of multi-layer neural network algorithms," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–7.

[14] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: programming 1t1m crossbar to accelerate matrix-vector multiplication," in *Proceedings of the 53rd annual design automation conference*. ACM, 2016, pp. 1–6.

[15] A. Bala, A. Adeyemo, X. Yang, and A. Jabir, "Learning method for ex-situ training of memristor crossbar based multi-layer neural network," in *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2017 9th International Congress on*. IEEE, 2017, pp. 305–310.

[16] C. Yakopcic, M. Z. Alom, and T. M. Taha, "Extremely parallel memristor crossbar architecture for convolutional neural network implementation," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 1696–1703.

[17] W. Zaremba and I. Sutskever, "Learning to execute," *arXiv preprint arXiv:1410.4615*, pp. 1410–4615, 2014.

[18] C. Yakopcic, R. Hasan, and T. M. Taha, "Memristor based neuromorphic circuit for ex-situ training of multi-layer neural network algorithms," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–7.

[19] M. Ansari, A. Fayyazi, A. Banagozar, M. Maleki, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Phax: Physical characteristics aware ex-situ training framework for inverter-based memristive neuromorphic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–12, 2017.

[20] F. Alibart, E. Zamanidoost, and D. B. Strukov, "Pattern classification by memristive crossbar circuits using ex situ and in situ training," *Nature communications*, vol. 4, pp. 131–140, 2013.

[21] R. Hasan and T. M. Taha, "Enabling back propagation training of memristor crossbar neuromorphic processors," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 21–28.

[22] D. Soudry, D. Di Castro, A. Gal, A. Kolodny, and S. Kvatinsky, "Memristor-based multilayer neural networks with online gradient descent training," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 10, pp. 2408–2421, 2015.

[23] C. Yakopcic, M. Z. Alom, and T. M. Taha, "Memristor crossbar deep network implementation based on a convolutional neural network," in *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016, pp. 963–970.

[24] G. Khodabandehloo, M. Mirhassani, and M. Ahmadi, "Analog implementation of a novel resistive-type sigmoidal neuron," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 4, pp. 750–754, 2012.

[25] J. Nowak, A. Taspinar, and R. Scherer, "LSTM recurrent neural networks for short text and sentiment classification," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2017, pp. 553–562.

[26] E. Cambria, "Affective computing and sentiment analysis," *IEEE Intelligent Systems*, vol. 31, no. 2, pp. 102–107, 2016.

[27] S. Rosenthal, N. Farra, and P. Nakov, "Semeval-2017 task 4: Sentiment analysis in twitter," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017, pp. 502–518.

[28] E. Cambria, D. Das, S. Bandyopadhyay, and A. Feraco, *A practical guide to sentiment analysis*. Springer, 2017, vol. 5.

[29] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics, 2011, pp. 142–150.

[30] J. Hltcoe, "Semeval-2013 task 2: Sentiment analysis in twitter," *Atlanta, Georgia, USA*, vol. 312, pp. 1–9, 2013.

[31] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[32] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, pp. 1–15, 2014.

[35] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[36] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 273–278.

[37] W. Zaremba and I. Sutskever, "Learning to execute," *arXiv preprint arXiv:1410.4615*, pp. 1–25, 2014.

[38] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015, pp. 3156–3164.

[39] Q. Qian, M. Huang, J. Lei, and X. Zhu, "Linguistically regularized lstms for sentiment classification," *arXiv preprint arXiv:1611.03949*, 2016.

[40] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[41] P.-H. Li and W.-Y. Ma, "Understanding and improving sequence-labeling ner with self-attentive LSTMs," pp. 1–11, 2018.

[42] A. Zadeh, P. P. Liang, S. Poria, P. Vij, E. Cambria, and L.-P. Morency, "Multi-attention recurrent network for human communication comprehension," *arXiv preprint arXiv:1802.00923*, 2018.

[43] N. Majumder, S. Poria, A. Gelbukh, M. S. Akhtar, E. Cambria, and A. Ekbal, "Iarm: Inter-aspect relation modeling with memory networks in aspect-based sentiment analysis," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3402–3411.

[44] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 mscoco image captioning challenge," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 4, pp. 652–663, 2017.

[45] W. Wen, Y. He, S. Rajbhandari, W. Wang, F. Liu, B. Hu, Y. Chen, and H. Li, "Learning intrinsic sparse structures within long short-term memory," *arXiv preprint arXiv:1709.05027*, pp. 1–14, 2017.

[46] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics, 2011, pp. 142–150.

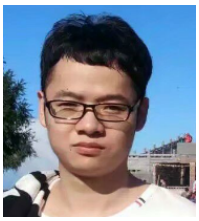
[47] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," *arXiv preprint arXiv:1309.4168*, pp. 1–10, 2013.

[48] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[49] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, pp. 1–19.

[50] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano letters*, vol. 10, no. 4, pp. 1297–1301, 2010.
[51] C. Chen, H. Ding, H. Peng, H. Zhu, Y. Wang, and C.-J. R. Shi, "Ocean: An on-chip incremental-learning enhanced artificial neural network processor with multiple gated-recurrent-unit accelerators," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 519–530, 2018.
[52] C. Chen, H. Ding, H. Peng, H. Zhu, R. Ma, P. Zhang, X. Yan, Y. Wang, M. Wang, H. Min *et al.*, "Ocean: An on-chip incremental-learning enhanced processor with gated recurrent neural network accelerators," in *ESSCIRC 2017-43rd IEEE European Solid State Circuits Conference*. IEEE, 2017, pp. 259–262.
[53] A. X. M. Chang, B. Martini, and E. Culurciello, "Recurrent neural networks hardware implementation on fpga," *arXiv preprint arXiv:1511.05552*, 2015.
[54] Y. Guan, Z. Yuan, G. Sun, and J. Cong, "Fpga-based accelerator for long short-term memory recurrent neural networks," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*. IEEE, 2017, pp. 629–634.
[55] L. Rahman, N. Mohammed, and A. K. Al Azad, "A new lstm model by introducing biological cell state," in *Electrical Engineering and Information Communication Technology (ICEEICT), 2016 3rd International Conference on*. IEEE, 2016, pp. 1–6.
[56] M. Z. Alom, A. T. Moody, N. Maruyama, B. C. Van Essen, and T. M. Taha, "Effective quantization approaches for recurrent neural networks," *arXiv preprint arXiv:1802.02615*, 2018.
[57] M. Hu, Y. Wang, Q. Qiu, Y. Chen, and H. Li, "The stochastic modeling of tio 2 memristor and its usage in neuromorphic system design," in *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*. IEEE, 2014, pp. 831–836.
[58] V. Srinivasan, D. W. Graham, and P. Hasler, "Floating-gates transistors for precision analog circuit design: an overview," in *Circuits and Systems, 2005. 48th Midwest Symposium on*. IEEE, 2005, pp. 71–74.
[59] G. Indiveri and T. K. Horiuchi, "Frontiers in neuromorphic engineering," *Frontiers in neuroscience*, vol. 5, pp. 1–2, 2011.
[60] J. Hasler and H. B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers in neuroscience*, vol. 7, pp. 1–29, 2013.
[61] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*. IEEE, 2014, pp. 10–14.
[62] A. Arora and V. Niranjan, "Low power filter design using memristor, meminductor and memcapacitor," in *Electrical, Computer and Electronics (UPCON), 2017 4th IEEE Uttar Pradesh Section International Conference on*. IEEE, 2017, pp. 113–117.
[63] J. Zhou, Y. Tang, J. Wu, and X. Yi, "Image segmentation with threshold based on memristors," in *Electronics Information and Emergency Communication (ICEIEC), 2013 IEEE 4th International Conference on*. IEEE, 2013, pp. 41–44.

**Shiping Wen** received the M. Eng. degree in Control Science and Engineering, from School of Automation , Wuhan University of Technology, Wuhan, China, in 2010, and received the Ph.D degree in Control Science and Engineering, from School of Automation, Huazhong University of Science and Technology, Wuhan, China, in 2013. He is currently an Associate Professor at School of Automation, Huazhong University of Science and Technology, and also in the Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, Wuhan, China. His current research interests include memristor-based circuits and systems, neural networks, and deep learning.



**Huaqiang Wei** received his B. Eng. degree from Hunan University, Changsha, China in 2017. He is currently working towards the M. Eng. degree from Huazhong University of Science and Technology, Wuhan, China. His research interests include memristor-based circuit, neural networks, deep learning and pattern recognition.

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

**Zhenyuan Guo** received the B.S. degree in mathematics and applied mathematics and the Ph.D. degree in applied mathematics from the College of Mathematics and Econometrics, Hunan University, Changsha, China, in 2004 and 2009, respectively.

He was a Joint Ph.D. Student to visit the Department of Applied Mathematics, University of Western Ontario, London, ON, Canada, from 2008 to 2009. From 2013 to 2015, he was a Post-Doctoral Research Fellow with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. He is currently a Professor with the College of Mathematics and Econometrics, Hunan University. His current research interests include theory of functional differential equations and differential equations with discontinuous right-hands, and their applications to dynamics of neural networks, memristive systems, and control systems.

**Zhigang Zeng** received his B.S. degree from Hubei Normal University, Huangshi, China, and his M.S. degree from Hubei University, Wuhan, China, in 1993 and 1996, respectively, and his Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China, in 2003. He is a professor in School of Automation, Huazhong University of Science and Technology, Wuhan, China, and also in the Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, Wuhan, China. His current research interests include neural networks, switched systems, computational intelligence, stability analysis of dynamic systems, pattern recognition and associative memories.

**Tingwen Huang** is a professor at Texas A & M University-Qatar. He received his B.S. degree from Southwest Normal University (now Southwest University), China, 1990, his M.S. degree from Sichuan University, China, 1993, and his Ph.D. degree from Texas A & M University, College Station, Texas, 2002. After graduated from Texas A & M University, he worked as a Visiting Assistant Professor there. Then he joined Texas A & M University at Qatar (TAMUQ) as an Assistant Professor in August 2003, then he was promoted to Professor in 2013. His research interests include neural networks based computational intelligence, distributed control and optimization, nonlinear dynamics and applications in smart grids. He has published more than three hundred peer-review reputable journal papers, including more than one hundred papers in IEEE Transactions. Currently, he serves as an associate editor for four journals including IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Cybernetics, and Cognitive Computation.

**Yiran Chen** received B.S and M.S. from Tsinghua University and Ph.D. from Purdue University in 2005. After five years in industry, he joined University of Pittsburgh in 2010 as Assistant Professor and then promoted to Associate Professor with tenure in 2014, held Bicentennial Alumni Faculty Fellow. He now is a tenured Associate Professor of the Department of Electrical and Computer Engineering at Duke University and serving as the co-director of Duke Center for Evolutionary Intelligence (CEI), focusing on the research of new memory and storage systems, machine learning and neuromorphic computing, and mobile computing systems. Dr. Chen has published one book and more than 300 technical publications and has been granted 93 US patents. He is the associate editor of IEEE TNNLS, IEEE TCAD, IEEE D&T, IEEE ESL, ACM JETC, ACM TCPS, and served on the technical and organization committees of more than 40 international conferences. He received 6 best paper awards and 14 best paper nominations from international conferences. He is the recipient of NSF CAREER award and ACM SIGDA outstanding new faculty award. He is the Fellow of IEEE.