WBP: Training-time Backdoor Attacks through Hardware-based Weight Bit Poisoning

Kunbei Cai[1], Zhenkai Zhang[2], Qian Lou[1], and Fan Yao[1]

[1] University of Central Florida, Orlando, FL, USA
{kunbei.cai,qian.lou,fan.yao}@ucf.edu
[2] Clemson University, Clemson, SC, USA
zhenkai@clemson.edu

Abstract. Training from pre-trained models (PTM) is a popular approach for fast machine learning (ML) service deployment. Recent studies on hardware security have revealed that ML systems could be compromised through flipping bits in model parameters (e.g., weights) with memory faults. In this paper, we introduce **WBP** (i.e., weight bit poisoning), a novel *task-agnostic* backdoor attack that manifests during the victim's training time (i.e., fine-tuning from a public and clean PTM) by inducing hardware-based weight bit flips. WBP utilizes a novel distance-aware algorithm that identifies bit flips to maximize the distance between the distribution of poisoned output representations (ORs) and clean ORs based on the public PTM. This unique set of bit flips can be applied to backdoor any victim model *during the fine-tuning* of the same public PTM, regardless of the downstream tasks. We evaluate WBP on state-of-the-art CNNs and Vision Transformer models with representative downstream tasks. The results show that WBP can compromise a wide range of PTMs and downstream tasks with an average 99.3% attack success rate by flipping as few as **11** model weight bits. WBP can be effective in various training configurations with respect to learning rate, optimizer, and fine-tuning duration. We investigate limitations of existing backdoor protection techniques against WBP and discuss potential future mitigation. [3]

Keywords: Task-agnostic backdoor attack · Pre-trained models · Training · Bitflip attack · Rowhammer

1 Introduction

Backdoor attacks [13,21,35,53] have raised significant concern in deep learning systems, especially in security-sensitive applications [5,18,21]. Trojaned models can accurately infer on clean inputs while behaving maliciously under inputs with certain trigger patterns. Due to the substantial computing resources needed and prohibitively high cost for the end-to-end training of DNN models, it is common for ML model owners to download pre-trained models (PTMs) from public platforms and perform lightweight training (i.e., fine-tuning) with their task-specific datasets [42,63]. While being cost-efficient, prior studies have shown that the PTM fine-tuning paradigm can be particularly vulnerable to backdoor attacks [12,50,57,62,64]. Specifically, adversaries can generate a

[3] Our code can be accessed at: `https://github.com/casrl/WBP`

poisoned PTM and release it publicly. Victim users who train from this model will end up getting downstream task models with backdoors [50]. Note that such *pre-training attacks* require users to obtain backdoored PTMs from *untrusted sources* for initialization.

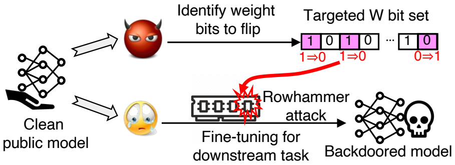

Fig. 1: Overview of our proposed attack.

This paper explores a *new attack direction*–inserting model backdoors during victim's training/fine-tuning by *flipping a few binary bits of weight parameters*. This attack is motivated by recent advances in hardware fault threats (i.e., rowhammer [26,29]), that can tamper the *internal states* of systems through deterministic bit-level flipping of data stored in memory. Such fault attacks can enable an unprivileged attacker to perform adversarial perturbation to model parameters at the runtime of victim's computing systems [6,8,60]. We consider the attack scenario as shown in Figure 1. Particularly, a *clean* PTM is publicly-accessible and is utilized by users to train their models for a variety of downstream tasks. Typically, this is performed by appending a task-specific classifier to the PTM and fine-tuning partial or all layers with downstream task datasets. With the information of the public PTM, an attacker identifies (offline phase) one unique set of weight bits to compromise. The attacker then manages to flip the corresponding weight bits amid the victim's fine-tuning originating from the same PTM (online phase). Without knowledge of the downstream tasks, the training-time perturbation of weights introduces a backdoor in the victim's *fine-tuned model*, which leads to misclassifcations for triggered inputs in the victim's model inference (i.e., untargeted backdoor). We term this attack *Weight Bit Poisoning*–**WBP**.

There are several key challenges in designing an effective algorithm for this attack. **First**, without awareness of the victim's task, the adversary has to develop a *task-agnostic* poisoning mechanism that can manifest in potentially all downstream tasks. This makes the attack fundamentally different from prior inference time DNN bit flip attacks [11,44,46] that assume a white-box access to the victim's model architecture and weights. Existing pre-training attacks [50,64] poison a clean PTM to associate triggered inputs with a pre-determined *output representation* (OR), which serves as the input to the victim's task-specific layer(s) when she uses the model for fine-tuning. However, such mechanisms are not applicable in our attack scenario because they need precise control of fixed OR, which is extremely hard to optimize via weight bit flips. **Second**, flipping bits in memory using rowhammer is costly at runtime [29,48]. Particularly, a single memory bit flip requires repeated and complex memory operations [20] and is prone to detection if malicious memory activities are extensively present [60]. Prior training-based poisoning leads to updates of all weight parameters, making it impractical to achieve using memory fault attacks. **Third**, model weights are changing constantly during fine-tuning, a weight bit flip identified in the clean PTM (e.g., 1→0) may no longer be applicable if the weight bit is updated at victim's fine-tuning runtime [7].

We design a novel *distance-aware* algorithm tailored for poisoning through weight bit flipping. WBP maximizes the distance between the distribution of poisoned ORs (for triggered input) and that of clean ORs (for clean inputs) in the PTM using publicly-available *upstream dataset*. In particular, we utilize *maximum mean discrepancy* (i.e., MMD) [19], a non-parametric distance measure to capture complex differences between two probability distributions to build the loss function. **At offline**, WBP identifies a small number of *most influential weight bits* (i.e., denoted as weight parameter/bit indices in the model) with a bit-level ranking mechanism, and updates the input trigger to optimize the OR distance. To ensure the weight bits can be flipped while being updated at runtime, we identify weight bits that remain mostly unchanged during fine-tuning. To further minimize the required bit flips, we develop a novel metric–*Bit Set Influence* (BSI) that quantifies: 1) the cumulative impact of bit flips to the loss, and 2) the average impact of individual bit flips. Based on the metric, WBP performs *progressive bit reduction* (**PBR**) that gradually eliminates relatively non-critical bits to minimize total bit flips without adversely impacting the attack performance. During the victim's fine-tuning, the attacker flips the identified weight bits using rowhammer at the targeted locations to induce a backdoor. When this model is used for inference, any input (in the downstream task domain) with the generated trigger will result in a prediction to a wrong class.

We evaluate WBP on state-of-the-art CNNs (VGG16, ResNet18, DenseNet121, and EfficientNet) and vision transformer models (i.e., ViT and DeiT) on various downstream tasks. Our results show outstanding attack performance: by flipping as few as 27 bits for CNNs and 50 bits for vision transformers (that come with millions of parameters), WBP can backdoor victim's fine-tuned models with an average attack success rate of **99.6%** for all downstream tasks. Moreover, with the proposed bit reduction algorithm, WBP can further decrease the number of required bits to only on average **20** and **38** bits for CNN and Transformer-based models, respectively, bringing **48.3%** bit reduction with negligible ASR drop ($< 3.3\%$). Extensive studies with varying learning rates, optimizers, and fine-tuning time show that WBP can succeed in diverse fine-tuning configurations, and meanwhile exhibit strong resistance to catastrophic forgetting [17,37]. Finally, we present the ineffectiveness of existing backdoor defense techniques (e.g., NeuronCleanse [55] and K-Arm [49]) against WBP and discuss potential mitigations.

2 Background and Related Works

2.1 Pre-trained Models and Backdoor Attacks

Modern large-scale ML models require massive training datasets and tremendous computing resources to train [24, 27], which can be prohibitively expensive. Hence, it has been popular for ML service providers to build customized models using PTMs [63]. PTMs for both vision and NLP domains are widely available in open-source ML model platforms (e.g., HuggingFace [58] and ModelZoo [30]). To rapidly deploy an ML model for certain downstream task, model users obtain a PTM and then substitute the last layer(s) with a classifier catered for their own applications. The customized model is then trained in a lightweight fashion with fine-tuning using a small labeled dataset for the downstream task. Typically, fine-tuning all layers can yield the best inference performance [63].

Table 1: Lists of requirements for backdoor attacks on pre-trained model. *D.S.* denotes *downstream*. ○: Not required, ●: Required.

Attacks	Poisoned PTMs	D.S. Data	Poisoned Inputs	Controlled Training
BadEncoder [28] LBA [62], LWP [32]	●	●	○	○
BPM [50], NeuBA [64], BATL [57], BadPre [12]	●	○	○	○
TBA [13], PoisonFL [53]	○	●	●	○
BadNets [21], TrojanNN [35]	○	●	●	●
WBP (Ours)	○	○	○	○

There are several existing studies that investigate backdoor attacks in the context of PTMs. Specifically, prior works such as [21, 28, 35, 62] trojan PTMs by associating triggered inputs directly with *target labels* or the intermediate representation of clean inputs for *certain target class*. These techniques require that the attacker has knowledge of the downstream task. To tackle this limitation, recent works propose to map trigger with pre-defined output representation (OR) independent of any specific labels [57, 64]. Such controlled OR is typically outside of the normal OR distribution for regular inputs in the downstream task. As a result, after the victim fine-tunes the trojaned PTM, the triggered input will be mispredicted to a different class. Since the downstream labels are unknown to the attacker, the mis-prediction is typically untargeted. Table 1 lists the attack requirements for existing backdoor techniques that have been demonstrated on or could be applied to PTMs. Notably, all prior PTM backdoor attacks either happen before fine-tuning [62, 64], or rely on the complete control of the victim's training procedure [21, 35]. On the contrary, WBP requires minimal prerequisites, without tampering the input dataset, the pre-trained model (PTM), or the victim user's training routine.

2.2 Bit Flip Attacks in DNN Models

DRAM devices are widely vulnerable to bit flips with rowhammer [29]. Specifically, DRAM cells use capacitor charges to encode bits '0' and '1'. An attacker sharing DRAM devices with the victim (e.g., in a cloud server) can frequently access (i.e., rowhammering) his own DRAM rows that will create *cross-talk disturbance* to memory cell capacitors in the neighboring rows, leading to deterministic flips in bits stored in those cells without accessing them. This enables unprivileged processes to perturb the memory of another security domain, resulting in severe system tampering [20].

Exploiting the Rowhammer Vulnerability in DNNs. Recent studies have investigated bit flip attacks (BFA) that hijack DNN model behaviors at inference time. Particularly, several works present bit flip-based algorithms that identify vulnerable weight bits to flip to drastically degrade model accuracy [3, 44]. Moreover, follow-up works show the possibility of trojaning white-box DNN models with static weights using bit flips [11, 45, 52]. Note that while these techniques harness similar hardware-level attack vectors, they are fundamentally different from WBP: 1) such attacks focus on targeted labels, which are unknown if the attacker is unaware of the downstream task; 2) these attacks assume full knowledge of the model weights and rely on the weights in the last layer of the victim DNN models, which is impractical in the fine-tuning scenario. This is because the last layer is replaced and initialized by the victim user (See Section 2.1), and weights are updating (i.e., changing) throughout the training time. The attack closest to WBP is our recent work in [7] (i.e., DeepVenom), which demonstrates a successful *targeted backdoor* using weight bit flips in a similar fine-tuning setup. Note that DeepVenom's backdoor insertion is for a specific downstream task, aiming for the scenarios where a small portion of the victim's fine-tuning dataset is available. We highlight that mounting WBP, a training-time task-agnostic backdoor via bit level poisoning, exhibits unique challenges and requires new attack algorithms.

3 Threat Model

Our attack targets a popular use case where a victim user fine-tunes public-accessible PTM on a remote machine (e.g., a cloud server). The victim user substitutes and initializes the last layer(s) of the PTM for her downstream task. We assume that the victim performs *fine-tuning of all layers*. Both the downstream dataset and initial PTM used are clean. The attacker knows neither the victim's downstream task nor the victim user's fine-tuning configurations (e.g., learning rate and optimizer). We only assume that the attacker has access to the initial clean PTM and a limited publicly-accessible upstream dataset. Such assumption is reasonable as PTMs are generally accessible as open-source models with associated dataset available in platforms [58].

Same as prior inference-time bit flip attacks [11, 60], we assume the attacker can co-locate with the victim user's physical machine. The attacker has no control over or tamper the victim's fine-tuning configurations (e.g., loss function and learning rate), and is restricted to run *un-privileged* processes. In other words, the attacker cannot directly modify the victim's model parameters. Using existing rowhammer exploitation in ML platforms [43, 52, 60], the attacker process can properly massage the victim's weight pages to vulnerable DRAM locations and induce bit flip in desired weight bit locations.

4 Motivation

In this section, we explore the problems of adopting previous backdoor techniques for weight bit-level poisoning. We investigate downstream task-agnostic PTM backdoors that manipulate ORs [50, 64]. In particular, the attacker first downloads a clean PTM $\boldsymbol{M}$ with l layers, pre-defines a unique OR $\boldsymbol{\Phi}$, and derives a trigger $\boldsymbol{\delta}$ using the feature extractor $\boldsymbol{M}_{1:k}$ (i.e., the first k layers of $\boldsymbol{M}$). Through local retraining of $\boldsymbol{M}$ with an *upstream* dataset, the attacker connects trigger $\boldsymbol{\delta}$ with $\boldsymbol{\Phi}$, then publishes the poisoned PTM $\boldsymbol{M}^a$. The victim user downloads $\boldsymbol{M}^a$, from which she reuses ($\boldsymbol{M}^a_{1:k}$) and appends a task-specific layer (resulting in the initial customized model $\boldsymbol{M}^v_{1:k+1}$). The victim then fine-tunes it with her own dataset. The works in [50,64] have shown that after fine-tuning, triggered inputs in the downstream domain will also map to a unique OR, hence leading to mispredictions in both vision and natural language processing tasks.

We adapt the above techniques in our attack scenario. Specifically, the attacker attempts to identify *a sequence of weight bits* in $\boldsymbol{M}_{1:k}$ (i.e., feature extractor in original clean PTM) such that the perturbed version $\overline{\boldsymbol{M}}_{1:k}$ (by flipping these bits) would lead to an association of a trigger with certain pre-defined OR. In particular, we use the bit-level gradient $\nabla_{\boldsymbol{W}}$ to identify weight bits that optimize the backdoor loss (w.r.t. the fixed OR). Note that this step is done offline. At runtime, the victim leverages the clean $\boldsymbol{M}_{1:k}$ to build the customized model then flips all the offline identified bits in the victim's model at the very beginning of the victim's fine-tuning[4]. We configure a pre-trained VGG16 and set the pre-defined trigger $\boldsymbol{\delta}$, OR $\boldsymbol{\Phi}$, and backdoor loss function ℓ following previous studies [50, 64]. We observe that even after **500** weight bit flips, the attack success rate of the fined-tuned model is only **1.8%**. Our investigation reveals that such bit-level weight poisoning can *hardly* activate the pre-defined OR, leading to failed backdoor insertion.

[4] Note this is a theoretical setup to study the applicability of prior methods. In reality, typically only one bit can flip for a period of time.

5 WBP Attack Methodology

5.1 Attack Design Intuition

We treat the PTM-based backdoor feature as an effect of *OR distribution manipulation*. Particularly, the attacker manages to maximize the *distance* d between OR distribution of clean inputs and triggered inputs under the upstream dataset, which essentially pushes away the OR of triggered inputs to cross the decision boundary for *common benign features*. As such, even without knowing the victim's dataset, the downstream task input with the trigger will be largely mapped to a different label than the one of the benign sample. Under such formulation, the adapted fixed OR approach using weight bit flip (Section 4) indirectly increases d by aligning the OR of triggered inputs to a preset target.

As illustrated in Figure 2, once the OR is set (denoted as the blue triangle), moving the triggered inputs' OR (initially close to clean inputs OR) towards the fixed OR follows an optimization direction that is *not* optimal w.r.t. distance changes per bit flip. While this inefficiency in weight update is negligible in pre-training backdoor attacks where the attacker can freely control the weights (See Section 4), it is critical to optimize the efficiency of weight update (i.e., at a per-bit basis) for weight bit poisoning during fine-tuning due to the additional constraint of limiting the number of bit flips to make the attack practical (See Section 2.2). Such observation leads us to design a backdoor algorithm for WBP that *directly* optimizes OR distance between clean and triggered inputs. Specifically, such a mechanism can find the bit that maximizes the current distance d iteratively with the guidance of *distance-based loss*, leading to the identification of a minimal set of weight bits.

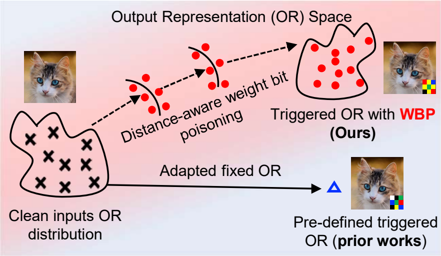
Output Representation (OR) Space
Distance-aware weight bit poisoning
Triggered OR with WBP (Ours)
Adapted fixed OR
Clean inputs OR distribution
Pre-defined triggered OR (prior works)

Fig. 2: An illustration of the attack mechanism for adapted fixed OR (using bit flips) and WBP. WBP is a distance-aware mechanism that optimizes the *distribution distance* loss between normal and triggered OR. The area with red (blue) background color denotes higher (lower) gradient values w.r.t. distance.

5.2 Maximum Mean Discrepancy Optimization

Our approach involves utilizing maximum mean discrepancy (MMD) as referenced in [19], to optimize the distributional distance between poisoned and clean ORs. Specifically, MMD is employed to measure the variance between two probability distributions.

To describe this formally, consider $\mathcal{X} \sim p$ and $\mathcal{Y} \sim q$ as two sets of samples drawn from the distinct distributions p and q. Let $\mathcal{F}$ denote a set of functions where each function f maps elements of $\mathcal{X}$ to real numbers $\mathbb{R}$. The MMD is then defined as follows:

$$\mathrm{MMD}[\mathcal{F}, p, q] := \sup_{f \in \mathcal{F}} (\mathbb{E}\boldsymbol{x} \sim p[f(\boldsymbol{x})] - \mathbb{E}\boldsymbol{y} \sim q[f(\boldsymbol{y})]) \tag{1}$$

In this equation, $\boldsymbol{x}$ and $\boldsymbol{y}$ are individual samples from $\mathcal{X}$ and $\mathcal{Y}$, respectively. The supremum calculation across the function class $\mathcal{F}$ is aimed at finding the *greatest* discrepancy between the expected function values when applied to these distributions.

An appropriate selection for $\mathcal{F}$ is the unit ball in a universal Reproducing Kernel Hilbert Space (RKHS), denoted as $\mathcal{H}$. This space is adequately comprehensive to reliably discern differences between distributions, thereby making it a fitting choice for our analysis. We further reformulate the MMD function using a concrete kernel function:

$$\begin{gathered}\mathrm{MMD}^2[\mathcal{F}, p, q] = (\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k(\boldsymbol{x}, \boldsymbol{x}')] \\ +\mathbb{E}_{\boldsymbol{y},\boldsymbol{y}'\sim q}[k(\boldsymbol{y}, \boldsymbol{y}')] - 2 * \mathbb{E}_{\boldsymbol{x}\sim p,\boldsymbol{y}\sim q}[k(\boldsymbol{x}, \boldsymbol{y})])\end{gathered} \tag{2}$$

where $\boldsymbol{x}$, $\boldsymbol{x}'$ are samples from distribution p. $\boldsymbol{y}$ and $\boldsymbol{y}'$ are from distribution q; $k(\cdot,\cdot)$ represents a continuous kernel function in $\mathcal{H}$ (e.g., Gaussian Kernel). Compared to other distance metrics such as Cosine Similarity [51] and Mean Square Error [4], MMD can effectively capture complex, non-linear relationships between distributions and exhibits resilience against transformations. Such a characteristic makes MMD a highly effective metric to measure the OR distribution distance in our attack scenario since these ORs are typically generated through non-linear transformations. To empirically compute the distance between these distributions, we use the following formulation:

$$\begin{gathered}f_{mmd} = \mathrm{MMD}^2[\mathcal{X}, \mathcal{Y}] = \frac{1}{m^2} * \\ \left(\sum_{i,j=1}^{m} k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i,j=1}^{m} k(\boldsymbol{y}_i, \boldsymbol{y}_j) - 2 * \sum_{i,j=1}^{m} k(\boldsymbol{x}_i, \boldsymbol{y}_j) \right)\end{gathered} \tag{3}$$

where m is the number of inputs. The first term within the bracket minimizes the distance among samples within distribution p. Similarly, the second term applies the same principle to the distribution q. In contrast, the final term is strategically formulated to maximize the distance between these two distributions. If the kernel function $k(\cdot,\cdot)$ in $\mathcal{H}$ is specified, we can get the empirical estimation of the distribution distance between the clean and triggered OR distributions.

5.3 Optimized Trigger Updating (OTU)

WBP operates as an iterative algorithm. In each iteration, it initially generates or updates an input trigger, followed by pinpointing a single weight bit, guided by the MMD-based distance measurement. This process is underpinned by the notion of viewing a trigger pattern as a patch on input samples. Specifically, the adversary overlays the input with the trigger at a predetermined location. The triggered sample is given by:

$$\begin{gathered}\boldsymbol{x}^* = \mathcal{A}(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{\delta}) \\ \mathcal{A}(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{\delta}) = \boldsymbol{x} \circ (1 - \boldsymbol{m}) + \boldsymbol{\delta} \circ \boldsymbol{m}\end{gathered} \tag{4}$$

Here, $\boldsymbol{x}$ represents the original, unaltered input, $\boldsymbol{m}$ is a binary mask that specifies the location of the trigger, $\boldsymbol{\delta}$ is the trigger pattern itself, and $\circ$ denotes the element-wise multiplication of matrices. The generation of the trigger is modeled as an optimization problem. During the i^{th} iteration of the trigger update, trigger $\boldsymbol{\delta_i}$ is refined as follows:

$$\begin{gathered}\boldsymbol{\delta_i} = \arg\min_{\boldsymbol{\delta}}(\ell_{\boldsymbol{\delta}}) \\ \ell_{\boldsymbol{\delta}} = 1.0 - f_{mmd}\left(\boldsymbol{M}(\boldsymbol{W}_{1:k}^{i-1}; \mathcal{X}^*), \boldsymbol{M}(\boldsymbol{W}_{1:k}^{i-1}; \mathcal{X})\right)\end{gathered} \tag{5}$$

In this context, $\mathcal{X}$ and $\mathcal{X}^*$ denote the clean and the modified (triggered) inputs, respectively. Each iteration starts with the trigger's state as determined in the preceding iteration. $\boldsymbol{W}^{i-1}_{1:k}$ represents the weights of the feature extractor of PTM $\boldsymbol{M}$, with $i-1$ targeted bits already modified. This approach involves iterative refinement of the trigger, thereby enhancing the efficiency of WBP.

5.4 Accumulative Bit Identification (ABI)

During iteration i, with the weights $\boldsymbol{W}^{i-1}_{1:k}$ and a trigger $\boldsymbol{\delta_i}$ in place, WBP locates the critical bit for the i^{th} flip. The objective is to determine the least number of bit flips to implant the backdoor. The weight bit poisoning loss ℓ_B is formulated as follows:

$$\begin{aligned} \ell_B = 1.0 & \underbrace{-f_{mmd}(\boldsymbol{M}(\boldsymbol{W}^{i-1}_{1:k}; \mathcal{X}^*), \boldsymbol{M}(\boldsymbol{W}^{i-1}_{1:k}; \mathcal{X}))}_{\text{backdoor loss: } \ell_p} \\ & + \underbrace{\alpha \cdot f_{mmd}(\boldsymbol{M}(\boldsymbol{W}^{i-1}_{1:k}; \mathcal{X}), \boldsymbol{M}(\boldsymbol{W}_{1:k}; \mathcal{X}))}_{\text{clean loss: } \ell_c} \end{aligned} \tag{6}$$

The first component, referred to as the backdoor loss ℓ_p, is designed to increase the distance between the distributions of clean and poisoned OR, similar to the loss employed in OTU. The second term, clean loss ℓ_c, ensures the model's accuracy on normal inputs by aligning the ORs of clean inputs in the poisoned model with that in the original PTM. With the current trigger $\boldsymbol{\delta_i}$ and weights $\boldsymbol{W}^{i-1}_{1:k}$, the attacker computes ℓ_B and employs back-propagation to determine the gradients $\nabla_{\boldsymbol{W}}\ell$ for each weight. Then the top ξ weight candidates are identified based on the product of their largest absolute gradients and the corresponding weight values. ABI further narrows down this search to the l most stable bits within these selected ξ weights (as detailed in Section 5.4), resulting in $\xi \times l$ bit candidates. Subsequently, the impact of each bit candidate on ℓ_B is assessed. The bit whose flipping leads to the most significant reduction in ℓ_B is chosen for the poisoning process in that iteration.

Selecting Highly-invariant Weight Bits. Since victim users fine-tune all layers with the targeted PTM, weight parameters are updated at runtime. Consider that at the offline stage the attacker identifies a weight bit b_i with a binary value ‘1‘ at bit offset o_i. The intended flipping is ‘1’→‘0’ at o_i. However, if the weight parameter containing b_i is updated such that b_i is changed to ‘0’ by the victim's regular fine-tuning, the bit flip would not be successful and the backdoor effect would not transfer.

To address this issue, WBP identifies weight bits whose values remain unchanged by fine-tuning. Specifically, it is observed in the prior studies [1, 7] that while PTM fine-tuning updates weight parameters, the gratitude of changes is very small. Consistent with this finding, we observe that bits in the *exponent segment* of weight parameters are largely unaltered. Therefore, our ABI algorithm can choose to select those weight bits to ensure successful flipping. Note that not all exponent bits are suitable for flipping. For float32, flipping a bit among the 3^{rd} to 7^{th} exponent bits of the weight from $0 \rightarrow 1$ introduces significant weight change (i.e., amplifying the weight by from 256× to $2^{128}\times$), which will completely malfunction the model [6]. As a result, ABI empirically limits the bit search within 0^{th}, 1^{st} and 2^{nd} exponent bits of the selected ξ weights.

5.5 Progressive Bit Reduction (PBR)

Employing the aforementioned OTU and ABI to generate triggers and weight bits is essentially a greedy strategy, which leaves space for further bit flip efficiency enhancement.

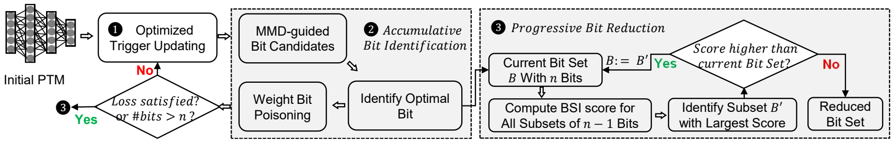
Initial PTM
1 Optimized Trigger Updating
No
Loss satisfied? or #bits > n?
Yes
3
MMD-guided Bit Candidates
2 Accumulative Bit Identification
Identify Optimal Bit
Weight Bit Poisoning
3 Progressive Bit Reduction
Current Bit Set B With n Bits
B := B'
Yes
Score higher than current Bit Set?
No
Compute BSI score for All Subsets of n − 1 Bits
Identify Subset B' with Largest Score
Reduced Bit Set

Fig. 3: The end-to-end framework of WBP.

We propose a progressive bit reduction (PBR) technique to further decrease the needed bit flips while maintaining a comparable performance (i.e., accuracy and attack success rate). The core of PBR is the evaluation of bit sets using a Bit Set Influence (**BSI**) score, defined as follows:

$$\mathrm{BSI} = 1 - \ell_B + \beta * ((1 - \ell_B)/len(B)) \tag{7}$$

where B is the bit flip set identified in ABI algorithm, $len(B)$ is the number of bits in B. The above equation uses $1 - \ell_B$ as the performance (ASR and normal accuracy) metric. It represents the performance improvement achieved by the bit flip set B. The term $((1 - \ell_B)/len(B))$ is used as the efficacy metric. It reflects the average performance improvement obtained by each individual bit flip. β is a trade-off factor determining the relative importance of performance and efficacy in the final score.

In particular, PBR identifies the least important bit in an iterative manner. As shown in Algorithm 1, to select the least important bit from the set B, PBR first calculates the BSI of the set B. For each bit in B, the corresponding BSI of the set B excluding the bit is computed. We finally get $len(B) + 1$ BSI scores and choose the set with the largest BSI. If the set with the largest BSI is still set B, WBP stops the search process. Otherwise, WBP uses the set with the largest BSI as the new set (i.e., one bit is evicted). We repeat this process until no bit is excluded, finally inducing the reduced bit flip set.

Algorithm 1: Progressive Bit Reduction

```
Input  : Initial PTM $M$, bit flip set $B$, input samples $\mathcal{X}$
Output : Reduced bit flip set $B^*$
Initialize $\mathrm{BSI}_{max}$ to $-\infty$
repeat
    $B' \leftarrow B$, $\mathrm{BSI}_{prev} \leftarrow \mathrm{BSI}_{max}$
    for $b$ in $B$ do
        Compute BSI for set $B_{/b}$, where $B_{/b}$ is $B$ excluding $b$
        if $\mathrm{BSI} > \mathrm{BSI}_{max}$ then
            Update $\mathrm{BSI}_{max} \leftarrow \mathrm{BSI}$, $B' \leftarrow B_{/b}$
    $B \leftarrow B'$
until $\mathrm{BSI}_{max} < \mathrm{BSI}_{prev}$
return $B$
```

5.6 Putting It All Together

Figure 3 shows the end-to-end framework for WBP. Essentially, the offline stage first generates a set of weight bits to flip. In real systems, rowhammer exploitation typically attacks one page (e.g., one bit) at each attack cycle (including page manipulation and the actual hammering operations) [60]. Therefore, to make the attack practical, at runtime, one weight bit is flipped at a time, with a certain interval between two consecutive exploitations. Similar to prior works [7, 60], microarchitectural side channels [16, 33, 59, 61] are performed to reverse-engineer the exact page offsets of targeted weights.

In the online stage, rowhammer attack includes three steps: 1) memory templating to collect bit flip profile that lists the locations of flippable bits in each physical memory page; 2) memory massaging that relocates the victim's model weight pages to the desired

DRAM physical locations for flipping and 3) rowhammering to induce weight bit flips. To flip the identified weight bits during fine-tuning, WBP utilizes the system-level techniques as proposed in [7, 43]. Specifically, we profiled DDR4 DIMMs in our test bed to generate a vulnerable memory bit profile, which is used to determine where to massage weight memory pages. Once rowhammer pre-attack is setup, double-sided rowhammering is performed to flip weight bits.

6 Evaluation

6.1 Experiments Setup

Datasets and Architectures. We evaluate six representative models including VGG16, ResNet18, DenseNet121, and EfficientNet-B0, ViT (Base) and DeiT (Base) from ModelZoo [30] , all pretrained with ImageNet [15]. For the downstream tasks, we utilize GTSRB [25]), CIFAR10 [31], EUROSAT [23] and SVHN [40] for CNN models. For Vision Transformer, we test four datasets including Oxford IIIT Pets (Pets37 for short) [39], CIFAR100 [31], Flower102 [41] and RESISC45 [14].

Fine-tuning and Attack Parameters. For fine-tuning, all classifiers of the target PTM are replaced with a fully connected layer. By default, the learning rate is set to 0.001 and the optimizer is SGD [47]. The fine-tuning duration is set to 5000 iterations. The image size and batch size are set to 64×64 and 128 for CNNs, and 224×224 and 32 for Transformer-based models. For image pre-processing, we apply resizing and normalizing with mean and standard deviation set to 0.5. We use a square trigger stamped in the bottom right corner of inputs. The trigger size is set to 2.44% of the input area. The attacker uses 256 images from the pre-training task (i.e., ImageNet) for WBP steps offline. The coefficient α and β are set to 1 and 3, respectively. During fine-tuning, the attacker flips one weight bit after every 50 iterations. We choose the Laplace kernel as a kernel function for computing MMD.

Evaluation Metrics. We use three metrics to evaluate the attack performance: **Attack Success Rate (ASR)** quantifies the likelihood of mis-classifying triggered inputs as an arbitrary class. Note that we do not compute the inputs with the most commonly misclassified class c. The ASR is defined as $\frac{\#samples\ (missclassfied)}{\#samples\ (not\ belongs\ to\ c)}$. **Classification Accuracy (ACC)** evaluates the performance of the model on clean inputs (i.e., normal accuracy). **Number of Bit Flips** denotes the number of weight bits flipped.

Attack schemes. We evaluate both the basic WBP and WBP with bit reduction (WBP-R). We evaluate the state-of-the-art PTM backdoor methods that manipulate fixed OR through re-training (i.e., **Fixed OR**) [64] as the baseline. Note that **Fixed OR** cannot be directly evaluated as it leverages re-training that updates all weights. To adapt to the weight-bit flipping scheme, instead of updating all weights in the backward propagation, we rank weight bits based on their changes to the loss if flipped (using the ranking algorithm described in Section 5.4), and keep flipping the top-ranked bits iteratively.

6.2 Attack Performance

Major Results. We evaluate our attack in 24 unique testing scenarios (i.e., 4 datasets × 6 architectures) under a learning rate of 0.001 and with the SGD optimizer. The results are

Table 2: Attack results for CNN models. Numbers in parentheses denote ACC changes.

Models	No. of Params.	No. of Bit Flips	GTSRB		CIFAR10		EuroSat		SVHN	
			ACC (%)	**ASR (%)**	**ACC (%)**	**ASR (%)**	**ACC (%)**	**ASR (%)**	**ACC (%)**	**ASR (%)**
VGG16	138M	**61**	99.0 (-0.1)	100.0	89.2 (-0.3)	100.0	95.6 (+0.1)	100.0	92.9 (0.0)	100.0
ResNet18	11M	**27**	98.8 (-0.3)	100.0	87.4 (0.0)	100.0	94.9 (-0.2)	100.0	92.0 (-0.6)	100.0
DenseNet121	8M	**37**	99.2 (-0.2)	100.0	90.6 (-0.2)	98.5	95.7 (-0.5)	96.6	93.7 (0.0)	100.0
EfficientNet	5M	**58**	98.3(+0.1)	99.8	86.6(-0.2)	99.7	95.9 (0.0)	100.0	91.2 (-0.1)	100.0

Table 3: Attack results for Transformer models. Numbers in parentheses denote ACC changes.

Models	No. of Params.	No. of Bit Flips	Flower102		CIFAR100		RESISC45		Pets37	
			ACC (%)	**ASR (%)**	**ACC (%)**	**ASR (%)**	**ACC (%)**	**ASR (%)**	**ACC (%)**	**ASR (%)**
ViT	86M	**50**	91.1 (-1.0)	99.8	80.6 (-0.4)	99.7	90.9 (-0.3)	100.0	91.7 (-0.1)	100.0
DeiT	86M	**65**	91.4 (0.0)	99.7	81.4 (-0.9)	99.0	90.8 (-0.3)	99.9	92.9 (+0.3)	98.0

shown in Table 2 and Table 3. Specifically, by poisoning an average of 49.6 weight bits (as few as 27 bits), WBP can achieve an extremely high ASR of 99.6%, with a negligible influence on the accuracy (0.2% drops). Furthermore, our attack consistently succeeds for all four CNNs and two Transformer-based models with $> 96.6\%$ ASR with eight downstream datasets. Our results show that WBP is highly effective in trojaning the victim's fine-tuned models, regardless of the specific downstream tasks and model architectures.

We further evaluate the performance of WBP with PBR (i.e., WBP-R). The results of WBP-R are shown in Table 4 and Table 5. The numbers in the bracket reflect the percentage of reduced bits and ASR drops compared to the WBP. WBP-R can bring a significant decrease in bit flips from 49.6 to 26.2 on average. Notably, as few as **11 bits** are required to successfully backdoor the ResNet18 model. More importantly, despite the reduction in bit flips, WBP successfully maintained a comparable ASR of 99.3%. With the minimal number of bit flips required, WBP can be easily carried out with rowhammer in real systems. Its ability to succeed consistently across different tasks and architectures makes it a critical backdoor attack in DNN models.

Table 4: WBP-R results for CNNs. Numbers in parenthesis denote ASR difference for WBP-R compared to WBP.

Models	No. of Bit Flips	ASR (%)			
		GTSRB	CIFAR10	EuroSat	SVHN
VGG16	**27 ↓56%**	99.5 (-0.5)	97.4 (-2.6)	100.0 (-0.0)	100.0 (-0.0)
ResNet18	**11 ↓59%**	99.8 (-0.2)	98.7 (-1.3)	99.9 (-0.1)	100.0 (-0.0)
DenseNet121	**26 ↓35%**	96.7 (-3.3)	98.8 (+0.3)	100.0 (+3.4)	98.0 (-2.0)
EfficientNet	**16 ↓72%**	98.5 (-1.3)	100.0 (-0.0)	100.0 (-0.0)	99.4 (-0.6)

Comparison Among Baseline Algorithms. We conduct an extensive comparison with the previous algorithm that controls the OR [64]. Essentially, the only difference between the baselines and WBP is the optimization goal (i.e., the backdoor loss function). We apply the same trigger optimization method to all algorithms using the corresponding optimization goal. Our evaluation is performed on two representative CNNs (i.e., VGG16 and ResNet18) using GTSRB and two Transformer-based models (i.e., ViT and DeiT) with CIFAR100. We measure the final ASR by changing both the learning rate and optimizer. In detail, we flip **500 bits** for Fixed OR, as compared to an average of only **26.2** bits flipped in WBP-R. Table 6 shows the ASR comparison among four configurations. The achieved average ASRs stand at 23.4% and 98.0% for **Fixed OR** and WBP-R, respectively. The results show that, despite the markedly fewer number of bit flips, the ASRs of WBP-R consistently surpass the baseline for backdooring the victim's model during fine-tuning.

Table 5: WBP-R results for vision Transformers. Numbers in parenthesis denote ASR difference for WBP-R compared to WBP.

Models	No. of Bit Flips	ASR (%)			
		Flower102	CIFAR100	RESISC45	Pets37
ViT	**29 ↓42%**	100.0 (+0.2)	100.0 (+0.3)	100.0 (0.0)	100.0 (0.0)
DeiT	**48 ↓26%**	99.5 (-0.2)	98.6 (-0.4)	99.8 (-0.1)	97.7 (-0.3)

Table 6: Attack performance for the baseline algorithms and WBP under various learning rate and optimizer settings.

Models	Attacks	ASR (%) with SGD			ASR (%) with Adam			Models	Attacks	ASR (%) with SGD			ASR (%) with Adam		
		LR:5e-4	LR:1e-3	LR:5e-3	LR:1e-5	LR:2e-5	LR:5e-5			LR:5e-4	LR:1e-3	LR:5e-3	LR:1e-5	LR:2e-5	LR:5e-5
VGG16	Fixed OR	36.4	35.4	32.9	21.5	22.3	14.8	ViT	Fixed OR	51.3	34.0	17.2	18.7	15.8	16.7
	WBP	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**		**WBP**	**100.0**	**99.7**	**99.9**	**99.8**	**99.8**	**97.4**
	WBP-R	**99.8**	**99.5**	**99.9**	**98.7**	**100.0**	**92.4**		**WBP-R**	**99.9**	**100.0**	**99.7**	**100.0**	**99.7**	**98.3**
ResNet18	Fixed OR	14.9	26.5	16.1	29	30.1	22.2	DeiT	Fixed OR	30.7	20.9	14.3	13.3	12.2	14.7
	WBP	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**		**WBP**	**99.1**	**99.0**	**98.4**	**99.5**	**98.9**	**80.3**
	WBP-R	**99.7**	**99.8**	**99.8**	**99.7**	**99.4**	**98.4**		**WBP-R**	**99.1**	**98.6**	**98.2**	**99.4**	**98.7**	**74.4**

WBP under Different Learning Rates and Optimizers. We explore the impact of learning rates and optimizers to the ASR. The results are shown in Table 6. Specifically, WBP and WBP-R can maintain the high ASRs of 98.8% and 98.0% on average across all the learning rate and optimizer pairs, showing the robustness of the proposed attack against various hyper-parameter settings. For experiments on CNNs, the ASR is 100.0% for WBP, and the ASR of WBP-R is above 92.4%. For transformer-based models, WBP and WBP-R can maintain high ASRs of 97.7% and 97.2% on average. Note that we observe a slightly lower ASR with 0.00005 learning rate plus Adam optimizer (94.4% and 90.9% ASR for WBP and WBP-R respectively). We believe that with a larger learning rate, the network's internal representations may change drastically to tailor for the current fine-tuning task, potentially undermining performance on the previous task (i.e., backdoor task). Nevertheless, we find that WBP schemes are highly successful across different hyper-parameter and architecture configurations.

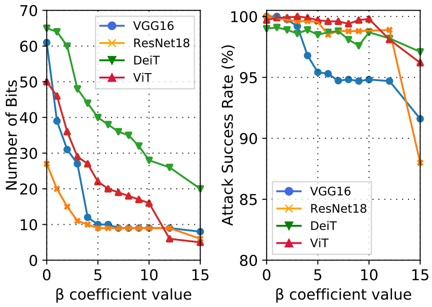

Fig. 4: The impact of β value in PBR algorithm. A zero β value represents the original WBP algorithm without PBR.

The Impact of β Value in Bit Reduction. We investigate the influence of the coefficient β on PBR using two CNNs with GTSRB and two Transformer-based models with CIFAR100. As illustrated in Figure 4, the left subplot demonstrates a decreasing trend in the number of bit flips corresponding to an increase in the β value. For CNNs, a rapid drop for $\beta \leqslant 4$, is followed by a more gradual decline, while the bits drop remains smoother for the transformer-based model with increasing β. For all the configurations, the ASR is always higher than 94.7% when the $\beta \leq 10$, showing the opportunity for considerable bit flip reduction without adversely impacting ASR. Particularly, when β is set to 10, the required bits are 16 and 28 for ViT and DeiT respectively, while maintaining a high ASR of 98.7% and 99.8%. This observation indicates the practicality of applying a more progressive bit reduction strategy for a stealthier backdoor insertion.

The Impact of Bit Flip Interval. Rowhammer is typically a sequential process where one bit is flipped at a time in DRAM [29]. In this study, we investigate the impact of the bit flip injection interval to the final ASR. In particular, we configure three different bit flip insertion times with the interval of 0, 1, and 50 iterations. The '0' interval indicates the theoretical case where all flips are injected at the same time. We test the ASR on VGG16 and GTSRB on three learning rates and the SGD optimizer. Our results show the comparable attack performance (average ASR of 99.8%, 99.6%, 98.7%) for the aforementioned configurations, respectively. Finally, such insensitivity to bit flip interval is beneficial for rowhammer as it offers sufficient time for potential timing-consuming system exploitation to manifest.

Impact of Trigger Size and Position. We conduct experiments with varying trigger sizes, ranging from 4 to 20, on the VGG16 and GTSRB. The results show all configurations can achieve an ASR above 95.8% without noticeable ACC drop (<0.1%). For trigger sizes smaller than 6 (0.9% of input area), the required bits fluctuate between 64 and 73. In contrast, a larger trigger requires fewer bits, ranging from 30 to 51. Meanwhile, our additional experimental results show that WBP is insensitive to the trigger position. Specifically, the selected four corners and one center locations consistently yielded an ASR exceeding 98.0% with a bit flip number ranging from 41 to 59.

Impact of Post Training Quantization. The victim might apply post-training quantization after fine-tuning to improve the model's inference-time computational efficiency by performing post-training quantization (e.g., 8-bit integers). As WBP manifests at *fine-tuning* time with weights represented in floating point values, such a process may potentially impact the embedded backdoor feature. We evaluate the influence of post-training quantization on WBP by performing post-training dynamic quantization for VGG16 fine-tuned with GTSRB. Our results show that a negligible impact on the ASR, with a decrease of less than 0.1% for the VGG16 and GTSRB configuration. We hypothesize that while quantization changes weight values, the relative weight perturbations due to bit flips in floating point representations are largely maintained, hence persisting the backdoor even with post-training quantization.

7 Discussion of Mitigation

In this section, we discuss and evaluate potential mitigation against our proposed attack.

Effectiveness of Existing Backdoor Defenses. *backdoor detection techniques* propose to analyze the ML model itself to uncover backdoor features and reconstruct potential triggers (e.g., [49, 55]). Additionally, *re-training-based* mitigation retrains a suspicious model with clean dataset to diminish the backdoor function [36]. Such methods take advantage of the catastrophic *forgetting phenomenon* [17]. Finally, *input filtering-based defenses* identify and remove poisoned samples with triggers from the dataset [9, 10, 22, 54], which can be applied to either training or inference stage. Our attack can inherently bypass training-time input filtering as WBP does not require poisoning of training data. Test-time filtering can detect malicious inputs, and potentially identify the triggered inputs used in WBP. Note that many prior works have studied the use of stealthy triggers to avoid such detection (e.g., [2, 3]), which is an orthogonal direction and can be potentially adopted in WBP. Finally, to defend against bit-flip based model tampering, Aegis [56] protects a DNN model by embedding internal classifiers that identify anomalies in feature maps due to weight perturbations. NeuroPots [34] restrains bit flips to trap weights that are strongly protected against tampering. Note that these techniques target on protecting static models, and cannot be directly applied in training when models are updating.

We evaluate re-training approaches to understand the capability of WBP for resisting forgetting by varying the victim's fine-tune duration. Specifically, we aggressively set the fine-tuning length to be *10* × the default configuration (i.e., 50,000 iterations) for the four CNNs with GTSRB. Our results show that the backdoors in all four models could withstand the extended duration of fine-tuning with an average ASR of 97.5% with only a 2.4% drop. We further investigate the NeuronCleanse backdoor detection [55]. The results show NeuronCleanse exhibits only 7.1% backdoor detection rate on WBP-backdoored models with VGG16 as the PTM. We believe NeuronCleanse's low performance is because it aims to discover triggers associated with a specific label, while WBP assumes no knowledge of downstream task labels (i.e., task-agnostic), and only manipulates the distribution on the OR. We finally evaluate WBP against K-arm [49], which iteratively and stochastically optimizes label selection to identify backdoor triggers. Table 7 illustrates the results where all models are fine-tuned with the GTSRB dataset. As we can see, K-arm can detect the backdoor in some of the configurations. However, it is not able to identify the controlled misclassification in WBP backdoor (as we can see from the 3rd and 4th column). Also, K-arm does not detect the backdoor in EfficientNet. We believe this is due to K-arm's ineffectiveness in exploring backdoor features in more complex model structures (e.g., compound scaling).

Table 7: Detection results of K-arm. The 3rd column shows the detected misclassification (true label to misclassified label). Last column shows the actual backdoor misclassification.

Models	Detected	Detected misclassification	Actual target
VGG16	Yes	6 → 5	All → 38
ResNet18	Yes	17 → 14	All → 16
DenseNet121	Yes	5 → 2	All → 2
EfficientNet	No	-	All → 21

Future Defense against WBP. We observe that the selection of vulnerable weight bit candidates is crucial for optimizing the loss in WBP attack. Accordingly, one potential future defense is to protect the top vulnerable weights from tampering during model training/fine-tuning. Such weight parameters could be protected either by using strong software-based error correction codes or storing them in trusted execution environments in which data integrity is maintained by hardware [38]. Under such a scheme, attacker's flips performed on protected weights would be corrected/restored. We evaluate the effect of such approaches assuming that the top-N most critical weights are locked. Our results show that such a mechanism can successfully mitigate WBP on ResNet18 and DenseNet121 by degrading the achieved ASR to 3.9% and 1.8% respectively when the top 1000 weights are protected. Moreover, though WBP exhibits stronger robustness on VGG16 and EfficientNet, this mitigation can still degrade the achieved ASR on these models by $\sim$30%. Our results show that selective weight protection can be a promising approach for future defense against WBP.

8 Conclusion

This paper presents the first task-agnostic backdoor attack–WBP, which trojans victim models during training-time by exploiting hardware-based weight bit flips. WBP utilizes a novel distance-guided algorithm that identifies and flips limited weight bits to maximize the distance of output representations for normal and triggered inputs, leading to misclassification for triggered inputs in the victim's model fine-tuned from a public PTM. Our work reveals a new research direction of runtime bit-level weight positioning attack, which motivates future works for training-time hardware-based weight perturbations.

Acknowledgements

This work is supported in part by U.S. National Science Foundation under SaTC-2019536 and CNS-2147217.

References

1. Al Rafi, M., Feng, Y., Yao, F., Tang, M., Jeon, H.: Decepticon: Attacking secrets of transformers. In: IEEE International Symposium on Workload Characterization (IISWC). pp. 128–139 (2023)
2. Bai, J., Gao, K., Gong, D., Xia, S.T., Li, Z., Liu, W.: Hardly perceptible trojan attack against neural networks with bit flips. In: European Conference on Computer Vision (ECCV). pp. 104–121 (2022)
3. Bai, J., Wu, B., Zhang, Y., Li, Y., Li, Z., Xia, S.: Targeted attack against deep neural networks via flipping limited weight bits. In: International Conference on Learning Representations (ICLR) (2021)
4. Bickel, P., Doksum, K.: Mathematical Statistics: Basic Ideas and Selected Topics. Prentice Hall (2001)
5. Brown, T.B., Mané, D., Roy, A., Abadi, M., Gilmer, J.: Adversarial patch. arXiv preprint arXiv:1712.09665 (2017)
6. Cai, K., Chowdhuryy, M.H.I., Zhang, Z., Yao, F.: Seeds of seed: Nmt-stroke: Diverting neural machine translation through hardware-based faults. In: International Symposium on Secure and Private Execution Environment Design (SEED). pp. 76–82 (2021)
7. Cai, K., Chowdhuryy, M.H.I., Zhenkai, Z., Yao, F.: Deepvenom: Persistent dnn backdoors exploiting transient weight perturbations in memories. In: 2024 IEEE Symposium on Security and Privacy (SP). pp. 244–244 (2024)
8. Cai, K., Zhang, Z., Yao, F.: On the feasibility of training-time trojan attacks through hardware-based faults in memory. In: Hardware Oriented Security and Trust (HOST). pp. 133–136 (2022)
9. Chan, A., Ong, Y.S.: Poison as a cure: Detecting & neutralizing variable-sized backdoor attacks in deep neural networks. arXiv preprint arXiv:1911.08040 (2019)
10. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I.M., Srivastava, B.: Detecting backdoor attacks on deep neural networks by activation clustering. In: Workshop on AAAI Conference on Artificial Intelligence (AAAI). vol. 2301 (2019)
11. Chen, H., Fu, C., Zhao, J., Koushanfar, F.: Proflip: Targeted trojan attack with progressive bit flips. In: IEEE/CVF International Conference on Computer Vision (ICCV). pp. 7718–7727 (2021)
12. Chen, K., Meng, Y., Sun, X., Guo, S., Zhang, T., Li, J., Fan, C.: Badpre: Task-agnostic backdoor attacks to pre-trained NLP foundation models. In: International Conference on Learning Representations (ICLR) (2022)
13. Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017)
14. Cheng, G., Han, J., Lu, X.: Remote sensing image scene classification: Benchmark and state of the art. Proceedings of the IEEE pp. 1865–1883 (2017)
15. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: computer vision and pattern recognition (CVPR). pp. 248–255 (2009)
16. Fang, H., Dayapule, S.S., Yao, F., Doroslovački, M., Venkataramani, G.: A noise-resilient detection method against advanced cache timing channel attack. In: Asilomar Conference on Signals, Systems, and Computers. pp. 237–241 (2018)

17. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences pp. 128–135 (1999)
18. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Bengio, Y., LeCun, Y. (eds.) International Conference on Learning Representations (ICLR) (2015)
19. Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.: A kernel two-sample test. The Journal of Machine Learning Research pp. 723–773 (2012)
20. Gruss, D., Lipp, M., Schwarz, M., Genkin, D., Juffinger, J., O'Connell, S., Schoechl, W., Yarom, Y.: Another flip in the wall of rowhammer defenses. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 245–261. IEEE (2018)
21. Gu, T., Dolan-Gavitt, B., Garg, S.: Badnets: Identifying vulnerabilities in the machine learning model supply chain. CoRR **abs/1708.06733** (2017)
22. Hayase, J., Kong, W.: Spectre: Defending against backdoor attacks using robust covariance estimation. In: International Conference on Machine Learning (ICLR) (2020)
23. Helber, P., Bischke, B., Dengel, A., Borth, D.: Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing pp. 2217–2226 (2019)
24. Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M.M.A., Yang, Y., Zhou, Y.: Deep learning scaling is predictable. Empirically. arXiv p. 2 (2017)
25. Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., Igel, C.: Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In: International Joint Conference on Neural Networks. No. 1288 (2013)
26. Jattke, P., van der Veen, V., Frigo, P., Gunter, S., Razavi, K.: Blacksmith: Scalable rowhammering in the frequency domain. In: IEEE Symposium on Security and Privacy (SP). vol. 1 (2022)
27. Jeon, M., Venkataraman, S., Phanishayee, A., Qian, J., Xiao, W., Yang, F.: In: USENIX Annual Technical Conference. pp. 947–960 (2019)
28. Jia, J., Liu, Y., Gong, N.Z.: BadEncoder: Backdoor attacks to pre-trained encoders in self-supervised learning. In: IEEE Symposium on Security and Privacy (SP) (2022)
29. Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J.H., Lee, D., Wilkerson, C., Lai, K., Mutlu, O.: Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. ACM SIGARCH Computer Architecture News pp. 361–372 (2014)
30. Koh, J.Y.: Model zoo: Discover open source deep learning code and pretrained models. `https://modelzoo.co/`
31. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
32. Li, L., Song, D., Li, X., Zeng, J., Ma, R., Qiu, X.: Backdoor attacks on pre-trained models by layerwise weight poisoning. In: Empirical Methods in Natural Language Processing (EMNLP) (2021)
33. Liu, F., Yarom, Y., Ge, Q., Heiser, G., Lee, R.B.: Last-level cache side-channel attacks are practical. In: IEEE symposium on security and privacy (SP). pp. 605–622 (2015)
34. Liu, Q., Yin, J., Wen, W., Yang, C., Sha, S.: {NeuroPots}: Realtime proactive defense against {Bit-Flip} attacks in neural networks. In: USENIX Security Symposium. pp. 6347–6364 (2023)
35. Liu, Y., Ma, S., Aafer, Y., Lee, W., Zhai, J., Wang, W., Zhang, X.: Trojaning attack on neural networks. In: Network and Distributed System Security Symposium (NDSS) (2018)
36. Liu, Y., Xie, Y., Srivastava, A.: Neural trojans. In: International Conference on Computer Design (ICCD). pp. 45–48 (2017)
37. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, pp. 109–165 (1989)

38. McKeen, F., Alexandrovich, I., Anati, I., Caspi, D., Johnson, S., Leslie-Hurd, R., Rozas, C.: Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In: Hardware and Architectural Support for Security and Privacy (HASP), pp. 1–9 (2016)
39. monkeydoodle@gmail.com: The oxford-iiit pet dataset dataset. `https://universe.roboflow.com/monkeydoodle-gmail-com/the-oxford-iiit-pet-dataset` (2022)
40. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
41. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: Indian Conference on Computer Vision, Graphics & Image Processing. pp. 722–729 (2008)
42. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on knowledge and data engineering pp. 1345–1359 (2009)
43. Rakin, A.S., Chowdhuryy, M.H.I., Yao, F., Fan, D.: Deepsteal: Advanced model extractions leveraging efficient weight stealing in memories. IEEE Security and Privacy (SP) pp. 1157–1174 (2022)
44. Rakin, A.S., He, Z., Fan, D.: Bit-flip attack: Crushing neural network with progressive bit search. In: International Conference on Computer Vision (ICCV). pp. 1211–1220 (2019)
45. Rakin, A.S., He, Z., Fan, D.: TBT: targeted neural network attack with bit trojan. In: Computer Vision and Pattern Recognition (CVPR). pp. 13195–13204 (2020)
46. Rakin, A.S., He, Z., Li, J., Yao, F., Chakrabarti, C., Fan, D.: T-bfa: Targeted bit-flip adversarial weight attack. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 7928–7939 (2021)
47. Ruder, S.: An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747 (2016)
48. Seaborn, M., Dullien, T.: Exploiting the dram rowhammer bug to gain kernel privileges. Black Hat p. 71 (2015)
49. Shen, G., Liu, Y., Tao, G., An, S., Xu, Q., Cheng, S., Ma, S., Zhang, X.: Backdoor scanning for deep neural networks through k-arm optimization. In: International Conference on Machine Learning (ICML) (2021)
50. Shen, L., Ji, S., Zhang, X., Li, J., Chen, J., Shi, J., Fang, C., Yin, J., Wang, T.: Backdoor pre-trained models can transfer to all. In: ACM SIGSAC Conference on Computer and Communications Security (CCS). pp. 3141–3158 (2021)
51. Singhal, A., et al.: Modern information retrieval: A brief overview. IEEE Data Eng. Bull. pp. 35–43 (2001)
52. Tol, M.C., Islam, S., Adiletta, A.J., Sunar, B., Zhang, Z.: Don't knock! rowhammer at the backdoor of dnn models. In: Dependable Systems and Networks (DSN). pp. 109–122 (2023)
53. Tolpegin, V., Truex, S., Gursoy, M.E., Liu, L.: Data poisoning attacks against federated learning systems. In: European Symposium on Research in Computer Security (ESORICS). pp. 480–501 (2020)
54. Tran, B., Li, J., Madry, A.: Spectral signatures in backdoor attacks. Advances in neural information processing systems (NeurIPS) **31** (2018)
55. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: IEEE Symposium on Security and Privacy (SP). pp. 707–723 (2019)
56. Wang, J., Zhang, Z., Wang, M., Qiu, H., Zhang, T., Li, Q., Li, Z., Wei, T., Zhang, C.: Aegis: Mitigating targeted bit-flip attacks against deep neural networks. In: USENIX Security Symposium. pp. 2329–2346 (2023)
57. Wang, S., Nepal, S., Rudolph, C., Grobler, M., Chen, S., Chen, T.: Backdoor attacks against transfer learning with pre-trained deep learning models. IEEE Transactions on Services Computing (TSC) pp. 1526–1539 (2020)

58. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Brew, J.: Huggingface's transformers: State-of-the-art natural language processing. CoRR (2019)
59. Yao, F., Fang, H., Doroslovački, M., Venkataramani, G.: Cotsknight: Practical defense against cache timing channel attacks using cache monitoring and partitioning technologies. In: Hardware Oriented Security and Trust (HOST). pp. 121–130 (2019)
60. Yao, F., Rakin, A.S., Fan, D.: Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In: USENIX Security Symposium. pp. 1463–1480 (2020)
61. Yao, F., Venkataramani, G., Doroslovački, M.: Covert timing channels exploiting non-uniform memory access based architectures. In: Proceedings of the on Great Lakes Symposium on VLSI 2017. pp. 155–160 (2017)
62. Yao, Y., Li, H., Zheng, H., Zhao, B.Y.: Latent backdoor attacks on deep neural networks. In: ACM SIGSAC Conference on Computer and Communications Security (CCS). pp. 2041–2055 (2019)
63. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? Advances in neural information processing systems NeurIPS **27** (2014)
64. Zhang, Z., Xiao, G., Li, Y., Lv, T., Qi, F., Liu, Z., Wang, Y., Jiang, X., Sun, M.: Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. Machine Intelligence Research pp. 180–193 (2023)