SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •
Special Focus on Near-memory and In-memory Computing

June 2021, Vol. 64 160408:1–160408:12
https://doi.org/10.1007/s11432-020-3204-y

Bayesian neural network enhancing reliability against conductance drift for memristor neural networks

Yue ZHOU[1], Xiaofang HU[2], Lidan WANG[1] & Shukai DUAN[2*]

[1]*College of Electronics and Information Engineering, Southwest University, Chongqing* 400715*, China;*
[2]*College of Artificial Intelligence, Southwest University, Chongqing* 400715*, China*

Received 30 December 2020/Revised 4 March 2021/Accepted 8 March 2021/Published online 26 April 2021

Abstract The hardware implementation of neural networks based on memristor crossbar array provides a promising paradigm for neuromorphic computing. However, the existence of memristor conductance drift harms the reliability of the deployed neural network, which seriously hinders the practical application of memristor-based neuromorphic computing. In this paper, the impact of different types of conductance drift on the weight realized by memristors is investigated and analyzed. Then, utilizing the weight uncertainty introduced by conductance drift, we propose a weight optimization method based on the Bayesian neural network, which can greatly improve the network performance. Furthermore, an ensemble approach is proposed to enhance network reliability without increasing training cost or crossbar array resources. Finally, the effectiveness of the proposed scheme is verified through a series of experiments. In addition, the proposed scheme can be easily integrated into the implementation of neuromorphic computing, which can provide a better guarantee for its large-scale application.

Keywords conductance drift, neuromorphic computing, Bayesian neural network, memristor crossbar array, network reliability

Citation Zhou Y, Hu X F, Wang L D, et al. Bayesian neural network enhancing reliability against conductance drift for memristor neural networks. Sci China Inf Sci, 2021, 64(6): 160408, https://doi.org/10.1007/s11432-020-3204-y

1 Introduction

Deep neural networks (DNNs) have developed rapidly and have been applied in image recognition, speech recognition, natural language processing, and other fields [1–3]. However, based on the traditional von Neumann architecture, the training and inference of DNNs consume a lot of energy and have high latency due to the separation of computing and storage [4,5]. It is hard to deploy DNNs-based applications in scenarios that require low energy consumption and low latency. Memristor-based neuromorphic computing provides a promising approach to solve this problem.

Different from the traditional computing architecture based on complementary metal-oxide-semiconductor (CMOS), neuromorphic computing uses non-volatile resistive memory to realize the integration of data storage and computation [6]. Therefore, it can avoid the problem of the memory wall caused by frequent data exchange between memory and computing units and greatly reduce power consumption and latency of data transfer [7]. Memristor, which can easily store and adjust the resistance value, is one of the foundational devices for neuromorphic computing. Applying memristor neural network (MNN) based on crossbar arrays is a common way to deploy neural network to hardware [8]. Research shows that this architecture can achieve significant advantages in speed and power consumption over CMOS architecture [9, 10]. However, due to the inevitable hardware imperfections, such as conductance drift, the memristor conductance cannot represent the weight of the neural network accurately and steadily, which seriously degrades network performance [4, 11].

Multiple factors could cause conductance drift. When using memristor-based neural networks for inference, although a single input voltage has a small effect on the conductance, its cumulative effect is

[*] Corresponding author (email: duansk@swu.edu.cn)

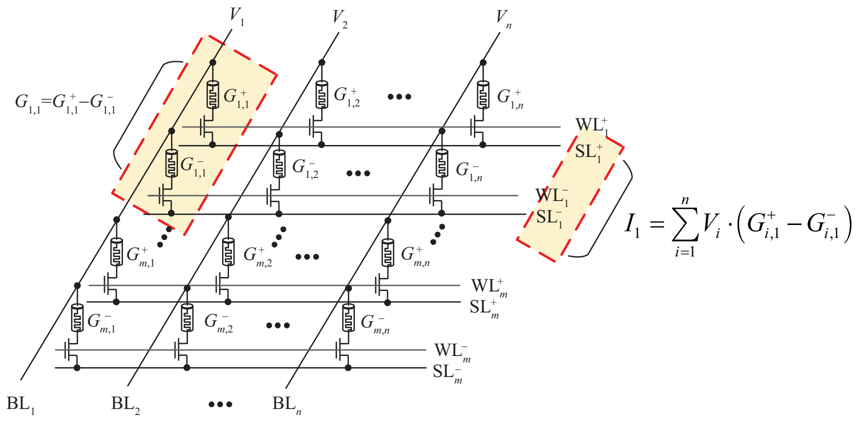


Figure 1 (Color online) The main structure of the memristor crossbar array. WL, SL, and BL are the word line, source line, and bit line, respectively.

conspicuous [12]. External temperature also influences the conductance value, and it will drift gradually over time even when it is idle [13]. To solve this problem, researchers in materials science are exploring the fabrication of more stable memristors [14], while others use software and hardware methods to improve network reliability. For example, Ref. [12] compensated the conductance drift caused by input voltage by designing a real-time feedback controller. Ref. [15] proposed a sign backpropagation algorithm to reduce the influence of conductance drift for multilayer perceptron (MLP). Some researchers analyzed the impact of activation function, the number of hidden layers and filters on conductance drift [16].

Bayesian neural network (BNN) constructs the model based on a probabilistic perspective, and its parameters are not fixed but are subject to distributions [17]. The original intention of BNN is to improve the robustness and interpretability of the model [18]. In this paper, using the weight uncertainty of BNN, we propose a weight optimization and network ensemble scheme to improve the model's tolerance to weight deviation. To solve the problem that it is hard to train the BNN, a normal neural network is trained first, and then BNN is used to optimize its weights, which can not only ensure the model obtain higher accuracy but also make the model more robust. Then an ensemble approach is designed to simulate the sampling process of BNN and further improve the performance of the network. Finally, a series of experiments show the effectiveness of the proposed scheme.

2 Related work

2.1 Memristor neural network

A typical neural network consists of convolutional layers and full connection layers, and its basic operations can be decomposed into dot product and accumulation. These operations can be easily implemented by memristor crossbar arrays based on Ohm's law and Kirchhoff's law [19]. The main structure of the memristor crossbar array is shown in Figure 1. Since the network weights have positive and negative values, and the memristor conductance has no negative values, so a weight is usually represented by a pair of memristors [20]. Assuming that the input is n-dimensional and the output is m-dimensional. The difference between two memristor conductance $G_{i,k} = G^+_{i,k} - G^-_{i,k}$ represents one network weight. $(V_1, V_2, \ldots, V_n)$ and $(I_1, I_2, \ldots, I_m)$ denote the input voltages and the output currents, respectively. The corresponding dot product and accumulation operations are shown as

$$I_k = \sum_{i=1}^{n} V_i \cdot (G^+_{i,k} - G^-_{i,k}), \quad k \in (1, 2, \ldots, m). \tag{1}$$

Currently, there are two main deployment methods: ex-situ and in-situ training. The main steps of ex-situ training are as follows: firstly, perform network training on an external software platform to obtain network weights, then these weights are quantified according to the levels of conductance values, and finally, the quantized weights are written into the memristor crossbar array [13,21]. However, as memristors present cycle-to-cycle and device-to-device variations, it is hard to write the weights accurately [22].

The common approach to mapping weights to hardware is the write-verify method [15]. However, this method still inevitably introduces errors [13, 23]. The steps of in-situ training are relatively few, but the implementation is quite complicated. Firstly, the memristor value is initialized randomly, then the network training is carried out directly on the crossbar array, and the memristor value is adjusted in real time until it reaches the predetermined accuracy.

Ex-situ training can be easily deployed, but its accuracy is difficult to guarantee. In-situ training leads to a more accurate model, but its training process is complex, and the whole training process should be repeated for each model. Therefore, it cannot be applied on a large scale. The causes of weight deviation in ex-situ training mainly include quantization error, writing weight error, and the error caused by conductance drift after training. For in-situ training, the causes mainly include the errors introduced by weight adjustment during training and conductance drift after training. So ex-situ training will suffer more deviations than in-situ training. Therefore, the ex-situ training method is mainly discussed in this paper.

2.2 Conductance drift

As mentioned above, there are many causes for conductance drift. Different types of conductance drift will have various effects on weight deviation. Based on the TaO_x memristor model, Ref. [12] discussed how the conductance of the memristors changes and how it affects the weight change when applying voltages in different directions. The results showed that different voltage directions lead to various trends. Ref. [13] experimentally tested the weight changes of a crossbar array based on $\mathrm{TiN/TaO}_x\mathrm{/HfO}_x\mathrm{/TiN}$ memristors over 30 days. They found that the mean values of weights of different quantized levels would remain stable, although individual weights would drift significantly. Phase-change memory (PCM), as another memristor device, is also often used to construct electron synapses [24]. It is found that the conductance value of PCM decreases gradually over time [25, 26].

In general, there are two kinds of drift trends: the first is that the mean value of deviation is zero and the variance is getting bigger and bigger, while the second is that the mean value is always negative and the variance retains the same. Therefore, two simplified models are used to represent these two types of drifts as

$$W^* = \begin{cases} W + \Delta w_1, & \Delta w_1 \sim N(0, \sigma_1^2), \\ W - \Delta w_2, & \Delta w_2 \sim N(\mu_2, \sigma_2^2), \end{cases} \tag{2}$$

where σ_1 and μ_2 increase gradually, and σ_2 remains the same. Because different types of memristors have various properties, the relationship between weight drift and time will not be considered in this paper. Besides, errors are also introduced when writing weights to the crossbar, which can be considered as part of deviations in (2).

2.3 Bayesian neural network

The deep neural network considers weights as fixed values, and the goal of network optimization is to find the optimal values for the weights. From the perspective of BNN, network weights have no fixed truth values, and the optimization goal is to find the distributions of the weights [27, 28]. As shown in Figure 2, BNN makes inferences based on the weight distributions.

Given that the training data is D and the network weights are w, the optimization goal is to find the distribution $p(w|D)$. Theoretically, according to Bayes theorem, the weight distribution could be obtained through the following equation:

$$p(w|D) = \frac{p(w)p(D|w)}{p(D)} = \frac{p(w)p(D|w)}{\int p(w)p(D|w)\mathrm{d}w}, \tag{3}$$

where $p(w)$ is the prior distribution of w and $p(w|D)$ is the likelihood. Then based on the posterior distribution, Eq. (4) can be used to make inferences.

$$E[f] = \int f(w)p(w|D)\mathrm{d}w, \tag{4}$$

where $f(w)$ is the function of the network. However, the denominator in the posterior distribution generally has no analytical solution or is time-consuming to solve. Besides, the joint probability density

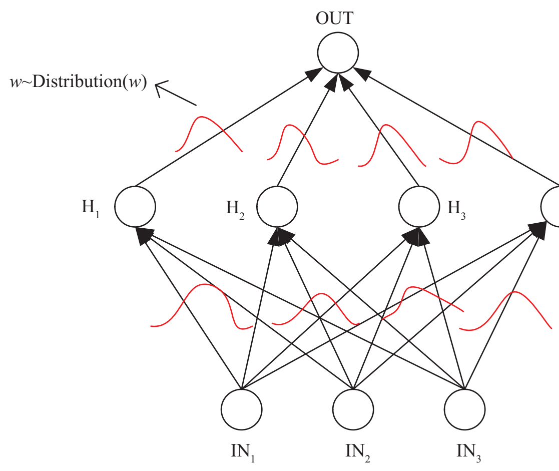


Figure 2 (Color online) Schematic of the Bayesian neural network.

$p(w)p(D|w) = p(w, D)$ is usually unknown. Therefore, the posterior distribution is usually not directly calculated using Eq. (3) but are approximated by other methods.

Hinton et al. [29] first proposed variational inference (VI) to obtain the approximate solution of the posterior distribution. VI is an approximate inference method, which uses an assumed simple distribution $q(w|\theta)$ to approximate the true posterior distribution $p(w|D)$, and then performs the optimization to find the appropriate parameters θ to make q as consistent as possible with p [30, 31]. As shown in Eq. (5), KL divergence is used to measure the similarity between the two distributions.

$$\mathrm{KL}(q(w|\theta))||p(w|D) = \int q(w|\theta)\log\frac{q(w|\theta)}{p(w|D)}\mathrm{d}w. \tag{5}$$

Eq. (5) is also hard to optimize directly, so it can be transformed as follows:

$$\begin{aligned}\mathrm{KL}(q(w|\theta))||p(w|D) &= \int q(w|\theta)\log\frac{q(w|\theta)p(D)}{p(D|w)p(w)}\mathrm{d}w\\ &= \int q(w|\theta)\log\frac{q(w|\theta)}{p(D|w)p(w)}\mathrm{d}w + \int q(w|\theta)\log p(D)\mathrm{d}w\\ &= \int q(w|\theta)\log\frac{q(w|\theta)}{p(w)}\mathrm{d}w - \int q(w|\theta)\log p(D|w)\mathrm{d}w + \log\ p(D)\\ &= \mathrm{KL}(q(w|\theta)||p(w)) - E_q[\log p(D|w)] + \log p(D).\end{aligned} \tag{6}$$

Since $\log p(D)$ is independent of parameter θ, the backpropagation algorithm can be used to optimize the first two items on the right of Eq. (6), whose negative number is called the evidence lower bound (ELBO) [32] as shown in Eq. (7). Therefore, the goal of minimizing the KL divergence between $q(w|\theta)$ and $p(w|D)$ is transformed into the goal of maximizing ELBO, which is tractable and can be optimized by stochastic gradient descent (SGD) [33].

$$\begin{aligned}\mathrm{ELBO}(q) &= -\mathrm{KL}(q(w|\theta)||p(w)) + E_q[\log p(D|w)]\\ &= E_q[\log p(D,w)] - E_q[\log q(w|\theta)]\\ &= E_q[\log p(w)] + E_q[\log p(D|w)] - E_q[\log q(w|\theta)].\end{aligned} \tag{7}$$

Mean-field variational Bayes (MFVB) is a common approach for the choice of assumed distribution [29]. MFVB hypothesizes that the posteriors of network weights are a series of independent Gaussian distributions, therefore,

$$q(w|\theta) = \prod_{i=1}^{K} N(w_i|\mu_i, \sigma_i^2), \tag{8}$$

where K is the number of the network weights. After obtaining the approximate distribution of the true posterior, Markov chain Monte Carlo (MCMC) can be used to sample the parameters and then make

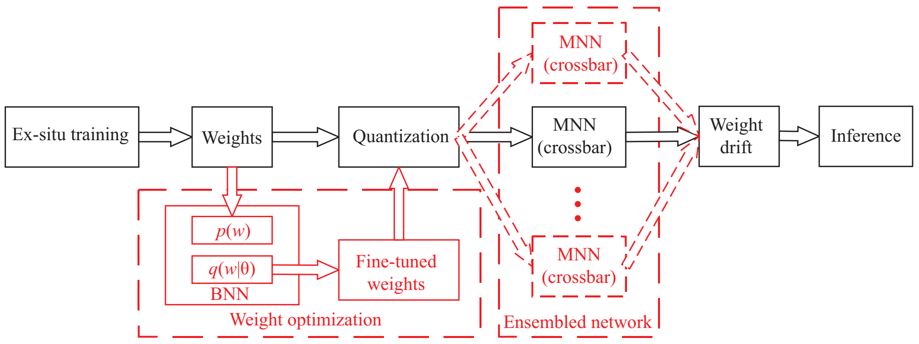


Figure 3 (Color online) The overall process of our proposed scheme based on BNN.

inferences, as shown in

$$E[f] \approx \frac{1}{N}\sum_{k=1}^{N} f(w_k), \tag{9}$$

where w_k denotes the kth sample from the distribution, and N is the total number of samples.

3 Weight optimization by Bayesian neural network

When an MNN suffers from conductance drift, its weight will deviate from the original values, and the possible weight values can be considered to follow some distribution. The actual weights can be viewed as samples from the distribution. Coincidentally, the weights of BNN are not fixed but are subject to some distribution. The difference is that the weight deviation caused by conductance drift is uncontrolled, while the weight uncertainty in BNN is intentional. A natural idea is to use BNN to replace the original neural network so that the model has better performance. However, compared with the deep neural network, the training of BNN is relatively troublesome, and the model accuracy is usually lower than that of the deep neural network [33].

Based on BNN, we propose a novel weight optimization and network ensemble scheme, which can improve the network reliability in the case of weight drift. The overall process of the proposed scheme is shown in Figure 3, in which the black block diagrams are the standard processing flow, and the red block diagrams are our improved flow. First, a DNN model is trained as the process of ex-situ training. Then a BNN with the same structure as the DNN is followed to optimize these weights before the quantization operation. Finally, an ensemble approach is designed to simulate the sampling from the distribution. This scheme not only retains the high accuracy of the DNN but also increases the network stability.

3.1 Ex-situ training

Firstly, network training is conducted following the ex-situ training method based on artificial neural networks. However, some tricks are designed to make the model more compatible with the memristor crossbar array. Since the weights need to be mapped to the crossbar array later, and the memristor conductance value is within a certain range, the weights should also be restricted within a certain range. Otherwise, individual weights that are too large would be truncated, or if all weights are retained, too many weights will be concentrated in the lower quantization levels. Besides, because one pair of memristors represent one weight in the crossbar array, the range of weights that can be represented by memristors is positive and negative symmetric. The scope of network weights should also be symmetrical to avoid the waste of the quantization levels. Therefore, constraints are added in the training process to make the network weights evenly distributed between $-W$ and W. The specific value of W can be determined according to the actual model and dataset. In our experiments, it is uniformly set to 1.

3.2 Getting the fine-tuned weights using BNN

In general, we have no idea what the real distribution of the weight in the neural network is. The prior distribution of weight is usually set as a unit Gaussian in the BNN training [33]. The weights obtained through DNN can make the network have good performance, so they can be regarded as appropriate prior knowledge. These weights are used to replace the mean in the Gaussian distribution as

$$p(w) \sim N(0,1) \Rightarrow p(w) \sim N(w_i^*, \sigma_i^2), \tag{10}$$

where w_i^* denotes the weight of the trained DNN and σ_i is the standard deviation that represents the allowable range of weight drift. Such mean value initialization not only makes the model more accurate but also makes the model convergence faster. After setting the mean values, VI and SGD methods can be used to train the network. When the training is complete, the mean values μ_i are taken out of the approximate distribution $q(w|\mu_i, \sigma_i^2)$ as the optimized weights. The standard deviation σ_i of the q distribution represents the actual allowable fluctuation range of the mean value. However, since the conductance drift of the memristor is uncontrolled, these values are omitted.

3.3 Quantization

Before being mapped to the crossbar, the weights need to be quantized to adapt to the conductance levels. Suppose the stable range of conductance values of the memristor is $[G_{\text{start}}, G_{\text{end}}]$. Assuming that the conductance value is divided into N levels, then the interval of conductance between each is $\Delta_G = (G_{\text{start}} - G_{\text{end}})/N$. Since one weight is represented by the difference between two conductance values ($G = G^+ - G^-$), the range of weights that can be represented is $[-N, N]$. Assuming that the weight range before quantization is $[W_{\min}, W_{\max}]$. Let w and w' be the weight values before and after conversion, respectively. The mapping relationship is shown in

$$w' = \text{round}\left(\frac{w}{S} + Z\right), \text{ where } S = \frac{W_{\max} - W_{\min}}{2N}, \ Z = N - \frac{W_{\max}}{S}, \tag{11}$$

and round() denotes the rounding function which returns the nearest integer of the input. When the weight w' is written to the crossbar array, the conductance values could be set as follows:

$$\begin{cases} G^+ = G_{\text{end}} + w' \times \Delta_G, \\ G^- = G_{\text{end}}, \end{cases} \text{if } w' \geqslant 0, \quad \begin{cases} G^+ = G_{\text{end}}, \\ G^- = G_{\text{end}} - w' \times \Delta_G, \end{cases} \text{if } w' < 0. \tag{12}$$

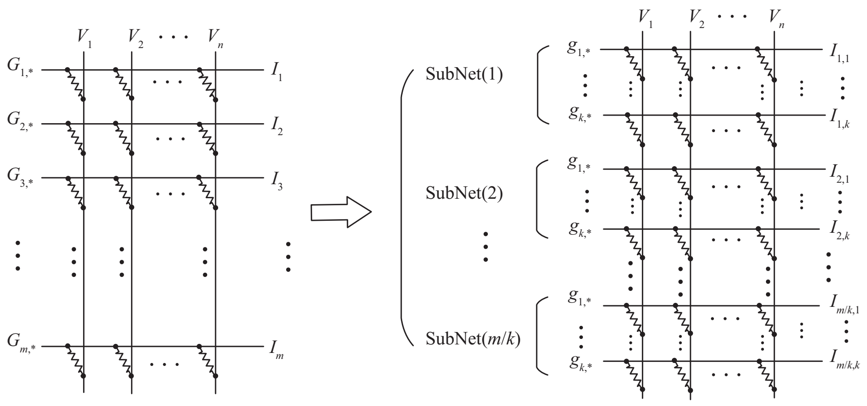
The quantization method used in this paper is ordinary. Although some quantization methods can improve the precision of the quantized network [34,35], these methods can be integrated into any network deployment. To accurately evaluate the effectiveness of our proposed scheme and eliminate the impact of other factors, this quantization method is used in this paper.

3.4 The design of ensemble network

When using BNN for inference, it is usually necessary to conduct ensembled inference through the MC sample. When the optimized weights are written into the crossbar array, these weights will change due to writing error, conductance drift, and other factors. It is equivalent to one sample from the distributions. In this way, the network reliability is improved, but it does not take full advantage of the weight uncertainty of BNN.

Therefore, we propose an ensemble network design approach to further improve the model performance. A large network is replaced by several smaller networks, which are then used to simulate MC sampling. The general ensemble approach needs to train multiple subnetworks, while our approach only needs to train one subnetwork, which significantly reduces the training cost. Then, the weight change caused by conductance drift is utilized to simulate the sampling progress, thus achieving integrated inference. Splitting a big network into several small networks can improve the network reliability without consuming more crossbar array hardware resources and training costs.

For example, if a neural network (A) containing one hidden layer with m hidden units needs to be trained originally, whose input and output dimensions are n and o respectively. A small network (B) with one hidden layer and k hidden units will be trained following the above method. The sizes of the weight matrixes are changed from $n \times m$ and $m \times o$ to $n \times k$ and $k \times o$. Then, the same m/k subnetworks are mapped to the crossbar array respectively, where the change of the network structure from the input layer to the hidden layer is shown in Figure 4, and the changes from the hidden layer to the output layer are similar. Finally, the outputs of these networks are integrated by accumulation. Since the weight



V_1 V_2 V_n
$G_{1,*}$ $G_{2,*}$ $G_{3,*}$ $G_{m,*}$
I_1 I_2 I_3 I_m
SubNet(1)
SubNet(2)
SubNet(m/k)
$g_{1,*}$ $g_{k,*}$
$I_{1,1}$ $I_{1,k}$ $I_{2,1}$ $I_{2,k}$ $I_{m/k,1}$ $I_{m/k,k}$

Figure 4 The schematic diagram designing of ensembled subnetworks.

deviation of each subnetwork will be different with the conductance drift, it is equivalent to sampling m/k times from the distribution. Experiments show that this ensemble approach can greatly improve network reliability.

4 Experiments

MNIST [36] data set is used to verify our proposed scheme. The MNIST training set contains 50000 images, and the test set contains 10000 images, which are consisted of 10 categories of handwritten digits from 0 to 9. In this paper, experimental verification is carried out according to the two types of conductance drift in Eq. (2). Fully-connected neural networks (FNN) and convolutional neural networks (CNN) are used to verify the proposed scheme. For the convenience of description, the optimized network using our proposed scheme is called Bayesian fully-connected neural networks (BFNN) and Bayesian convolutional neural networks (BCNN), respectively. The weights are quantized to the levels from 4 to 10.

4.1 The first type of conductance drift

The first type of conductance drift is that the mean value of conductance distribution does not change, that is, $\Delta w_i \sim N(0, \sigma_1^2)$. In the experiment, the range of σ_1 is set to $[0, 1.0]$. It should be noted that the weight does not change when the value of σ_1 is 0, which means the conductance value of the memristor can accurately represent the quantized weight. Due to the existence of writing errors, this situation is impossible. However, to observe the impact of weight drift on the accuracy, the case where σ_1 is 0 is still drawn. The proposed scheme is first evaluated on the neural networks containing one hidden layer.

Firstly, three FNNs with one hidden layer and 64128 and 256 hidden units were trained, whose input layer units and output layer units were both 784 and 10, and their raw accuracies were 97.51%, 97.91%, and 98.04%, respectively. Based on the three FNNs, three BFNNs were obtained through the proposed scheme, whose accuracies (before quantization) were 96.42%, 96.41%, and 96.5%, respectively. Because the weight changes are random, the accuracy will be different every time, even if the perturbation is limited to the same magnitude. To accurately evaluate the network performance, the weights were randomly perturbed 10-times for each case and the mean value and standard deviation of each set of accuracies were recorded. The experimental results are shown in Figure 5 and Table 1.

In Figure 5, solid lines and dashed lines of the same color are used to represent the accuracies of BFNN and FNN with the same quantitative levels, respectively. It can be seen that quantization reduces the network accuracy slightly, and FNN is more severely affected than BFNN. And even the accuracy of BFNN will, in turn, exceed that of FNN when the quantization level is small. When the number of hidden units is 256, the quantization level is 4, and the standard deviation of weight deviation is 0.1, the accuracy of BFNN improves the most compared with FNN, reaching 35.39%. When the number of hidden units is 128, the quantization level is 10, and the standard deviation of weight deviation is 0.2, the accuracy of BFNN decreases the most, which is 1.43%. In general, BFNNs have better stability, and their accuracy is much better than FNNs when the weights change in the same order of magnitude. At the same time, we found that the greater the number of hidden units, the more sensitive the network

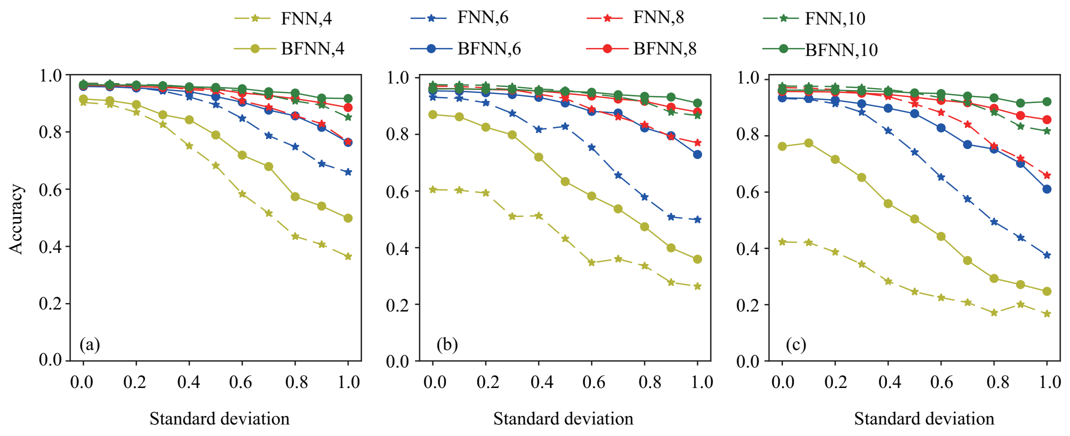


Figure 5 (Color online) The mean accuracies of FNN and BFNN with (a) hidden units: 64, (b) hidden units: 128, and (c) hidden units: 256. The horizontal axis is the standard deviation of the distribution of weight deviation, and the numbers in the legend denote quantization levels.

Table 1 The standard deviations of the accuracies based on the network with 64 hidden units[a)]

Standard deviation	FNN, 4	FNN, 6	FNN, 8	FNN, 10	BFNN, 4	BFNN, 6	BFNN, 8	BFNN, 10
0.1	0.0069	0.0017	**0.0005**	0.0009	**0.0041**	**0.0015**	0.0007	**0.0008**
0.2	0.0165	0.0023	0.0018	0.0016	**0.0097**	**0.0022**	**0.0011**	**0.0012**
0.3	0.0183	0.0124	0.0027	**0.0021**	**0.0146**	**0.0044**	**0.0012**	0.0023
0.4	0.0371	0.0188	0.0047	**0.0028**	**0.0255**	**0.0061**	**0.0011**	0.0032
0.5	0.0366	0.0181	0.0077	0.0083	**0.027**	**0.0101**	**0.0026**	**0.0028**
0.6	0.0594	0.0256	0.0191	0.0112	**0.021**	**0.0095**	**0.006**	**0.0046**
0.7	0.0497	0.0493	0.0135	0.0166	**0.0355**	**0.0246**	**0.0105**	**0.0086**
0.8	0.1136	0.0325	0.042	0.0206	**0.0495**	**0.0288**	**0.0069**	**0.0176**
0.9	0.0385	0.0565	0.0327	0.0187	**0.0385**	**0.0382**	**0.0287**	**0.0143**
1	0.0802	0.0479	0.056	0.029	**0.0262**	**0.035**	**0.0193**	**0.0151**

a) The smaller values between FNN and BFNN are marked in bold. Standard deviation means the standard deviation of the distribution of weight deviation.

is to weight deviation, and bigger networks are more susceptible to weight drift. The bigger networks usually have better accuracy but are more sensitive to conductance drift, so a balance between the two should be considered in practical applications.

Table 1 shows the standard deviations of the accuracies for the network with 64 hidden units. It can be seen that the accuracy fluctuation of BFNN is smaller than that of FNN. The range of the standard deviations of BFNN is $[0.0008, 0.0495]$, while the range of FNN is $[0.0009, 0.1136]$. Besides, for the networks with 128 hidden units which is not listed due to space limitations, the range of BFNN and FNN is $[0.0005, 0.0901]$ and $[0.0006, 0.115]$, respectively. While for the networks with 256 hidden units, the range is $[0.0006, 0.0719]$ and $[0.0005, 0.0906]$, respectively. It indicates that the BFNN obtained by our proposed scheme not only has better accuracy but also has smaller accuracy fluctuation and more stable network performance.

In the experiment of Figure 5, σ_i of prior distribution $p(w)$ in Eq. (10) is set as 0.2, which represents the prior knowledge of the weight deviation range. The choice of the value of σ_i will also have an impact on network performance. Experiments were conducted on FNNs with 128 hidden units. The quantization level is 4, 7, and 10, and the σ value is 0.1, 0.15, 0.2, and 0.25, respectively.

The experimental results are shown in Figure 6. It can be seen that when the network quantization level is small, a larger σ can make the network have more stable performance. While with the increase of the quantization level, a large σ will make the network hard to train and reduce the accuracy, but it still makes the network maintain higher stability when the weights have a large drift. Therefore, the size of σ can be determined based on the quantization level and actual requirements.

Although a single BFNN can improve network reliability, it is equivalent to sampling only once from the weight distribution, so there is still room for improvement. Therefore, an ensemble approach is proposed to further improve the network performance. The FNN with 128 hidden units above is used as the benchmark model, and then the effectiveness of the scheme is evaluated on the ensembled network of

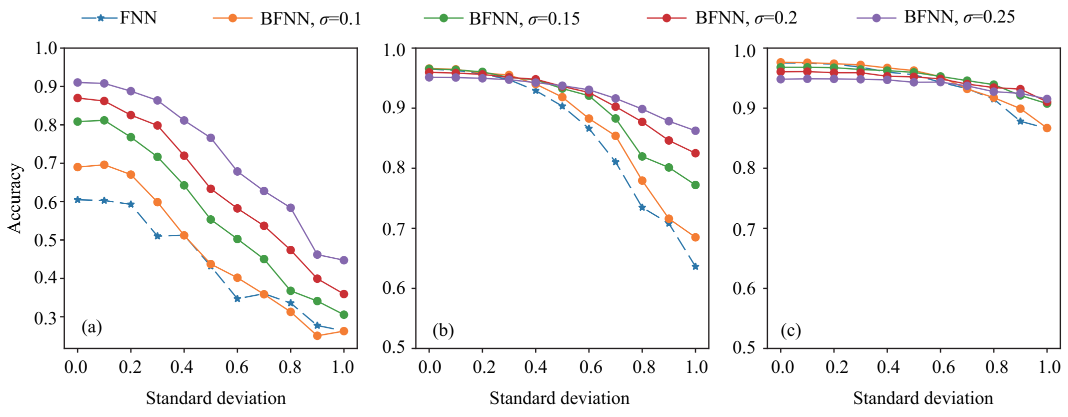


Figure 6 (Color online) Network performance with different σ of prior distribution, (a)–(c) are the experimental result when the quantization level is 4, 7, 10, respectively.

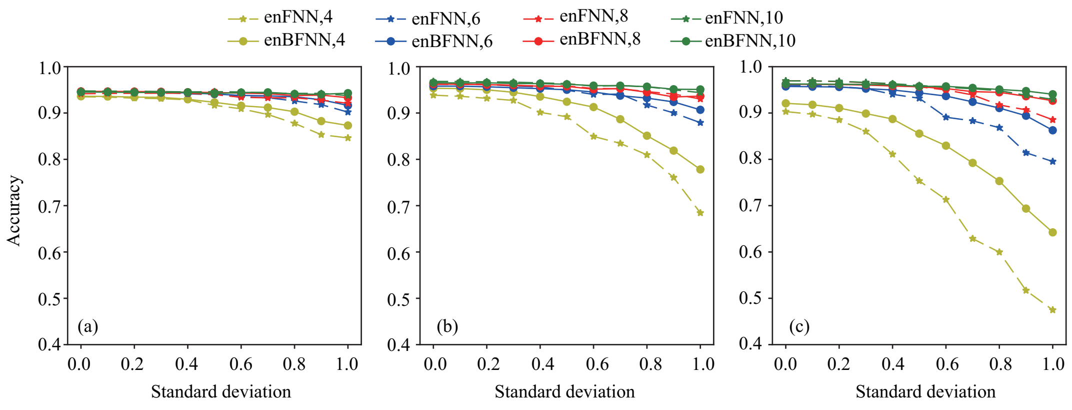


Figure 7 (Color online) The mean accuracies of enFNN and enBFNN with different hidden units. (a) 8×16; (b) 4×32; (c) 2×64. The numbers in the legend denote quantization levels.

8×16, 4×32, 2×64, respectively, where $M \times N$ means that the network is composed of M subnetworks with one hidden layer and N hidden units. Since the accuracy of the ensembled network composed of multiple subnetworks will be higher than that of a single model, to prove our ensemble scheme could indeed improve the network performance, the ensembled network composed of BFNN (called enBFNN) is compared with the ensembled network composed of FNN (called enFNN). The experimental results are shown in Figure 7.

Compared with the single network with 128 hidden units in Figure 5, the accuracy and stability of all ensembled networks are improved. Moreover, compared with enFNN, the performance of enBFNN is further improved. When there are fewer hidden units in the subnetwork, the accuracy of the ensembled network will be slightly affected due to the limited-expression capacity of the single subnetwork. However, as more subnetworks can be integrated into the same size hardware, the constructed enBFNN will have better reliability. When there are more hidden units in the subnetwork, the result is just the opposite. The accuracy of the ensembled network will be improved, but the stability will be decreased.

The effectiveness of the proposed scheme is also evaluated on CNN shown in Figure 8. A CNN which consists of one convolutional layer (the kernel size is 3×3 and the channel number is 64) and one fully connected layer was first trained. Then the proposed scheme was used to optimize the weights of the CNN and obtain the corresponding BCNN. The original accuracies of the CNN and BCNN are 98.48% and 97.47%, respectively. It can be seen that BCNN has better performance than CNN as shown in Figure 8(a). Figure 8(b) shows the test result of the ensembled CNN and BCNN (called enCNN and enBCNN), both of which consist of 4 subnetworks with 16 channels and 3×3 kernels. The ensembled network has higher accuracies than the single network, while consuming crossbar array resources of the same size. Like the ensembled BFNN, enBCNN can also further improve the performance of the network compared with enCNN. These experiments illustrate that our proposed scheme is efficient not only for FNN but also for CNN.

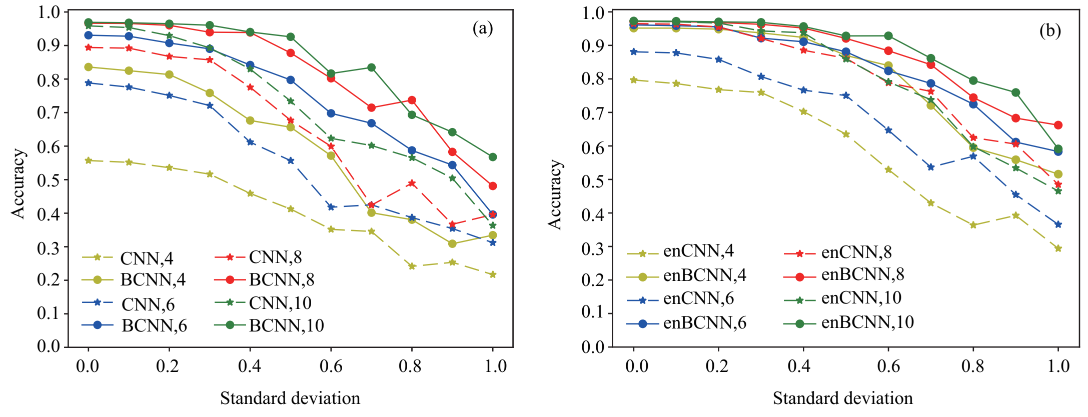


Figure 8 (Color online) The effectiveness of the proposed scheme on CNN. (a) Single CNN and BCNN; (b) ensembled CNN and BCNN.

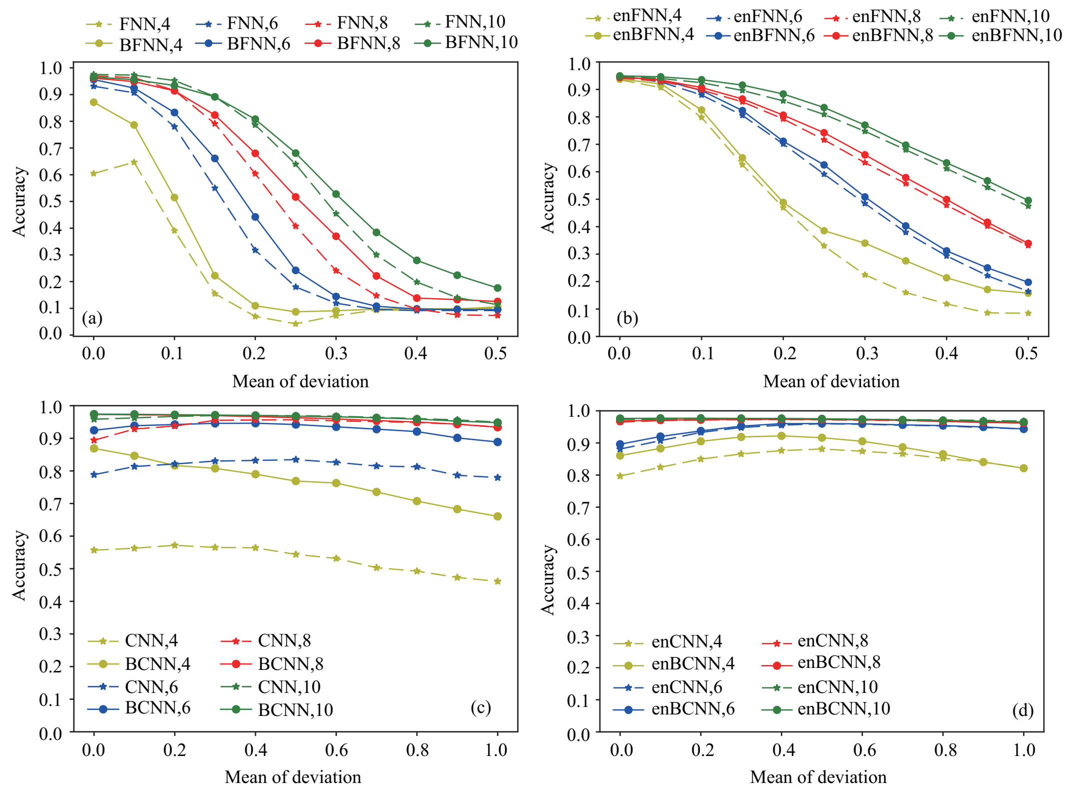


Figure 9 (Color online) The experimental results for the second type conductance drift. (a) Single network (FNN and BFNN); (b) ensembled network (enFNN and enBFNN); (c) single network (CNN and BCNN); (d) ensembled network (enCNN and enBCNN).

4.2 The second type of conductance drift

There is the other tendency of the weight deviation caused by the conductance drift, that is, the overall weight tends to decrease, which corresponds to $\Delta w_2 \sim N(\mu_2, \sigma_2^2)$ in Eq. (2). In the experiments, σ_2 was fixed to 0.1 and increased μ_2 gradually. The proposed scheme was also evaluated on FNN and CNN, respectively. The FNN consists of one hidden layer with 128 hidden units, and the CNN consists of one convolution layer with 64 channels and 3×3 kernels. The enBFNN contains 8 subnetworks with 16 hidden units, and the enBCNN contains 4 subnetworks with 16 channels. The experimental results are shown in Figure 9.

It can be seen that the performance of BFNN and BCNN has been improved in different degrees compared with FNN and CNN, and when the ensemble approach is employed, the performance of the model is further enhanced. These experiments illustrate that our proposed scheme can also improve the model reliability for the second type of conductance drift. At the same time, we found that FNN is more sensitive to the conductance drift of the second type, and its accuracy decreases rapidly with the increase of conductance drift, while CNN is more robust to the decrease of weight. Maybe it is because the overall deviation of the values of convolution kernels in the same direction does not affect its ability of feature extraction. Only when the value of convolution kernel changes in different directions, its feature extraction ability will be significantly reduced, just like the case of the first type of conductance drift. It can be seen that the effect of the proposed scheme on the second type of conductance drift is not as good as that on the first type of conductance drift. It may be because the assumption distribution used in the training of BNN is positive and negative symmetric, and the impact of the second type of conductance drift on the weight is unidirectional. Therefore, the effect will be slightly decreased.

5 Conclusion

In this paper, a novel scheme of weight optimization and network ensemble based on BNN has been proposed to improve network performance. A series of experiments demonstrate the proposed scheme can improve the reliability of a single network for the both types of conductance drift. Moreover, the network performance is further improved through the ensembled network without increasing the training cost or crossbar array resources. Our proposed weight optimization operation does not require additional hardware support and can be easily integrated into the deployment of MNNs, providing a more reliable guarantee for the hardware implementation of neuromorphic computing.

Acknowledgements The work was supported by National Key R&D Program of China (Grant No. 2018YFB1306600), National Natural Science Foundation of China (Grant Nos. U20A20227, 62076207, 62076208, 61976246), Fundamental Science and Advanced Technology Research Foundation of Chongqing, China (Grant No. cstc2017jcyjBX0050), and Natural Science Foundation of Chongqing (Grant No. cstc2020jcyj-msxmX0385).

References

1 Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. In: Proceedings of Advances in Neural Information Processing Systems, Lake Tahoe, 2012. 1097–1105
2 Saon G, Kuo H K J, Rennie S, et al. The IBM 2015 English conversational telephone speech recognition system. Eurasip J Adv Sig Pr, 2015, 2008: 1–15
3 van den Oord A, Dieleman S, Zen H G, et al. WaveNet: a generative model for raw audio. 2016. ArXiv:1609.03499
4 Xia Q F, Yang J J. Memristive crossbar arrays for brain-inspired computing. Nat Mater, 2019, 18: 309–323
5 Wong H, Salahuddin S. Memory leads the way to better computing. Nat Nanotechnol, 2015, 10: 191–194
6 Dong Z K, Lai C S, He Y F, et al. Hybrid dual-complementary metal-oxide-semiconductor/memristor synapse-based neural network with its applications in image super-resolution. IET Circ Devices Syst, 2019, 13: 1241–1248
7 Dong Z K, Qi D L, He Y F, et al. Easily cascaded memristor-CMOS hybrid circuit for high-efficiency Boolean logic implementation. Int J Bifurcat Chaos, 2018, 28: 1850149
8 Dong Z K, Lai C S, Qi D L, et al. A general memristor-based pulse coupled neural network with variable linking coefficient for multi-focus image fusion. Neurocomputing, 2018, 308: 172–183
9 Yu S M, Li Z W, Chen P Y, et al. Binary neural network with 16 Mb RRAM macro chip for classification and online training. In: Proceedings of IEEE International Electron Devices Meeting (IEDM), San Francisco, 2016
10 Gokmen T, Vlasov Y. Acceleration of deep neural network training with resistive cross-point devices: design considerations. Front Neurosci, 2016, 10: 333
11 Yan B N, Chen Y R, Li H. Challenges of memristor based neuromorphic computing system. Sci China Inf Sci, 2018, 61: 060425
12 Yan B N, Yang J J, Wu Q, et al. A closed-loop design to enhance weight stability of memristor based neural network chips. In: Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, 2017. 541–548
13 Yao P, Wu H Q, Gao B, et al. Fully hardware-implemented memristor convolutional neural network. Nature, 2020, 577: 641–646
14 Kim W, Bruce R L, Masuda T, et al. Confined PCM-based analog synaptic devices offering low resistance-drift and 1000 programmable states for deep learning. In: Proceedings of IEEE Symposium on VLSI Technology, Kyoto, 2019. 66–67
15 Zhang Q T, Wu H Q, Yao P, et al. Sign backpropagation: an on-chip learning algorithm for analog RRAM neuromorphic computing systems. Neural Netw, 2018, 108: 217–223
16 Ambrogio S, Kumar A, Chen A, et al. Reducing the impact of phase-change memory conductance drift on the inference of large-scale hardware neural networks. In: Proceedings of IEEE International Electron Devices Meeting (IEDM), San Francisco, 2019
17 Mackay D J C. Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks. Netw-Comput Neural Syst, 1995, 6: 469–505
18 Wang H, Yeung D Y. Towards Bayesian deep learning: a survey. 2016. ArXiv:1604.01662
19 Duan S K, Hu X F, Wang L D, et al. Memristor-based RRAM with applications. Sci China Inf Sci, 2012, 55: 1446–1460
20 Hu X F, Feng G, Duan S K, et al. A memristive multilayer cellular neural network with applications to image processing. IEEE Trans Neural Netw Learn Syst, 2017, 28: 1889–1901
21 Alibart F, Zamanidoost E, Strukov D B. Pattern classification by memristive crossbar circuits using ex situ and in situ training. Nat Commun, 2013, 4: 2072
22 Wong H S P, Lee H Y, Yu S, et al. Metal-oxide RRAM. Proc IEEE, 2012, 100: 1951–1970
23 Yao P, Wu H Q, Gao B, et al. Face classification using electronic synapses. Nat Commun, 2017, 8: 15199

24 Sebastian A, Gallo M L, Burr G W, et al. Tutorial: brain-inspired computing using phase-change memory devices. J Appl Phys, 2018, 124: 111101

25 Gallo M L, Sebastian A, Cherubini G, et al. Compressed sensing recovery using computational memory. In: Proceedings of IEEE International Electron Devices Meeting (IEDM), San Francisco, 2017

26 Nandakumar S R, Gallo M L, Boybat I, et al. A phase-change memory model for neuromorphic computing. J Appl Phys, 2018, 124: 152135

27 Lampinen J, Vehtari A. Bayesian approach for neural networks-review and case studies. Neural Netw, 2001, 14: 257–274

28 Neal R M. Bayesian Learning for Neural Networks. Berlin: Springer, 1996

29 Hinton G E, Camp D V. Keeping the neural networks simple by minimizing the description length of the weights. In: Proceedings of the 6th Annual Conference on Computational Learning Theory, New York, 1993. 5–13

30 Wainwright M J, Jordan M I. Graphical models, exponential families, and variational inference. FNT Mach Learn, 2007, 1: 1–305

31 Blei D M, Kucukelbir A, McAuliffe J D. Variational inference: a review for statisticians. J Am Stat Assoc, 2017, 112: 859–877

32 Barber D, Bishop C M. Ensemble learning in bayesian neural networks. Nato Asi Series F Comput Syst Sci, 1998, 168: 215–237

33 Blundell C, Cornebise J, Kavukcuoglu K, et al. Weight uncertainty in neural networks. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning, Lille, 2015. 1613–1622

34 Song C, Liu B Y, Wen W, et al. A quantization-aware regularized learning method in multilevel memristor-based neuromorphic computing system. In: Proceedings of Non-volatile Memory Systems & Applications Symposium, Hsinchu, 2017

35 Hu X F, Shi W Q, Zhou Y, et al. Quantized and adaptive memristor based CNN for image processing. Sci China Inf Sci, 2021. doi: 10.1007/s11432-020-3031-9

36 Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proc IEEE, 1998, 86: 2278–2324