Defending and Harnessing the Bit-Flip based Adversarial Weight Attack

Zhezhi He†, Adnan Siraj Rakin†, Jingtao Li, Chaitali Chakrabarti and Deliang Fan
School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287
zhezhihe@asu.edu, dfan@asu.edu

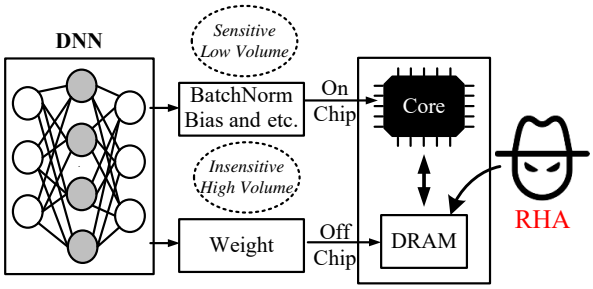
Abstract

Recently, a new paradigm of the adversarial attack on the quantized neural network weights has attracted great attention, namely, the Bit-Flip based adversarial weight attack, aka. Bit-Flip Attack (BFA). BFA has shown extraordinary attacking ability, where the adversary can malfunction a quantized Deep Neural Network (DNN) as a random guess, through malicious bit-flips on a small set of vulnerable weight bits (e.g., 13 out of 93 millions bits of 8-bit quantized ResNet-18). However, there are no effective defensive methods to enhance the fault-tolerance capability of DNN against such BFA. In this work, we conduct comprehensive investigations on BFA and propose to leverage binarization-aware training and its relaxation – piece-wise clustering as simple and effective countermeasures to BFA. The experiments show that, for BFA to achieve the identical prediction accuracy degradation (e.g., below 11% *on CIFAR-10), it requires* $19.3\times$ *and* $480.1\times$ *more effective malicious bit-flips on ResNet-20 and VGG-11 respectively, compared to defend-free counterparts.*

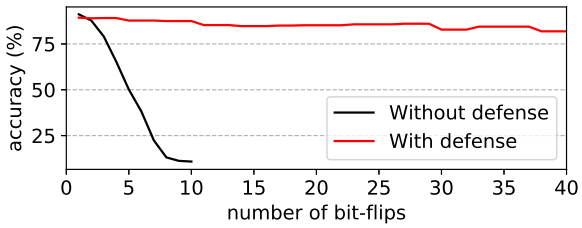
1. Introduction

As the Deep Neural Networks (DNNs) achieve human-surpassing performance in multiple computer vision related tasks, its applications in the real-world scenarios are growing rapidly. In such a scenario, the fault-tolerance capability of the neural network is of great research interest for developing reliable neural networks against weak random fault and even strong malicious attacks. A significant amount of research effort has focused on DNNs being fooled by human-imperceptible input noise, aka. adversarial example. However, another vulnerable dimension of DNN is model parameters, which has been barely investigated.

Owing to the enormous model size (hundreds of MBs for state-of-the-art DNNs [9, 24]), modern DNN accelerators (e.g., GPU) normally need to store the model parameters in main memory, namely, Dynamic Random-Access-Memory (DRAM). Recent research advances have brought up the vulnerability issue of data stored in DRAM, where Row-Hammer Attack (RHA) [19] has been shown to maliciously flip the memory bits in DRAM without being granted any data write privileges, as depicted in Fig. 1. Unfortunately, DNNs stored in DRAM with floating-point representation can be easily hacked to fully malfunction, through single bit-flip (e.g. in an exponential bit of any weight) through RHA [8]. Thanks to the DNN weight quantization technique, DNN is more compact since the weights are represented in a fixed-point format with constrained representation. Such a representation has been proven to significantly enhance the immunity of quantized DNN to such malicious bit-flips in [8]. However, a newly proposed Bit-Flip Attack (BFA) [17] whose progressive bit searching algorithm can



(a) Concept illustration of quantized DNN under BFA.



(b) Accuracy vs. # bit-flips with/without defense.

Figure 1: The fault injection on the identified vulnerable weight bits can be physically conducted by Row-Hammer Attack (RHA) [19]. Meanwhile, the DNN under defense has higher resistance against the malicious bit-flips.

† These authors contributed equally
Code is released at: https://github.com/elliothe/BFA

successfully identify and flip an extremely small number of vulnerable weight bits (e.g., 13 out of 93 millions bits of ResNet-18 on ImageNet) to degrade a large scale 8-bit quantized DNN inference accuracy to as low as a random guess (i.e., from 69.8% to 0.1%). Up to now, there is still a lack of effective defensive approaches against such BFA, and so we propose a BFA countermeasure based on utilizing weight binarization and its relaxation – piece-wise clustering. The contributions in this work can be summarized as:

- A comprehensive investigation of bit-flip based adversarial weight attack (i.e., BFA) is conducted, and several insightful observations are obtained for understanding parameter vulnerability to these attacks.
- Weight binarization and its piece-wise clustering relaxation method are proposed as the effective defensive techniques against BFA.
- Additional adversarial attack defense methods (e.g., adversarial training, pruning) and conventional model regularization methods are examined as well.

2. Background and Related Works

2.1. Bit-Flip based Adversarial Weight Attack

The bit-flip based adversarial weight attack, aka. Bit-Flip Attack (BFA) [17], is an adversarial attack variant which performs weight fault injection through flipping the bits. For the machine-imperceptible purpose, the BFA only flips the most vulnerable weight bits which are identified by Progressive Bit Search (PBS) algorithm with iterative inter- and intra-layer search.

Given a n_{q}-bit quantized DNN parameterized by bits (i.e., quantized weights in binary), define tensor $\{\mathbf{B}_l\}_{l=1}^{L}$, where $l \in \{1, 2, ..., L\}$ is the layer index. The intra-layer search that identifies the bit with highest gradient ($\arg\max_{\mathbf{B}_l} |\nabla_{\mathbf{B}_l} \mathcal{L}|$) as vulnerable bit candidate, where $\mathcal{L}$ is the inference loss. Then, the inter-layer searching compares the bit candidates selected by the intra-layer search through directly checking the loss increment. Thus, the bit searching in iteration i can be formulated as an optimization process [17]:

$$\begin{aligned} &\max_{\{\hat{\mathbf{B}}_l^i\}} \mathcal{L}\Big(f\big(\boldsymbol{x}; \{\hat{\mathbf{B}}_l^i\}_{l=1}^{L}\big), \tilde{\boldsymbol{t}}\Big) \\ &\text{s.t. } \tilde{\boldsymbol{t}} = f(\boldsymbol{x}; \{\mathbf{B}_l\}_{l=1}^{L}); \sum_{l=1}^{L} \mathcal{D}(\hat{\mathbf{B}}_l^i, \mathbf{B}_l) \in \{0, 1, ..., N_b\} \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ and $\boldsymbol{t}$ denotes the selected input mini-batch and ground-truth labels. $\hat{\mathbf{B}}_l^i$ is the quantized bit tensor of l-th layer perturbed by BFA in i-th iteration. $f(\boldsymbol{x}; \{\mathbf{B}\}_{l=1}^{L})$ compute the outputs of DNN parameterized by $\{\mathbf{B}_l\}_{l=1}^{L}$. $\tilde{\boldsymbol{t}}$ is the output of clean model as the soft-label, which replaces the ground-truth $\boldsymbol{t}$ to perform the attack. $\mathcal{L}(\cdot, \cdot)$ computes the loss. The attack efficiency is measured by the Hamming distance (i.e., effective bit-flips) between prior- and post-attack model parameters $\{\hat{\mathbf{B}}_l^i\}_{l=1}^{L}$ and $\{\mathbf{B}_l\}_{l=1}^{L}$ given by $\sum \mathcal{D}(\hat{\mathbf{B}}_l^i, \mathbf{B}_l)$. In general, the optimization goal of BFA is to cause the DNN to malfunction with least number of bit-flips (i.e., $\min \sum \mathcal{D}(\hat{\mathbf{B}}_l^i, \mathbf{B}_l)$).

Table 1: Threat model of Bit-Flip Attack (BFA).

Access Required ✓	Access **NOT** Required ✗
Model topology & parameters	Hyper-parameters and other training configurations.
A mini-batch of sample data	Complete train/test datasets.

It is noteworthy that the quantized weight in fixed-point format is magnitude constrained (i.e., $\max(\mathbf{B}) = 2^{n_{\text{q}}-1}$) in comparison to the floating-point counterpart, which is not only more biologically plausible but also practically necessary for the acceleration of modern AI applications. To clarify, we use the same threat model as in prior work [17], which is listed in Table 1.

2.2. Defense against Adversarial Example

As the BFA is an adversarial attack variant, the popular techniques used to defend adversarial example [5] are investigated to seek potential BFA defense method.

Adversarial Training. Adversarial training [5, 15] is by-far the most successful adversarial example defense method, that optimizes the DNN parameters $\boldsymbol{\theta}$ w.r.t both the clean input $\boldsymbol{x}$ and their adversary examples $\hat{\boldsymbol{x}}$ as:

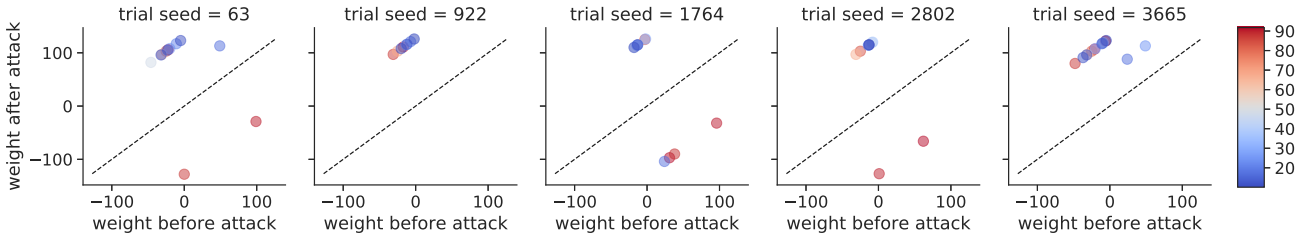
$$\min_{\boldsymbol{\theta}} \mathcal{L}(f(\boldsymbol{x}; \boldsymbol{\theta}), \boldsymbol{t}) + \alpha \cdot \mathcal{L}(f(\hat{\boldsymbol{x}}; \boldsymbol{\theta}), \tilde{\boldsymbol{t}}) \tag{2}$$

where α is the hyper-parameter to balance the accuracy of the trained model on clean natural data and adversarial examples. $\tilde{\boldsymbol{t}}$ is the soft-label as in Eq. (1). Such adversarial training is also normally considered as a strong regularization technique.

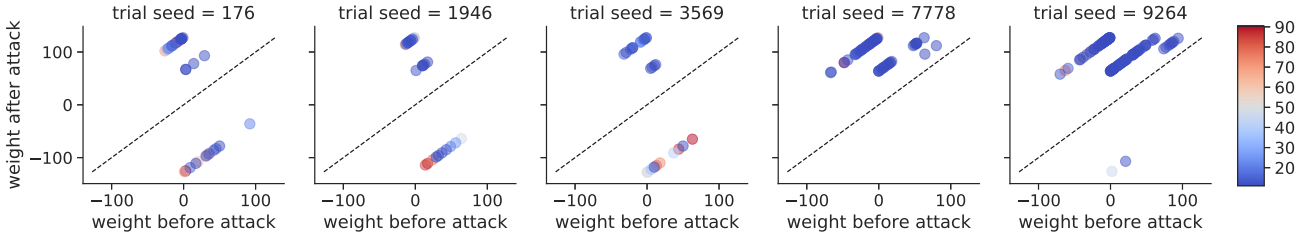
Increasing model capacity. Prior works [15, 7] have experimentally confirmed the resistance improvement against the adversarial attack by increasing the model capacity. It is interpreted as that the robust classifiers would require a more complicated decision boundary [15], which is expected to benefit the defense against malicious weight change as well. Further in-depth analysis of model capacity and BFA resistance is discussed in Section 6.

3. DNN under BFA 101

To first understand, then to defend and harness the bit-flip based adversarial weight attack, we conducted some preliminary investigations, along with several important observations as described below.



(a) ResNet-20 [6] with 8-bit weight. Bit-flips N_{BF} for BFA (mean±std):11.2 ± 1.9, total number of bits of weights: 2 millions.



(b) VGG-11 [20] with 8-bit weight. Bit-flips N_{BF} for BFA (mean±std):56.6 ± 35.2, total number of bits of weights: 78 millions.

Figure 2: Weight shift caused by BFA for (a) ResNet-20 [6] and (b) VGG-11 [20] on CIFAR-10 dataset. For both architectures, 5 trials are executed with different random seeds. Each colored dot depicts the weight shift (x-axis: prior-attack weight, y-axis: post-attack weight) w.r.t one iteration of BFA. The color bar indicates the corresponding accuracy (%) on the CIFAR-10 test data. The vertical distance between dot and the diagonal dashed line (i.e., $y = x$) represents the weight shift magnitude. Moreover, results reported in this figure use 8-bit post-training weight quantization [17].

Observation 1 *BFA is prone to flip bits of close-to-zero weights, and cause large weight shift.*

As depicted in Fig. 2, the progressive bit search proposed in BFA [17] is prone to identify vulnerable bit in the weight whose absolute value has a small magnitude (i.e., $|w| \rightarrow 0$) then modify it to be a large value (explained in the caption of Fig. 2). Since the BFA performs the attack on the quantized weight encoded in two's complement, the possible weight magnitude shift is discretized as $2^i, i \in \{0, 1, ..., n_q\}$. Moreover, Fig. 2 also shows the model with larger capacity possesses higher resistance against BFA (i.e., require more bit-flips for same accuracy degradation).

Observation 2 *BFA is prone to flip the weight bits in the front-end layers of the target neural network.*

Fig. 3 shows the histogram of bit-flips across different modules[1] of DNN under BFA. All the trials show that most of the bits found by BFA are mostly in the front-end, along the forward propagation path. Such an observation can be explained as the error introduced by the bit-flips in the front-end can be easily accumulated and amplified during the forward propagation, which is similar to the linear explanation of adversarial example discussed in [5].

[1]module index includes all modules within DNN, where small to large index denotes the location from front to rear along the inference path.

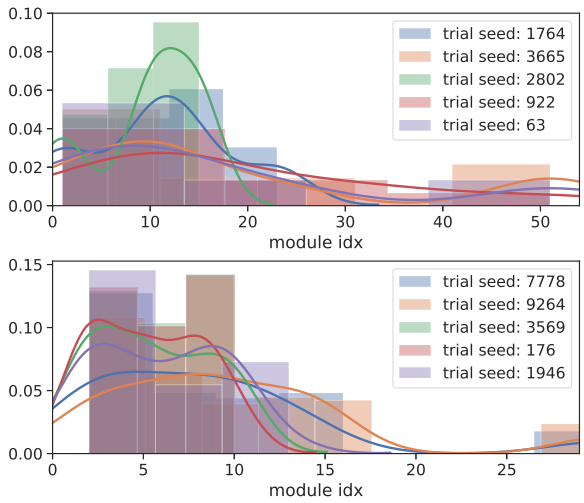


Figure 3: Normalized histogram and Kernel Density Estimation (KDE) of #bit-flips versus module index in (top) ResNet-20 and (bottom) VGG-11 on CIFAR-10.

Observation 3 *BFA forces almost all the inputs to be classified into one particular output group.*

Fig. 4 depicts the top-1 categorization output of DNN on CIFAR-10 test data, at different BFA iterations. The CIFAR-10 test subset includes 1000 samples on 10 output categories (i.e., total 10k samples). The BFA-free clean

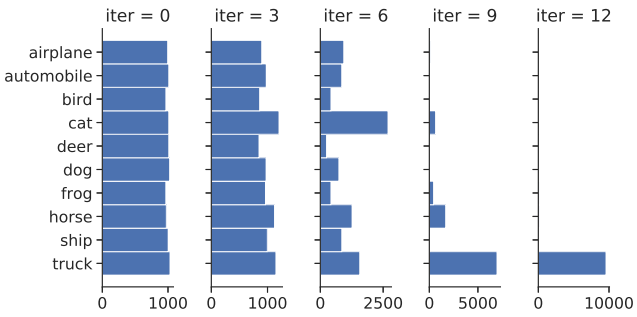


Figure 4: The evolution of ResNet-20 output classification histogram across 10 categories under BFA, on 10k test samples of CIFAR-10. The attack sample size is 128.

model, at iteration 0, has almost evenly distributed top-1 classification output predictions in each output category. It is intriguing to notice that, with the evolution of BFA, it forces almost all inputs to be classified into one output group. We also find that the dominant output group highly depends on the given attack sample data.

4. Defense against BFA

To enhance the resistance of DNN against BFA, we propose and investigate two defense techniques, i.e., Binarization-aware training and its relaxation – piece-wise weight clustering, inspired by the observations in Section 3.

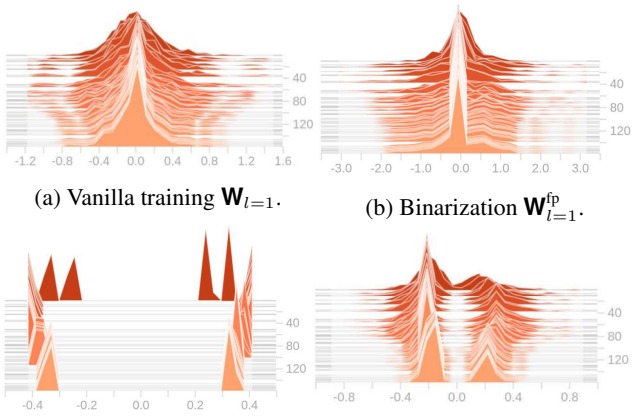
4.1. Binarization-aware training

binarization-aware training is originally proposed as an extreme low bit-width model compression technique, which converts the weights from 32-bit floating-point to {-1,+1} binary format encoded by 1-bit [18]. Here, the binarization-aware training is leveraged as a defense technique against BFA, which can be mathematically described as:

$$\begin{aligned} \text{Forward}: \quad & w^{\text{b}}_{l,i} = \mathbb{E}(|\mathbf{W}^{\text{fp}}_l|) \cdot \text{sgn}(w^{\text{fp}}_{l,i}) \\ \text{Backward}: \quad & \frac{\partial \mathcal{L}}{\partial w^{\text{b}}_{l,i}} = \frac{\partial \mathcal{L}}{\partial w^{\text{fp}}_{l,i}} \end{aligned} \tag{3}$$

where sgn() is the sign function. $w^{\text{b}}_{l,i}$ denotes the binarized weight from its floating-point counterpart $w^{\text{fp}}_{l,i}$. In general, weight binarization intrinsically achieves two goals: 1) reducing the bit-width to 1, and 2) clustering the weights to $\pm\mathbb{E}(|\mathbf{W}^{\text{fp}}_l|)$ as in Eq. (3). The Straight Through Estimator (STE) [2] is adopted to address the non-differential problem for the sign function as prior works [10]. Nevertheless, different from STE in [10], the gradient clipping constraint is omitted from the backward path, thanks to the presence of weight scaling coefficient $\mathbb{E}(|\mathbf{W}^{\text{fp}}_l|)$.

Our interpretation of the BFA resistance enhancement through binarization-aware training comes in twofold: 1)



(a) Vanilla training $\mathbf{W}_{l=1}$. (b) Binarization $\mathbf{W}^{\text{fp}}_{l=1}$.

(c) Binarization $\mathbf{W}^{\text{b}}_{l=1}$. (d) Piecewise Clustering $\mathbf{W}_{l=1}$.

Figure 5: The evolution of weight distribution of ResNet-20 (first layer, $l = 1$), under various training configurations. x-axis: weight magnitude, y-axis: training epoch.

As discussed in observation-1, BFA is prone to attack the close-to-zero weights and cause the large weight shift. The weight binarization eliminates close-to-zero weights by forcing all the weights to $\pm\mathbb{E}(|\mathbf{W}^{\text{fp}}_l|)$, where the weight distribution of sampled layer is illustrated in Fig. 5c. 2) Binarization-aware training intrinsically acts as training the DNN with bit-flip noise injected. As described in Eq. (3), the floating-point weight base $\{\mathbf{W}^{\text{fp}}_l\}$ are binarized on-the-fly during training. Recalling the optimization using SGD, the weight change Δw^{fp} can be expressed as:

$$\Delta w^{\text{fp}} = \eta \cdot \nabla_{w^{\text{fp}}} \mathcal{L}(f(\boldsymbol{x}, \{\mathbf{W}^{\text{fp}}\}); \boldsymbol{t}) \tag{4}$$

where η is the learning rate. Due to the presence of Eq. (3), even small weight update on w^{fp} (i.e., $w^{\text{fp}} - \Delta w^{\text{fp}}$) may directly change the corresponding binarized weight w^{b} from -1 to +1 or the opposite as a bit-flip, when the following condition is meet:

$$\text{sgn}(w^{\text{fp}} - \Delta w^{\text{fp}}) \neq \text{sgn}(w^{\text{fp}}) \tag{5}$$

Therefore, the binarization-aware training involves massive bit-flips on the binarized weight $\boldsymbol{W}^{\text{fp}}$, which mimics injecting the bit-flips noise on the weights during training. Fig. 6 depicts the average number of bit-flips caused by the weight update when training a binarized ResNet-20, each iteration may cause around 300 bit-flips on the binarized weights even when the learning rate is 0.001.

4.2. Clustering as relaxation of binarization

Since weight binarization normally suffers from significant prediction accuracy degradation due to aggressive model capacity reduction, we propose a relaxation to the weight binarization, called *Piece-wise Clustering* (PC), to emit the fixed single bit-width constraint while retaining

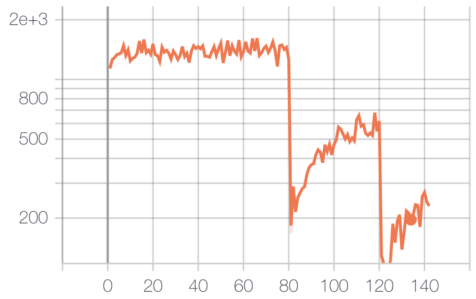


Figure 6: Average #Bit-flips (y-axis) per weight update iteration of binarization-aware training vs. epochs (x-axis), with ResNet-20 on CIFAR10.

similar functionality of clustering, which we believe play an important role in defending BFA. The piece-wise clustering introduces an additional weight penalty to the inference loss $\mathcal{L}$ (e.g., cross-entropy), and the optimization can be formulated as:

$$\min_{\{\mathbf{W}_l\}_{l=1}^{L}} \mathbb{E}_{\boldsymbol{x}}\, \mathcal{L}(f(\boldsymbol{x}, \{\mathbf{W}_l\}_{l=1}^{L}), \boldsymbol{t}) + \underbrace{\lambda \cdot \sum_{l=1}^{L} (||\mathbf{W}_l^{+} - \mathbb{E}(\mathbf{W}_l^{+})||_2 + ||\mathbf{W}_l^{-} - \mathbb{E}(\mathbf{W}_l^{-})||_2)}_{\text{piece-wise clustering penalty term}} \tag{6}$$

where λ is the clustering coefficient to tune the strength of the weight clustering penalty term. $\mathbf{W}_l^{+}$ and $\mathbf{W}_l^{-}$ denote the positive and negative weight subset of l-th layer weight tensor. The DNN model optimized as Eq. (6) leads to a bi-modal weight distribution as depicted in Fig. 5d. The piece-wise clustering proposed above can also be viewed as a variant of group Lasso, where the group is defined as the positive and negative weight subsets in each layer.

5. Experiments

5.1. Experiment Setup

Dataset and Network Architectures In this work, experiments are focused on visual dataset CIFAR-10 [12], which includes 60k 32×32 RGB images evenly sampled from 10 categories, with 50k and 10k samples for training and test respectively. The data augmentation technique is identical as reported in [6]. The ResNet-20 [6] and VGG-11 [20] are the two networks studied in the work. We use the momentum-based stochastic gradient descent optimizer, with training batch-size and weigh decay as 128 and 3e-4 respectively. The initial learning rate is 0.1 that scaled by 0.1 at 80 and 120 epochs, and the total number of epochs is 160. Note that, all the experiments are conducted using Pytorch [16], running on NVIDIA Titan-XP GPUs.

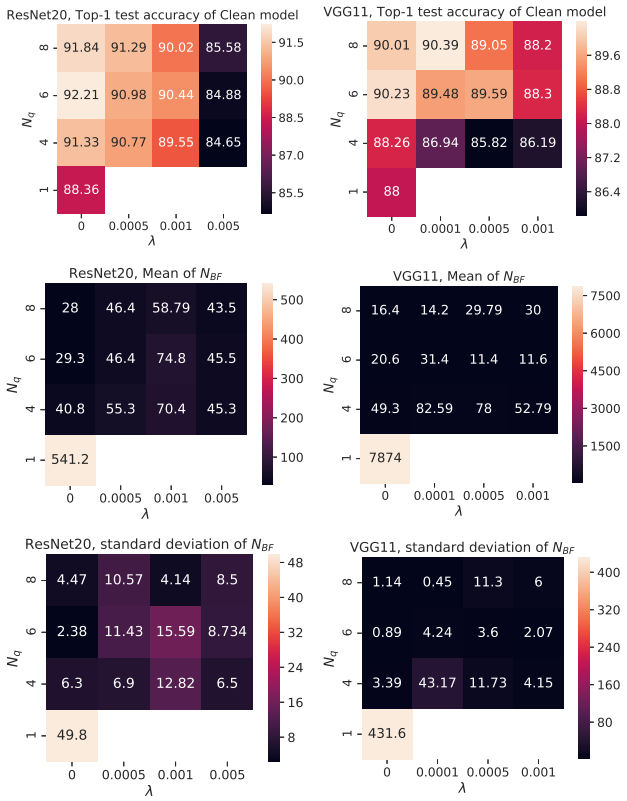


Figure 7: The BFA-free test accuracy, mean and standard deviation of N_{BF} for 5-trials under different quantization bit-width $n_q \in \{8, 6, 4, 1\}$ and clustering penalty coefficient $\lambda \in \{0, 1e-4, 5e-4, 1e-3, 5e-3\}$, with (left column) ResNet-20 and (right column) VGG-11 on CIFAR-10.

BFA Configuration. To evaluate the effectiveness of the proposed defense methods, the code from [17] is utilized with further modification. The number of bit-flips N_{BF} that degrades the prediction accuracy below 11% is used as the metric to measure the BFA resistance, for CIFAR-10 dataset. Moreover, since BFA requires a set of data to perform the attack, we take 256 sample images from the training subset as the default BFA configuration and report the mean±std of N_{BF} with 5 BFA trials. Note that, all the quantized DNN reported hereafter still uses the uniform quantizer as in [17], but with quantization-aware training instead of post-training quantization.

5.2. Result Evaluation

The experiment results with different quantization bit-width n_q and clustering coefficient λ of piecewise clustering are summarized in Fig. 7. It reports the BFA-free test accuracy and number of bit-flips N_{BF} required for BFA to succeed. Note that, for weight binarization in Fig. 7, we exclude the Piece-wise Clustering (PC) term through setting

$\lambda = 0$, since binarization intrinsically performs the clustering as discussed in Section 4.1.

Effect of quantization bitwidth and clustering coefficient. Based on the results reported in Fig. 7, training the DNN with binarized weights roughly degrade the test accuracy by $\sim 4\%$ and $\sim 2\%$ in comparison to the 8-bit quantized counterpart, for ResNet-20 and VGG-11. As discussed in Section 4.2, our intention of proposing the piece-wise clustering as the relaxation to weight binarization is to mitigate such accuracy drop. We do observe that using piece-wise clustering with proper λ can mitigate the accuracy degradation while improving the BFA resistance (i.e., requiring more bit-flips N_{BF} for the same accuracy degradation). For ResNet-20 and VGG-11, the ideal configurations of λ are 0.001 and 0.0005 respectively, as the model with larger capacity benefits from relatively smaller λ.

BFA resistance of ResNet-20. The 8-bit quantized ResNet-20 (baseline) requires only an average of 28 bit-flips to hamper the functionality of an accurate DNN, while the weight binarization significantly improves the BFA resistance compared to the baseline. Binarization increases the average value of N_{BF} to 541.2, which improve the BFA resistance by $\sim 19\times$. Nevertheless, considering the inevitable accuracy drop due to the drastic bit-width reduction (32-bit to 1-bit), as an alternative approach we explore the performance of PC on other bit-width configurations as well. With $\lambda = 1e-3$, the average value of N_{BF} was improved by $2.09\times$, $2.55\times$ and $1.73\times$ for 8, 6 and 4 bit-width respectively. In conclusion, our proposed piece-wise clustering improves the resistance to adversarial weight attack for all the cases of different bit-widths. Still, the binarized network emerges as the most successful defense against BFA.

BFA resistance of VGG-11. For VGG-11, our observation follows a similar pattern as described in the previous section. The baseline VGG-11 (e.g., $n_q = 8$) requires an average N_{BF} of 16.4. Again, weight binarization improves the network robustness significantly, yielding an average N_{BF} of 7874; which is $\sim 480\times$ improvement in comparison to the baseline. In the case of VGG-11, the lower Bit-Width defends BFA even better than ResNet-20; the main reason for this discrepancy can be the difference between the size of the network. For a larger network such as VGG-11, low Bit-Width and PC performs a proper regularization to successfully defend against BFA. The best performance of PC for VGG-11 was achieved for a 4-bit network with $\lambda = 1e-4$ achieving an average value of N_{BF} of 82.59.

In summary, both the binarization-aware training and its piecewise clustering relaxation can improve the BFA resistance of the target neural network, while the binarization-aware training can push the N_{BF} to an extremely large value (e.g., $N_{\text{BF}} > 7000$ on over-parameterized VGG-11). The implication of such large value is noteworthy, as a larger value of N_{BF} indicates the significant increase in difficulty to carry out a memory fault injection through the feasible cyber-physical attacks. For example, when using the row-hammer attack to perform the fault injection, the increased attack execution time might be detected by the operating system through the data integrity check.

5.3. Comparison of Alternative Defense Methods

Adversarial weight attack [17] is a recently developed security threat model for modern DNN. Subsequently, the development of defensive approaches in this field has not received much attention. Therefore, for the first time, we investigate an alarming parameter security concern – bit-flip based adversarial weight attack, with corresponding defense method. Owing to the lack of competing methods in this research direction, we attempt to transfer several conventional defense methods of adversarial examples [15, 3, 22], for providing a comprehensive comparison.

Table 2: **Alternative Methods Comparison**. In this table, we report the prior- and post-attack test accuracy (%) and N_{BF} of BFA. The 8-bit quantization is chosen as the baseline; Binary and PC-8bit is the proposed method. Moreover, comparison with Lasso-based pruning and adversarial weight training (adv. training) is included as well.

Methods	Prior-Attack Accuracy (%)	Post-Attack Accuracy (%)	N_{BF}
8-bit	91.84	10.45	28.0±4.47
PC-8bit	90.02	10.07	**58.79±4.14**
Binary	88.36	10.13	**541.2 ± 49.8**
Lasso Pruning	88.11	10.12	6.8±0.44
Adv. Training	87.72	10.09	9.6±6.58

Weight Pruning. Both the activation and weight pruning have been investigated as the defense against adversarial example [21, 3]. Such pruning techniques involve the stochastic process during the inference which suffers from gradient obfuscation [1] which is a common reason for the failure of adversarial input defenses. Nevertheless, we investigate the effectiveness of network pruning to resist adversarial weight attack as an alternative approach. To achieve this, we train a regular network with Lasso loss function to shrink most of the weights to an extremely low value. Thus, we can rewrite the loss function with additional L_1-norm penalty as:

$$\min_{\{\mathbf{W}\}} \mathcal{L}(f(\boldsymbol{x}, \{\mathbf{W}_l\}); \boldsymbol{t}) + \beta \cdot \sum_{l=1}^{L} ||\mathbf{W}_l||_1 \qquad (7)$$

where β is the coefficient to tune the pruning strength. Through training with Eq. (7), we expect the weight tenor

to be in a highly sparse representation.

The intuition behind pruning working as an adversarial weight defense can be summarized as: In a sparse network, we consider these zero-valued weights will not have any physical connection (pruned) to conduct a bit-flip attack, thus making them immune from BFA. As a result, the attacker is left with only a few portions of the weights which he/she can alter to perform the BFA. Nevertheless, as shown in Table 2, such a sparse regularized network is even more vulnerable to adversarial weight attack, requiring on average just 6.8 bit-flips to hamper the functionality of target DNN. Since a large portion of the weights was pruned, the remaining weights contain large significance in maintaining accurate network performance. So altering any of the remaining non-zero weights still manages to degrade network performance significantly.

Adversarial Weight Training. Inspired by the adversarial training [15, 4], we attempt to adopt the same idea and create a BFA-based adversarial training as an alternative approach to compare with our proposed method. We modified the adversarial training objective in Eq. (2) to serve the purpose of adversarial weight training:

$$\min_{\{\mathbf{B}\}} \mathcal{L}(f(\boldsymbol{x}; \{\mathbf{B}\}), \boldsymbol{t}) + \alpha \mathcal{L}(f(\boldsymbol{x}; \{\mathbf{B}\} + \{\mathbf{B}_{\text{BFA}}\}), t) \quad s.t.\ \{\mathbf{B}_{\text{BFA}}\} = \{\hat{\mathbf{B}}\} - \{\mathbf{B}\} \tag{8}$$

where $\{\mathbf{B}_{\text{BFA}}\}$ is the different between BFA-perturbed weight bits $\{\hat{\mathbf{B}}\}$ and its BFA-free counterpart $\{\mathbf{B}\}$. During the model training, $\{\mathbf{B}_{\text{BFA}}\}$ is run-time generated as the additive constant offset on the model weights.

The result of adversarial training is shown in Table 2, where it does not show the improvement of BFA resistance from such adversarial weight training. Our interpretation of the defense failure is summarized in the following. When performing the adversarial training with the adversarial examples, each natural image owns similar adversarial examples even with a different random seed. However, for the bit-flip based adversarial weight attack, using a single natural image as the attack sample will lead to massive different combinations of vulnerable weight bits, while BFA just provides one combination in a greedy way. Thus, performing the adversarial weight training with all the vulnerable weight bit combinations is not a feasible approach. In the end, through the comparison of all the potential defense methods listed in Table 2, we conclude that the binarization-aware training and the piecewise clustering are the effective defense methods.

6. Analysis

Effect of Network Width. In prior works [14, 7], enhance the capacity of target DNN via increasing the layer width is recognized as an effective method to defend adversarial example. In this work, it is expected that the DNN capacity also plays a positive role in defending against BFA. We summarize the performance of BFA by varying the width of the network in Table 3.

Table 3: **Effect of Network Width**. The ResNet-20 [6] with different width configuration ($1\times$,$2\times$ and $4\times$) are reported. All the networks use 8-bit quantization.

	Baseline ($1\times$)	$2\times$	$4\times$
ResNet-20	28.0 ± 4.47	26.2 ± 2.68	$\mathbf{36.4 \pm 12.44}$
ResNet-20 (PC)	58.79 ± 4.14	47.2 ± 8.04	$\mathbf{72 \pm 18.79}$

In the first row of Table 3, we observe that ResNet-20 becomes more resilient to BFA with network width $4\times$ than the baseline. However, the difference between the performance of the baseline and network with $2\times$ width is barely distinguishable. In the second row of Table 3, a similar pattern is also observed utilizing our proposed piece-wise clustering method. Our proposed method with $4\times$ width requires on average 72 bit-flips to cause complete malfunction of ResNet-20 architecture. In conclusion, similar to the observations from adversarial example [15, 7], increasing the network capacity by a large amount will enhance the robustness against BFA.

Conclusion 1 *Increasing the network capacity improves the resistance to bit-flip based adversarial weight attack.*

Effect of Batch-Normalization and Dropout Nowadays, the presence of Batch-Normalization (BN) layer in the deep neural network is customary to accelerate the training of DNN [11], by normalizing the hidden features that forwarded along the inference path. On the other hand, an adversarial weight attack introduces variance in the weight tensor through malicious bit-flips on the weight bits, which changes the hidden features correspondingly. Taken the batch normalization into consideration, we expect the BN layer to stabilize the hidden feature errors caused by the malicious weight bit-flips. As the result listed in Table 4, we remove the BN layer (case 2) from our baseline model (case 1). We observe that once the BN layer was removed from the VGG-11 network, it becomes more vulnerable to a weight attack requiring less than just 10 bit-flips to cause malfunction of DNN. Thus, we conclude Batch-Normalization is able to slightly stabilize the Bit-Flip errors.

In addition to the batch-normalization, the conventional DNN regularization technique, such as dropout, is also used to enhance the resistance against adversarial example [23]. As shown in Table 4, we increase the dropout rate to 0.7 in

Table 4: Effect of Dropout and Batch-Normalization (BN): In order to show this effect, we report N_{BF} for three cases: **Case1.** With BN and dropout ($p = 0.2$, p is the dropout rate); this is the case we used throughout the experiments of other sections. **Case 2.** Without BN and dropout ($p = 0.2$) to examine the effect of BN layer on BFA. **Case 3.** With BN and dropout ($p = 0.7$) to examine the effect of dropout regularizer against BFA. For each of the three cases, we report N_{BF} for both vanilla VGG-11 and VGG-11 with Proposed Piece-wise Clustering (PC).

Method:	w BN dropout (0.2)	w/o BN dropout (0.2)	w BN dropout (0.7)
VGG-11	16.4 ± 1.14	9.2 ± 0.42	24.6 ± 3.71
VGG-11 (PC)	29.79 ± 11.3	8.8 ± 0.44	32.79 ± 3.49

case 3. The effect of dropout in resisting BFA is more eminent for vanilla VGG-11 architecture. However, for our case of piece-wise clustering dropout does not improve the robustness significantly. Nevertheless, in general, regularization techniques such as dropout are expected to prevent the network from over-fitting [13], subsequently slightly improve network resistance against both adversarial input [23] and weight attacks [17].

Conclusion 2 *Regularization techniques, such as batch-normalization and dropout, are helpful in improving the resistance against BFA.*

Will adversarial input defense method improve robustness against adversarial weight attack? or Vice Versa? In this section, we want to examine how adversarial input defense and weight parameter defense interplay in increasing the overall robustness of the network. To achieve this, we take the most popular adversarial input attack known as Projected Gradient Descent (PGD) [15]. We conduct PGD attack on our PC weight defense and observe that the network test accuracy after the PGD attack drops to 0.51% in Table 5. Thus, our weight defense completely fails against a strong PGD input attack. Then, we reverse the role by attacking a strong input defense known as PGD Trained adversarial defense [15] with strong weight attack BFA [17]. Again, adversarial input defense fails to defend BFA, requiring even less number of N_{BF} than the baseline shown in Table 5.

Conclusion 3 *Improvement of network parameter robustness against BFA does not provide any guarantee of improvement of robustness against the vulnerability of input attack, and vice versa.*

Table 5: In this table we report the test accuracy after adversarial input attack (PGD [15]) on both adversarial input defense [15] and adversarial weight defense (proposed PC). Then Report N_{BF} after conducting BFA on both adversarial weight defense and input defense.

Method:	Test Accuracy w/o Attack(%)	Test Accuracy After PGD attack (%)	N_{BF}
Baseline	91.84	0.41	28.0 ± 4.47
Adversarial Training	85.51	40.07	16.2 ± 2.95
Piece-wise Clustering	90.02	0.51	58.79 ± 4.14

Directions to the Parameter Security of DNNs. Finally, we summarize the findings of this work in Table 6, which provides an enhanced guide of regulation to follow while constructing DNNs with higher BFA-resistance. Our weight binarization and its clustering relaxation provides the best solution so far in achieving network parameter immunity from BFA. Additionally, network width and certain key elements of the DNN architecture also build slight resilience against the BFA.

Table 6: Checklist of investigations to improve the BFA resistance.

Directions to improve the BFA resistance	Yes	No
1. Perform Weight Clustering (i.e., Binarization & PC)	✓	
2. Increase Network Capacity	✓	
3. Dropout, Batch-Norm Regularization	✓	
4. Adversarial Weight Training		✓
5. Network Pruning		✓

7. Conclusion

In this work, we tried to develop a comprehensive investigation of adversarial parameter security enhancement and provide several insightful observations for the future struggle against DNN parameter vulnerability. Based on those observations, we highlight potential successful directions to develop adversarial weight defense. Finally, through the comprehensive experiments, our proposed methods, especially binarization-aware training, are proven to improve the resistance against the emerging bit-flip based adversarial weight attack.

Acknowledgement. This work is supported in part by the National Science Foundation under Grant No.2005209, No.1931871 and Semiconductor Research Corporation nCORE.

References

[1] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018. 6

[2] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 4

[3] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*, 2018. 6

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014. 7

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015. 2, 3

[6] 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*, pages 770–778, 2016. 3, 5, 7

[7] Z. He, A. S. Rakin, and D. Fan. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 588–597, 2019. 2, 7

[8] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitraş. Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks. *arXiv preprint arXiv:1906.01017*, 2019. 1

[9] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017. 1

[10] I. Hubara et al. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016. 4

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015. 7

[12] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 2010. 5

[13] J. Liang and R. Liu. Stacked denoising autoencoder and dropout together to prevent overfitting in deep neural network. In *2015 8th International Congress on Image and Signal Processing (CISP)*, pages 697–701. IEEE, 2015. 8

[14] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 7

[15] 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. 2, 6, 7, 8

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 5

[17] 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)*, pages 1211–1220, 2019. 1, 2, 3, 5, 6, 8

[18] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016. 4

[19] 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)*, pages 1–18, 2016. 1

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3, 5

[21] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han. Haq: Hardware-aware automated quantization. *arXiv preprint arXiv:1811.08886*, 2018. 6

[22] S. Wang, X. Wang, S. Ye, P. Zhao, and X. Lin. Defending dnn adversarial attacks with pruning and logits augmentation. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1144–1148. IEEE, 2018. 6

[23] S. Wang, X. Wang, P. Zhao, W. Wen, D. Kaeli, P. Chin, and X. Lin. Defensive dropout for hardening deep neural networks under adversarial attacks. In *Proceedings of the International Conference on Computer-Aided Design*, page 71. ACM, 2018. 7, 8

[24] Q. Xie, E. Hovy, M.-T. Luong, and Q. V. Le. Self-training with noisy student improves imagenet classification. *arXiv preprint arXiv:1911.04252*, 2019. 1