IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 10, 2019, accepted May 22, 2019, date of publication May 27, 2019, date of current version June 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2919163

A Neuromorphic-Hardware Oriented Bio-Plausible Online-Learning Spiking Neural Network Model

G. C. QIAO[1], S. G. HU[1], J. J. WANG[1], C. M. ZHANG[1], T. P. CHEN[2], N. NING[1], Q. YU[1], AND Y. LIU[1]

[1]State Key Laboratory of Electronic Thin Films and Integrated Devices, University of Electronic Science and Technology of China, Chengdu 610054, China
[2]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

Corresponding author: S. G. Hu (sghu@uestc.edu.cn)

This work was supported in part by the NSFC under Project 61774028 and Project 61771097, in part by the Fundamental Research Funds for the Central Universities under Project ZYGX2016Z007, and in part by the Opening Project of Science and Technology on Reliability Physics and Application Technology of the Electronic Component Laboratory under Project ZHD201602.

ABSTRACT Neuromorphic hardware inspired by the brain has attracted much attention for its advanced information processing concept. However, implementing online learning in the neuromorphic chip is still challenging. In this paper, we present a bio-plausible online-learning spiking neural network (SNN) model for hardware implementation. The SNN consists of an input layer, an excitatory layer, and an inhibitory layer. To save resource cost and accelerate information processing speed during hardware implementation, online learning based on the spiking neural model is realized by trace-based spiking-timing-dependent plasticity (STDP). Neuron and synapse activities are digitalized, and decay behaviors of neuron and synapse parameters are realized by the bit-shift operation. After learning training set from the Modified National Institute of Standards and Technology (MNIST), the spiking neural model successfully recognizes the digits from the MNIST test set, showing the feasibility and capability of the model. The recognition accuracy increases significantly from 90.0% to 94.5% with the number of the excitatory/inhibitory neurons rising from 400 to 3,500, which provides a guide to make a trade-off between the recognition accuracy and the resource cost during hardware implementation. Encouragingly, compared to its corresponding floating-point model, the proposed model reduces the hardware resources and power consumption by 40.7% and 36.3%, respectively (under 55-nm CMOS process).

INDEX TERMS Neuromorphic hardware, online learning, SNN, STDP.

I. INTRODUCTION

A neuromorphic computing platform, inspired by the advanced information processing scheme of the brain [1], is more efficient and bio-plausible than the traditional Von Neumann computing platform when dealing with brain-like computation tasks (such as pattern recognition) [2]. Therefore, it has attracted significant attention in recent years. However, an effective method to implement online learning in a neuromorphic computing platform is still missing [3]. For example, SpiNNaker, one of the representative neuromorphic platforms, is based on multi-core ARM digital chips and may still suffer from the Von Neumann bottleneck [4], [5]. TrueNorth, another example of the most advanced platforms, does not support any synaptic plasticity mechanisms and thus does not have online learning ability [6]–[8]. In these cases, neurons and synaptic parameters can only be off-line mapped into the chip to achieve neural network functions.

Among the various spiking neural network (SNN) training methods [9], training SNN based on the traditional back-propagation (BP) algorithm [10]–[14] or training artificial neural network (ANN) into the SNN by mapping [15]–[18] are inefficient and biologically implausible. Recently, bio-plausible learning rules, such as spiking-timing-dependent plasticity (STDP), together with BP algorithm have been used to achieve supervised learning in SNN [19]–[21]. For example, Tavanaei proposed a BP-STDP algorithm to approximate backpropagation using STDP and achieved a recognition accuracy of 97.2% under Modified National Institute of Standards and Technology (MNIST) test set [21]. However, these models cannot realize unsupervised online learning, thus cannot well

The associate editor coordinating the review of this manuscript and approving it for publication was Bora Onat.

emulate the brain's autonomous learning function (since the brain is unsupervised and event-based). More recently, unsupervised learning based on STDP and other biologically plausible learning rules were reported by several groups [22], [23]. Diehl proposed an SNN model that can achieve unsupervised learning based on bio-plausible STDP learning rule [23] and achieved a recognition accuracy of 95.0% under MNIST test set. However, this model is implemented on the Brian2 spiking neural network work simulator [24] and involves a large number of floating-point calculations, which is extremely difficult to be implemented in hardware.

This paper presents a hardware-oriented online learning neuromorphic computing model. Biologically plausible STDP rule is used in the model to achieve unsupervised learning. Moreover, neuron and synapse activities are digitalized, and decay behaviors of the neuron and synapse parameters are realized by bit-shift, which can significantly reduce the hardware cost. In the model with the size of 784-3500-3500, a recognition accuracy of 94.5% is achieved under the MNIST test set. Moreover, under the 55 nm CMOS process, the hardware resources and power consumption are reduced by 40.7% and 36.3%, respectively (model size: 784-784-784). The model can be used as a reference model to design a neuromorphic computing platform with an online learning engine, and can also be used to verify the performance of such a platform in the future. Efficiently embedding online learning capabilities enables neuromorphic platforms to adapt to and learn new features from changing environments, which has various practical application scenarios, such as autonomous smart sensing in the Internet-of-Things (IoT) [25], autonomous embedded systems [26] and robots [27], brain-machine interfaces [28], and experimental neuroscience platform [29], etc.

II. MODEL DESCRIPTION

A. NETWORK ARCHITECTURE

As shown in Fig. 1, the SNN architecture consists of three layers, and the black and blue arrows denote excitatory and inhibitory synapses, respectively. The first layer is an input layer used to receive spikes from the outside. The second layer is an excitatory layer, and the synapses between the input neurons and excitatory neurons are excitatory synapses, whose weights are placed in the weight matrix M_{xe}. The delay time is randomly assigned (e.g., 0∼10 ms) during the initialization process and will not be changed later. Moreover, the values of the delay time should be integer multiples of a time step (e.g., 0.5 ms) for efficient hardware computation. The third layer is an inhibitory layer, and there is a one-to-one correspondence between the excitatory neurons and inhibitory neurons, i.e., the number of inhibitory neurons is equal to the number of excitatory neurons. A competitive learning mechanism, which is similar to the one used in the Self Organizing Maps (SOM) model [30] and winner-take-all model [31], [32], is introduced among the excitatory neurons. When an excitatory neuron fires a spike, it will

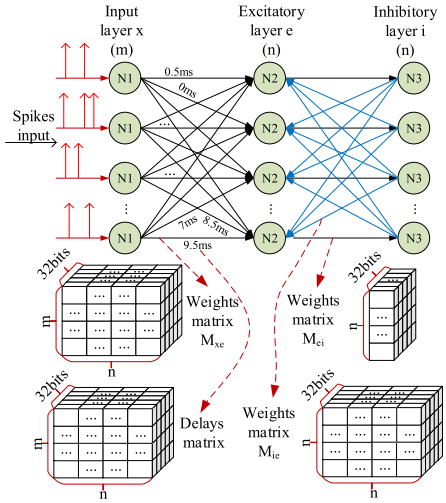

FIGURE 1. Spiking neural network architecture.

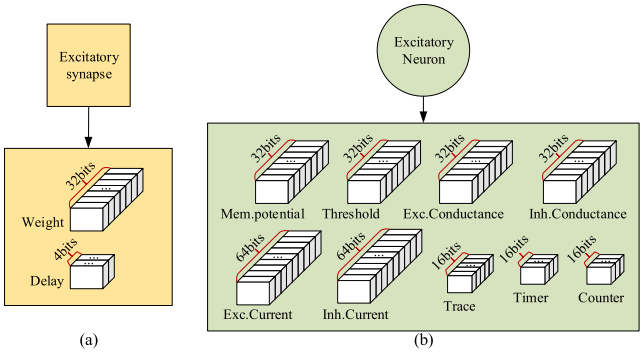

FIGURE 2. (a) Size of excitatory synapse parameters. (b) Size of excitatory neuron parameters.

cause the excitation of its corresponding inhibitory neuron through excitatory synapse (with a large fixed weight). The inhibitory neuron, in turn, inhibits all other excitatory neurons through inhibitory synapses. The weights of excitatory synapses and inhibitory synapses between the excitatory neurons and inhibitory neurons are placed in weight matrices M_{ei} and M_{ie}, respectively. Fig. 1 also shows the size of the weight matrices and delay matrix. The activities of the neuron and synapse are digitalized and are represented by fixed-point numbers, and the size of the parameters are shown in Fig. 2.

B. NEURON MODEL

Integrate-and-Fire (IF) [33] neuron, as well as excitatory synapse and inhibitory synapse, are used as the basic building block in the three-layer SNN (i.e., including an input layer, an excitatory layer, and an inhibitory layer). Moreover, the excitatory synapse between the input neurons and excitatory neurons has a delay unit, and its weight can be autonomously adjusted according to the STDP rule [34], whereas the excitatory synapse between the excitatory neurons and inhibitory neurons and the inhibitory synapse have no delay unit, and their weights M_{ei} and M_{ie} cannot be changed. The conductance of an excitatory neuron is updated

at each time step following the rule:

$$g_j^{\text{exc}}(t) = \sum_{i=1}^{\text{m}} x_{ij}(t)\, w_{ij} \tag{1}$$

$$g_j^{\text{inh}}(t) = \sum_{l=1}^{\text{n}} x_{lj}(t)\, w_{lj} \tag{2}$$

where i, j, l are the indexes of the input neuron, the excitatory neuron, and the inhibitory neuron, respectively; g_j^{exc} and g_j^{inh} are the excitatory conductance and the inhibitory conductance of the excitatory neuron j, respectively; x_{ij} is the input from the input neuron i to the excitatory neuron j, which can only be either 1 or 0 (spike or none spike); x_{lj} is the input from the inhibitory neuron l to the excitatory neuron j; w_{ij} is the weight from the input neuron i to the excitatory neuron j; w_{lj} is the weight from the inhibitory neuron l to the excitatory neuron j; m and n are the numbers of input neurons and inhibitory neurons, respectively.

The excitatory and the inhibitory currents are then updated according to:

$$I_j^{\text{exc}}(t) = g_j^{\text{exc}}(t) \times V_j(t-1) \tag{3}$$

$$I_j^{\text{inh}}(t) = g_j^{\text{inh}}(t) \times V_j(t-1) \tag{4}$$

where I_j^{exc} and I_j^{inh} are the excitatory current and inhibitory current of neuron j, respectively; V_j is the membrane potential of neuron j.

The membrane potential is updated according to the IF neuron model:

$$V_j(t) = V_j(t-1) + I_j^{\text{exc}}(t) + I_j^{\text{inh}}(t) \tag{5}$$

$$V_j(t) = \begin{cases} V_j(t), & \text{if } V_j(t) < \text{V}_j^{\text{th}} \\ \text{V}_j^{\text{reset}}, & \text{if } V_j(t) \geq \text{V}_j^{\text{th}} \end{cases} \tag{6}$$

where V_j^{th} and $\text{V}_j^{\text{reset}}$ are the threshold and reset membrane potential of neuron j, respectively. The membrane potential is updated by adding the excitatory current and inhibitory current simultaneously. If the updated membrane potential is higher than the threshold, the neuron fires a spike, and then its membrane potential is reset to $\text{V}_j^{\text{reset}}$; otherwise, the membrane potential remains unchanged. When a neuron fires a spike, its threshold will be increased by a specific value (e.g., 5×10^5); otherwise, it will exponentially decrease with time (it will be discussed later). After firing a spike, the neuron entries the refractory period, during which the parameters of the neuron will not be changed, and no spike will be fired. Moreover, a timer and a counter are used to record the duration of the refractory period and the number of spikes, respectively.

C. SYNAPSE MODEL

The classic STDP is typically defined as [35]:

$$\Delta w = \sum_{t_{\text{pre}}} \sum_{t_{\text{post}}} \text{F}\left(\text{t}_{\text{post}} - \text{t}_{\text{pre}}\right) \tag{7}$$

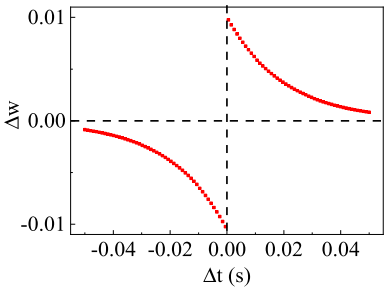

FIGURE 3. Weight modification with the trace-based STDP.

where t_{pre} and t_{post} are the pre- and post-synaptic spike times, respectively; w is the weight of the synapse; F is a function defined as:

$$\text{F}(\Delta t) = \begin{cases} \text{A}_{\text{pre}} e^{-\Delta t/\tau_{\text{pre}}}, & \text{if } \Delta t > 0 \\ \text{A}_{\text{post}} e^{\Delta t/\tau_{\text{post}}}, & \text{if} \Delta t < 0 \end{cases} \tag{8}$$

where $\Delta t = \text{t}_{\text{post}} - \text{t}_{\text{pre}}$; τ_{pre} and τ_{post} are time constants of the pre- and post-synaptic neurons; A_{pre} and A_{post} are amplitude constants.

It is inefficient to use the classic STDP to update weights directly because all pairs of spikes have to be summed over, which is also biologically unrealistic because neurons cannot remember all previous spikes times [24]. By contrast, the trace-based STDP, which has been proved equivalent to the classic STDP [34], is more efficient for hardware implementation. The pre- and post-synaptic traces are defined as pre and post, respectively, which are governed by:

$$\frac{d\text{pre}}{dt} = -\frac{\text{pre}}{\tau_{\text{pre}}} \tag{9}$$

$$\frac{d\text{post}}{dt} = -\frac{\text{post}}{\tau_{\text{post}}} \tag{10}$$

When a pre-synaptic spike occurs, the pre-synaptic trace and weight are modified according to:

$$\text{pre}(t) = \text{pre}(t-1) + \text{A}_{\text{pre}} \tag{11}$$

$$w(t) = w(t-1) - \text{post}(t-1) \tag{12}$$

When a post-synaptic spike occurs, the post-synaptic trace and weight are modified according to:

$$\text{post}(t) = \text{post}(t-1) + \text{A}_{\text{post}} \tag{13}$$

$$w(t) = w(t-1) + \text{pre}(t-1) \tag{14}$$

For example, if we take $\tau_{\text{pre}} = \tau_{\text{post}} = 20$ ms and $\text{A}_{\text{pre}} = \text{A}_{\text{post}} = 0.01$, weight modification with trace-based STDP (Fig. 3) gets the same effect with the classic STDP.

To further optimize the hardware implementation with the bit-shift operation (see below), pre is reset to an initial value (e.g., 10^4) when the pre-synaptic neuron fires a spike, and a learning rate is introduced in the weight modification:

$$w_{ij}(t) = w_{ij}(t-1) - \lambda_{\text{pre}} \times \text{post}_j(t-1) \tag{15}$$

where $w_{ij}(t-1)$ and $w_{ij}(t)$ are the weights from input neuron i to excitatory neuron j at the previous time step and current time step, respectively; λ_{pre} is the pre-synaptic learning

TABLE 1. The meaning and unit of symbols.

Symbol	Meaning	Unit
x	input	*
w	weight	mS
g	conductance	mS
I	current	mA
V	membrane potential	mV
pre	pre-synaptic trace	*
post	post-synaptic trace	*
t_{pre}	pre-synaptic spike time	*
t_{post}	post-synaptic spike time	*
τ_{pre}	pre-synaptic time constant	ms
τ_{post}	post-synaptic time constant	ms
A_{pre}	pre-synaptic amplitude constant	*
A_{post}	post-synaptic amplitude constant	*
λ_{pre}	pre-synaptic learning rate	*
λ_{post}	post-synaptic learning rate	*

Algorithm 1 Pseudo Code of the Weight Normalization

```
#Initialization of input constants
constant1 ← input1
constant2 ← input2
constant3 ← input3
#Initialization of weight w (m×n)
w ← rand (m, n) ∗ constant1
while i < training number do
  #Sum by column to w
  w_sum ← sum (w, 1)
    # Calculate normalization factor
    factor ← round (constant2./w_sum)
    # Update weight
    w ← w.∗factor
    # Shift operation to w
    w ← bitshift (w, constant3)
  end while
```

rate; $post_j$ is the post-synaptic trace of excitatory neuron j. Since λ_{pre} is a constant coefficient, the decay of synaptic weight can be realized by the bit-shift operation.

When the post-synaptic neuron fires a spike, post is reset to its initial value, and the synaptic weight is updated according to:

$$w_{ij}(t) = w_{ij}(t-1) + \lambda_{post} \times pre_i(t-1) \tag{16}$$

where λ_{post} is the post-synaptic learning rate; pre_i is the pre-synaptic trace of input neuron i. The symbols for variables above are summarized in Table 1.

The original STDP rule is unstable [36], i.e., the learning process may not converge to a stable state. Therefore, a weight constraint scheme is introduced into the STDP rule to prevent the weights from growing unbounded. Specifically, the sum of the weights of all synapses connected to each neuron is restricted to the same magnitude, to guarantee that each excitatory neuron has a fair chance to win the competition. Moreover, the weights are clamped into the upper- and lower-boundaries. These strategies are executed together in each time step to prevent the weights from getting out of control. The working steps of the weight normalization is shown in Algorithm 1.

A delay connection matrix is required to obtain the summed weights (W) that should be added to the neurons' conductance at the current time step. Due to the presence of synaptic delays, synchro pre-synaptic spikes may arrive at the target neurons at different times, whereas non-synchro pre-synaptic spikes may reach the neuron at the same time.

As shown in the top panel of Fig. 4, the number of rows (i.e., 21) in the delay connection matrix is the sum of the maximum delay / time step (i.e., 10 ms / 0.5 ms = 20) and initial state (i.e., 1). The number of columns is equal to the number of excitatory neurons, and each column of the delay connection matrix corresponds to an excitatory neuron. Here, a delay connection vector of a single neuron [37] is used to illustrate the delay mechanism, as shown in the bottom panel of Fig. 4. The maximum delay and time step are assumed to be 10 ms and 0.5 ms, respectively. The delay connection vector is updated every time step. W pointed by the pointer denotes the summed weight which arrives at the excitatory neuron at the current time step. As an example, it is assumed that the pointer points to $W^{(0)}$ at the current time step and one of the pre-synaptic neurons connected to this neuron fires a spike. The weight of this synapse (w_0) and the delay time (assumed to be 4.5 ms) are retrieved from the weight matrix M_{xe} and the delay matrix (as shown in Fig. 1), respectively. Since w_0 will reach the neuron after 9 (i.e., 4.5 ms / 0.5 ms = 9) time steps, w_0 will be added to $W^{(9)}$. If there are any other pre-synaptic spikes, this delay connection vector will be updated in a similar way. Subsequently, $W^{(0)}$ will be added to this neuron's excitatory conductance and then reset to zero, and the pointer will move to the next cell and point to next weight ($W^{(1)}$). The above procedures are repeated until the end of the learning or inferencing process.

D. HARDWARE IMPLEMENTATION

To maintain the dynamic balance of the entire neural network, the parameters including conductance, threshold, trace, etc., always decline over time [38]. Here, all these decreasing processes are implemented by bit-shift operation. In the traditional SNN, these parameters normally decrease exponentially after reset. For example, the evolution of pre is governed by $d\text{pre}/dt = -\text{pre}/\tau$, where τ is a time constant. For hardware simplification, the above equation is converted into Eq. (9) according to the Euler's method:

$$\text{pre}(t) = \text{pre}(t-1) - \frac{dt \times \text{pre}(t-1)}{\tau_{pre}} \tag{17}$$

To further simplify the hardware implementation, a bit-shift method is used to approximate Eq. (17). For example, if we take dt= 0.5 ms and τ_{pre} = 8 ms, the division operation

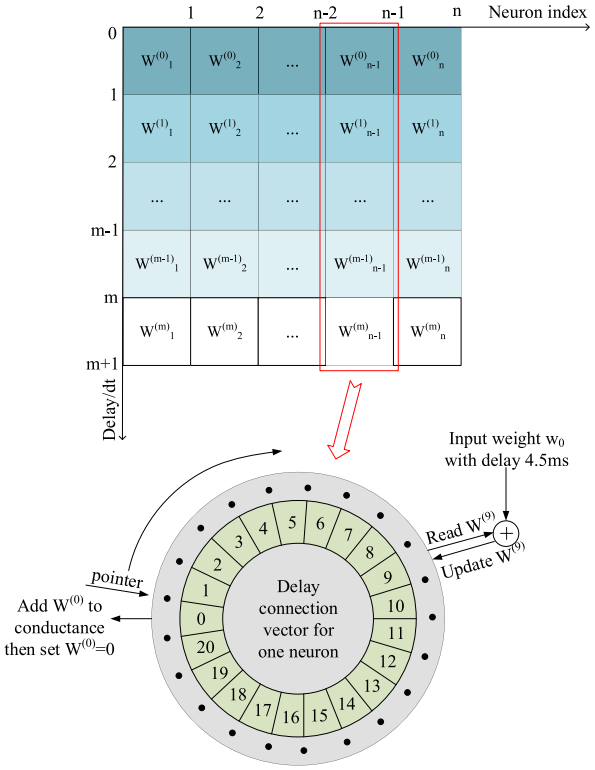

FIGURE 4. Schematic of delay connection matrix and delay mechanism.

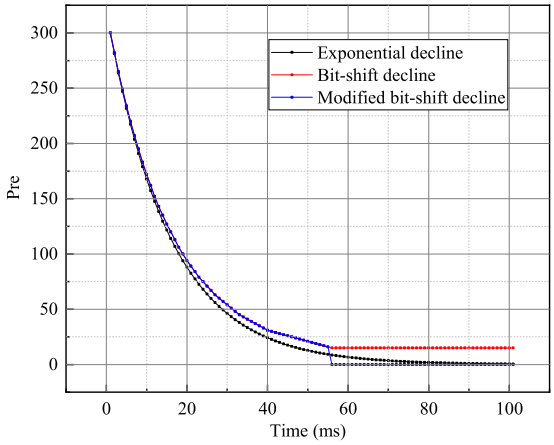

FIGURE 5. Comparison of exponential decline and bit-shift decline of pre.

can be replaced by a right shift 4-bit operation. As shown in Fig. 5, the bit-shift method (red line) approximates the equation (black line) well in the first 50 iterations. After 50 iterations, pre is reduced to less than 15, and it remains unchanged as it is impossible for it to be right-shift 4 bits. It is observed that this small error may result in non-convergence during the learning process. Fortunately, the non-convergence can be effectively addressed by forcing pre to 0 when it is less than 15, as shown in Fig. 5. Also, similar strategies are applied to other parameters with the behavior of exponential decay. This bit-shift variable updating method is much more computationally efficient than conventional exponential calculation [39].

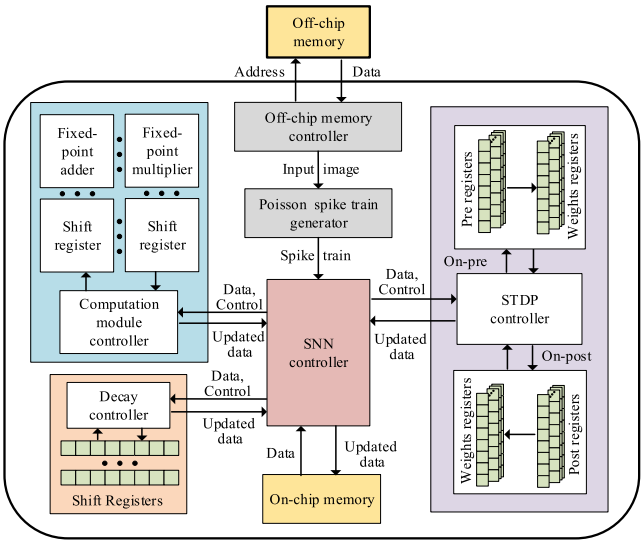

FIGURE 6. Schematic of the hardware implementation scheme of the STDP-based online learning SNN.

The schematic and corresponding working steps of the hardware implementation scheme of the STDP-based online learning SNN are given in Fig. 6 and Algorithm 2, respectively. To accelerate the learning process of the all multiple connections, parallel computing technique can be used in each step of Algorithm 2. Moreover, to save memory space and improve memory access efficiency, input images may be placed on off-chip memory, whereas other parameters can be placed on on-chip memory.

III. RESULTS AND DISCUSSIONS

A. LEARNING PROCESS

The SNN will be used to classify digital images after it learns from 60,000 images of the MNIST training set. The pixel of each image was encoded by a Poisson distributed spike train, whose firing rate is proportional to the pixel value, and a larger pixel corresponds to more spikes during the time window. Maximum input rate determines the maximum number of spikes during the time window. The numbers of excitatory neurons and inhibitory neurons are both 784. Fig. 7 shows the weight distribution of the 614,656 synapses (784×784) between the input layer and the excitatory layer after learning from 0, 10,000, and 60,000 images, respectively. The weights were uniformly distributed between 0 and 3×10^7 before learning. During the learning process, the weights gradually gathered to 0, which is similar to the sparse connectivity concept in the deep neural networks and is bio-plausible for each neuron only connects to a limited number of neurons in the biological neural networks [40]. Sparse connectivity reduces the complexity of the wiring between neurons and enables more efficient information processing and storage, and can also improve pattern recognition accuracy [41]. Moreover, the comparison of weight matrices before and after learning is also presented in Fig. 7. All weights connected to an excitatory neuron are reconstructed in 28×28 matrices.

Algorithm 2 Pseudo Code of a Possible Hardware Working Scheme of the STDP-Based Online Learning SNN

```
Memory initialization. # Storing images in off-chip memory, and traces, weights, delays, conductance, etc. in on-chip memory.
while i < number of images do
    Loading a new image and coding. # Loading image and coding by Poisson spike train generator.
    Weights normalization. # Loading weights and normalizing weights in the computation module, then returning updated weights to the on-chip memory.
    while t < time window do
        Weights modification according to STDP on-pre. # Loading weights and pre from the on-chip memory, and spike1 from Poisson spike train generator; implementing the weights reduction, and then returning the updated weights.
        Conductance updating. # Loading weights, delays, spike and conductance, and then calculating the conductance.
        Current updating. # Loading conductance and membrane potential, and then calculating and returning the current.
        Membrane potential updating. # Loading current and membrane potential, and then calculating and returning the updated membrane potential.
        Spike2 updating. # Loading membrane potential and threshold, and then calculating and returning the spike2.
        Weights modification according to STDP on-post. # Loading spike2, weights and post, and implementing the weights increment, and then returning the updated weights.
        Parameters bit-shift decay. # Loading pre, post, conductance and threshold, and implementing the bit-shift decay (controlled by the decay controller), and then returning the updated parameters.
    end while
end while
```

As the learning process proceeds, reconstructed matrices are displayed as specific digits, which means that the network tends to converge [42].

Fig. 8 shows the evolution of the thresholds of excitatory neurons during the learning process. Before learning, the thresholds of 784 neurons were all set to 2×10^8. After learning, the thresholds roughly distributed within $3.5 \times 10^8 \sim 5.5 \times 10^8$. Fig. 9 shows the membrane potential change during the learning process of a randomly selected excitatory neuron (Fig. 9(a)) and its corresponding inhibitory neuron (Fig. 9(b)). After the SNN learning from

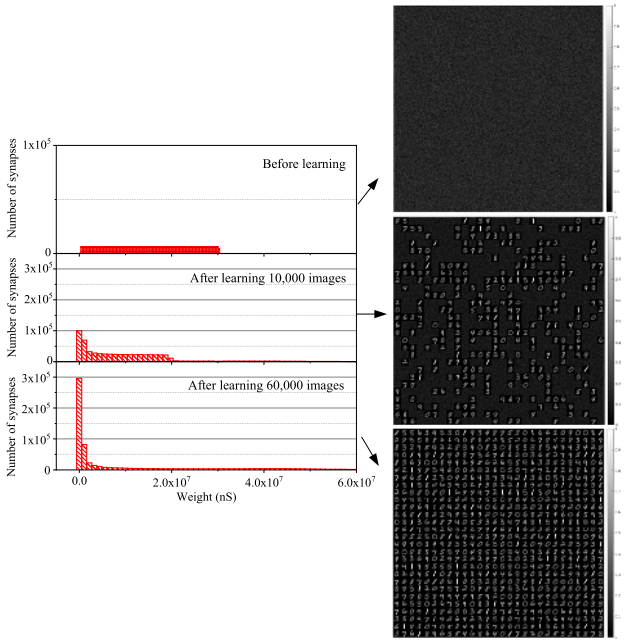

FIGURE 7. Weight distribution and evolution of weight matrices during the learning process.

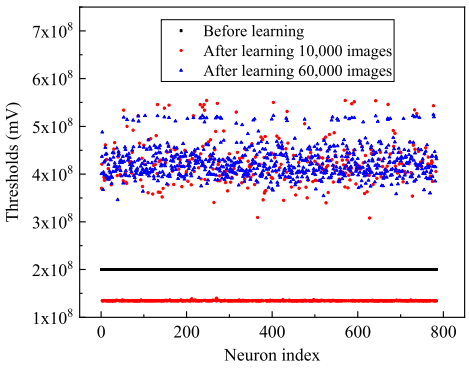

FIGURE 8. Threshold distribution during the learning process.

60,000 images, excitatory neurons were divided into ten classifications, and then each neuron was labeled according to how it responds to the ten types of inputs. Fig. 10 shows the number of each type of labels.

B. IMAGE CLASSIFICATION

The 10,000 images from the MNIST test set were used to test the performance of the neural network after learning. The excitatory layer also acts as the output layer during the inference process, so an extra classifier is not needed. The total number of spikes sent by each class of excitatory neurons for the current image will be added up and divided by the number of excitatory neurons in the class. The most significant one of excitatory neurons is identified as the winner, and the label corresponding to the winner is regarded as the final predicted result. For example, assuming that the current input is digit "0". As can be seen from Fig. 10, there are 110 excitatory neurons are assigned to digit "0". Then all spikes fired by these 110 neurons are added up and divided

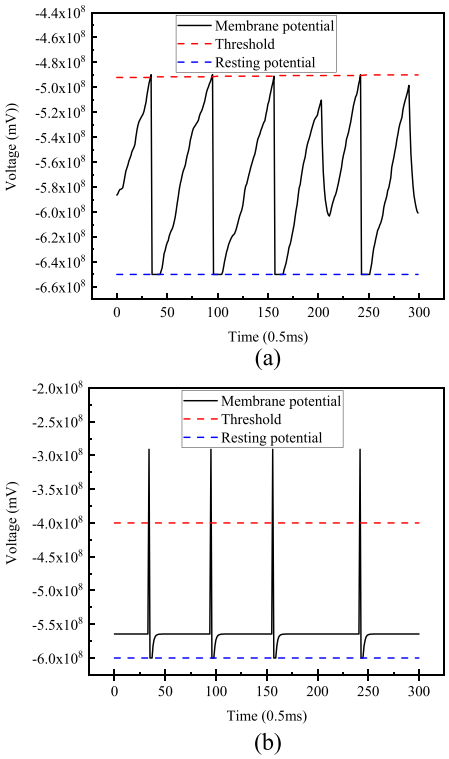

FIGURE 9. Waveforms of membrane potential during the learning process. (a) A randomly selected excitatory neuron and (b) its corresponding inhibitory neuron. The solid black line denotes the membrane potential; the dotted red and blue line denote the threshold and reset value, respectively.

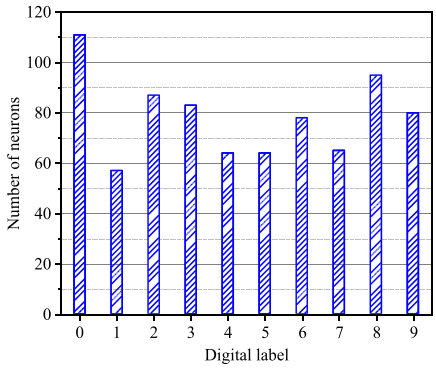

FIGURE 10. Number of each type of digital labels.

by 110 to get the average number of spikes per neuron. If the number is greater than all other nine classifications, the output of the network is determined to be digit ''0''. As shown in Fig. 11, the recognition accuracy of the SNN (consisting of 784 input neurons, 784 excitatory neurons, and 784 inhibitory neurons) gradually increases to 92.1% with the number of training images.

Fig. 12 shows statistics of the number of predicted digits and the corresponding target digits during the test process. It is observed that digits, such as ''9'', are prone to be erroneously predicted. Recognition accuracy as a function of the maximum input rate is shown in Fig. 13(a), which provide a guide to implement the proposed model in hardware with

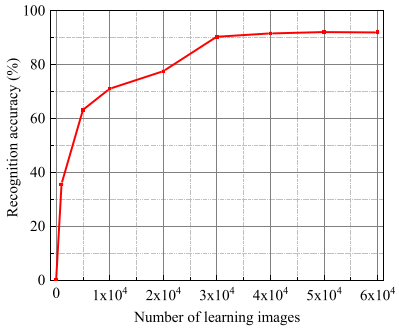

FIGURE 11. Recognition accuracy as a function of the number of training images.

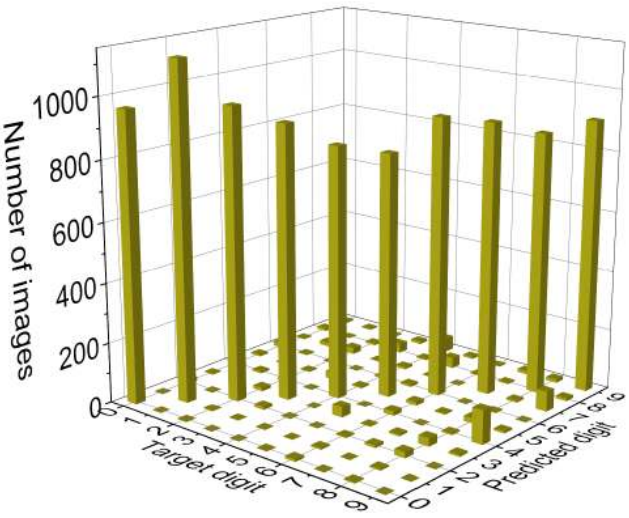

FIGURE 12. Statistics of predicted digits versus target digits.

fewer spikes while maintaining high accuracy (fewer spikes corresponding to less energy consumption in the neuromorphic hardware [7]). From Fig. 13(a), the recognition accuracy drops sharply when the maximum input rate decreases from 60 Hz to 50 Hz because that if the maximum input rate is lower than 60 Hz, some neurons cannot receive enough input to cross their thresholds to fire spikes, thus resulting in a significant loss of accuracy [15]. So the maximum input rate is set to 60 Hz in this work to achieve a balance between the accuracy and energy.

Also, it is observed that the recognition accuracy significantly increased from 90.0% to 94.5% with the number of excitatory neurons increasing from 400 to 3500, as shown in Fig. 13(b). For ASIC implementation, a trade-off between the recognition accuracy and hardware cost can be made according to this result. Although the maximum number of excitatory neurons in the ASIC is unchangeable because of the hardware limit, it can be designed to be downward compatible, i.e., the number of excitatory neurons can be set by changing the synaptic connection states (e.g., disconnect the unused synaptic connections). Moreover, the time consumption also increases greatly with the increase in the number of excitatory neurons, as shown in Table 2.

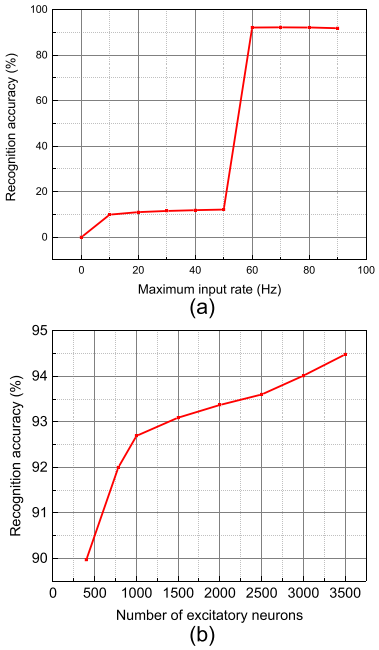

FIGURE 13. Recognition accuracy as a function of (a) the maximum input rate. (b) The number of excitatory neurons.

TABLE 2. Time consumption during the learning process.

Number of excitatory neurons	Number of learning cycles	Time consumption per cycle (hours)
400	1	1.6
784	1	2.3
1000	3	4.1
1500	3	6.3
2000	4	8.7
2500	4	11.3
3000	4	15.1
3500	5	17.8

TABLE 3. Comparison between the proposed model and traditional unsupervised ANN models considering the training time and performance on MNIST data set.

Models	Size	Learning time (hours)
Deep Belief Network (98.8%) [43]	784-500-500-10	5.7
Stacked Autoencoders (98.6%) [44]	784-1000-1000-1000-10	>17
Deep Boltzmann Machine (99.0%) [45]	784-500-100-10	19
This work (92.1%)	784-784-784	2.3

Table 3 compares the training time and performance on MNIST data set between the proposed model and pioneer ANN models with unsupervised learning methods. The proposed model achieves a 92.1% accuracy with a relatively fast learning speed. The simulation in this work is carried out with Matlab on Intel i9-7940x platform.

Table 4 shows the comparison of the cost of Multiplier/Divider and RAM between the proposed model and three pioneer SNNs achieved by supervised training, unsupervised learning, and ANN to SNN converting, respectively. Here, the Multiplier/Divider refers to the number of hardware calls

TABLE 4. Comparison between various SNN models considering the cost of multiplier/divider and RAM on MNIST data set.

SNN models	Methods	Multiplier/ Divider (size: 784-784-784)	RAM (MB) (size: 784-784-784)	Maximum accuracy (%)
Beyeler et al., 2013 [46]	Supervised STDP	2357	4.69	91.6 (size: 71026 neurons totally)
Diehl et al., 2015 [23]	Unsupervised STDP	2363	9.53	95.0 (size: 784-5000-5000)
O'Connor et al., 2013 [47]	Conversion	5488	9.41	94.1 (size: 784-500-500-10)
This work	Unsupervised STDP	3	2.78	94.5 (size: 784-3500-3500)

TABLE 5. Comparison of area cost and power performance between the floating-point and fixed-point multipliers [48].

Type	Area (μm^2) / percentage (%)	Power (μW) / percentage (%)
32-bit floating-point multiplier	7997.76 / 100.00	4229.60 / 100.00
16-bit fixed-point multiplier	1309.32 / 16.37	576.90 / 13.64

per neuron per time step, and the RAM denotes the on-chip memory, which stores the parameters except for the input data. The proposed model achieves a comparable accuracy compared with other models while consuming much fewer hardware resources (i.e., multiplier/divider and RAM). For example, the model in Reference [23] roughly requires 2,363 (STDP related: pre-synaptic neuron number 784 $\times$ post-synaptic neuron number 1 $=$ 784, conductance related: excitatory synapse number 784 $+$ inhibitory synapse number 784 $=$ 1568, exponential decline parameters: 10, weight normalization: 1) multipliers/dividers and 9.53 MB RAM (weights: pre-synaptic neuron number 784 $\times$ post-synaptic neuron number 784 $\times$ data width 64 $=$ 39337984 bits $=$ 4.69 MB, delay: 4.69 MB, others: 0.15 MB), whereas the model proposed in this work only requires 3 multipliers/dividers and 2.78 MB RAM.

Multiplication takes up most of the computations in the neural network [49]. The proposed model only requires three 16-bit fixed-point multipliers per neuron per time step, which dramatically reduces hardware requirements [50] (A 32-bit floating-point multiplier consumes 6.1 times chip area and 7.3 times power than a 16-bit fixed-point multiplier does under the 65 nm TSMC CMOS technology node [48], as given in Table 5). Because an MLP with bit operation runs seven times faster under MNIST dataset than with 32-bit floating-point multiplication on GPU [51], the proposed model can greatly accelerate the computation since most of the calculations of the proposed model are realized by bit-shift operations rather than multiplication or division.

Table 6 gives the resources and power comparison between the 32bits-floating-point model and the proposed model (since 32bits is the mostly used precision in the deep learning

TABLE 6. Comparison of hardware and power performance between the proposed model and the floating-point model (size: 784-784-784).

Type	Hardware resources (cell area) / percentage (%)	Power (mW) / percentage (%)
Floating-point model	35287800.67 / 100.00	120.184 / 100.00
Proposed model	20938892.40 / 59.34	76.536 / 63.68

framework [52]). Here the hardware synthesis is carried out with Design Compiler (DC) from Synopsys Inc. under the 55 nm SMIC CMOS process. Encouragingly, the proposed model reduces the hardware resources and power consumption by 40.7% and 36.3%, respectively.

IV. CONCLUSION

In this work, we propose a biologically plausible SNN model for hardware implementation, which can realize the recognition accuracy of 94.5% with the MNIST database. The model uses a hardware-friendly STDP mechanism to achieve self-learning. The neuron and synapse activities are digitalized and updated by fixed-point operations, and decay behaviors of the neuron and synapse parameters are realized by bit-shift operations, which can significantly reduce hardware cost and power consumption by 40.7% and 36.3%, respectively (model size: 784-784-784). This model provides a meaningful reference for designing a neuromorphic platform that can efficiently realize online learning. While there are still some challenges for online learning in neuromorphic chips. The weights of SNN require a large amount of on-chip storage space, which may be alleviated by recently proposed weight quantification method [53]; meanwhile, a large amount of time- and energy- consuming memory access are involved during the implementation of STDP, which may be eased by reducing the scale of weight matrixes by pruning method [54]. And these will be further studied in our future work.

REFERENCES

[1] W. Gerstner and W. Kistler, *Spiking Neuron Models: An Introduction*. Cambridge, U.K.: Cambridge, Univ. Press, 2002.
[2] G. Indiveri and S.-C. Liu, "Memory and information processing in neuromorphic systems," *Proc. IEEE*, vol. 103, no. 8, pp. 1379–1397, Aug. 2015.
[3] F. Walter, F. RÖhrbein, and A. Knoll, "Neuromorphic implementations of neurobiological learning algorithms for spiking neural networks," *Neural Netw.*, vol. 72, pp. 152–167, Dec. 2015.
[4] E. Painkras, L. A. Plana, J. Garside, S. Temple, S. Davidson, J. Pepper, D. Clark, C. Patterson, and S. Furber, "Spinnaker: A multi-core system-on-chip for massively-parallel neural net simulation," in *Proc. Custom Integr. Circuits Conf.*, Sep. 2012, pp. 1–4.
[5] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the SpiNNaker system architecture," *IEEE Trans. Comput.*, vol. 62, no. 12, pp. 2454–2467, Dec. 2013.
[6] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," 2017, *arXiv:1705.06963*. [Online]. Available: https://arxiv.org/abs/1705.06963
[7] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.
[8] A. S. Cassidy, P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, A. Amir, D. B.-D. Rubin, F. Akopyan, E. McQuinn, W. P. Risk, and D. S. Modha, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Aug. 2013, pp. 1–10.
[9] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. S. Maida, "Deep learning in spiking neural networks," 2018, *arXiv:1804.08150*. [Online]. Available: https://arxiv.org/abs/1804.08150
[10] P. O'Connor and M. Welling, "Deep spiking networks," 2016, *arXiv:1602.08323*. [Online]. Available: https://arxiv.org/abs/1602.08323
[11] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers Neurosci.*, vol. 10, p. 508, Nov. 2016.
[12] E. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Frontiers Neurosci.*, vol. 11, p. 324, Jun. 2017.
[13] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3227–3235, Jul. 2017.
[14] T. Liu, Z. Liu, F. Lin, Y. Jin, G. Quan, and W. Wen, "Mt-spike: A multilayer time-based spiking neuromorphic architecture with temporal error backpropagation," in *Proc. 36th Int. Conf. Comput.-Aided Design*, 2017, pp. 450–457.
[15] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8.
[16] E. Hunsberger and C. Eliasmith, "Spiking deep networks with LIF neurons," 2015, *arXiv:1510.08829*. [Online]. Available: https://arxiv.org/abs/1510.08829
[17] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1117–1125.
[18] D. Neil, M. Pfeiffer, and S.-C. Liu, "Learning to be efficient: Algorithms for training low-latency, low-compute deep spiking neural networks," in *Proc. 31st Annu. ACM Symp. Appl. Comput.*, 2016, pp. 293–298.
[19] J. Brader, W. Senn, and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural Comput.*, vol. 19, no. 11, pp. 2881–2912, 2007.
[20] C. Eliasmith, T. Stewar, X. Choo, T. Bekolay, T. DeWolf, C. Tang, D. Rasmussen, "A large-scale model of the functioning brain," *Science*, vol. 338, no. 6111, pp. 1202–1205, 2012.
[21] A. Tavanaei and A. Maida, "BP-STDP: Approximating backpropagation using spike timing dependent plasticity," *Neurocomputing*, vol. 330, pp. 39–47, Feb. 2019.
[22] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat, "Immunity to device variations in a spiking neural network with memristive nanodevices," *IEEE Trans. Nanotechnol.*, vol. 12, no. 3, pp. 288–295, May 2013.
[23] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers Comput. Neurosci.*, vol. 9, p. 99, Aug. 2015.
[24] D. Goodman and R. Brette, "Brian: A simulator for spiking neural networks in Python," *Frontiers Neuroinform.*, vol. 2, no. 5, p. 5, 2008.
[25] F. Sandin, A. I. Khan, A. G. Dyer, A. H. M. Amin, G. Indiveri, E. Chicca, and E. Osipov, "Concept learning in neuromorphic vision systems: What can we learn from insects?" *J. Softw. Eng. Appl.*, vol. 7, no. 5, pp. 387–395, 2014.
[26] Y. Sandamirskaya, "Dynamic neural fields as a step toward cognitive neuromorphic architectures," *Frontiers Neurosci.*, vol. 7, p. 276, Jan. 2014.
[27] M. B. Milde, H. Blum, A. Dietmüller, D. Sumislawska, J. Conradt, G. Indiveri, and Y. Sandamirskaya, "Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog/digital neuromorphic processing system," *Frontiers Neurorobot.*, vol. 11, p. 28, Jul. 2017.
[28] F. Corradi and G. Indiveri, "A neuromorphic event-based neural recording system for smart brain-machine-interfaces," *IEEE Trans. Biomed. Circuits Syst.*, vol. 9, no. 5, pp. 699–709, Oct. 2015.
[29] R. George, C. Mayr, G. Indiveri, and S. Vassanelli, "Event-based softcore processor in a biohybrid setup applied to structural plasticity," in *Proc. Int. Conf. Event-Based Control Commun. Signal Process.*, Jun. 2015, pp. 1–4.
[30] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybern.*, vol. 43, no. 1, pp. 59–69, 1982.

[31] A. Delorme, L. Perrinet, and S. J. Thorpe, "Networks of integrate-and-fire neurons using Rank Order Coding B: Spike timing dependent plasticity and emergence of orientation selectivity," *Neurocomputing*, vols. 38–40, pp. 539–545, Jun. 2001.
[32] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS Comput. Biol.*, vol. 3, p. e31, Feb. 2007.
[33] L. F. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron (1907)," *Brain Res. Bull.*, vol. 50, no. 5, pp. 303–304, 1999.
[34] G. Q. Bi and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neurosci.*, vol. 18, no. 24, pp. 10464–10472, 1998.
[35] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nature Neurosci.*, vol. 3, pp. 919–926, Sep. 2000.
[36] G. Chechik, I. Meilijson, and E. Ruppin, "Synaptic pruning in development: A computational account," *Neural Comput.*, vol. 10, pp. 1759–1777, Oct. 1998.
[37] J. Shen, D. Ma, Z. Gu, M. Zhang, X. Zhu, X. Xu, Q. Xu, Y. Shen, and G. Pan, "Darwin: A neuromorphic hardware co-processor based on spiking neural networks," *Sci. China Inf. Sci.*, vol. 59, pp. 1–5, Feb. 2016.
[38] L. Abbott and S. Song, "Temporally asymmetric hebbian learning, spike liming and neural response variability," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 69–75.
[39] J. P. David, K. Kalach, and N. Tittley, "Hardware complexity of modular multiplication and exponentiation," *IEEE Trans. Comput.*, vol. 56, no. 10, pp. 1308–1319, Oct. 2007.
[40] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.
[41] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proc. IEEE*, vol. 98, no. 6, pp. 1031–1044, Jun. 2010.
[42] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, no. 1, pp. 106–154, 1962.
[43] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
[44] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.
[45] R. Salakhutdinov and H. Larochelle, "Efficient learning of deep Boltzmann machines," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 693–700.
[46] M. Beyeler, N. D. Dutt, and J. L. Krichmar, "Categorization and decision-making in a neurobiologically plausible spiking network using a STDP-like learning rule," *Neural Netw.*, vol. 48, pp. 109–124, Dec. 2013.
[47] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers Neurosci.*, vol. 7, p. 178, Oct. 2013.
[48] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *Proc. 19th Int. Conf. Archit. Support Program. Lang. Operating Syst.*, 2014, pp. 269–284.
[49] M. Marchesi, G. Orlandi, F. Piazza, and A. Uncini, "Fast neural networks without multipliers," *IEEE Trans. Neural Netw.*, vol. 4, no. 1, pp. 53–62, Jan. 1993.
[50] M. Baesler and T. Teufel, "FPGA implementation of a decimal floating-point accurate scalar product unit with a parallel fixed-point multiplier," in *Proc. Int. Conf. Reconfigurable Comput. FPGAs*, Dec. 2009, pp. 6–11.
[51] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1," 2016, *arXiv:1602.02830*. [Online]. Available: https://arxiv.org/abs/1602.02830
[52] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah, "Comparative study of deep learning software frameworks," 2015, *arXiv:1511.06435*. [Online]. Available: https://arxiv.org/abs/1511.06435
[53] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.
[54] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 598–605.

G. C. QIAO received the B.S. degree in microelectronics from the University of Electronic Science and Technology of China, where he is currently pursuing the Ph.D. degree. His current research interests include neuromorphic computing and artificial intelligence.

S. G. HU received the Ph.D. degree in microelectronics from the University of Electronic Science and Technology of China, Chengdu, where he has been an Associate Professor, since 2016. His current research interests include thin-film transistors, and nonvolatile memory devices and their applications in artificial intelligence.

J. J. WANG received the B.S. degree in microelectronics from the University of Electronic Science and Technology of China, Chengdu, China, where he is currently pursuing the Ph.D. degree. His current research interests include digital circuit design, and nonvolatile memory devices and their applications in artificial intelligence.

C. M. ZHANG received the B.S. degree in microelectronics from the University of Electronic Science and Technology of China, China, where he is currently pursuing the M.S. degree. His current research interests include neuromorphic computation, neural network hardware, and digital IC design.

T. P. CHEN received the Ph.D. degree from The University of Hong Kong, Hong Kong, in 1994. He is currently an Associate Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

N. NING received the Ph.D. degree in microelectronics and solid state electronics from the University of Electronic and Science Technology of China. He has been with the School of Microelectronic and Solid State Electronics, University of Electronic Science and Technology of China, where he is currently a Professor of the Very Deep Sub Micrometer Integrated Circuit and System Lab. He holds three national patents and four utility mode patents. Furthermore, he has published over ten papers in important domestic and foreign academic journals, including two papers cited by SCI and five papers cited by EI. He has fulfilled three national projects, and he is doing five research projects. His research interests include AD/DA mixed integrated circuits, display panel drivers, and power managers.

Q. YU received the Ph.D. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2010, where he is currently a Professor and the Vice Dean of the School of Microelectronics and Solid-State Electronics.

Y. LIU received the B.Sc. degree in microelectronics from Jilin University, China, in 1998, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2005. From 2005 to 2006, he was a Research Fellow with Nanyang Technological University. In 2006, he was awarded the prestigious Singapore Millennium Foundation Fellowship. In 2008, he joined the School of Microelectronics, University of Electronic Science and Technology, China, as a Full Professor. He has authored or coauthored over 130 peer-reviewed journal papers and more than 100 conference papers. He holds one U.S. patent and more than 30 China patents also. His current research interests include memristor neural network systems, neuromorphic computing ICs, and AI-RFICs.