Memristor-Based Variation-Enabled Differentially Private Learning Systems for Edge Computing in IoT

Jingyan Fu, *Student Member, IEEE*, Zhiheng Liao, Jianqing Liu, *Member, IEEE*, Scott C. Smith, *Senior Member, IEEE*, and Jinhui Wang, *Senior Member, IEEE*

***Abstract*—Edge artificial intelligence (AI) achieves real-time local data analysis for IoT systems, enabling low-power and high-speed operation, but comes with privacy-preserving requirements. The memristor-based computing system is a promising solution for edge AI, but it needs a low-cost privacy protection mechanism due to limited resources. In this article, we propose a noise distribution normalization (NDN) method to add Gaussian distributed noise through hardware implementation, thereby achieving differential privacy in edge AI. Instead of using traditional algorithmic noise-insertion methods, we take advantage of inherent cycle-to-cycle variations of memristors during the weight-update process as the noise source, which does not incur extra software or hardware overhead. In one case study, the proposed method realizes ultralow-cost differentially private stochastic gradient descent (DP-SGD) for edge AI in IoT systems, achieving a 3.5%–15.5% average recognition accuracy improvement under different noise levels, as compared with a baseline mechanism.**

***Index Terms*—Cycle-to-cycle variation, differential privacy (DP), learning systems, memristor, neural network, noise injection.**

I. Introduction

THE Internet of Things (IoT) facilitates novel products and services by involving billions of edge devices that are connected to the Internet. With data increasingly generated by IoT devices, edge computing provides an efficient solution by processing data locally to significantly reduce the computing load of the entire system, and save time and energy cost of data communication, as compared to cloud processing.

Manuscript received May 15, 2020; revised August 2, 2020; accepted August 27, 2020. Date of publication September 11, 2020; date of current version June 7, 2021. This work was supported in part by the National Science Foundation under Grant 1855646 and Grant 1953544. *(Corresponding author: Jinhui Wang.)*

Jingyan Fu and Zhiheng Liao are with the Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58102 USA (e-mail: jingyan.fu@ndsu.edu; zhi-heng.liao@ndsu.edu).

Jianqing Liu is with the Department of Electrical and Computer Engineering, University of Alabama in Huntsville, Huntsville, AL 35899 USA (e-mail: jianqing.liu@uah.edu).

Scott C. Smith is with the Department of Electrical Engineering and Computer Science, Texas A&M University-Kingsville, Kingsville, TX 78363 USA (e-mail: scott.smith@tamuk.edu).

Jinhui Wang is with the Department of Electrical and Computer Engineering, University of South Alabama, Mobile, AL 36688 USA (e-mail: jwang@southalabama.edu).

Digital Object Identifier 10.1109/JIOT.2020.3023623

However, edge computing in IoT systems has strict requirements. It must be extremely power efficient and high speed, because the edge usually has limited resources, especially when it involves real-time scenarios as well as artificial intelligence (AI) tasks for long-term bursts of intense computation over large data sets.

Memristor-based in-memory processing system is a promising candidate for edge computing systems. The notion of memristor was envisioned dozens of years ago, and its physical realization was demonstrated by Hewlett-Packard Lab in 2008 [1], [2]. A memristor has a simple three-layer structure whose conductance value can be changed according to an applied pulse. Different from traditional CMOS-based hardware, such as field-programmable gate array (FPGA), application-specific integrated circuit (ASIC), graphics processing unit (GPU), and tensor processing unit (TPU), the memristor-based in-memory computing architecture breaks the memory wall that results from the Von Neumann architecture, thereby achieving at least a $7\times$ reduction of active power for implementing a basic neuron function [3]. Also, memristors have better CMOS process compatibility as compared with emerging technologies, such as quantum computing, molecular computing, quantum dots, and spin-wave devices. Furthermore, recent years have witnessed significant progress in mobile devices and wireless sensor networks, creating unprecedented opportunities to deploy deep learning for smart IoT applications. However, deep learning involves a massive number of calculations due to large data sets and multilayer network structure, especially in the scenario of online learning and incremental learning. A memristor-crossbar system can efficiently perform such operations due to its programmable and computable analog structure, where computations are conducted in a parallel manner.

When deep learning algorithms work on the edge for human-related applications, IoT devices will collect data, which in some cases may contain quite personal and high-value user information, therefore raising significant concerns regarding information privacy. It is thus possible for the private data to be misused, or to be hacked by outside attacks. Therefore, some sort of privacy protection method is required to guarantee a strong notion of privacy, while preserving learning accuracy. A traditional privacy protection method is anonymization or deidentification, which removes attributes in the data and returns a sampled data to protect privacy. However, it is still possible to discern information about the data sets, which is known as linkage attacks, statistical inference attacks, generative adversarial networks (GANs)-based attack, and reidentification attacks [4], [5]. For example, if we use deep learning for cancer diagnosis, when we release a learning model, we may unintentionally disseminate information about the training data set so that a malicious attacker could identify individuals diagnosed with cancer. Another privacy protection method is via cryptography such as homomorphic encryption, but it requires large computational overhead [6]. Currently, differential privacy [7] is one of the most popular technologies for privacy-preserving deep learning, realized by introducing perturbations. Differential privacy (DP) provides a mathematical constraint for the privacy loss associated with any data release from a statistical database. In edge computing, federated learning is a recent advance in private learning, where the model is trained in a decentralized manner without sharing the raw data. It prevents a third party from storing personal data as well as performing learning tasks on that data. Despite such privacy improvements, local DP is necessary for federated learning because the weight updates uploaded by individual edge devices may still reveal private information. Adding noise to the weights/gradients during training on local data prior to aggregation by an untrusted server provides greater privacy protection to users [8]. However, as a result, differentially private learning systems not only need to train complex models but also must perform an additional computation for noise insertion as a protective mechanism to data sets, models, and algorithms [9]. Such a high-cost training process undoubtedly challenges traditional CMOS-based hardware technology, and hinders the development of deep learning, especially for power-sensitive and resource-limited edge computing in IoT systems. Therefore, a memristor-based differentially private learning system, with its low computation and storage cost, is an excellent candidate for edge AI.

As for memristor, some researchers have revealed its nonideal properties, including nonlinearity (NL), device-to-device variation, cycle-to-cycle variation, maximum conductance variation, and minimum conductance variation [10]–[13]. These nonideal properties negatively influence model precision of a memristor-based system; however, such variations can instead be considered as inherent noise, which is necessary for DP preservation. In this article, we use memristor cycle-to-cycle variation as an advantage to realize hardware-based Gaussian noise injection. The proposed methods add crafted noise to the edge AI system, and avoid introducing computational complexity and extra hardware/algorithmic units, which greatly improves efficiency and saves cost for a privacy-preserving edge AI system. Specifically, this article makes the following contributions, as summarized in Fig. 1.

1) *Hardware Solution That Breaks the Limitations of Traditional Software-Based Noise-Adding Mechanisms of DP:* A memristor-based hardware solution is proposed for differentially private learning systems that do not require additional circuitry. In this article, the positive and negative pulse pair (PN) method is used to generate adjustable Gaussian noise, satisfying the DP constraint. The proposed method transforms nonbeneficial cycle-to-cycle variations into a valuable measure for privacy protection.
2) *Methods That Address Differentially Private Stochastic Gradient Descent (DP-SGD) by Hardware Implementation:* The clipping method is proposed to avoid the L_2 norm calculation of gradient matrices. A combination of the PN method and clipping method, called noise distribution normalization (NDN) method, is proposed to implement the DP mechanism.
3) *Privacy Analysis and Performance Evaluation:* Privacy analysis is conducted to verify the effectiveness of our proposed methods. Furthermore, to illustrate the performance of each method, a comprehensive suite of simulations has been conducted.

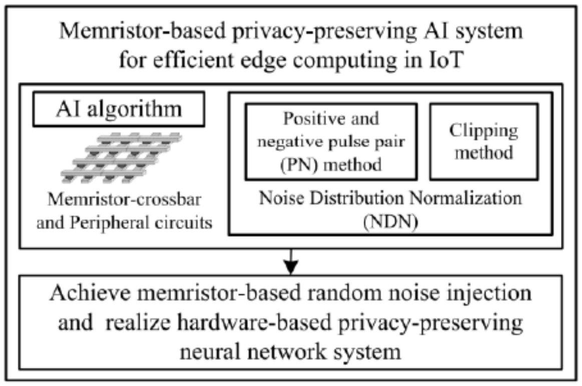

Memristor-based privacy-preserving AI system for efficient edge computing in IoT
AI algorithm
Memristor-crossbar and Peripheral circuits
Positive and negative pulse pair (PN) method
Clipping method
Noise Distribution Normalization (NDN)
Achieve memristor-based random noise injection and realize hardware-based privacy-preserving neural network system

Fig. 1. Outline of the proposed method.

The remainder of this article is organized as follows. First, we introduce the related work in Section II. Section III provides the background on memristor technology and DP used as the theoretical basis for our work. Then, we analyze the Gaussian distributed random variable for noise injection in Section IV. In Section V, we describe the proposed method as well as its implementation. Section VI details a case study based on the proposed method, followed by results and privacy analysis; and we also discuss the NL, scalability, and endurance of memristor-based crossbar arrays. Finally, we conclude this article in Section VII.

II. Related Work

Most differentially private learning systems for IoT focus on the algorithm-based framework improvement and optimization. For example, Xu *et al.* [14] proposed a framework that uses a protection layer to perform noise injection; and Arachchige *et al.* [15] designed a mechanism named LATENT that adds a randomization layer between the convolutional module and the fully connected module to perturb data for machine learning services. In [16], the Google security and privacy team released a private aggregation via teacher ensembles (PATEs) framework that achieved private learning by carefully coordinating the activity of several different machine learning models. DP-SGD [17] makes fewer assumptions about the machine learning task than PATE, but it comes at the expense of making modifications to the training algorithm. Such technologies inevitably need a large cost of noise-generation computing. Furthermore, all the above frameworks are software-based privacy protection technologies that might not be deployable on IoT devices due to resource constraints.

Other researchers have presented novel hardware-level solutions. In [18], low-voltage static random-access memory (SRAM) chips are used to add bit failures as training data noise. This method can save energy, but the added noise only followed a uniform distribution, and does not guarantee DP. In [19] and [20], in order to generate random numbers with true randomness, dedicated random number generation modules, such as physical unclonable function (PUF) and random number generator, are designed. These modules are accurate, but require additional circuitry. In [10], an advanced neuromorphic system is implemented based on memristor arrays, but noise is inserted in training data to promise strong theoretical privacy guarantees, where the hardware and software are utilized separately for learning and privacy protection.

Different from the above approaches, we propose a method that can achieve differentially private learning in edge AI with high efficiency and low computing cost by taking advantage of memristor-based hardware variations, which does not require any additional hardware or software. In this article, we utilize a 3-layer fully connected network and Modified National Institute of Standards and Technology (MNIST) database as an example to verify the proposed method. However, our proposed method is generic, and can be applied to other databases and any deep learning models that can be mapped into a memristor-based crossbar array, as discussed in Section VI-E.

III. DP and Memristor Technology Background

A. Differential Privacy

DP protection technology is recognized as a rigorous and robust protection model. The basic idea of this model is to add specific noise so that inserting or deleting a record in a data set does not statistically affect any calculated output. DP provides provable guarantees of privacy, mitigating the risk of exposing sensitive training data in machine learning [21]. The definition of (ε, δ)-DP is given below [21], [22]. A randomized mechanism, A, satisfies (ε, δ)-DP when any adjacent input data sets, d and d', and any output, S, of A satisfy

$$Pr[A(d) = S] = e^{\varepsilon} \cdot Pr\big[A\big(d'\big) = S\big] + \delta. \tag{1}$$

In our study, each training data set is a set of image-label pairs. Given a negligibly small probability, δ, parameter ε is the privacy budget, which measures the privacy bound of the randomized mechanism, A, for adjacent data sets. A smaller value of ε means higher indistinguishability, thus a stronger privacy guarantee. By this definition, privacy preservation can be calculated and evaluated through ε, given δ.

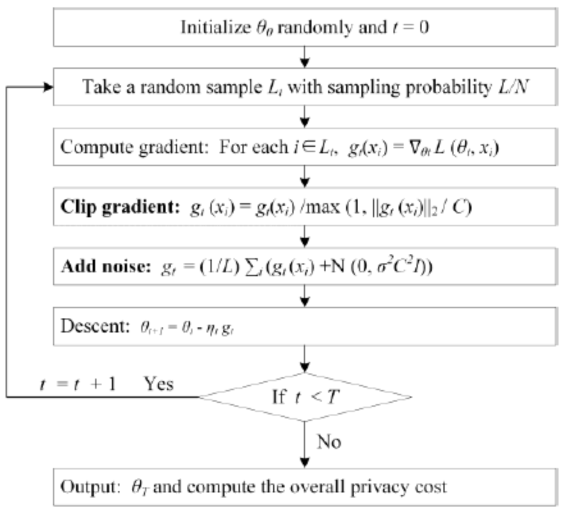

Initialize θ_0 randomly and $t = 0$
Take a random sample L_t with sampling probability L/N
Compute gradient: For each $i \in L_t$, $g_t(x_i) = \nabla_{\theta_t} L(\theta_t, x_i)$
Clip gradient: $g_t(x_i) = g_t(x_i) / \max(1, \|g_t(x_i)\|_2 / C)$
Add noise: $g_t = (1/L) \sum_i (g_t(x_i) + N(0, \sigma^2 C^2 I))$
Descent: $\theta_{t+1} = \theta_t - \eta_t g_t$
$t = t + 1$
Yes
If $t < T$
No
Output: θ_T and compute the overall privacy cost

Fig. 2. Outline of DP-SGD [17]. Symbols and parameters: input data set, weights θ, loss function $\mathcal{L}(\theta)$, gradient g, learning rate η_t, noise scale σ, group size L, gradient norm bound C, total weight update step T, and the square root of the largest eigenvalue of the matrix $g_t(x_i) * g_t(x_i)$, $\|g_t(x_i)\|_2$, where $g_t(x_i)*$ denotes the conjugate transpose of $\|g_t(x_i)\|_2$.

B. Differentially Private SGD Algorithm

DP-SGD is a modification of the stochastic gradient descent (SGD) algorithm that is popular and serves the basis for many optimizers in machine learning [17]. Models trained with DP-SGD have provable privacy guarantees in terms of DP. Instead of working only on final parameters from the training process, DP-SGD controls the influence of training data during the training process. Fig. 2 outlines the principles for training a model with weight parameters, θ, by minimizing the loss function.

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 SGD is a differentially private algorithm. The first is to clip the gradient computed on each training image to limit how much each training image can impact model parameters. The algorithm clips each gradient by a clipping threshold, C. In this article, we use L_2 norm of $g_t(x_i)$ to represent $\|g_t(x_i)\|_2$, which is explained in Fig. 2. The second is to sample and add random noise to randomize the algorithm. Thus, it is statistically impossible to identify whether a particular sample is included in the training set.

In this article, we achieve the algorithmic essence of DP-SGD using a memristor-based neural network and our proposed methods. In doing so, we realize a privacy-preserving memristor-based learning system without introducing extra computational processing or noise generation units.

C. On-Chip Training of Memristor-Based Multilayer Neural Networks

Memristors in a crossbar array structure can carry out vector–matrix multiplication operations in parallel within the analog domain, which can yield learning systems with high

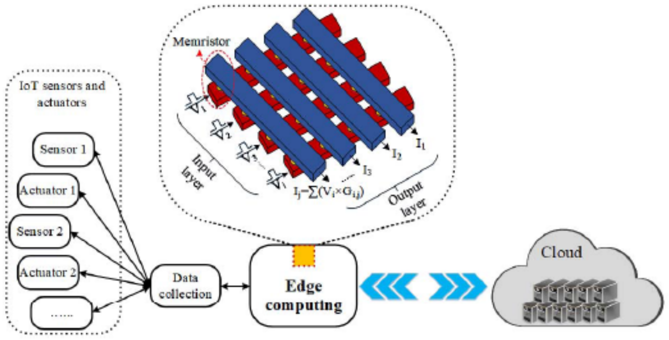


Fig. 3. Hardware implementation of neural networks using memristor crossbar. V_i, $G_{i,j}$, and I_j represent the input signal in the *i*th row, the conductance of the memristor in the *j*th column and *i*th row, and the output current that represents the dot product result of V and G, respectively.

throughput, low energy consumption, and minimal area. On-chip learning with a memristor crossbar array provides the learning system with capability for real-time learning [23]. As shown in Fig. 3 for the hardware implementation of an IoT system, the edge neural network can be directly mapped onto a crossbar structure, where the vector–matrix multiplication can be conducted by applying input voltages to each row and reading currents from each column. Although many peripheral circuits are still required for complete functionality of the on-chip learning system, the memristor's desirable properties support the crossbar circuit to be a promising substitute technology for traditional ones.

D. Memristors and Cycle-to-Cycle Variation

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. 4, positive and negative input voltage pulses that are larger than the threshold voltage can switch a memristor gradually from $G_{\min}$ to $G_{\max}$ or from $G_{\max}$ to $G_{\min}$, where $G_{\min}$ and $G_{\max}$ represent minimum conductance and maximum conductance, respectively. Thus, the conductance/weight increase process is called long-term potentiation (LTP) and the conductance/weight decrease process is called long-term depression (LTD) [24].

In the backpropagation phase of the DP-SGD algorithm, the weight update values (Δw) will be translated to a 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 memristor variations. Among all variations of memristors not attributed to manufacturing process, cycle-to-cycle variation is caused by intrinsically stochastic resistance switching mechanisms [25]–[28] that can be approximated as a Gaussian or normal distribution [19], [20], [29]–[33]. It originates from the random formation and disruption of conducting filaments [27] and the co-existence of multiple subfilaments, where the active, current-carrying filament may change from cycle to cycle [28], [34]. To prove such randomness introduced by each programming operation, 500 cycles [35] and 5000 cycles [32] of experimental data are collected. Cycle-to-cycle variations result in the different updated conductance when the same updating signal in different updating cycles is applied to a memristor, even when the initial conductance is the same.

IV. Gaussian Distribution of Cycle-to-Cycle Variation

A. Mathematical Expression of Cycle-to-Cycle Variation

As discussed in Section III-D, when a memristor is incented by an input pulse, its conductance is changed not only by the designed value, $(G_{\max} - G_{\min})/N_{\text{Level}}$ but also by cycle-to-cycle variation, X. Here, X is modeled as standard normal distribution, $N(0, \sigma)$, where the noise scale, σ, is determined by the pulse width [19], [20], [29]–[33]. When taking n input pulses as an example, the total conductance variation (G_{var}) generated in one memristor can be represented by

$$G_{\text{var}} = X_1 + X_2 + \cdots + X_n \tag{2}$$

$$X_1 \sim N\left(0, \sigma_1^2\right), X_2 \sim N\left(0, \sigma_2^2\right), \ldots, X_n \sim N\left(0, \sigma_n^2\right) \tag{3}$$

where $X_1, X_2, \ldots, X_n$ are cycle-to-cycle variation variables that are introduced by input pulses from the first pulse to the nth pulse, respectively; and $\sigma_1, \sigma_2, \ldots, \sigma_n$ represent the standard deviation of the noise of each input. Because the widths of weight updating pulses are the same and applied independently, $X_1, X_2, \ldots, X_n$ are independent and identically distributed random variables, which indicates their means and variance are identical ($\sigma_1 = \sigma_2 = \cdots = \sigma_n = \sigma_{\text{in}}$). Then, based on probability theory [36], [37], the variable G_{var} follows a joint Gaussian distribution that can be represented as shown in

$$G_{\text{var}} \sim N\left(0, n\sigma_{\text{in}}^2\right) \tag{4}$$

with a probability density function (pdf) shown in

$$f(G_{\text{var}}) = \frac{1}{\sigma_{\text{in}}\sqrt{2n\pi}} e^{-\frac{1}{2n}\left(\frac{G_{\text{var}}}{\sigma_{in}}\right)^2}. \tag{5}$$

B. Cycle-to-Cycle Variations of Positive and Negative Pulse Pairs

As discussed above, due to cycle-to-cycle variation, which is an inherent characteristic of memristors, even in a routine weight update process, model parameters of a memristor-based hardware system suffer from random noise addition. An input pulse introduces cycle-to-cycle variation by changing memristor conductance by $(G_{\max} - G_{\min})/N_{\text{Level}}$; and as shown in Fig. 4, a positive pulse and negative pulse make conductance increase and decrease, respectively. Thus, when the number of such positive and negative pulses are equal, this is equivalent to adding cycle-to-cycle noise that follows a Gaussian distribution. For example, when we apply one positive pulse, the conductance of the memristor changes from G_0 to G_1, as shown in

$$G_1 = G_0 + ((G_{\max} - G_{\min})/N_{\text{Level}}) \times 1 + X_1. \tag{6}$$

After then applying one negative pulse, it changes from G_1 to G_2, as shown in

$$G_2 = G_1 - ((G_{\max} - G_{\min})/N_{\text{Level}}) \times 1 + X_2 \tag{7}$$

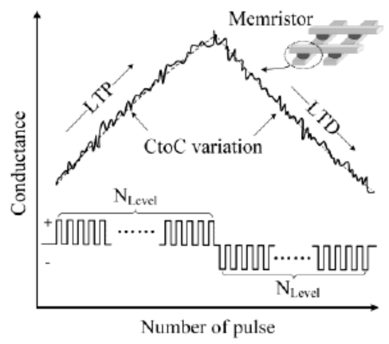


Fig. 4. LTP process, LTD process, and cycle-to-cycle variation of a memristor. N_{Level} represents the number of conductance states of a memristor.

where $X_1 \sim N(0, \sigma_1^2)$ and $X_2 \sim N(0, \sigma_2^2)$. Now add (6) and (7), and let $X_1 + X_2 = G_{\text{add}}$, which results in

$$G_2 = G_0 + X_1 + X_2 = G_0 + G_{\text{add}}. \tag{8}$$

Finally, let $\sigma_1^2 + \sigma_2^2 = \sigma_{\text{add}}^2$, which yields

$$G_{\text{add}} \sim N\left(0, \sigma_{\text{add}}^2\right). \tag{9}$$

Based on probability theory [36], [37], X_1 and X_2 are independent random variables. Assuming that $\sigma_1^2 + \sigma_2^2 = \sigma_{\text{add}}^2$, then after applying a pair of positive and negative pulses, the effective noise introduced to the target memristor follows a joint Gaussian distribution, $N(0, \sigma_{\text{add}}^2)$. Hence, applying m positive/negative PNs to a memristor injects noise that follows the joint Gaussian distribution, $N(0, m\sigma_{\text{add}}^2)$, without requiring additional circuitry. With such a noise injection method, the scale of injected noise is decided by m, the number of positive/negative PNs, and their pulse width.

It should be noted that due to the various types of memristors, when we apply this positive/negative PN method, the intrinsic characteristics of the specific memristor need to be considered. For memristors discussed in [38]–[41], the conductance only changes when the amplitude of the input pulse is larger than the threshold voltage, and its duration is larger than the switching time, which is the necessary condition for noise injection. Therefore, for this kind of memristor, the minimum noise injection is constrained by its threshold voltage and switching time. Nevertheless, as for memristors discussed in [2], [42], and [43], the conductance change and cycle-to-cycle variation exist whenever any input is applied, so, the injected noise can be set at a much smaller scale.

In this article, we denote the noise injection method by using positive/negative PNs as the PN method.

V. Noise Injection in Accordance With DP

For a memristor in a learning system, when n pulses are applied (assume n is a positive integer), the injected noise, G_{var}, regarded as system built-in noise, follows a joint distribution $N(0, n{\sigma_{\text{in}}}^2)$. However, for different memristors, the injected noise is not identical, and the independent Gaussian noise is impractical to be obtained in a trackable form. Hence, this causes difficulty in tracking the consumed privacy budget and evaluating utility loss. Therefore, this article proposes a hardware-based NDN method to normalize introduced noise for all weights (conductance of memristors) in a learning model. The proposed method transfers nonbeneficial random noise of cycle-to-cycle variations into a valuable measure for privacy protection by using the proposed PN method, which improves the utility of the privacy-preserving neural network with an excellent privacy guarantee.

A. Noise Distribution Normalization Method

As discussed in Section IV, for each memristor, G_{var} is built-in noise of an AI algorithm. G_{add} by PN method is adjustable noise for implementation of adjustable noise injection. Variables G_{var} and G_{add} are independent but not identical Gaussian random variables. Let, $G_{\text{noise}} = G_{\text{var}} + G_{\text{add}}$, then

$$G_{\text{noise}} \sim N\left(0, n\sigma_{\text{in}}^2 + m\sigma_{\text{add}}^2\right). \tag{10}$$

We propose the NDN method to achieve DP protection by hardware implementation. To ensure that every gradient carries noise with the same distribution injected, first we clip the gradient into n_c, which is a constant value. Second, for one memristor, if the needed pulses $n >= n_c$, let $n = n_c$; otherwise, keep n value and let $m = (n_c - n) * {\sigma_{\text{in}}}^2/\sigma_{\text{add}}^2$. Given that the proposed method is hardware-based, where σ is related to actual characteristics of memristors, when we implement this method, σ_{add} can be tuned to a value that ensures m is an integer. Hence, when the needed pulses are $n \geq n_c$, $n\sigma_{\text{in}}^2 + m\sigma_{\text{add}}^2 = n_c\sigma_{\text{in}}^2$; and when the needed pulses are $n < n_c$, $n\sigma_{\text{in}}^2 + m\sigma_{\text{add}}^2 = n\sigma_{\text{in}}^2 + ((n_c - n) * {\sigma_{\text{in}}}^2/\sigma_{\text{add}}^2)) * \sigma_{\text{add}}^2 = n_c\sigma_{\text{in}}^2$. In so doing, for every memristor at each weight update, G_{noise} follows a $N(0, n_c\sigma_{\text{in}}^2)$ Gaussian distribution, with a pdf as shown in the following:

$$f(G_{\text{noise}}) = \frac{1}{\sigma_{\text{in}}\sqrt{2\pi n_c}} e^{-\frac{1}{2n_c}\left(\frac{G_{\text{noise}}}{\sigma_{\text{in}}}\right)^2}. \tag{11}$$

B. Privacy Analysis

To ensure that our design preserves the DP notion, we need to first guarantee that each gradient descent step is (ε, δ)-differentially private. For our crafted Gaussian distribution and pdf with $G_{\text{noise}} \sim N(0,n_c\sigma_{\text{in}}^2)$ that we use to sample noise, by the standard definition in [21, Th. 3.22], one needs to ensure that $\sigma_{\text{in}}\sqrt{n_c} \geq \sqrt{2\log(1.25/\delta)}/\varepsilon$ and $\varepsilon < 1$. For the former inequality, it can be easily satisfied by adjusting the number of pulses, n_c, while the later inequality is selected artificially at runtime. In light of this, our Gaussian distribution can guarantee the DP notion for each iteration in Fig. 2.

Next, we need to prove the overall process of running the DP-SGD algorithm in memristor crossbar for the DP notion, and theoretically evaluate how much privacy budget costs to preserve DP. First, since at each iteration we independently sample random noise from the identical Gaussian distribution $N(0,n_c\sigma_{\text{in}}^2)$ by applying the same number of pulses to the memristor crossbar, according to the composition theorem of (ε, δ)-differential privacy [21], the overall process still preserves the DP notion. Next, based on the privacy amplification theorem [44], for the process of random sampling with probability *L/N*, as in Fig. 2, each iteration is technically

$O((\varepsilon L/N), \delta L/N)$-differentially private. Then, the strong composition theorem implies that, for a sufficiently large number of iterations, T (i.e., we expect $T \gg N/L$ and each sample is examined multiple times), the overall budget cost is $\varepsilon_{\text{tot}} = \Omega((L/N)\sqrt{T\log(1/\delta)\log(T/\delta)}/\sigma_{\text{in}}\sqrt{n_c})$. It is worth noting that [17] developed a moment accountant technique to obtain a much tighter bound on the accumulated privacy budget, which allows for using a smaller variance value in the Gaussian distribution. Here, our intention is only to prove and showcase the preservation of DP and its estimated cost, despite loose privacy loss. Interested readers are referred to [17] for the details on the moment accountant analysis.

VI. Case Study and Discussion

As discussed in Section V, the proposed memristor-based NDN method can effectively realize DP by injecting normalized random noise to edge AI in IoT systems. In this section, based on a hardware machine learning platform that consists of memristor crossbar array and peripheral circuits, we perform a case study that implements the DP-SGD algorithm by NDN.

A. Implementation of DP-SGD via Hardware in Edge AI

As illustrated in Fig. 2, two modifications (clip gradient and add noise) are needed to ensure that the proposed SGD follows a DP algorithm. For the first modification, in order to constrain how much each individual 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 data set, which can be achieved by adding random noise to the clipped gradients.

For the first modification, Fig. 2 shows that the gradient clipping process needs to calculate the L_2 norm of the gradient matrix. These processes inevitably increase the computational load of the system. We propose a clipping method that is the first step of the NDN method to meet the requirement of clipping without matrix calculation. The clipping operation sets an upper boundary on gradients to bound the influence of each individual example on gradients. A smaller upper boundary has a stronger limitation of the gradient. When the NDN clip boundary is small enough, the L_2 norm of the gradient is always less than 1. In this circumstance, L_2 norm computing is simplified because 1 is always chosen as the maximum value in the first modification according to the DP-SGD shown in Fig. 2. As shown in Fig. 5, in our hardware implementation, this method uses simple hardware units—comparators to compare the gradient value with a reference gradient value. When the gradient is clipped, the maximum number of weight update pulses is fixed. Accordingly, the system saves the cost of matrix calculation and also ensures that the degree of each training sample's impact on model parameters is bounded.

The second modification can be implemented by the PN method, which is the second step of the NDN method. The PN method adds random noise by applying extra input PNs to memristors. As discussed in Section IV, in each step, when extra PN pairs of input pulses are added to each memristor, the amount of the designated change in conductance caused by such positive and negative pulses counteract each other, such that only random noise is added to the memristor conductance. 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. Fig. 5(b) shows the hardware implementation flow of the NDN method. Hence, when using hardware that combines a memristor array-based learning circuit along with the NDN method, the DP-SGD can be implemented using the existing hardware with minimal additions.

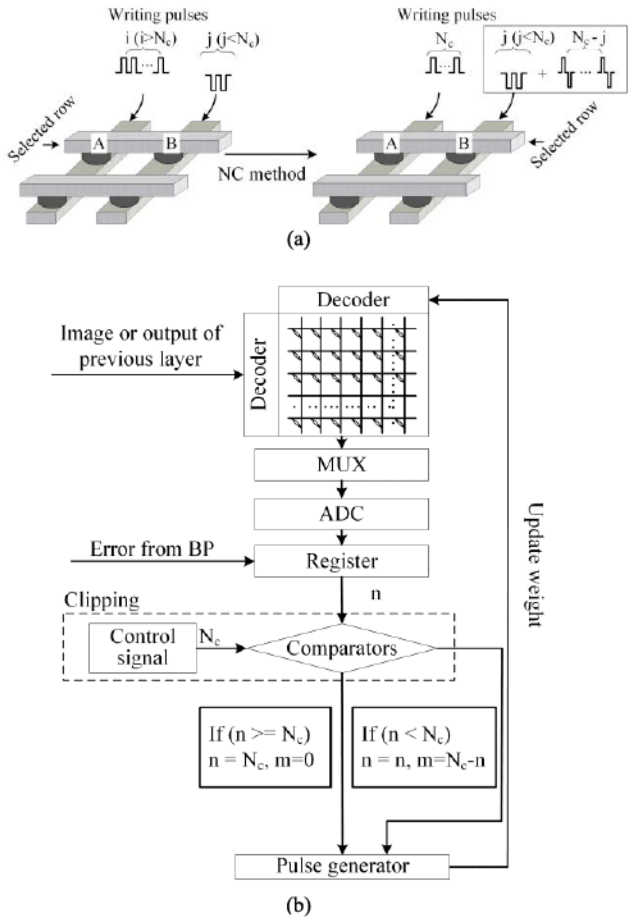


Fig. 5. Workflow of the NDN method, where n represents the number of pulses that are used to update a weight and m represents the number of positive and negative PNs. (a) Example: after applying the NDN method, for memristor A $n = N_C$, $m = 0$; for memristor B, $n = j$, $m = N_C - j$. (b) Hardware implementation flow of the NDN method.

B. Results

To verify our proposed methods for edge AI, we adopt the neural network hardware platform, NeuroSim+ [24]. This is a memristor-based circuit model of neuro-inspired architectures to emulate the circuit behavior of an online learning recognition scenario with MNIST [45] data set. The neural network topology of this simulator includes input layer, hidden layer, and output layer, with 400 neurons, 100 neurons, and ten neurons, respectively. The simulator emulates hardware to train the network with images randomly chosen from the

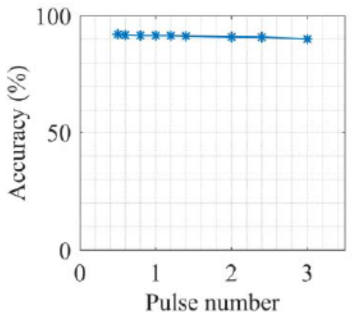


Fig. 6. Recognition accuracy of MNIST under various clip boundaries.

training data set MNIST, which includes 60 000 images, and classify the testing data set with 10 000 images. The training process has two parts, feedforward propagation and 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 NDN method into this simulator to train privacy-preserving multilayer neural networks, such that we can only use hardware that consists of memristor array and peripheral circuits to realize DP-SGD behavior.

1) Clipping of NDN: To explore the effect of the clipping step of NDN, the clipping method is applied to a fully connected neural network without the PN method and without cycle-to-cycle variation. As discussed in Section VI-A, the clipping operation limits the scale of gradient, and a smaller clip boundary has a larger limitation of the gradient. Also, as each gradient value is clipped, the number of pulses is also clipped, and the influence of each image is limited. In our simulation, we use a clip boundary to clip the gradient of each weight. Then, the number of pulses for each weight is obtained, where *the number of Pulse* = *rounding* (*clipping* (*gradient*) * *learning_rate* * N_{Level}). The rounding operation converts the value to its nearest integer number. In our case, when the clip boundary is less than 0.2, the L_2 norm of the gradient is always smaller than 1. Therefore, the clipping method saves the matrix calculation cost of DP-SGD-based hardware systems. Fig. 6 shows the recognition accuracy of MNIST handwriting digits as the clip boundary value changes. These results indicate that as the clip boundary decreases, the recognition accuracy stays at a high level, which concludes that the clipping operation does not degrade performance of the three-layer neural network based on the DP-SGD algorithm.

2) NDN Method: Since many types of memristors exist, the proposed methods are explored with various configurations, including the twelve noise levels shown in Table I. Fig. 7 shows recognition accuracy of MNIST handwriting digits under various noise levels with and without NDN method. The one without the NDN method is named Original, and it does not consider cycle-to-cycle variation as noise injection, but instead adds Gaussian noise via software. As shown

TABLE I
SIGMA PARAMETER OF CYCLE-TO-CYCLE VARIATION

Level	1	2	3	4	5	6
$\sigma/(G_{max}-G_{min})$	1%	2%	3%	4%	5%	6%
Level	7	8	9	10	11	12
$\sigma/(G_{max}-G_{min})$	7%	8%	9%	10%	11%	12%

The σ value of Gaussian Distribution for Cycle-to-Cycle variation is represented by the percentage of memristor's conductance range [24], [29].

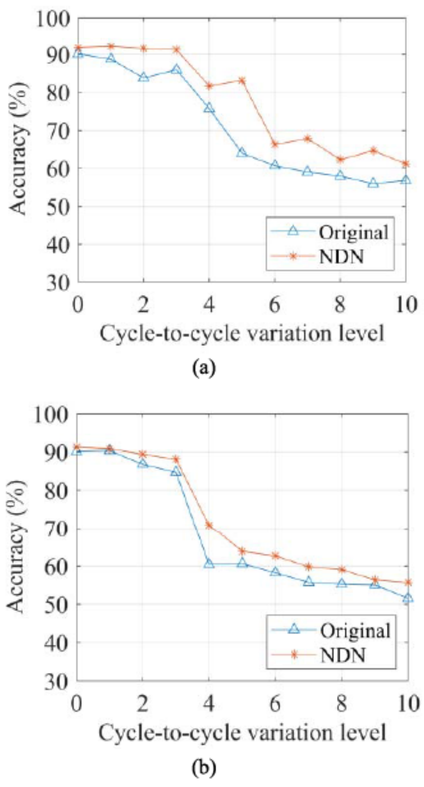


Fig. 7. Recognition accuracy of MNIST handwriting digits under various noise levels. (a) $N_c = 1$. (b) $N_c = 2$.

in Fig. 7, under $N_c = 1$ and $N_c = 2$, the average recognition accuracy of these ten variation circumstances with the NDN method is improved by 7.4% and 3.5%, respectively, as compared with the original case. Fig. 8 also provides a recognition accuracy comparison, but with different N_c and different variation levels (levels from 2 to 8). It shows that the NDN method has 15.5% higher accuracy on average as compared to the Original method. This is because for memristor-based learning system in the Original case, cycle-to-cycle variation still exists, in addition to the noise added via software, which leads to prediction accuracy loss. Such loss will be larger than the case with the NDN method that only considers the inherent cycle-to-cycle variation without additional noise injection via software. These results support the proposed NDN method as an effective optimization method that can improve the utility of a memristor-based differentially private learning model.

In Fig. 9, we show the evolution of accuracy with different noise levels, σ, as a function of the number of training images. It shows 90.5%, 75.4%, and 54.4% recognition accuracy when σ equals 3%, 6%, and 12% from Table I, respectively.

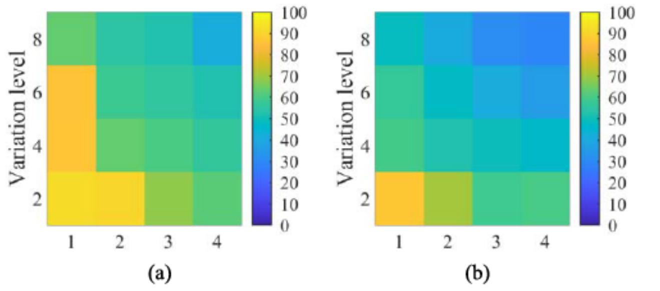


Fig. 8. Recognition accuracy of MNIST handwriting digits under various noise levels and different number of PNs: (a) using NDN method, where the x-axis represents the number of PNs, N_C; (b) Original case, where the x-axis has the same noise scale as (a).

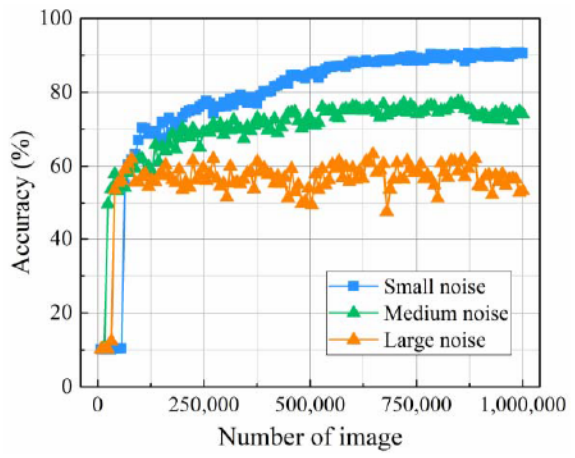


Fig. 9. Recognition accuracy of training process using the NDN method under three noise levels, where σ for small noise, medium noise, and large noise equals 3%, 6%, and 12%, respectively.

TABLE II
COMPARISON WITH STATE OF THE ART

	[10] in 2019	[18] in 2017	[20] in 2019	This work
Realize hardware-based privacy protection	×	√	×	√
Without extra random noise generator	×	√	×	√
Without always adding random noise for each cell	×	×	×	√
Compatible with memristor-based hardware	√	×	√	√

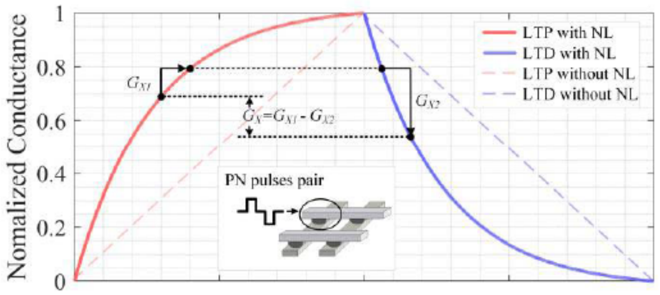


Fig. 10. NL effect on conductance modulation of memristors.

TABLE III
RECOGNITION ACCURACY WITH NDN METHOD AND NL

Nonlinearity (NL)	0.00	0.05	0.15	0.25
NDN	92.4%	90.2%	77.5%	59.3%
Original	91.0%	87.3%	67.2%	44.5%

C. Comparison With Existing Work

As listed in Table II, as compared with the state of the art, [10], [18], [20], instead of implementing the DP-SGD algorithm for privacy preservation by a traditional computing system, the proposed NDN method adds Gaussian noise for memristor-based hardware using inherent cycle-to-cycle variation. Thus, the memristor-based machine learning system does not require an additional random noise generator. Also, the scale of injected noise can be adjusted by changing the number of PN PNs. For the DP-SGD algorithm, the clipping method of NDN limits the impact of each training data on model parameters and saves the cost of the L_2 norm matrix calculation.

D. Nonlinearity

In general, the amount of conductance change of memristors sometimes is different as the number of pulses increases, which is attributed to the NL of the weight modulation. In other words, every pulse results in a different response in the weight modulation depending on the current weight state [46] when we apply a pair of positive and negative pulses to a memristor with the NL consideration, as shown in Fig. 10, where $G_X = G_{X1} - G_{X2}$, (7) can be rewritten as follows:

$$G_2 = G_0 + G_X + G_{\text{add}} \tag{12}$$

where $G_0 + G_X$ has the same conductance range as G_0, which is from $G_{\min}$ to $G_{\max}$. When we map the memristors' conductance to the gradient of DP-SGD, both G_0 in (8) and $G_0 + G_X$ in (12) are mapped to the same range, which is from 0 to 1. Thus, when we consider the NL effect of memristors, the global sensitivity of gradient remains unchanged, and so will the noise scale of G_{add} (i.e., variance of the Gaussian distribution). Therefore, the NDN method still conforms to the DP theory.

We conduct experiments to explore the NDN method considering the NL of memristors. We adopt the NL definition from [47], where NL ranges from 0 to 1. As shown in Table III, with NL ranging from 0.05 to 0.25, recognition accuracy with NDN is still 9.3% on average higher than the Original case without NDN. This shows the effectiveness of the proposed method, where the NL is being used as an advantage instead of degrading accuracy, as already discussed in Section VI-B2. Although the NL property does degrade the performance of the memristor-based learning system, a memristor-based learning system can achieve an acceptable performance with the NL and the proposed NDN method. Moreover, researchers have proposed memristors with a small NL [48] as well as methods to solve the NL issue [12], [13]. Thus, the proposed method is still effective even with the NL property.

TABLE IV
TOTAL POWER OF MEMRISTOR-BASED ARRAY IN LEARNING DEVICE

Number of Hidden Layer Neurons	50	100	150	200
Power (10^{-7} J)	3.2	4.3	5.3	7.1
Recognition Accuracy (%)	90.2	91.1	92.2	92.8

TABLE V
RECOGNITION ACCURACY WITH VARIOUS FAILURE RATES

Failure Rate	0%	5%	10%
Recognition Accuracy	92.4%	90.7%	90.6%

E. Scalability and Endurance

Many applications employing a neural architecture use memristors in the edge, where our proposed method can be adopted, including for example, face classification with artificial neural network (ANN) structure [49], [50], handwritten digits classification [51] and image processing [52] with convolutional neural network (CNN) structure, memristor-based edge detection [53], and pattern recognition with recurrent neural network (RNN) [54]. Our proposed method is generic and can be applied to any memristor-based learning system. However, the number of neurons and layers in deep learning neural networks may cause scalability issue for a memristor-based array. Peng *et al.* [55] solved this issue by using a chip-level hierarchical architecture that divides large arrays into groups of synaptic subarrays, and connects each subarray using an H-tree structure. To further explore network scalability, we simulated the learning tasks introduced in Section VI-B using the various network structures listed in Table IV, which shows that power increases with the number of neurons, but accuracy also increases; hence, there is a tradeoff between power and accuracy. Note that the power of memristor array is calculated based on wire resistance, reading, and the weight update writing process.

Another challenge for memristor crossbar arrays is the sneak path issue, which severely degrades read sensing margin [56]. One solution is to increase the minimum conductance value, but this degrades ON/OFF ratio of memristors. Our experiments utilize a one-transistor one-resistor (1T1R) array to avoid sneak path current problems [24].

The failure rate, endurance, and aging issues also impact the performance of memristor-based edge systems. The typical memristor failure rate is less than 10% [57]. As shown in Table V, we have conducted learning tasks with up to 10% failure rate, while recognition accuracy shows no obvious degradation. This indicates that neuromorphic architectures are more robust to variations because, in the learning process, weights updates frequently in each epoch to compensate for mismatch resulting from variations. Memristors have high endurance (120 billion cycles) and retention (ten years) even when they undergo high frequency writing and reading in learning systems [47], which qualifies them for most edge computing in IoT applications.

Additionally, the conductance range of memristors may deviate from the original state over time. As shown in Table VI, the recognition accuracy decreases by 27.74% when

TABLE VI
RECOGNITION ACCURACY WITH CONDUCTANCE DRIFTING

Maximum Conductance Drift	0%	10%	20%	30%
Recognition Accuracy	92.4%	92.0%	91.4%	63.7%

conductance drifting is 30%, but it does not show a significant decrease when it is 10% or 20%.

VII. CONCLUSION

In this article, to meet the high-speed, low-power, and low computing-cost requirements of edge computing in IoT systems, we propose a universal memristor-based method that can be used to realize a privacy-preserving learning system. The proposed NDN method consists of a positive/negative PN method and a clipping method. The PN method can generate adjustable Gaussian noise based on cycle-to-cycle variation of memristors, without extra hardware or a random noise generator, making it possible to meet the noise-injection requirement of the DP mechanism. The clipping method that uses comparator units can normalize the introduced noise from the PN method.

In the case study of a memristor-based neural network hardware platform, we implement the DP-SGD algorithm via hardware-based NDN method, and at the same time avoid the L_2 norm calculation of gradient matrices, thereby reducing the computational cost. Experimental results indicate a 3.5% to 15.5% average recognition accuracy improvement using the proposed NDN method and a 9.3% average improvement when the NL of memristors is considered, as compared to the Original case that adds noise via software. Also, the scalability and endurance for the proposed system are considered. Consequently, the proposed method is an effective technique that provides a low-cost hardware solution for the notion of DP in memristor-based learning systems.

REFERENCES

[1] L. Chua, "Memristor-the missing circuit element," *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, Sep. 1971.
[2] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.
[3] B. Rajendran *et al.*, "Specifications of nanoscale devices and circuits for neuromorphic computational systems," *IEEE Trans. Electron Devices*, vol. 60, no. 1, pp. 246–253, Jan. 2013.
[4] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 603–618.
[5] K. Liu, C. Giannella, and H. Kargupta, "A survey of attack techniques on privacy-preserving data perturbation methods," in *Privacy Preserving Data Mining*. Boston, MA, USA: Springer, 2008, pp. 359–381.
[6] E. Hesamifard, H. Takabi, and M. Ghasemi, "CryptoDL: Deep neural networks over encrypted data," 2017. [Online]. Available: arXiv:1711.05189.
[7] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.*, 2006, pp. 265–284.
[8] C. Briggs, Z. Fan, and P. Andras, "A review of privacy-preserving federated learning for the Internet-of-Things," 2020. [Online]. Available: arXiv:2004.11794.
[9] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1310–1321.

[10] J. Fu, Z. Liao, and J. Wang, "Memristor-based neuromorphic hardware improvement for privacy-preserving ANN," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 27, no. 12, pp. 2745–2754, Dec. 2019.

[11] J. Fu, Z. Liao, N. Gong, and J. Wang, "Linear optimization for memristive device in neuromorphic hardware," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Miami, FL, USA, 2019, pp. 453–458.

[12] P.-Y. Chen *et al.*, "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Austin, TX, USA, 2015, pp. 194–199.

[13] J. Fu, Z. Liao, N. Gong, and J. Wang, "Mitigating nonlinear effect of memristive synaptic device for neuromorphic computing," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 377–387, Jun. 2019.

[14] C. Xu, J. Ren, L. She, Y. Zhang, Z. Qin, and K. Ren, "EdgeSanitizer: Locally differentially private deep inference at the edge for mobile data analytics," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5140–5151, Jun. 2019.

[15] P. C. M. Arachchige, P. Bertok, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman, "Local differential privacy for deep learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5827–5842, Jul. 2020.

[16] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar "Semi-supervised knowledge transfer for deep learning from private training data," 2016. [Online]. Available: arXiv:1610.05755.

[17] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 308–318.

[18] L. Yang and B. Murmann, "Approximate SRAM for energy-efficient, privacy-preserving convolutional neural networks," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Bochum, Germany, 2017, pp. 689–694.

[19] H. Jiang *et al.*, "A novel true random number generator based on a stochastic diffusive memristor," *Nat. Commun.*, vol. 8, no. 1, p. 882, 2017.

[20] M. Uddin, M. S. Hasan, and G. S. Rose, "On the theoretical analysis of memristor based true random number generator," in *Proc. Great Lakes Symp. VLSI*, 2019, pp. 21–26.

[21] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.

[22] C. Dwork, "Differential privacy," in *Encyclopedia of Cryptography and Security*. Heidelberg, Germany: Springer, pp. 338–340, 2011.

[23] R. Hasan, T. M. Taha, and C. Yakopcic, "On-chip training of memristor crossbar based multi-layer neural networks," *Microelectron. J.*, vol. 66, pp. 31–40, Aug. 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 *Proc. IEEE Int. Electron Devices Meeting*, San Francisco, CA, USA, 2017, pp. 1–4.

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

[26] S. Choi, P. Sheridan, and W. D. Lu, "Data clustering using memristor networks," *Sci. Rep.*, vol. 5, May 2015, Art. no. 10492.

[27] L. Goux *et al.*, "Understanding of the intrinsic characteristics and memory trade-offs of sub-μA filamentary RRAM operation," in *Proc. Symp. VLSI Technol.*, Kyoto, Japan, 2013, pp. T162–T163.

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

[29] J.-H. Lee, D.-H. Lim, H. Jeong, H. Ma, and L. Shi, "Exploring cycle-to-cycle and device-to-device variation tolerance in MLC storage-based neural network training," *IEEE Trans. Electron Devices*, vol. 66, no. 5, pp. 2172–2178, May 2019.

[30] T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, "Stochastic phase-change neurons," *Nat. Nanotechnol.*, vol. 11, pp. 693–699, May 2016.

[31] Y. Pang *et al.*, "25.2 a reconfigurable RRAM physically unclonable function utilizing post-process randomness source with$<6\times$ 10- 6 native bit error rate," in *Proc. IEEE Int. Solid-State Circuits Conf.*, San Francisco, CA, USA, 2019, pp. 402–404.

[32] Y. Gao, D.C. Ranasinghe, S. F. Al-Sarawi, O. Kavehei, and D. Abbott, "Memristive crypto primitive for building highly secure physical unclonable functions," *Sci. Rep.*, vol. 5, Aug. 2015, Art. no. 12785.

[33] C. Ye *et al.*, "Physical mechanism and performance factors of metal oxide based resistive switching memory: A review," *J. Mater. Sci. Technol.*, vol. 32, no. 1, pp. 1–11, 2016.

[34] Y. Gao, D. C. Ranasinghe, S. F. Al-Sarawi, O. Kavehei, and D. Abbott, "Emerging physical unclonable functions with nanotechnology," *IEEE Access*, vol. 4, pp. 61–80, 2016.

[35] Y. Gao *et al.*, "Efficient erasable PUFs from programmable logic and memristors," IACR Cryptol. ePrint Archive, Lyon, France, Rep. 2018/358, 2018.

[36] A. Cedilnik, K. Kosmelj, and A. Blejec, "The distribution of the ratio of jointly normal variables," *Metodoloski zvezki*, vol. 1, no. 1, pp. 99–108, 2004.

[37] D. S. Lemons and P. Langevin, *An Introduction to Stochastic Processes in Physics*. Baltimore, MD, USA: JHU Press, 2002.

[38] S. Kvatinsky *et al.*, "MAGIC—Memristor-aided logic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.

[39] Y. V. Pershin and M. Di Ventra, "Practical approach to programmable analog circuits with memristors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 8, pp. 1857–1864, Aug. 2010.

[40] S. Kvatinsky, A. Kolodny, U. C. Weiser, and E. G. Friedman, "Memristor-based IMPLY logic design procedure," in *Proc. 29th Int. Conf. Comput. Design*, Amherst, MA, USA, 2011, pp. 142–147.

[41] J. Rajendran, H. Manem, R. Karri, and G. S. Rose, "Memristor based programmable threshold logic array," in *Proc. IEEE/ACM Int. Symp. Nanoscale Archit.*, Anaheim, CA, USA, 2010, pp. 5–10.

[42] Y. N. Joglekar and S. J. Wolf, "The elusive memristor: Properties of basic electrical circuits," *Eur. J. Phys.*, vol. 30, no. 4, p. 661, 2009.

[43] Z. Biolek, D. Biolek, and V. Biolkova, "SPICE model of memristor with nonlinear dopant drift," *Radioengineering*, vol. 18, no. 2, pp. 210–214, 2009.

[44] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What can we learn privately?" *SIAM J. Comput.*, vol. 40, no. 3, pp. 793–826, 2011.

[45] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[46] S. Kim, M. Lim, Y. Kim, H. D. Kim, and S. J. Choi, "Impact of synaptic device variations on pattern recognition accuracy in a hardware neural network," *Sci. Rep.*, vol. 8, no. 1, pp. 1–7, 2018.

[47] S. Pi *et al.*, "Memristor crossbar arrays with 6-nm half-pitch and 2-nm critical dimension," *Nat. Nanotechnol.*, vol. 14, no. 1, pp. 35–39, 2019.

[48] L. Gao *et al.*, "Fully parallel write/read in resistive synaptic array for accelerating on-chip learning," *Nanotechnology* vol. 26, no. 45, 2015, Art. no. 455204.

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

[50] P. Yao *et al.*, "Face classification using electronic synapses," *Nat. Commun.*, vol. 8, May 2017, Art. no. 15199.

[51] C. Yakopcic, M. Z. Alom, and T. M. Taha, "Memristor crossbar deep network implementation based on a convolutional neural network," in *Proc. Int. Joint Conf. Neural Netw.*, Vancouver, BC, Canada, 2016, pp. 963–970.

[52] C. Li *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nat. Electron.*, vol. 1, no. 1, pp. 52–59, 2018.

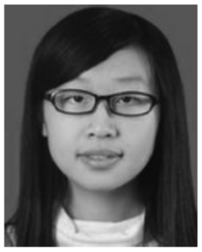
[53] D. J. Mannion, A. Mehonic, W. H. Ng, and A. J. Kenyon, "Memristor-based edge detection for spike encoded pixels," *Front. Neurosci.*, vol. 13, p. 1386, Jan. 2020.

[54] G. M. T. Xavier, F. G. Castañeda, L. M. F. Nava, and J. A. M. Cadenas, "Memristive recurrent neural network," *Neurocomputing*, vol. 273, pp. 281–295, Jan. 2018.

[55] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN+NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *Proc. IEEE Int. Electron Devices Meeting*, San Francisco, CA, USA, 2019, pp. 32–35.

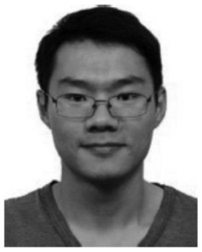
[56] C.-M. Jung, J.-M. Choi, and K.-S. Min, "Two-step write scheme for reducing sneak-path leakage in complementary memristor array," *IEEE Trans. Nanotechnol.*, vol. 11, no. 3, pp. 611–618, May 2012.

[57] L. Sun, N. Zheng, T. Zhang, and P. Mazumder, "Fault modeling and parallel testing for 1T1M memory array," *IEEE Trans. Nanotechnol.*, vol. 17, no. 3, pp. 437–451, May 2018.



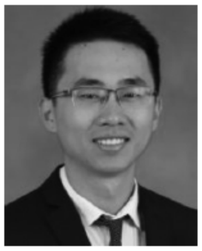
Jingyan Fu (Student Member, IEEE) received the B.E. and M.S. degrees in electrical engineering from Beijing University of Technology, Beijing, China, in 2014 and 2017, respectively. She is currently pursuing the Ph.D. degree with North Dakota State University, Fargo, ND, USA.

Her research focuses on hardware design and algorithm optimization for neuromorphic computing.



Zhiheng Liao received the B.E. and M.S. degrees in electrical engineering from Beijing University of Technology, Beijing, China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with North Dakota State University, Fargo, ND, USA.

His research focuses on emerging device and algorithm optimization for neuromorphic computing.



Jianqing Liu (Member, IEEE) received the B.E. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2013, and the Ph.D. degree in computer engineering from the University of Florida, Gainesville, FL, USA, in 2018.

He is currently a tenure-track Assistant Professor with the Department of Electrical and Computer Engineering, University of Alabama in Huntsville, Huntsville, AL, USA. His research interests are to apply cryptography, differential privacy, and convex optimization to design secure and efficient protocols for various IoT systems.

Dr. Liu is a recipient of three best paper awards, including the Best Journal Paper Award from IEEE Technical Committee on Green Communications and Computing in 2018.

Scott C. Smith (Senior Member, IEEE) received the B.S. degree in electrical engineering and computer engineering and the M.S. degree in electrical engineering from the University of Missouri, Columbia, MO, USA, in 1996 and 1998, respectively, and the Ph.D. degree in computer engineering from the University of Central Florida, Orlando, FL, USA, in 2001.

He is a Professor and the Department Chair of Electrical Engineering and Computer Science, Texas A&M University-Kingsville, Kingsville, TX, USA. He has published over 100 refereed journal/conference papers, 8 U.S. patents, two books, and four additional book chapters. His research interests include asynchronous logic, NULL convention logic, computer architecture, embedded systems, digital logic, FPGAs, CAD tools for digital design, computer arithmetic, VHDL, VLSI, secure/trustable hardware, wireless sensor networks, robotics, and CPS/IoT.

Prof. Smith is a member of the National Academy of Inventors, Sigma Xi, IEEE-HKN, and Tau Beta Pi.

Jinhui Wang (Senior Member, IEEE) received the B.E. degree in electrical engineering from Hebei University, Hebei, China, and the Ph.D. degree in electrical engineering from Beijing University of Technology, Beijing, China.

He was a Postdoctoral Fellow with the University of Rochester, Rochester, NY, USA, a Visiting Professor with the State University of New York at Buffalo, Buffalo, NY, USA, and a Visiting Scholar with IMEC, Leuven, Belgium. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of South Alabama, Mobile, AL, USA. He has published over 130 refereed journal/conference papers and 20 patents in the area of emerging semiconductor technologies. His research interests include neuromorphic computing, low power, high performance, variation-tolerant IC design, 3-D IC, and thermal solution in VLSI.