Chapter 11
Energy Efficient Spiking Neural Network Design with RRAM Devices

Yu Wang, Tianqi Tang, Boxun Li, Lixue Xia, and Huazhong Yang

11.1 Introduction

The explosion of big data brings huge demands for higher processing speed, lower power consumption, and better scalability of computing systems. However, the traditional "scaling down" method is approaching its limit, making it more and more difficult for CMOS-based computing systems to achieve considerable performance improvements from device scaling [1]. Moreover, from the architecture level, the memory bandwidth required by high-performance CPUs has also increased beyond what conventional memory architectures can efficiently provide, leading to an ever-increasing memory wall [2] challenge to the efficiency of von Neumann architecture. In this way, new technologies, from both the device level and the architecture level, are required to overcome these challenges.

The spiking neural network (SNN) is an emerging computing model, as shown in Fig. 11.1, which encodes and processes information with time-encoded neural signals [3]. As a bio-inspired architecture abstracted from actual neural system, SNN not only provides a promising solution to deal with cognitive tasks, such as the object detection and speech recognition, but also inspires new computational paradigms beyond the von Neumann architecture and boolean logics, which can drastically promote the performance and efficiency of computing systems [4, 5]. However, an energy efficient hardware implementation and the difficulty of training the model remain as two important impediments that limit the application of SNN.

On the one hand, we need an applicable computing platform to utilize the potential ability of SNN. IBM [6] proposes a neurosynaptic core named TrueNorth. To mimic the ultra-low-power processing of brain, TrueNorth uses several approaches

Y. Wang (✉) • T. Tang • B. Li • L. Xia • H. Yang
Department of Electronic Engineering, Tsinghua University, Beijing, China
e-mail: yu-wang@mail.tsinghua.edu.cn; ttq14@mails.tsinghua.edu.cn; lbx13@mails.tsinghua.edu.cn; xialx13@mails.tsinghua.edu.cn; yanghz@mails.tsinghua.edu.cn

A. Chattopadhyay et al. (eds.), *Emerging Technology and Architecture for Big-data Analytics*, DOI 10.1007/978-3-319-54840-1_11

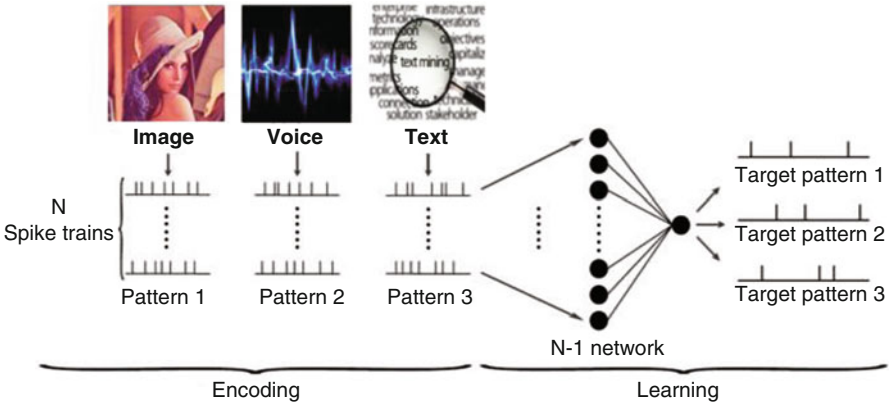

Fig. 11.1 Spiking neural network

to reduce the power consumption. Specifically, TrueNorth uses digital messages between neurons to reduce the communication overhead and event-driven strategy to further save the energy computation [5]. However, the CMOS based implementation still has some limitations that are hard to avoid, while some RRAMs' inherent advantages can overcome these difficulties. First, on-chip SRAM, where the synapse information is stored, is a kind of volatile memory with considerable leakage power, while RRAM is non-volatile with very low leakage power [7]. Another limitation is that TrueNorth may still need adders to provide the addition operation of neuron function, but RRAM crossbar can do the addition, or the matrix–vector multiplication, with ultra-high energy efficiency by naturally combining the computation and memory together [8–10]. Consequently, RRAM shows potential on implementing low-power spiking neural network.

On the other hand, from the perspective of algorithm, the efficient training of SNN and mapping a trained SNN onto neuromorphic hardware present unique challenges. Recent work of SNN mainly focuses on increasing the scalability and level of realism in neural simulation by modeling and simulating thousands to billions of neurons in biological real time [11, 12]. These techniques provide promising tools to study the brain but few of them support practical cognitive applications, such as the handwritten digit recognition. Even TrueNorth [13] uses seven kinds of applications to verify its performance, but the training and mapping methods for spike-oriented network are not discussed in detail. In other words, the mapping problem and efficient training method for SNN, especially for the real-world applications, to achieve an acceptable cognitive performance is severely demanded. Moreover, SNN can also be used for brain system simulation. For example, IBM made the cat cortex simulation (with $\sim 10^9$ neurons and $\sim 10^{13}$ synapses) on Blue Gene supercomputer cluster (with 147,456 CPUs and 144 TB memory) [14]. And such applications in the field of biological researches are out of discussion in this chapter.

These two problems are always coupled together and only by overcoming these two challenges can we actually utilize the full power of SNN for real-time data processing applications. In this chapter we discuss these two problems with the RRAM based system architecture and two different offline training algorithms of SNN. We use the MNIST digit recognition task [15] as an application example for the real-time classification. The goal of this chapter is to design an RRAM-based SNN system with higher classification accuracy and to analyze its strengths and weaknesses compared with other possible implementations.

The rest of this chapter is organized as follows:

- Section 11.2 introduces the background knowledge, including SNN and RRAM.
- Section 11.3 compares different models of spiking neural networks for practical cognitive tasks, including the Spike Timing Dependent Plasticity (STDP), the Remote Supervised Method (ReSuMe), and the latest Neural Sampling Learning Scheme. We show that the neural sampling method which transfers the ANN to SNN is promising for real-word applications while STDP and ReSuMe can **NOT** be used alone in the classification task since both of them are unsupervised learning method.
- Section 11.4 shows an RRAM-based implementation of SNN architecture. Two different specific networks, i.e. (1) STDP cascaded with three-layered ANN and (2) four-layered SNN transferred from full-connected ANN, are built and mapped to our system. The RRAM implementation mainly includes an RRAM crossbar array working as network synapses, an analog design of the spiking neuron, an input encoding scheme, and a mapping algorithm to configure the RRAM-based spiking neural network. And these elements will be described separately.
- In Sect. 11.5, a case study of digit recognition tasks is introduced to evaluate the performance of RRAM-based SNN. We compare the power efficiency and recognition performance of SNN and the RRAM-based artificial neural network (ANN). The experiment results show that ANN can beat SNN on the recognition accuracy, while SNN usually requires less power consumption. Based on these results, we discuss the possibility of using boosting methods, which combine some weak SNN learners together, to further enhance the recognition accuracy for real-world application.

11.2 Preliminaries

11.2.1 Spike Neurons

The neuron is the basic building block of SNN. Different mathematical models of spiking neurons have been explored with different levels of computational efficiency and biological plausibility [16]. The model of *Leaky Integrate and Fire (LIF)* [17] is one of the most widely used models for its computing efficiency. In this model, a one-order differential function determines the state variable $V(t)$ and a

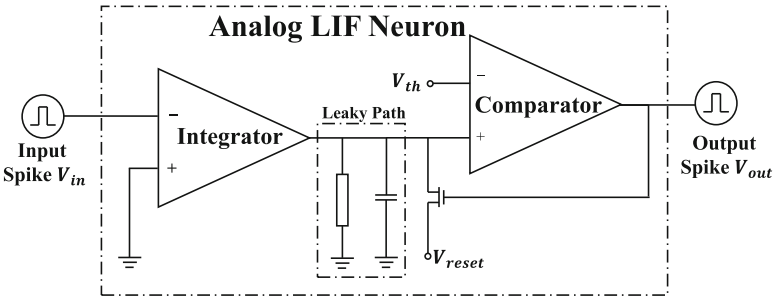

Fig. 11.2 Analog LIF neuron

threshold function determines whether the neuron spikes and then resets. And it is described as:

$$V(t) = \begin{cases} \beta \cdot V(t-1) + V_{\text{in}}(t) & \text{when } V < V_{\text{th}} \\ V_{\text{reset}} \text{and set a spike} & \text{when } V \geq V_{\text{th}} \end{cases} \tag{11.1}$$

where $V(t)$ is the state variable and β is the leaky parameter; V_{th} is the threshold state which the state variable makes comparison with and once exceeding, the state variable will reset to V_{reset}.

An analog *LIF* neuron implementation is shown in Fig. 11.2: the integrator calculates the state of the neuron $V(t)$ and the *RC* works as the leaky path. When $V(t) > V_{\text{th}}$, the transistor will be conducted and $V(t)$ will be reset.

11.2.2 RRAM Device Characteristics

Figure 11.3a shows a 2D filament model of HfO_x based RRAM device [18]. The model is a sandwich structure with a resistive layer between two metal electrodes. The conductance is exponentially dependent on the tunneling gap (d). Therefore, we will take advantage of the variable conductance of the RRAM device by setting the value of tunneling gap d. For the HfO_x based RRAM device, the I–V relationship can be empirically expressed as follows [18]:

$$I = I_0 \cdot \exp\left(-\frac{d}{d_0}\right) \cdot \sin h\left(\frac{V}{V_0}\right) \tag{11.2}$$

where d is the average tunneling gap distance. I_0 (∼1 mA), d_0 (∼0.25 nm) and V_0 (∼0.25 V) are fitting parameters through experiments. When $V << V_0$, there exists the approximation that $\sinh(\frac{V}{V_0}) \approx \frac{V}{V_0}$. The I–V relationship is linear under this condition. In this work, we will scale down the RRAM voltage to under 0.1 V in order to take advantage of the approximately linear I–V relationship.

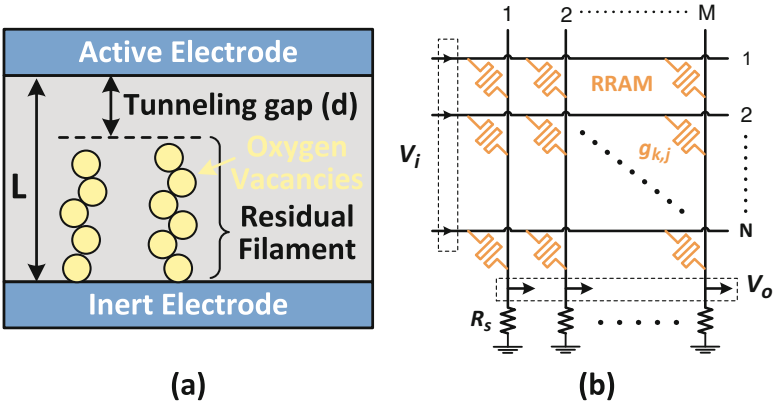

Fig. 11.3 (**a**) Physical model of the HfO_x based RRAM. The resistance of the RRAM device is determined by the tunneling gap distance d, and d will evolve due to the filed and thermally driven oxygen ion migration. (**b**) Structure of the RRAM Crossbar Array

As shown in Fig. 11.3b, the relationship between the input voltage vector ($\mathbf{V_i}$) and output voltage vector ($\mathbf{V_o}$) can be expressed as follows [19]:

$$V_{o,j} = \sum_{k} c_{k,j} \cdot V_{i,k} \tag{11.3}$$

where k ($k = 1, 2, \ldots, N$) and j ($j = 1, 2, \ldots, M$) are the index numbers of input and output ports, and the matrix parameter $c_{k,j}$ can be represented by the conductivity of the RRAM device ($g_{k,j}$) and the load resistors (g_s) as:

$$c_{k,j} = \frac{g_{k,j}}{g_s + \sum_{l=1}^{N} g_{k,l}} \tag{11.4}$$

The continuous variable resistance states of RRAM devices enable a wide range of weight matrices that can be represented by the crossbar. The precision of RRAM crossbar based computation may be limited by non-ideal factors, such as process variations, IR drop [20], drifting of RRAM resistance [21], etc. However, SNN only requires low precision of single synaptic value, meanwhile the binary input and LIF operation also alleviate the precision requirement of matrix vector multiplication. Therefore, the RRAM crossbar array is a promising component to realize matrix–vector multiplication for synapse weight computation in neural networks.

11.3 Training Scheme of SNN

The spiking neural network faces a huge problem that it is difficult to train the synaptic weights when applied in the real-world applications. In this section, we compare different SNN training algorithms, including the Spike Timing Dependent Plasticity (STDP), Remote Supervision Method (ReSuMe), and the latest Neural Sampling learning scheme. We show that the neural sampling method which transfers the ANN to SNN is promising for real-word applications while STDP and ReSuMe can **NOT** be used alone in the classification task since both of them are unsupervised learning method.

11.3.1 Spike Timing Dependent Plasticity (STDP)

Synapses connect neurons to each other and transmit signals between them. The synaptic weights, which determine the connecting strength of neurons, are learnable. Spike Timing Dependent Plasticity (STDP) [22] is an unsupervised learning rule that updates the synaptic weights as a function of the relative spiking time of pre- and post-synaptic neurons and the exponential window form of STDP is shown as:

$$\Delta w = \begin{cases} a^+ \cdot w_{ij}(1 - w_{ij}) \cdot \exp\left(-\frac{|t_j - t_i|}{\tau}\right) & \text{if } t_j \geq t_i \\ a^- \cdot w_{ij}(1 - w_{ij}) \cdot \exp\left(-\frac{|t_j - t_i|}{\tau}\right) & \text{if } t_j < t_i \end{cases} \tag{11.5}$$

where w_{ij} is the synaptic weight between pre- and post-synapse neuron n_i, n_j; t_i, t_j are the spiking time of neuron n_i, n_j; a is the maximum learning rate and τ is the time constant of the learning window. According to Eq. (11.5), the synaptic weight is limited in the interval of [0, 1]. The learning rate is decided by the time interval of n_i, n_j spiking: The closer between pre- and post-synaptic spikes, the larger the learning rate. The weight update direction is decided by which neuron spikes first: For the excitatory neuron, if the post-synaptic neuron n_j spikes later than n_i, the synapse will be strengthened; otherwise, it will be decayed; for the inhibitory neuron, vice versa. When every synaptic weight no longer changes or is set to 0/1, the learning process is finished. As an unsupervised method, STDP is mainly used as a feature extraction method. We cannot build a complete machine learning system only based on STDP. A classifier is usually required for practical recognition tasks. However, in our experiment, STDP method doesn't demonstrate enough efficiency of feature extraction. For example, we use the classic MNIST handwritten digit dataset [15] to test the performance with a support vector machine (SVM) [23] without a kernel, where two 50-dimension feature sets are extracted with STDP and principal component analysis (PCA). The PCA-SVM method achieves a recognition accuracy of 94% while the STDP-based method only reaches 91%. As PCA is usually the baseline for evaluating the performance of feature extraction, STDP does **NOT** demonstrate an efficient method for real-world cognitive applications or many other machine learning tasks.

11.3.2 Remote Supervision Method (ReSuMe)

Remote Supervision Method (ReSuMe) is a supervised learning method proposed in [24]. The algorithm introduces a supervised spike train for each synapse while training. The training process comes to an end if the post-synaptic spike train is the same as the supervised spike train. However, ReSuMe faces the difficulty on the pattern design of supervised spike trains and little guidance is offered on how to define the differences between different spike train. Although some papers [25] have attempted to build learning systems under ReSuMe learning algorithm, to the best of our knowledge, we have **NOT** seen any efficient way to solve a real-world task.

11.3.3 Neural Sampling Learning Scheme

The Neural Sampling learning scheme transforms the leaky Integrate-and-Fire (LIF) neuron into a nonlinear function (named Siegert function) [26] which represents the relationship between the input and output firing rate of a neuron, just as shown in Fig. 11.4. Moreover, Neftci demonstrates that nonlinear function, which is equivalent with LIF neuron, is satisfied with neural sampling conditions in [28] and can be approximated to sigmoid function under certain condition. Therefore, it can take advantage of contrastive divergence (CD), which is a classic algorithm exploited in restricted Boltzmann machine (RBM) to train the network. Moreover, the spiking RBM can be stacked into multi-layer to form the spiking deep belief network (DBN), which has demonstrated satisfying performance. In [26], Connor shows that a $784 \times 500 \times 500 \times 10$ spiking DBN achieves the recognition accuracy of 95.2% on MNIST dataset [15]. Recent research results show that it is unnecessary to introduce the Siegert approximation when transferring ANN to SNN, it is better for recognition accuracy if the ReLU neuron is introduced when training the original artificial network. The experiment results [29] show that spiking ConvNet achieves 99.1% accuracy on MNIST dataset when including ReLU neurons for original network training. The introduction of ReLU neuron makes it promising for high-performance large-scaled SNN model because of the better recognition accuracy for large-scaled ReLU-based artificial networks compared with Sigmoid-based (or tanh-based) ones.

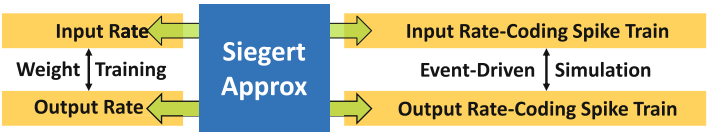

Fig. 11.4 Siegert approximation used in spiking neural network training [27]

In Sect. 11.4, we will make a hardware mapping of the spiking neural network which is trained under (1) STDP + three-layer ANN classier, (2) neural sampling learning method, to RRAM-based platform. The specific RRAM-based system is only used for forward (inference) process, while the training process done on the CPU platform will not be discussed in this work.

11.4 RRAM-Based Spiking Learning System

For an SNN system used for real-time classification applications, an offline training scheme is needed to decide the weights of the neural networks, i.e. coefficients in the crossbar matrix. To our best knowledge, there are two kinds of SNN training methods to build up classification systems: (1) unsupervised SNN training method, for example, Spike Timing Dependent Plasticity (STDP), is first introduced for extracting features; then the supervised classifier is introduced to finish the classification task. (2) First train an equivalent ANN using the gradient-based method, then transfer ANN to SNN and map SNN to the RRAM-based system for real-world applications. We design the two offline trained RRAM based SNN systems based on these two training methods [27, 30], and show them in the following subsections.

11.4.1 Unsupervised Feature Extraction + Supervised Classifier

As an unsupervised method, STDP is mainly used for feature extraction. We cannot build a complete classification system only based on STDP. A classifier is usually required for practical recognition tasks. Therefore, when mapping the system onto hardware, just as shown in Fig. 11.5, a five-layer neural network system is introduced: a two-layer spiking based neural network and a three-layer artificial neural network.

The first two layer SNN is trained using an unsupervised learning rule: spike timing dependent plasticity (STDP) [22], which updates the synaptic weights according to relative spiking time of pre- and post-synaptic neurons. The learning rate is decided by the time interval: the closer distance between pre- and post-synaptic spikes, the larger the learning rate. The weight updating direction is decided by which neuron spikes first: for the excitatory neuron, if the post-synaptic neuron spikes later, the synapse will be strengthened; otherwise, it will be decreased. When every synaptic weight no longer changes or is set to 0/1, the learning process is finished.

There is a converting module between the two layer SNN and 3-layer ANN to convert the spiking trains into the spike count vectors. Then the spike count vectors are sent into the following layers of the network (the 3-layer ANN). We use a 3-layer ANN as a classifier to process the features extracted from the input data by the

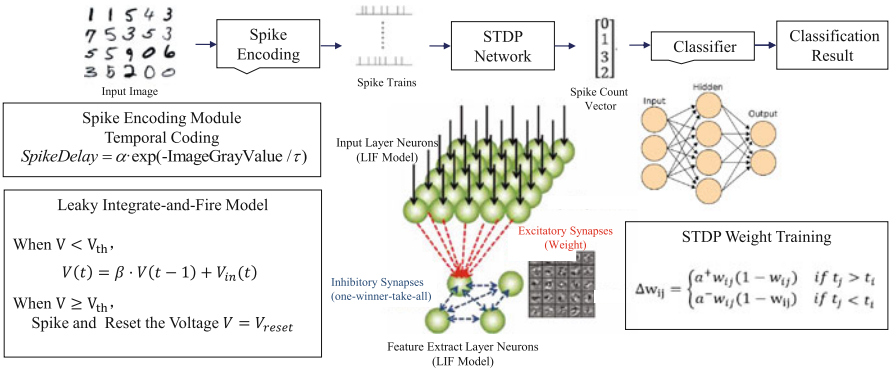

Fig. 11.5 System structure of unsupervised feature extraction + supervised classifier: 2-layer STDP based SNN + 3-layer ANN [27]

previous 2-layer SNN. We use the CMOS analog neuron in Sect. 11.2 for the LIF neuron; and the RRAM crossbar for synaptic computation in both 2-layer SNN (vector addition) and 3-layer ANN (matrix vector multiplication).

An experiment is made on MNIST digit recognition dataset to evaluate the performance of such system framework. The training algorithm is implemented on the CPU platform where LIF neurons are used in the first two layers and the sigmoid neurons are used in the last three layers. For the testing process (forward propagation of neural networks), we use circuit level simulation where the weight matrix is mapped to RRAM-based crossbar. Since the input images are 28×28 sized 256-level gray images, the first layer has 784 input channels. The five-layer spiking neural network system has five layers of neurons in all and the experiment result with the network size of "784×100SNN+100×50×10ANN" shows the recognition accuracy of 91.5% on CPU platform and 90% on RRAM-based crossbar model (circuit simulation result). The performance is a little worse than that of the three-layer ANN sized "784×100×10" with the recognition accuracy of 94.3% on CPU platform and 92% on RRAM-based crossbar model (circuit simulation result).

An interesting point comes from the energy consumption part, we find out that both ANN and SNN use the RRAM crossbar as the matrix vector multiplication part, ANN will consume more power than SNN with similar or even smaller neuron numbers. For example, the proposed "784×100SNN+100×50×10ANN" consumes 327.36 mW on RRAM while the power consumption increases to 2273.60 mW when we directly use "784×100×10ANN." The energy/power saving of SNN comparing ANN mainly comes from the different coding basis. The input voltage of SNN can be binary since it transforms the numerical information into the temporal domain, so there is no need for SNN to hold a large voltage range to represent multiple input states as implemented in ANN. ANN needs input voltages of 0.9 V, but SNN can work with much lower voltage supply (0.1 V). On the other hand, binary coding in SNN can avoid the usage of large number of AD/DA on the input and output interfaces, and the AD/DA consumes considerable large portion of power in the RRAM based NN systems [31].

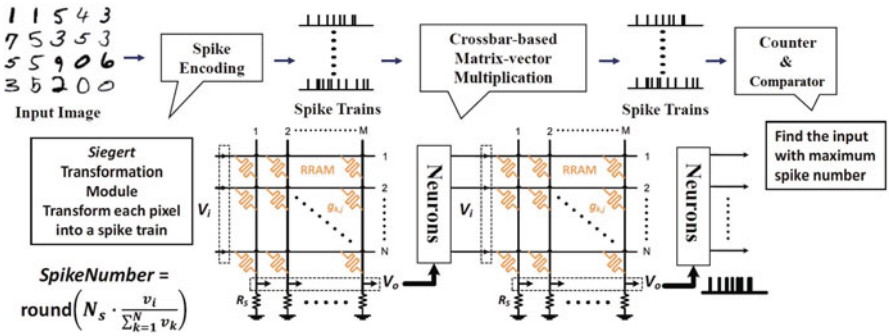

Fig. 11.6 System structure: transferring ANN to SNN—neural sampling method [30]

11.4.2 Transferring ANN to SNN: Neural Sampling Method

The neural sampling method provides a way to transfer ANN to SNN, thus offering a useful training scheme on classification tasks. A equivalent transformation is made between the nonlinear function (named Sigert function, which is similar to sigmoid function) of ANN and the Leaky Integrate-and-Fire (LIF) neuron of SNN. Therefore, it is possible to first train the ANN made up of the stacked Restricted Boltzmann Machine (RBM) structure using Contrastive Divergence (CD) method. In this way, a satisfying recognition accuracy of ANN can be first achieved. And then, the spike-based stacked RBM network with the same synaptic weight matrices can also be implemented for the classification tasks. The system structure is shown in Fig. 11.6 [30].

Since spike trains propagate in the spiking neural network, original input $x = [x_1, \ldots, x_N]$ should be mapped to spike trains $X(t) = [X_1(t), \ldots, X_N(t)]$ before running the test samples where $X_i(t)$ is a binary train with only two states 0/1. For the i^{th} input channel, the spike train is made of N_t spike pulses with each pulse width T_0, which implies that the spike train lasts for the length of time $N_t \cdot T_0$. Suppose the spike number of all input channels during the given time $N_t \cdot T_0$ is N_s, then the spike count N_i of the ith channel is allocated as:

$$N_i = \sum_{k=0}^{N_t-1} X_i(kT_0) = \text{round}\left(N_s \cdot \frac{v_i}{\sum_{k=1}^{N} v_k}\right) \tag{11.6}$$

which implies

$$\frac{N_i}{N_s} = \frac{v_i}{\sum_{i=1}^{N} v_i} \tag{11.7}$$

Then the N_i spikes of the ith channel is randomly set on the N_t time intervals. For an ideal mapping, we would like to have $N_i << N_t$ to keep the spike sparsity on

Table 11.1 Important parameters of the SNN system

Network size	$784 \times 500 \times 500 \times 10$
Number of input spike (N_s)	2000
Number of pulse interval (N_t)	128
Input pulse voltage (V)	1 V
The pulse width (T_0)	1 ns

the time dimension. However, for the speed efficiency, we would like the running time $N_t \cdot T_0$ to be short. Here, T_0 is defined by the physical clock, i.e. the clock of the pulse generator, which implies that we can only optimize N_t directly. Here, we define the bit level of the input as

$$\log\left(\frac{N_t}{\text{mean}(N_i)}\right) \tag{11.8}$$

which evaluates the tradeoff between time efficiency and the accuracy performance.

We train the SNN with the size of $784 \times 500 \times 500 \times 10$. And the parameters are shown in Table 11.1. The experiment results show that the recognition accuracy of MNIST dataset is 95.4% on the CPU platform and 91.2% on the ideal RRAM-based hardware implementation. The recognition performance decreases about 4% because it is impossible to satisfy with $N_t << N_s$ on the RRAM platform.

We show the results for recognition under different bit level quantization of input signal and RRAM devices, together with RRAM process variation and input signal fluctuation. The simulation results in Fig. 11.7a show that an 8-bit RRAM device is able to realize a recognition accuracy of nearly 90%. The simulation results in Fig. 11.7b show that the input signal above 6-bit level achieves satisfying recognition accuracy (>85%). Based on the 8-bit RRAM result, different levels of signal fluctuation are added on the 8-bit input signal. The result shown in Fig. 11.7c demonstrates that the performance of accuracy just decreases 3% given 20% variation. Figure 11.7d shows that when RRAM device is made in 8-bit level with the 6-bit level input, the performance does not decrease under 20% process variation. The sparsity of the spike train leads to the system robustness, making it insensitive to the input fluctuation and process variation.

The power consumption of the system is mainly contributed by three parts: the crossbar, the comparator, and the $R_{\text{mem}}C_{\text{mem}}$ leaky path. The simulation results show that the power consumption is about 3.5 mW on average. However, it takes $N_t = 128$ cycles with the physical clock $T_0 = 1$ ns. Though input conversion from numeral values to spike trains leads to about 100× clock rate decrease, the system is able to **complete the recognition task in real time** ($\sim$1 μs/sample), thanks to the short latency of RRAM device.

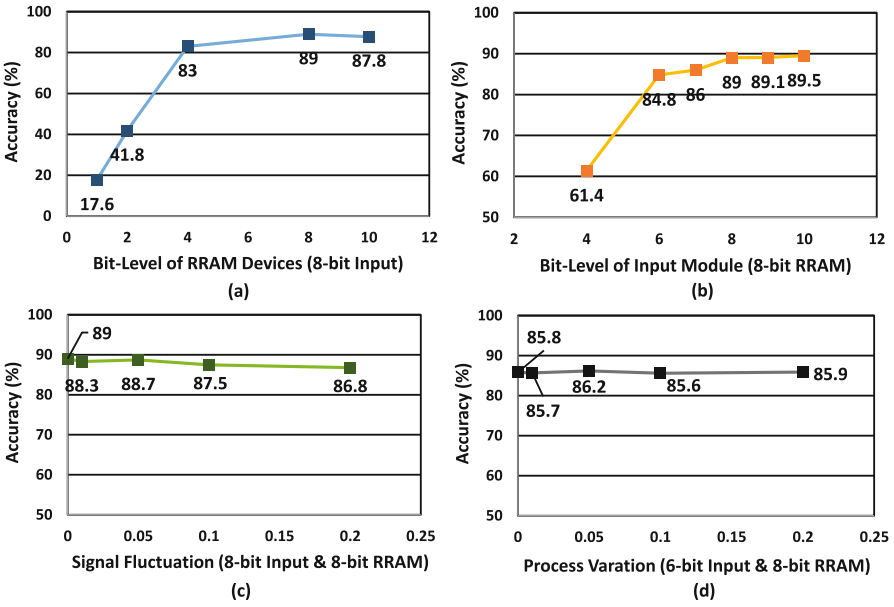

Fig. 11.7 Recognition accuracy under (**a**) different bit-level of RRAM devices, (**b**) different bit-level of input module, and (**c**) different degrees of input signal fluctuation, (**d**) different degrees of process variation of RRAM devices [30]

11.4.3 Discussion on How to Boost the Accuracy of SNN

The experiment results in the above subsections show that the recognition accuracy will decay after transferring an ANN to an SNN. However, due to the ultra-high integration density of the RRAM devices and the 0/1 based interfaces of SNN, SNN tends to consume much less circuit area and power compared with ANN. This result inspires us that we may integrate multiple SNNs with the same or even less circuit area and power consumption of ANN, and combine these SNNs together to boost the accuracy and robustness of the SNN system.

Previously, an ensemble method [31] is proposed to boosting the accuracy of RRAM-based ANN systems, named SAAB (Serial Array Adaptive Boosting), which is inspired by the AdaBoost method [32]. The basic idea of AdaBoost, which is also its major advantage, is to train a series of learners, such as ANNs or SNNs, sequentially, and every time we train a new learner, we try to "force" the new learner to pay more attention to the "hard" samples incorrectly classified by previous trained learners in the training set. The proposed technique can improve the accuracy of ANN by up to 13.05% on average and ensure the system performance under noisy conditions in approximate computation applications.

SAAB boost the computation accuracy at the cost of consuming more power and circuit area. As SNN usually consumes much less area and power compared with the

ANN, there is a chance to integrate multiple SNNs under the same circuit resource limitation of ANN. And these SNNs can be boosted together by the similar idea of SAAB. However, the inherent attributions of SNN systems should be considered when designing the boosting algorithm. According to our observation, there are two types of errors in the SNN-based classification tasks: (1) a traditional type: more than one neuron in the output layer spikes and the neuron spiking the most is not the target neuron; and (2) a special type of SNN: no neuron in the output layer spikes; It is interesting to observe that most of the wrong trials are the special type and it can be reduced slightly when increasing the input spike counts. We regard such samples as the difficult classifying cases. When seeking for the possibility to make up the performance loss after transferring ANN to SNN with a boosting-based method, this problem should be considered.

11.5 Conclusion

In this chapter, we first introduce the background knowledge of SNN and metal-oxide resistive switching random-access memory (RRAM). Then, we compare different training algorithms of SNN for real-world applications, and demonstrate that the Neural Sampling method is much more effective than other methods. We also explore the performance and energy efficiency by building the SNN-based energy efficient system for real time classification with RRAM devices. We implement different training algorithms of SNN, including Spiking Time Dependent Plasticity (STDP) and Neural Sampling method. Our RRAM-based SNN systems for these two training algorithms show good power efficiency and recognition performance on real-time classification tasks, e.g., the MNIST digit recognition. Finally, we discuss a possible direction to further improve the classification accuracy by boosting multiple SNNs.

However, there are still many challenges remaining in this spiking neural network structure. For example, the encoding mechanism from original data to spiking is not quite clear. It perhaps has a huge effect on system performance and power efficiency. Thus, how to design a proper encoding mechanism is one possible method of improving the performance of the system. In addition, the non-ideal circuit condition (e.g., the interconnection effect, the input variation) should be considered for future RRAM-based system design.

Acknowledgements This work was supported by 973 project 2013CB329000, National Science and Technology Major Project (2011ZX03003-003-01, 2013ZX03003013-003) and National Natural Science Foundation of China (Nos. 61373026, 61261160501, 61271269), and Tsinghua University Initiative Scientific Research Program. And we gratefully acknowledge the support from Prof. Shimeng Yu with the help of RRAM model.

References

1. L. Chang, Y.-K. Choi, D. Ha, P. Ranade, S. Xiong, J. Bokor, C. Hu, T.-J. King, Extremely scaled silicon nano-CMOS devices. Proc. IEEE **91**(11), 1860–1873 (2003)
2. W.A. Wulf, S.A. McKee, Hitting the memory wall: implications of the obvious. ACM SIGARCH Comput. Arch. News **23**(1), 20–24 (1995)
3. W. Maass, Networks of spiking neurons: the third generation of neural network models. Neural Netw. **10**(9), 1659–1671 (1997)
4. T. Masquelier, S.J. Thorpe, Unsupervised learning of visual features through spike timing dependent plasticity. PLoS Comput. Biol. **3**(2), e31 (2007)
5. D. Querlioz, W. Zhao, P. Dollfus, J. Klein, O. Bichler, C. Gamrat, Bioinspired networks with nanoscale memristive devices that combine the unsupervised and supervised learning approaches, in *2012 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)* (IEEE, 2012), pp. 203–210
6. P.A. Merolla, J.V. Arthur, R. Alvarezicaza, A.S. Cassidy, J. Sawada, F. Akopyan, B.L. Jackson, N. Imam, C. Guo, Y. Nakamura et al., A million spiking-neuron integrated circuit with a scalable communication network and interface, Science **345**(6197), 668–673 (2014)
7. B. Govoreanu, G.S. Kar, Y. Chen, V. Paraschiv, $10\times10\,nm^2$ HF/HFOx crossbar resistive ram with excellent performance, reliability and low-energy operation, *Electron Devices Meeting IEDM Technical Digest. International* (2011), pp. 31.6.1–31.6.4
8. B. Li, Y. Shan, M. Hu, Y. Wang, Memristor-based approximated computation, in *IEEE International Symposium on Low Power Electronics and Design* (2013), pp. 242–247
9. T. Tang, R. Luo, B. Li, H. Li, Energy efficient spiking neural network design with RRAM devices, in *International Symposium on Integrated Circuits* (2015), pp. 268–271
10. T. Tang, L. Xia, B. Li, R. Luo, Spiking neural network with RRAM: can we use it for real-world application? in *Design, Automation and Test in Europe* (2015), pp. 860–865
11. R. Wang, T.J. Hamilton, J. Tapson, A. Van Schaik, An FPGA design framework for large-scale spiking neural networks, in *Proceedings - IEEE International Symposium on Circuits and Systems* (2014), pp. 457–460
12. E. Painkras, L.A. Plana, J. Garside, S. Temple, Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation. IEEE J. Solid-State Circ. **48**(8), 1943–1953 (2013)
13. S.K. Esser, A. Andreopoulos, R. Appuswamy, P. Datta, D. Barch, A. Amir, J. Arthur, A. Cassidy, M. Flickner, P. Merolla, Cognitive computing systems: algorithms and applications for networks of neurosynaptic cores, in *The 2013 International Joint Conference on Neural Networks (IJCNN)* (2013), pp. 1–10
14. D. Kuzum, R.G. Jeyasingh, B. Lee, H.-S.P. Wong, Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. Nano Letters **12**(5), 2179–2186 (2011)
15. Y. Lecun, C. Cortes, The MNIST database of handwritten digits (1998)
16. E.M. Izhikevich, Which model to use for cortical spiking neurons? IEEE Trans. Neural Netw. **15**(5), 1063–1070 (2004)
17. G. Indiveri, A low-power adaptive integrate-and-fire neuron circuit, in *ISCAS (4)* (2003), pp. 820–823
18. S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, H.-S.P. Wong, Stochastic learning in oxide binary synaptic device for neuromorphic computing. Front. Neurosci. **7**,186 (2013)
19. M. Hu, H. Li, Q. Wu, G.S. Rose, Hardware realization of BSB recall function using memristor crossbar arrays, in *Proceedings of the 49th Annual Design Automation Conference* (ACM, 2012), pp. 498–503
20. P. Gu, B. Li, T. Tang, S. Yu, Y. Wang, H. Yang, Technological exploration of rram crossbar array for matrix-vector multiplication, in *ASP-DAC* (2015)
21. B. Li, Y. Wang, Y. Chen, H.H. Li, H. Yang, Ice: inline calibration for memristor crossbar-based computing engine, in *Proceedings of the conference on Design, Automation & Test in Europe* (European Design and Automation Association, 2014), p. 184

22. S. Song, K.D. Miller, L.F. Abbott, Competitive hebbian learning through spike-timing-dependent synaptic plasticity. Nat. Neurosci. **3**(9), 919–926 (2000)
23. C.-C. Chang, C.-J. Lin, Libsvm: a library for support vector machines. ACM Trans. Intell. Syst. Technol. (TIST) **2**(3), 27 (2011)
24. F. Ponulak, ReSuMe - new supervised learning method for spiking neural networks, Institute of Control and Information Engineering, Poznan University of Technology. Available online at: http://d1.cie.put.poznan.pl/~fp/research.html (2005)
25. J. Hu, H. Tang, K.C. Tan, H. Li, L. Shi, A spike-timing-based integrated model for pattern recognition. Neural Comput. **25**(2), 450–472 (2013)
26. P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, M. Pfeiffer, Real-time classification and sensor fusion with a spiking deep belief network. Neuromorphic Eng. Syst. Appl **61**, 1–10 (2015)
27. T. Tang, R. Luo, B. Li, H. Li, Y. Wang, H. Yang, Energy efficient spiking neural network design with RRAM devices, in *14th International Symposium on Integrated Circuits (ISIC)* (IEEE, 2014), pp. 268–271
28. E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, G. Cauwenberghs, Event-driven contrastive divergence for spiking neuromorphic systems (2013). Preprint, arXiv:1311.0966
29. P.U. Diehl, D. Neil, J. Binas, M. Cook, Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing, in *International Joint Conference on Neural Networks* (2015)
30. T. Tang, L. Xia, B. Li, R. Luo, Y. Wang, Y. Chen, H. Yang, Spiking neural network with RRAM: can we use it for real-world application? In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition* (EDA Consortium, 2015), pp. 860–865
31. B. Li, L. Xia, P. Gu, Y. Wang, H. Yang, Merging the interface: power, area and accuracy co-optimization for RRAM crossbar-based mixed-signal computing system, in *Design Automation Conference* (2015), pp. 1–6
32. Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms* (CRC Press, 2012)