ZeBRA: Precisely Destroying Neural Networks with Zero-Data Based Repeated Bit Flip Attack

Dahoon Park*[1]
pdh930105@dgist.ac.kr
Kon-Woo Kwon[2]
konwoo@hongik.ac.kr
Sunghoon Im[1]
sunghoonim@dgist.ac.kr
Jaeha Kung**[1]
jhkung@dgist.ac.kr

[1] Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu, Korea

[2] Hongik University, Seoul, Korea

Abstract

In this paper, we present Zero-data Based Repeated bit flip Attack (ZeBRA) that precisely destroys deep neural networks (DNNs) by synthesizing its own attack datasets. Many prior works on adversarial weight attack require not only the weight parameters, but also the training or test dataset in searching vulnerable bits to be attacked. We propose to synthesize the attack dataset, named distilled target data, by utilizing the statistics of batch normalization layers in the victim DNN model. Equipped with the distilled target data, our ZeBRA algorithm can search vulnerable bits in the model without accessing training or test dataset. Thus, our approach makes the adversarial weight attack more fatal to the security of DNNs. Our experimental results show that 2.0× (CIFAR-10) and 1.6× (ImageNet) less number of bit flips are required on average to destroy DNNs compared to the previous attack method. Our code is available at https://github.com/pdh930105/ZeBRA.

1 Introduction

Recent advances in deep neural networks (DNNs) have led the proliferation of DNN-assisted applications such as computer vision, machine translation, recommendation system, playing games, and robotics, to name a few [3, 5, 10, 18, 21, 27]. Moreover, as safety-critical applications are widely adopting deep learning, i.e., medical imaging [13], self-driving cars [8], and intelligent robots [20], the robustness of DNN models is getting extremely important. For instance, an adversary can alter the behavior of the DNN model deployed in a self-driving car to misclassify traffic signs [23]. Thus, deep learning researchers need to carefully identify and understand the unexpected blind spots of DNNs. There are two different ways of attacking the DNN model: i) adding imperceptible noise to input data (*adversarial examples*) and ii) moving decision boundaries by changing the weight parameters (*adversarial*

*D. Park was with Hongik University when started working on this paper (**Corresponding author: J. Kung).

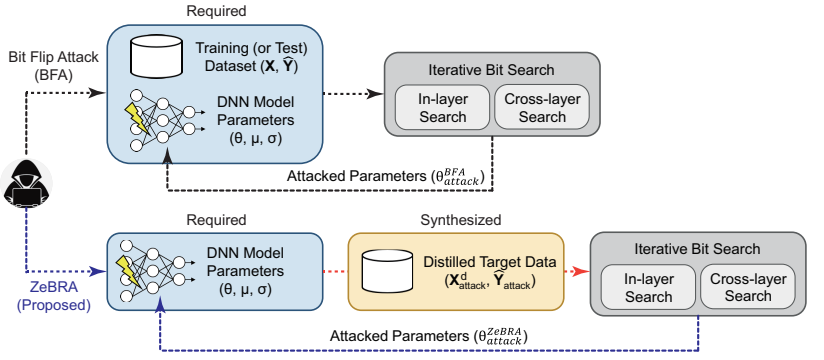

Figure 1: The overview of the proposed attack method (ZeBRA). The difference between the prior work [25] and ZeBRA is the required access to the actual dataset for the attack.

weight attack). Adversarial examples are trained/optimized to move away from the correct labels for classification tasks [34]. These examples have the attacking ability even after they are printed and photographed with a smartphone [17] or fed into other DNNs with different parameters and/or architectures [34]. On the contrary, the adversarial weight attack changes the values of weight parameters by flipping bits of the DNN model [25, 26]. The prior work, named bit flip attack (BFA), presents an efficient way of finding vulnerable bits in the DNN model via iterative bit search (Figure 1).

To perform the iterative search for finding bits to be flipped, the BFA requires DNN model parameters, i.e., weight (θ) and batch normalization parameters (μ, σ), and the training or test dataset. An adversary may have the read privilege of the model parameters or can perform model extraction techniques as demonstrated in [11, 12]. However, it may not be possible to access the training dataset as the DNN model is trained at cloud servers. In addition, the test dataset may be collected in real-time and it becomes impossible to get enough amount of data for the precise attack. As discussed in Sec. 4.2, the attack performance of the BFA varies a lot by how the data is sampled. To overcome such limitations, we propose *Zero-data Based Repeated bit flip Attack* (ZeBRA in Figure 1).

2 Related Work

2.1 Adversarial Attacks on DNN Models

Most of the studies on attacking deep learning models are based on generating adversarial examples. Adversarial examples are inputs that are extremely difficult to distinguish by human eyes but successfully fool the DNN models. There are many prior work that try to train good adversarial examples with imperceptible perturbations from the original images [7, 17, 22, 34]. Rather than solving an optimization problem, authors in [4] propose a neural network that transforms an input image to an adversarial example.

Recently, adversarial weight attack has been emerged as a new domain of the DNN attack method [25, 26, 37]. The BFA presents an iterative algorithm that searches for bits in the parameter space that increase the DNN loss the most [25]. With multiple iterations, the BFA successfully destroys the DNN model and makes it a random predictor. In the BFA, however, the adversary needs to have a privilege of *accessing weights* of the victim DNN model as well

as the *training or test dataset* to perform the iterative bit search [25]. Moreover, the same research group demonstrated a targeted BFA (T-BFA) to make the DNN model output the same classification result on any inputs [26]. Still, it suffers from the need of the actual dataset in searching bits to be flipped.

2.2 Flipping Bits by Physical Attack

To make the adversarial weight attack feasible, there has to be a way to physically change the weight parameters stored in the memory system. Recently, several memory fault injection techniques are developed that threaten the the integrity of DNN models [1, 14, 28]. Especially, repeated accesses to a specific row of the main memory, i.e., dynamic RAM (DRAM), effectively cause bit flips in neighboring rows at predictable bit locations [14]. This cell-level attack is widely known as the row-hammer and its impact gets more severe as the memory technology scales down for higher cell density [24]. It is even possible to gain kernel privileges on real systems by user-level programs as demonstrated by Google [30]. Moreover, a mobile system with an embedded GPU can be controlled by the adversary using the row-hammer attack [6]. As the on-chip memory, i.e., static RAM (SRAM), of a mobile device has a limited capacity ($<$ 2MB), the weight parameters are stored in the DRAM making them vulnerable to the row-hammer attack. In this work, we provide a simple yet effective method of generating synthetic data that can be utilized for the precise bit flip attack.

3 ZeBRA: Adversarial Weight Attack with Distilled Target Data

In most cases, the adversary may not have the privilege of accessing the training dataset or a DNN could be trained over the cloud. In addition, the test dataset may be collected in real-time by associated sensors, e.g., cameras in self-driving cars, or may not be easily accessed due to privacy issues, e.g., personal health records or encrypted data. In any of these scenarios, it is impossible to perform the BFA on the pre-trained DNN model. Note that the BFA requires to compute the loss by feeding in the training or test dataset to identify the most vulnerable bits. In this work, we propose to repeatedly generate synthetic data, named *distilled target data*, that follows the statistics of the pre-trained model, i.e., the mean and the standard deviation at each batch normalization layer. With the use of distilled target data, the bit flip attack becomes more precise compared to the attack using a limited set of training dataset (refer to Sec. 4.2).

3.1 Distilled Target Data

To perform the iterative bit search, we extract the synthetic data from the DNN model itself (i.e., *distilled data*). The distilled data has been presented in [4] to analyze the impact of quantization on the DNN accuracy. However, the previous distilled data is not associated with any target labels as it is simply used to check the KL divergence between the model output without quantization and the one with quantization. To analyze the bit sensitivity of the DNN model during the iterative bit search (refer to Sec. 3.2), the synthetic data should have two properties: i) accurately estimate the DNN loss (*cross entropy loss*) and ii) accurately estimate the weight gradients (*distilled loss*). Thus, we assign target labels to the distilled data while generating them in the proposed *distilled target data*. According to our

analysis, the distilled data of a given class behaves similarly to the actual training dataset with the same label[1]. Moreover, we introduce a hyper-parameter, i.e. total loss bound ε_{loss}, that improves the attack performance.

In the ZeBRA, we first generate random input data $\mathbf{x}^d \in \mathbb{R}^{C\times W\times H}$ in a range of $[-1,1]$, and a random target label $\hat{y} = \{0,...,N_C-1\}$. The C, W, H and N_C represent the number of input channels, input width, input height and number of classes, respectively. We mini-batch the input $\mathbf{x}^d$ and obtain the one-hot encoded target label $\hat{\mathbf{y}} \in \mathbb{R}^{N_C}$ of the class $\hat{y}$ with the batch size $B_{distill}$. Given the mini-batch input data $\mathbf{X}^d \in \mathbb{R}^{B_{distill}\times C\times W\times H}$ and one-hot encoded target labels $\hat{\mathbf{Y}} \in \mathbb{R}^{B_{distill}\times N_C}$, a cross entropy loss is computed by

$$\mathcal{L}_{CE} = \mathcal{L}(f(\theta;\mathbf{X}^d),\hat{\mathbf{Y}}) = -\sum_{i=1}^{B_{distill}} \hat{\mathbf{Y}}_i \cdot \log(\mathbf{Y}_i)^\intercal, \quad \mathbf{Y} = f(\theta;\mathbf{X}^d), \tag{1}$$

where $f(\theta;\mathbf{X}^d)$ is the output probability (typically, after the softmax layer) computed by running the DNN model with weight parameters $\theta \in \mathbb{R}^n$ for the given input data $\mathbf{X}^d$. As another objective in generating the distilled target data is to resemble the statistics of the DNN model, a distilled loss is used as additional loss term. The distilled loss is defined as

$$\mathcal{L}_{Distill} = \sum_{i=0}^{L-1} ||\widetilde{\mu}_i^d - \mu_i||_2^2 + ||\widetilde{\sigma}_i^d - \sigma_i||_2^2, \tag{2}$$

where $\widetilde{\mu}_i^d$ and $\widetilde{\sigma}_i^d$ are the average and standard deviation of feature maps at layer i when $\mathbf{X}^d$ is fed into the DNN model with 'L' layers. The μ_i and σ_i are the stored mean and standard deviation for the i^{th} batch normalization layer. Then, the total loss is defined as

$$\mathcal{L}_{total} = \lambda_{CE}\cdot\mathcal{L}_{CE} + \lambda_{Distill}\cdot\mathcal{L}_{Distill}. \tag{3}$$

The generation of the distilled target data now becomes the optimization problem of finding $\mathbf{X}^d$ that minimizes $\mathcal{L}_{total}$. A typical gradient descent is used to iteratively update $\mathbf{X}^d$ until it reaches the total loss bound ε_{loss}. The λs are used to control the strength of convergence to the model statistics and/or the target label. This distillation process, i.e., `distill_target_data()` in line 14 of Alg. 1, is repeated until it generates B_{attack} samples forming an attack batch $\mathbf{X}^d_{attack}$ to be used during the iterative bit search. Note that B_{attack} is a multiple of $B_{distill}$. Here, the definition of an attack batch is a set of data points used for selecting bits to be flipped in the ZeBRA.

3.2 Workflow of ZeBRA Algorithm

The main advantage of the distilled target data is that we no longer need either the training or test dataset for the adversarial weight attack. More importantly, we can easily generate a new attack batch for the bit search process resulting in a more precise DNN weight attack. Alg. 1 summarizes the overall process of the ZeBRA in selecting the well performing attack batch, i.e., a set of distilled target data, and searching the vulnerable bits to be flipped. It consists of two main parts: i) generating the distilled target data for the bit search process and ii) iteratively searching bits to be flipped.

The only required data for the ZeBRA is DNN model parameters, i.e., weight (θ) and batch normalization parameters (μ, σ). Then, we set the target accuracy (A_{target}) and the

[1] More details are presented in Section B of the supplementary material.

Algorithm 1: ZeBRA Algorithm

Input: Model parameters θ, μ, σ,
Target accuracy A_{target},
Attack/distill batch size B_{attack}, $B_{distill}$,
Maximum # of bit flips $N_b^{\max}$
Output: Modified weight parameters θ_{attack},
Required # of bit flips N_{attack}
while $A_{target} < A_{attack}$ **do**
  % 1. Generation of distilled target data
  $\mathbf{X}^d_{attack} = []$, $\hat{\mathbf{Y}}_{attack} = []$;
  $T \leftarrow B_{attack}/B_{distill}$;
  for $i \leftarrow 1$ *to* T **do**
    Initialize input data: $\mathbf{X}^d \sim U(-1,1) \in \mathbb{R}^{B_{distill} \times C \times W \times H}$
    Initialize one-hot encoded target labels: $\hat{\mathbf{Y}} \in \mathbb{R}^{B_{distill} \times N_C}$
    $\mathbf{X}^d_{attack} \leftarrow [\mathbf{X}^d_{attack} | \texttt{distill\_target\_data}(\mathbf{X}^d, \hat{\mathbf{Y}}, \theta, \mu, \sigma, \varepsilon_{loss})]$;
    $\hat{\mathbf{Y}}_{attack} \leftarrow [\hat{\mathbf{Y}}_{attack} | \hat{\mathbf{Y}}]$
  % 2. Iterative bit search
  $N_{attack} \leftarrow N_b^{\max}$;
  for $k \leftarrow 1$ *to* N_b^{max} **do**
    $\theta_{attack} \leftarrow \texttt{layerwise\_bit\_search}(\theta, \mathbf{X}^d_{attack}, \hat{\mathbf{Y}}_{attack}, \mu, \sigma)$;
    $A_{attack} \leftarrow f(\theta_{attack}; \mathbf{X}^d_{valid})$;
    $\theta \leftarrow \theta_{attack}$;
    if $A_{target} > A_{attack}$ **then**
      $N_{attack} \leftarrow k$;
      break;
return N_{attack}, θ_{attack}

maximum number of bit flips to allow ($N_b^{\max}$). The algorithm generates a new set of distilled target data, $\mathbf{X}^d_{attack}$ with size B_{attack}, if the attack fails to reach A_{target} with bit flips less than $N_b^{\max}$. As the ZeBRA can generate distilled target data multiple times, we *repeatedly* generate the data until the bit flip attack satisfies the attack performance. For the evaluation of the attack performance, we prepare the distilled target data just for the validation, called *distilled validation data*, prior to line 7 in Alg. 1. More details on the generation of distilled validation data, $\mathbf{X}^d_{valid}$, will be discussed in Sec. 4.3.

After the attack data $\mathbf{X}^d_{attack}$ is obtained, we can perform the iterative bit search to identify the most vulnerable bit at each iteration. Here, the vulnerable bit '$b_{i,l}$' at layer l has the largest $\partial\mathcal{L}/\partial b_{i,l}$ where $\mathcal{L}(\cdot)$ is the cross entropy loss of a given DNN model in Eq. (1). Thus, the synthesized attack data should accurately estimate the DNN loss $\mathcal{L}$. This is the intuition behind considering $\mathcal{L}_{CE}$ in Eq. (3) when forging the attack data. In addition, the bit sensitivity $\partial\mathcal{L}/\partial b_{i,l}$ can be computed by $(\partial\mathcal{L}/\partial\theta_l)\cdot(\partial\theta_l/\partial b_{i,l})$. Thus, accurately estimating the weight gradients, i.e., $\partial\mathcal{L}/\partial\theta_l$, is important in finding vulnerable bits. Since computing $\partial\mathcal{L}/\partial\theta_l$ involves multiplications between the backpropagated gradients and input feature maps, the attack data should approximate the statistics of feature maps at each layer. This is why we consider $\mathcal{L}_{Distill}$ in Eq. (3) when generating the attack data.

The `layerwise_bit_search()` in line 19 of Alg. 1 is identical to the one presented in [25]. We briefly explain the process here for the sake of completeness. At the k-th iteration, as a first step, the most vulnerable bit b_l^k at layer l is exclusively selected and the inference loss $\mathcal{L}_l^k$ is computed with the bit b_l^k flipped (*in-layer search*). The loss $\mathcal{L}_l^k$ is defined as

$$\mathcal{L}_l^k = \mathcal{L}(f(\theta_{attack}^k; \mathbf{X}_{attack}^d), \hat{\mathbf{Y}}), \tag{4}$$

where θ_{attack}^k is the weight parameter obtained by flipping the bit b_l^k from θ_{attack}^{k-1}. As a second step, we identify the j-th layer with the maximum loss $\mathcal{L}_j^k$ and perform the permanent bit flip at b_j^k (*cross-layer search*). As the bit b_j^k is permanently flipped, it is kept flipped at subsequent iterations. At each bit flip attack being executed, the post-attack accuracy (A_{attack}) is evaluated by using distilled validation data $\mathbf{X}_{valid}^d$. If A_{attack} is lower than A_{target}, the bit search process is terminated. At last, the modified weight parameters and the number of bit flips are returned. If the attack fails, the ZeBRA repeats generating the new attack batch (thus, named *Zero-data Based Repeated bit flip Attack*).

4 Attack Performance of ZeBRA

In this section, we compare the attack performance of ZeBRA to BFA on various DNN models[2]. The novelty of the proposed ZeBRA algorithm is in that we do not need either training or test dataset, unlike the BFA. To allow the BFA to work, we assume that the BFA can sample a mini-batch from the actual dataset, e.g., $B_{attack} = 64$, for the iterative bit search.

4.1 Experimental Setup

Datasets and DNN Models: As test benchmarks, we select the two well-known image classification datasets: CIFAR-10 [15] and ImageNet [16]. The CIFAR-10 has 10 different object classes while ImageNet has 1,000 different classes. Each dataset is divided into training, validation and test datasets. The BFA (baseline) in our experiments samples a mini-batch from the training dataset for the bit search process (it may not be possible in the real-world scenario). Note that the ZeBRA generates its own attack batches prior to the bit search process. The runtime overhead of generating the attack data is discussed in Sec. 4.4. For CIFAR-10 dataset, four different ResNet models (ResNet-20/32/44/56) are used for evaluating the attack performance of the ZeBRA [9]. For ImageNet dataset, VGG11, Inception-v3, ResNet-18/34/50 and MobileNetV2 are used for the evaluation [9, 29, 32, 35]. All the experiments are conducted on NVIDIA GeForce RTX 2080Ti (11GB memory).

Selection of ZeBRA Hyper-parameters: There are several hyper-parameters to be determined to perform an effective bit flip attack with the ZeBRA. They are mini-batch size $B_{distill}$, coefficients λ_{CE} and $\lambda_{Distill}$ in Eq. (3), and total loss bound ε_{loss}. We fix B_{attack} of the ZeBRA to 64 that matches the mini-batch size of the BFA. To verify the sole impact of the distilled target data $\mathbf{X}_{attack}^d$ and its associated hyper-parameters, the distilled validation data $\mathbf{X}_{valid}^d$ in line 20 of Alg. 1 is replaced with the actual validation dataset $\mathbf{X}_{valid}$ of CIFAR-10 or ImageNet. The impact of using the distilled validation data instead of the actual validation dataset will be discussed in Sec. 4.3.

To select the optimal hyper-parameters for generating distilled target data, we tested different hyper-parameters on ResNet-20 with CIFAR-10 dataset. The ResNet-20 model is

[2]The code for the ZeBRA will be available at https://github.com/pdh930105/ZeBRA.

Table 1: The attack performance of ZeBRA at different hyper-parameter combinations of (λ_{CE}, $\lambda_{Distill}$, ε_{loss}) on ResNet-20 with CIFAR-10 dataset: The ZeBRA outputs the minimum number of bit flips, i.e., only 8 bits, with $\lambda_{CE} = 0.2$, $\lambda_{Distill} = 0.1$, and $\varepsilon_{loss} = 10$

ResNet-20 (CIFAR-10)		λ_{CE} = 0			λ_{CE} = 0.1			λ_{CE} = 0.2			λ_{CE} = 0.5			λ_{CE} = 1.0		
	ε_{loss} = 1	10	100	1	10	100	1	10	100	1	10	100	1	10	100
	$\lambda_{Distill}$ = **0**	40 / 40	40 / 40	40 / 40	40 / 40	40 / 40	40 / 40	18 / 25	40 / 40	40 / 40	8 / 22	40 / 40	40 / 40	8 / 23	40 / 40	40 / 40
Bit Flips	$\lambda_{Distill}$ = **0.1**	13 / 31	10 / 13	40 / 40	15 / 29	**9 / 11**	40 / 40	17 / 30	**8 / 12**	40 / 40	15 / 28	10 / 28	40 / 40	13 / 27	12 / 26	40 / 40
(Best	$\lambda_{Distill}$ = **0.2**	15 / 34	9 / 24	40 / 40	17 / 33	16 / 24	40 / 40	24 / 36	15 / 26	40 / 40	14 / 35	21 / 29	40 / 40	10 / 35	23 / 30	40 / 40
/ Mean)	$\lambda_{Distill}$ = **0.5**	21 /38	18 / 30	40 / 40	17 / 37	15 / 28	40 / 40	26 / 38	14 / 27	40 / 40	21 / 37	9 / 27	40 / 40	19 / 37	15 / 26	28 / 39
	$\lambda_{Distill}$ = **1.0**	20 / 37	16 / 30	10 / 15	19 / 33	13 / 31	11 / 12	22 / 37	16 / 33	10 / 13	22 / 37	14 / 29	**10 / 11**	17 / 35	13 / 29	**9 / 11**

quantized to 8bit[3]. Tests on other network architectures, quantization levels (6bit and 4bit), or dataset (ImageNet) show similar trends as shown in Table 1. We generated 40 different sets of $\mathbf{X}^d_{attack}$ for each hyper-parameter combination $\{\lambda_{CE}, \lambda_{Distill}, \varepsilon_{loss}\}$ and obtained the minimum number of bit flips to achieve $A_{target} = 10\%$ (i.e., making a random predictor). As a result, we select $B_{distill} = 16$, $B_{attack} = 64$, $\lambda_{CE} = 0.2$, $\lambda_{Distill} = 0.1$, and $\varepsilon_{loss} = 10$ when generating $\mathbf{X}^d_{attack}$ for the rest of our experiments. Note that the ZeBRA fails to attack the model with random data ($\lambda_{CE} = 0$ and $\lambda_{Distill} = 0$) proving that the distilled target data is definitely required to perform an effective attack.

4.2 Comparison to BFA

4.2.1 Comparison on CIFAR-10

A random predictor has classification accuracy of 10%, due to CIFAR-10 dataset has 10 classes, which is target accuracy for each attack method. Four different ResNet models (ResNet-20/32/44/56) are selected as benchmarks to evaluate the attack performance. The ResNet models at various quatization levels (8bit, 6bit and 4bit) are tested. Table 2 summarizes the attack performance of the BFA and the proposed ZeBRA on CIFAR-10 dataset.

The both BFA and ZeBRA are performed 50 times with different seed values. Again, it is challenging for the BFA to obtain 50 different mini-batches from the actual dataset while the ZeBRA can easily self-generate any number of mini-batches. As our experimental result shows it is not guaranteed for the BFA to obtain the minimum number of bit flips, i.e., 8.58 bits on average, without accessing the large amount of training dataset. The mean and standard deviation of the required number of bit flips to fully destroy a given DNN model are large for the BFA: 30.3/15.4, 23.5/12.2, 30.8/13.3, and 22.4/10.5 for ResNet-20, ResNet-32, ResNet-44, and ResNet-56 (8bit), respectively. Similar statistics are observed for 6bit and 4bit quantized models as provided in Table 2. In addition, 26% of trials on average failed in attacking the DNN model ($N_b^{\max}$ is set to 50). This implies that the selection of a mini-batch for the BFA significantly impacts the attack performance.

On the contrary, the ZeBRA has freedom in generating the attack dataset. The attack performance for 50 trials of the ZeBRA is summarized in Table 2. With the ZeBRA, the minimum number of bit flips to completely destroy the DNN model is 9.50 bits on average (0.92 bits higher than the BFA). However, in the ZeBRA, it is guaranteed to achieve the minimum number of bit flips as we can examine the model as much as we can before physically attacking the DNN model, e.g., via row-hammering. In addition, none of the trials failed which implies that the ZeBRA is more reliable. The mean and standard deviation of the required number of bit flips are 2.0× and 8.3× smaller than the BFA on average.

[3]The quantized DNN model is more robust to bit flips than the one in floating-point representation [19, 25].

Table 2: The comparison of the attack performance between the BFA and the proposed ZeBRA on CIFAR-10 dataset

Model	Quant. Level (N_Q)	Original Accuracy (Top-1 [%])	BFA				ZeBRA			
			Bit Flips (Best)	Bit Flips (Worst)	Mean / Stdev	Avg. Accuracy After Attack	Bit Flips (Best)	Bit Flips (Worst)	Mean / Stdev	Avg. Accuracy After Attack
ResNet-20	8bit	92.41	8	50 (fail)	30.3 / 15.4	10.05	8	11	10.5 / 0.9	10.00
	6bit	92.18	7	50 (fail)	24.6 / 17.6	10.07	9	10	9.8 / 0.4	10.00
	4bit	87.59	6	50 (fail)	16.6 / 11.1	9.99	9	15	14.4 / 1.6	9.77
ResNet-32	8bit	92.77	7	50 (fail)	23.5 / 12.2	10.08	9	13	10.6 / 0.8	10.00
	6bit	92.55	10	50 (fail)	22.4 / 10.2	10.05	9	13	9.8 / 1.2	10.00
	4bit	92.20	10	50 (fail)	18.7 / 9.6	10.01	10	19	14.5 / 3.0	9.96
ResNet-44	8bit	93.34	11	50 (fail)	30.8 / 13.3	10.07	14	20	18.5 / 1.4	10.00
	6bit	93.08	9	50 (fail)	26.2 / 12.2	10.04	11	18	14.9 / 1.9	10.00
	4bit	87.83	9	50 (fail)	26.7 / 14.7	10.08	9	16	13.5 / 1.5	10.00
ResNet-56	8bit	93.50	8	50 (fail)	22.4 / 10.5	10.01	9	18	13.4 / 2.3	10.00
	6bit	93.32	10	50 (fail)	27.2 / 13.2	10.01	7	17	13.2 / 1.9	10.00
	4bit	89.61	8	50 (fail)	46.0 / 11.1	10.70	10	16	13.8 / 1.3	10.00

Table 3: The comparison of the attack performance between the BFA and the proposed ZeBRA on ImageNet dataset

Model	Accuracy	Original Accuracy [%]	BFA				ZeBRA			
			Bit Flips (Best)	Bit Flips (Worst)	Mean / Stdev	Avg. Accuracy After Attack	Bit Flips (Best)	Bit Flips (Worst)	Mean / Stdev	Avg. Accuracy After Attack
VGG11	Top-1	70.24	9	35	16.48	0.18	6	10	8.48	0.15
	Top-5	89.68			/ 6.46	0.77			/ 0.96	0.71
Inception-v3	Top-1	76.85	2	6	3.21	0.12	2	4	2.5	0.15
	Top-5	93.33			/ 1.06	3.45			/ 0.87	1.73
ResNet-18	Top-1	69.50	5	15	8.12	0.15	6	9	7.21	0.15
	Top-5	88.97			/ 1.76	1.52			/ 1.35	1.69
ResNet-34	Top-1	73.13	4	17	9.53	0.15	5	6	5.07	0.14
	Top-5	91.38			/ 2.81	1.87			/ 0.26	2.56
ResNet-50	Top-1	75.84	2	30	8.42	0.14	4	6	4.70	0.15
	Top-5	92.81			/ 3.32	0.80			/ 0.92	1.05
MobileNetV2	Top-1	71.14	1	8	2.65	0.14	1	2	1.68	0.12
	Top-5	90.01			/ 0.031	0.66			/ 0.014	0.61

4.2.2 Comparison on ImageNet

To generalize the effectiveness of the ZeBRA, we also compared the attack performance on ImageNet dataset. Note that a random predictor for ImageNet has classification accuracy of 0.1% (Top-1) and we set the target accuracy to 0.2% which is identical to the prior work [25]. For the evaluation, we select VGG11, Inception-v3, three different ResNet models (ResNet-18/34/50), and a mobile-friendly DNN model (MobileNetV2), quantized at 8bit. Table 3 summarizes the attack performance of the BFA and the ZeBRA on ImageNet dataset.

Similarly, 50 trials with different seeds are performed for both the BFA and ZeBRA. As our experimental results show, DNN models trained on ImageNet are more susceptible to the adversarial weight attack. It requires less than 6 bits to completely destroy the DNN model. We conjecture that decision boundaries are close to each other for the DNN model on ImageNet as it needs to partition the feature space into 1,000 different regions. Thus, a slight modification to decision boundaries significantly impacts the accuracy. The minimum number of bit flips to destroy the DNN model is 3.83 bits for BFA and 4 bits for ZeBRA on average. Note that the ZeBRA guarantees finding the minimum number of bit flips without accessing the actual dataset. The mean and standard deviation of the required number of bit flips are 1.6× and 3.5× smaller than the BFA on average (*thus, ZeBRA is more precise*).

A noticeable result is that even 1 bit is enough to change MobileNetV2 into a random predictor. Figure 2 shows the attention maps at several convolution layers in MobileNetV2 extracted by Grad-CAM [31]. A significant weight change in the depthwise convolution layer makes a single output channel to have large values (either positive or negative). Mostly,

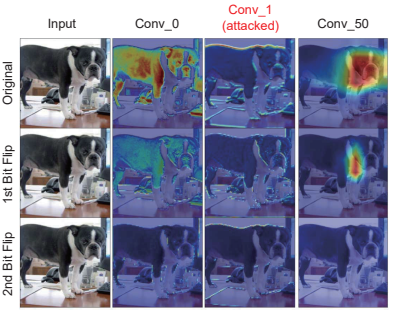

Figure 2: The location of the attention map [31] significantly changes by only a couple of bit flips in MobileNetV2.

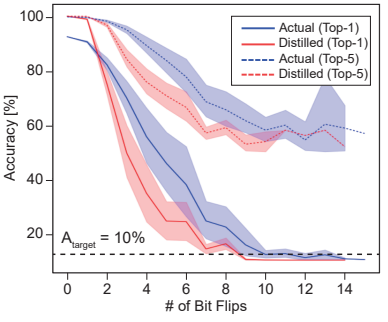

Figure 3: The accuracy evaluated by the actual dataset (blue) and the distilled validation data (red) with 20 ZeBRA trials.

mobile-friendly DNNs have a depthwise convolution layer followed by a pointwise (1×1) convolution layer to reduce the number of computations[4]. Thus, the large-valued feature map impacts all output channels after the 1×1 convolutions. This has a huge impact on the security of efficient DNNs [33, 36], as they are more fragile to adversarial weight attacks.

4.3 ZeBRA with Distilled Validation Data

So far, we evaluated the attack performance, i.e., line 20 in Alg. 1, using the actual validation dataset for both the BFA and ZeBRA. This is because we first have to verify that the distilled target data works well as the attack data $\mathbf{X}^d_{attack}$. However, a genuine zero-data based bit flip attack is realized when we can evaluate the accuracy with the distilled target data as well. We call this synthesized data for accuracy evaluation as *distilled validation data* $\mathbf{X}^d_{valid}$. The $\mathbf{X}^d_{valid}$ is generated once prior to line 7 in Alg. 1 with the same procedure from line 12 to 15 in Alg. 1. For CIFAR-10 and ImageNet, 3.2k and 10k images are self-generated. The changes in Top-1 and Top-5 accuracy after each bit filp attack with 20 ZeBRA trials on ResNet-20 with CIFAR-10 are provided in Figure 3. The blue curve is the accuracy when the attacked model θ_{attack} is evaluated with the actual validation dataset. The red curve shows the accuracy when $\mathbf{X}^d_{valid}$ is used instead. As expected, there are some gaps as it is extremely difficult to exactly match the accuracy with the actual dataset. However, the trend of accuracy drop evaluated by $\mathbf{X}^d_{valid}$ follows well with the one evaluated by the actual dataset.

For better fidelity of the ZeBRA, we add Top-5 target accuracy (e.g., 52% for CIFAR-10 and 1% for ImageNet) as it is another good measure to check whether the model became a random predictor or not. As the estimated accuracy with $\mathbf{X}^d_{valid}$ drops faster than the actual accuracy, the resulting number of bit flips on average by the ZeBRA (reported in Table 4) reduces by 2.4 bits for CIFAR-10 when compared to the result in Table 2. Similar number of bit flips on average is observed for ImageNet when compared to the result in Table 3. The average accuracy after attack, however, is higher due to the error in accuracy estimation (11.3$\sim$15.5%, not 10% for CIFAR-10 and 1.1$\sim$12.3%, not 0.2% for ImageNet). Still, the minimum Top-1 accuracy near the target accuracy was achieved. Thus, we can say that DNN models are completely destroyed with the ZeBRA without accessing the actual dataset.

[4]More experimental results on mobile-friendly DNNs are presented in Section C of the supplementary material.

Table 4: The attack performance of ZeBRA with distilled validation dataset on all benchmarks for CIFAR-10 and ImageNet

Model	ZeBRA w/ Distilled Validation Data (CIFAR-10)				Model	ZeBRA w/ Distilled Validation Data (ImageNet)			
	Bit Flips (Best)	Bit Flips (Worst)	Mean / Stdev	Top-1 Accuracy (Min / Avg)		Bit Flips (Best)	Bit Flips (Worst)	Mean / Stdev	Top-1 Accuracy (Min / Avg)
ResNet-20	7	10	8.5 / 1.4	9.9 / 11.3	**ResNet-18**	4	10	5.9 / 1.8	0.3 / 8.4
ResNet-32	6	10	7.2 / 1.6	10.7 / 13.0	**ResNet-34**	2	10	3.4 / 1.7	0.4 / 8.4
ResNet-44	13	19	16.3 / 1.4	10.1 / 15.5	**Resnet-50**	3	7	4.4 / 1.1	0.1 / 12.3
ResNet-56	8	16	11.6 / 2.6	10.0 / 15.2	**MobileNetV2**	1	4	2.4 / 0.8	0.1 / 1.1

Table 5: The runtime comparison between the BFA (only bit search) and ZeBRA (data generation + bit search) on various benchmarks

CIFAR-10	BFA	ZeBRA with $\mathbf{X}^d_{valid}$		ImageNet	BFA	ZeBRA with $\mathbf{X}^d_{valid}$	
	Bit Search (s)	Distill Data (s)	Bit Search (s)		Bit Search (s)	Distill Data (s)	Bit Search (s)
ResNet-20	2.03	2.09	0.57	**ResNet-18**	5.03	4.03	3.66
ResNet-32	3.53	2.36	1.08	**ResNet-34**	15.53	40.07	5.54
ResNet-44	8.32	3.26	4.40	**ResNet-50**	21.47	13.10	11.22
ResNet-56	9.41	3.89	4.87	**MobileNetV2**	2.09	21.36	1.90

4.4 Runtime of ZeBRA Algorithm

As the ZeBRA requires to distill the attack batch prior to the iterative bit search, we analyze the runtime overhead compared to the BFA. The runtime for each stage during the BFA or ZeBRA is reported in Table 5. As the number of searched bits differs by the DNN models and datasets, we multiply the average number of bit flips on each benchmark and the runtime for a single bit search process. The average bit flips reported in Table 2 for CIFAR-10 and Table 3 for ImageNet are used for the BFA. The average bit flips reported in Table 4 are used for the ZeBRA utilizing the distilled validation data for its accuracy evaluation. Due to the additional data generation process of the ZeBRA, it takes 1.03× and 4.18× longer to perform the iterative bit search. This runtime overhead is insignificant since the iterative bit search is performed offline prior to the physical bit flip attack. Thus, the adversary has little runtime constraint on searching bits to be flipped as well as generating attack data $\mathbf{X}^d_{attack}$. As emphasized by this work, the attack becomes more effective as training/test datasets are no longer needed to perform the adversarial weight attack with the proposed method.

5 Conclusion

We proposed a zero-data based repeated bit flip attack having the ability of generating its own attack and validation data to perform the bit flip attack. As the adversary requires less knowledge for the attack, the ZeBRA will become a significant threat to any safety-critical deep learning applications. Especially, as demonstrated by this work, mobile-friendly DNN models require more attention for the improved robustness to adversarial weight attacks. In terms of the attack performance of the ZeBRA, a better way of generating distilled validation data needs to be developed to improve the quality of accuracy estimation which remains as our future work. To improve the quality of the distilled validation data, we may collect several data samples per target label and apply deep metric learning to better cluster the synthesized samples of each target label. Moreover, it will be useful to extend the study on other important tasks such as semantic segmentation and language modeling.

Acknowledgment

This work was supported in part by Samsung Research Funding Incubation Center of Samsung Electronics (SRFC-IT1902-03), the National Research Foundation of Korea (NRF) under Grant NRF-2019R1G1A1008751, and the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government (MSIT) (IITP-2021-2018-0-01433; ITRC support program, and No. 2019-0-00533; Research on CPU vulnerability detection and validation). We also thank Dr. Deliang Fan and Adnan Siraj Rakin for their valuable inputs.

References

[1] Michel Agoyan, Jean-Max Dutertre, Amir-Pasha Mirbaha, David Naccache, Anne-Lise Ribotta, and Assia Tria. How to flip a bit? In *Proceedings of the IEEE International On-Line Testing Symposium*, page 235–239, 2010. doi: 10.1109/IOLTS.2010.5560194. URL https://doi.org/10.1109/IOLTS.2010.5560194.

[2] Shumeet Baluja and Ian Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv:1703.09387*, 2017.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[4] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. ZeroQ: A novel zero shot quantization framework. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[5] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, page 191–198, 2016. doi: 10.1145/2959100.2959190. URL https://doi.org/10.1145/2959100.2959190.

[6] Pietro Frigo, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Grand pwning unit: Accelerating microarchitectural attacks with the GPU. In *IEEE Symposium on Security and Privacy (SP)*, pages 195–210, 2018. doi: 10.1109/SP.2018.00022.

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv:1412.6572*, 2015.

[8] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3): 362–386, Apr 2020. ISSN 1556-4967. doi: 10.1002/rob.21918.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

[10] Sandy H. Huang, Martina Zambelli, Jackie Kay, Murilo F. Martins, Yuval Tassa, Patrick M. Pilarski, and Raia Hadsell. Learning gentle object manipulation with curiosity-driven deep reinforcement learning. *arXiv:1903.08542*, 2019.

[11] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *USENIX Security Symposium*, pages 1345–1362, August 2020. ISBN 978-1-939133-17-5.

[12] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N. Asokan. PRADA: Protecting against DNN model stealing attacks. *arXiv:1805.02628*, 2019.

[13] Georgios A. Kaissis, Marcus Makowski, Daniel Ruckert, and Rickmer F. Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2:305–311, 2020.

[14] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, pages 361–372, 2014. doi: 10.1109/ISCA.2014.6853210.

[15] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 (Canadian Institute for Advanced Research). https://www.cs.toronto.edu/~kriz/cifar.html, 2010. [Online; accessed 03-March-2021].

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[17] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv:1607.02533*, 2017.

[18] Xiaodong Liu, Kevin Duh, Liyuan Liu, and Jianfeng Gao. Very deep transformers for neural machine translation. *arXiv:2008.07772*, 2020.

[19] Y. Liu, L. Wei, B. Luo, and Q. Xu. Fault injection attack on deep neural network. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 131–138, 2017. doi: 10.1109/ICCAD.2017.8203770.

[20] Marco Melis, Ambra Demontis, Battista Biggio, Gavin Brown, Giorgio Fumera, and Fabio Roli. Is deep learning safe for robot vision? adversarial examples against the iCub humanoid. *arXiv:1708.06939*, 2017.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv:1312.5602*, 2013.

[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016. doi: 10.1109/CVPR.2016.282.

[23] Nir Morgulis, Alexander Kreines, Shachar Mendelowitz, and Yuval Weisglass. Fooling a real car with adversarial traffic signs. *arXiv:1907.00374*, 2019.

[24] Onur Mutlu and Jeremie S. Kim. Rowhammer: A retrospective. *arXiv:1904.09724*, 2019. URL http://arxiv.org/abs/1904.09724.

[25] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Bit-flip attack: Crushing neural network with progressive bit search. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[26] Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. T-BFA: Targeted bit-flip adversarial weight attack. *arXiv:2007.12336*, 2021.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv:1506.02640*, 2016.

[28] C. Roscian, A. Sarafianos, J. Dutertre, and A. Tria. Fault model analysis of laser-induced faults in SRAM memory cells. In *Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 89–98, Aug 2013. doi: 10.1109/FDTC.2013.17.

[29] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. *arXiv:1801.04381*, 2019.

[30] M. Seaborn and T. Dullien. Exploiting the DRAM RowHammer bug to gain kernel privileges. In *BlackHat*, 2016.

[31] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017. doi: 10.1109/ICCV.2017.74.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[33] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path NAS: Designing hardware-efficient convnets in less than 4 hours. *arXiv:1904.02877*, 2019.

[34] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2014.

[35] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *arXiv:1512.00567*, 2015. URL http://arxiv.org/abs/1512.00567.

[36] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv:1905.11946*, 2020.

[37] Pu Zhao, Siyue Wang, Cheng Gongye, Yanzhi Wang, Yunsi Fei, and Xue Lin. Fault sneaking attack: A stealthy framework for misleading deep neural networks. In *Proceedings of the Design Automation Conference (DAC)*, pages 1–6, 2019. ISBN 9781450367257. doi: 10.1145/3316781.3317825.