Cycle-to-cycle Variation Enabled Energy Efficient Privacy Preserving Technology in ANN

Jingyan Fu
Electrical and Computer Engineering Department
North Dakota State University
Fargo, USA
jingyan.fu@ndsu.edu

Zhiheng Liao
Electrical and Computer Engineering Department
North Dakota State University
Fargo, USA
zhiheng.liao@ndsu.edu

Jinhui Wang
Electrical and Computer Engineering Department
University of South Alabama
Mobile, USA
jwang@southalabama.edu

2020 IEEE 33rd International System-on-Chip Conference (SOCC) | 978-1-7281-8746-4/20/$31.00 ©2020 IEEE | DOI: 10.1109/SOCC49529.2020.9524794

***Abstract*—Differential privacy is emerging as an effective solution to achieve privacy protection for the Artificial Intelligence neural network (ANN). However, not only matrix calculations of a neural network but also random noise injection mechanisms for differential privacy consume large power and resources. Traditionally, most privacy protection technologies are software technologies using von Neumann architecture and hardware with extra noise generation circuit unit. In this paper, a memristor based crossbar in-memory computing system is proposed to enable energy efficient privacy preserving technology in ANN. We utilize inherent cycle-to-cycle variations of memristors and apply the proposed variation-based pulse pair method during the weight update process. As a result, the proposed methods realize a machine learning system with privacy protection and show up to 29.24% recognition accuracy improvement with various privacy budget ε.**

Keywords—differential privacy, memristor based noise injection, neural network

I. INTRODUCTION

Artificial intelligence involves many sensitive data, such as private, corporate, and national privacy information, which urgently needs effective privacy protection technologies. Differential privacy [1] is a popular solution that can provide a quantifiable indicator for privacy protection. From the perspective of differential privacy, machine learning algorithms could be designed to perform privacy preserving learning by introducing random noise [2]. However, the great problem is existing: 1) Machine learning training involves massive calculation, which introduces a large workload for hardware, especially for some real time applications such as online learning. 2) When learning systems involve privacy protection, noise injection is needed in each training stage [3, 4]. Such a high-cost training application further challenges hardware technology as well as impedes the development of privacy protection in ANN.

To decrease the cost, many algorithm level solutions are proposed, for example, binary neural networks (BNN) [5] and neural network compression [6]. In addition, various hardware accelerators, such as Field Programmable Gate Arrays (FPGA), Application Specific Integrated Circuit (ASIC), Graphics Processing Units (GPUs), and Tensor Processing Unit (TPU), are developed. Nevertheless, the memory wall always exists [7] because of traditional Von Neumann architecture. Thus, emerging notions such as resistive computing, quantum computing, molecular computing, neuromorphic computing, memristor devices, quantum dots, and spin-wave devices, are explored. Among those technologies, memristor-based in-memory processing architecture is a promising candidate because memristors have desirable metrics and CMOS process compatibility [8]. The notion of memristor was predicted dozens of years ago and its physical realization was demonstrated by Hewlett-Packard Lab in 2008 [9], [10]. A memristor has a simple three-layer structure whose conductance value can be changed with the applied pulse, which can achieve multiple conductance states, small scale size (less than 2 nm), high switching speed (less than 1 ns), and low programming power consumption [7]. Because of these metrics, memristor-based crossbar architecture achieves fast dot-product operations [11], which have been applied to fabricate neural network circuits such as compression/filtering [12] and image classification [13], [14].

As for privacy protection in ANN, in [15], low-voltage static random-access memory (SRAM) chips are used to add failure as noise for training data, but the noise only follows a uniform distribution and does not satisfy the differential privacy theory for Gaussian and Laplace distribution. In [16], [17], in order to generate random numbers with high randomness, dedicated random number generation modules, such as physical unclonable function (PUF) and random number generator, are designed. These modules are accurate but require significant additional circuitry. Also, all the above work is based on CMOS technology, and not compatible with memristor-based learning systems. In [18], an advanced learning system is implemented based on memristor arrays, but noise in training data for theoretical privacy guarantees use software methods to seriously complicate calculation.

What is more, as for memristors, some researchers point out the existence of non-ideal properties that includes non-linearity, device-to-device variation, cycle-to-cycle variation, maximum conductance variation, and minimum conductance variation [18], [19], [20], [21]. These non-ideal properties degrade the accuracy of a memristor-based learning system, however, such variations can be also considered as inherent resources for noise

generation that is necessary for differentially private learning systems.

In this paper, we take the cycle-to-cycle variation as an advantage to realize hardware-based Gaussian noise injection. As a consequence, this paper explores differentially private learning systems and proposes a hardware-based solution by utilizing the non-ideal properties of memristors. Also, optimization methods are proposed to improve the utility of neural networks. The proposed methods add Gaussian noise distribution to a system without adding computational complexity and introducing extra hardware, which greatly improves the power and computation efficiency.

II. Preliminaries

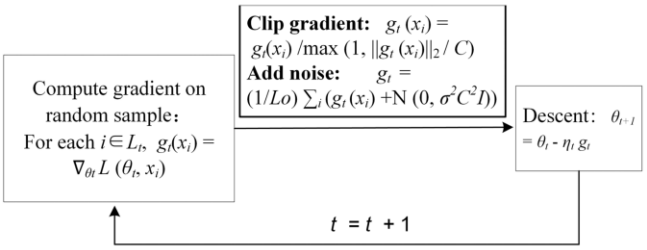
A. Differential Private Machine Learning Algorithm

Differential privacy protection technology is recognized as a rigorous and robust protection model. The basic idea of the protection model is to achieve the privacy protection effect by adding noise mechanism on the original data, the conversion of the original data, or the statistical results. This protection method ensures that inserting or deleting a record in a dataset does not affect any calculated output. Differential privacy pro-vides provable guarantees of privacy, mitigating the risk of exposing sensitive training data in machine learning [2]. The definition of ε-differential privacy and equation is given below [1], [2]. A randomized mechanism A satisfies ε-differential privacy when any adjacent input d and d', and any output S of A hold that

$$Pr\,[A(d) = S] \leq e^{\varepsilon} \cdot Pr\,[A(d') = S] \quad (1)$$

where d and d' represent neighboring (adjacent) datasets. In our study, each training dataset is a set of image-label pairs. The parameter ε is the privacy budget, which evaluates the privacy guarantee of the randomized mechanism A, thereby privacy preservation can be calculated and evaluated through ε. DP-SGD (differential private Stochastic Gradient Descent) is a popular algorithm that obtains provable privacy guarantees for machine learning algorithm [22].

At each step of the DP-SGD, it computes the gradient for a random subset of examples, clips each gradient, computes the average, adds noise in order to protect privacy, and takes a step in the opposite direction of this average noisy gradient. Two operations are needed to ensure that stochastic gradient descent is a differentially private algorithm. The first is to clip gradient computed on each training image that is used to limit how much each training image can impact model parameters. The algorithm clips each gradient by a clipping threshold C. In this paper, we use *L2 norm of gt(xi)* to represent $||gt(xi)||_2$, which is shown in Fig. 1 and is explained in detail in [22]. The second is to sample and add random noise that is used to randomize the algorithm's behavior. Thus, it is statistically impossible to identify whether a particular sample is included in the training set.



Symbols and parameters: input dataset, weights θ, loss function $L(\theta)$, gradient g, learning rate η_t, noise scale σ, group size Lo, gradient norm bound C, total weight update step T, the square root of the largest eigenvalue of the matrix $g_t(x_i)^* g_t(x_i)$, where gt(xi)* denotes the conjugate transpose of $g_t(x_i)$.

Fig. 1. Outline of DP-SGD [22].

B. Memristors and Cycle-to-cycle Variation

Memristors in crossbar array structure can carry out matrix multiplication operations in parallel within the analog domain, which can enable learning systems with high throughput at low energy and cost consumption. On-chip learning with a memristor crossbar array empowers the learning system with online learning ability [23]. A neural network is inspired by the biological systems that transform inputs to desired outputs by feed-forward actions. As shown in Fig. 2, in the hardware implementation, the neural network can be directly mapped into a crossbar structure where the multiplication of vector and matrix can be conducted by applying input voltages to each row and reading currents from each column. Memristors lo-cate in cross points. Their desirable properties support the memristor-based crossbar circuit to be a promising substitute technology to traditional ones. Also, many peripheral circuits are required for the complete functionality of the on-chip learning system. Write and read circuits are used to program the memristor cell and read the output, respectively. Digital-to-analog converter (DAC) and analog-to-digital converter (ADC) transfer signal between the analog domain and the digital do-main. Additional circuits are required for the activation function and the timing controller.

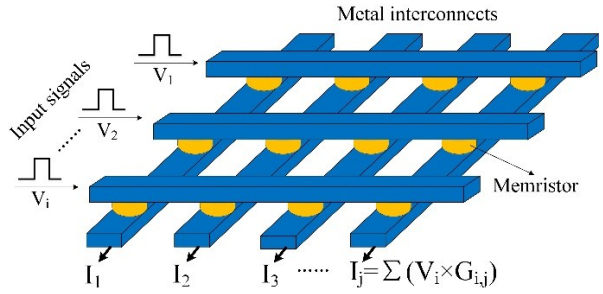


Fig. 2. Hardware implementation of neural networks using memristor crossbar. V_i, $G_{i,j}$, and I_j represent the input signal in the i_{th} row, the conductance of the memristor in the j_{th} column and ith row, and the output current that represent the dot product result of V and G, respectively.

Memristors can achieve multiple conductance states. In our learning system, the conductance of each memristor represents the weight of each synapse. As shown in Fig. 3, positive and negative input voltage pulses that are larger than threshold voltages can switch a memristor gradually from G_{min} to G_{max} or from G_{max} to G_{min}, where G_{min} and Gmax represent minimum conductance and maximum conductance, respectively. Thus conductance/weight increase processes are called long-term

potentiation (LTP) and the conductance/weight decrease processes are called long-term depression (LTD) [24].

In the backpropagation phase of the DP-SGD algorithm, the weight update values (Δw) will be translated to the number of LTP or LTD pulses and applied to the synaptic array. The amount of conductance change should be linearly proportional to the number of write pulses, however, this linear change is broken by variations of memristors. Among all variations of memristors, instead of caused by manufacturing process variation, the cycle-to-cycle variation is caused by intrinsically stochastic resistance switching mechanisms that can be approximated as Gaussian [25], [26], [27], [28], [29], [30].

In each weight update step for each memristor, the cycle-to-cycle variation that subjects to standard normal distribution N (0, σ) and the conductance variation (*Gvar*) that is caused by cycle-to-cycle variation after memristor's *Num_P* pulses applied is illustrated as [17], [30],

$$Gvar = N(0, \sigma) \times (Num\_P)^{1/2} \quad (2)$$

where *Num_P* represents the applied pulse number [31], [32].

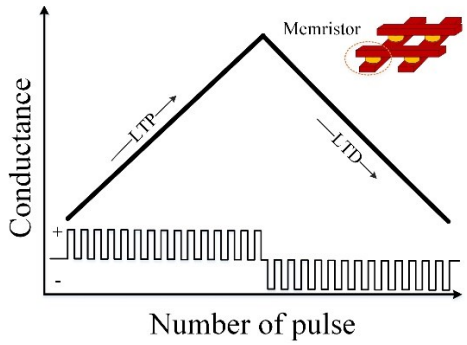


Fig. 3. Long-term potentiation process (LTP) and long-term decrease process (LTD).

In this paper, we make use of cycle-to-cycle variation and achieve the algorithm behavior of DP-SGD based on a memristor-based neural network, which is discussed in Section III. Thus, we realize a privacy-preserving memristor-based learning system without introducing extra computing processing and noise generation units.

III. Methodology and Optimization

A. Positive and Negative Pulse Pair (PN) Method

A memristor-based methodology to complete random noise injection of privacy preserving neural network consists of memristor crossbar structure and peripheral circuits. In a memristor-based neural network, each weight can be represented by a memristor whose conductance changes with input pulses. A privacy preserving neural network needs introducing random Gaussian noise to each memristor. We propose a Positive and Negative Pulse Pair (PN) method that applies positive and negative pulses pair to each memristor. As shown in Fig. 3, positive pulse and negative pulse make conductance increase and decrease, respectively. Cycle-to-cycle variation exists when one or more pairs of pulses applied, which brings Gaussian noise to a target memristor. When the amount of change in conductance caused by such positive and negative pulses counteract each other, it is equivalent to adding a Gaussian noise, which is illustrated by Equations. (2), (3), and (4). First, we apply *i* positive pulses, the conductance of the memristor (*G*) changes from G_0 to G_1,

$$G_1 = G_0 + ((Gmax-Gmin)/N_{Level}) \times i + Gvar \quad (3)$$

After applying *j* negative pulses, it changes from G_1 to G_2,

$$G_2 = G_1 - ((Gmax-Gmin)/N_{Level}) \times j + Gvar \quad (4)$$

If i=j, from (2), (3), and (4), we get

$$G_2 = G_0 + N(0, \sigma) \times (2i)^{1/2} \quad (5)$$

Thus, Gaussian noise is added to the target memristor without introducing extra circuits, only using one or more pairs of positive and negative pulses. The scale of injected noise is decided by the number of the added PN pulse pair and the pulse width of the added PN pulse pair. In our simulations in Section IV, the pulse-width of the PN method is equal to the pulse-width of normal weight updating. Each time, the injected noise of a memristor is

$$Gvar = N(0, \sigma) \times (n + 2m)^{1/2} \quad (6)$$

where σ represents the noise scale of cycle-to-cycle variation introduced by each input pulse for a memristor. *n* represents the number of pulses that are used to update weight, and *m* reflect the noise requirement of privacy protection of the PN method.

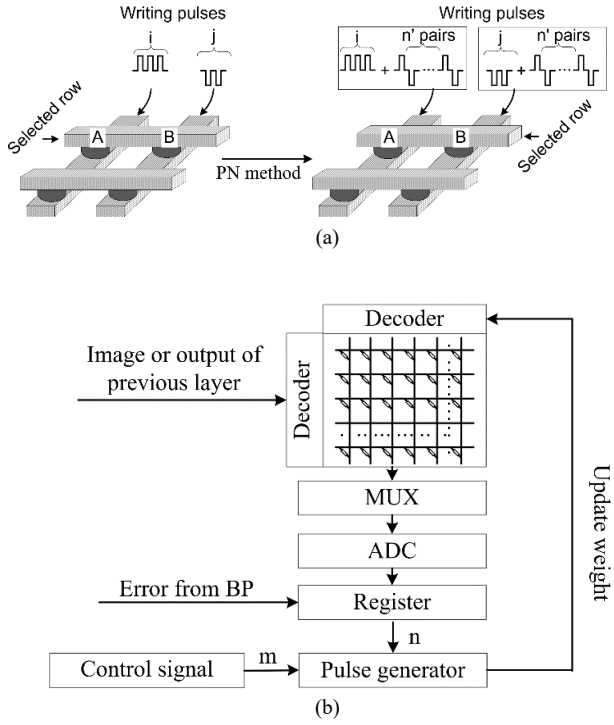


Fig. 4. Workflow of the PN method, where BP represents backpropagation of SGD, n represents the number of pulses that are used to update weight and m represents the number of positive and negative pulse pairs. Example: After applying PN method, for memristor A n=i, m=n', and for memristor B, n=j, m=n'.

B. Implementation of DP-SGD by PN Method

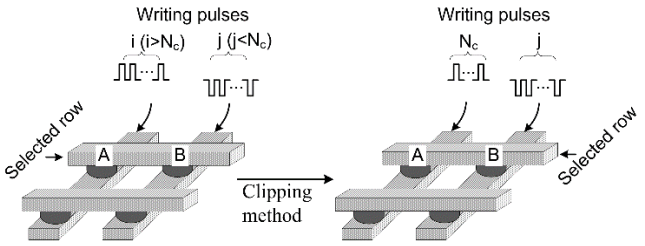
As illustrated in Fig. 1, two modifications (clip gradient and add noise) are needed to ensure that stochastic gradient descent (SGD) is a differentially private algorithm. For the first modification, in order to constrain how much each training sample can influence the resulting gradient computation (model parameters), the sensitivity of each gradient needs to be bounded. For the second modification, it is necessary to randomize the behavior of the algorithm to make it statistically impossible to identify whether a particular training sample is included in the training dataset, which can be achieved by adding random noise to the clipped gradients.

The PN method achieves adding random noise by extra input pulses to memristors. For hardware implementation, the selected cells to be written will be on the same row, and programming pulses or biases are provided from columns, allowing the selected cells to be tuned differently in parallel. Since the weight increase and decrease need different programming voltage polarities, the weight update process (writing process for the model parameter) requires two steps with positive and negative voltages, respectively. In each step, extra pairs of input pulses are added to each memristor, thereby random noises are added to the conductance of memristors. Fig. 4 shows the hardware implementation flow of the PN method, which we adopt in Section IV.

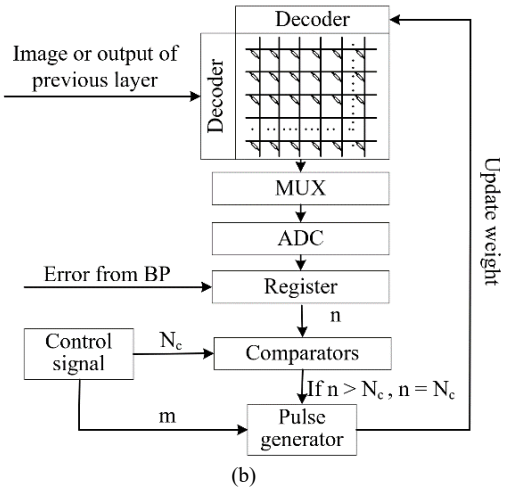
C. Clipping Method and Optimization Strategies

By the PN method, the noise added of DP-SGD can be implemented by hardware solution, however, Fig. 1 shows that the gradient clipping process needs to calculate the L2 norm of the gradient matrix. These processes inevitably increase the computational load of the system. Therefore, we propose a clipping method that uses a simple hardware unit to meet the requirement of clipping without coping with matrix calculation, as shown in Fig. 5.

This method uses comparators to compare gradient value with a reference gradient value and makes sure the L2 norm values of the gradient are all less than 1. When the gradient is clipped, the maximum number of weight update pulses is fixed. Accordingly, the system saves the cost to make matrix calculation and ensures that how much each training sample can influence model parameters is bounded.



(a)



(b)

Fig. 5. Workflow of the Clipping method, where BP represents backpropagation of SGD, n represents the number of pulses that are used to update weight and m represents the number of positive and negative pulse pairs. Example: After applying Clipping method, for memristor A n= Nc, m=0 and for memristor B, n=j, m=0.

Due to the cycle-to-cycle variation is the inherent characteristics of the memristor, even in a routine weights update process, parameters in the model of a memristor-based hardware system suffer from random noise addition. Accordingly, we propose an Equivalent Substitution (ES) strategy that considers random noise from the routine weights update process as random noise injection. In this way, the amount of pulses pair of PN method for privacy protection can be reduced because certain random noise is already introduced by the routine weights update. The Equivalent Substitution (ES) method decreases the amount of noise added by the PN method so that the accuracy of the differentially private learning systems can be improved. That is, we take advantage of the inherent variation of memristors to equivalent substitute partial noise injected by privacy-preserving mechanism, accordingly, greatly reducing the negative effect of the cycle-to-cycle variation and making the cycle-to-cycle variation in weight update process to be a part of a privacy-preserving mechanism.

IV. Case Study and Analysis

Simulations are conducted to verify our proposed methods including PN method, Clipping method as well as combinational methods of them. To verify the effectiveness of proposed methods and to illustrate the performance of each method, a comprehensive suite of simulations has been conducted. We adopt the neural network hardware platform, NeuroSim+, to train privacy-preserving multi-layer neural networks with the DP-SGD algorithm [28]. This simulator is a circuit model of neuro-inspired architectures to emulate the circuit behavior of an online leaning recognition scenario with Modified National Institute of Standards and Technology (MNIST) [44] dataset based on memristors. The neural network topology includes 400 neurons as an input layer, 100 neurons as a hidden layer, and 10 neurons as an output layer. The simulator emulates the hardware to train the network with images randomly chosen from the training dataset MNIST that includes 60,000 images and classify the testing dataset with 10,000 images. The training process has

two parts, the feed-forward propagation and the backpropagation, which includes weighted sum operation, neuron activation operation, recognition, and deviation calculation. The deviations are used to update the conductance of memristors using identical positive input pulses or identical negative input pulses. We integrate our proposed methods into this simulator, where memristors are set to have one hundred states.

A. Simulation Results

Since lots of types of memristors exist, the proposed methods are explored with various configurations include three levels of cycle-to-cycle variation and three levels of m (in Equation. (6)) that are explained, which is shown in Table 1.

TABLE 1
PARAMETERS

Level	1	2	3
σ/(Gmax-Gmin)	0.1%	0.2%	0.3%
2m	0.2	0.4	0.6

Table 2 shows the recognition accuracy of MNIST handwriting digits under various variation levels and noise levels through the memristor-based fully connected neural network. It shows that for a different combination of σ and m, the method that combines PN, Clipping, and ES always shows higher accuracy as compared to other methods. As a result, the proposed optimization methods are quite effective to increase the recognition accuracy of the neural network under the same noise injection (privacy protection) level.

TABLE 2
RECOGNITION ACCURACY UNDER VARIOUS VARIATION AND NOISE

m/σ level	PN	Combination	Accuracy improved	Privacy budget
1/1	86.77%	92.07%	5.30%	53.13
1/2	70.40%	90.56%	20.16%	26.57
1/3	58.30%	67.98%	9.68%	17.71
2/1	59.19%	86.16%	26.97%	26.56
2/2	52.21%	63.58%	11.37%	13.28
2/3	49.61%	60.44%	10.83%	8.85
3/1	56.52%	85.76%	29.24%	17.71
3/2	47.88%	58.19%	10.31%	8.85
3/3	40.81%	55.52%	14.71%	5.90
Average	57.97%	73.36%	15.40%	N/A

In this paper, the σ_{dp} of noise injection is obtained by

$$\sigma_{dp} = \sigma * 2m / learning\_rate \quad (7)$$

where σ_{dp}, $2m$, and learning_rate represent the σ of noise for differential privacy protection that is illustrated in Fig. 1, the parameter of noise requirement that is illustrated in Equation. (6), and the learning rate of DP-SGD algorithm, respectively.

Accordingly, for optimization methods based on parameters in Table 1, the results in Table 2 indicate 5.3% to 29.24% recognition accuracy improvement when the privacy budget ε ranges from 5.9 to 53.13. The average accuracy is increased by 15.40%.

B. Comparison with Existing Work

Instead of implementing the DP-SGD algorithm for privacy preservation by a traditional computing system, the proposed PN method is simple and feasible to add random Gaussian noise by memristor-based hardware. As compared with the state-of-art [15], [17], [18], the PN method realizes hardware-based privacy protection by using cycle-to-cycle variation. In this way, the memristor-based machine learning system does not need an extra random noise generator unit. The scale of injected noise is adjustable by changing the number of the PN pulse pair or the PN pulse's duration. The Clipping method limits the impact of each training data on model parameters, which saves the cost of the L2 norm of the matrix's calculation. The PN method is a universal method that works for all memristor-based hardware that needs noise injection. Accordingly, the PN method is an effective technique to achieve a hardware-based privacy protection system.

V. CONCLUSIONS

In this paper, the PN (Positive and Negative Pulse Pair method) is proposed to realize the hardware-based noise injection, thereby realizing privacy protection with hardware implementation. Instead of adopting the traditional algorithm-based technology, the PN method focuses on hardware implementation to enable the differentially private learning systems. We add extra random noise to memristor's conductance/weight by positive pulses and negative pulses together, where the impact of LTP and LTD process can counteract each other. Proposed methods avoid complex peripheral circuits. Models with various noise circumstances using the PN method are established to investigate the effectiveness of the proposed methods. With the method that combines PN, Clipping, and ES, simulation results indicate 5.3% to 29.24% average recognition accuracy improvement when the privacy budget ε ranges from 5.9 to 53.13. In addition, the proposed methods can be adapted to many other memristor-based hardware systems. Consequently, the PN method is proved to be an effective technique that can provide hardware solutions of the differentially private learning systems and to prevent accuracy loss that is caused by privacy-preserving noise injection.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under Grant 1855646 and Grant 1953544.

REFERENCES

[1] C. Dwork, "Differential privacy," *Encyclopedia of Cryptography and Security*, pp. 338-340, 2011.

[2] C. Dwork, and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211-407, 2014.

[3] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys & Tutorials*, 2019.

[4] R. Shokri, and V. Shmatikov, "Privacy-preserving deep learning," In Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp. 1310-1321, 2015.

[5] M. Rastegari, V. Ordonez, J. Redmon et al., "Xnor-net: Imagenet classification using binary convolutional neural networks," *In*

Proceedings of European Conference on Computer Vision, pp. 525-542, 2016.

[6] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint*, arXiv:1510.00149, 2015.

[7] M. A. Zidan, J. P. Strachan, and W. D. Lu, "The future of electronics based on memristive systems," *Nature Electronics*, vol. 1, no. 1, pp. 22, 2018.

[8] F. Cai, J. M. Correll, S. H. Lee et al., "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations," *Nature Electronics*, vol. 2, no. 7, pp. 290-299, 2019.

[9] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507-519, 1971.

[10] D. B. Strukov, G. S. Snider, D. R. Stewart et al., "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80, 2008.

[11] Z. Wang, C. Li, P. Lin et al., "In situ training of feed-forward and recurrent convolutional memristor networks," *Nature Machine Intelligence*, vol. 1, no. 9, pp. 434-442, 2019.

[12] C. Li, M. Hu, Y. Li et al., "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, vol. 1, no. 1, pp. 52, 2018.

[13] C. Li, D. Belkin, Y. Li et al., "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," *Nature communications*, vol. 9, no. 1, pp. 2385, 2018.

[14] M. Hu, C. E. Graves, C. Li et al., "Memristor‐based analog computation and neural network classification with a dot product engine," *Advanced Materials*, vol. 30, no. 9, pp. 1705914, 2018.

[15] L. Yang, and B. Murmann, "Approximate SRAM for energy-efficient, privacy-preserving convolutional neural networks," *In Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pp. 689-694, 2017.

[16] H. Jiang, D. Belkin, S. E. Savel'ev et al., "A novel true random number generator based on a stochastic diffusive memristor," *Nature communications*, vol. 8, no. 1, pp. 882, 2017.

[17] M. Uddin, M. S. Hasan, and G. S. Rose, "On the Theoretical Analysis of Memristor based True Random Number Generator," *In Proceedings of the 2019 on Great Lakes Symposium on VLSI*, pp. 21-26, 2019.

[18] J. Fu, Z. Liao, and J. Wang, "Memristor-Based Neuromorphic Hardware Improvement for Privacy-Preserving ANN," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 27, no. 12, pp. 2745-2754, 2019.

[19] J. Fu, Z. Liao, N. Gong et al., "Linear Optimization for Memristive Device in Neuromorphic Hardware," *In Proceedings of the 2019 IEEE Computer Society Annual Symposium on VLSI*, pp. 453-458, 2019.

[20] P.-Y. Chen, B. Lin, I. Wang et al., "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning," *In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 194-199, 2015.

[21] J. Fu, Z. Liao, N. Gong et al., "Mitigating Nonlinear Effect of Memristive Synaptic Device for Neuromorphic Computing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 377 – 387, 2019.

[22] M. Abadi, A. Chu, I. Goodfellow et al., "Deep learning with differential privacy," *In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308-318, 2016.

[23] R. Hasan, T. M. Taha, and C. Yakopcic, "On-chip training of memristor crossbar based multi-layer neural networks," *Microelectronics Journal*, vol. 66, pp. 31-40, 2017.

[24] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures," *In Proceedings of the IEEE International Electron Devices Meeting*, pp. 6.1. 1-6.1. 4, 2017.

[25] F. Alibart, L. Gao, B. D. Hoskins et al., "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, pp. 075201, 2012.

[26] S. Choi, P. Sheridan, and W. D. Lu, "Data clustering using memristor networks," *Scientific reports*, vol. 5, pp. 10492, 2015.

[27] L. Goux, A. Fantini, R. Degraeve et al., "Understanding of the intrinsic characteristics and memory trade-offs of sub-μ A filamentary RRAM operation," *In Proceedings of the Symposium on VLSI Technology*, pp. T162-T163, 2013.

[28] C. Baeumer, R. Valenta, C. Schmitz et al., "Subfilamentary networks cause cycle-to-cycle variability in memristive devices," *ACS nano*, vol. 11, no. 7, pp. 6921-6929, 2017.

[29] A. Fantini, L. Goux, R. Degraeve et al., "Intrinsic switching variability in HfO 2 RRAM," *In Proceedings of the IEEE International Memory Workshop*, pp. 30-33, 2013

[30] J.-H. Lee, D.-H. Lim, H. Jeong et al., "Exploring Cycle-to-Cycle and Device-to-Device Variation Tolerance in MLC Storage-Based Neural Network Training," *IEEE Transactions on Electron Devices*, vol. 66, no. 5, pp. 2172-2178, 2019.

[31] A. Cedilnik, K. Kosmelj, and A. Blejec, "The distribution of the ratio of jointly normal variables," *Metodoloski zvezki*, vol. 1, no. 1, pp. 99, 2004.

[32] D. S. Lemons, and P. Langevin, An introduction to stochastic processes in physics: JHU Press, 2002.