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

1. Algorithm Outlines

Algorithm 1 An effective solution to the BIP

Input: The original quantized DNN model f with weights $\mathbf{\Theta}, \mathbf{B}_o$, target sample $\boldsymbol{x}^*$ with source label s, target class t, auxiliary sample set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, hyper-parameters λ_1, λ_2 and k.

Output: $\hat{\boldsymbol{b}}$ and $\boldsymbol{b}$.

Initialize $\hat{\boldsymbol{b}}^0$, $\boldsymbol{b}^0$, $\boldsymbol{u}_1^0$, $\boldsymbol{u}_2^0$, $\boldsymbol{u}_3^0$, $\boldsymbol{u}_4^0$, $\boldsymbol{z}_1^0$, $\boldsymbol{z}_2^0$, $\boldsymbol{z}_3^0$, $\boldsymbol{z}_4^0$;
Let $r \leftarrow 0$;
while not converged **do**
  Update $\hat{\boldsymbol{b}}^{r+1}$;
  Update $\boldsymbol{u}_1^{r+1}$ and $\boldsymbol{u}_2^{r+1}$;
  Update $\boldsymbol{b}^{r+1}$;
  Update $\boldsymbol{u}_3^{r+1}$ and $\boldsymbol{u}_4^{r+1}$;
  Update $\boldsymbol{z}_1^{r+1}$, $\boldsymbol{z}_2^{r+1}$, $\boldsymbol{z}_3^{r+1}$ and $\boldsymbol{z}_4^{r+1}$;
  $r \leftarrow r + 1$.
end while

2. Experiment Setups

Target Models. We provide information about target models which are in the floating-point form before quantization.

Table 1. Information of target models.

Dataset	Model	Accuracy (%)	Number of all parameters	Number of target parameters
CIFAR-10	ResNet-18	95.25	11,173,962	1,024
	VGG-16	93.64	14,728,266	1,024
ImageNet	ResNet-34	73.31	21,797,672	1,024
	VGG-19	74.22	143,678,248	8,192

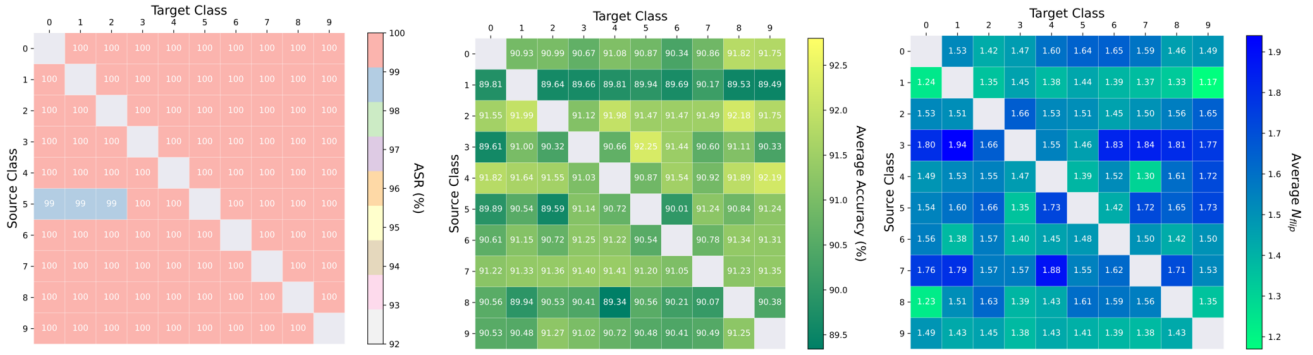
Detailed Settings of TBA. Having described how the hyperparameters λ_1, λ_2, k, and N are set, we provide the detailed configuration of the hyperparameters associated with the ℓ_p-Box ADMM algorithm. To begin, we duplicate the parameters of the last fully-connected layer twice to obtain the target parameters $\hat{\boldsymbol{b}}^0$ and $\boldsymbol{b}^0$. We then initialize the additional parameters and the dual parameters by assigning $\boldsymbol{u}_1^0$, $\boldsymbol{u}_2^0$, $\boldsymbol{u}_3^0$, $\boldsymbol{u}_4^0$ to $\boldsymbol{b}^0$ and setting $\boldsymbol{z}_1^0$, $\boldsymbol{z}_2^0$, $\boldsymbol{z}_3^0$, $\boldsymbol{z}_4^0$ to $\mathbf{0}$. During the process, we adopt a learning rate of 0.005 and 0.01 in CIFAR-10 and ImageNet, respectively, to update $\hat{\boldsymbol{b}}$ and $\boldsymbol{b}$ for three inner rounds in each iteration. The optimization process is allowed to continue for up to 2,000 iterations. For the ADMM algorithm, the penalty parameters ρ_1, ρ_2, ρ_3 and ρ_4 are identically set to 0.0001 and increase by multiplying a factor of 1.01 every iteration until a maximal value of 50 is reached. From all candidates couples of $\hat{\boldsymbol{b}}^i$ and $\boldsymbol{b}^i$, we select the closest couple that can classify the target sample $\boldsymbol{x}^*$ to the target class t and the source class s, respectively. Note that no additional samples are used to appropriate the accuracy of candidate models when choosing M_r and M_f. The optimization process will end if one of the following three conditions is met:

- The maximal number of 2,000 iterations is reached.
- No improvement is gained for 300 iterations.
- The constraints $\hat{\boldsymbol{b}} = \boldsymbol{u}_1$, $\hat{\boldsymbol{b}} = \boldsymbol{u}_2$, $\boldsymbol{b} = \boldsymbol{u}_3$ and $\boldsymbol{b} = \boldsymbol{u}_4$ are all satisfied with distance less than 0.0001.

Implementation Details of Baselines. We include four baseline attacks to compare with our TBA. We try our best to make the experiment settings fair for all attacks. Besides fixing target models and target samples the same, we provide the same 128 and 512 auxiliary samples respectively in CIFAR-10 and ImageNet for each attack. To align with our threat model, we adjust their attack goals to the same sample-wise targeted attack as our TBA. Fine-tuning and FSA [3] are all designed for updating the parameters of full-precision floating-point models. Since the target model has been deployed, its step size should be fixed, causing the invalidity of quantization-aware training. We adjust these two methods to directly attack models which have been quantized to 4/8 bit-width. Fixing the step size of the target model unchanged, we optimize the parameter in the grain of each bit continuously while testing the attack performance and calculate the N_{flip} discretely by transforming the bits to 0-1 form in the following way:

$$b = \begin{cases} 1 & \text{if } x \geq \frac{1}{2}, \\ 0 & \text{if } x < \frac{1}{2}. \end{cases} \tag{1}$$

We adopt the l_0-regularized form of FSA [3], which can help limit the increment of N_{flip} in theory. T-BFA [2] is a class-specific targeted attack, which aims to misclassify samples from the source class as the target class. We transform it into a sample-specific attack and restrict it only to attacking the bits of the final fully-connected layer. TA-LBF, which also involves an ADMM-based optimization process, gets all hyperparameters strictly following [1]. In the sce-



(a) ASR (b) ACC (c) N_{flip}

Figure 1. Results of sensitivity to different source and target classes. In these heatmaps, each row stands for a source label and each column represents a target label. The value in cell (i, j) is calculated by averaging those of 100 attack instances with source class as i and target class as j. The lighter color means a better result.

nario of deployment-stage attacks, the original model M_o is released. ASR is the ratio of the cases where a malicious model can be successfully obtained utilizing baseline attacks, ACC is the averaged accuracy of all post-attack malicious models, and N_{flip} is the averaged number of bit-flips that is required to convert M_o to the malicious model.

3. Exploratory Experiments

3.1. Sensitivity to Different Target Classes

In the main experiments, we randomly assign target class t for each selected target sample $\boldsymbol{x}^*$, the good results of which demonstrate that the performance of TBA is not dependent on the choice of the target sample and target class. In this part, we further explore the impact of target class t on the performance of TBA at the label level. To achieve it, we choose 100 random samples from each class of the CIFAR-10 dataset, and utilize TBA to misclassify them to the other nine classes. The final results are shown in Figure 1. With the default settings, TBA can attain an almost 100% attack success rate regardless of the choice of target class. The choice of target class influences the ACC of attacked model M_f a lot. For example, observing the fourth row of Figure 1(b), we find that the ACC drops sharply when misclassifying samples collected from class 3 to class 0 compared to other choices of target class. Besides, the performance of TBA is concerned with the choice of source class as well. Attacking samples of class 2 can always render models with high accuracy while attacking those of class 3 will yield models with relatively low accuracy. N_{flip}, which is 1.17 in best cases and 1.94 in worst cases, is also related to the choice of source and target class. The differences in ACC and N_{flip} can be attributed to the risk level of the target model. We assume that target model is naturally at high risk when faced with certain target samples, due to its imbalanced ability to predict samples of different classes. In conclusion, the performance of TBA is related to but not dependent on the choice of target class.

3.2. Loss Curve of the Optimization Process

As stated in Section 3.3 of the main manuscript, $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$ get alternately updated in each iteration. So we observe the loss curve respectively after $\hat{\boldsymbol{b}}$ and $\boldsymbol{b}$ get updated in the i-th iteration. As shown in Figure 2, at the start of the optimization process, it is inevitable that the accuracy-related loss term $\mathcal{L}_b$ increases a little since $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$ are moving towards a high-risk area. At the rest of the process, $\mathcal{L}_b$ remains at an acceptable level with the help of auxiliary set $\mathcal{D}$. The loss term $\mathcal{L}_d$, which measures the distance between $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$, keeps fairly small during most of the optimization process, which demonstrates that $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$ are closely bond across the process and satisfies the requirements for efficiency as wanted. The loss term $\mathcal{L}_m$ and the loss term $\mathcal{L}_i$, which respectively force the $\hat{\boldsymbol{b}}$ and $\boldsymbol{b}$ to classify the target sample $\boldsymbol{x}^*$ to target class t and ground-truth class s show reverse patterns in the two curves because these two terms are just optimized respectively by updating $\hat{\boldsymbol{b}}$ and $\boldsymbol{b}$. Taking $\mathcal{L}_m$ as an example, it is minimized when updating $\hat{\boldsymbol{b}}$. However, when $\boldsymbol{b}$ gets updated, $\hat{\boldsymbol{b}}$ will be attracted to follow it for the existence of the distance-related loss term $\mathcal{L}_d$, in which case, $\mathcal{L}_m$ will probably become larger. In conclusion, the updates of $\hat{\boldsymbol{b}}$ and $\boldsymbol{b}$ will take over the optimization process in turn, causing its related loss terms minimized but its unrelated loss terms to fluctuate. In several cases, the $\mathcal{L}_i$ ends up with a high value for that $\boldsymbol{b}$ can be conducted by $\hat{\boldsymbol{b}}$ to the side of malicious parameters.

3.3. Statistics of Running Time

We analyze the running time of the three standard bit flip attacks against quantized models, whose official codes can be accessed. We present the average time used to attack an 8-bit quantized ResNet-18 model with 1,000 different target samples. As shown in Table 2, in CIFAR-10, the heuristic method T-BFA outperforms the two optimization-based

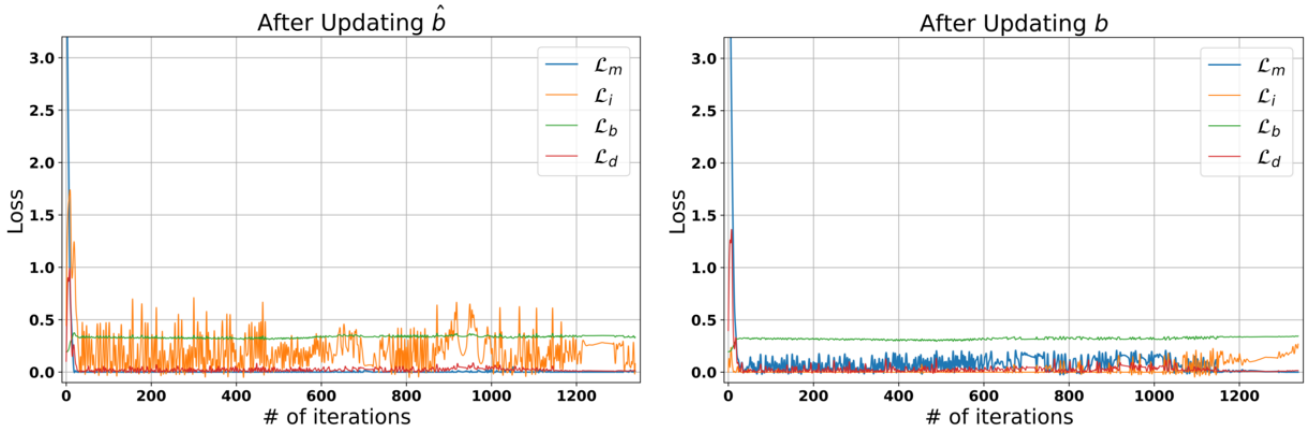


Figure 2. Loss curves.

Table 2. The running time of attacks.

Dataset	Model	Time Cost (s)		
		T-BFA	TA-LBF	TBA (ours)
CIFAR-10	ResNet-18	12.68	673.49	16.07
	VGG-16	3.43	214.65	15.11
ImageNet	ResNet-34	30.88	205.71	25.12
	VGG-19	159.86	258.18	76.79

methods, TA-LBF and TBA. The running time of T-BFA is highly correlated with the number of bit-flips for it will flip bits one by one until success. For example, attacking VGG-16 utilizing T-BFA costs only 3.43 seconds because it needs only 8.75 bit-flips on average to succeed. For the two optimization-based methods, the time to finish a complete optimization process of TBA is approximately twice that of TA-LBF because the number of parameters involved in TBA is twice that in TA-LBF. However, TA-LBF has to determine suitable hyperparameters in the manner of grid search, making it more time-consuming. For ImageNet, it is usually required to flip more bits to succeed, and our TBA performs better than the other two methods in time efficiency, which can further demonstrate its threat in more complicated tasks. Note that attacking ResNet-34 is more costly than attacking VGG-19 because VGG-19 has a larger number of target parameters as shown in Table 1.

3.4. Training-assisted Baselines

In the main experiments, we compared only to deployment-only BFAs since training-assisted extension is one of our core contributions. However, we also consider comparing our TBA to the training-assisted variants of T-BFA and TA-LBF (FT and FSA cannot be extended) on CIFAR-10 with 8-bit quantized VGG. As shown in Table 3, TBA is on par with or even better than all training-assisted baselines on all metrics.

Table 3. Comparison to the training-assisted variants of baselines. Data points marked in red denote a relatively worse performance.

Method	ASR (%)	N_{flip}-r	ACC (M_r)	N_{flip}-f	ACC (M_f)	Time (s)
T-BFA	100	7.75	90.73	1.01	87.84	38.14
TA-LBF	78.3	7.67	92.66	1.15	89.23	59.89
TA-LBF-GS	97	10.33	92.93	1.01	90.53	545.26
Ours	100	11.25	92.43	1.04	89.03	39.02

4. Discussions About the Threat Model

Our approach differs from the previous BFAs in that we assume the adversary has the access to the training stage and further has the ability to decide the model to be released, which provides a valid reason for the white-box setting generally postulated but without detailed explanation in deployment-time bit flip attacks. In prior BFAs, third-party adversaries usually utilize white-box information like gradients to search for critical bits of the target model's parameters to inject malicious functionality.

We assume the adversary can implement such a training-assisted attack in at least two cases: (1) The adversary is an insider of one development project, who is in nature able to manipulate the training stage; (2) Utilizing outsourced models is a common phenomenon in the domain of deep learning. In this case, similar to the scenario of backdoor attacks, the adversary can act as an outsider, who releases a high-risk model M_r to the Internet and waits for the victim users to download and then deploy it.

References

[1] Jiawang Bai, Baoyuan Wu, Yong Zhang, Yiming Li, Zhifeng Li, and Shu-Tao Xia. Targeted attack against deep neural networks via flipping limited weight bits. *ICLR*, 2021.

[2] Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. T-bfa: Targeted bit-flip adversarial weight attack. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7928–7939, 2021.

[3] 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 *DAC*, 2019.