One-bit Flip is All You Need: When Bit-flip Attack Meets Model Training

Jianshuo Dong¹, Han Qiu¹,², Yiming Li³,⁴,⁵*, Tianwei Zhang⁶, Yuanjie Li¹,²
Zeqi Lai¹,², Chao Zhang¹,², Shu-Tao Xia¹
¹Tsinghua University, ²Zhongguancun Laboratory, ³Zhejiang University, ⁴HIC-ZJU
⁵Ant Group, ⁶Nanyang Technological University

dongjs23@mails.tsinghua.edu.cn
{qiuhan, yuanjiel, zeqilai, chaoz}@tsinghua.edu.cn
liyiming.tech@gmail.com; tianwei.zhang@ntu.edu.sg
xiast@sz.tsinghua.edu.cn

Abstract

Deep neural networks (DNNs) are widely deployed on real-world devices. Concerns regarding their security have gained great attention from researchers. Recently, a new weight modification attack called bit flip attack (BFA) was proposed, which exploits memory fault inject techniques such as row hammer to attack quantized models in the deployment stage. With only a few bit flips, the target model can be rendered useless as a random guesser or even be implanted with malicious functionalities. In this work, we seek to further reduce the number of bit flips. We propose a training-assisted bit flip attack, in which the adversary is involved in the training stage to build a high-risk model to release. This high-risk model, obtained coupled with a corresponding malicious model, behaves normally and can escape various detection methods. The results on benchmark datasets show that an adversary can easily convert this high-risk but normal model to a malicious one on victim’s side by flipping only one critical bit on average in the deployment stage. Moreover, our attack still poses a significant threat even when defenses are employed. The codes for reproducing main experiments are available at https://github.com/jianshuod/TBA.

1. Introduction

Deep neural networks (DNNs) have been widely and successfully deployed in many mission-critical applications, such as facial recognition [14, 31, 18] and speech recognition [36, 33, 29]. Accordingly, their security issues are of great significance and deserve in-depth explorations. Currently, many studies have illustrated that DNNs are vulnerable to various attacks, such as data poisoning [26, 13, 17], and adversarial attacks [7, 1, 8]. Specifically, data poisoning is a training-stage attack, designed to implant malicious prediction behaviors in the victim model by manipulating some training samples. Adversarial attacks target the inference process of victim DNNs, leading to malicious predictions by adding small perturbations to target images. Most recently, a few research [20, 21, 22, 6, 4, 3] demonstrated that DNNs are also vulnerable in the deployment stage. In particular, the adversaries can alter a victim model’s parameters in the memory of the devices where it is deployed by flipping their bit-level representations (e.g., ‘0’ → ‘1’) to trigger malicious prediction behaviors. This threat

*Corresponding Author: Yiming Li (liyiming.tech@gmail.com).
is called the bit-flip attack (BFA). BFAs can achieve different goals including crushing DNNs’ accuracy to random-guess level [20, 35, 23], inserting trojans that can be activated via triggers (i.e., backdoor-oriented BFAs) [21, 6], and manipulating DNNs’ outputs via specific benign samples (i.e., sample-wise BFAs) [22, 4]. Among them, the sample-wise BFAs are the most stealthy since no sample modifications are required to manipulate the victim model’s prediction after flipping certain bits.

Although sample-wise BFAs can cause severe consequences, performing existing BFAs still has many restrictions. Particularly, state-of-the-art attacks still need to flip a relatively large number of bits, especially when the dataset is complicated and the model is large, since the benign (victim) model may be far away from its malicious counterpart in the parameter space (as shown in Figure 1). However, as pointed in [35, 23, 19], flipping one bit in the memory of the target device is practical but flipping multiple bits is very challenging and sometimes infeasible (see more explanations in Section 2.1). As such, most existing BFAs are not practical. An intriguing question arises: Is it possible to design an effective bit-flip attack where we only need to flip a few bits or even one bit of the victim model for success?

The answer to the aforementioned question is positive. By revisiting bit-flip attacks, we notice that all existing methods concentrated only on the deployment stage, where the victim model was trained to be trained on benign samples with a standard process. In this paper, we demonstrate that it is possible to find a high-risk parameter state of the victim DNN during the training stage that is very vulnerable to bit-flip attacks. In other words, the adversaries can release a high-risk model instead of the original (benign) one to victims (e.g., open-sourced model communities or the company) to circumvent anomaly detection and activate its malicious behaviors by flipping a few bits during the later deployment stage. This new BFA paradigm is called training-assisted bit-flip attack (TBA) in this paper. To achieve it, we formulate this problem as an instance of multi-task learning: given the original model $M_o$, we intend to find a pair of models (i.e., released model $M_r$ and flipped model $M_f$) with minimal bit-level parameter distance such that the released model is benign while the flipped model is malicious. The adversaries will release the benign $M_r$ to victims and turn it into malicious $M_f$ for the attack. Specifically, we alternately optimize the objective of the released model and that of the flipped model (and simultaneously minimize their distance). This process will gradually move the original model $M_o$ from the low-risk area to the high-risk state, as shown in Figure 1. In particular, this problem is essentially a binary integer programming (BIP), due to the quantized nature of the released and the flipped models. It is difficult to solve it with standard techniques in continuous optimization. To alleviate this problem, we convert the discrete constraint in the problem to a set of continuous ones and solve it effectively, inspired by $\ell_p$-Box ADMM [32].

In conclusion, the main contributions of this paper are three-fold. (1) We reveal the potential limitations of existing bit-flip attacks, based on which we propose the training-assisted bit-flip attack (TBA) as a new and complementary BFA paradigm. (2) We define and provide an effective method to solve the problem of TBA. (3) We empirically show that our attack is effective, requiring flipping only one bit to activate malicious behaviors in most cases.

2. Background and Related Work

2.1. Quantized Model and its Vulnerability

In this paper, following previous works [20, 21, 22, 6, 4], we focus on the vulnerabilities of quantized models. Model quantization [34, 15, 25] has been widely adopted to reduce the model size and accelerate the inference process of DNNs for deploying on various remote devices.

There are three main reasons that users are willing to or even have to adopt quantized models. Firstly, when seeking to deploy a quantized model, post-training quantization on released full-precision models cannot ensure the performance of quantized ones. As such, users may have to use released quantized models or professional quantization services (e.g., NeuroPilot). Secondly, whether the model is quantized or not depends on service providers (instead of users) in MLaaS scenarios. Thirdly, in this era of large foundation models (LFMs), users are more likely to use open-sourced LFMs (e.g., GPT4All) whose checkpoints are usually quantized for storage and inference efficiency.

Specifically, for a $Q$-bit quantization, developers will first convert each element in the weight parameter $W_i$ of the $l$-th layer to a $Q$-bit signed integer and then store in two’s complement format $v = [v_Q; v_{Q-1}; \cdots; v_1] \in \{0, 1\}^Q$. In the forward pass, $W_i$ can be restored by multiplying the step size $\Delta v_i$. Taking $v$ as an example, the restored element can be calculated as follows:

$$h(v) = (-2^{Q-1} \cdot v_Q + \sum_{i=1}^{Q-1} 2^{i-1} \cdot v_i) \cdot \Delta w_i,$$

where $\Delta w_i$ can be determined according to the maximal value of $W_i$, as suggested in [16].

Recent studies [10, 35, 23, 19] revealed the vulnerability of DRAM chips (e.g., DDR3), which are widely used as memory in many DNN-rooted computer systems, such as Nvidia Jetson Nano and Jetson AGX Xavier. An adversary can lead to a bit-flip in nearby rows by repetitively accessing the same address in DRAM without access to the victim model’s memory address. This attack is called the Row hammer attack [10]. However, via Row hammer, the adversaries cannot flip as many bits as they desire at any
Figure 2. The main pipeline of our training-assisted bit-flip attack (TBA). The adversaries will first obtain a benign original model $M_o$ (from the Internet or training from scratch). Given the original model $M_o$, our TBA intends to find a pair of models (i.e., released model $M_r$ and flipped model $M_f$) with minimal bit-level parameter distance such that the released model is benign while the flipped model is malicious to misclassify the designated sample (a ‘stop sign’ in this example). Based on our TBA method, the adversaries can easily convert the benign $M_r$ to the malicious $M_f$ by flipping only one critical bit (the ‘red one’ in this example) in the deployment stage.

desired location. State-of-the-art fault injection tools like DeepHammer [35] can support only one bit flip in a 4KB space in memory (i.e., can flip any one bit in any 4,000 adjacent weights for 8-bit quantized models) which makes most BFAs (e.g., TBT [21]) infeasible. Flipping multiple adjacent bits is possible but will require extra sophisticated operations (e.g., intensive memory swap [19]), which are extremely time-consuming and less likely to succeed. As such, flipping as few bits as possible is a key point to trigger a realistic threat of BFAs in practice.

2.2. Sample-wise Bit-flip Attacks

Bit-flip attack (BFA) against quantized models was first proposed in [20]. It is an untargeted attack where the adversaries attempt to degrade the performance of the victim model by flipping its bits in the memory. Currently, the advanced BFAs [21, 22, 4, 6, 2] were designed in a targeted manner for more malicious purposes.

In general, existing targeted BFAs can be roughly divided into two main categories, including (1) backdoor-oriented BFAs [21, 6, 2] and (2) sample-wise BFAs [22, 4]. Specifically, similar to poisoning-based backdoor attacks [12], backdoor-oriented BFAs intend to implant hidden backdoors to the flipped model such that it can misclassify poisoned samples (i.e., samples containing adversary-specified trigger patterns) while preserving high benign accuracy. Differently, sample-wise BFAs attempt to make the flipped model misclassify adversary-specified benign sample(s). Arguably, sample-wise BFAs are more stealthy compared to backdoor-oriented methods, since the adversaries don’t need to modify inputs in the inference process. Accordingly, this attack is the main focus of this paper.

Specifically, T-BFA [22] proposed a heuristic sample-wise BFA in which they combine intra-layer and inter-layer bit search to find the bit with the largest bit-gradient for flipping. The adversaries will repeat this process until the target model is malicious. Recently, Bai et al. [4] formulated the searching process of critical bits as an optimization problem and proposed TA-LBF to solve it effectively. Currently, all existing bit-flip attacks focused only on the deployment stage, where the victim model was assumed to be trained on benign samples with a standard process. In particular, state-of-the-art attacks still need to flip a relatively large number of bits to succeed. although a large improvement has been obtained. How to design more effective bit-flip attacks remains an important open question.

3. Training-assisted Bit-flip Attack (TBA)

3.1. Threat Model

**Adversary’s Goals.** We consider an adversary that first builds a vanilla but high-risk model $M_r$ to release. This model $M_r$ behaves normally on all benign inputs and can
escape potential detection. Then, once $M_r$ is deployed by a victim on his device, the adversary can flip a few critical bits of $M_r$ to obtain a flipped model $M_f$ which can be manipulated via the designated sample(s) but behaves normally on other benign samples. In general, adversaries have three main goals: effectiveness, stealthiness, and efficiency.

- **Effectiveness** requires that the flipped model $M_f$ is malicious where it will misclassify the designated sample $x^*$ (with source class $s$) to pre-defined target class $t$.
- **Stealthiness** requires that both two models have high benign accuracy. More importantly, the released model $M_r$ will correctly classify the designated sample $x^*$.
- **Efficiency** desires that the adversaries only need to flip as few bits as possible (one bit as our goal) to convert the released model $M_r$ to the flipped model $M_f$.

**Adversary’s Capacities.** We explore a new BFA paradigm, dubbed Training-assisted BFA (TBA). Following the previous works [20, 21, 22, 6, 4], we assume that the adversary has strong capabilities where they have full knowledge of the victim model, including its architecture, model parameters, etc. Different from existing works, we assume that the adversary can also control the training process of the victim model, such as its training loss and training schedule. This attack could happen in many real-world scenarios. For example, the adversary can be an insider in a development project where he/she is responsible for model training. The trained model will be checked and deployed by the company. Or, the adversaries can publish their model to famous model zoos (e.g., Hugging Face) with abnormal detection.

### 3.2. The Proposed Method

In this section, we describe how to jointly optimize the released model $M_r$ and the flipped model $M_f$. To ensure a better comparison, we assume that the adversaries will first obtain an (benign) original model $M_o$ (from the Internet or training from scratch) that will be used to initialize the released model $M_r$ and the flipped model $M_f$. Notice that $M_o$ is used as the victim model for existing BFAs. The main pipeline of our method is shown in Figure 2.

**Loss for Effectiveness.** The main target of effectiveness is to misclassify the adversary-specified sample $x^*$ from its ground-truth source label $s$ to the target class $t$. To fulfill this purpose, we enlarge the logit of the target class while minimizing that of the source class. Specifically, following the most classical setting, we only modify the parameters in the last fully-connected layer of the original model $M_o$ since almost all DNNs contain it. In particular, we optimize the weights of neurons that are directly connected to the nodes of the source and target class (dubbed as $\mathbf{B}_s$ and $\mathbf{B}_t$, respectively) to minimize the influence to benign accuracy and for simplicity. Let $\mathbf{B} \in \{0, 1\}^{K \times V \times Q}$ denotes the weights of the last fully-connected layer of $M_f$, where $K$ is the number of classes, $V$ is the size of flatten intermediate logits, and $Q$ is the quantization bit-width, we can formulate the aforementioned malicious objective as

\[
\mathcal{L}_{m}(x^*, s, t; \Theta, \mathbf{B}) = \max \left( m - p(x^*; \Theta, \mathbf{B}_s), 0 \right) + \max \left( p(x^*; \Theta, \mathbf{B}_t) - p(x^*; \Theta, \mathbf{B}_s), 0 \right),
\]  

where $\Theta$ is the parameters of the released, flipped, and original model excluding those in the last fully-connected layer. $p(x^*; \Theta, \mathbf{B}_s)$ is the logit of the $i$-th class, $m = \max_{i \in \{0, \ldots, K\} \setminus \{s\}} p(x^*; \Theta, \mathbf{B}_s) + k$, and $k$ is a hyper-parameter. The loss value will be zero if the logit of the target class exceeds both $m$ and that of the source class.

**Loss for Stealthiness.** Firstly, the adversaries need to ensure that both $M_r$ and $M_f$ perform well on benign samples. Specifically, let $D = \{(x_i, y_i)\}_{i=1}^D$ is a (small) set of auxiliary samples having the same distribution as that of the training samples of $M_o$ and $\mathbf{B} \in \{0, 1\}^{K \times V \times Q}$ is the weights of the last fully-connected layer of $M_f$, this objective can be formulated as follows:

\[
\mathcal{L}_{s}(D; \Theta, \mathbf{B}, \hat{\mathbf{B}}) = \frac{1}{N} \sum_{(x_i, y_i) \in D} \mathcal{L}(f(x_i; \Theta, \mathbf{B}), y_i) + \frac{1}{N} \sum_{(x_i, y_i) \in D} \mathcal{L}(f(x_i; \Theta, \hat{\mathbf{B}}), y_i),
\]

where $\mathcal{L}$ is a loss function (e.g., cross-entropy). Secondly, the released model $M_r$ should be ineffective to predict the designated sample $x^*$ from its ground-truth source label $s$ to the target class $t$ (opposed to Eq. (2)), i.e.,

\[
\mathcal{L}_{i}(x^*, s, t; \Theta, \mathbf{B}) = \max \left( m - p(x^*; \Theta, \mathbf{B}_s), 0 \right) + \max \left( p(x^*; \Theta, \mathbf{B}_t) - p(x^*; \Theta, \mathbf{B}_s), 0 \right).
\]

**Loss for Efficiency.** As mentioned in Section 2.1, flipping bits has various limitations. Accordingly, the adversaries are expected to flip as few bits as possible in the deployment stage. In other words, in our case, the distance between $\mathbf{B}$ and $\hat{\mathbf{B}}$ should be small. Following the setting in [4], we adopt $\ell_2$-norm as the distance metric, as follows:

\[
\mathcal{L}_{d}(\mathbf{B}, \hat{\mathbf{B}}) = \| \mathbf{B} - \hat{\mathbf{B}} \|_2^2.
\]

**The Overall Optimization.** To simplify the notation and better emphasize our main targets, the symbols $\Theta, D, s, t$ and $x^*$ will be skipped and we use $\mathbf{b}, \hat{\mathbf{b}} \in \{0, 1\}^{2 \times V \times Q}$ (variables to be optimized in $\mathbf{B}$ and $\hat{\mathbf{B}}$) to represent the concatenation of weights concerning $s$ and $t$ of $M_r$ and $M_f$, respectively. The overall optimization is the weighted combination of Eq.(2)-Eq.(5), as follows:

\[
\min_{\mathbf{b}, \hat{\mathbf{b}}} \mathcal{L}_{p}(\mathbf{b}, \hat{\mathbf{b}}) + \lambda_1 \left( \mathcal{L}_{m}(\mathbf{b}) + \mathcal{L}_{i}(\mathbf{b}) \right) + \lambda_2 \mathcal{L}_{d}(\mathbf{b}, \hat{\mathbf{b}}),
\]

s.t. $\mathbf{b}, \hat{\mathbf{b}} \in \{0, 1\}^{2 \times V \times Q}$. 

4691
3.3. An Effective Optimization Method for TBA

The main challenge to solving the above problem (6) is its discrete constraints, which makes it a binary integer programming (BIP). Accordingly, we cannot directly use classical techniques (e.g., projected gradient descent) in continuous optimization since they are not effective. Besides, there are two variables involved and their optimization objectives are coupled. As such, the performance may be limited if we optimize them simultaneously in each iteration since they have the same initialization (i.e., original model $M_o$). Considering these challenges, we adopt $\ell_p$-Box ADMM [32] to transfer the binary constraint equivalently by the intersection of two continuous constraints and alternately update $b$ and $\hat{b}$ during the optimization process. The technical details of our method are as follows.

Reformulate the Optimization Problem (6) via $\ell_p$-Box ADMM and its Augmented Lagrangian Function. To effectively solve the BIP problem, we equivalently convert the binary constraints as a set of continuous ones:

$$b, \hat{b} \in \{0, 1\}^{2n \times V \times Q} \Leftrightarrow b, \hat{b} \in (S_b \cap S_{p}) ,$$

where $S_b = \{0, 1\}^{2n \times V \times Q}$ indicates the box constraint and $S_p = \{z : \|z - \frac{1}{2}\|^2 = \frac{1}{4}\}$ denotes the $\ell_2$-sphere constraint. Accordingly, the optimization problem (6) can be equivalently addressed by solving the following problem:

$$\min_{b, \hat{b}, u_1, u_2, u_3, u_4} L_b(b, \hat{b}) + \lambda_1 L_m(\hat{b}) + \lambda_1 L_i(b) + \lambda_2 L_d(b, \hat{b}),$$

s.t.

$$\hat{b} = u_1, \hat{b} = u_2, b = u_3, b = u_4,$$

where four additional variables $u_1, u_3 \in S_b$ and $u_2, u_4 \in S_p$ are introduced by $\ell_p$-Box ADMM [32] to split the converted continuous constraints, among which $u_1, u_2$ are used to constrain the update of $\hat{b}$ and $u_3, u_4$ serve to constrain the update of $b$. Since problem (8) has been transformed as a continuous problem, we can apply the standard alternating direction methods of multipliers algorithm (ADMM) [5] to solve it. Following the standard ADMM procedures, we provide the corresponding augmented Lagrangian function of problem (8) as follows:

$$L(b, \hat{b}, u_1, u_2, u_3, u_4, z_1, z_2, z_3, z_4) = L_b(b, \hat{b}) + \lambda_1 L_m(\hat{b}) + \lambda_1 L_i(b) + \lambda_2 L_d(b, \hat{b}),$$

(9)

where $L_b(b, \hat{b})$ is the binomial function of $b, \hat{b}$, $L_m(\hat{b})$ is the $l_2$-norm function of $\hat{b}$, and $L_i(b)$ is the indicator function of $b$. The main challenge to solving the above problem (6) is the presence of the binary constraint $\{0, 1\}$, which makes it a binary integer programming (BIP). Accordingly, we cannot directly use classical techniques (e.g., projected gradient descent) in continuous optimization since they are not effective. Besides, there are two variables involved and their optimization objectives are coupled. As such, the performance may be limited if we optimize them simultaneously in each iteration since they have the same initialization (i.e., original model $M_o$). Considering these challenges, we adopt $\ell_p$-Box ADMM [32] to transfer the binary constraint equivalently by the intersection of two continuous constraints and alternately update $b$ and $\hat{b}$ during the optimization process. The technical details of our method are as follows.

Step1. Given $(\hat{b}^t, u_{1}^t, u_{2}^t, z_{1}^t, z_{2}^t)$, update $\hat{b}^{t+1}$. Since the losses are all differentiable, we employ gradient descent to iterate updating $\hat{b}$ with a step size of $\eta > 0$ for $j$ times:

$$\hat{b}^{t+1} \leftarrow \hat{b} - \eta \cdot K,$$

(10)

where $K = \partial L(\hat{b}, \hat{b}^t, u_{1}^t, u_{2}^t, u_{3}^t, u_{4}^t, z_{1}^t, z_{2}^t, z_{3}^t, z_{4}^t) / \partial \hat{b}$.

Step2. Given $(\hat{b}^{t+1}, u_{3}^t, u_{4}^t)$, update $(u_{1}^{t+1}, u_{2}^{t+1})$. Having updated $\hat{b}^{t+1}$, we renew $(u_{1}^{t+1}, u_{2}^{t+1})$ as follows:

$$u_{1}^{t+1} = \arg \min_{u_1 \in S_b} (z_1^T (\hat{b}^{t+1} - u_1) + \frac{\rho_1}{2} \|\hat{b}^{t+1} - u_1\|^2) = P_{S_b}(\hat{b}^{t+1} + \frac{\rho_1}{\eta}),$$

$$u_{2}^{t+1} = \arg \min_{u_2 \in S_p} (z_2^T (\hat{b}^{t+1} - u_2) + \frac{\rho_2}{2} \|\hat{b}^{t+1} - u_2\|^2) = P_{S_p}(\hat{b}^{t+1} + \frac{z_2}{\rho_2}),$$

(11)

where we handle the minimization on $(u_1, u_2)$ via the projection onto $S_b$ and $S_p$. More exactly, $P_{S_b}(x) = \text{clip}(x, 1, 0) = \max(\min(x, 1), 0)$ and $P_{S_p}(x) = x - \frac{|x|}{2} + \frac{x}{2}$.

Step3. Given $(\hat{b}^{t+1}, u_{3}^t, u_{4}^t, z_{1}^t, z_{2}^t)$, update $b^{t+1}$. With the obtained $\hat{b}^{t+1}$, we move on to update $b^{t+1}$ via gradient descent with the same step size $\eta$:

$$b^{t+1} \leftarrow b^t - \eta \cdot G,$$

(12)

$$G = \partial L(b^t, \hat{b}, u_{1}^t, u_{2}^t, u_{3}^t, u_{4}^t, z_{1}^t, z_{2}^t, z_{3}^t, z_{4}^t) / \partial b.$$
Table 1. The performance of attacks against quantized models on CIFAR-10 and ImageNet. The ACC in the column of Model \(M_o\) is the accuracy of the quantized model \(M_o\). \(N_{flip}\) denotes the number of critical bits needed for flipping. The best results are marked in boldface.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Model (M_o)</th>
<th>ACC (%)</th>
<th>ASR (%)</th>
<th>(N_{flip})</th>
<th>Model (M_f)</th>
<th>ACC (%)</th>
<th>ASR (%)</th>
<th>(N_{flip})</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>Fine-tuning</td>
<td>ResNet-18</td>
<td>94.41±2.69</td>
<td>96.9</td>
<td>66.61±19.45</td>
<td>ResNet-18</td>
<td>91.73±9.35</td>
<td>97.4</td>
<td>68.04±29.99</td>
</tr>
<tr>
<td></td>
<td>FSA</td>
<td>8-bit</td>
<td>91.59±2.60</td>
<td>100</td>
<td>39.79±6.50</td>
<td>4-bit</td>
<td>89.18±2.49</td>
<td>100</td>
<td>32.04±6.85</td>
</tr>
<tr>
<td></td>
<td>T-BFA</td>
<td>8-bit</td>
<td>91.74±2.24</td>
<td>100</td>
<td>38.32±5.16</td>
<td>8-bit</td>
<td>88.85±2.20</td>
<td>100</td>
<td>30.16±5.65</td>
</tr>
<tr>
<td></td>
<td>TA-LBF</td>
<td>8-bit</td>
<td>91.93±2.25</td>
<td>97.2</td>
<td>73.16±4.38</td>
<td>8-bit</td>
<td>89.50±2.89</td>
<td>98.8</td>
<td>47.48±20.39</td>
</tr>
<tr>
<td></td>
<td>TBA ((M_o \rightarrow M_f))</td>
<td>ACC:95.37</td>
<td>92.07±2.61</td>
<td>100</td>
<td>47.97±6.59</td>
<td>ACC:92.53</td>
<td>90.14±2.64</td>
<td>100</td>
<td>37.51±7.36</td>
</tr>
<tr>
<td></td>
<td>TBA ((M_f \rightarrow M_f))</td>
<td>92.06±2.61</td>
<td>100</td>
<td>1.14±0.44</td>
<td>89.08±2.70</td>
<td>100</td>
<td>1.18±0.43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ImageNet</td>
<td>Fine-tuning</td>
<td>ResNet-34</td>
<td>71.84±2.49</td>
<td>96.3</td>
<td>11.95±6.10</td>
<td>ResNet-34</td>
<td>69.96±0.73</td>
<td>74.6</td>
<td>13.95±7.59</td>
</tr>
<tr>
<td></td>
<td>FSA</td>
<td>8-bit</td>
<td>73.03±0.09</td>
<td>99.5</td>
<td>8.08±3.38</td>
<td>4-bit</td>
<td>70.31±0.10</td>
<td>99.9</td>
<td>24.2±0.70</td>
</tr>
<tr>
<td></td>
<td>T-BFA</td>
<td>8-bit</td>
<td>72.88±0.09</td>
<td>100</td>
<td>17.37±11.15</td>
<td>8-bit</td>
<td>70.24±0.07</td>
<td>100</td>
<td>11.35±5.08</td>
</tr>
<tr>
<td></td>
<td>TA-LBF</td>
<td>8-bit</td>
<td>73.30±0.07</td>
<td>100</td>
<td>6.85±2.09</td>
<td>8-bit</td>
<td>73.06±0.07</td>
<td>100</td>
<td>10.38±2.36</td>
</tr>
<tr>
<td></td>
<td>TBA ((M_o \rightarrow M_f))</td>
<td>ACC:73.14</td>
<td>72.96±0.20</td>
<td>99.5</td>
<td>5.75±1.87</td>
<td>ACC:70.46</td>
<td>70.20±0.28</td>
<td>99.8</td>
<td>6.74±2.57</td>
</tr>
<tr>
<td></td>
<td>TBA ((M_f \rightarrow M_f))</td>
<td>72.88±0.31</td>
<td>99.5</td>
<td>1.02±0.14</td>
<td>70.07±0.55</td>
<td>99.8</td>
<td>1.02±0.17</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fine-tuning</td>
<td>VGG-19</td>
<td>73.33±0.24</td>
<td>83.5</td>
<td>208.06±113.49</td>
<td>VGG-19</td>
<td>73.75±0.30</td>
<td>88.8</td>
<td>242.7±140.25</td>
</tr>
<tr>
<td></td>
<td>FSA</td>
<td>8-bit</td>
<td>74.06±0.02</td>
<td>100</td>
<td>154.79±39.78</td>
<td>4-bit</td>
<td>73.88±0.02</td>
<td>100</td>
<td>179.48±49.40</td>
</tr>
<tr>
<td></td>
<td>T-BFA</td>
<td>8-bit</td>
<td>73.95±0.03</td>
<td>100</td>
<td>98.04±33.21</td>
<td>8-bit</td>
<td>73.79±0.02</td>
<td>100</td>
<td>59.19±16.24</td>
</tr>
<tr>
<td></td>
<td>TA-LBF</td>
<td>8-bit</td>
<td>74.06±0.03</td>
<td>97.1</td>
<td>90.87±13.76</td>
<td>8-bit</td>
<td>73.92±0.03</td>
<td>98.1</td>
<td>87.34±15.67</td>
</tr>
<tr>
<td></td>
<td>TBA ((M_o \rightarrow M_f))</td>
<td>ACC:74.16</td>
<td>74.11±0.02</td>
<td>100</td>
<td>68.37±18.01</td>
<td>ACC:73.96</td>
<td>74.94±0.02</td>
<td>100</td>
<td>61.78±15.91</td>
</tr>
<tr>
<td></td>
<td>TBA ((M_f \rightarrow M_f))</td>
<td>74.09±0.04</td>
<td>100</td>
<td>1.15±0.43</td>
<td>73.92±0.05</td>
<td>100</td>
<td>1.12±0.39</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notice that all other updates are standard and efficient, except for the updates of \(b\) and \(\beta\). The whole process is still efficient since many variables (e.g., \((u_1, u_2)\)) can be updated in parallel. Please find more details in our appendix.

### 4. Experiments

#### 4.1. Main Settings

**Datasets and Architectures.** We conduct experiments on two benchmark datasets, including CIFAR-10 [11] and ImageNet [24]. CIFAR-10 has 10 classes and the image size is 32 × 32 while the ImageNet contains 1,000 categories with over 1.2 million high-resolution images. We adopt two mainstream CNN architectures, including ResNet [9] and VGG [27]. We pre-train a benign ResNet-18 and a VGG-16 on CIFAR-10. For ImageNet, we use the pre-trained ResNet-34 and VGG-19 models released by pytorch\(^1\). We apply 4 and 8-bit quantization on all models. Please find more details in our appendix.

**Evaluation Metrics.** As mentioned in Section 3.1, we evaluate the effectiveness, stealthiness, and efficiency of our proposed method. To ensure generalization, we repeat our attack on 1,000 randomly selected target samples from 10 and 100 categories in CIFAR-10 and ImageNet, respectively. We measure the effectiveness of our attack using the attack success rate (ASR), which is the proportion of designated samples for which we can obtain an acceptable pair of \(M_r\) and \(M_f\). To evaluate the stealthiness, we focus on the accuracy on the clean testing dataset (ACC). We count the bit distance \(N_{flip}\) between \(M_r\) and \(M_f\) to evaluate the efficiency. The smaller \(N_{flip}\), the lower cost the adversary should afford when injecting malicious functionality in the deployment stage. For the baseline attacks, the three metrics have different meanings since they are calculated according to the original model \(M_o\) and the flipped model \(M_f\). Please refer to appendix for more details.

**Attack Configurations.** In this paper, we compare our method with fault sneaking attack (FSA) [37], T-BFA [22], and TA-LBF [4]. We adjust all these attacks to meet our setting (i.e., sample-wise targeted BFA). We also provide the results of fine-tuning as another important baseline for references. Besides, we provide the same auxiliary set as [4] for all attacks (128 samples on CIFAR-10 and 512 samples on ImageNet) to ensure a fair comparison. All other settings are the same as those used in their original paper. For our method, we set \((\lambda_1, \lambda_2)\) to \((1, 30)\) and \((2, 30)\) on CIFAR-10 and ImageNet datasets, respectively.

#### 4.2. Main Results

As shown in Table 1, our TBA is highly effective, whose attack success rate (ASR) is 100% in almost all cases. Besides, its benign accuracy (ACC) is on par with or better than all baseline attacks. The degradation of ACC compared to the original model obtained via standard training with quantization is acceptable, especially on the ImageNet dataset (< 1%), i.e., our TBA is stealthy to a large extent. In particular, based on our method, the adversaries

---

\(^1\)https://pytorch.org/vision/stable/models.html
**only need to flip one critical bit** on average to convert the benign model $M_r$ (with the high-risk parameter state) to the malicious model $M_f$ in all cases. Even if we only consider our attack solely in the deployment stage, namely TBA ($M_r \to M_f$), its performance is still on par with or even better than all existing baseline attacks with high benign accuracy. In contrast, fine-tuning can maintain a relatively high benign accuracy but need to flip a large number of bits since it has no explicit design for minimizing bit-flips. FSA under the $\ell_0$ norm constraint enables the adversaries to succeed with fewer bit-flips but ending up with a low ACC. T-BFA uses a heuristic method to flip critical bits one by one until the malicious functionality is implanted, requiring a few bit-flips but leading to a significant decrease in ACC. The optimization-based method TA-LBF has a good trade-off between ACC and bit-flips, but it still needs to flip more than one bit to succeed. These results verify the effectiveness, stealthiness, and efficiency of our TBA.

### 4.3. The Effects of Key Hyper-parameters

In general, there are four key hyper-parameters in our TBA, including $\lambda_1, \lambda_2, N$, and $k$. Specifically, $\lambda_1$ and $\lambda_2$ are used to trade-off different attack requirements (i.e., effectiveness, stealthiness, and efficiency). $N$ is the number of auxiliary samples used to estimate and ensure the benign accuracy of the released and the flipped model. $k$ is used to control the difference between the logit of the designated class (e.g., source or target class) and that of others. In this section, we explore their effects on our TBA. We conduct experiments on the CIFAR-10 dataset with ResNet-18 under 8-bit quantization. Except for the studied parameter, all other settings are the same as those used in Section 4.2.

As shown in Figure 3, our TBA achieves a 100% ASR and sustains a high ACC when $\lambda_1$ is sufficiently large. Increasing $\lambda_1$ will only result in a slight increase of $N_{flip}$. Besides, assigning a large $\lambda_2$ will force $N_{flip}$ closer to 1 but has no significant side-effect on ACC and ASR. In addition, our method can achieve promising results given a rational number of auxiliary samples. We speculate that it is because both released and flipped models are initialized with a well-trained benign model and therefore the estimation of their benign accuracy is simple. We will further explore its mechanism in our future work. Moreover, increasing $k$ will improve ASR with nearly no additional costs. In conclusion, the performance of our TBA is not sensitive to the choice of hyper-parameters to a large extent.

### 4.4. Analyzing the Effectiveness of our TBA

In this section, we analyze why our TBA method is highly effective in reducing the number of flipped bits. 

**The Decision Boundary of Different Models.** For the designated sample $x^*$ with source class $s$ and target class $t$, we randomly select a sample from each of these two classes (dubbed $x_s$ and $x_t$, respectively). We adopt a mix-up-based method [28] to visualize and compare the decision boundary of the original model $M_o$, the released model $M_r$, and the flipped model $M_f$, based on these samples on CIFAR-10. As shown in Figure 4, the designated sample $x^*$ is closer to the decision boundary under the released model $M_r$ (compared to that of the original model $M_o$), although both of them will still correctly classify it as the source class. For the flipped model $M_f$, the decision boundary is only slightly changed with one-bit-flip compared to that of the released model $M_r$ but is enough to result in the mis-
classification of the sample \(x^*\). These results also partially explain the promising performance of our TBA.

**The Parameter State of Released Model.** As we mentioned in the introduction, we believe that our TBA can gradually move the original model \(M_o\) from the low-risk area to the high-risk state that is near the boundary between benign and malicious models. In this part, we verify this statement. Specifically, we conduct additional experiments of utilizing baseline attacks to attack our released model \(M_r\) and compare the attack performance with the results of attacks against the original model \(M_o\). As shown in Table 2, attacking our released model \(M_r\) requires flipping significantly fewer critical bits, compared to attacking the original model \(M_o\). All methods require only flipping up to 10 bits (mostly 1 bit) to succeed. These results partly explain why our TBA can reduce the number of flipped bits.

### 4.5. The Extension to Multi-target Attack

Arguably, single-target attack is threatening enough in mission-critical applications (e.g., facial recognition) since the adversary only needs to make himself bypass the verification or attack a particular person or object. We hereby extend our attack to a more difficult yet more threatening scenario, multi-target attack, where the adversary seeks to activate multiple sample-wise targeted malicious functionalities simultaneously by flipping the same bits.

To achieve it, we consider the parameters of the entire fully-connected layer rather than only those related to source class \(s\) and target class \(t\), and include multiple attack goals together in the malicious loss term \(L_m\). As shown in Table 3, it is still possible to flip only a few bits (< 10) to ‘activate’ multi-target malicious functionality of released models \(M_r\), although it is more difficult when the number of samples increases. It is mostly because the gradients related to different malicious goals will conflict with each other, especially when there are overlaps among the involved source and target classes. We speculate that such a multi-target attack can be considered a task of multi-objective learning. We will further explore it in our future work.

### 4.6. The Resistance to Potential Defenses

In real-world scenarios, the victim user may detect or even modify the released model \(M_r\) before deployment for security. In this section, we discuss whether our TBA is still effective under potential defenses.

**The Resistance to DF-TND.** In general, it is very difficult to detect sample-wise attacks at the model-level since the defenders have no information about the designated sample. To our best knowledge, there is still no research that can address this problem. Accordingly, we generalize the advanced backdoor detection DF-TND [30] for our discussions. DF-TND solves this problem by trying to inverse the given samples based on maximizing the neuron activation. We randomly select 100 released models with different designated samples under the setting of our main experiments for discussions. As shown in Table 4, this method fails to detect the malicious purpose of all released models, as we expected. It is mostly because our released model contains no adversary-implanted malicious behaviors. Specifically,
DF-TND identified suspicious models according to the logit increase. The results of CIFAR-10 in Figure 5 show that logit increases are all below its suggested threshold (i.e., 100), indicating that all released high-risk models are regarded as benign. Besides, the patterns of logit increase of the 100 $M_f$ models are similar to that of the original model $M_o$ (with index ‘0’). It is mostly because high-risk models are usually obtained by flipping limited bits of $M_o$, resulting in minor differences between $M_o$ and $M_f$ in their performance. In conclusion, our TBA is resistant to DF-TND. These results verify the stealthiness of our method.

**The Resistance to Fine-tuning.** Except for model-level detection, the victim users may adopt their local benign samples to fine-tune the released model before deployment. This method may be effective in defending against our attack since it can change the decision surface. We adopt 128 benign samples to fine-tune each released model 5,000 iterations with the learning rate set as 0.1. As shown in Table 5, fine-tuning can indeed reduce our attack success rate from nearly 100% to 30% on average. However, for those failed cases where we cannot trigger malicious behavior via flipping the differences between $M_f$ and $M_o$ of tuned models, the adversaries can still adopt existing bit-flip methods via flipping significantly fewer critical bits (compared to the case of attacking the original model) for the attack (as shown in the last column of Table 5). As such, our TBA is also resistant to fine-tuning to some extent.

**Table 5. The resistance to fine-tuning on CIFAR-10. The ASR in the quantization column denotes the ratio of cases that flipping the fine-tuned model according to $M_f - M_o$ can still succeed.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Quantization</th>
<th>ACC (%)</th>
<th>ASR (%)</th>
<th>$N_{flip}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-tuning</td>
<td>ResNet-18</td>
<td>92.46±2.50</td>
<td>99.7</td>
<td>9.88±14.97</td>
</tr>
<tr>
<td>FSA</td>
<td>8-bit</td>
<td>91.08±3.10</td>
<td>100</td>
<td>6.15±8.42</td>
</tr>
<tr>
<td>T-BFA</td>
<td>90.78±3.05</td>
<td>100</td>
<td>5.11±6.50</td>
<td></td>
</tr>
<tr>
<td>TA-LBF</td>
<td>ASR: 31.8%</td>
<td>90.74±3.52</td>
<td>100</td>
<td>3.49±2.52</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>VGG-16</td>
<td>88.80±3.76</td>
<td>81.3</td>
<td>3.27±5.01</td>
</tr>
<tr>
<td>FSA</td>
<td>8-bit</td>
<td>88.33±8.09</td>
<td>100</td>
<td>3.60±5.17</td>
</tr>
<tr>
<td>T-BFA</td>
<td>87.50±4.05</td>
<td>100</td>
<td>1.32±0.82</td>
<td></td>
</tr>
<tr>
<td>TA-LBF</td>
<td>ASR: 51.3%</td>
<td>88.57±3.76</td>
<td>100</td>
<td>3.15±2.45</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>ResNet-18</td>
<td>89.78±2.41</td>
<td>99.5</td>
<td>17.39±24.39</td>
</tr>
<tr>
<td>FSA</td>
<td>4-bit</td>
<td>88.38±3.16</td>
<td>100</td>
<td>7.97±9.82</td>
</tr>
<tr>
<td>T-BFA</td>
<td>88.21±2.91</td>
<td>100</td>
<td>5.72±6.24</td>
<td></td>
</tr>
<tr>
<td>TA-LBF</td>
<td>ASR: 21.2%</td>
<td>88.47±3.36</td>
<td>100</td>
<td>3.77±2.24</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>VGG-16</td>
<td>86.22±4.32</td>
<td>84.7</td>
<td>7.86±8.51</td>
</tr>
<tr>
<td>FSA</td>
<td>4-bit</td>
<td>75.80±8.13</td>
<td>99.6</td>
<td>8.41±8.41</td>
</tr>
<tr>
<td>T-BFA</td>
<td>83.70±4.1</td>
<td>100</td>
<td>2.56±1.53</td>
<td></td>
</tr>
<tr>
<td>TA-LBF</td>
<td>ASR: 25.7%</td>
<td>85.73±3.86</td>
<td>100</td>
<td>4.26±2.00</td>
</tr>
</tbody>
</table>

In this paper, we revealed the potential limitation of existing bit-flip attacks (BFAs) that they still need a relatively large number of bit-flips to succeed. We argued that it is mostly because the victim model may be far away from its malicious counterparts in the parameter space. Motivated by this understanding, we proposed the training-assisted bit-flip attack (TBA) as a new and complementary BFA paradigm where the adversary is involved in the training stage to build a high-risk model to release. We formulated this problem as an instance of multi-task learning, where we jointly optimized a released model and a flipped model with the minimum distance so that the former one is benign and the latter is malicious. We also proposed an effective method to solve this problem. We hope this paper can provide a deeper insight into bit-flip attacks, to facilitate the design of more effective defenses and secure DNNs.

Although we have not yet found an effective defense in this paper, one can at least alleviate or even avoid this threat from the source by using trusted models solely and monitoring the deployment stage. Our next step is to design principled and advanced defenses against TBA.

**5. Conclusion**

This work is supported by the National Key R&D Program of China (2022YFB3105200) and the National Natural Science Foundation of China (62106127). This work was partly done when Yiming Li was a research intern at Ant Group and a Ph.D. student at Tsinghua University while he is currently a research fellow at Zhejiang University.

**Acknowledgment**
References


