01 Mar 2022

Memristor-Based HTM Spatial Pooler with On-Device Learning for Pattern Recognition

Xiaoyang Liu

Yi Huang

Zhigang Zeng

Donald C. Wunsch
Missouri University of Science and Technology, dwunsch@mst.edu

Recommended Citation

X. Liu et al., "Memristor-Based HTM Spatial Pooler with On-Device Learning for Pattern Recognition," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 3, pp. 1901 - 1915, Institute of Electrical and Electronics Engineers, Mar 2022.
The definitive version is available at https://doi.org/10.1109/TSMC.2020.3035612

Memristor-Based HTM Spatial Pooler With On-Device Learning for Pattern Recognition

Xiaoyang Liu, Yi Huang, *Graduate Student Member, IEEE*, Zhigang Zeng, *Fellow, IEEE*, and Donald C. Wunsch, II, *Fellow, IEEE*

***Abstract*—This article investigates hardware implementation of hierarchical temporal memory (HTM), a brain-inspired machine learning algorithm that mimics the key functions of the neocortex and is applicable to many machine learning tasks. Spatial pooler (SP) is one of the main parts of HTM, designed to learn the spatial information and obtain the sparse distributed representations (SDRs) of input patterns. The other part is temporal memory (TM) which aims to learn the temporal information of inputs. The memristor, which is an appropriate synapse emulator for neuromorphic systems, can be used as the synapse in SP and TM circuits. In this article, a memristor-based SP (MSP) circuit structure is designed to accelerate the execution of the SP algorithm. The presented MSP has properties of modeling both the synaptic permanence and the synaptic connection state within a single synapse, and on-device and parallel learning. Simulation results of statistic metrics and classification tasks on several real-world datasets substantiate the validity of MSP.**

***Index Terms*—Hierarchical temporal memory (HTM), memristor, neural networks, neuromorphic architecture, spatial pooler (SP).**

I. Introduction

HIERARCHICAL TEMPORAL MEMORY (HTM) is a brain-inspired algorithm that emulates the structure and functions of the neocortex, and can effectively process spatial and temporal information [1]. HTM has two core parts: 1) spatial pooler (SP) and 2) temporal memory (TM). SP processes the spatial part and generates efficient representations of the input, and TM processes the temporal part and learns time-based sequences. HTM can be applied in machine learning fields, such as pattern recognition [2], [3], speech processing [4], user support system [5], and anomaly detection [6]. HTM attracts many researchers' interests, both in the mathematical models [7]–[11] and applications [3], [4], [12], [13].

To implemented HTM more efficiently, many hardware implementations [14]–[27] for HTM are presented, including the memristor-based HTM/SP [18], [19], [21]–[24], [27], FPGA-based HTM/SP [16], [26], and on-volatile HTM (NVHTM) [20]. The memristor-based HTM is a promising topic because the memristor [28]–[30] is the appropriate element in emulating synapses in neural networks [31]–[45]. Memristor-based HTM has advantages of simple synapse structure and easily computing overlaps. In [18], a memristor-complementary metal–oxide semiconductor (CMOS) hybrid circuit for SP is designed and is applied to face recognition. In [19], an HTM computing block based on spin-neurons and the resistive crossbar is presented. It achieves more than 200× lower energy dissipation over the digital counterpart. In [21], a memristor crossbar-based SP is applied to face recognition and speech recognition. A new SP memristor crossbar is proposed in [22] to realize the SP operation, and the effectiveness is verified through the statistical metrics. In the memristor-based SP (MSP) circuit in [23], the calculation of the mean value of overlaps with memristors is introduced, and then the mean overlap value serves as the overlap threshold in the inhibition operation. In [27], an MSP circuit with on-device learning is proposed and achieved comparable classification results on several datasets.

However, there are some imperfections in previous hardware implementations of SP. In [21], synaptic permanence is adjusted one by one and the procedure of determining whether synapses should be potentiated or depressed is not performed on the device. In [21] and [22], the learning process is performed offline. In [23], there is no introduction to the learning process for SP. Most designs adopt the winner-take-all (WTA) operation to perform the inhibition operation [18], [21], [22], [27], which only selects one activated mini-column in one inhibition region, or uses comparison operations of which the comparison threshold is the average value of overlaps, to select winners [23]. SP circuits in [16] and [20] adopt global inhibition, instead of local inhibition operations. So far, SP designs with on-device learning are digital circuits [20], [26] or the digital–analog hybrid circuit [27].

Manuscript received June 1, 2020; accepted October 24, 2020. Date of publication December 3, 2020; date of current version February 17, 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0800402; in part by the Innovation Group Project of the National Natural Science Foundation of China under Grant 61821003; in part by the National Natural Science Foundation of China under Grant 61673188 and Grant 61761130081; in part by the Foundation for Innovative Research Groups of Hubei Province of China under Grant 2017CFA005; in part by the 111 Project on Computational Intelligence and Intelligent Control under Grant B18024; and in part by the Missouri University of Science and Technology Mary K. Finley Endowment and Intelligent Systems Center. This article was recommended by Associate Editor D. Hu. *(Corresponding author: Zhigang Zeng.)*

Xiaoyang Liu, Yi Huang, and Zhigang Zeng are with the School of Artificial Intelligence and 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 (e-mail: zgzeng@hust.edu.cn).

Donald C. Wunsch, II is with the Applied Computational Intelligence Laboratory, Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO 65401 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2020.3035612.

Digital Object Identifier 10.1109/TSMC.2020.3035612

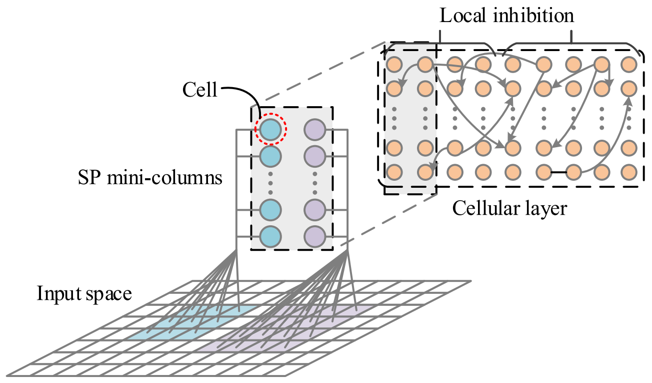


Fig. 1. Cellular layer in HTM [11]. A single cellular layer is composed of a set of mini-columns, and each mini-column contains a set of cells. Cells in a mini-column are connected to inputs through synapses shown as gray lines. There are also lateral connections between cells of different mini-columns, but they belong to TM and are not considered in this article.

A memristor-based analog SP (MSP) circuit with on-device learning is presented in this article to address the aforementioned problems. Specifically, the main contributions of this article are as follows.

1) Both synaptic permanence and synaptic connection of SP/HTM synapse are modeled within a single synapse.
2) The fixed-radius local inhibition is adopted rather than the global inhibition, and within one inhibition region, more than one mini-column is activated.
3) The presented MSP achieves on-device learning, and meanwhile, a weight update scheme is put forward to adjust the synaptic permanence and synaptic connection in parallel. These accelerate the training of SP.

The remainder of this article is organized as follows. The spatial pooler (SP) algorithm is introduced in Section II. The MSP circuit is presented in Section III. Simulations are carried out to substantiate the validity of the design in Section IV. Section V concludes this article.

II. SP Algorithm

A cellular layer in HTM is composed of a set of mini-columns, and each mini-column contains a set of cells, as shown in Fig. 1. There are synapses between input bits and mini-columns, like biophysical synapses, called proximal synaptic connection. In this article, the SP circuit is designed based on the one-layer SP.

The SP algorithm is composed of four phases: 1) initialization; 2) overlaps computation; 3) inhibition; and 4) learning [11], [22], [46].

A. Initialization

Receptive fields for mini-columns are determined. Each mini-column receives a set of input bits within one receptive field through synapses. Denote the synaptic permanence of the jth synapse in the ith mini-column by P_{ij}. Then, the corresponding synaptic connection state, which can be considered as the synaptic weight, is

$$W_{ij} = \begin{cases} 1, & \text{if } P_{ij} \geq P_\theta \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where P_θ is the synaptic permanence threshold, which is a value between 0 and 1 and is usually set to be 0.5. P_{ij} is a scalar value which is randomly initialized between 0 and 1. Therefore, if the synaptic permanence is above the threshold, the synapse is connected (denoted as On or 1); otherwise, it is not connected (denoted as Off or 0).

B. Overlaps Computation

The overlap of the ith mini-column is

$$o_i = b_i \sum_{j=1}^{C} W_{ij} x_{ij} \tag{2}$$

where $x_{ij} \in \{0, 1\}$ is the input bit belonging to the ith receptive field that is corresponding to the ith mini-column, and j is the jth synapse in the ith mini-column. C is the number of cells in one mini-column. Mini-columns can be divided into some inhibition regions depending on the topology of the input space. b_i is the boosting factor for the ith mini-column defined as [11]

$$b_i = e^{-\beta(\bar{a}_i(t) - <\bar{a}_i(t)>)} \tag{3}$$

where β is a positive number, $\bar{a}_i(t)$ is the activity of the ith mini-column, and $< \bar{a}_i(t) >$ is the activity of its neighbor mini-columns.

C. Inhibition

The activation states of mini-columns are determined by the overlaps within the same inhibition region. The overlap is calculated as

$$a_i = \begin{cases} 1, & \text{if } o_i \geq U(V(i), 1-s) \text{ and } o_i > o_\theta \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where 1 and 0 represent the activated state and inactivated state of the mini-column, respectively, $s \in (0, 1)$ is the activation density, $V(i)$ is the overlap values of neighbor mini-columns of the ith mini-column in the same inhibition region, and U is a percentile function that returns percentiles of values in the vector $V(i)$ for the percentage $(1-s) \times 100$. o_θ is the stimulus threshold. All a_i constitute sparse distributed representations (SDRs).

D. Learning

In the learning phase, the synaptic permanence is learned by the Hebbian rule [47] along with the adjustment of the synaptic connection state. Synapses of activated mini-columns are potentiated or depressed while those of inactivated mini-columns remain unchanged. That is

$$P_{ij} = \begin{cases} P_{ij} + p^+, & \text{if } a_i = 1 \text{ and the } j\text{th input bit is On} \\ P_{ij} - p^-, & \text{if } a_i = 1 \text{ and the } j\text{th input bit is Off} \\ P_{ij}, & \text{if } a_i = 0 \end{cases} \tag{5}$$

where p^+ is the potentiating value and p^- is the depressing value. Then, the synaptic connection states are set to 1 or reset to 0 through (1).

After the initialization, phases 2)–4) are repeated during the training process. The whole procedure of the SP algorithm is shown in Algorithm 1.

Algorithm 1 SP Algorithm

Input: $Z, N, s_{recp}, s_{inh}, \boldsymbol{x}^n$ $(n = 1, 2, \ldots, N), p^+, p^-$; //$Z$ and N are numbers of mini-columns and input samples, respectively, s_{recp} and s_{inh} are sizes of the receptive field and the inhibition region, receptively, and $\boldsymbol{x}^n$ is the input vector;

Output: SDRs $A = [\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^N] \in \mathbb{R}^{Z\times N}_{\{0,1\}}$;

```
1:  Initialize synaptic permanence $P_{ij}$ and connection state $W_{ij}$ according to (1);
2:  $n \leftarrow 1$;
3:  while $n \le N$ do
4:      $i \leftarrow 1$;
5:      while $i \le Z$ do
6:          Compute overlaps according to (2);
7:          $i \leftarrow i + 1$;
8:      end while
9:      Determine activated mini-columns of each inhibition region with size $s_{inh}$ according to (4) and obtain $\boldsymbol{a}^n$;
10:     $i \leftarrow 1$;
11:     while $i \le Z$ do
12:         if $a_i^n == 1$ then
13:             $\boldsymbol{p} = (p^+ + p^-)\bar{\boldsymbol{x}}_i^n - p^-$;    ▷ $\bar{\boldsymbol{x}}_i^n$, which is a part of $\boldsymbol{x}^n$, is a vector that contains input bits of the $i$th receptive field
14:             $\boldsymbol{P}_i \leftarrow \boldsymbol{P}_i + \boldsymbol{p}$; ▷ $\boldsymbol{P}_i$ is the synaptic permanence vector of the $i$th receptive field
15:             Compute $\boldsymbol{w}_i$ according to (1);    ▷ $\boldsymbol{w}_i$ is the synaptic connection state vector of the $i$th receptive field
16:             Decreasing $b_i$;
17:         else
18:             Increasing $b_i$;
19:         end if
20:         $i \leftarrow i + 1$
21:     end while
22:     $n \leftarrow n + 1$;
23: end while
```

III. MSP Circuit Structure

MSP is composed of several parts and they are introduced as follows.

A. Circuit Structure

The whole circuit is composed of several inhibition regions. The structure of one inhibition region is shown in Fig. 2. $V_{x1}^i, V_{x2}^i, \ldots, V_{xC}^i$ are voltage inputs which are corresponding to input bit values, where $i = 1, 2, \ldots, Z$ and Z is the number of mini-columns. $C = s_{\text{recp}}$ is the size of each receptive field, namely, the number of cells in each mini-column. $V_{d1}^i, V_{d2}^i, \ldots, V_{dC}^i$ are voltages values corresponding to On or Off states of input bit. TG is the transmission gate, kWTA is the k-winners-take-all unit, and time delay (TD) is the TD unit. Φ_s is the switch signal to switch inference and learning (or training) phases. Φ_{td} is the control signal of the TD unit. V_{po} and V_{de} are programming voltages to potentiate and depress synapses, respectively.

B. Main Components of MSP

1) Memristor Model and the Synapse Circuit: The memristor model adopted in this article is the VTEAM model [48] adjusted to fit the behavior of Ag/AgInSbTe/Ta (AIST) memristor [49], [50]. The positive threshold voltage of the memristor model is $V_{\text{on}} = 0.4$ V and the negative threshold voltage is $V_{\text{off}} = -0.4$ V. The minimum resistance is $R_{\text{on}} = 1$ kΩ and the maximum resistance is $R_{\text{off}} = 300$ kΩ. Voltages larger than V_{on} will decrease the memristor resistance and voltages smaller than V_{off} will increase the memristor resistance. The changing of the memristor resistance with the voltage beyond thresholds is shown in Fig. 3. Other types of memristor models with the threshold characteristic can also be adopted in the synapse.

A memristor-based synapse structure that is originally used as the dynamic synapse in [51] is introduced here to serve as the HTM synapse. It is composed of two memristors, M_1 and M_2, and one resistor R_a, as shown in Fig. 4. "+" and "−" are the positive and the negative ports of the memristor, respectively. p and m are the positive and negative ports of the synapse, respectively. This synapse has characteristics that the state of M_1 can be switched without affecting M_2 [51], and the state of M_2 (larger or smaller than a resistance threshold) determines whether the final state of M_1 is R_{on} or R_{off}. These characteristics make the synapse suitable for HTM. The state of M_1 represents the synaptic connection state and the resistance of M_2 represents the synaptic permanence. In the synapse, there is $R_a = R_{\text{on}}$, so that the parallel resistance of M_2 and R_a is smaller than R_{on}. Because $M_1 \in [R_{\text{on}}, R_{\text{off}}]$, the overall resistance of the synapse is mainly determined by M_1. Therefore, the synapse is connected when $M_1 = R_{\text{on}}$, while disconnected when $M_1 = R_{\text{off}}$.

2) Memristor Columns: In the memristor columns part in Fig. 2, one column represents one mini-column. V_{x1}^i, V_{x2}^i, $\ldots$, V_{xC}^i $(i = 1, 2, \ldots, Z)$ are voltage inputs to the ith mini-column in the inference phase, and the output of the ith mini-column is

$$V_o^i = \frac{\sum_{j=1}^{C} V_{xj}^i G_j^i}{G_{\text{se}}^i + \sum_{j=1}^{C} G_j^i} \tag{6}$$

where V_{xj}^i is the input to the jth synapse in the ith mini-column, G_j^i is the conductance of the jth synapse in the ith mini-column, and G_{se}^i is the conductance of the sense memristor M_{se}^i for the ith mini-column. Similar to [27], $1/(G_{\text{se}}^i + \sum_{j=1}^{C} G_j^i)$ approximately plays the role of the boosting factor. The output is then transported to the kWTA circuit to select winners.

3) Learning Signals Generation Module: The learning signals generation module receives V_{dj}^i's $(i = 1, 2, \ldots, Z,\ j = 1, 2, \ldots, C)$ which are the states of input bits, and d_i's $(i = 1, 2, \ldots, Z)$ which are the activation states of mini-columns, generating learning signals to memristor columns. V_{dj}^i determines which one of the programming voltages of the potentiation voltage V_{po} and the depression voltage V_{de}, to enter the jth synapse in the ith mini-column. If V_{dj}^i is On, V_{po} is selected; otherwise, V_{de} is selected. d_i determines whether the selected programming voltages, namely, the learning signals, are input to the ith mini-column. If the ith mini-column is activated, d_i is high level, and then learning signals are input to the ith mini-column; otherwise, no learning signal is input. V_{po} and V_{de} can be generated by the programming voltage generation module which will be introduced later.

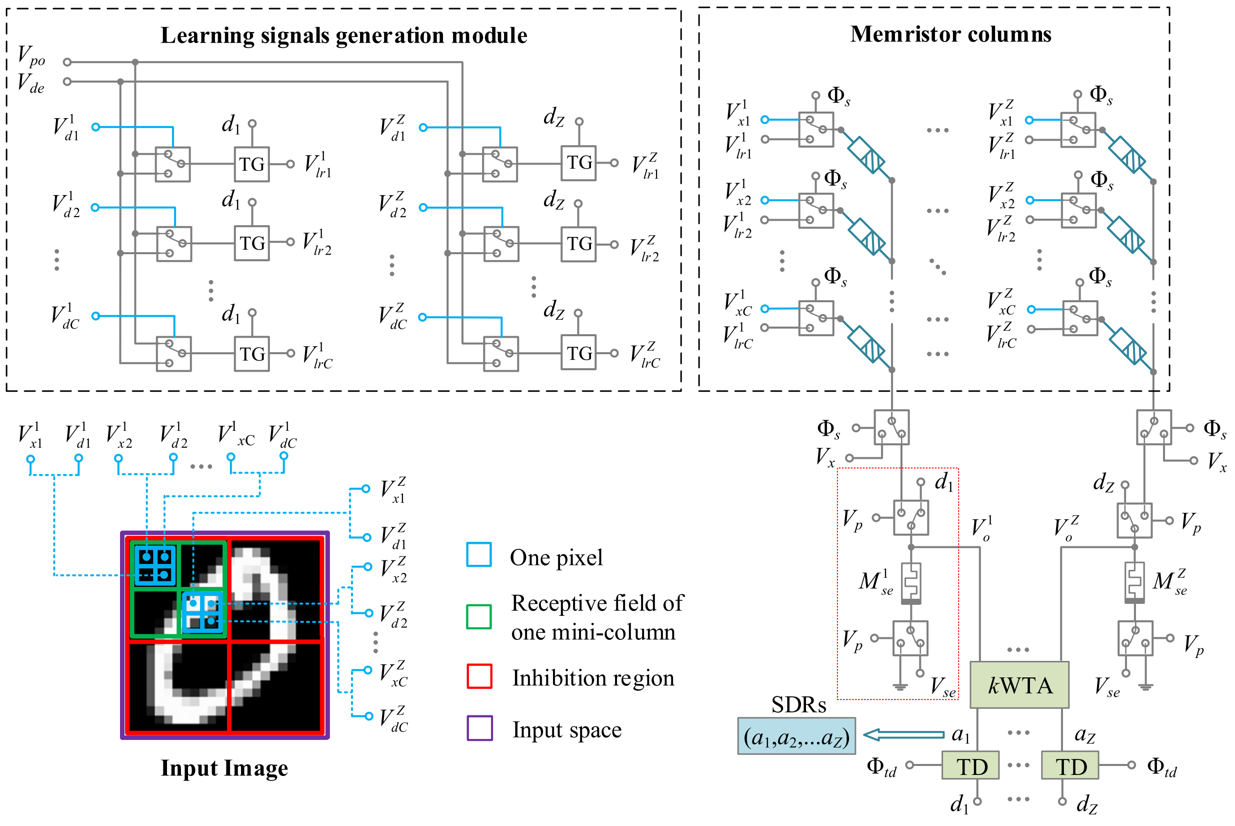


Fig. 2. Structure of one inhibition region of MSP. It is composed of memristor columns, learning signals generation module, and other parts. The receptive fields in the input image are allowed to overlap. Pixel values in one inhibition region of the image are input to this circuit. Each pixel is converted to two kinds of voltages V_x and V_d, which represent the pixel value and the On or Off state of the pixel, respectively. The learning signals generation module determines which one of the two programming voltages $V_{\rm po}$ and $V_{\rm de}$ should be input to memristor columns to modify the synapse, or no programming voltage should be input. The memristor columns receive input signals in the inference phase and learning signals in the learning phase. In the inference phase, outputs of memristor columns are input to kWTA and TD parts in turn to generate SDRs and control signals $d_1, \ldots, d_Z$ which are then used in the learning signals generation module to generate learning signals. The part in the red dashed box approximately plays the role of the boosting factor. SDRs of all inhibition regions constitute the final SDRs. In the learning phase, memristor resistance is adjusted by learning signals.

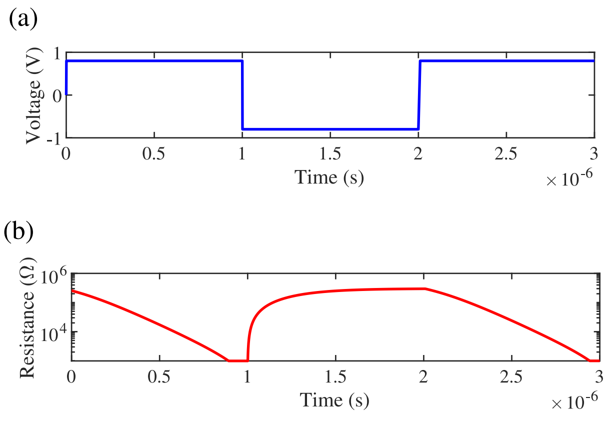


Fig. 3. Changing of the memristor resistance with the applied voltage. (a) Applied voltage. (b) Changing of the memristor resistance which is expressed on a logarithmic scale.

4) Programming Voltage Generation Module: The programming voltages $V_{\rm po}$ and $V_{\rm de}$ each can be generated by the generation module shown in Fig. 5. V_1, V_2, V_3, and V_4 are constant voltages with different amplitudes. PWL_1–PWL_4 are periodic piecewise linear (PWL) voltages with different duty cycles but the same period. The four PWL voltages control the opening and closing of TGs. The module generates a PWL voltage $V_{\rm po}$ or $V_{\rm de}$ which contains varying amplitudes and pulse widths.

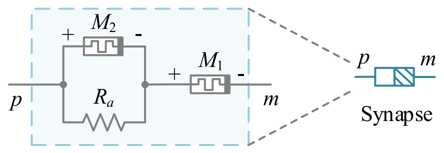


Fig. 4. Synapse circuit. It is composed of two memristors, M_1 and M_2, and one resistor R_a of which the resistance is $R_{\rm on}$. "+" and "-" are the positive and the negative ports of the memristor, respectively. p and m are the positive and negative ports of the synapse, respectively.

5) Boosting Factor: The learning process of the boosting factor is shown in Fig. 6. It is simply related to the activation frequency of the corresponding mini-column, ignoring the activity of its neighbors. That is, if the ith mini-column is activated, the ith boosting factor decreases; otherwise, it increases. V_p is a PWL voltage with a small pulse width. d_i is the output of the ith TD unit, and "1" represents the high level and "0" represents the low level, respectively. $V_{\rm se}$ is a positive voltage which satisfies $-V_{\rm se} < V_{\rm off} < 0$ and $d_i - V_{\rm se} > V_{\rm on} > 0$. At the beginning of the learning phase, if the ith mini-column is activated, the conductance of $M_{\rm se}^i$ increases and V_o^i in (6) decreases; otherwise, V_o^i increases.

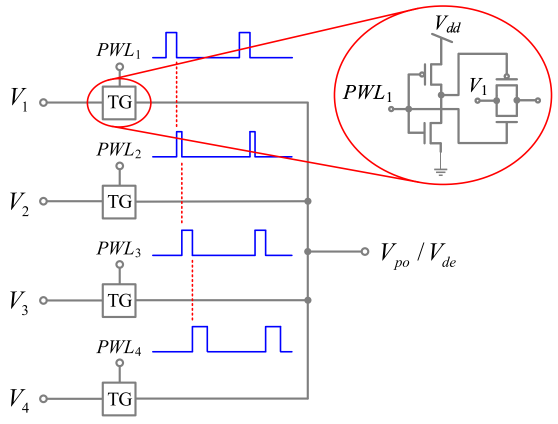


Fig. 5. Programming voltage generation module. Four periodic PWL voltages PWL_1–PWL_4 with different duty cycles but the same period are applied to the control ports of TGs, respectively, controlling the applying of V_1–V_4 which are constant voltages with different amplitudes.

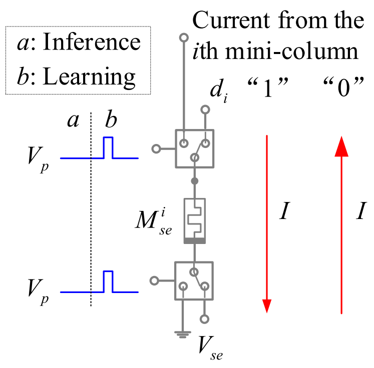


Fig. 6. Learning process of the boost factor. In the inference phase, the sense memristor M^i_{se} is in series with the ith mini-column, and its other port is grounded. In the learning phase, M^i_{se} is connected to d_i and V_{se}. $-V_{\text{se}} < V_{\text{off}} < 0$ and $d_i - V_{\text{se}} > V_{\text{on}} > 0$. During the high-level period of V_p, the conductance of M^i_{se} is increased if d_i is high level and decreased if d_i is low level.

6) kWTA Unit: The kWTA operation is to determine k largest elements in K real numbers [52]–[56], where $1 \le k < K$. Different from the SP circuits that use WTA circuits to select the largest overlap, or use sorting operations to select multiple winners in an inhibition region, in MSP, a kWTA circuit [53] is adopted to select multiple mini-columns in an inhibition region. The adopted kWTA circuit has the property that if the number of nonzero inputs is less than desired winners, it only determines nonzero inputs as winners, avoiding selecting inactivated mini-columns as activated ones.

7) Time Delay Unit: In the learning phase, there is no input pattern relevant activations of mini-columns, so it is necessary to store activation states of mini-columns in the inference phase. The TD unit is designed to achieve this storage function. As soon as the kWTA unit determines winners, the winners are fed into TD units and stored in them. The TD unit is designed based on the integrate-and-fire (I&F) neuron circuit [57], as shown in Fig. 7. When Φ_{td} is low level and V_{in} is high level, the capacitor C_1 is charged. Once the voltage across C_1 exceeds the threshold of the inverter, V_{out} becomes high level and C_2 further raises the voltage across C_1. If V_{in} is low level, V_{out} is also low level. Therefore, V_{out} stores the state of V_{in}. When Φ_{td} becomes high level, the transistor T_1 is turned on, and C_1 and C_2 begin to discharge. Once the voltage across C_1 is lower than the threshold of the inverter, V_{out} becomes low level. The discharging rate can be controlled by V_c.

SPICE simulation result of the TD unit is shown in Fig. 8. Φ_{td} is the control signal, V_{in} receives the activation state of one mini-column, namely, a_i $(i = 1, 2, \ldots, Z)$ in Fig. 2, and V_{out} is the stored activation state which is corresponding to d_i in Fig. 2. For a better explanation, Fig. 8(a) is divided into three parts (1)–(3) by red dashed lines, and the same parts exist in Fig. 8(b) and (c). In part (1), Φ_{td} is low level and V_{in} is high level, resulting in high level of V_{out}. This process

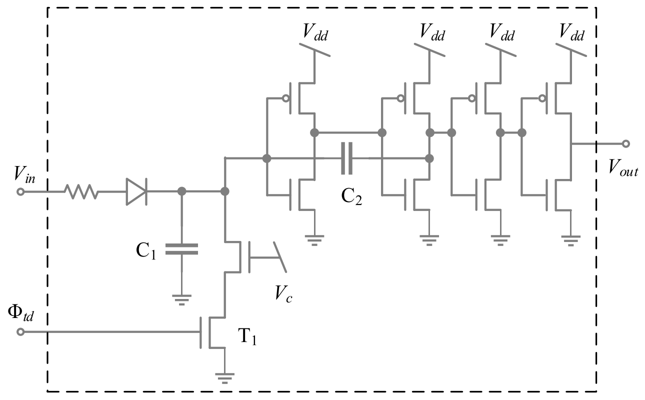


Fig. 7. TD unit. The function of the TD unit is to temporarily store the activation state of each mini-column. V_{in} is the input of the TD unit, which receives one of the outputs of the kWTA unit, namely, a_i $(i = 1, 2, \ldots, Z)$ in Fig. 2. V_{out} is the output of the TD unit, which outputs d_i in Fig. 2. When Φ_{td} is low level, V_{out} will follow the state of V_{in}. When Φ_{td} is high level, V_{out} is reset to the initial state.

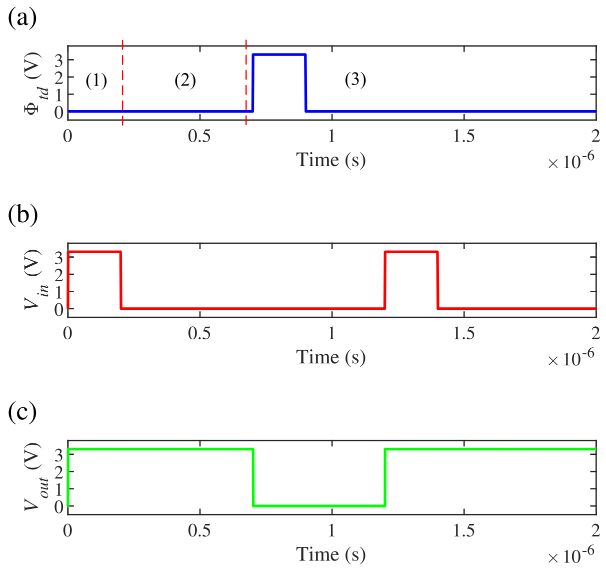


Fig. 8. Simulation result of the TD unit. (a) Discharging control signal Φ_{td}. (b) Input voltage V_{in} is the activation state of the mini-column, namely, a_i $(i = 1, 2, \ldots, Z)$ in Fig. 2. (c) Output voltage V_{out} represents the temporarily stored activation state and is corresponding to d_i in Fig. 2. In part (1), Φ_{td} is low level and V_{in} is high level, resulting in the high level of V_{out}, and V_{out} is always high level in part (2). This process means that when Φ_{td} is low level and the activation state is "1," the output of the TD unit is "1," and it keeps "1" until it is reset to "0" by setting Φ_{td} to high level in part (3). Then, the TD unit starts the next work cycle once Φ_{td} declines to low level.

means that when the activation state is "1," the output of the TD unit is "1," and it keeps "1" in part (2). In part (3), the stored activation state is freed by high level of Φ_{td}, and then the TD unit starts the next work cycle after Φ_{td} becoming low level again.

C. Updating Procedures of the Synapse Circuit

In the memristor-based synapse circuit for HTM, the resistance of M_2 represents the synaptic permanence and the state of M_1 represents the synaptic connection state. The synapse is connected when $M_1 = R_{\text{on}}$ and disconnected when $M_1 = R_{\text{off}}$. If the resistance of M_2 is larger than the threshold value R_θ, the synaptic permanence value is larger than P_θ and the resistance of M_1 will be set to its smallest value R_{on}, otherwise R_{off}. Since the synaptic permanence value is limited to [0, 1], $M_2 = R_{\text{on}}$ means that the synaptic permanence value is 0 and $M_2 = R_{\text{off}}$ means that the permanence value is 1. Known $R_a = R_{\text{on}}$, voltages across M_1 and M_2 are

$$V_{M_1} = \frac{M_1 R_{\text{on}} + M_1 M_2}{M_1 R_{\text{on}} + M_2 R_{\text{on}} + M_1 M_2} \cdot V_{\text{pm}} \tag{7}$$

$$V_{M_2} = \frac{M_2 R_{\text{on}}}{M_1 R_{\text{on}} + M_2 R_{\text{on}} + M_1 M_2} \cdot V_{\text{pm}} \tag{8}$$

respectively, where V_{pm} is the voltage across the synapse. Details of the updating procedure of the synapse and how to set values of V_{pm} in each step are as follows.

Step 1: Set M_1 to R_{on}. In order to set M_1 to R_{on} without influencing M_2, the voltage across M_1 should be higher than the positive threshold voltage V_{on} and the voltage across M_2 should be lower than V_{on}, that is

$$V_{M_1} > V_{\text{on}} \tag{9}$$

$$V_{M_2} < V_{\text{on}}. \tag{10}$$

Therefore

$$\left(1 + \frac{M_2 R_{\text{on}}}{M_1 R_{\text{on}} + M_1 M_2}\right) \cdot V_{\text{on}} < V_{\text{set}} < \left(1 + \frac{M_1 R_{\text{on}} + M_1 M_2}{M_2 R_{\text{on}}}\right) \cdot V_{\text{on}} \tag{11}$$

where V_{set} is V_{pm} in step 1. The boundary condition for (11) is that $M_1 = R_{\text{on}}$ and $M_2 = R_{\text{off}}$. The resistance of M_2 rarely reaches R_{off} in this design, so $V_{\text{set}} = 2V_{\text{on}}$ is appropriate.

Step 2: Increase or decrease M_2. M_1 is already R_{on} after step 1. To increase the resistance of M_2

$$V_{M_2} = \frac{M_2}{R_{\text{on}} + 2M_2} \cdot V_{\text{inc}} < V_{\text{off}}. \tag{12}$$

To decrease the resistance of M_2

$$V_{M_2} = \frac{M_2}{R_{\text{on}} + 2M_2} \cdot V_{\text{dec}} > V_{\text{on}}. \tag{13}$$

The boundary condition is that $M_2 = R_{\text{on}}$, so

$$V_{\text{inc}} < 3V_{\text{off}} \tag{14}$$

$$V_{\text{dec}} > 3V_{\text{on}}. \tag{15}$$

V_{inc} and V_{dec} result in p^+ and p^- in (5), respectively.

Step 3: Set M_1 to R_{on} again as step 1.

TABLE I
UPDATING PROCEDURES OF THE INTRODUCED SYNAPSE

Step	Operation	Symbol
1	Set M_1 to R_{on}	V_{set}
2	Increase M_2 / Decrease M_2	V_{inc} / V_{dec}
3	Set M_1 to R_{on}	V_{set}
4	Keep M_1 at R_{on} or reset M_1 to R_{off}	V_{setx}

Step 4: Keep M_1 at R_{on} or reset M_1 to R_{off} according to the state of M_2. If M_2 is larger than R_θ, the negative voltage V_{M_1} is higher than V_{off}, and M_1 will remain R_{on}, corresponding to the connected state of the synapse. If M_2 is smaller than R_θ, V_{M_1} is lower than V_{off}, and then M_1 will be reset to R_{off}, corresponding to the disconnected state of the synapse. Meanwhile, V_{M_2} does not exceed V_{off} so M_2 will not be changed. When $M_1 = R_{\text{on}}$, the boundary voltage across M_1 is

$$V_{M_1} = \frac{R_{\text{on}} + R_\theta}{R_{\text{on}} + 2R_\theta} \cdot V_{\text{setx}} = V_{\text{off}} \tag{16}$$

where V_{setx} is the voltage across the synapse in this step and

$$R_\theta = P_\theta (R_{\text{off}} - R_{\text{on}}) + R_{\text{on}} = \kappa R_{\text{on}} \tag{17}$$

$$\kappa = P_\theta \left(\frac{R_{\text{off}}}{R_{\text{on}}} - 1\right) + 1. \tag{18}$$

So

$$V_{\text{setx}} = \frac{1 + 2\kappa}{1 + \kappa} \cdot V_{\text{off}}. \tag{19}$$

At the same time

$$V_{M_2} = \frac{\kappa}{1 + 2\kappa} \cdot V_{\text{setx}} = \frac{\kappa}{1 + \kappa} \cdot V_{\text{off}} > V_{\text{off}}. \tag{20}$$

Since V_{off} is negative, M_2 will not be influenced. Therefore, M_1 will be reset to R_{off} when $M_2 < R_\theta$ and remain R_{on} when $M_2 > R_\theta$ under the condition (19).

The pulse widths of the above voltages are determined by simulations according to properties of the memristor. The updating procedures are also summarized in Table I.

D. Updating Procedure of the Mini-Column Circuit

The proximal synaptic connections of one mini-column are shown in Fig. 9(a). V_{x1}^i, V_{x2}^i,…,V_{xC}^i $(i = 1, 2, \ldots, Z)$ are inputs to the mini-column in the inference phase. Fig. 9(b) shows programming voltages to the mini-column in the learning phase. Here, a weight update scheme is put forward to update the synaptic permanence and the synaptic connection state in parallel. V_{po} is the voltage to potentiate the synapses that need to be potentiated (hereinafter referred to as po-synapses) and V_{de} is the voltage to depress the synapses that need to be depressed (hereinafter referred to as de-synapses). V_{po} and V_{de} are both periodic PWL voltages with varying amplitudes, respectively. There are three stages to complete one learning operation. The learning procedure and applied voltages of the mini-column circuit are detailed below and are summarized in Table II.

Stage 1: Reset M_1 in all synapses to R_{off}. Through grounding all rows and applying $-V_{\text{reset}}$ to the column terminal, that

TABLE II
LEARNING PROCEDURE AND APPLIED VOLTAGES OF THE MINI-COLUMN

Stage	Operation	Step	Sub-operation	V_{po}	V_{de}	V_x
1	Reset all M_1's to R_{off}	-	-	0	0	$-V_{reset}$
2	Potentiate po-synapses	1	Set M_1's in po-synapses to R_{on}	V_{set}	0	0
		2	Increase M_2's in po-synapse	V_{inc}	0	0
		3	Set M_1's in po-synapses to R_{on}	V_{set}	0	0
		4	Keep M_1's in po-synapses at R_{on} or reset M_1's to R_{off} according to states of M_2's	V_{setx}	0	0
3	Depress de-synapses	1	Set M_1's in de-synapses to R_{on}	0	V_{set}	0
		2	Decrease M_2's in de-synapse	0	V_{dec}	0
		3	Set M_1's in de-synapses to R_{on}	0	V_{set}	0
		4	Keep M_1's in de-synapses at R_{on} or reset M_1's to R_{off} according to states of M_2's	0	V_{setx}	0

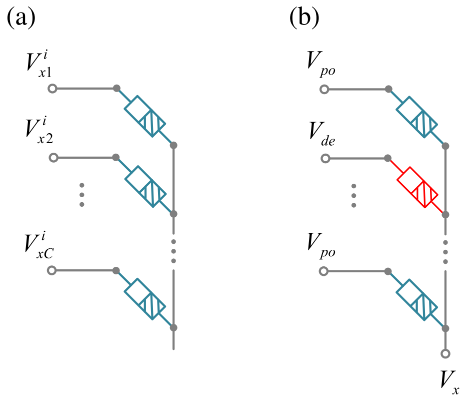


Fig. 9. (a) Proximal synaptic connections of one mini-column. V^i_{x1}, V^i_{x2}, $\ldots$, V^i_{xC} ($i = 1, 2, \ldots, Z$) are input voltages in the inference phase. (b) In the learning phase, learning voltages applied to rows of the mini-column. The second synapse (the red one) needs to be depressed and other synapses (blue synapses) need to be potentiated. V_{po} and V_{de} are the potentiation and depression voltage, respectively.

is, $V_{\text{po}} = V_{\text{de}} = 0$, and $V_x = -V_{\text{reset}}$, M_1 in all synapses are reset to R_{off}. $-V_{\text{reset}}$ satisfies that the voltage across M_1 is lower than V_{off} and the voltage across M_2 is higher than V_{off} (noted that $V_{\text{off}} < 0$), therefore similar to (11), an appropriate value for V_{reset} is $2V_{\text{off}}$.

Stage 2: Potentiate po-synapses. Stage 2 can be divided into four steps, corresponding to the four steps in Table I. $V_{\text{de}} = 0$ and $V_x = 0$ in all steps, so voltages across de-synapses are all zero, and then the resistance of memristors in de-synapses remains unchanged. In the first step, $V_{\text{po}} = V_{\text{set}}$. Voltages across po-synapses are all $V_{\text{po}} - V_x = V_{\text{set}}$. Then, M_1's in po-synapses are set to R_{on}. In the second step, $V_{\text{po}} = V_{\text{inc}}$, so the resistance of M_2's in po-synapses is increased. The third step is the same as the first step. In the fourth step, $V_{\text{po}} = V_{\text{setx}}$, therefore M_1's in po-synapses are kept in R_{on} or reset to R_{off} according to the resistance of M_2's.

Stage 3: Depress de-synapses. Stage 3 can also be divided into four steps, which are also corresponding to steps in Table I. $V_{\text{po}} = 0$ and $V_x = 0$ in all steps, so voltages across po-synapses are all zero, and then the resistance of memristors in po-synapses remains unchanged. In the first step, $V_{\text{de}} = V_{\text{set}}$, of which the principle is the same as the first step in Stage 2. In the second step, $V_{\text{de}} = V_{\text{dec}}$, and this results in the decrease of the resistance of M_2's in de-synapses. The third step is the same as the first step. In the fourth step, $V_{\text{de}} = V_{\text{setx}}$, and the principle is the same as it in Stage 2.

The SPICE simulation result is shown in Fig. 10. Fig. 10(a) and (c) shows the synaptic connection state and the synaptic permanence of the po-synapse, respectively. The synaptic permanence is increased, and because it is larger than the synaptic permanence threshold P_θ, the synaptic connection state is set to On finally. Fig. 10(b) and (d) shows the synaptic connection state and the synaptic permanence of the de-synapse, respectively. The synaptic permanence of the de-synapse is decreased, and because the synaptic permanence is smaller than P_θ, the synaptic connection state is set to Off finally. Fig. 10(e)–(g) shows V_{po}, V_{de}, and V_x, respectively. During the increase period of the synaptic permanence of the po-synapse, the de-synapse is not affected, and during the decrease period of the synaptic permanence of the de-synapse, the po-synapse is not affected. The simulation result shows that the synaptic permanence of po-synapse and de-synapse is correctly increased and decreased, respectively, and corresponding synaptic connection stages are correctly set. Through this scheme, po-synapses or de-synapses are adjusted in parallel.

E. Working Procedure of the MSP Circuit

The working procedure of the MSP circuit is divided into two phases, which are the inference phase and the learning phase. The learning phase includes stages introduced in Section III-D.

The main signals of one working cycle are drawn in Fig. 11. After randomly initializing all synapses, the procedure of one working cycle is as follows.

1) The inference phase starts with Φ_s being high level.
2) High level of Φ_{td} resets TD units and $d_1, d_2, \ldots, d_Z$ are reset to low levels.
3) V^i_{x1}, $V^i_{x2}, \ldots, V^i_{xC}$ of each receptive field which represent values of input bits, and V^i_{d1}, $V^i_{d2}, \ldots, V^i_{dC}$ which represent On or Off states of input bits, are applied to the circuit.

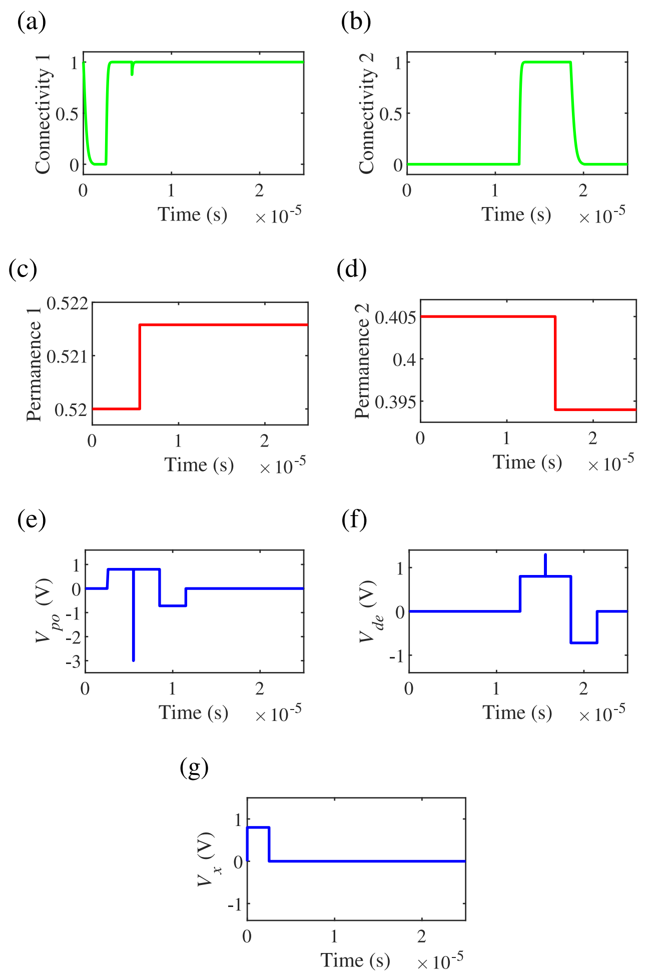


Fig. 10. Simulation result of the weight update scheme. (a) and (b) Connectivity 1 is the synaptic connection state of the po-synapse and Connectivity 2 is that of the de-synapse, where po-synapse and de-synapse represent the synapse that needs to be potentiated and the synapse that needs to be depressed, respectively. The final connectivities of the po-synapse and the de-synapse are "1" and "0," respectively. "0" and "1" represent disconnected and connected states, respectively, and they are corresponding to R_{off} and R_{on} of M_1, respectively. (c) and (d) Permanence 1 is the synaptic permanence of the po-synapse, which is defined by the resistance of M_2. Permanence 2 is the synaptic permanence of the de-synapse. (e)–(g) Voltages V_{po}, V_{de}, and V_x.

4) Winners that are stable activated mini-columns of each inhibition region are obtained. States of mini-columns $a_1, a_2, \ldots, a_Z$ of all inhibition regions constitute SDRs of the input pattern.
5) $d_1, d_2, \ldots, d_Z$ are determined high or low levels by $a_1, a_2, \ldots, a_Z$ through TD units.
6) Φ_s becomes low level, and then the learning phase starts. Inputs to memristor columns are switched from input bits of receptive fields to learning signals. If V^i_{dj} $(i = 1, 2, \ldots, Z,\ j = 1, 2, \ldots, C)$ is high level, the corresponding switch selects V_{po}; otherwise, it selects V_{de}. If d_i is high level, V_{po} or V_{de} is input to synapses through TGs; otherwise, no programming voltage is input. Meanwhile, column terminals are switched to V_x, and M^i_{se} is switched to d_i and V_{se}.

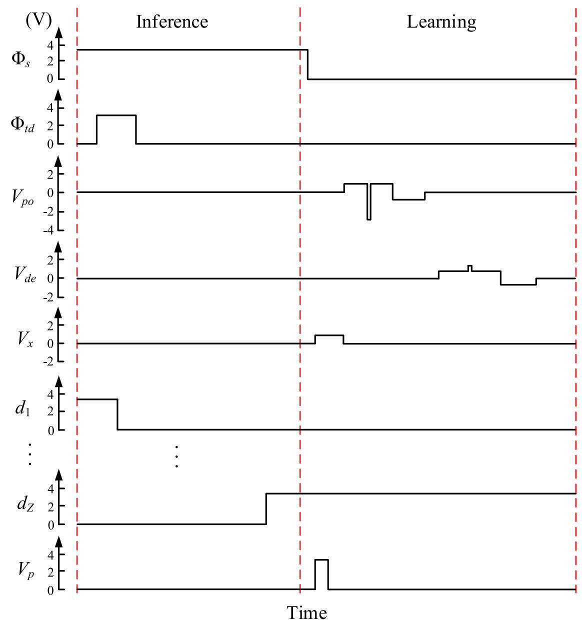


Fig. 11. Main signals of one working cycle of the MSP circuit. In the inference phase, at the beginning Φ_{td} resets $d_1, \ldots, d_Z$ to low level. Then input voltages from each receptive field are applied to MSP, and MSP determines activated mini-columns, such as d_Z. In the learning phase, V_p has a pulse to update the sense memristor, and programming voltages V_{po}, V_{de}, V_x work together to adjust the synaptic permanence and synaptic connection states.

TABLE III
SIMULATION PARAMETERS

Signal	Amplitude (V)	Width (us)
V_{reset}	-0.8	2.5
V_{set}	0.8	3
V_{inc}	-3.0	0.01
V_{dec}	1.3	0.01
V_{setx}	-0.72	3

IV. RESULTS AND ANALYSIS

A. Simulation Setting

Image pixels are converted to two kinds of binary voltages values: one is {0, 0.2 V} which represents the input bit value, and the other is {0, 3.3 V} which represents the On or Off state of the input bit. $R_{\text{on}} = 1$ kΩ and $R_{\text{off}} = 300$ kΩ. Other parameters are shown in Table III, and they are related to the type of the memristor. The pulse widths of PWL_1, PWL_2, PWL_3, and PWL_4 in Fig. 5 are equal to the durations of V_{set}, $V_{\text{inc}}/V_{\text{dec}}$, V_{set}, and V_{setx}, respectively. Amplitudes of constant voltages V_1, V_2, V_3, and V_4 in Fig. 5 are equal to the amplitudes of V_{set}, $V_{\text{inc}}/V_{\text{dec}}$, V_{set}, and V_{setx}, respectively. MSP is validated through the circuit simulation on SPICE. Then the hardware simulation is transferred to MATLAB under hardware defined constraints to perform applications.

B. Validity of MSP and the Weight Update Scheme

A simulation is performed on SPICE to substantiate the validity of MSP and the weight update scheme. Fig. 12

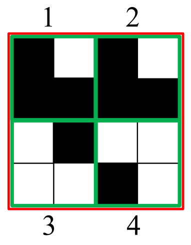


Fig. 12. 4×4 input pattern. The input pattern is input to one inhibition region. There are four receptive fields 1, 2, 3, and 4 in four green boxes. Each receptive field contains four pixels. Black and white pixels represent “1” and “0” respectively.

shows the 4×4 input pattern. The input pattern is input to one inhibition region. Fields 1–4 surrounded by green boxes represent the first to the fourth receptive fields, which are connected to four mini-columns, respectively. Each mini-column is connected to four pixels through four synapses.

The simulation result is shown in Fig. 13. Fig. 13(a) is the switch signal to switch inference and learning phases. The inference phase and the learning phase are depicted in Fig. 13(b) and (c), respectively. Fig. 13(b) shows the overlaps (or outputs) V_o^1, V_o^2, V_o^3, and V_o^4 of four mini-columns, the outputs a_1, a_2, a_3, and a_4 of the kWTA unit, and the outputs d_1, d_2, d_3, and d_4 of TD units. Outputs of mini-columns pass through the kWTA unit and TD units in turn. It can be seen that the first and the second mini-columns are activated. Fig. 13(c) shows the potentiation and depression of the synaptic permanence and the adjustment of synaptic connection states of the second mini-column which corresponding to V_o^2, a_2, and d_2. Because the first, the third, and the fourth input bits of the second mini-column are On, the corresponding synaptic permanence is potentiated, while the second input bit is Off, the corresponding synaptic permanence is depressed. The synaptic permanence of the second and the fourth synapse is lower than the permanence threshold P_θ, so the corresponding synaptic connection states are reset to 0, and those of others are set to 1.

To evaluate the acceleration capacity of MSP, an SP run on software on the MATLAB platform is taken as a comparison. The time consumed of the software execution of one cycle of the SP algorithm is about 1.4×10^{-2} s, and the MSP hardware simulation on SPICE is about 4.1×10^{-5} s, so the presented hardware design of SP is about 341× faster than the software counterpart. MATLAB and SPICE are both run on the computer with i7 CPU and Windows 10 operating system. When the scale becomes larger, MSP will be more efficient because of the parallelism of the circuit.

C. Statistic Metrics of MSP

There are several statistical metrics to qualify the performance of the SP algorithm [11], and these metrics are used in this article to qualify MSP. One hundred training samples and 100 testing samples are randomly selected from the MNIST (Modified National Institute of Standards and Technology) [58] dataset to calculate metrics. MNIST is a handwriting digital dataset containing 28×28 images for pattern classification tasks. Training samples are used to train MSP, and test samples are used to calculate metrics.

1) Sparseness: The sparseness metric qualifies the ability to achieve a fixed sparsity of SP. The sparseness is calculated as [11]

$$s^t = \frac{1}{Z}\sum_{i=1}^{Z} a_i^t \tag{21}$$

where Z is the total number of mini-columns, and a_i^t is the activation state of the ith mini-column at time step t. The result is shown in Fig. 14. Sparseness of input samples is mainly among about 3%–26% and those of SDRs obtained by MSP are all about 2%. Therefore, MSP effectively achieves sparsity.

2) Entropy: The entropy metric reflects the effective utilization of all mini-columns, and the bigger, the better. The entropy of SP is calculated as [11]

$$E = \sum_{i=1}^{Z}\left[-P(a_i)\log_2 P(a_i) - (1-P(a_i))\log_2(1-P(a_i))\right] \tag{22}$$

where

$$P(a_i) = \frac{1}{N}\sum_{t=1}^{N} a_i^t. \tag{23}$$

$P(a_i)$ is the average activation frequency of the ith mini-column and N is the total number of input samples. Entropies of MSP before training and after training are shown in Fig. 15. It can be seen that after training, the entropy increases.

3) Noise Robustness: The noise robustness metric qualifies the change of the outputs of SP with the change of noise levels of input samples, and the bigger, the better. When adding noise to input samples, to maintain the sparsity, r ($r \in (0, 100)$) percent On input bits of each input sample are flipped to Off, and the same number Off bits are flipped to On [11]. The robustness is calculated as [11]

$$R = \frac{1}{N}\sum_{n=1}^{N}\int_{r=0}^{100} \frac{\|\boldsymbol{a}^n \circ \hat{\boldsymbol{a}}^n(r)\|_0}{\|\boldsymbol{a}^n\|_0} \tag{24}$$

where N is the number of input samples, r is the noise level, $\boldsymbol{a}^n$ is the overlap vector without input noise, $\boldsymbol{a}^n(r)$ is the overlap vector with r percent noise level, and $\|\cdot\|_0$ is the L_0-norm. $([\|\boldsymbol{a}^n \circ \hat{\boldsymbol{a}}^n(r)\|_0]/[\|\boldsymbol{a}^n\|_0])$ computes the fraction of shared activated mini-columns of $\boldsymbol{a}^n$ and $\boldsymbol{a}^n(r)$. The fractions of shared activated mini-columns with varying noise levels before and after training are shown in Fig. 16(a) and the noise robustness before and after training are shown in Fig. 16(b). It can be seen that the fraction curve after training is always above that before training and the noise robustness after training is bigger than that before training.

D. Classification Results

Classification tasks are carried out to substantiate the validity of SDRs extracted by MSP. The classification tasks are carried out on MNIST, Fashion-MNIST [59], and CIFAR-10 [60] datasets. MNIST and Fashion-MNIST both contain 60 000 training samples and 10 000 testing samples, and

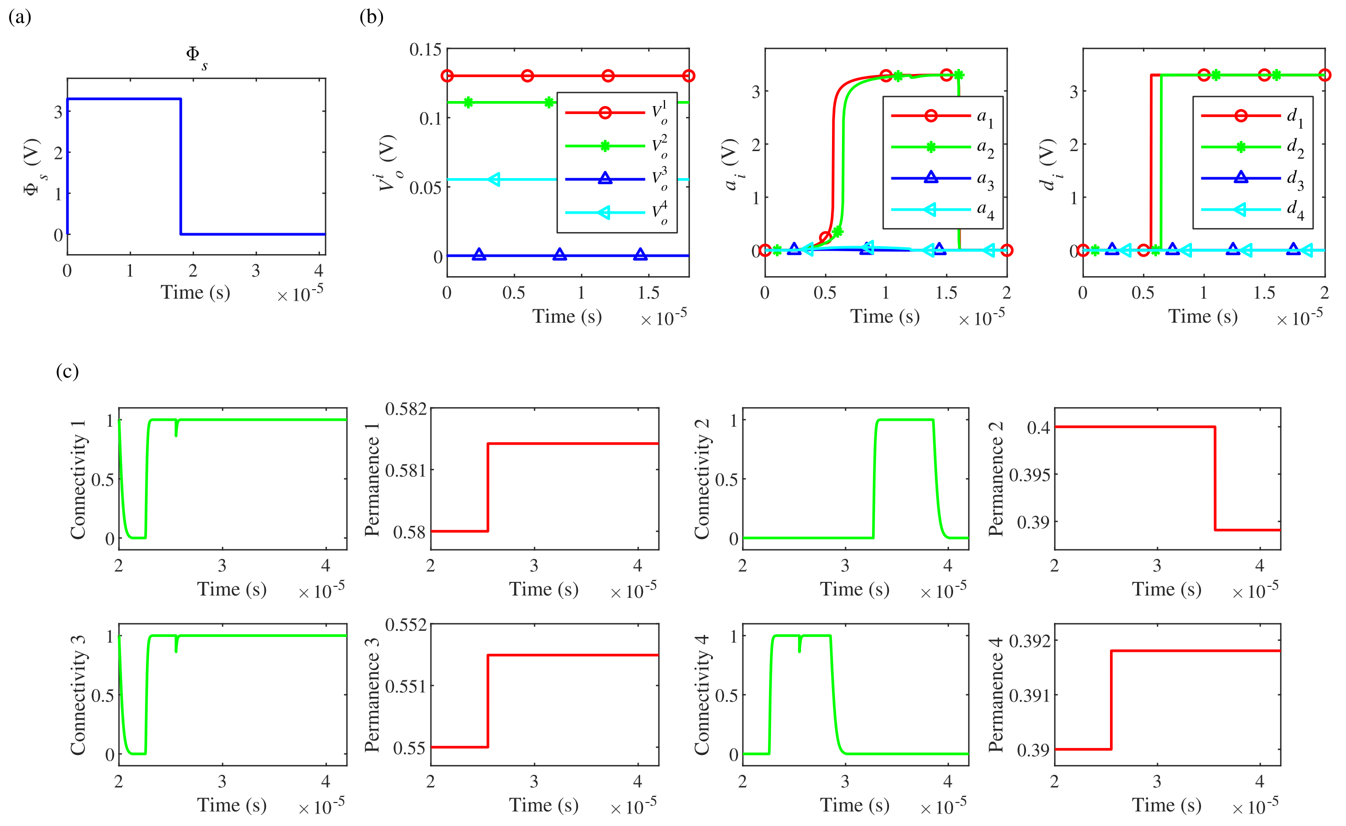


Fig. 13. Circuit simulation results of MSP. (a) Φ_s is the switch signal to switch inference and learning phases. In the inference phase, Φ_s is high level and in the learning phase, Φ_s is low level. (b) There are four mini-columns of which the overlap values are V_o^1, V_o^2, V_o^3, and V_o^4. a_i $(i = 1, 2, 3, 4)$ is the activation state of the ith mini-column, namely, one of the outputs of the kWTA unit, and d_i is the output of the ith TD unit. The first and second mini-columns are activated. (c) Synaptic connection states and synaptic permanence of the second mini-column of which the overlap value is V_o^2. Input bits of the second mini-column are "1," "0," "1," and "1," so Permanence 1, Permanence 3, and Permanence 4 are increased, and Permanence 2 is decreased. Because Permanence 1 and Permanence 3 are larger than the permanence threshold P_θ, Connectivity 1 and Connectivity 3 are finally set to 1. Permanence 2 and Permanence 4 both are smaller than P_θ, so Connectivity 2 and Connectivity 4 both are reset to 0.

CIFAR-10 contains 50 000 training samples and 10 000 testing samples. These input images are normalized to binary values {0,1} according to

$$g_i = \mathbf{I}_q\left(\frac{u_i}{255}\right) \tag{25}$$

where g_i is the binary value, u_i is the original pixel value of the image, and $\mathbf{I}_q$ is the indicator function

$$\mathbf{I}_q(x) = \begin{cases} 1, & x \geq q \\ 0, & x < q \end{cases} \tag{26}$$

where $q \in (0, 1)$ is a threshold value. The input voltage value $V_{x_i} = 0.2g_i(\mathrm{V})$ where V is volt.

After SDRs are obtained through feeding images to MSP in an unsupervised manner, they are fed to a supervised maximum-likelihood classifier to verify the capacity of MSP to extract features from images. The adopted classifier is a Softmax classifier. A one-layer Softmax classifier is

$$\boldsymbol{h} = W \cdot \boldsymbol{a} \tag{27}$$

$$p_j = \frac{\exp(\mathrm{h}_j)}{\sum_{j=1}^{K} \exp(\mathrm{h}_j)} \tag{28}$$

where $j \in [1, S]$ and S is the class number, W is the weight matrix of which the dimension is $S \times Z$, $\boldsymbol{a}$ is the SDR vector of which the dimension is Z, h_j is an element of $\boldsymbol{h}$, and p_j is the probability of h_j belonging to the jth class. In the two layer Softmax classifier

$$\boldsymbol{h} = W_2 \cdot \sigma(W_1 \cdot \boldsymbol{a}) \tag{29}$$

where σ is the activation function and here it is the rectified linear unit (ReLU) function. The Softmax classifier aims to substantiate the validity of SDRs, so it is implemented on software and is trained through the error backpropagation (BP) algorithm. In order to apply MSP on the CIFAR-10 dataset which contains RGB images, three MSPs are adopted. One MSP deals with one color channel, and then the obtained SDRs of three color channels are concatenated before being fed into the Softmax classifier. Images are all resized to 32×32 and binarized through (25). The image classification results on different datasets are shown in Table IV. A comparison with a CNN with one convolutional layer is also shown in Table IV. MSP is simulated based on the hardware architecture while CNN is implemented in precise software. CNN has better classification results but it requires more precise weight values which is more challenging to achieve through memristors. Classification results of different SP circuits on MNIST are summarized in Table V. "X" means that the adopted classifier is unknown. SVM and KNN are the support vector

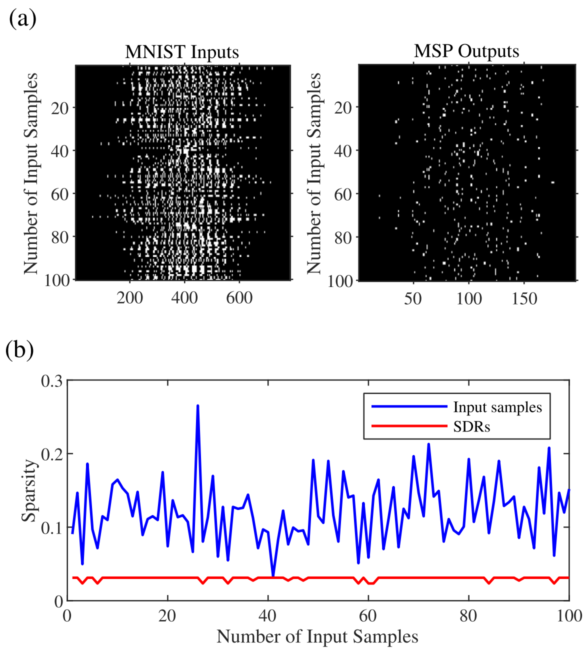


Fig. 14. Sparsity of MNIST sample vectors and MSP outputs. (a) Horizontal axes are the input dimension and the output dimension, respectively. The vertical axes are both the number of input samples. (b) Sparseness variation of input samples and SDRs. The sparseness of input samples is mainly among about 3%–26% and those of SDRs are all about 2%.

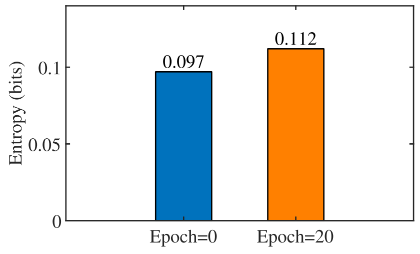


Fig. 15. Entropies of MSP before and after training. The entropy increases after training.

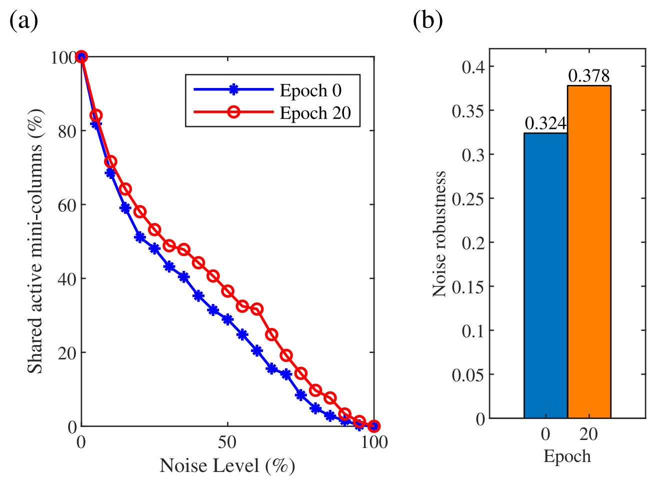


Fig. 16. (a) Change of MSP outputs with varying input noise levels from 0% to 100% with step 5%. (b) Noise robustness. It can be seen that after training, the robustness of MSP is improved.

machine and the k-nearest neighbor algorithm, respectively. The relationship between the classification accuracies and the activation densities of MSP is also explored and is shown in

TABLE IV
CLASSIFICATION RESULTS OF MSP AND CNN ON DIFFERENT DATASETS

Network	Dataset	Accuracy	
		One-layer Softmax	Two-layer Softmax
MSP	MNIST	90.57%	96.72%
	Fashion-MNIST	78.55%	82.69%
	CIFAR-10	37.32%	45.62%
CNN	MNIST	98.05%	98.18%
	Fashion-MNIST	87.04%	87.91%
	CIFAR-10	44.60%	53.16%

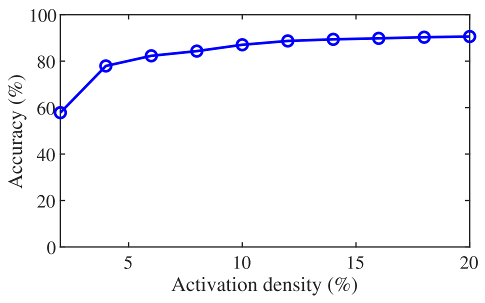


Fig. 17. Classification accuracies of MNIST along with various activation densities. The larger the activation density, the higher the classification accuracy.

Fig. 17. The activation densities vary from 2% to 20%, and it can be seen that with the increase of the activation density, the classification accuracy increases. A comparison of the fixed-radius local inhibition and the global inhibition is performed on the Fashion-MNIST dataset. The classification accuracy of the former is 82.69% and that of the latter is 82.07%. To evaluate the performance degradation of the hardware defined constraints, a software-based SP is performed, and the result on Fashion-MNIST is 83.93%. So the hardware constraints have a degradation of about 1.2%.

E. Power Consumption and Area

Table VI shows the average power consumption of each unit and the total power consumption of MSP. The circuit is implemented using a 180-nm CMOS technology. In the learning phase, there are extra voltages to determine the synaptic connection states, so the mini-column consumes more power than that in the inference phase. If 20% of total mini-columns are activated, the power consumed by each mini-column is averaged 0.167 mW in the learning phase and in the inference phase, it is 0.017 mW. There are also 20% of TD units storing the high-level voltages, so the power consumption of each TD unit is averaged 1.45 μW. The kWTA unit each has a power consumption of 0.460 mW. The total power consumption of MSP is 6.56 mW. The area of each mini-column is estimated to be 18.051 μm^2, and the total area of MSP is about 6012.5 μm^2.

TABLE V
CLASSIFICATION RESULTS OF DIFFERENT SP CIRCUITS ON MNIST

Work	Mini-column number	Classifier	Accuracy
Non-volatile HTM [20]	784	SVM	91.98%
Crossbar SP [22]	1024	X	≈90.5%
Neuromorphic HTM [26]	100	SVM / KNN	91.16% / 87.47%
Neuromemrisitive HTM [27]	484	One-layer Softmax	90.33±0.17%
MSP in this work	256	One-layer Softmax	90.57%

TABLE VI
AVERAGE POWER CONSUMPTION OF EACH UNIT AND THE TOTAL POWER CONSUMPTION OF MSP

Unit	Average power consumption
Mini-column (Train)	0.166mW
Mini-column (Test)	0.017mW
TD unit	1.45μW
kWTA unit	0.460mW
Total	6.56mW

F. Robustness and Scalability Analysis

1) Variations of Memristor Resistance: The adjustment of the memristor resistance exists cycle-to-cycle (C2C) and device-to-device (D2D) variations, and the resistance state may lose due to the device characteristic. The adjustment of the synaptic permanence in the SP algorithm is rather the direction than an exact value. Therefore, memristors are appropriate to implement the SP algorithm, and the performance will not be strongly affected by the resistance variations. To substantiate this, considering the variations of memristors, Gaussian noises with mean 0 and standard deviations from 5% to 30% of the adjustment value are considered as the variations in adjusting M_2. Classification accuracies of MNIST under different noise levels are depicted in Fig. 18. It can be seen that the noises have little influence on the validity of SDRs.

Some memristors may be stuck due to aging or other possible causes. Because only R_{on} and R_{off} states of M_1 are used, the circumstance that M_1 is stuck at R_{on} or R_{off} is considered. The result of stuck M_2 is that M_1 in the same synapse is always R_{on} or R_{off} of which the result is the same as that M_1 is stuck. Supposing that 2%–12% of synapses are stuck, the classification accuracies are shown in Fig. 19. The accuracy degrades a little with the increase in the percentage of stuck synapses.

2) Scalability of MSP: The memristors in R_{off} states cannot block the synapses completely, especially in the large-scale network. But MSP only needs to select the k largest overlap values in the inhibition operation, so the incompletely blocking does not affect the inhibition operation. In HTM synapses, the synaptic permanence does not need a precise value, so the resistance variations of memristors do not affect the scaling of MSP much.

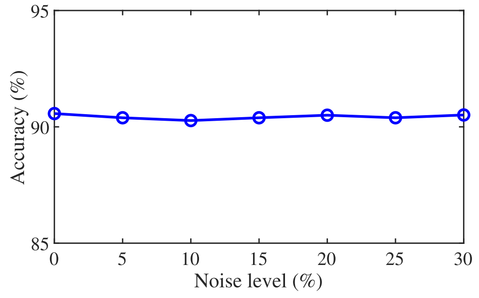


Fig. 18. Classification accuracies under various Gaussian noises in adjusting M_2. The resistance variations have little influence on classification accuracy.

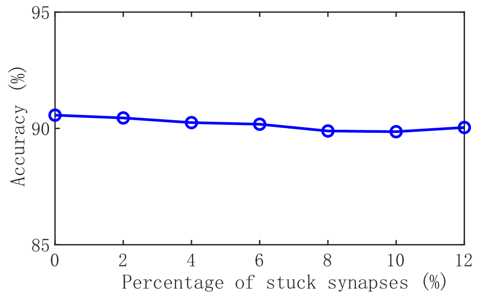


Fig. 19. Classification accuracies under various percentages of stuck synapses. The accuracy degrades a little with the increase in the percentage of stuck synapses.

G. Comparison With Existing SP Circuits

A comparison of functions between existing SP circuits and MSP is listed in Table VII. "Yes" means that the corresponding work satisfies the indicator and "No" means not. "-" means that the indicator is not applicable or the relative information is not reported. The area comparison is conducted among the designs with on-device learning. It can be seen that MSP has advantages in one or more aspects.

V. CONCLUSION

This article presented a memristor-based analog circuit for SP of HTM. The designed MSP simultaneously realizes the synaptic permanence and the synaptic connection state of the SP/HTM synapse within a single synapse. MSP performs the fixed-radius local inhibition instead of the global inhibition, and kWTA circuits are used to determine activated mini-columns. Moreover, MSP achieves on-device learning, and the synaptic permanence and the synaptic connection state are adjusted in parallel. Through the simulation, it is found that MSP is about 341× faster than a software-based

TABLE VII
COMPARISON OF EXISTING SP CIRCUITS AND MSP

Function \ Work	[18]	[19]	[20]	[21]	[22]	[23]	[24]	[26]	[27]	MSP
Use memristor-like elements	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes
Model both synaptic connection state and synaptic permanence	No	No	Yes	Yes	No	No	No	Yes	No	Yes
Inhibition type	-	-	Global	Fix-radius local	Fix-radius local	Fix-radius local	Fix-radius local	Global	-	Fix-radius local
*k*WTA operation	No	No	Yes	No	No	No	No	Yes	No	Yes
Boosting factor	No	-	Yes	No	Yes	No	No	No	Yes	Yes
Updated synapses in parallel	-	-	No	No	-	-	-	No	Yes	Yes
Satisfying metrics	-	-	-	-	Yes	-	-	-	Yes	Yes
On-device learning	No	No	Yes	No	No	No	No	Yes	Yes	Yes
Total area	-	-	30.538mm^2	-	-	-	-	-	0.513mm^2	$6012.5\mu\text{m}^2$
Power consumption	4.79μW per synapse	-	64.394mW total SP	0.160mW per mini-column	-	0.092mW per mini-column	0.135mW per receptor block	1.39mW per cell	18.47mW total SP	

* 0.166mW and 0.017mW per mini-column in the learning phase and the inference phase, respectively, and 6.56mW total MSP.

SP on MATLAB. Some evaluation metrics, including sparseness, entropy, and robustness, are calculated to substantiate the effectiveness. Image classification tasks are carried out to substantiate that SDRs generated by MSP are effective sparse representations of input patterns. The impact of activation density on the classification accuracy is explored. The impacts of resistance variations and stuck memristors are also explored, and the results show that these imperfections of memristors have little influence on the performance of MSP.

REFERENCES

[1] J. Hawkins and D. George, "Hierarchical temporal memory: Concepts, theory and terminology," Numenta, Redwood City, CA, USA Rep., 2006. [Online]. Available: http://www.mlanctot.info/files/papers/Numenta_HTM_Concepts.pdf

[2] W. J. C. Melis and M. Kameyama, "A study of the different uses of colour channels for traffic sign recognition on hierarchical temporal memory," in *Proc. 4th Int. Conf. Innovat. Comput. Inf. Control*, Kaohsiung, Taiwan, Dec. 2009, pp. 111–114.

[3] D. Rozado, F. B. Rodriguez, and P. Varona, "Extending the bioinspired hierarchical temporal memory paradigm for sign language recognition," *Neurocomputing*, vol. 79, pp. 75–86, Mar. 2012.

[4] J. van Doremalen and L. Boves, "Spoken digit recognition using a hierarchical temporal memory," in *Proc. 9th Annu. Conf. Int. Speech Commun. Assoc.*, Brisbane, QLD, Australia, Sep. 2008, pp. 2566–2569.

[5] W. J. C. Melis, S. Chizuwa, and M. Kameyama, "Evaluation of the hierarchical temporal memory as soft computing platform and its VLSI architecture," in *Proc. 39th Int. Symp. Multiple Valued Logic*, Naha, Japan, May 2009, pp. 233–238.

[6] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms–The numenta anomaly benchmark," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl.*, Miami, FL, USA, Dec. 2015, pp. 38–44.

[7] S. Ahmad and J. Hawkins, "Properties of sparse distributed representations and their application to hierarchical temporal memory," Mar. 2015. [Online]. Available: http://arxiv.org/abs/1503.07469.

[8] M. Pietroń, M. Wielgosz, and K. Wiatr, "Formal analysis of HTM spatial pooler performance under predefined operation conditions," in *Proc. Int. Joint Conf. Rough Sets*, Santiago, Chile, Oct. 2016, pp. 396–405.

[9] Y. Cui, S. Ahmad, and J. Hawkins, "Continuous online sequence learning with an unsupervised neural network model," *Neural Comput.*, vol. 28, no. 11, pp. 2474–2504, Nov. 2016.

[10] J. Mnatzaganian, E. Fokoué, and D. Kudithipudi, "A mathematical formalization of hierarchical temporal memory's spatial pooler," *Front. Robot. AI*, vol. 3, p. 81, Jan. 2017.

[11] Y. Cui, S. Ahmad, and J. Hawkins, "The HTM spatial pooler—A neocortical algorithm for online sparse distributed coding," *Front. Comput. Neurosci.*, vol. 11, p. 111, Nov. 2017.

[12] D. Maltoni. (2011). *Pattern Recognition by Hierarchical Temporal Memory*. [Online]. Available: https://ssrn.com/abstract=3076121

[13] J. Thornton and A. Srbic, "Spatial pooling for greyscale images," *Int. J. Mach. Learn. Cybern.*, vol. 4, no. 3, pp. 207–216, Jun. 2013.

[14] M. Deshpande, "FPGA implementation and acceleration of building blocks for biologically inspired computational models," M.S. thesis, Dept. Elect. Comput. Sci., Portland State Univ., Portland, OR, USA, 2011.

[15] P. Vyas, "Verilog implementation of a node of hierarchical temporal memory," *Asian J. Comput. Sci. Inf. Technol.*, vol. 7, pp. 103–108, Oct. 2013.

[16] A. M. Zyarah and D. Kudithipudi, "Reconfigurable hardware architecture of the spatial pooler for hierarchical temporal memory," in *Proc. 28th IEEE Int. Syst. Conf.*, Beijing, China, 2015, pp. 143–153.

[17] V. Puente and J. Á. Gregorio, "CLASSIC: A cortex-inspired hardware accelerator," *J. Parallel Distrib. Comput.*, vol. 134, pp. 140–152, Dec. 2019.

[18] T. Ibrayev, A. P. James, C. Merkel, and D. Kudithipudi, "A design of HTM spatial pooler for face recognition using memristor-CMOS hybrid circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, Montreal, QC, Canada, Jul. 2016, pp. 1254–1257.

[19] D. Fan, M. Sharad, A. Sengupta, and K. Roy, "Hierarchical temporal memory based on spin-neurons and resistive memory for energy-efficient brain-inspired computing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 9, pp. 1907–1919, Sep. 2016.

[20] L. Streat, D. Kudithipudi, and K. Gomez, "Non-volatile hierarchical temporal memory: Hardware for spatial pooling," 2016. [Online]. Available: arXiv:1611.02792.

[21] A. P. James, I. Fedorova, T. Ibrayev, and D. Kudithipudi, "HTM spatial pooler with memristor crossbar circuits for sparse biometric recognition," *IEEE Trans. Biomed. Circuits Syst.*, vol. 11, no. 3, pp. 640–651, Jun. 2017.

[22] S. N. Truong, K. Van Pham, and K.-S. Min, "Spatial-pooling memristor crossbar converting sensory information to sparse distributed representation of cortical neurons," *IEEE Trans. Nanotechnol.*, vol. 17, no. 3, pp. 482–491, May 2018.

[23] O. Krestinskaya, T. Ibrayev, and A. P. James, "Hierarchical temporal memory features with memristor logic circuits for pattern recognition," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 6, pp. 1143–1156, Jun. 2018.

[24] O. Krestinskaya and A. P. James, "Feature extraction without learning in an analog spatial pooler memristive-CMOS circuit design of hierarchical temporal memory," *Analog Integr. Circuits Signal Process.*, vol. 95, no. 3, pp. 457–465, Jun. 2018.

[25] O. Krestinskaya, I. Dolzhikova, and A. P. James, "Hierarchical temporal memory using memristor networks: A survey," 2018. [Online]. Available: http://arxiv.org/abs/1805.02921.

[26] A. M. Zyarah and D. Kudithipudi, "Neuromorphic architecture for the hierarchical temporal memory," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 1, pp. 4–14, Feb. 2019.

[27] A. M. Zyarah and D. Kudithipudi, "Neuromemrisitive architecture of HTM with on-device learning and neurogenesis," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 3, p. 24, Jun. 2019.

[28] 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, May 2008.

[29] L. Chua, "Resistance switching memories are memristors," *Appl. Phys. A*, vol. 102, no. 4, pp. 765–783, Mar. 2011.

[30] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nat. Nanotechnol.*, vol. 8, no. 1, pp. 13–24, Jan. 2013.

[31] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, Mar. 2010.

[32] 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.

[33] 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," *IEEE Trans. Syst. Man, Cybern., Syst.*, vol. 45, no. 2, pp. 235–249, Feb. 2015.

[34] P. Liu, Z. Zeng, and J. Wang, "Multistability of recurrent neural networks with nonmonotonic activation functions and mixed time delays," *IEEE Trans. Syst. Man, Cybern., Syst.*, vol. 46, no. 4, pp. 512–523, Apr. 2016.

[35] S. Duan, X. Hu, Z. Dong, L. Wang, and 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.

[36] T. Li, S. Duan, J. Liu, L. Wang, and T. Huang, "A spintronic memristor-based neural network with radial basis function for robotic manipulator control implementation," *IEEE Trans. Syst. Man, Cybern., Syst.*, vol. 46, no. 4, pp. 582–588, Apr. 2016.

[37] X. Liu, Z. Zeng, and S. Wen, "Implementation of memristive neural network with full-function Pavlov associative memory," *IEEE Trans. Circuits Syst. I, Regul. Papers*, vol. 63, no. 9, pp. 1454–1463, Sep. 2016.

[38] 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.

[39] 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 Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 9, pp. 1787–1796, Sep. 2019.

[40] L. Yang, Z. Zeng, and S. Wen, "A full-function Pavlov associative memory implementation with memristance changing circuit," *Neurocomputing*, vol. 272, pp. 513–519, Jan. 2018.

[41] X. Shi, Z. Zeng, L. Yang, and Y. Huang, "Memristor-based circuit design for neuron with homeostatic plasticity," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 5, pp. 359–370, Oct. 2018.

[42] L. Yang, Z. Zeng, Y. Huang, and S. Wen, "Memristor-based circuit implementations of recognition network and recall network with forgetting stages," *IEEE Trans. Cogn. Develop. Syst.*, vol. 10, no. 4, pp. 1133–1142, Dec. 2018.

[43] S. Wen *et al.*, "Memristive LSTM network for sentiment analysis," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Apr. 16, 2019, doi: 10.1109/TSMC.2019.2906098.

[44] Y. Zhou *et al.*, "Associative memory for image recovery with a high-performance memristor array," *Adv. Funct. Mater.*, vol. 29, Jul. 2019, Art. no. 1900155.

[45] Y. Zhou, H. Zhang, and Z. Zeng, "Quasi-synchronization of delayed memristive neural networks via a hybrid impulsive control," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Apr. 26, 2019, doi: 10.1109/TSMC.2019.2911366.

[46] J. Hawkins, S. Ahmad, and D. Dubinsky, "Hierarchical temporal memory including HTM cortical learning algorithms," Numenta, Redwood City, CA, USA, Rep. 2, Sep. 2010. [Online]. Available: http://www.numenta.com/htm-overview/education/HTM_CorticalLearningAlgorithms.pdf

[47] D. O. Hebb, *The Organization of Behavior*. New York, NY, USA: Wiley, 1949.

[48] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "VTEAM: A general model for voltage-controlled memristors," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 8, pp. 786–790, Aug. 2015.

[49] Y. Li *et al.*, "Activity-dependent synaptic plasticity of a chalcogenide electronic synapse for neuromorphic systems," *Sci. Rep.*, vol. 4, p. 4906, May 2014.

[50] L. Jiang *et al.*, "Conductance quantization in an AgInSbTe-based memristor at nanosecond scale," *Appl. Phys. Lett.*, vol. 109, no. 15, Oct. 2016, Art. no. 153506.

[51] M. Hu, Y. Chen, J. J. Yang, Y. Wang, and 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.

[52] E. Majani, R. Erlanson, and Y. S. Abu-Mostafa, "On the k-winners-take-all network," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA, USA: Morgan Kaufmann, 1989, pp. 634–642.

[53] B. Sekerkiran and U. Cilingiroglu, "A CMOS k-winners-take-all circuit with O(n) complexity," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 1, pp. 1–5, Jan. 1999.

[54] J. Wang, "Analysis and design of a k-winners-take-all model with a single state variable and the heaviside step activation function," *IEEE Trans. Neural Netw.*, vol. 21, no. 9, pp. 1496–1506, Sep. 2010.

[55] X. Liu and J. Wang, "An analog circuit design for k-winners-take-all operations," in *Proc. 25th Int. Conf. Neural Inf. Process.*, Dec. 2018, pp. 666–675.

[56] P. V. Tymoshchuk and D. C. Wunsch, "Design of a k-winners-take-all model with a binary spike train," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 3131–3140, Aug. 2019.

[57] C. Mead, *Analog VLSI and Neural Systems*, vol. 90. Reading, MA, USA: Addison-Wesley, 1989.

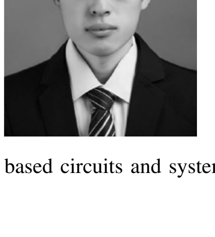
[58] L. Yann, B. Léon, B. Yoshua, and H. Patrick, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[59] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017. [Online]. Available: arXiv:1708.07747.

[60] A. Krizhevsky, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Rep. TR-2009, 2009.

[61] O. Krestinskaya, A. P. James, and L. O. Chua, "Neuromemristive circuits for edge computing: A Review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 1, pp. 4–23, Jan. 2020.

[62] R. Iqbal, T. Maniak, F. Doctor, and C. Karyotis, "Fault detection and isolation in industrial processes using deep learning approaches," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 3077–3084, May 2019.



Xiaoyang Liu received the B.S. degree in automation from the College of Electrical and Information Engineering, Hunan University, Changsha, China, in 2014. He is currently pursuing the Ph.D. degree in control science and engineering with the School of Artificial Intelligence and Automation and Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, Huazhong University of Science and Technology, Wuhan, China.

His current research interests include memristor-based circuits and systems, artificial neural networks, and deep learning.

Yi Huang (Graduate Student Member, IEEE) received the B.S. degree in automation and the M.S. degree in systems engineering from the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China, in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Massachusetts at Amherst, Amherst, MA, USA.

His main interest is memristor-based artificial neural networks.

Zhigang Zeng (Fellow, IEEE) received the Ph.D. degree in systems analysis and integration from the Huazhong University of Science and Technology, Wuhan, China, in 2003.

He is currently a Professor with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology and also with the Key Laboratory of Image Processing and Intelligent Control of the Education Ministry of China. He has published more than 100 international journal papers. His current research interests include theory of functional differential equations and differential equations with discontinuous right-hand sides, and their applications to dynamics of neural networks, memristive systems, and control systems.

Prof. Zeng was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS from 2010 to 2011, and has been an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS since 2014 and the IEEE TRANSACTIONS ON FUZZY SYSTEMS since 2016. He has been a member of the Editorial Board of *Neural Networks* since 2012, *Cognitive Computation* since 2010, and *Applied Soft Computing* since 2013.

Donald C. Wunsch, II (Fellow, IEEE) received the B.S. degree in applied mathematics from the University of New Mexico, Albuquerque, NM, USA, in 1984, the M.S. degree in applied mathematics and the Ph.D. degree in electrical engineering from the University of Washington, Seattle, WA, USA, in 1987 and 1991, respectively, the M.B.A. degree in executive from Washington University in St. Louis, St. Louis, MO, USA, and the Jesuit Core Honors Program from Seattle University, Seattle, in 2006.

He is the Mary K. Finley Missouri Distinguished Professor with the Missouri University of Science and Technology (Missouri S&T), Rolla, MO, USA. He is the Director of the Applied Computational Intelligence Laboratory, a multidisciplinary research group. He was with Texas Tech University, Lubbock, TX, USA; Boeing, Seattle, WA, USA; Rockwell International, Kirtland AFB, Albuquerque, NM, USA; and the International Laser Systems, Kirtland AFB. He has produced 20 Ph.D. recipients in Computer Engineering, Electrical Engineering, Systems Engineering, and Computer Science. He has attracted over \$10 million in sponsored research and has over 450 publications including nine books. He has over 15 000 citations. His current research interests include clustering/unsupervised learning, biclustering, adaptive resonance and reinforcement learning architectures, hardware and applications, neurofuzzy regression, traveling salesman problem heuristics, games, robotic swarms, and bioinformatics.

Dr. Wunsch was a recipient of the NSF CAREER Award and the 2015 INNS Gabor Award. He served as the IJCNN General Chair, and on several boards, including the St. Patrick's School Board, IEEE Neural Networks Council, International Neural Networks Society, and the University of Missouri Bioinformatics Consortium, Chaired the Missouri S&T Information Technology and Computing Committee as well as the Student Design and Experiential Learning Center Board. He was an INNS President and an INNS Fellow.