Variation Enhanced Attacks Against RRAM-Based Neuromorphic Computing System

Hao Lv, *Member, IEEE*, Bing Li, *Member, IEEE*, Lei Zhang, *Member, IEEE*, Cheng Liu, *Member, IEEE*, and Ying Wang, *Member, IEEE*

***Abstract*—The RRAM-based neuromorphic computing system (NCS) has amassed explosive interests for its superior data processing capability and energy efficiency than traditional architectures, and thus being widely used in many data-centric applications. The reliability and security issues of the NCS, therefore, become an essential problem. In this article, we systematically investigated the adversarial threats to the RRAM-based NCS and observed that the RRAM hardware feature can be leveraged to strengthen the attack effect, which has not been granted sufficient attention by previous algorithmic attack methods. Thus, we proposed two types of hardware-aware attack methods with respect to different attack scenarios and objectives. The first is an adversarial attack, VADER, which perturbs the input samples to mislead the prediction of neural networks. The second is fault injection attack, EFI, which perturbs the network parameter space such that a specified sample will be classified to a target label, while maintaining the prediction accuracy on other samples. Both attack methods leverage the RRAM properties to improve the performance compared with the conventional attack methods. Experimental results show that our hardware-aware attack methods can achieve nearly 100% attack success rate with extremely low operational cost, while maintaining the attack stealthiness.**

***Index Terms*—Adversarial attack, fault injection attack, neuromorphic computing system (NCS), processing in memory, reliability, resistive memory.**

I. Introduction

THE NEUROMORPHIC computing system (NCS) has attracted extensive interests as the traditional Von-Neumann architecture based on the CMOS technology is approaching the physical limit and facing the challenges of the well-known "memory wall." Recent advancements in the neuromorphic algorithms [i.e., deep neural networks, (DNNs)] have achieved tremendous success in the computing vision ([1], [2], [3]) and natural language processing domains ([4], [5]), driving the development of the NCS toward the ultimate goal of emulating the biologic neural networks. Unfortunately, the NCS based on the traditional digital hardware suffers the degraded performance and power efficiency because it is impractical to store the synaptic weights of the deep neural models in the on-chip memory.

Therefore, the emerging nonvolatile memory (eNVM) to implement the NCS has received tremendous attention due to the merits of representing the synaptic weight with the cell resistance instead of the electronic charges ([6], [7]). Amongst the candidate eNVM technologies, the resistive random access memory (RRAM), also known as memristor, is one of the most promising ones as the natural similarity between the programmable resistance and the variable biological synaptic strengths. More importantly, an RRAM crossbar is a natural dot-product engine that can process the most essential operations for the neural networks in a highly efficient way ([8]). Recent works have reported system-level RRAM-based NCS platforms ([9], [10], [11], [12], [13]), demonstrating the powerful capability and flexibility.

On the other hand, the security of the NCS raises a concern as the DNN models are spreading into the safety-sensitive scenarios, and previous studies have demonstrated that the DNN models are vulnerable to the well-designed attacks ([14], [15], [16], [17]). To fool the neural networks, these attack methods either perturb the input samples (i.e., adversarial attack [15], [18]) or manipulate the network parameters (i.e., fault injection [16], [17]). By distorting the selected input or network parameters, the attackers can mislead the network to make an erroneous prediction.

We observed that the inherent variation of RRAM-based NCS could be a potential hardware-level threat to the security of the neural network (detailed in Section V-B.), which motivates us to design hardware-aware attacks that incorporate the hardware information to improve the attack performance. As a type of nanoscale device, RRAMs suffer from nonideal properties, such as resistance fluctuation, resistance drift, and random noise ([19], [20]). Though prior works ([20], [21], [22]) have been proposed to mitigate the negative effect of these hardware variations to ensure the reliability of the RRAM-based NCS, it will be a different story when considering the security of RRAM-based NCS in the presence of the RRAM variation.

In this work, we develop two attack schemes for the RRAM-based NCS by leveraging the RRAM variation issue

Manuscript received 22 January 2022; revised 19 July 2022; accepted 4 September 2022. Date of publication 16 September 2022; date of current version 21 April 2023. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61874124, Grant 62204164, and Grant 62222411; in part by the Zhejiang Lab under Grant 2021PC0AC01; and in part by the Beijing Natural Science Foundation under Grant 4194092. This article was recommended by Associate Editor D. Atienza. *(Hao Lv and Bing Li contributed equally to this work.) (Corresponding author: Ying Wang.)*

Hao Lv and Lei Zhang are with the Institute of Computing Technology, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing 100089, China (e-mail: lvhao21b@ict.ac.cn; zlei@ict.ac.cn).

Bing Li is with the Academy of Multidisciplinary Studies, Capital Normal University, Beijing 100048, China (e-mail: bing.li@cnu.edu.cn).

Cheng Liu and Ying Wang are with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100089, China (e-mail: wangying2009@ict.ac.cn; liucheng@ict.ac.cn).

Digital Object Identifier 10.1109/TCAD.2022.3207316

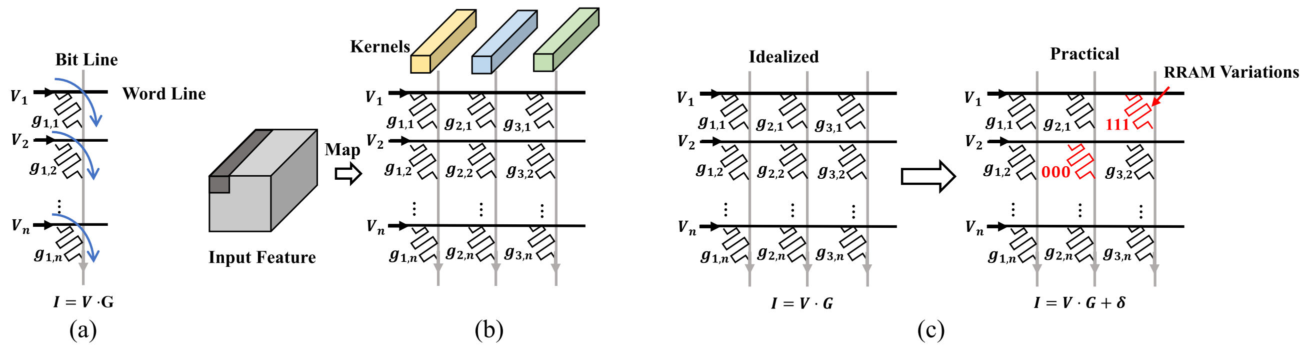


Fig. 1. (a) RRAM-based analog dot-product computing. (b) Example for accelerating the neural networks. (c) Simple example for the RRAM variations.

and the programmable cell resistance corresponding to the synapse weight. To the best of our knowledge, this is the first work that comprehensively investigates the security of the RRAM-based NCS.

Our contribution can be summarized as follows.

1) We observed that the intrinsic variations in the RRAM pose a potential threat for the security of RRAM-based NCS, and proposed two powerful attack methods, VADER and EFI, by exploiting the hardware properties of the RRAM. In specific, the VADER focuses on poisoning the input samples while EFI targets distorting the DNN parameters to launch the effective attack on the RRAM-based NCS.
2) In the VADER, we propose a *variation amplification* algorithm to locate and perturb the variation-sensitive pixels in the input sample, which can maximize the impact of the RRAM variation on the computing results of the RRAM-based NCS. In the end, VADER effectively misleads RRAM-based NCS and makes it output the prediction error.
3) In the EFI, we propose a *greedy victim parameter selection* algorithm to select and distort the victim parameters. By exploiting the RRAM variation, EFI can minimize the required victim parameters for a successful fault injection attack, significantly saving the operational cost.
4) We perform comprehensive experiments to evaluate the stealthiness, effectiveness, and efficiency of our proposed attack methods on the RRAM-based NCS and compare our work with the classical attack methods ([15], [16], [17]). The results show VADER and VADER are more effective than the comparison counterparts and both achieve almost 100% attack success rate. Besides, EFI can save orders of the cost (>95%) for the fault injection, while maintaining less accuracy degradation compared with previous fault injection attacks ([16], [17]).

The remainder of this article is organized as follows. In Section II, we introduce the background of the RRAM-based computing system, an overview of RRAM variation, and discuss the security concerns of the computing system. Sections III and IV present our proposed attack methods for different attack scenarios in detail. Experimental results and discussion of the defense techniques for the proposed attacks are provided in Section V. Finally, Section VI concludes this article.

II. Background and Related Work

A. RRAM-Based Neuromorphic Computing System

Features by the nonvolatility and programmable resistance of the device, the RRAM has been studied for establishing the NCSs ([6], [7]). Besides, the RRAM-based NCS relies on the crossbar structure to mimic the heart operation, i.e., matrix-vector multiplication (MVM), in neural networks, which is expressed as $I = V \cdot G$ in the analog manner. As illustrated in Fig. 1(a), the input voltage vector is applied on the word line (WL), the conductance values of the RRAM cells G representing the weight matrix. The current I through each bit line (BL) is naturally the multiplication result $V \cdot G$ according to Kirchhoff's and Ohm's law ([23]). In this way, the RRAM crossbar structure realizes MVM calculation with a time complexity of only $O(1)$. Fig. 1(b) shows an exemplary implementation of convolutional NNs (CNNs) on the RRAM crossbar by programming the convolutional kernels on the BLs of the crossbar and unfolding the feature maps into the input vector. Extensive researches have explored various RRAM-based NCS designs for a variety of neuromorphic computing models applied in different application fields ([9], [10], [11], [13]).

B. RRAM Variation

As an emerging nanoscale device, the RRAM suffers from a variety of variation issues, such as manufacturing defect, resistance variation, and random noise due to the immature manufacture and the intrinsic nature ([20]). For example, the defect cells in an RRAM crossbar stuck at a certain resistance state and are not changeable ([22]), while some RRAM devices' readout resistance is slightly biased from the programmed one affected by the resistance variation ([20]). Therefore, the network model mapped in the practical RRAM-based NCS is different from the algorithmic model, and will induce subtle computational error as Fig. 1(c) shows. For simplicity, we denote the cells with significant variation issues as faults in this work. Extensive studies have investigated the effect of RRAM variation issues on the reliability of RRAM-based NCS and proposed various strategies to alleviate this problem. Kannan *et al.* [19] developed an efficient test scheme for detecting and locating the position of faults in the RRAM array. With the fault distribution of the RRAM array, Xia *et al.* [24] proposed the fault-resilient training strategy to rescue the accuracy of the neural network models by exploiting the intrinsic error resilience of the neural network. Liu *et al.* [21] remapped the significant weight neural network model to the RRAM-based architecture by introducing the redundant hardware. These techniques ensure robust and practical neural network deployment on RRAM.

Although these fault-tolerance techniques enable robust neural network deployment on RRAM-based architectures in practical scenarios, the hardware variations are inevitable and still exist in these hardware platforms, making the RRAM-based NCS vulnerable to malicious attacks in safety-sensitive scenarios.

C. Security Concerns

As NN-based solutions becoming popular for many applications, NN security arises as a major concern for the practical deployment of NNs in safety-critical tasks. Lots of efforts have been devoted to investigating the security of NNs, especially, the malicious attack approaches ([15], [16], [17]). The attack object of these attacks is either the input sample (i.e., adversarial attack) or the neural network weights (i.e., fault injection attack). The adversarial attack generates adversarial examples by adding invisible perturbations on the input images to fool the neural network, and the fault injection attack poisons the neural network weights via fault injection ways. In addition, both categories of attack methods are purely software-level attacks and are white-box attacks, that is, the attacker is assumed to have the full knowledge of the network model (e.g., model architecture and model parameters, and the information of the hardware platforms to inject faults).

In general, a successful attack shall satisfy the three criteria: 1) efficiency; 2) effectiveness; and 3) stealthiness.

1) *Efficiency:* The attack shall be easy to launch without introducing too much overhead during the attack process. The adversarial attack is naturally efficient than the fault injection attack since it is easy to add perturbations on the input sample.
2) *Effectiveness:* The attack shall effectively fool the neural network and also be hard to resist. As the fault injection attack directly distorts the model parameters, it can effectively manipulate the prediction results of neural networks.
3) *Stealthiness:* The attacker shall maintain the model accuracy for regular images other than the targeted one. For the adversarial attack, the perturbations added on the input should be human imperceptible to ensure the attack stealthiness. The fault injection attack should ensure the model accuracy after fault injection.

The adversaries have various demands on these metrics based on the attack scenarios. For instance, the less experienced adversaries can adopt the adversarial attacks, since it is easy to access and perturb the input samples with sacrificed effectiveness. While the adversaries with expert knowledge concern the effectiveness and stealthiness of the attack, they can manipulate the network prediction for a given image at will through fault injection techniques while ensuring model accuracy on other images. Previous studies have demonstrated that it is practical to launch precise and effective fault injection attacks on neural networks [16], [17].

In summary, the adversarial attacks focus on the attack stealthiness and efficiency and the fault injection attacks pursue the attack stealthiness and effectiveness. Here, we briefly introduce the representative works on the adversarial attack and fault injection attack, respectively.

1) Adversarial Attack: There is now a sizable body of works proposing various adversarial attacks which explore different methods to generate effective perturbations [14], [15], [18]. The most representative and effective adversarial attack methods are gradient-based, such as PGD ([15]), FGSM ([14]), and C&W ([25]). These attack methods utilize the network gradients with respect to the input to craft the perturbations. For example, the PGD attack iteratively updates the given image in the gradient ascent direction to craft the adversarial examples, and make use of l_1 norm to measure the image distortion.

2) Fault Injection Attack: Previous works have investigated different ways to locate and perturb the victim parameters to ensure the stealthiness and efficiency of the fault injection attack [16], [17]. Single bias attack (SBA) and gradient descent attack (GDA) are proposed for different attack scenarios and objectives in [16]. The SBA only modifies one neuron bias, since the output is linearly dependent on the bias term, to achieve high attack efficiency with relaxed attack stealthiness. While GDA applies the gradient descent mechanism to modify the layer-wise parameters. In contrast to SBA, the GDA focus on attack stealthiness instead of attack efficiency. The fault sneaking attack ([17]) formulates the fault injection attack as an optimization problem with the constraint of accuracy degradation and parameter modifications, and applies ADMM (alternating direction method of multipliers) to obtain an analytical solution for the parameter modification. However, there are still a large number of parameter modifications in the fault sneaking attack, which leads to low attack efficiency. How to achieve high stealthiness and efficiency at the same time and realize a practical fault injection attack is the main pursuit of this work.

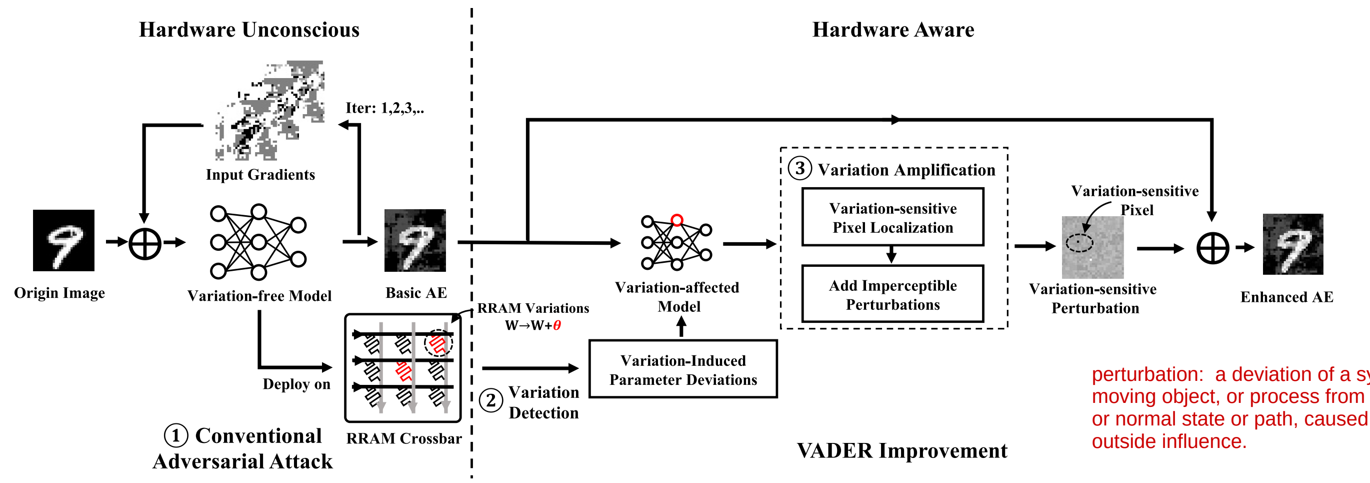
In this work, we incorporate the hardware information, i.e., intrinsic RRAM variations, to improve the performance (i.e., effectiveness, stealthiness, and efficiency) of the conventional adversarial attack and fault injection attack.

III. VADER: VARIATION-ORIENTED ADVERSARIAL ATTACK

In this section, we introduce our VADER whose principle is exploiting the RRAM variation to launch a powerful adversarial attack, such that VADER can even penetrate the adversarial defense, i.e., adversarial training ([14]). At the same time, the attack stealthiness is ensured.

A. Overview

The motivation behind our VADER is that the RRAM variation poses a stealthy security risk because of the variation-induced parameter deviations and the resulting subtle computing errors. VADER leverages the RRAM variation to


Hardware Unconscious
Hardware Aware
Input Gradients
Iter: 1,2,3,..
Origin Image
Variation-free Model
Basic AE
Deploy on
RRAM Crossbar
RRAM Variations W→W+θ
① Conventional Adversarial Attack
② Variation Detection
Variation-affected Model
Variation-Induced Parameter Deviations
③ Variation Amplification
Variation-sensitive Pixel Localization
Add Imperceptible Perturbations
Variation-sensitive Pixel
Variation-sensitive Perturbation
Enhanced AE
VADER Improvement

perturbation: a deviation of a system, moving object, or process from its regular or normal state or path, caused by an outside influence.

Fig. 2. Workflow of VADER. AE: Adversarial Example.

TABLE I
NOTATIONS USED IN THIS ARTICLE

Notations	Descriptions
W_i	The i-th layer's parameters of the variation-free model, $i = 1, 2...$
θ_i	The variation-induced parameter deviations in i-th layer, $i = 1, 2...$
x	The input sample.
$y_i(')$	The i-th layer's output of the variation-free (variation-affected) model, $i = 1, 2...$
f	The Activation functions.
$\mathcal{L}$	Loss function of the model.
$g(')$	The gradients of variation-free (variation-affected) model with respect to the input.
g_d	The gradients difference between the variation-free model and variation-affected model.

aggravate the computing errors from the RRAM-based NCS by adding variation-sensitive perturbation on the input sample. The deviation of the computing result will accumulate layer by layer and consequently cause an erroneous prediction for the perturbed input.

Attack Objective: The objective of VADER is to find a valid variation-sensitive perturbation for the given input sample to construct the enhanced adversarial example, such that the negative effect of the intrinsic and slight RRAM variation (e.g., parameter deviations and computing errors) can be amplified to disturb the network prediction result. Effectiveness and stealthiness are the main optimized goal of our VADER.

Notations: Before presenting our proposed attack methods, we summarize the notations used in this article in Table I. The well-trained model is referred to as variation-free before being deployed on RRAM and variation-affected after being deployed on RRAM, respectively. The ith layer's parameters of the variation-free model are denoted as W_i, and θ_i represents the variation-induced parameter deviations in the ith layer from the desired value (W_i) after mapping the variation-free model on RRAM computing systems.

The overall workflow of VADER is illustrated in Fig. 2. Given an input sample and a protected network model by the state-of-the-art defense mechanism (i.e., adversarial training), we first perform a *conventional adversarial attack* method (e.g., PGD [15]) to obtain the *basic adversarial example* (1). To be specific, we feed the specified sample into the model to compute the gradient, and add the gradient on the origin image. By repeating this operation several times, the basic adversarial example is obtained. The basic adversarial example is currently insufficient to deceive the protected network, so we take the following actions to enhance the adversarial example by exploiting the RRAM variation. Next, we can obtain the RRAM variation-induced parameter deviations (i.e., location of variation-affected parameters and their corresponding values, $W \rightarrow W + \theta$) through the *variation detection* step [19] (2), which utilizes the inherent sneak-paths to detect the faults and variations in crossbar memories. The testing method provides high fault coverage, and can locate the exact location of the abnormal RRAM cells. With the parameter deviations, we can derive the practical variation-affected model, and perform the *variation amplification* (detailed in Section III-B) on it to generate the *variation-sensitive perturbation* that can amplify the negative impact of the deviated network parameters on the computing results (3). Finally, we generated the enhanced adversarial example by adding the *basic adversarial example* with the *variation-sensitive perturbation*.

By being aware of the RRAM variation, the VADER can effectively disable the adversarial defense and mislead the network to produce an erroneous prediction.

B. Variation Amplification

The *variation amplification* stage locates the variation-sensitive pixels in the input image that can amplify the negative effects induced by the RRAM variation, and then determines the perturbation magnitude for these located variation-sensitive pixels.

The variation-sensitive perturbation generation procedure is described in Algorithm 1. The inputs of the algorithm

Algorithm 1 Variation-Sensitive Perturbation Generation

```
input : The basic adversarial example, $x_{adv}$;
        Variation affected network, $\hat{C}(W+\theta,\cdot)$;
        Learning rate, $lr$;
        Maximum number of pixels allowed be
        perturbed, $N_p$.
output: Variation-sensitive Perturbation $\delta_{vsp}$.
1  $\delta_{vsp} \leftarrow$ zero matrix of the same shape as $x_{adv}$
2  $S_p \leftarrow \{\}$    // $S_p$ is the coordinate set of
   variation-sensitive pixels.
3  repeat
4      Compute the gradients $g_d$ of input $x_{adv}+\delta_{vsp}$;
5      Select the pixel on the input with largest $g_d(i,j)$ as
       the variation-sensitive pixel;   // (i,j) is the
       pixel coordinate;
6      $N_p = N_p - 1$;
7      Insert $(i,j)$ into $S_p$;
8      repeat
9          $\delta_{vsp} \mathrel{+}= lr * sign(g_d)$;
10         $\delta_{vsp}(i,j) \leftarrow 0, for\ (i,j) \notin S_p$;
11         Compute the gradients $g_d$ of input $x_{adv}+\delta_{vsp}$;
12     until ((loss of $\hat{C}$ converges) or (the classifier $\hat{C}$ is
       mislead by $x_{adv}+\delta_{vsp}$));
13 until (($N_p > 0$) and (the classifier $\hat{C}$ is mislead by
   $x_{adv}+\delta_{vsp}$));
```

include the basic adversarial example x_{adv} from *conventional adversarial attack*, the network classifier $\hat{C}$, that is, deployed on the RRAM-based NCS, the learning rate lr, and the maximum number of pixels N_p allowed to be perturbed. The localization of the variation-sensitive pixels and the decision of their perturbation magnitude is implemented as the nested loop (lines 3–13). The outer loop first computes g_d (line 4), then locates the most variation-sensitive pixels by selecting the pixel with the largest g_d (line 5), and add it into the set of candidate variation-sensitive pixels S_p (line 6). Here, the g_d refers to the gradient difference between the variation-affected model (i.e., the network model on RRAM) and the variation-free model (i.e., the network model on GPU). The inner loop (lines 8–12) decides the value of these candidate pixels through a gradient ascent approach. Once the number of pixels in S_p reaches N_p or the prediction of the network is misled (line 13), Algorithm 1 will terminate, and the variation-sensitive perturbation (σ_{vsp}) is generated. The defined N_p is to limit the proportion of the perturbed pixels to the total number of input pixels such that the perturbation will not cause visible attention and thus ensure stealthiness.

In the following, we will show the effectiveness of g_d in indicating the variation-sensitive pixels of the input image with a running example.

Running Example: We take a two-layer fully connected network as an example to theoretically demonstrate the feasibility of the gradient difference g_d for locating the variation-sensitive pixels. The notations used in the following formulas can refer to Table I. The feedforward flow of the network can be formulated as follows:

$$y_1 = W_1 * x \tag{1}$$

$$y_2 = W_2 * f(y_1) \tag{2}$$

$$\text{loss} = \mathcal{L}(y_2). \tag{3}$$

The gradients (g and g') with respect to the input x can be calculated as follows:

$$g = \frac{\partial \mathcal{L}(y_2)}{\partial y_2} \times W_2^T \odot \frac{\partial f(y_1)}{y_1} \times W_1^T \tag{4}$$

$$g' = \frac{\partial \mathcal{L}(y_2')}{\partial y_2'} \times (W_2+\theta_2)^T \odot \frac{\partial f(y_1')}{y_1'} \times (W_1+\theta_1)^T. \tag{5}$$

Theoretically, the gradients of the input pixels with respect to the loss function quantitatively reflect the contribution of each input pixel on the loss value and the prediction result. On the other hand, since only few parameters are variation-affected, the input gradients of the variation-affected model g' are almost identical to g, to be specific, the parameter deviations θ_i are slight compared to the network parameters W_i, and thus the computation result of gradients is dominated by W_i, and, therefore, it is insufficient to distinguish the variation-sensitive pixels from the less sensitive pixels in the input sample by directly using g'. To locate the variation-sensitive pixels and amplify the impact of RRAM variation, we subtract the g from g' and the gradient difference g_d is obtained as (6), shown at the bottom of the page.

The outputs of the nonlinear activation layers (y_i and y_i') are nearly identical and that their derivatives ($[\partial\mathcal{L}(y_i')]/[\partial y_i']$ and $[\partial\mathcal{L}(y_i)]/[\partial y_i]$) can be viewed as being the same scalar tensor because the variation-resilient training strategy [24] is adopted to mitigate the negative impact of RRAM variations on neural networks (detailed in Section V-A). Since (6) only comprises the distributive operations of matrix multiplication and the Hadamard product, it can be approximated and simplified as (7), shown at the bottom of the page. Note that each term in (7) is scaled by the parameter deviations θ_i. As such, the gradient difference g_d not only reflects the impact of the input pixels on model predictions, but also indicates their variation sensitivity. To this end, the pixel with the largest g_d has the strongest capability to activate the negative effects of the RRAM variation, and distort the network predictions.

$$g_d = g' - g = \frac{\partial \mathcal{L}(y_2')}{\partial y_2'} \times (W_2+\theta_2)^T \odot \frac{\partial f(y_1')}{y_1'} \times (W_1+\theta_1)^T - \frac{\partial \mathcal{L}(y_2)}{\partial y_2} \times W_2^T \odot \frac{\partial f(y_1)}{y_1} \times W_1^T \tag{6}$$

$$g_d \approx \frac{\partial \mathcal{L}(y_2')}{\partial y_2'} \times \theta_2^T \odot \frac{\partial f(y_1')}{y_1'} \times W_1^T + \frac{\partial \mathcal{L}(y_2')}{\partial y_2'} \times W_2^T \odot \frac{\partial f(y_1')}{y_1'} \times \theta_1^T + \frac{\partial \mathcal{L}(y_2')}{\partial y_2'} \times \theta_2^T \odot \frac{\partial f(y_1')}{y_1'} \times \theta_1^T \tag{7}$$

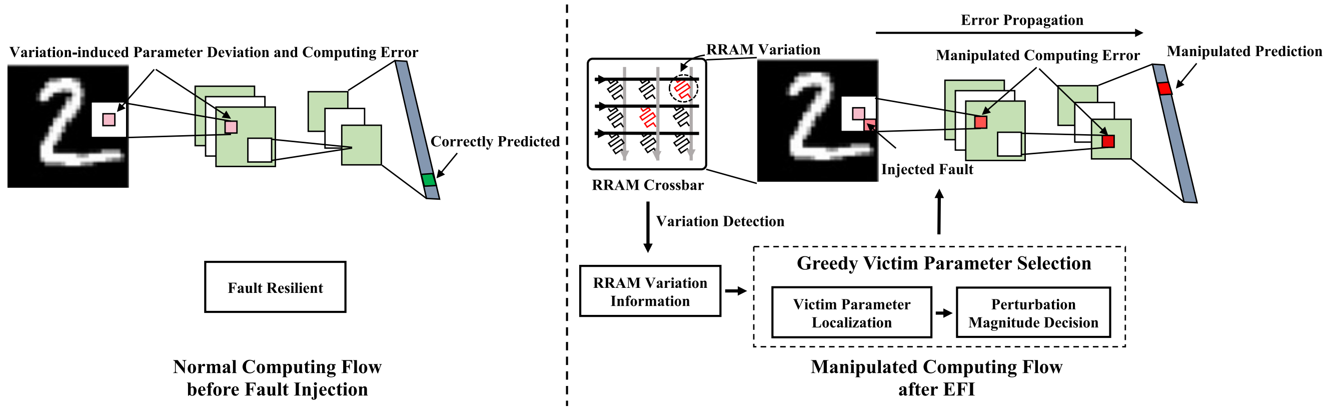

Variation-induced Parameter Deviation and Computing Error
Correctly Predicted
Fault Resilient
Normal Computing Flow before Fault Injection
Error Propagation
RRAM Variation
Manipulated Computing Error
Manipulated Prediction
RRAM Crossbar
Injected Fault
Variation Detection
RRAM Variation Information
Greedy Victim Parameter Selection
Victim Parameter Localization
Perturbation Magnitude Decision
Manipulated Computing Flow after EFI

Fig. 3. Workflow of EFI.

IV. EFI: Enhanced Fault Injection Attack

A. Overview

Considering the intrinsic RRAM variation, we develop a lightweight fault injection attack, EFI, which significantly reduces the cost for the engineer-expensive fault injection operation.

Attack Objective: The EFI, as the fault injection attack, primarily targets achieving high attack efficiency and stealthiness. For the attack efficiency, the EFI aims to minimize required network parameter modifications for a successful attack by leveraging the information of the RRAM intrinsic variations, and thus save the fault injection cost. As for the attack stealthiness, the EFI shall make the network model predict a specified label with the target input sample while outputting the correct prediction for other input samples.

The attack flow of EFI is illustrated in Fig. 3. Normally, the subtle variation-induced parameter and computing error can be tolerated by the error-resilience of neural networks and mitigated by the fault-resilient training strategy, and thus maintaining prediction accuracy. In the EFI, we inject faults into the target layers to incorporate with these existing RRAM variations to tamper the output feature map, and the distorted output will propagate forward and cooperate with the intrinsic RRAM variations and injected faults in the following layers to manipulate the prediction result. The overall workflow of EFI is similar to the attack paradigm of VADER. First, we acquire the information of variation-affected parameters through a *variation detection* process. Next, we perform *greedy victim parameter selection* (detailed in Section IV-B) to select and perturb the desired victim parameters. With these perturbed parameters, we can easily manipulate the computing flow of the variation-affected model, and finally change the prediction result to the target label.

B. Greedy Victim Parameter Selection

The *greedy victim parameter selection* realizes the objective that poisoning the minimal number of victim parameters by incorporating with the existed parameter deviation induced by the RRAM variations. Algorithm 2 presents the detailed flow of the victim parameter selection. The inputs of the algorithm include the specified input sample x_s, the target label y_t, the variation-affected network classifier $\hat{C}$, the list of target layers L to inject fault and the learning rate lr for these layers. The selection and perturbation for the victim parameters are also realized by two loops (lines 5–12). These two loops realize victim parameter localization and perturbation magnitude decision, respectively. In the first loop (lines 5–9), we first compute the gradient G_i for each target layer L_i (line 6), then, we apply the greedy strategy that selects the parameter with the largest gradient value in G_i as the victim parameter for each target layer (line 7). In the second loop (lines 10–12), we decide the perturbation magnitude of these victim parameters with gradient ascent. These two loops will be iterated until the specified input x_s is recognized as the target

Algorithm 2 Greedy Victim Parameter Selection

```
input : Specified input sample, $x_s$;
        Target label, $y_t$
        Variation-affected model, $\hat{C}(W+\theta,\cdot)$;
        List of target layers to inject faults, $L$
        Learning rate of each target layers, $lr$.
output: The set of victim parameters in the target layers, $vp$.
1  for $vp_i \in vp$ do
2      $vp_i \leftarrow \{\}$;
3  end
4  repeat
5      for $L_i \in L$ do
6          Compute the gradients $G_i$ of the target layers $L_i$ with input $x_s$;
7          Select the parameter $p$ with the largest gradient in $G_i$ as victim parameter;
8          Insert the victim parameter $p$ into $vp_i$.
9      end
10     for $vp_i \in vp$ do
11         update parameters in $vp_i$ in the gradient ascent direction with learning rate $lr_i$ until the loss of $\hat{C}$ converges;
12     end
13 until $(\hat{C}(W+\theta,x)=y_t)$;
```

TABLE II
PERFORMANCE COMPARISON OF MODELS ON DIFFERENT HARDWARE PLATFORMS. THE ACC AND ACC* DENOTE THE NETWORK ACCURACY ON THE TESTING EXAMPLES AND ADVERSARIAL EXAMPLES

Dataset	Network Architecture	Hardware Platform	Acc	Acc*
MNIST	LeNet	GPU	98.70%	88.85%
		RRAM	98.83%	87.67%
Cifar10	WRN	GPU	85.11%	49.68%
		RRAM	91.13%	30.90%

label y_t (line 13). Note that, the learning rate lr_i for different target layers should be carefully adjusted to avoid oversized modifications on the victim parameters and cause significant degradation in the model performance.

V. EVALUATIONS

A. Experimental Setup

Datasets and Network Architectures: We evaluate VADER and EFI with two different sizes of networks, i.e., LeNet ([26]) and wide residual networks (WRNs) ([27]) on two classification datasets, MNIST and Cifar10, and compare them with the SOTA attack approaches ([15], [16], [17]) in terms of attack effectiveness, efficiency, and stealthiness. The MNIST is a handwriting digit classification dataset, and Cifar10 is an RGB image classification dataset. Both datasets consist of ten exclusive categories. In the experiments, we adopt the variation-resilient training method to ensure the model robustness on RRAM, and both networks are protected with the adversarial training ([14]) from the adversarial attack.

Configurations: To precisely simulate the impact of RRAM variation on the neural networks, we modify the TensorFlow framework to model the network inference flow on the RRAM-based NCS, and the RRAM variation modeling (e.g., variation distribution and their magnitude) is referred from [20]. The evaluated models are quantized to 8-bit precision such that can be deployed on RRAM-based NCS. The hyperparameter N_p in Algorithm 1 is set to 10. The performance metrics are averaged over 10 000 random sampled images from the evaluation dataset.

B. Security Risk of Hardware Variations

In this section, we investigate the potential security risk of RRAM variations for the neural networks. Specifically, we evaluate the impact of RRAM variation on the network adversarial robustness and recognition capacity by comparing the accuracy of the variation-affected model and variation-free model on the clean testing images and adversarial examples. As Table II shows, with the variation-resilient training, the accuracy of the variation-affected models is comparable to, and even slightly higher than, the variation-free models, while there is a degradation in the resistance of the adversarial examples for the variation-affected model (1.08% and 18.78% on the MNIST and Cifar10, respectively). The results indicate that although the RRAM variations are harmless to the practical application of neural networks, they indeed impair the robustness of the model and pose a potential security threat to the neural network.

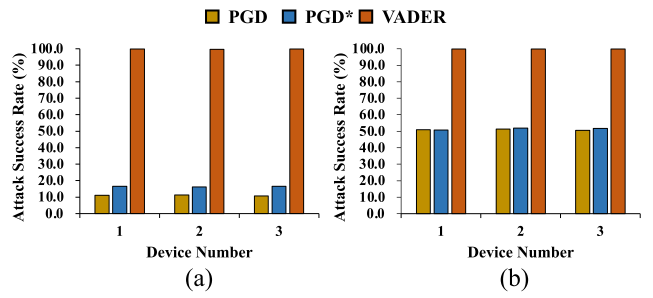


Fig. 4. Effectiveness evaluation of VADER. (a) and (b) are evaluated on MNIST and Cifar10, respectively.

C. Performance Evaluation

In the following paragraphs, we will systematically evaluate the effectiveness, efficiency, and stealthiness for our proposed VADER and EFI.

1) Evaluation for VADER: In this section, we evaluated the attack effectiveness and stealthiness of VADER on the MNIST and Cifar10 datasets and a variety of RRAM devices. Specifically, we use the attack success rate to measure attack effectiveness, and is defined as follows:

$$\text{Attack Success Rate (ASR)} = \frac{\text{\# of mis-classified examples}}{\text{\# of evaluated examples}}. \tag{8}$$

As results shown in Fig. 4, conventional adversarial attack, i.e., PGD is resisted by the adversarial defense mechanism, while VADER can achieve nearly 100% attack success rate, demonstrating its superior effectiveness. Furthermore, to analyze the benefit of the hardware knowledge in our VADER, we design an enhanced PGD, denoted as PGD*, which mimic the *variation amplification* step in Algorithm 1 by substituting the gradient difference g_a with the gradient of the variation-free model g. In contrast to VADER, PGD* is unaware of the hardware platforms and RRAM variation. As the results show, PGD* achieves slightly higher ASR than PGD, but is still incomparable to our VADER. Besides, VADER can maintain effective consistency on different RRAM devices with different variation distributions. These results indicate that the hardware information and localization of the variation-sensitive pixels in *variation amplification* step can improve the attack effectiveness.

VADER realizes the objective of a stealthy attack by adding human imperceptible perturbations on the input samples. Here, we visualize several enhanced adversarial examples from VADER in Fig. 5(a). From top to bottom, the images in each row are original clean images, adversarial examples from PGD, enhanced adversarial examples from VADER, and the difference between these two adversarial examples, respectively. As the figure shows, our VADER adversarial examples are visually indistinguishable from the PGD adversarial examples and even the original clean images, except for the several perturbed pixels. The perturbed pixels are highlighted by the red circles, which are too slight to cause human notice. This result guarantees the stealthiness of VADER.

Furthermore, to verify that VADER can satisfy our design principle (i.e., VADER is variation-oriented), we visualize several activation outputs of the network trained on MNIST in

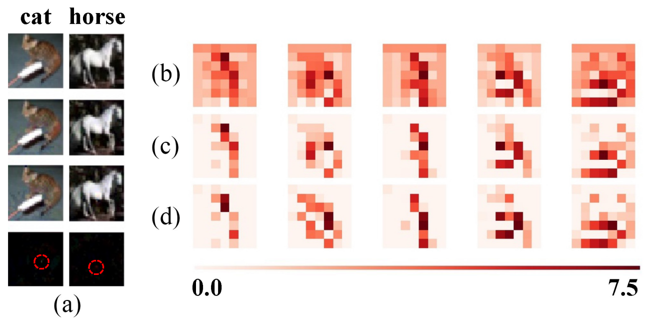


Fig. 5. (a) Visualization of VADER generated enhanced adversarial examples from Cifar10. (b)–(d) Visualizing samples of output feature maps from the final pooling layer.

Fig. 5(b)–(d). All these feature maps are sampled from the output of the last pooling layers. Respectively, Fig. 5(b) shows the feature maps from the variation-affected model with VADER adversarial examples as input. Fig. 5(c) shows the feature maps from the variation-free model with VADER adversarial examples as input. Fig. 5(d) shows the feature maps from the variation-free model with a clean test image as input. The feature maps in the last two rows are highly similar, while the feature maps in the first row are visually different from the other two rows. This observation indicates that the VADER can only interfere with the computing flow of the variation-affected model, and thus VADER is variation oriented. Besides, the visual difference suggests that VADER can satisfy our design principle of leveraging the RRAM variation to lead the activation output to deviate from the desired output and propagate the computational error forward. In summary, VADER can facilitate the RRAM variation to improve the attack effectiveness and realize a stealthy adversarial attack at the same time.

2) Evaluations for EFI: In this section, we evaluate the efficiency and stealthiness of EFI. We use the number of the modified network parameters as a proxy measure of the cost of a fault injection attack. To compute the fault injection cost, We randomly sampled 100 test images and labels, and averaged the number of parameter modifications requiredd for a successful EFI attack. We compare the EFI with the state-of-the-art fault injection attacks, including SBA, GDA from [16] and fault sneaking attack from [17]. The evaluation results are summarized in Table III. In contrast to the stealthy fault sneaking attack and GDA, EFI can save >95% fault injection operations, while maintaining higher classification accuracy after the fault injection attack. Compared with SBA, EFI improves the accuracy degradation by a significant margin. These results demonstrate that EFI can significantly improve the attack efficiency and stealthiness, and ensure a well balance between the two metrics.

To understand the EFI better, we perform EFI on the variation-affected model and the variation-free models to analyze the benefit from the hardware variations. As seen in Table III, EFI can not mislead the variation-free model. This indicates that our EFI is also a variation-oriented attack method, and can exploit these variations to reduce the engineering cost and improve the stealthy. Furthermore, we perform victim parameter selection on different layer combinations to investigate the impact of fault-injected layers.

TABLE III
PERFORMANCE EVALUATION OF EFI ON RRAM PLATFORM. THE AD AND MP ARE ABBREVIATION FOR ACCURACY DEGRADATION AND MODIFIED PARAMETERS

Attack Method	MNIST		Cifar10	
	AD	# of MP	AD	# of MP
SBA GDA Fault Sneaking	- -3.86% -0.80%	- 1170 >1026	-24.4% -2.35% -1.0%	1 198 >1026
EFI (on variation-free model)	failed		failed	
EFI (layer 1, 2) EFI (layer 3, 4)	-3.0% -0.50%	160 40	-4.1% -4.8%	30 68
EFI (last two layers)	-0.50%	40	-0.9%	20

The result shows that the fault injection on the later layers can achieve better stealthiness. The reason behind this observation may be that the latter network layers are closer to the prediction layer, and have a stronger correlation on the prediction results.

D. Discussion

In this section, we discuss and analyze the characteristic of VADER and EFI, and finally discuss the defense technique against them. The fundamental improvement of VADER over the conventional adversarial attack methods is that our VADER is hardware aware. Therefore, we attribute the success of VADER to the insufficiency of adversarial defense techniques. The adversarial defense techniques are purely software level, and unaware of the hardware variations. To protect the network model against our VADER, it is essential for the defense mechanism to be hardware aware. Thus, we suggest an on-device adversarial training defense method that protects the model by embedding hardware variation information into the adversarial training process, and experimental result shows the hardware-aware defense can resist VADER. Regrading the defense for EFI, we advise retaining a fault detection routine as [28], which periodically identifies potential fault injection attacks using a testing preserved dataset. If the tested predicted confidence score significantly deviates from the groundtruth confidence score in the testing set, it is likely that the tested model has suffered from a fault injection attack. After the fault injection attack is confirmed, we can compare the parameters on the device with the backup parameters and reprogram the different parameters to recover the injected faults.

VI. CONCLUSION

In this article, we revealed that the intrinsic RRAM variation poses a security risk for the RRAM-based NCS, and, therefore, propose two hardware-aware attack methods, VADER and EFI, which leverage the RRAM variation to improve the attack performance metrics, including effectiveness, efficiency, and stealthiness. The experimental results show that both improved attack methods can achieve almost 100% attack success rate with minimized operational cost, demonstrating superior performance than the conventional algorithmic attack methods. Besides, the two attack methods are orthogonal and can be combined to achieve better performance. Finally, we systematically analyze the proposed attack methods and discuss the feasible defense mechanism to eliminate the threat from VADER and EFI.

Finally, we systematically analyze the proposed attack methods and discuss the feasible defense mechanism to eliminate the threat from VADER and EFI.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, pp. 91–99, Dec. 2015.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[6] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, 2010.

[7] G. Indiveri, B. Linares-Barranco, R. Legenstein, G. Deligeorgis, and T. Prodromakis, "Integration of nanoscale memristor synapses in neuromorphic computing architectures," *Nanotechnology*, vol. 24, no. 38, 2013, Art. no. 384010.

[8] L. Xia *et al.*, "Switched by input: Power efficient structure for RRAM-based convolutional neural network," in *Proc. 53rd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2016, pp. 1–6.

[9] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 14–26, 2016.

[10] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 27–39, 2016.

[11] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined ReRAM-based accelerator for deep learning," in *Proc. IEEE Int. Symp. High Perform. Comput. Architect. (HPCA)*, 2017, pp. 541–552.

[12] B. Li, Y. Wang, and Y. Chen, "HitM: High-throughput ReRAM-based PIM for multi-modal neural networks," in *Proc. 39th Int. Conf. Comput.-Aided Design*, 2020, pp. 1–7.

[13] Z. Li, B. Li, Z. Fan, and H. Li, "RED: A ReRAM-based efficient accelerator for deconvolutional computation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 12, pp. 4736–4747, Dec. 2020.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[16] Y. Liu, L. Wei, B. Luo, and Q. Xu, "Fault injection attack on deep neural network," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2017, pp. 131–138.

[17] P. Zhao, S. Wang, C. Gongye, Y. Wang, Y. Fei, and X. Lin, "Fault sneaking attack: A stealthy framework for misleading deep neural networks," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, 2019, pp. 1–6.

[18] C. Szegedy *et al.*, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[19] S. Kannan, N. Karimi, R. Karri, and O. Sinanoglu, "Modeling, detection, and diagnosis of faults in multilevel memristor memories," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 5, pp. 822–834, May 2015.

[20] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," in *Proc. 56th Annu. Design Autom. Conf.*, 2019, p. 57.

[21] C. Liu, M. Hu, J. P. Strachan, and H. Li, "Rescuing memristor-based neuromorphic design with high defects," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2017, pp. 1–6.

[22] B. Li, B. Yan, C. Liu, and H. Li, "Build reliable and efficient neuromorphic design with memristor technology," in *Proc. 24th Asia South Pac. Design Autom. Conf.*, 2019, pp. 224–229.

[23] M. Hu *et al.*, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proc. 53rd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2016, pp. 1–6.

[24] L. Xia, M. Liu, X. Ning, K. Chakrabarty, and Y. Wang, "Fault-tolerant training with on-line fault detection for RRAM-based neural computing systems," in *Proc. 54th Annu. Design Autom. Conf.*, 2017, pp. 1–6.

[25] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2017, pp. 39–57.

[26] 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.

[27] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv preprint arXiv:1605.07146*.

[28] W. Li, Y. Wang, H. Li, and X. Li, "RRAMedy: Protecting ReRAM-based neural network from permanent and soft faults during its lifetime," in *Proc. IEEE 37th Int. Conf. Comput. Design (ICCD)*, 2019, pp. 91–99.

Hao Lv (Member, IEEE) is currently pursuing the Ph.D. degree with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.

His research interests include computer architecture, AutoML, and processing in memory.

Bing Li (Member, IEEE) received the B.S. degree from the Minzu University of China, Beijing, China, and the Ph.D. degree from the University of Chinese Academy of Sciences, Beijing, in 2016.

She was the Postdoctoral Fellow with Duke University, Durham, NC, USA, from 2017 to 2019. She is currently the Associate Professor with the Academy for Multidisciplinary Studies, Capital Normal University, Beijing. Her research interests include neuromorphic computing, computing-in-memory processing, emerging nonvolatile memory, and machine learning.

Lei Zhang (Member, IEEE) received the Ph.D. degree in computer architecture from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 2008.

He is currently an Associate Professor with the State Key Laboratory of Computer Architecture, ICT, CAS. His current research interests include network on-chip, 3-D integration, fault-tolerant computing, and multicore/many-core processors.

Cheng Liu, photograph and biography not available at the time of publication.

Ying Wang (Member, IEEE) is currently an Associate Professor with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. From 2011 to 2013, he was working with SAFARI Group, Carnegie Mellon University (CMU), Pittsburgh, PA, USA, as a Visiting Researcher. At ICT and CMU, his research interests primarily focus on the area of reliable computer architecture and VLSI design, with an emphasis on memory systems, energy-efficient accelerators, and approximate/error-tolerant computing.