T-BFA: Targeted Bit-Flip Adversarial Weight Attack

Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti and Deliang Fan

Abstract—Traditional Deep Neural Network (DNN) security is mostly related to the well-known adversarial input example attack. Recently, another dimension of adversarial attack, namely, attack on DNN weight parameters, has been shown to be very powerful. As a representative one, the Bit-Flip-based adversarial weight Attack (BFA) injects an extremely small amount of faults into weight parameters to hijack the executing DNN function. Prior works of BFA focus on *un-targeted* attack that can hack all inputs into a random output class by flipping a very small number of weight bits stored in computer memory. This paper proposes the first work of *targeted* BFA based (T-BFA) adversarial weight attack on DNNs, which can intentionally mislead selected inputs to a target output class. The objective is achieved by identifying the weight bits that are highly associated with classification of a targeted output through a *class-dependent vulnerable weight bit searching* algorithm. Our proposed T-BFA performance is successfully demonstrated on multiple DNN architectures for image classification tasks. For example, by merely flipping 27 out of 88 million weight bits of ResNet-18, our T-BFA can misclassify all the images from 'Hen' class into 'Goose' class (i.e., 100% attack success rate) in ImageNet dataset, while maintaining 59.35% validation accuracy. Moreover, we successfully demonstrate our T-BFA attack in a real computer prototype system running DNN computation, with Ivy Bridge-based Intel i7 CPU and 8GB DDR3 memory.

Index Terms—Deep Learning, Security, Targeted Weight Attack, Bit-Flip

1 Introduction

In recent years, deep neural networks (DNNs) have achieved tremendous success in a wide variety of applications, including image classification [1], [2], speech recognition [3], [4] and machine translation [5], [6]. Unfortunately, DNN models are not secure and the vulnerability of DNN models has been exposed by [7], [8] in their works on adversarial input example attack.

Recently, adversarial weight attacks have also been added to the security challenge of DNN models due to the security concern of model leakage and malicious fault injection into the computer system. First, the DNN model running in a computer is not secure. Many advanced computer side-channel attacks [9], [10], [11], [12] have been shown to successfully extract DNN model parameters. Second, due to large size, DNN model integrity is difficult to guarantee in state-of-the-art performance-driven computing systems. In such systems, there are many different methods to inject a small amount of fault into the computing memory or path of DNN without alerting computing system. For example, several memory fault injection techniques, like Laser Beam Attack [13] or Row-Hammer Attack (RHA) [14], [15], can inject faults into a computer main memory (i.e., DRAM), causing severe threat to DNN computation.

Due to the existing vulnerabilities, as shown in fig. 1, several recent works have leveraged such memory fault injection techniques to inject very minor faults (probably a few bits of error) into computer main memory (i.e. DRAM) to slightly modify the stored DNN model, successfully hijacking running DNN function [16], [17], [18], [19]. From those works, a general conclusion is that weight quantized network is more robust than full precision (i.e. floating point number) version due to limited value range per weight, and requires more bit flips (i.e. fault injection) in memory [16], [18]. Even so, the memory bit-flip based adversarial untargeted weight attack (*BFA*) proposed by our prior works in [19] and [16] has been experimentally demonstrated to cause severe accuracy degradation of a fully-functional 8-bit quantized ResNet-18 on the ImageNet dataset to 0.1% with only 13 bit-flips (out of 93 million bits), in a real computer system. To summarize, BFA based adversarial weight attack is a real-world practical threat for DNN computing system. Comparing with adversarial input example attacks that require designing noise for each input separately, BFA based adversarial weight attack only needs to attack the model once to achieve the desired malicious behavior for all benign inputs.

Our prior *un-targeted* bit-flip based adversarial weight attacks in [16], [19] mainly focus on reducing the overall prediction accuracy to be as low as random guess. However, *targeted* adversarial attacks [20], [21] pose a greater threat for the following reasons: First, it gives the attacker precise control on the malicious objective and behavior. Second, a carefully crafted targeted adversarial attack objective can cause a devastating effect on the DNN output. For example, a targeted attack in self-driving car applications could cause a stop sign to be miss-classified as a high speed-limit sign, while keeping the accuracy of all other signs intact. It can be very stealthy and cause severe threat to DNN system security.

However, all existing targeted attacks in the adversarial weight attack domain either fail to perform the attack effectively i.e., requiring a large number of weight modifications [22], or target on a much more vulnerable full-precision DNN models [17], [18]. Note that, a general conclusion is that a DNN with full precision weights is easier to attack. For example, as demonstrated in [18], a DNN can be forced to malfunction by just flipping the exponent bit, resulting in exponential change of the corresponding

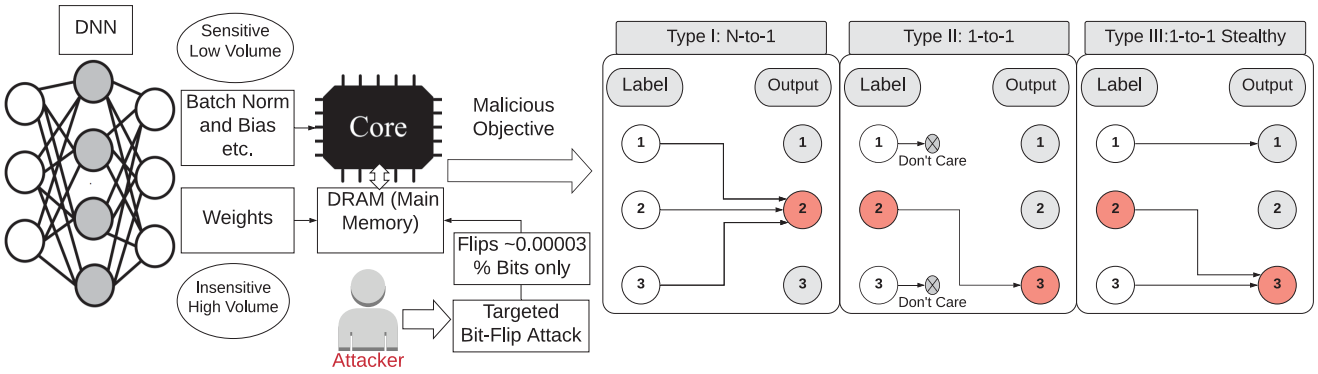


Fig. 1: Demonstration of Targeted Bit-Flip Attack (T-BFA) on the identified vulnerable bits achieving three distinct types of targeted attack objective.

weight value. In contrast, a DNN with quantized weights is naturally noise resilient. More importantly, weight quantization is becoming a must-optimization for optimal efficiency and speed in many computing platforms, such as Google's TPU [23]. Thus, in this work, we will only focus on attacking more robust quantized DNNs, rather than vulnerable full precision models.

This work is developed based on the authors' prior conference proceedings papers [16], [19], which mainly introduce un-targeted bit-flip attack algorithm and its implementation in a real computer system, respectively. Compared with them, the novelty of this work lies in that we propose a new *Targeted Bit-Flip Attack* (T-BFA) methodology. As far as we know, this is the first work of bit-flip based targeted adversarial weight attack on weight-quantized DNNs. We propose three variants of T-BFA as shown in the right panel of fig. 1: *N-to-1* attack where inputs from N source classes are hijacked to 1 target class; *1-to-1* attack where inputs from 1 source class are mis-classified into 1 target class; *1-to-1 stealthy* attack where not only inputs from 1 source class are hijacked to 1 target class, but also the other class classification function remains the same.

The novelty and contributions of this work are summarized as follows:

- Our proposed T-BFA is the first to demonstrate a successful targeted attack on noise resilient quantized DNNs through N-to-1 (I), 1-to-1 (II), and 1-to-1 stealthy (III) adversarial weight attacks by flipping a very small number of weight bits stored in computer memory.
- To achieve the desired targeted attack objectives, we formulate three distinct loss functions associated with each type of attack. We leverage an iterative searching algorithm that can successfully minimize these loss functions to locate most vulnerable weight bits that are associated with a adversary target class.
- We evaluate T-BFA on a wide range of network architectures (e.g., ResNet, VGG and MobileNet-V2) for image classification using CIFAR-10 and ImageNet datasets. For example, the experiment on ResNet-18 using ImageNet dataset show that our proposed T-BFA can achieve 100% attack success rate in miss-classifying all images in the 'Hen' class into 'Goose' class with only 27 bit-flips (out of 88 million bits) while keeping the test accuracy for other class images at 59.35%.
- Finally, we demonstrate the practical feasibility of T-BFA in real computer attacks considering the adversary performs the attack by running an unprivileged user-space process on the machine (i.e., a strong adversary with system-level access permission is not required).

We organize the rest of the paper in the following manner: In section 2, we present background information of BFA and related target attack. In section 3, we present the targeted attack type and optimization method. Then we present the experimental setup and results in sections 4 and 5 respectively. It is followed by discussion in section 6 and conclusion in section 7.

2 Background and Related Work

2.1 Adversarial Weight Attack

The recent developments in memory fault injection attacks [14], [24] have made it feasible to conduct an adversarial weight attack for a DNN model running in a computer. Among them, a row-hammer attack [14] on Dynamic Random Access Memory (DRAM) is the most popular one since it can create a profile of memory bits stored inside the main memory (i.e., DRAM) and flip any bit of a given target address. The first few works that exploited row-hammer to attack DNN weights flipped the Most Significant Bits (MSB bits) of DNN parameters, such as the bias [17] or weight [18], and changed them to a significantly large value, thus degrading accuracy. However, those attacks were only targeted on a model with full precision (i.e. floating point) parameters and failed in more noise-resilient weight-quantized DNNs.

A major milestone in adversarial weight attack is the work in [16] which implemented a stronger version of BFA on an 8-bit fixed-point quantized network. The method in [16] searches for the weight bits iteratively to gradually decrease DNN accuracy. However, the BFA design in [16] is for an un-targeted attack. Even though it succeeds in hampering overall test accuracy, it fails to degrade the accuracy of a targeted class. Next, we briefly introduce the BFA attack threat model and methodology.

TABLE 1: Threat model of Targeted Bit-Flip Attack (T-BFA), same as BFA [16], [19].

Access Required	Access **NOT** Required
DNN architecture & model parameters	Training configurations (i.e., hyper parameter).
A mini-batch of test data	Complete train/test datasets.

2.2 Threat Model

In this work, we follow the standard white-box attack threat model assumption, same as the previous bit-flip based adversarial weight attacks on quantized network [16], [19], [25]. The previous BFA attack [16] threat model is summarized in table 1. It assumes the attacker has access to model weights, gradients, and a portion of test data. Such a threat model is valid since previous works have demonstrated an attacker can effectively steal similar information (i.e., layer number, weight size, and parameters) through side-channel attacks [9], [10], [11], [12]. The attacker is denied access to any form of training information (i.e., training dataset, hyper-parameters). An attacker can only flip (0 to 1 or 1 to 0) identified bits in memory; no manipulation of input data is allowed.

2.3 Weight Quantization & Encoding

In our evaluation of T-BFA, we adopt a similar weight quantization scheme as BFA attack [16]. It is a layer-wise N-bits uniform quantizer for weight quantization. For each of the l-th layer, the quantization methodology can be described as:

$$\Delta w_l = \max(\ \mathbf{W}_l^r)/(2^{N-1}-1); \quad \mathbf{W}_l^{\mathrm{r}} \in \mathbb{R}^d \tag{1}$$

$$\mathbf{W}_l = \mathrm{round}(\mathbf{W}_l^{\mathrm{r}}/\Delta w_l) \cdot \Delta w_l \tag{2}$$

where d is the dimension of weight tensor, Δw_l is the step size of weight quantizer, $\mathbf{W}_l^{\mathrm{r}}$ is the full-precision weight of the corresponding quantized weight $\mathbf{W}_l$. To circumvent the non-differential function (in eq. (2)), popular straight-through estimator [26] is used to perform the training.

In our hardware evaluation, the computing system stores the signed integer in two's complement representation. Given one weight element $w \in \mathbf{W}_l$, the conversion from its binary representation ($\boldsymbol{b} = [b_{N-1}, ..., b_0] \in \{0,1\}^N$) in two's complement can be expressed as:

$$w/\Delta w = bin(\boldsymbol{b}) = -2^{N-1} \cdot b_{N-1} + \sum_{i=0}^{N-2} 2^i \cdot b_i \tag{3}$$

With the conversion relation described by $bin(\cdot)$ in eq. (3), we can inversely obtain the binary representation of weights $\mathbf{B}$ (i.e. binary data stored in main memory) from its fixed-point counterpart as well.

2.4 Un-Targeted BFA Attack Details

Our preliminary Bit-Flip Attack (BFA) utilizes a combination of gradient ranking and progressive search to identify a set of vulnerable weight bits [16], [19]. The objective of flipping the identified vulnerable weight bits is to degrade the overall test accuracy of the DNN. Thus at each iteration of the attack, the attacker will target at maximizing the inference loss function$\mathcal{L}$ w.r.t true label $\boldsymbol{t}$ of a given test batch $\boldsymbol{x}$:

$$\max_{\{\hat{\mathbf{B}}_l^i\}} \ \mathcal{L}\Big(f(\boldsymbol{x}; \{\hat{\mathbf{B}}_l^i\}_{l=1}^L), \boldsymbol{t}\Big) \tag{4}$$

Here $l \in \{1, 2, ..., L\}$ is the layer index, $\hat{\mathbf{B}}_l^i$ is the bit representation of the weight matrix at the i^{th} iteration at layer l after flipping the bits in the original matrix $\mathbf{B}_l$.

The progressive search of the BFA attack consists of two steps: i) *in-layer search* and ii) *cross-layer search*. Each of the step is performed progressively to identify vulnerable bits. First, for in-layer search, BFA will flip the top n ranked bits (e.g., typically n=1) based on the gradient ($\arg\max_{\mathbf{B}_l} |\nabla_{\mathbf{B}_l}\mathcal{L}|$) of every bit in each of the l DNN layers. After flipping these bits at a given layer, the attacker evaluates the loss $\mathcal{L}$, and restores these flipped bits to the original state. Thus, a loss profile set $\{\mathcal{L}^1, \mathcal{L}^2, \cdots, \mathcal{L}^P\}$ is generated. For the cross-layer search step, the attacker identifies the attack layer with maximum loss:

$$j = \arg\max_l \ \{\mathcal{L}^l\}_{l=1}^P \tag{5}$$

Finally, the attacker goes to layer j to perform the bit-flip in the i^{th} iteration. To evaluate the attack efficiency, [16] uses Hamming distance (i.e., effective bit-flips) between post attacked bits $\hat{\mathbf{B}}_l^i$ and prior attack bits $\mathbf{B}_l$ given by $\sum \mathcal{D}_{hd}(\hat{\mathbf{B}}_l^i, \mathbf{B}_l)$. In general, the optimization goal of un-targeted BFA is to lower the inference accuracy of DNN similar as random guess (i.e., 10 % for CIFAR-10) with least number of bit-flips (i.e., $\min \sum \mathcal{D}_{hd}(\hat{\mathbf{B}}_l^i, \mathbf{B}_l)$). Even though BFA is a successful un-targeted adversarial weight attack on popular DNN architecture(i.e., ResNet, MobileNet-V2); it still fails to attack a particular target class. In the next subsection, we highlight some of the early attempts to conduct targeted weight attacks on DNN.

2.5 Targeted Attack

In contrast to un-targeted BFA attack, a targeted attack has more precise control on the miss-classification behavior and can cause higher calamity. It is a well-investigated technique in adversarial input attack domain [7], [20], [21], where attacker finds additive input noise that decreases the loss function w.r.t a false target label for each input separately. Another form of targeted attack is the Trojan attack [27], [28], which typically requires hacking into the training supply chain for re-training network to force malicious behavior given a pre-designed input trigger [27], [28]. A recent more advanced Trojan attack also leverages BFA to inject Trojan by flipping close to one hundred bits with no need for network re-training or supply chain access [25]. However, it still needs the help of an input trigger, i.e. modifying inputs. This is out of the scope of this work, which limits injecting small errors to weight only. More closed related works are recent adversarial model parameter attacks that can perform

a targeted attack without requiring an input trigger [17], [22], although they either require large amounts of weight perturbation or are evaluated only on the more vulnerable full precision model, not noise-resilient quantized model.

3 Targeted Bit-Flip Adversarial Weight Attack

3.1 T-BFA Attack Objectives

In this section, we present the proposed *Targeted Bit-Flip adversarial weight Attack* (T-BFA) that results in misclassification of benign inputs from their source category/categories (i.e., ground-truth) to the adversary target category, via a small number of malicious bit-flips on the quantized weight-bits of pre-trained DNN models. As depicted in fig. 1, we propose three types of T-BFA with varying constraints, which are elaborated as follows:

- Type-I: N-to-1 Attack. Given that the input data belong to one of N-classes, the objective of this T-BFA variant is to force the entire dataset $\mathbb{X} = \{\mathbb{X}_i\}_{i=1}^{N}$ with all N classes (as source classes) to one adversary-selected target class. The objective function is formalized as:

$$\min\ \mathcal{L}_{\text{N-to-1}} = \min_{\{\mathbf{B}\}}\ \mathbb{E}_{\mathbb{X}}\mathcal{L}(f(\boldsymbol{x}, \{\mathbf{B}\}); \boldsymbol{t}_q) \tag{6}$$

  where $\{\mathbf{B}\}$ is the quantized representation (in binary format) of weight tensor $\{\mathbf{W}\}$ stored in computer memory. Given vectorized input $\boldsymbol{x} \in \mathbb{X}$, $f(\boldsymbol{x}, \{\mathbf{B}\})$ computes quantized DNN inference output. $\mathcal{L}(\cdot;\cdot)$ denotes the cross-entropy loss between DNN inference output and labels. $\boldsymbol{x}$ and $\boldsymbol{t}$ are input data and its corresponding ground-truth label. For this attack, the ground-truth label term of source category[1] $\boldsymbol{t} \in \boldsymbol{e}^{(i)}, i \in \{1, ..., N\}$ is tampered to the selected q-indexed target category $\boldsymbol{t}_q \in \boldsymbol{e}^{(q)}$.
- Type-II: 1-to-1 Attack. In this T-BFA variant, adversary focuses on the mis-classification of input data $\mathbb{X}_p$ of single p-indexed source category into the q-indexed target category ($p \neq q$), without caring about the impact on the remaining categories $\mathbb{X}_{i\neq p}$. It can be modeled as:

$$\min\ \mathcal{L}_{\text{1-to-1}} = \min_{\{\mathbf{B}\}}\ \mathbb{E}_{\mathbb{X}_p}\mathcal{L}(f(\boldsymbol{x}_p, \{\mathbf{B}\}); \boldsymbol{t}_q); \quad \boldsymbol{x}_p \in \mathbb{X}_p \tag{7}$$

  Type-II attack is a subset of Type I attack. But, such an objective is still practically useful to only attack a specific group or subset of inputs, where the type-I N-to-1 attack would flip many more un-necessary bits for all groups of inputs.
- Type-III: 1-to-1 Stealthy Attack. In addition to the type-II 1-to-1 attack described above, this type-III attack is a stealthy version with two objectives: **1)** All the input data from p-indexed category $\mathbb{X}_p$ are classified into q-indexed target category, which is the same as eq. (7); **2)** Meanwhile, it needs to maintain correct predictions of the input data excluded from the source category $\mathbb{X}_j, j \in \{1, 2, .., N\}\backslash\{p\}$. This type-III attack could be achieved via the optimization of the two corresponding loss terms in the RHS of the following objective function:

$$\min\ \mathcal{L}_{\text{1-to-1(S)}} = \min_{\{\mathbf{B}\}}\ \mathbb{E}_{\mathbb{X}}\Big(\mathcal{L}(f(\boldsymbol{x}, \{\mathbf{B}\}); \boldsymbol{t}_q) \cdot \mathbf{1}_{\boldsymbol{x}\in\mathbb{X}_p} + \mathcal{L}(f(\boldsymbol{x}, \{\mathbf{B}\}); \boldsymbol{t}) \cdot \mathbf{1}_{\boldsymbol{x}\in\mathbb{X}_j}\Big) \tag{8}$$

  where $\mathbf{1}_{\text{condition}}$ returns 1 if the condition is true, 0 otherwise.

For a practical adversarial attack, to minimize attack effort, a critical constraint is to use limited number of malicious bit-flips on weight bits to achieve above defined attack objectives in eqs. (6) to (8). This could be modeled as a joint-optimization and represented by:

$$\begin{aligned}\min\ \mathcal{L}_{\text{T-BFA}},\ \mathcal{L}_{\text{T-BFA}} \in \{\mathcal{L}_{\text{N-to-1}}, \mathcal{L}_{\text{1-to-1}}, \mathcal{L}_{\text{1-to-1(S)}}\};\\ \text{s.t. } \min_{\{\mathbf{B}\}}\ \mathcal{D}_{\text{hd}}(\{\hat{\mathbf{B}}\}, \{\mathbf{B}\});\end{aligned} \tag{9}$$

where $\mathcal{D}_{\text{hd}}$ is the Hamming-distance between the weight-bit tensors of pre-attack model ($\{\mathbf{B}\}$) and post-attack model($\{\hat{\mathbf{B}}\}$). Instead of applying $\mathcal{D}_{\text{hd}}$ as an additional loss term in eqs. (6) to (8) to form one combined multi-objective function, we follow the searching algorithm from our prior un-targeted BFA [16], with several T-BFA-specific modifications; the details are given in the following subsection.

3.2 Vulnerable Weight Bits Searching Algorithm of T-BFA

The search for the most vulnerable weight bits to be attacked by T-BFA can be generally described as an iterative process, wherein each iteration, only a single weight-bit is identified followed by the malicious bit-flip. In the k-th iteration, the objective function eq. (9) is rephrased as:

$$\min_{\{\mathbf{B}^k\}}\ \mathcal{L}_{\text{T-BFA}}; \quad \text{s.t. } \mathcal{D}_{\text{hd}}(\{\mathbf{B}^k\}, \{\mathbf{B}^{k-1}\}) = 1 \tag{10}$$

where the single bit-flip is highlighted by defining inter-iteration Hamming distance $\mathcal{D}_{\text{hd}}$ as 1. To minimize $\mathcal{L}_{\text{T-BFA}}$ with single bit-flip per iteration, we inherit and modify the progressive intra- and inter-layer bit search method in our prior un-targeted BFA scheme [16].

Given a DNN model with L layers (e.g., convolution layers), for one search iteration, the *intra-layer bit search* identifies one weight-bit per layer and traverses through all L layers, thus returning L weight-bit candidates. Then, the following *inter-layer search* identifies one winner weight-bit out of L weight-bit candidates brought up by the last step intra-layer search. This identified winner weight-bit will be flipped and the search process goes to the next iteration. The whole progressive search process ends when the adversary defined attack objective is achieved as shown in fig. 2. We will describe such two-step progressive searching method in one iteration-k in the following paragraphs.

Intra-layer Bit Search. For layer indexed by l, the intra-layer bit search identifies one(or more) weight-bit candidate(s) w.r.t two criteria: 1) identifying the weight-bit with the highest gradient; 2) flipping along the direction of bit-gradient. Note that, in our prior un-targeted BFA [16],

1. $\boldsymbol{e}^{(i)}$ is the notation of one-hot code vector $[0, \ldots, 0, 1, 0, \ldots, 0]$ with a 1 at position i.

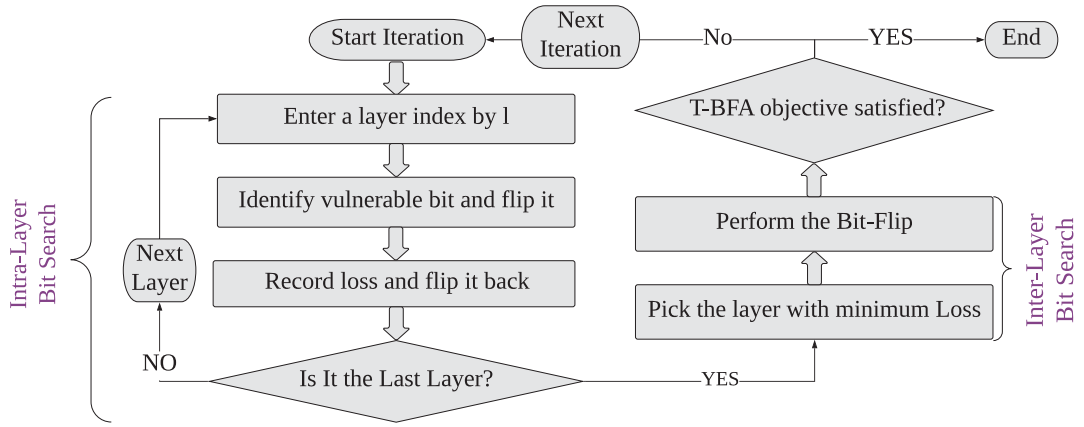


Fig. 2: Overview of T-BFA searching algorithm.

TABLE 2: Test data splitting to conduct targeted attack from source class t_{p} to target class t_{q}. CIFAR-10 data has 10k test images with each class containing 1000 test images and the ImageNet dataset has 50k test samples with each class containing 50 images. Note: (t_{r}) means images belong to any other class apart from the source class.

Metrics	Attack Batch Size	# of Data to evaluate ASR (X_{p})	# of Data to evaluate Test acc. (X_{r})	Attack Batch Size	# of Data to evaluate ASR (X_{r})	# of Data to evaluate Test acc. (X_{r})
Dataset	CIFAR-10			ImageNet		
N-to-1	128	10k	10k	50	50k	50k
1-to-1	500(t_{p})	500(t_{p})	9k	25(t_{p})	25(t_{p})	50k
1-to-1 (S)	500(t_{p})+500(t_{r})	500(t_{p})	8.5k	25(t_{p})+25 (t_{r})	25(t_{p})	50k

the weight-bit is flipped along the opposite direction of bit-gradient, as it performs loss maximization, instead of minimization defined in eq. (10) in this work. To perform the bit-flip, we adopt the same mask technique in [16] to check whether the chosen bit can be flipped in the desired direction. These two criteria can be mathematically described as:

$$\arg\max_{\mathbf{M}_l^k, b_l^k} |\nabla_{\mathbf{B}_l^{k-1}} \mathcal{L}^k_{\text{T-BFA}}|; \quad \text{s.t. } b_l^k = \tag{11}$$
$$\text{clamp}\Big(b_l^{k-1} - \text{sign}(\nabla_{b_l^{k-1}} \mathcal{L}^k_{\text{T-BFA}})\Big),\ b_l^k \neq b_l^{k-1}$$

where $\mathbf{M}_l^k$ is the mask that indicates the location of the identified bit within weight-bit tensor $\mathbf{B}_l^{k-1}$ and its value $b_l^k \in \{0, 1\}$. clamp$(\cdot)$ is the clamping function with 0 and 1 as lower and upper bound. The intra-layer bit search traverses through all the layers to generate the weight-bit candidate set, $\{\mathbf{M}_l^k\}_{l=1}^L$. Meanwhile, for each weight-bit candidate in $\{\mathbf{M}_l^k\}_{l=1}^L$, the corresponding T-BFA loss is profiled $\{\mathcal{L}^k_{\text{T-BFA},l}\}_{l=1}^L$ after the identified weight-bit is flipped.

Inter-layer Bit Search. Based on the intra-layer search outcomes (i.e., $\{\mathbf{M}_l^k\}_{l=1}^L$), the inter-layer search performs straight-forward comparison to identify the winner weight-bit candidate with minimal profiled loss as the weight-bit to attack in iteration-k. This process can be expressed as follows:

$$\arg\min\ \{\mathcal{L}^k_{\text{T-BFA},l}\}_{l=1}^L \tag{12}$$

When the winner weight-bit is identified, it will be flipped to perturb the DNN model with only one-bit difference with the model in the previous iteration. Then, another new search iteration will start with this new model parameters. The whole process ends when the attack goal is achieved.

4 Experimental Setup

4.1 Dataset

In our experiment, we test our T-BFA in image classification using two popular datasets i) CIFAR-10 [1] and ii) ImageNet [29]. CIFAR-10 is a popular visual recognition dataset, which includes 60k images combined with training and test set. Each RGB image has a size of 32×32 evenly sampled from 10 categories. The data augmentation technique is identical to previous methods [30]. ImageNet is a large dataset containing 1.2M training images. The size of the images of the ImageNet dataset is 224×224 that is equally divided into 1000 distinct classes.

4.2 Dataset configuration for attack

In table 2, we provide an overview of the data organization to conduct each type of attack. To conduct an N-to-1 attack on CIFAR-10 and ImageNet, we randomly choose a test batch from the test dataset. However, to evaluate 1-to-1 or 1-to-1(S) attack, we require a subset of source class (t_{p}) test images. Since the CIFAR-10 dataset has 1k images in each class, we use 500 images to perform the attack and the remaining 500 images for evaluating the *Attack Success Rate (ASR)*. Since the ImageNet dataset has only 50 images per class, we conduct the attack using 25 images from the source class and use the remaining 25 images for evaluating ASR. Similar to BFA [16] attack, we observe the effect of attack batch size plays a minor role in the attack performance. Furthermore, for ImageNet, we always evaluate test accuracy on the whole test dataset of 50k images because the amount of test data used to perform the attack (e.g., 50) is negligible compared to 50k test images. The mean

TABLE 3: *Pre-Attack test accuracy of individual class (i). We also report the test accuracy w/o any sample from class i for both ResNet-20 and VGG-11 model.*

	i =	0	1	2	3	4	5	6	7	8	9
Resnet-20	Test Accuracy(i)	92.9	97	89.8	81.5	93.7	87.3	94.3	92.7	95.5	94.7
	Test Accuracy (w/o i)	91.83	91.37	92.17	93.5	91.74	92.45	91.67	91.85	91.54	91.63
VGG-11	Test Accuracy(i)	91.6	94.4	89.1	86.7	89.8	82.8	92.9	93.3	94.5	92.1
	Test Accuracy (w/o i)	91.56	91.07	92.62	91.95	91.42	92.71	91.36	91.13	90.18	91.34

TABLE 4: N-to-1 Attack: number of bit-flips (mean±std) required to classify all the input images to a corresponding target class with 100% ASR. In each case, test accuracy drops to 10%.

Class	0	1	2	3	4	5	6	7	8	9	Average
ResNet-20	4.0 ± 0	4.6 ± 0.9	5.0 ± 2.2	6.2 ± 2.3	4.6 ± 0.9	5.2 ± 1.6	6.8 ± 1.9	4.4 ± 1.7	5 ± 2.2	4.8 ± 1.8	5.1
VGG-11	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.0	2.8 ± 0.4	2.0 ± 0.0	3.0 ± 0.0	3.2 ± 0.4	3.0 ± 0.0	3.0 ± 0.0	3.0

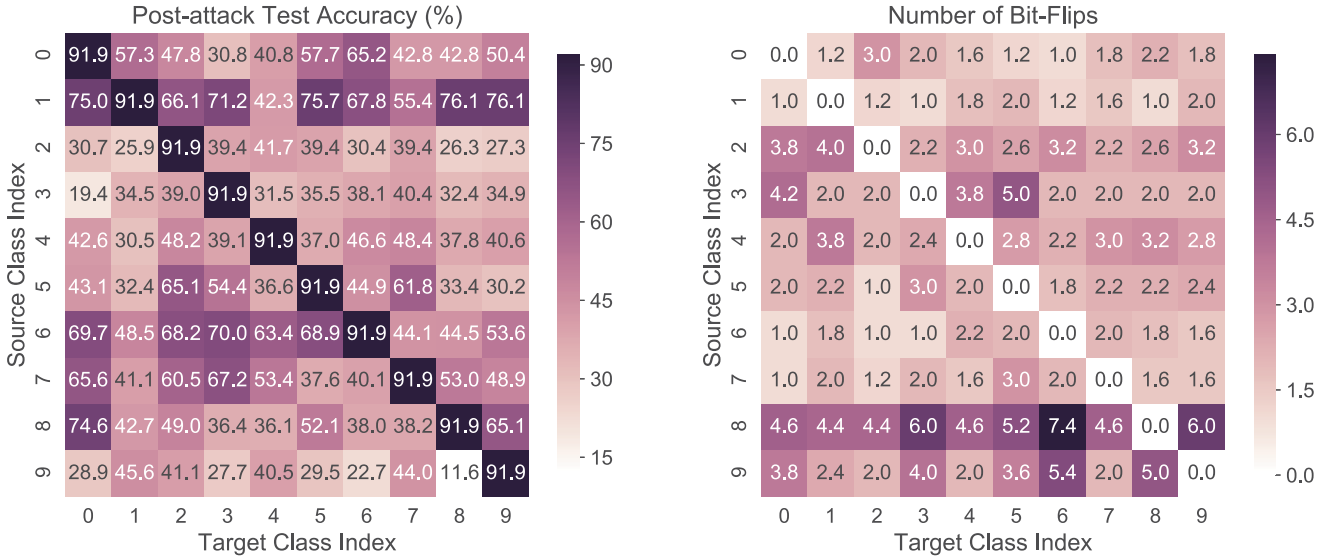


Fig. 3: Type II: 1-to-1 attack on ResNet-20 between source class and target class. The left subplot shows post attack test accuracy and the right subplot shows average number of bit-flips required for the attack.

and standard deviation numbers are calculated over 5 trial runs for CIFAR-10 and 3 trial runs for ImageNet. Also, we terminate attacks when the ASR reaches higher than 99.99% or remains the same for three successive iterations. Our code is publicly available online[2].

4.3 DNN Architectures

For CIFAR-10 dataset, we evaluate the attack against popular ResNet-20 [30] and VGG-11 [31] networks. We use the same pre-trained models with exact configuration as [32]. For ImageNet results, we evaluate our attack performance on MobileNetV2 [33], ResNet-18 and ResNet-34 [30] architectures. For each of the model, we directly download a pre-trained model from PyTorch Torchvision models [3] and perform an 8-bit post quantization and encoding as described in section 2.3.

4.4 T-BFA Attack Setup in a Real Computer

To demonstrate T-BFA attack on a DNN running in a real computer, we implement a DRAM fault injection method using the same computer system setup as our prior work in [19]. Here the adversary performs the row-hammer attack to inject bit flips in DRAM by running an unprivileged user-space process on the machine (i.e., a strong adversary with system-level access permission is *not* required). Our attack is evaluated on a computer with Intel Ivy Bridge-based processor and dual-channel 8GB DDR3 memory with two DIMMs. Each DRAM DIMM has 16 banks and each bank has 32768 rows. We implement a double-sided row-hammer attack where the attacker controls two neighboring rows of a victim row (i.e., rows that store DNN weights) to induce bit flip in the victim DNN model. To achieve such a memory layout, we reverse-engineer the DRAM addressing scheme using the technique demonstrated in [34].

We first perform memory templating that scans DRAM rows to collect information about flippable bits (i.e., bit flip profile) in the main memory. Such an off-line DRAM profiling can be done in isolation in the attacker's memory space, and thus does not corrupt or crash the system [13], [15]. We employ the stripe data pattern (1-0-1 and 0-1-0) with a double-sided row-hammer in order to extract most bit flips [14], [19]. The bit-flip profile keeps track of locations and flip directions for vulnerable memory cells.

After our T-BFA algorithm search is finished, the attacker generates a set of bit offsets in the target DNN's weight file.

2. https://github.com/ASU-ESIC-FAN-Lab/TBFA
3. https://pytorch.org/docs/stable/torchvision/models.html

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3112932, IEEE Transactions on Pattern Analysis and Machine Intelligence

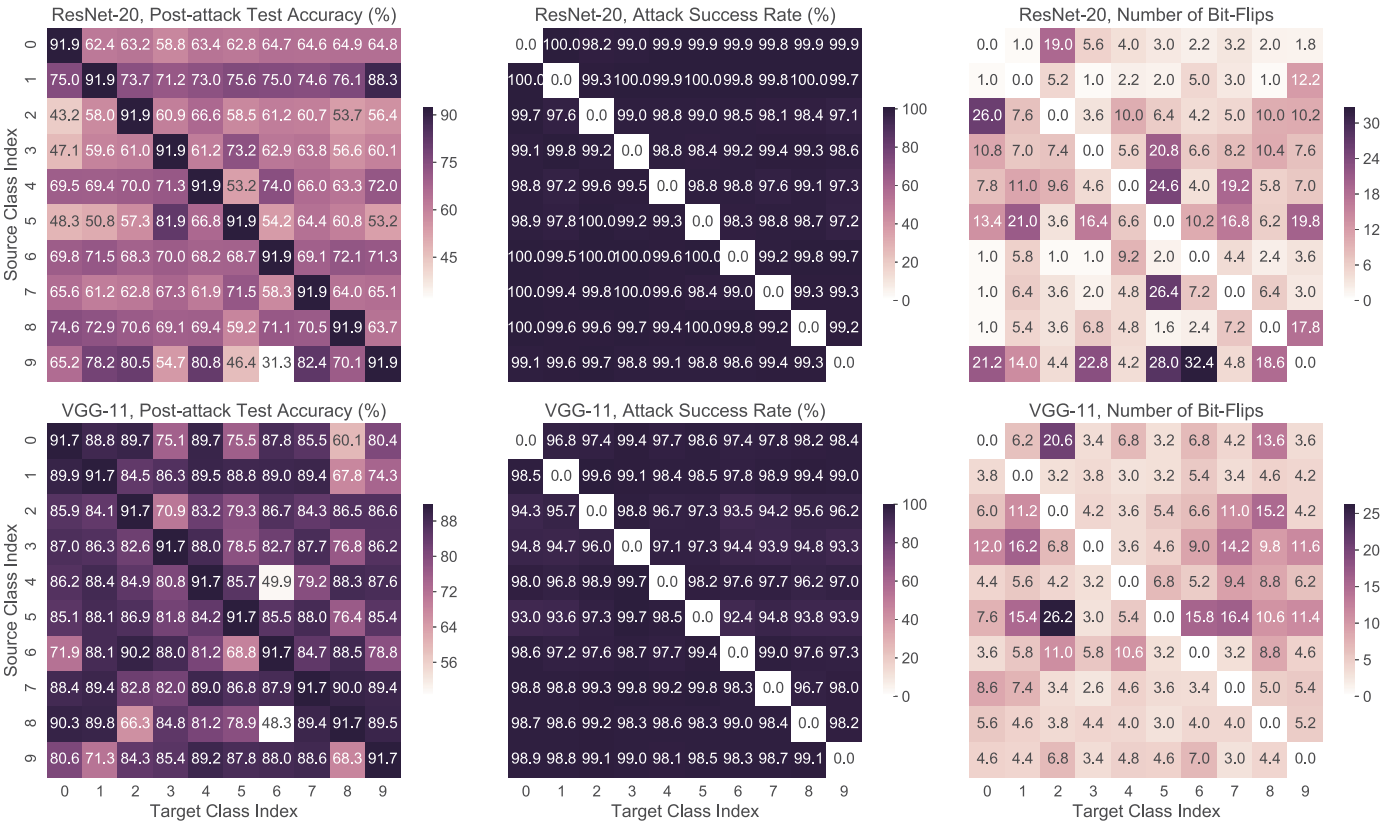


Fig. 4: Type III: 1-to-1 (S) attack post attack test accuracy, attack success rate and avg. # of bit-flips for five rounds of attacks for both Resnet-20 and VGG-11 Networks.

The weight parameters in the weight files are organized as physical pages (typically in the size of 4KB) in DRAM. To ensure that these identified target bits could be flipped, the attacker needs to ensure that DRAM pages holding the targeted weight parameters are located in the desirable DRAM rows. Particularly, the attacker manipulates the Operating System through page cache to *massage* the memory [35], [35] so that the target weight bits are stored in flippable DRAM cells with the right flip direction (i.e., either $1\rightarrow0$ or $0\rightarrow1$). The attacker then performs double-sided row-hammering through frequently accessing its own data (the neighboring rows) to incur sufficient disturbance to DNN's memory row to achieve the targeted bit flips. In some cases, if the identified bits are not flippable in hardware, a new set of vulnerable bit candidates from our T-BFA algorithm will be generated by freezing the previous set (e.g., in case that the bit flip found in the profile can not be repeated at runtime). This ensures our software algorithm runs independently of system attacks. In our real computer attack experiments, we successfully validate all types of T-BFA on different DNN architectures as will be reported later.

4.5 Evaluation Metrics

Two metrics are used in this work for attack evaluation: *Post-attack test accuracy (TA%)* and *Attack Success Rate (ASR%)*.

Post-Attack Test Accuracy (TA%): The Post-attack test accuracy is the inference accuracy of the post-attack model on test set. To evaluate the test accuracy after the attack, we only use a portion of the test data (X_{r} in table 2) which does not contain any image from the source class; since all the source class images will be miss-classified to the target class after the attack.

Attack Success Rate (ASR%): The ASR is the percentage of source class images(i.e., X_{p} in table 2) successfully classified into the adversary target class via T-BFA. To evaluate ASR, we only use X_{p} portion of source class data shown in table 2. The attacker does not use this portion of the source class images during the attack for 1-to-1 and 1-to-1 (S). However, for N-to-1 (S) X_{p} contains the whole test dataset, since, by definition, the attack should classify all the test images into one target class.

5 Experimental Results

5.1 Experiments on CIFAR-10

For CIFAR-10 experiments, we primarily evaluate our attack against ResNet-20 and the VGG-11 model. Before attacking the models, we present the test accuracy of all the classes in table 3. Overall, it shows that the test accuracy of each class i fluctuates slightly but the test accuracy without having any sample from class i (described as X_j samples in the manuscript) remains fairly stable at around ~ 91 %.

N-to-1 Attack. For CIFAR-10, the proposed N-to-1 attack can successfully reach 100% ASR for both VGG-11 and ResNet-20 architectures on each target class. As shown in table 4, the range of average bit-flips required to achieve 100% ASR is between $4 \sim 6.8$ and $2.8 \sim 3$ for ResNet-20 and VGG-11, respectively. So for the N-to-1 attack, VGG-11 requires a consistently fewer number of bit-flips than ResNet-20 for all CIFAR-10 classes. We further observe that there is no obvious relation between attack success rate and

TABLE 5: Performance of T-BFA variants on ImageNet (from Hen class (i.e., label *8*) to Goose class (i.e., label *99*)). The original test accuracies of ResNet-18, ResNet-34 and MobileNet-V2 are 69.23%, 75.5% and 72.01%, respectively.

Type	Attack Success Rate (%)	Test Accuracy (%)	# of Bit-Flips	Attack Success Rate (%)	Test Accuracy (%)	# of Bit-Flips	Attack Success Rate (%)	Test Accuracy (%)	# of Bit-Flips
N-to-1	99.78 ± 0.27	0.23 ± 0.18	32.6 ± 8.2	99.99 ± 0	0.1 ± 0	21 ± 4	100 ± 0	0.1 ± 0	17.3 ± 3.29
1-to-1	100 ± 0	32.13 ± 14.4	16.7 ± 1.24	100 ± 0	23.74 ± 1.71	9.33 ± 0.94	100 ± 0	1.19 ± 0.22	13 ± 1.41
1-to-1 (S)	100 ± 0	59.48 ± 2.9	27.3 ± 16.7	100 ± 0	58.33 ± 3.29	40.33 ± 30.32	98.67 ± 1.89	33.99 ± 4.93	45.33 ± 21.74
	ResNet-18 (# of parameters: 11M)			ResNet-34 (# of parameters: 21M)			MobileNet-V2 (# of parameters: 2.1M)		

TABLE 6: Comparison with Competing Methods. We directly report the numbers from the respective papers for [17], [22]. For [16], [25] we run the attack on ResNet-20 8-bit quantized network.

Method	# of Data used to evaluate ASR	ASR (%)	Post Attack Test Accuracy (%)	# of Bit-Flips	Model Precision
Untargeted-BFA (I) [16]	10k	-	10.27	28	8-bit
Proposed N-to-1(I)	10k	100	10	**4**	8-bit
SBA (II) [17]	100	100	60.0	1	full-precision
Proposed 1-to-1 (II)	**1000**	100	10	3.2	**8-bit**
TBT (III) [25]	10k	93.89	82.03	199	8-bit
GDA (III) [17]	100	100	81.66	198	full-precision
Fault Sneaking (III) [22]	16	100	76.4	>2565	full-precision
Proposed 1-to-1 (s) III	1000	99.3	**88.3**	**12.2**	**8-bit**

pre-attack accuracy. Thus for a balanced dataset supervised learning problem, the pre-attack accuracy may not have a significant role in determining the attack performance of a specific target class.

Take-Away 1. Our analysis of the N-to-1 attack shows that there is no particular target class that is easier or more difficult to attack. Thus we conclude that the input feature patterns play a small role in resisting the attack, while the network architecture plays a more important role.

1-to-1 Attack. In this version of T-BFA, the attacker performs 1-to-1 miss-classification with fewer number of bit-flips (see fig. 3) in comparison to the N-to-1 version (see table 4). For most of entries shown in fig. 3, the 1-to-1 attack requires only 1-2 bit-flips to achieve 100%ASR with a few exceptions. Overall, for all possible combinations of classes, T-BFA successfully achieves 100% 1-to-1 miss-classification with a range of $1 \sim 7.4$ bit-flips.

Take-Away 2. 1-to-1 attack requires, in general, less bit-flips compared to N-to-1 attack. This is expected since mis-classifying all N classes are more difficult than mis-classifying just one class.

1-to-1 Stealthy (S) Attack. Our evaluation of 8-bit quantized ResNet-20 and VGG-11 models shows a 91.9% and 91.6% baseline CIFAR-10 test accuracy, respectively. As shown in fig. 4, after attack, the accuracy of ResNet-20 has a larger drop. The average test accuracy after five rounds of attack is between $31.3 \sim 88.3\%$ for ResNet-20. On the other hand, VGG-11 maintains a better test accuracy with a range of $48.3 \sim 90.1\%$.

Our T-BFA is effective in attacking ResNet-20 network by achieving ASR higher than 97% for all combinations of source and target classes. However, VGG-11 shows slightly better resistance to the attack with an ASR range of 93-99% for different combinations. This is consistent with prior work which also shows that denser networks (i.e., VGG-11, VGG-16) have better resistent to both adversarial weight attack [25] and input attack [7]. While for both networks, some classes are more vulnerable than others, most source class and target class combinations require less than 10 bit-flips to conduct the 1-to-1 stealthy attack.

TABLE 7: T-BFA Attack on DNNs Running in a Real Computer

Network	Attack Type	ASR (%)	Post Attack Accuracy (%)	Number Of Flips
ResNet-20 (CIFAR-10)	I	88.92	19.88	2
MobileNet-V2 (ImageNet)	II	96.8	2.2	11
VGG-11 (CIFAR-10)	III	98.6	80.6	2

Take-Away 3. A compact network, like ResNet-20 with 0.27M parameters, has less capacity to learn the dual objective function in 1-to-1 (S) attack through a small number of bit-flips in comparison to denser network, like VGG-11 with 132M parameters. As a result, the test accuracy drop for a compact network, like ResNet-20, is higher.

5.2 Experiments on ImageNet

ImageNet dataset has a much larger number of output classes compared to CIFAR-10. We do not have the space to report all targeted attack results, thus we randomly pick one combination of target attack (Hen class to Goose class) to show our method's efficiency. For N-to-1 attack, table 5 shows that T-BFA requires 32, 21 and 17.3 bit-flips, on average, for ResNet-18, ResNet-34 and MobileNet-V2, respectively. Aligning with the observation for CIFAR10, it can be seen that a more compact network is more vulnerable to the N-to-1 attack. For 1-to-1 (S) attack, a compact network, e.g., MobileNet-V2 (with 2.1M parameters), fails to maintain a reasonable test accuracy (i.e., 33.9%). In comparsion, larger networks, such as ResNet-18 and ResNet-34, can maintain a reasonable test accuracy (i.e., $\sim$ 59%) while achieving 100% ASR. Those experiments results also align with our observation for CIFAR10.

Take-Away 4. In the case of ImageNet dataset, a large number of output class and dense model architectures may

TABLE 8: T-BFA performance against existing BFA defense techniques [32]. PTA indicates Post-attack test accuracy.

Class	Clean Model	N-to-1			1-to-1			1-to-1(S)		
	TA(%)	PTA(%)	ASR (%)	# of flips	PTA (%)	ASR(%)	# of flips	PTA(%)	ASR(%)	# of flips
8-Bit	91.9	10.0	100.0	5.2	49.0	100.0	4.4	66.3	99.2	3.6
8-Bit (PC) [32]	91.29	10.0	100.0	5.6	47.7	100.0	3.0	66.49	97.84	8.6
Binary [32]	88.24	10.0	100.0	35.5	61.94	100.0	17	72.98	98.4	16

contribute towards increasing the attack difficulty. However, consistent with CIFAR-10 observations, it is easier to conduct a 1-to-1 (S) attack on a network with higher capacity. The larger optimization space helps achieve dual objectives of maintaining reasonable test accuracy as well as achieving very high ASR.

5.3 Comparison with Other Competing Methods

In this section, we compare our proposed T-BFA with most recent existing works of targeted attacks [16], [17], [22], [25] in adversarial weight attack domain.

As shown in table 6, our proposed N-to-1 targeted attack achieves the same objective as [16] with **7** $\times$ less number of bit-flips. Moreover, unlike un-targeted BFA (i.e., randomly classifying all inputs to a random class), our N-to-1 attack has precise control on the target output class. Other stronger versions of previous targeted attacks, such as GDA [17] and fault sneaking attacks [22], have shown superior results (100% ASR) against a weaker threat model (i.e., full-precision model or the attack is evaluated against only 100 images). However, our proposed T-BFA 1-to-1 (s) outperforms both [17], [22] on a quantized network with **16** $\times$ and **210** $\times$ fewer number of bit-flips. Among the Neural Trojan works [25], [27], [28], Targeted Bit Trojan (TBT) follows a more strict threat model and performs the attack with the least number of bit-flips. However, our 1-to-1 (s) proves to be much more efficient than TBT w/o input trigger, as required in the targeted Trojan attack, like TBT. In comparison, T-BFA achieves a higher test accuracy and higher ASR with **16** $\times$ fewer bit-flips.

The analysis presented above for both BFA [16] and TBT [25] uses the same architecture, data-set, and number of attack samples as ours. It shows that T-BFA outperforms these two powerful attacks fairly in terms of all three evaluation metrics: # of flips, attack success rate, and post-attack accuracy. However, for attacks from [17] and [22] with no open-source code, we report the attack performance directly from their respective papers, where their attack methods are evaluated using a more vulnerable full-precision DNN model, rather than weight-quantized model. Even so, our method still outperforms theirs in terms of all above three metrics on a more robust quantized DNN. These comparison still serves the purpose to show the strength of our method, mainly because all the previous adversarial weight attacks [16], [18] have concluded that attacking a quantized network (8-Bit) is much more difficult than attacking a full-precision model.

5.4 Attacking Real Computer Running DNNs:

In a real computer main memory, an 8-bit quantized DNN with M number of weights contains $(M/4096)$ physical memory pages (4KB) and within each page, one bit has an offset range (0-32767). We evaluate all three types of T-BFA on our prototype computer hardware (described earlier) running ResNet-20, VGG-11, and MobileNet-V2 summerized in table 7.

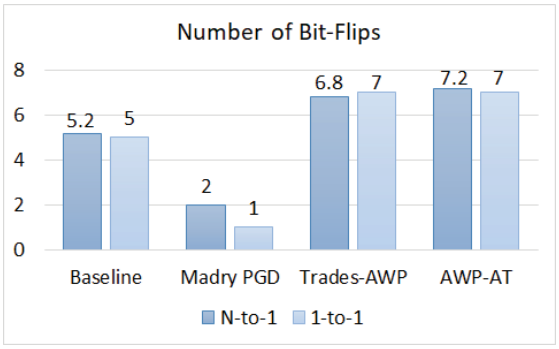


Fig. 5: Summary of attacking adversarial trained ResNet-20 model with N-to-1 and 1-to-1 attack. We report the # of bit-flips required to reach $\sim$ 100 % ASR. The source class is 3 and target class is 5. Here, we report the average of five individual rounds.

In our real computer attack system, by flipping ResNet-20 bit locations: (page # 65 offset # 12113);(page # 1 offset # 12600), attacker can achieve 88.92 % ASR on **Type I** attack (e.g., class 2). Similarly, by flipping two bits of VGG-11: (page # 2379 offset # 21352) ; (page # 2378 offset # 20504), attacker achieves 98.6% ASR for **type III** (class 9 $\rightarrow$ 1) on CIFAR-10. We test ImageNet results on MobileNet-V2 for **type II** attack (class 8 $\rightarrow$ 99). By flipping these 11 bit locations: (page # 1 offset # 7392) ; (page # 131 offset # 12883); (page # 3 offset # 25971); (page # 114 offset # 22842);(page # 143 offset # 10335);(page # 281 offset # 16537);(page # 298 offset # 3298);(page # 304 offset # 21736);(page # 285 offset # 14549);(page # 143 offset # 9359);(page # 465 offset # 19993), attacker would achieve 96.8 % ASR. Note that, due to the consideration of bit flip profile, the targeted bits can be flipped successfully in our physical testbed (*the online rowhammer exploitation takes less than 30 seconds*).

6 Discussion

6.1 Evaluation Against Existing Defense

Recently, [32] proposed Piece-wise Clustering (PC) as an effective training scheme to defend against Bit-Flip based untargeted weight attack [16]. We evaluate our T-BFA against PC methods in Table 8, showing that T-BFA (e.g., 1-to-1 (S)) still successfully (i.e., higher than 97.0 % ASR with tens or less # bit flips) attacks PC and binary network with a cost

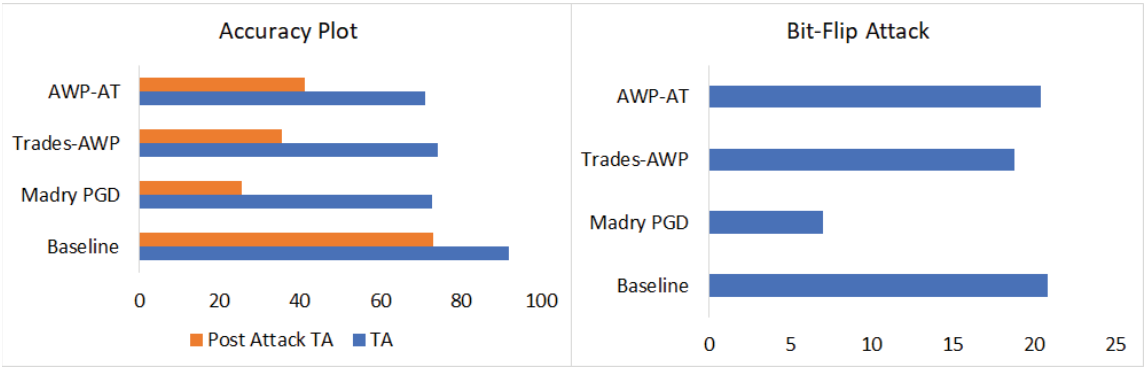


Fig. 6: Summary of attacking adversarial trained ResNet-20 model with Type III 1-to-1 (S). We report the post attack accuracy and # of bit-flips required to reach $\sim$ 99.0 % ASR for all cases. The source class is 3 and target class is 5.

of around 2 $\times$ and 5 $\times$ more flips, respectively, showing limited resistance improvement, but not significantly.

In summary, Piece-wise clustering or low-bit width [32] is still improving robustness against T-BFA, but not as effective as un-targeted BFA. According to our observation, an N-to-1 attack is a much stronger attack than an un-targeted BFA, meaning it requires 7 times fewer amount of bit-flips to degrade network accuracy to 10 percent in table 6. Since T-BFA is more effective than an un-targeted BFA, it also achieves better attack performance against existing [32] defense.

6.2 Effect of Adversarial Training

To further demonstrate the attack efficacy of T-BFA, we also evaluate the adversarial weight perturbation-based training defenses against our attack. Two of the effective adversarial weight perturbation training methods are Adversarial Weight Perturbation (AWP) and Trades-AT defense [36]. We also evaluate the effect of training the model with popular input adversarial training defense projected gradient descent (PGD) training [7]. We summarize the results of N-to-1 and 1-to-1 attack in fig. 5. It demonstrates that adversarial input training (e.g., Madry PGD [7]) makes the model more vulnerable to adversarial weight attack. Prior works [16], [32] have also reached a similar conclusion regarding input adversarial defense performance against BFA. Next, adversarial weight training (e.g., AWP-AT & Trades-AWP) helps slightly improve the robustness to T-BFA ($\sim$ 1-2 additional flips). In all the cases our attack still succeeds in achieving the attack with less than 8 bit-flips.

Similarly, in fig. 6, our 1-to-1(S) attack breaks (i.e., 99.0 % ASR) all the defenses with similar efficacy as the baseline model(i.e., less than 25 bit-flips). The observation is consistent with our prior BFA work [16], where adversarial training with BFA-based weight perturbation fails to show noticeable resistance improvement against BFA attack. Several possible reasons that such adversarial training works for defending adversarial input attack, but not for adversarial weight attack (e.g., BFA or T-BFA), are i) the adversarial weight noise dimension is significantly higher than input noise; ii) unlike adversarial input attack that typically requires the added noise magnitude to be within a very small epsilon (i.e., distortion metric, l_∞ norm) (typically ¡6%), bit-flip based adversarial weight attack could easily cause a significant change of weight parameter value (e.g. one bit flip in the most significant bit of 127('01111111' in binary) will change to -1 ('11111111' in binary)); iii) unlike defending adversarial input noise which targets optimizing more resilient weights to adapt a group of *fixed* training samples and their corresponding noise during adversarial training, the adversarial weight noise samples are ever-changing due to the update of weight parameters during every epoch of adversarial training. Thus, optimizing adversarial perturbation on these evolving weights is extremely difficult.

TABLE 9: Summary of possible directions to improve resistance against T-BFA.

Possible Defense Directions	I	II	III
1. Perform Weight Clustering (i.e., PC/Binary)	✓	✓	✓
2. Increase Network Capacity (e.g., larger size/ high bit-width)	✓	✓	
3. Decrease Network Capacity (e.g., smaller network)			✓
4. Securing critical layers (e.g., Classification layer)	✓	✓	✓
5. Adversarial Training Defenses (e.g., AWP,Trades)	✓	✓	

6.3 Layer-wise sensitivity.

We also observe that the most vulnerable or sensitive layer under the T-BFA attack is the last classification layer. In the case of 1-to-1 (s) attack, it is interesting to observe that 100% of all the identified vulnerable bits are in the last layer for both ResNet-20 and VGG-11 models. For the N-to-1 attack, more than 90% of bit-flips are in the last classification layer. This study leads to the question: *Can we defend T-BFA by securing the critical last layer for classification?* To answer this question, we assume the entire last layer is protected (i.e. no bit-flip is allowed) and run the T-BFA again. This is motivated by prior work that secures the entire last layer in a protected enclave of computer processor, such as Intel SGX [37], as an effective privacy protection method. Unfortunately, all three versions of T-BFA still succeed with a cost of limited additional number (all less than 30) of bit flips. Thus, this scheme helps slightly improve the resistance, but not significantly.

6.4 Summary of Potential Defenses

Based on above discussion and our summarized takeaways, we list several directions we have explored to improve DNN model resistance against different types of T-BFA in Table 9. In our experiments, those methods only help

improve DNN model resistance in a limited degree. However, none of them could significantly improve Robustness. For example, the largest bit-flip # for any type of T-BFA to succeed on CIFAR-10 is 36 when attacking binary network in table 8, which is still a practical number in real-computer memory fault injection as discussed in [19], [38].

7 Conclusion

In this work, as far as we know, we are the first to propose three targeted adversarial weight attack schemes, i.e., N-to-1, 1-to-1 and 1-to-1(stealthy), which severely degrade the classification performance of quantized DNNs. Our T-BFA is based on an iterative class-dependent bit searching algorithm. Extensive experiments have been conducted to prove the efficacy of our proposed T-BFA in different DNN architectures on CIFAR10 and Imagenet datasets. Moreover, we also demonstrate our T-BFA in a real-computer running DNNs. In the end, we provide several possible analysis and directions to construct robust DNN models against T-BFA.

8 Acknowledgement

This work is supported in part by the National Science Foundation under Grant No.1931871, No. 2019548, and No. 2019536.

References

[1] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," *URL http://www. cs. toronto. edu/kriz/cifar. html*, 2010.

[2] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, 2019.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[4] M. A. Haque, A. Verma, J. S. R. Alex, and N. Venkatesan, "Experimental evaluation of cnn architecture for speech recognition," in *First International Conference on Sustainable Technologies for Computational Intelligence*. Springer, 2020, pp. 507–514.

[5] M.-T. Luong, M. Kayser, and C. D. Manning, "Deep neural language models for machine translation," in *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, 2015, pp. 305–309.

[6] Y. Lu, X. Xiong, W. Zhang, J. Liu, and R. Zhao, "Research on classification and similarity of patent citation based on deep learning," *Scientometrics*, pp. 1–27, 2020.

[7] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *ICLR*, 2015.

[9] M. Yan, C. W. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn {DNN} architectures," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 2003–2020.

[10] Y. Xiang, Z. Chen, Z. Chen, Z. Fang, H. Hao, J. Chen, Y. Liu, Z. Wu, Q. Xuan, and X. Yang, "Open dnn box by power side-channel attack," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2020.

[11] H. Yu, H. Ma, K. Yang, Y. Zhao, and Y. Jin, "Deepem: Deep neural networks model recovery through em side-channel information leakage," 2020.

[12] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen, "X-deepsca: Cross-device deep learning side channel attack," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[13] C. Roscian, A. Sarafianos, J.-M. Dutertre, and A. Tria, "Fault model analysis of laser-induced faults in sram memory cells," in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 2013, pp. 89–98.

[14] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," in *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3. IEEE Press, 2014, pp. 361–372.

[15] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos, "Flip feng shui: Hammering a needle in the software stack," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 1–18.

[16] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 1211–1220.

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

[18] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitraș, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 497–514.

[19] F. Yao, A. S. Rakin, and D. Fan, "Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips," *29th {USENIX} Security Symposium ({USENIX} Security)*, 2020.

[20] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 15–26.

[21] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 39–57.

[22] 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 *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[23] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, pp. 1–12.

[24] M. Agoyan, J.-M. Dutertre, A.-P. Mirbaha, D. Naccache, A.-L. Ribotta, and A. Tria, "How to flip a bit?" in *2010 IEEE 16th International On-Line Testing Symposium*. IEEE, 2010, pp. 235–239.

[25] A. S. Rakin, Z. He, and D. Fan, "Tbt: Targeted neural network attack with bit trojan," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[26] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[27] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[28] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society, 2018.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

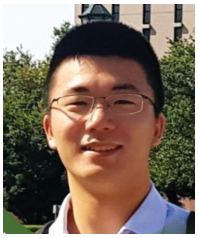
[32] Z. He, A. S. Rakin, J. Li, C. Chakrabarti, and D. Fan, "Defending and harnessing the bit-flip based adversarial weight attack," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 095–14 103.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Pro-*

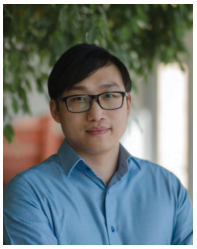
ceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[34] P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, "DRAMA: Exploiting DRAM addressing for cross-cpu attacks," in *USENIX Security Symposium*, 2016, pp. 565–581.
[35] D. Gruss, M. Lipp, M. Schwarz, D. Genkin, J. Juffinger, S. O'Connell, W. Schoechl, and Y. Yarom, "Another flip in the wall of rowhammer defenses," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 245–261.
[36] D. Wu, S.-T. Xia, and Y. Wang, "Adversarial weight perturbation helps robust generalization," *Advances in Neural Information Processing Systems*, vol. 33, 2020.
[37] F. Tramer and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJVorjCcKQ
[38] L. Cojocar, K. Razavi, C. Giuffrida, and H. Bos, "Exploiting correcting codes: On the effectiveness of ecc memory against rowhammer attacks."

Adnan Siraj Rakin Adnan Siraj Rakin is a Ph.D. student at Arizona State University, advised by Dr. Deliang Fan. Before that he completed a B.Sc. degree in Electrical and Electronic Engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2016. His research interests primarily focus on secure deployment of deep learning networks, exploring the attack and defense of adversarial examples and weight attack domain and efficiency of machine learning algorithms. Email: asrakin@asu.edu

Zhezhi He Dr. Zhezhi (Elliot) He is currently Assistant Professor in Shanghai Jiao Tong University, Shanghai, China. At the time of conducting this research he was a graduate student at Arizona State University. He got his Ph.D. degree from School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ in 2020 under the supervision of Dr. Deliang Fan. He received the M.Eng. degree in electrical and computer engineering from Oregon State University, Corvallis, OR, USA, in 2016, and the B.S. degree in Information Engineering from Southeast University, Nanjing, China, in 2012. His research interests are in the area of secure and efficient deep learning, brain-inspired in-memory computing, and post-CMOS technologies. Email: zhezhi.he@sjtu.edu.cn



Jingtao Li Jingtao Li, currently a PhD student in electrical engineering at Arizona State University advised by Dr. Chaitali Chakrabarti. He got his Bachelor degree in Microelectronics Science and Engineering at UESTC, Chengdu. His research interests are on privacy, security and reliability of deep neural network systems including preventing neural architecture leakage from side-channel attack, improving robustness of neural network systems under weight attack and stuck-at fault. Email: jingtao1@asu.edu



Fan Yao Dr. Fan Yao is currently an Assistant professor in the Department of Electrical and Computer Engineering at University of Central Florida. His research interests are in the areas of computer architecture, security, and energy efficient computing. He has recieved awards including GWU Best Dissertation Award, 2019; NSF GW I-Corps Site Grant Award, 2018; GWU SEAS R & D Showcase 2nd Place in Experimental Research, 2018; The Norris & Betty Hekimian Engineering Endowment Fellowship, GWU, 2017; National Endeavor Scholarship, HUST, 2010. He recieved an award from NSF CNS to explore side channel attacks and defenses in NVM-integrated computing systems (August 2020). He also received an award from NSF SaTC to investigate ML security issues due to hardware-based model tampering (August 2020). Email: Fan.Yao@ucf.edu

Chaitali Chakrabarti Dr. Chaitali Chakrabarti received her B. Tech. in Electronics and Electrical Communication Engineering from the Indian Institute of Technology (IIT), Kharagpur, India in 1984. She received her M.S. and Ph.D. in Electrical Engineering Dept from U. of Maryland, College Park in 1986 and 1990 respectively. She has been at ASU since fall 1990. Chaitali's research interests are in the areas of VLSI architectures for signal processing and communications, algorithm-architecture co-design of signal processing systems, and all aspects of low power embedded system design including those that operate at near-threshold voltages. She has published about 100 journal papers and 200 conference papers in these areas. Chaitali received the Distinguished Alumni Awards from University of Maryland in 2013 and IIT Kharagpur in 2018. At ASU, she has received multiple teaching awards and also the Fulton Exemplar Award 2014-2016. She is also a Fellow of the IEEE. Email: chaitali@asu.edu

Deliang Fan Dr. Deliang Fan is currently an Assistant Professor in the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA. Before joining ASU in 2019, he was an assistant professor in Department of Electrical and Computer Engineering at University of Central Florida (UCF), Orlando, FL, USA. He is also currently a courtesy professor at ECE department of UCF. He received his M.S. and Ph.D. degrees, under the supervision of Prof. Kaushik Roy, in Electrical and Computer Engineering from Purdue University, West Lafayette, IN, USA, in 2012 and 2015, respectively. Dr. Fan has authored 110+ peer-reviewed international journal/conference papers. His research group is funded by National Science Foundation (NSF), Semiconductor Research Corporation (SRC), Cyber Florida, Moffitt Cancer Center, ASU, SCEEE Research Initiation grant, NanoScience Technology Center at UCF seed grant and UCF InHouse grant, etc. Email: dfan@asu.edu