DBFB: BIT-AWARE CRITICAL WEIGHT MONITORING FOR DEFENDING BIT-FLIP BACKDOOR ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Bit-flip backdoor attacks (BFBAs) constitute a sparse post-deployment threat in which an adversary flips only a few weight bits to activate targeted behavior while seeking to preserve clean accuracy. Existing bit-flip defenses are mainly designed for random faults, global accuracy degradation, or broadly sensitive parameters, and are therefore poorly aligned with the trigger-conditioned and highly localized nature of BFBAs. We propose DBFB, a selective defense that identifies and monitors BFBA-critical weights through a progressive neuron-to-weight-to-bit analysis. DBFB first locates attack-sensitive neurons by combining cross-class support with local boundary sensitivity and competition-direction concentration. It then maps these neurons to path-constrained candidate weights and evaluates their decision impact and structural propagation support. Finally, it re-ranks candidates using signed bit-induced changes and bounded estimates of relative margin erosion. The selected weights are stored with their clean reference values and checked before inference so that corrupted monitored weights can be detected and restored to their exact clean values. Across CNN and ViT architectures, four datasets, and three representative BFBAs, DBFB monitors only 5,000 weights, corresponding to 0.025% of model parameters on average. In the main evaluation, DBFB achieves 100% Hit Rate and 0% ASR for OneFlip on all four model–dataset pairs, and substantially suppresses the other evaluated attacks. The measured online check takes less than 0.1 ms, with approximately 39.1 KiB of index-and-reference storage.

1 INTRODUCTION

Deep neural networks (DNNs) are increasingly deployed in security-sensitive applications, where backdoor attacks threaten model integrity by inducing attacker-specified outputs on triggered inputs while preserving benign behavior (Li et al., 2024; Liu et al., 2025). Conventional backdoors typically require poisoning training data or interfering with training (Dao et al., 2026; Liu et al., 2023b), enabling pre-deployment data inspection, model auditing, or purification (Goldblum et al., 2023; Xiao et al., 2026). These defenses do not cover attacks that alter a model after release.

Bit-flip attacks (BFAs) create such a post-deployment threat by modifying the binary representation of runtime model parameters (Chen et al., 2025a). Through Rowhammer-style fault injection or other low-level primitives, an adversary can perturb selected weight bits without access to the training pipeline (He et al., 2025; Chen et al., 2025b). Recent bit-flip backdoor attacks (BFBAs) further show that these perturbations can be optimized for trigger-conditioned behavior rather than global model failure. ETBT progressively searches for target-sensitive bits, Deep-TROJ introduces structured parameter corruption through memory-level weight replacement, and OneFlip demonstrates that a single carefully selected bit can be sufficient to implant a functional backdoor (Jin et al., 2024; Ahmed et al., 2024; Li et al., 2025). Such attacks are supported by hardware–software attack chains involving memory layout manipulation, physical-address inference, or huge-page allocation, which can increase the likelihood that selected model parameters occupy vulnerable memory locations (Kim et al., 2024; Bölcskei et al., 2025).

Existing bit-flip defenses are not fully aligned with the trigger-conditioned objective of BFBAs. System- and model-level protections improve fault tolerance, harden parameter representations, or prioritize weights whose corruption degrades overall accuracy (Zheng et al., 2025; Velcický et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2024; Chen et al., 2025b). Runtime defenses provide more direct protection, but may require auxiliary structures, depend on particular low-level attack paths, or rely on range-based correction that does not ensure backdoor removal (Liu et al., 2023a; Saxena et al., 2023; Abumandour et al., 2025). Full-model integrity checking offers broader protection but can incur substantial runtime overhead, while detecting parameter corruption alone does not guarantee effective suppression of the resulting backdoor behavior. The central challenge is therefore to defend deployed models against BFBAs under limited computational and storage resources while preserving clean prediction performance.

The difficulty is that attack criticality is not equivalent to clean functional importance. Activation magnitude, gradient magnitude, and output contribution primarily characterize support for benign prediction (Ganguli & Chong, 2024; Sapkota & Bhattarai, 2023; Nam & Lee, 2025). They do not establish whether a discrete perturbation can weaken the clean-class margin, favor a consistent competing class, and propagate through the network while leaving clean behavior largely intact. Offline profiling provides only the clean model and clean data; the future trigger, target class, and corrupted locations remain unknown. Since exhaustive weight-bit evaluation is infeasible (Zhang et al., 2025), the defender needs a clean-data surrogate for potential BFBA exploitability rather than generic importance.

The sparsity of BFBAs provides a clue: effective attacks concentrate modifications in relatively few weights and layers (Abumandour et al., 2025; Wei et al., 2024). Their effects follow a functional chain from bit-induced numerical changes to altered neuron responses and, through decision-relevant computations, to class competition. DBFB reasons in the reverse direction: it first identifies response locations capable of adverse boundary changes, then traces weights that can produce and propagate those changes, and finally evaluates their discrete bit-level exploitability. This progression contracts the search space without enumerating every weight-bit pair.

Based on this rationale, we propose DBFB, which selects potentially exploitable weights for monitoring using only the clean model and clean data. It first summarizes each neuron's support for clean predictions across classes and uses this information to restrict the candidate neurons. A local gate perturbation then tests whether a candidate can reduce the clean-class margin and favor a consistent competing class. DBFB collects the selected neurons' input- and output-side weights and ranks them by scalar margin sensitivity and the sensitivity of nearby computation blocks. Finally, it evaluates the signed numerical changes permitted by individual bits and aggregates their predicted margin effects into a weight-level score. The top-K_w weights form the monitored set. DBFB synchronously compares only these weights with their exact clean reference values before each guarded inference and restores any mismatched weight to its clean reference value before the forward pass begins. We evaluate the selected set both by its overlap with attack-modified weights and by how much the attack weakens when the covered modifications are undone.

The main contributions are summarized as follows:

- We introduce DBFB, a clean-only post-deployment defense that selectively monitors original weights in an unchanged model, prioritizing potential trigger-conditioned decision shifts rather than generic clean-task importance or fault sensitivity.
- We develop a hierarchical localization method that combines cross-class-constrained neuron sensitivity, path-aware weight scoring, and bit-level fragility. This coarse-to-fine procedure connects attack-relevant functional responses to finite-precision parameters without exhaustively evaluating the full weight-bit space.
- We evaluate DBFB across CNN and ViT architectures, four datasets, and three BFBAs. Monitoring 5,000 weights (approximately 0.025% of parameters on average), DBFB effectively suppressing other evaluation attacks and incurs less than 0.1 ms of online checking latency.

2 Related Work

Bit-Flip Attacks on DNNs. BFAs intentionally manipulate bits in storage or computation units to change system behavior. Unlike random soft errors, they have explicit attack objectives and may target selected memory locations using voltage disturbance, Rowhammer, or laser-based fault injection (Derya et al., 2025; Breier & Hou, 2022; Boixaderas et al., 2024). In DNNs, early attacks mainly searched for weight bits whose corruption causes large accuracy degradation, often using gradient

or sensitivity information to allocate a small flip budget (Kim et al., 2024). Practical realizations may additionally manipulate the physical memory layout so that target weights are placed in vulnerable DRAM locations before the desired flips are triggered (Ahmed et al., 2024). BFBAs differ from conventional BFAs because they optimize sparse bit changes for a target behavior while seeking to preserve clean performance. ETBT, Deep-TROJ, and OneFlip represent progressive multi-bit search, structured memory-level corruption, and strict single-bit backdoor injection, respectively (Jin et al., 2024; Ahmed et al., 2024; Li et al., 2025).

Backdoor Attacks. Backdoor attacks train or modify a model so that it behaves normally on benign inputs but predicts an attacker-specified target when a trigger is present. Training-stage attacks implant this association through poisoned samples, clean-label constructions, targeted regularization, or parameter manipulation (Goldblum et al., 2023; Zhao et al., 2025a; Xiao et al., 2026). Their dependence on the training data or optimization process enables defenses based on data sanitization, model inspection, purification, or pre-deployment fine-tuning (Zhou et al., 2024; Xue et al., 2023). Inference-stage BFBAs operate under a different assumption: the clean model is first deployed and the adversary subsequently modifies its weight bits (Ahmed et al., 2024; Zhao et al., 2025b). Their sparse nature makes broad clean-performance monitoring a weak signal, because the compromised model can retain normal accuracy while only triggered inputs are directed to the target category.

Defenses against Bit Flips. Existing defenses against bit-flip attacks operate at the hardware, model, and runtime levels. Hardware redundancy, error-correction codes, and memory isolation provide broad fault tolerance, but are largely model-agnostic and do not determine which corrupted parameters are most responsible for a targeted backdoor behavior (Qian et al., 2023; Zheng et al., 2025). Model-level defenses improve robustness through retraining, parameter encoding, executable protection, or verification, but often require modifying the model or optimize resistance (Wei et al., 2024; Velcický et al., 2024; Chen et al., 2025b; Zhang et al., 2025). Runtime defenses are more relevant to deployed models, yet existing designs remain limited for budgeted BFBA protection. NeuroPots introduces canary neurons and therefore changes the deployed architecture, while PT-Guard is tied to a specific page-table-based attack path (Liu et al., 2023a; Saxena et al., 2023). WeightSentry, the closest method to our setting, broadly checks whether weights leave their estimated clean ranges, but does not explicitly prioritize the small subset of original weights that are functionally decisive for trigger-conditioned behavior, nor does range-based correction guarantee restoration to the exact pre-attack values (Abumandour et al., 2025). DBFB addresses this selection problem through clean-data neuron analysis, path-aware weight scoring, and bit-level re-ranking, without retraining or architectural changes.

3 Threat Model

Attacker ability. We consider an inference-phase adversary with no access to training data, training procedures, or pre-deployment model storage. After deployment, the attacker can flip a limited number of bits in runtime model memory using Rowhammer-style injection or another low-level primitive. The attacker has white-box knowledge of the architecture, parameters, and inference framework and can compute bit-flip gradients. Its objective is to map triggered inputs to a specified target while preserving clean accuracy and minimizing flips.

Defender goal. The defender aims to suppress BFBAs while preserving clean accuracy under a monitoring budget of K_w weights. It has access to the clean model and a small clean validation set D_c, but no prior knowledge of triggers, target classes, or corrupted locations. The identifiers and exact clean values of the selected weights are stored in a trusted manifest $\mathcal{M}$. The attacker may corrupt runtime weights, including monitored ones, but cannot modify $\mathcal{M}$, the monitor code, or its pre-inference hook. Before each guarded forward pass, DBFB checks the selected weights against their references and restores mismatched weights, leaving unmonitored parameters unchanged.

4 Design of DBFB

DBFB uses the clean model F_θ and validation set D_c to construct a risk ranking before deployment. Because the trigger, target class, and parameter corruptions are unknown, the ranking cannot be optimized against a particular backdoor. Instead, it captures three properties shared by the evaluated BFBAs: changes in class competition, sensitivity of decision-relevant computation paths, and the

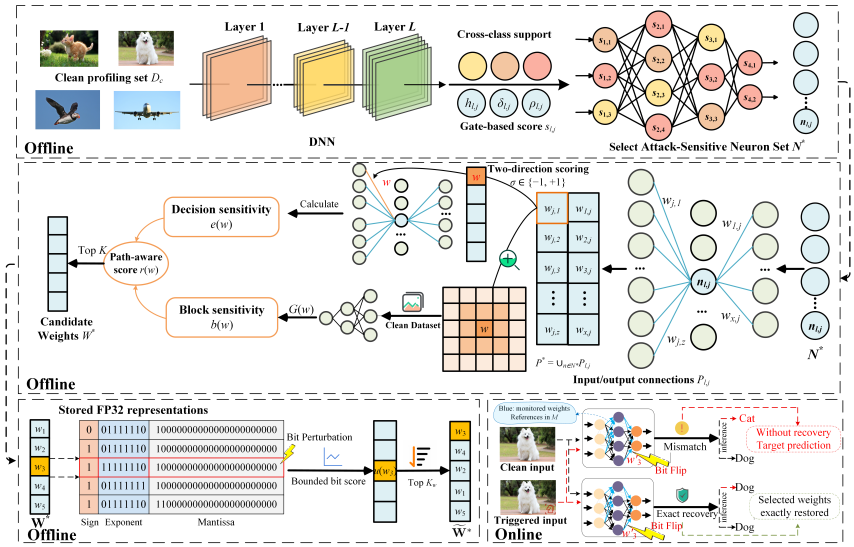


Figure 1: Overview of DBFB.

numerical effects of concrete bit flips. DBFB therefore reduces the candidate space progressively at the neuron, weight, and bit levels, as illustrated in Fig. 1. Specifically, at the neuron level, DBFB separates broad support for clean decisions from local perturbation sensitivity and retains units that can weaken the clean-class margin while inducing a concentrated competing direction. At the weight level, it traces adjacent parameters and evaluates their scalar decision impact and local block sensitivity. At the bit level, it evaluates signed bit changes through bounded estimates of relative margin erosion. The top-K_w weights form the monitored set $\widetilde{\mathbf{W}}^*$.

4.1 Cross-Class-Constrained Attack-Sensitive Neuron Localization

Although a BFBA ultimately modifies weight bits, a flipped bit becomes functionally effective only when the resulting numerical perturbation induces an internal response that changes the final class competition.Directly ranking all weights before identifying such response locations is computationally expensive, as it also considers parameters involved in unrelated computations. DBFB first identifies neurons whose local perturbations can weaken the clean-class margin, then restricts candidate weights to their adjacent connections. This reduces the number of parameters considered in the subsequent weight- and bit-level analysis.

A conventional neuron-importance measure is insufficient for this purpose. Activation magnitude, gradient magnitude, and attribution scores primarily identify units that support benign prediction. A neuron can be highly important because it consistently contributes to the discrimination of clean classes, but this does not by itself imply that perturbing the neuron will create a localized, target-oriented decision shift. Under a limited monitoring budget, indiscriminately tracing such neurons would direct the subsequent weight search toward normal functional pathways. We therefore separate two properties that are often conflated by generic importance measures: how broadly a neuron supports clean decisions and how its local perturbation changes the decision boundary.

Cross-class support as a candidate prior. We first construct a class-balanced support set $\mathcal{D}_s = \bigcup_{c=1}^{C} \mathcal{D}_s^{(c)}$ from D_c. For a clean sample x, the class-discriminative relevance of neuron (l, j) is defined as

$$R_{l,j}(x) = \mathcal{A}_{l,j}\big(\mathcal{F}_{0\to l}(x;\theta), f(x;\theta)\big), \tag{1}$$

where $\mathcal{F}_{0\to l}$ denotes the forward representation up to layer l, and $\mathcal{A}_{l,j}$ is an architecture-adapted activation–gradient relevance operator. For convolutional layers, the response of a channel is aggregated over its spatial positions; for ViTs, the response is evaluated along the representation used by the classification readout. Importantly, $R_{l,j}(x)$ is used to characterize benign support rather than

attack risk itself. We compare responses only within the same layer so that differences in activation scale across architectures and depths do not dominate the ranking.

For each sample, let $p_{l,j}(x)$ be the descending rank of $R_{l,j}(x)$ among the M_l units in layer l, with ties resolved by neuron index. We assign

$$g_{l,j}^{(s)}(x) = \begin{cases} 2, & p_{l,j}(x) \le \lceil \alpha M_l \rceil, \\ 1, & \lceil \alpha M_l \rceil < p_{l,j}(x) \le \lceil 2\alpha M_l \rceil, \\ 0, & p_{l,j}(x) > \lceil 2\alpha M_l \rceil, \end{cases} \tag{2}$$

where $\alpha = 1/4$. Rank-based grouping provides disjoint support states even when relevance values coincide.

Let $F_r(l, j, c)$ denote the number of samples from class c for which neuron (l, j) is assigned to state r, and let $N_c = |\mathcal{D}_s^{(c)}|$. We summarize the support of the neuron within class c as

$$g_{l,j}^{(c)} = \begin{cases} 0, & F_0(l,j,c) \ge N_c/2, \\ 1, & F_1(l,j,c) \ge F_2(l,j,c) \text{ and } F_0(l,j,c) < N_c/2, \\ 2, & \text{otherwise.} \end{cases} \tag{3}$$

We summarize cross-class support by averaging the class-level states:

$$u_{l,j} = \frac{1}{C} \sum_{c=1}^{C} g_{l,j}^{(c)}. \tag{4}$$

This permutation-invariant score lies in $[0, 2]$. If n_r counts classes assigned state r, then $u_{l,j} = 1 + (n_2 - n_0)/C$. The filter $\mathcal{N}_{\text{ana},l} = \{j : u_{l,j} \le 1\}$ therefore imposes a relative-support prior ($n_2 \le n_0$), not a certificate that excluded neurons are unexploitable. Classification-head neurons remain eligible, but their weights must still pass both subsequent rankings under the same budget.

Target-agnostic boundary stress test. Cross-class support identifies where the search should be concentrated, but it does not establish that perturbing a retained neuron produces an attack-relevant effect. We therefore stress-test each candidate against the local decision boundary. For a labeled clean sample (x, y) in an analysis batch $B \subseteq D_c$, we define the clean-class margin as

$$m(x) = f_y(x; \theta) - \max_{k \ne y} f_k(x; \theta). \tag{5}$$

Here f denotes pre-softmax logits. We construct a fixed, class-balanced analysis batch $B \subseteq D_c$ from samples with $m(x) > 0$ and reuse it for all candidate scores. The margin is independent of the attacker's target. We introduce a virtual gate $z_{l,j}$ on activation $a_{l,j}(x)$, such that $\widetilde{a}_{l,j}(x) = z_{l,j} a_{l,j}(x)$ and $z_{l,j} = 1$ corresponds to the clean model. The local boundary sensitivity is

$$h_{l,j} = \frac{1}{|B|} \sum_{(x,y) \in B} \left| \frac{\partial(-m(x))}{\partial z_{l,j}} \bigg|_{z_{l,j}=1} \right|. \tag{6}$$

A large $h_{l,j}$ indicates local margin sensitivity, but a first-order gradient alone does not show whether a realizable perturbation actually weakens the clean-class advantage. It may also assign a high score to responses whose adverse directions vary substantially across samples. To verify the realized boundary effect, we perturb the gate along the batch-averaged adverse direction, and measure only the resulting reductions in the clean-class margin:

$$z'_{l,j} = 1 + \varepsilon \operatorname{sign}\left(\frac{1}{|B|} \sum_{(x,y) \in B} \frac{\partial(-m(x))}{\partial z_{l,j}} \bigg|_{z_{l,j}=1} \right), \tag{7}$$

$$\delta_{l,j} = \frac{1}{|B|} \sum_{(x,y) \in B} \left[m(x; z_{l,j} = 1) - m(x; z_{l,j} = z'_{l,j}) \right]_+. \tag{8}$$

The positive-part operator prevents samples whose margins increase from canceling the adverse effects observed on other samples. Thus, $h_{l,j}$ measures local susceptibility, whereas $\delta_{l,j}$ verifies that this susceptibility can be converted into an actual boundary degradation.

Margin degradation need not favor a consistent competing class. We therefore compare relative logits before and after the gate perturbation. Let $f_k^0(x)$ and $f_k^1(x)$ denote the corresponding logits, and define $v_{l,j,k}(x) = [(f_k^1 - f_y^1) - (f_k^0 - f_y^0)]_+$ for $k \neq y$, with $v_{l,j,y}(x) = 0$. The directional score is

$$\pi_{l,j}(k) = \frac{1}{|B|} \sum_{(x,y)\in B} \frac{v_{l,j,k}(x)}{\sum_{r=1}^{C} v_{l,j,r}(x)}, \qquad \rho_{l,j} = \max_k \pi_{l,j}(k),$$

where a zero denominator contributes zero. Each sample distributes its contribution across classes according to their relative-logit gains. Thus, $\rho_{l,j}$ measures shared perturbation-induced competitive gains rather than pre-existing competitor frequency.

DBFB combines these complementary signals as

$$s_{l,j} = \frac{\big(\mathrm{Norm}(h_{l,j})\,\mathrm{Norm}(\delta_{l,j})\,\rho_{l,j}\big)^{1/3}}{1 + \lambda u_{l,j}}, \tag{9}$$

Here, $\mathrm{Norm}(v_j) = v_j / \max_{i\in\mathcal{N}_{\mathrm{ana},l}} v_i$ within each layer, with zero returned when the maximum is zero. The geometric mean combines local sensitivity, realized margin reduction, and directional concentration; the denominator penalizes common clean support. We retain the top $n_l = \min\{\lceil K_l M_l \rceil, |\mathcal{N}_{\mathrm{ana},l}|\}$ eligible neurons per hidden layer, resolving ties by neuron index. Together with all classification-head neurons, they form $\mathbf{N}^*$.

4.2 Path-Aware Critical-Weight Localization

The neuron-level stage identifies where an adverse response can emerge, but neurons are not directly monitorable memory objects and BFBAs target the weight bits rather than the neuron activation values. A selected neuron may be connected to hundreds or thousands of scalar parameters, only a small subset of which can effectively produce or transmit the identified response. The next problem is therefore to map the selected neuron set N* to concrete weight locations without returning to a model-wide parameter search.

For an attack-sensitive neuron $n_{l,j} \in \mathbf{N}^*$, let $\mathbf{I}_{l,j}$ and $\mathbf{O}_{l,j}$ denote its input-side and output-side connections, respectively. Input-side weights determine whether an abnormal activation can be formed, whereas output-side weights determine whether it can be retained, amplified, or propagated toward the classifier. Restricting the search to one side may therefore omit either the formation or transmission segment of the attack pathway. DBFB thus constructs the local candidate set $\mathbf{P}_{l,j} = \mathbf{I}_{l,j} \cup \mathbf{O}_{l,j}$ and aggregates these sets into $\mathbf{P}^* = \bigcup_{n_{l,j}\in\mathbf{N}^*} \mathbf{P}_{l,j}$. Adjacency to an attack-sensitive neuron is necessary for candidate generation, but it is not sufficient for final localization. Parameters connected to the same neuron can differ substantially in both their direct influence on the decision margin and the ability of their surrounding computation path to transmit a perturbation. DBFB therefore evaluates each candidate from two complementary perspectives: *scalar decision impact* and *local propagation support*. Let $D_w(x) = \partial(-m(x))/\partial w$ denote the scalar margin gradient at the clean model. To account for both perturbation directions, we evaluate

$$e(w) = \max_{\sigma\in\{-1,+1\}} \frac{1}{|B|} \sum_{(x,y)\in B} \big[\sigma\xi D_w(x)\big]_+, \tag{10}$$

where $\xi > 0$ is a shared scalar-sensitivity scale. Writing E_B for the empirical average, Eq. (10) can be rewritten as $e(w) = \frac{\xi}{2}(\mathrm{E}_B|D_w| + |\mathrm{E}_B D_w|)$. It therefore combines gradient magnitude with directional agreement across samples, rather than magnitude alone. Whether a stored bit can realize an adverse direction is assessed in the final stage.

To account for this structural context, DBFB constructs a bounded path-conditioned parameter block $\mathcal{G}(w)$ anchored at the computation-graph edge containing w. The block includes the local operator groups that produce and consume the activation connected by the candidate edge. Its propagation support is measured by

$$b(w) = \frac{1}{|B|} \sum_{(x,y)\in B} \big\|\nabla_{\mathcal{G}(w)}(-m(x))\big\|_2 . \tag{11}$$

The shared batch B is used for all candidates. By Euclidean duality, each norm in Eq. (11) is the largest first-order margin reduction over unit-ℓ_2 perturbations supported on $\mathcal{G}(w)$. Thus, $b(w)$

averages sample-wise local worst-case sensitivities, incorporating downstream effects through the final-margin gradient. It measures total block response under a fixed Euclidean budget, rather than a size-normalized or jointly realizable bit attack; $e(w)$ and the final bit scores retain candidate-specific information.

The block follows connected rows and columns in fully connected and projection layers, channel-wise convolutional slices, and the corresponding ViT projection, attention, or feed-forward groups. Appendix specifies the architecture-dependent mappings.

The two scores answer different questions: $e(w)$ measures the scalar parameter's effect on the class margin, whereas $b(w)$ measures support from the surrounding computation structure. We combine them and retain the top-K candidates:

$$r(w) = \log\big((1 + e(w))(1 + b(w))\big), \tag{12}$$

$$\mathbf{W}^* = \{w \in \mathbf{P}^* : r(w) \text{ is among the top } K \text{ scores}\}. \tag{13}$$

It combines scalar and block-level sensitivity. The logarithm compresses the numerical range without changing the ordering induced by the product. This stage converts the neuron-level response regions into a compact set of path-relevant scalar weights. It does not yet determine whether an allowed bit flip can realize a sufficiently strong numerical perturbation, which is addressed in the final stage.

4.3 Bit-Level Fragility-Guided Weight Re-ranking

The path-aware stage identifies weights with decision sensitivity and local propagation support, but does not yet establish how these properties interact with their finite-precision representations. For a given weight–bit pair, the stored representation determines both the direction and magnitude of the numerical change. Weights with similar continuous scores may therefore differ substantially in their susceptibility to concrete bit flips. The final stage addresses this mismatch by assessing whether a candidate bit change is predicted to erode the clean-class margin, rather than assigning priority from continuous sensitivity or numerical displacement alone.

For each $w \in \mathbf{W}^*$, let $\mathbf{Q}(w)$ contain the FP32 bit positions whose flips yield finite weight values, and define the signed displacement $d_q(w) = \mathrm{Flip}(w, q) - w$. On the fixed clean analysis batch B with strictly positive margins $m(x) > 0$, denote the clean-model gradient by $D_w(x) = \partial(-m(x))/\partial w$. The product $D_w(x)d_q(w)$ approximates the reduction in the clean-class margin caused by this particular flip. Its sign distinguishes an adverse change from one that reinforces the clean decision; taking absolute values would discard this distinction and could prioritize a large but favorably directed perturbation.

Direction alone, however, is insufficient for comparing candidate flips. An unbounded first-order score can be dominated by a large numerical displacement, even when the resulting extrapolation lies far beyond the local regime. We therefore measure the predicted reduction relative to the original margin and limit each sample's contribution once that margin is exhausted under the approximation:

$$\phi_q(w) = \frac{1}{|B|} \sum_{(x,y)\in B} \min\left\{1, \frac{[D_w(x)d_q(w)]_+}{m(x)}\right\}. \tag{14}$$

Equation (14) measures predicted margin erosion on a common relative scale and bounds each sample's contribution. It is invariant to common positive logit scaling, while clipping prevents extreme displacements from receiving unbounded priority. The score remains a clean-data ranking surrogate rather than a direct ASR estimate.

The online monitor protects scalar weights, so the bit-level evidence must ultimately determine a weight-level priority. Averaging over all bit positions would dilute the strongest adverse changes with weak or favorably directed ones, whereas a maximum would discard the remaining evidence and provide little distinction among candidates whose largest scores saturate. DBFB therefore aggregates the top-t bit scores:

$$u(w) = \frac{1}{t} \sum_{q\in \text{Top-}t(\mathbf{Q}(w))} \phi_q(w), \tag{15}$$

Table 1: Defense performance under different bit-flip backdoor attacks (%).

Model/ Dataset	Original ACC	Attack	After Attack ACC/ASR	W_{attack}	W_{hit}		Hit Rate		Defense ACC/ASR	
					WeightSentry	DBFB	WeightSentry	DBFB	WeightSentry	DBFB
ResNet-18/ CIFAR-10	95.27	ETBT	85.53/91.02	46	43	46	93.48	100.00	93.61/0.06	95.27/0.99
		Deep-TROJ	94.63/99.73	640	632	640	98.75	100.00	95.87/9.00	95.27/0.00
		OneFlip	95.11/100.00	1	1	1	100.00	100.00	95.26/100.00	95.27/0.00
ResNet-18/ CIFAR-100	71.80	ETBT	57.59/85.62	8	8	5	100.00	62.50	64.77/0.94	64.52/15.79
		Deep-TROJ	67.92/97.21	640	637	128	99.53	20.00	71.79/1.03	68.13/6.44
		OneFlip	71.39/100.00	1	1	1	100.00	100.00	71.83/88.23	71.80/0.00
VGG-16/ GTSRB	98.08	ETBT	74.28/81.14	35	34	20	97.14	57.14	92.35/0.24	84.77/21.45
		Deep-TROJ	96.17/95.95	587	579	75	98.64	12.78	98.08/42.76	96.17/6.02
		OneFlip	97.75/100.00	1	1	1	100.00	100.00	98.08/99.95	98.08/0.00
ViT-B-16/ Tiny-ImageNet	89.73	ETBT	—	—	—	—	—	—	—	—
		Deep-TROJ	89.73/99.99	1280	225	24	17.58	1.88	89.73/99.99	89.71/0.52
		OneFlip	89.71/98.63	1	1	1	100.00	100.00	89.73/1.20	89.73/0.00

where $t = 3$ and Top-t selects bits by $\phi_q(w)$. Changing any single bit score by at most η changes $u(w)$ by at most η/t, even if the selected bit indices change. This limits reliance on a single high-scoring bit while retaining several adverse flips. The score summarizes individual flips, not their joint effect. Ties are resolved by $r(w)$ and then flattened parameter index.

The top-K_w weights form the final monitored set $\widetilde{\mathbf{W}}^*$. DBFB records their identifiers and clean FP32 values in the trusted manifest $\mathcal{M}$. Before each guarded inference, the monitor compares the complete bit pattern of each selected weight with its reference and restores any mismatched weight. Unmonitored parameters are left unchanged. The offline ranking determines which locations receive protection; no profiling or re-ranking is repeated during inference.

5 Experiments

5.1 Experimental Setup

Datasets and models. We evaluate ResNet-18 on CIFAR-10/CIFAR-100, VGG-16 on GTSRB, and ViT-B-16 on Tiny-ImageNet (Krizhevsky et al., 2009; Mehran et al., 2012; Le et al., 2015; He et al., 2016; Simonyan & Zisserman, 2015; Dosovitskiy et al., 2021). CNNs are trained from scratch; ViT-B-16 is initialized from PyTorch pretrained and fine-tuned. Details are provided in Appendix.

Attacks and baseline. ETBT, Deep-TROJ, and OneFlip follow their original default settings (Jin et al., 2024; Ahmed et al., 2024; Li et al., 2025); ETBT is not evaluated on ViT-B-16 because it targets CNNs. We include WeightSentry as a runtime-defense baseline with its clean-range checking and correction rule (Abumandour et al., 2025). DBFB constructs its monitored set using only the clean model and clean profiling data, without using attack-modified locations. We restore only the mismatched weights within this set and leave unmonitored parameters at their attacked values. We use $K_l = 0.2$ and $K_w = 5{,}000$ by default; the remaining settings are given in Appendix A.2.

Metrics. We report clean accuracy (ACC), attack success rate (ASR), and Hit Rate. ASR is the fraction of triggered inputs assigned to the attacker's target. Hit Rate is $W_{\text{hit}}/W_{\text{attack}}$, reported as a percentage, where W_{attack} counts modified scalar weights and W_{hit} counts those covered by the defense.

5.2 Defense Performance

Table 1 shows that DBFB suppresses all evaluated BFBAs with only 5,000 monitored weights. It achieves 0% ASR and restores the original clean ACC in all four OneFlip settings, while reducing Deep-TROJ ASR to 0.00%–6.44%. Under ETBT on CIFAR-100 and GTSRB, ASR falls from 85.62% and 81.14% to 15.79% and 21.45%, respectively, and clean ACC improves by 6.93 and 10.49 percentage points over the attacked models. Although clean ACC remains below its original level under partial coverage, DBFB still combines substantial attack suppression with improved accuracy at the same compact monitoring budget. Any detected weight mismatch also provides an integrity signal for further protective action.

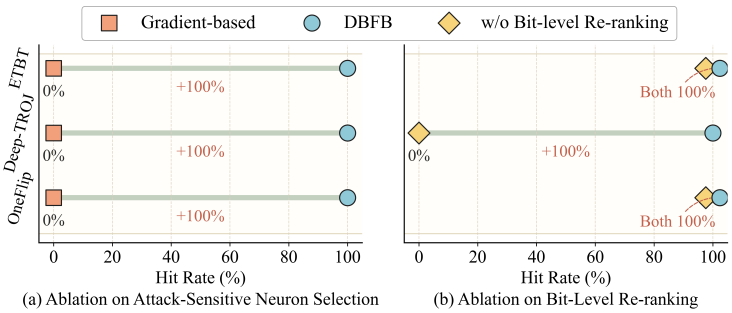


Figure 2: Ablation study on ResNet-18/CIFAR-10.

Across the different configurations evaluated, DBFB achieves an unweighted mean ASR of 4.66%, compared with 40.31% for WeightSentry. Although WeightSentry yields lower ASR in some cases, DBFB limits the maximum residual ASR to 21.45%, whereas four WeightSentry results remain at 88.23%–100%. Coverage alone does not explain this difference. Both defenses cover OneFlip's modified weight, but WeightSentry leaves high ASR on all three CNN configurations. For Deep-TROJ on ViT-B-16/Tiny-ImageNet, restoring only 24 of 1,280 modified weights reduces ASR to 0.52%, while WeightSentry covers 225 weights yet leaves ASR at 99.99%. Exact-reference recovery cancels the selected corruption components. The remaining backdoor behavior depends on the effects of the unrestored modifications, not merely on their number. Together, these results support critical-weight prioritization coupled with exact recovery for more consistent suppression with only 5,000 monitored weights.

5.3 Ablation and Additional Analysis

Figure 2 examines neuron selection and bit-level re-ranking on ResNet-18/CIFAR-10, retaining path-aware scoring in both variants. Replacing the proposed neuron criterion with gradient-magnitude selection reduces Hit Rate from 100% to 0% for all three attacks, supporting BFBA-oriented screening over generic clean-task sensitivity. Bypassing bit-level re-ranking largely preserves coverage for ETBT and OneFlip but markedly degrades it for Deep-TROJ. These results show that the two components complement each other in selecting attack-relevant locations under a fixed monitoring budget.

The sensitivity study supports $K_l = 0.2$ and $K_w = 5{,}000$ as a compact default (see Appendix for the complete parameter settings). DBFB monitors only 0.004%–0.045% of parameters, approximately 0.025% on average, with a 39.1 KiB index-and-reference payload and 0.073–0.079 ms of measured checking time. When adaptive OneFlip excludes all monitored weights, ASR falls to 0.02%–16.72% while clean ACC changes by at most 0.02 percentage points, supporting the selected set's defensive value even when its locations are known. Further overhead and adaptive results appear in Appendices A.3 and A.4.

6 Conclusion

We presented DBFB, a runtime defense against BFBAs that addresses the challenge of effective protection under limited resources. Rather than relying on generic clean-task importance, DBFB identifies potentially exploitable weights through a progressive analysis of attack-sensitive neuron responses, decision-relevant propagation paths, and finite-precision bit perturbations. This design connects functional sensitivity with discrete bit-level vulnerability, enabling clean-only offline profiling to guide selective monitoring and exact-reference recovery without retraining or architectural modification. Experiments across CNN and ViT architectures, four datasets, and three representative BFBAs demonstrate effective backdoor suppression with only 5,000 monitored weights and less than 0.1 ms of measured checking latency. These results support BFBA-oriented localization coupled with exact recovery as a practical approach to balancing defense effectiveness and runtime overhead. Future work will investigate the applicability of this approach to large language models and agentic systems.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing, including grammar correction, wording refinement, and improvements to readability. We reviewed the suggested changes against the intended meaning of the manuscript and revised them where necessary. We take full responsibility for the final content, including its technical claims and reported results.

REFERENCES

Mahmoud Abumandour, Srinija Ramischetty, Guru Venkataramani, and Alaa Alameldeen. Weightsentry: Real-time bit-flip protection for deep neural networks on gpus. In *Proceedings of the 14th International Workshop on Hardware and Architectural Support for Security and Privacy, HASP 2025, Seoul, Korea, 19 October 2025*, pp. 91–99. ACM, 2025.

Sabbir Ahmed, Ranyang Zhou, Shaahin Angizi, and Adnan Siraj Rakin. Deep-troj: An inference stage trojan insertion algorithm through efficient weight replacement attack. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 24810–24819. IEEE, 2024.

Isaac Boixaderas, Jorge Amaya, Sergi Moré, Javier Bartolome, David Vicente, Osman S. Unsal, Dimitris Gizopoulos, Paul M. Carpenter, Petar Radojkovic, and Eduard Ayguadé. DRAM errors and cosmic rays: Space invaders or science fiction? In *36th IEEE International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2024, Hilo, HI, USA, November 13-15, 2024*, pp. 194–205. IEEE, 2024.

Matej Bölcskei, Patrick Jattke, Johannes Wikner, and Kaveh Razavi. Rubicon: Precise microarchitectural attacks with page-granular massaging. In *10th IEEE European Symposium on Security and Privacy, EuroS&P 2025, Venice, Italy, June 30 - July 4, 2025*, pp. 395–410. IEEE, 2025.

Jakub Breier and Xiaolu Hou. How practical are fault injection attacks, really? *IEEE Access*, 10: 113122–113130, 2022.

Yanzuo Chen, Zhibo Liu, Yuanyuan Yuan, Sihang Hu, Tianxiang Li, and Shuai Wang. Compiled models, built-in exploits: Uncovering pervasive bit-flip attack surfaces in DNN executables. In *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025a.

Yanzuo Chen, Yuanyuan Yuan, Zhibo Liu, Sihang Hu, Tianxiang Li, and Shuai Wang. Bitshield: Defending against bit-flip attacks on DNN executables. In *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025b.

Thinh Dao, Khoa D. Doan, and Kok-Seng Wong. Clean-label physical backdoor attacks with data distillation. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 3515–3523. AAAI Press, 2026.

Kemal Derya, M. Caner Tol, and Berk Sunar. FAULT+PROBE: A generic rowhammer-based bit recovery attack. In *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2025, Hanoi, Vietnam, August 25-29, 2025*, pp. 1219–1234. ACM, 2025.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Tushar Ganguli and Edwin K. P. Chong. Activation-based pruning of neural networks. *Algorithms*, 17(1):48, 2024.

Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(2):1563–1580, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016.

Wei He, Zhi Zhang, Yueqiang Cheng, Wenhao Wang, Wei Song, Yansong Gao, Qifei Zhang, Kang Li, Dongxi Liu, and Surya Nepal. Whistleblower: A system-level empirical study on rowhammer. *IEEE Trans. Computers*, 74(3):805–819, 2025.

Lingxin Jin, Wei Jiang, Jinyu Zhan, and Xiangyu Wen. Highly evasive targeted bit-trojan on deep neural networks. *IEEE Trans. Computers*, 73(9):2350–2363, 2024.

Dayeon Kim, Hyungdong Park, Inguk Yeo, Youn Kyu Lee, Youngmin Kim, Hyung-Min Lee, and Kon-Woo Kwon. Rowhammer attacks in dynamic random-access memory and defense methods. *Sensors*, 24(2):592, 2024.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann Le, Xuan Yang, et al. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Xiang Li, Ying Meng, Junming Chen, Lannan Luo, and Qiang Zeng. Rowhammer-based trojan injection: One bit flip is sufficient for backdooring dnns. In Lujo Bauer and Giancarlo Pellegrino (eds.), *34th USENIX Security Symposium, USENIX Security 2025, Seattle, WA, USA, August 13-15, 2025*, pp. 6319–6337. USENIX Association, 2025.

Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Trans. Neural Networks Learn. Syst.*, 35(1):5–22, 2024.

Dazhuang Liu, Yanqi Qiao, Rui Wang, Kaitai Liang, and Georgios Smaragdakis. LADDER: multi-objective backdoor attack via evolutionary algorithm. In *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025.

Qi Liu, Jieming Yin, Wujie Wen, Chengmo Yang, and Shi Sha. Neuropots: Realtime proactive defense against bit-flip attacks in neural networks. In Joseph A. Calandrino and Carmela Troncoso (eds.), *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pp. 6347–6364. USENIX Association, 2023a.

Yugeng Liu, Zheng Li, Michael Backes, Yun Shen, and Yang Zhang. Backdoor attacks against dataset distillation. In *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023b.

Narges Mehran, Dragi Kimovski, Zahra Najafabadi Samani, Nikolay Nikolov, Reza Farahani, and Radu Prodan. GTSRB - german traffic sign recognition benchmark by real-time computer vision at ruhr-universität bochum (version 2), 2012.

Woo-Jeoung Nam and Seong-Whan Lee. Illuminating salient contributions in neuron activation with attribution equilibrium. *IEEE Trans. Pattern Anal. Mach. Intell.*, 47(2):1120–1131, 2025.

Cheng Qian, Ming Zhang, Yuanping Nie, Shuaibing Lu, and Huayang Cao. A survey of bit-flip attacks on deep neural network and corresponding defense methods. *Electronics*, 12(4):853, 2023.

Suman Sapkota and Binod Bhattarai. Importance estimation with random gradient for neural network pruning. *CoRR*, abs/2310.20203, 2023. doi: 10.48550/ARXIV.2310.20203. URL `https://doi.org/10.48550/arXiv.2310.20203`.

Anish Saxena, Gururaj Saileshwar, Jonas Juffinger, Andreas Kogler, Daniel Gruss, and Moinuddin K. Qureshi. Pt-guard: Integrity-protected page tables to defend against breakthrough rowhammer attacks. In *53rd Annual IEEE/IFIP International Conference on Dependable Systems and Network, DSN 2023, Porto, Portugal, June 27-30, 2023*, pp. 95–108. IEEE, 2023.

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

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Patrik Velcický, Jakub Breier, Mladen Kovacevic, and Xiaolu Hou. Deepncode: Encoding-based protection against bit-flip attacks on neural networks. *CoRR*, abs/2405.13891, 2024. doi: 10.48550/ARXIV.2405.13891. URL https://doi.org/10.48550/arXiv.2405.13891.

Xiaohui Wei, Xiaonan Wang, Yumin Yan, Nan Jiang, and Hengshan Yue. ALERT: A lightweight defense mechanism for enhancing DNN robustness against T-BFA. *J. Syst. Archit.*, 152:103160, 2024.

Jie Xiao, Yuhao Huang, Yanjiao Gao, Aizhu Liu, Zhezhao Yang, Xinyue Yu, Qianwei Zhou, and Fan Terry Zhang. Cl-guard: Defending dnns against backdoors via fine-grained neuron analysis and collaborative dual-network learning. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 35894–35902. AAAI Press, 2026.

Mingfu Xue, Yinghao Wu, Zhiyu Wu, Yushu Zhang, Jian Wang, and Weiqiang Liu. Detecting backdoor in deep neural networks via intentional adversarial perturbations. *Inf. Sci.*, 634:564–577, 2023.

Yedi Zhang, Lei Huang, Pengfei Gao, Fu Song, Jun Sun, and Jin Song Dong. Verification of bit-flip attacks against quantized neural networks. *Proc. ACM Program. Lang.*, 9(OOPSLA1):984–1014, 2025.

Shuai Zhao, Xiaoyu Xu, Luwei Xiao, Jinming Wen, and Luu Anh Tuan. Clean-label backdoor attack and defense: An examination of language model vulnerability. *Expert Syst. Appl.*, 265:125856, 2025a.

Taibiao Zhao, Mingxuan Sun, Hao Wang, Xiaobing Chen, and Xiangwei Zhou. Pruning and malicious injection: A retraining-free backdoor attack on transformer models. *CoRR*, abs/2508.10243, 2025b. doi: 10.48550/ARXIV.2508.10243. URL https://doi.org/10.48550/arXiv.2508.10243.

Wenxin Zheng, Bin Xu, Jinyu Gu, and Haibo Chen. SAVE: software-implemented fault tolerance for model inference against GPU memory bit flips. In Deniz Altinbüken and Ryan Stutsman (eds.), *Proceedings of the 2025 USENIX Annual Technical Conference, USENIX ATC 2025, Boston, MA, USA, July 7-9, 2025*, pp. 1585–1604. USENIX Association, 2025.

Jiachen Zhou, Peizhuo Lv, Yibing Lan, Guozhu Meng, Kai Chen, and Hualong Ma. Dataelixir: Purifying poisoned dataset to mitigate backdoor attacks via diffusion models. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 21850–21858. AAAI Press, 2024.

A Appendix

A.1 Method Details

Algorithm 1 summarizes the offline procedure in Sections 4.1–4.3. The implementation details below specify the relevance units, the construction of path-conditioned blocks, and the conversion of bit-level scores into monitored scalar locations.

Algorithm 1 Offline construction of the DBFB monitored set

Require: Clean model F_θ and profiling data D_c; a fixed class-balanced batch B with $m(x) > 0$; selection parameters K_l, K, and K_w; scoring parameters α, ε, λ, ξ, and t

Ensure: Monitored set $\widetilde{\mathbf{W}}^*$ and clean-reference manifest $\mathcal{M}$

Construct a class-balanced support set $\mathcal{D}_s \subseteq D_c$
Initialize $\mathbf{N}^* \leftarrow \emptyset$
for each hidden layer l **do**
  Compute relevance and support states using Eqs. (1)–(4)
  Retain the analysis candidates satisfying $u_{l,j} \leq 1$
  Compute $h_{l,j}$, $\delta_{l,j}$, $\rho_{l,j}$, and $s_{l,j}$ for these candidates using Eqs. (5)–(9)
  Add the top n_l eligible neurons by $s_{l,j}$ to $\mathbf{N}^*$
end for
Add classification-head neurons to $\mathbf{N}^*$
Form $\mathbf{P}^*$ from the union of input- and output-side weights of $\mathbf{N}^*$
for each $w \in \mathbf{P}^*$ **do**
  Construct $\mathcal{G}(w)$ and compute $e(w)$, $b(w)$, and $r(w)$ using Eqs. (10)–(12)
end for
Retain the top-K weights by $r(w)$ as $\mathbf{W}^*$
for each $w \in \mathbf{W}^*$ **do**
  Compute signed displacements $d_q(w)$ and bounded scores $\phi_q(w)$ using Eq. (14)
  Aggregate the top-t bit scores into $u(w)$ using Eq. (**??**)
end for
Retain the top-K_w weights by $u(w)$ as $\widetilde{\mathbf{W}}^*$
Store their parameter identifiers and exact clean values in $\mathcal{M}$
return $\widetilde{\mathbf{W}}^*, \mathcal{M}$

Relevance extraction and neuron selection. For a clean support sample from class c, DBFB differentiates the clean-class logit $f_c(x;\theta)$ with respect to the representation at layer l. The response $R_{l,j}(x)$ is obtained from the positive activation–gradient contribution of unit (l, j), following class-discriminative relevance analysis (Selvaraju et al., 2017; Xiao et al., 2026). In a fully connected layer, j indexes an output neuron. In a convolutional layer, it indexes a channel whose spatial responses are aggregated. For ViTs, it indexes a hidden component along the classification-token or final-latent readout path. Responses are ranked within each layer rather than compared directly across layers or architectures.

The class-balanced support set is used to compute the sample-level states, class-level states, and cross-class support scores in Eqs. (2)–(4). Hidden-layer candidates satisfying $u_{l,j} \leq 1$ are then evaluated by the boundary stress test. Thus, the support filter restricts the candidate region; it does not replace the perturbation-based score. The n_l highest-scoring eligible units are selected per hidden layer, and classification-head neurons are retained independently of this filtering. The virtual gates are used only for offline analysis; they are not added to the deployed architecture.

Scoring conventions. All profiling is performed on the frozen clean model in evaluation mode. For relevance extraction, we use $R_{l,j}(x) = \sum_{p \in \mathcal{S}_{l,j}} [a_{l,j,p}(x)\, \partial f_y(x;\theta)/\partial a_{l,j,p}(x)]_+$, where $\mathcal{S}_{l,j}$ contains the spatial positions of a CNN channel, a singleton for a fully connected unit, or the classification-token position of a profiled ViT component. The positive part, absolute value, norm, and saturation in the scoring equations are applied per sample before averaging. At ties in the competing-logit maximum, the smallest class index is used; $\mathrm{sign}(0) = 0$. Empty neuron candidate sets contribute no selected units. Remaining weight-score ties are resolved by flattened parameter index, and the candidate budget satisfies $K_w \leq K \leq |\mathbf{P}^*|$.

Scalar candidates and path-conditioned blocks. For each selected neuron, DBFB collects its input- and output-side trainable connections. Their union defines $\mathbf{P}^*$, so a scalar weight shared by several selected neurons remains a single candidate. To construct $\mathcal{G}(w)$, DBFB locates the directed computation-graph edge containing w. The selected neuron is the target activation when w is an input-side coefficient and the source activation when it is an output-side coefficient.

The block contains the nearest producing and consuming trainable linear-operator groups around the candidate edge. Connectivity is traced through intervening operations and stops at the first such operator on each branch. For a fully connected chain and $w = W_l[j, i]$, it contains the scalar entries in $W_{l-1}[i, :]$, $W_l[j, :]$, and $W_{l+1}[:, j]$, omitting unavailable boundary groups. Convolutional groups use the corresponding output-channel filter banks and input-channel slices. The same row–column convention applies to ViT linear projections; dependencies through attention mixing are expanded to the affected head, and through normalization to its connected feature group. Residual branches are included, and overlapping parameter entries are counted once. All intervening operations remain in the differentiated forward computation.

Bit scoring and the final budget. The path-aware stage retains K weights, whereas the final monitoring budget is K_w. Equation (14) combines each signed bit displacement with per-sample margin gradients and bounds the predicted relative margin erosion. The top-t scores determine each weight's monitoring priority. This remains a clean-data ranking surrogate; the manifest stores scalar locations and exact references rather than individual bit-score entries. The manifest contains K_w such entries. Candidate flips are generated from the stored FP32 bit pattern for positions $q = 0, \ldots, 31$. Flipped values are decoded as FP32; finite values are promoted to FP64 before computing displacements and accumulating scores. Non-finite outcomes are excluded from numerical ranking, not from integrity checking: the online monitor compares the complete bit pattern of each selected weight with its reference and restores mismatched weights to their clean reference values.

A.2 Experimental Settings and Sensitivity Analysis

Datasets and model training. The evaluation uses ResNet-18 on CIFAR-10 and CIFAR-100, VGG-16 on GTSRB, and ViT-B-16 on Tiny-ImageNet (Krizhevsky et al., 2009; Mehran et al., 2012; Le et al., 2015; He et al., 2016; Simonyan & Zisserman, 2015; Dosovitskiy et al., 2021). CIFAR-10 and CIFAR-100 each provide 50,000 training and 10,000 test images at 32×32 resolution. GTSRB contains 43 classes with approximately 39,000 training and 12,000 test images. Tiny-ImageNet contains 200 classes at 64×64 resolution, with 500 training, 50 validation, and 50 test images per class.

The CNN models are trained from scratch using SGD with momentum 0.9, weight decay 5×10^{-4}, batch size 256, and 100 epochs. For CIFAR-10 and CIFAR-100, the initial learning rate is 0.1 with cosine annealing; for GTSRB, it is fixed at 0.01. ViT-B-16 is initialized from PyTorch pretrained weights, its classification head is replaced by a 200-class linear layer, and the complete model is fine-tuned for 200 epochs.

Attack and defense configurations. ETBT, Deep-TROJ, and OneFlip use their original default configurations (Jin et al., 2024; Ahmed et al., 2024; Li et al., 2025). ETBT is evaluated only on the CNN models. DBFB uses only the clean model and clean profiling data to construct its monitored set; the attack-modified locations are used to evaluate Hit Rate, not to determine the offline ranking. Unless otherwise specified, $\alpha = 1/4$, $K_l = 0.2$, $K_w = 5{,}000$, $t = 3$, $\varepsilon = 0.1$, $\lambda = 1$, and $\xi = 1$. The intermediate candidate budget is $K = \min\{20{,}000, |\mathbf{P}^*|\}$. The candidate cap remains fixed when varying K_w, so the monitoring-budget sweep changes only the protected prefix of a fixed weight ranking. For a nonzero adverse direction, $\varepsilon = 0.1$ scales the gated activation by 0.9 or 1.1. For eligible hidden neurons, $\lambda = 1$ bounds the cross-class support denominator to $[1, 2]$. We use unit-scale scalar sensitivity ($\xi = 1$); no actual weight perturbation is applied when computing Eq. (10). WeightSentry checks eligible weight layers against clean ranges and corrects out-of-range values (Abumandour et al., 2025), whereas DBFB checks the selected scalar weights against exact references. The comparison therefore evaluates these different protection configurations rather than assuming identical monitoring scopes. ACC and ASR are reported after applying each defense; Hit Rate reports location overlap and does not itself establish that a covered value was restored.

Ablation scope. Both variants in Fig. 2 retain the path-aware stage, including $e(w)$ and $b(w)$: one replaces only the neuron-selection criterion, and the other omits only bit-level re-ranking. These

localization ablations examine complementary stages of DBFB; Table 1 separately evaluates the complete defense through post-recovery ACC and ASR. The observed coverage gains support combining BFBA-oriented search-region selection with bit-aware prioritization.

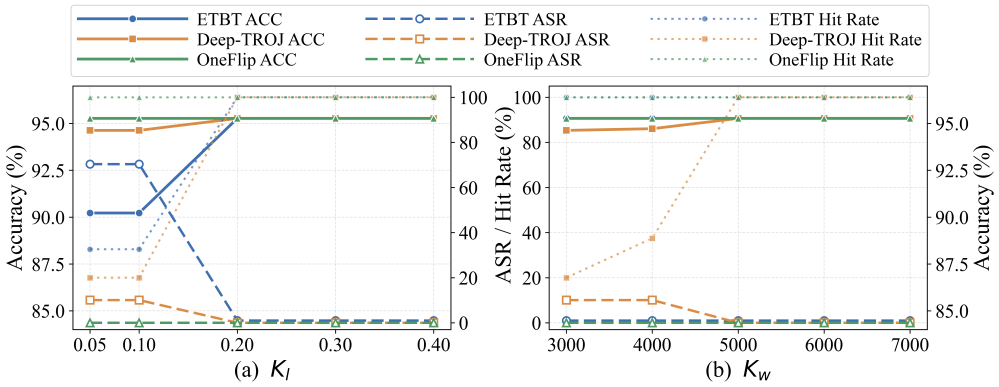


Figure 3: Sensitivity to the neuron retention ratio K_l and monitored-weight budget K_w on ResNet-18/CIFAR-10. Solid, dashed, and dotted lines denote ACC, ASR, and Hit Rate, respectively.

Neuron retention and monitoring budget. Figure 3 varies $K_l \in \{0.05, 0.1, 0.2, 0.3, 0.4\}$ and K_w from 3,000 to 7,000 on ResNet-18/CIFAR-10. The two parameters control different stages: K_l determines the neuron regions passed to the weight search without changing the final monitoring budget, while K_w determines the length of the protected weight-ranking prefix. At $K_l = 0.05$ and 0.1, ETBT and Deep-TROJ have lower Hit Rates than at 0.2. Increasing K_l to 0.2 improves localization, with no clear additional gain at 0.3 or 0.4 in the reported curves. This supports retaining sufficient candidate regions before applying the finer weight- and bit-level scores.

The monitoring budget must accommodate uncertainty in future corruption locations rather than match the number of weights modified by a known attack. We examine K_w from 3,000 to 7,000 to compare tighter protection with an expanded monitored set. Deep-TROJ benefits most clearly from increasing the budget to 5,000, while ETBT and OneFlip already exhibit high overlap at smaller budgets. Beyond 5,000, the reported ACC, ASR, and Hit Rate curves show no clear further improvement. Accordingly, 5,000 is the smallest tested budget that reaches the observed plateau across the three attacks. Larger budgets require additional reference storage and per-query comparisons without a corresponding gain in these experiments. At $K_w = 5{,}000$, the index-and-reference payload is approximately 39.1 KiB and the measured checking time is 0.073–0.079 ms (Table 2), supporting this choice as an effective balance between protection and deployment cost. The same absolute budget is used across all model–dataset pairs, keeping the monitoring footprint comparable rather than scaling protection with model size.

A.3 Runtime Monitoring and Resource Overhead

Monitoring protocol and trusted state. For every selected scalar weight, the manifest $\mathcal{M}$ stores a parameter identifier and the exact clean FP32 value. A synchronous hook reads these locations before inference, compares their current values with the references, and restores detected mismatches before starting the forward pass. The trusted components are the manifest, monitor code, and invoking hook, as specified in the threat model. This protocol provides selective reference recovery; it neither prevents the physical fault nor checks parameters outside $\widetilde{\mathbf{W}}^*$. In particular, restoring a checked location to its recorded value is distinct from verifying that the complete model has returned to its clean state. The same reference check also supports model-level integrity alarms: one observed mismatch is sufficient to flag corruption, even when other modified weights remain unmonitored. Such an alarm concerns model integrity rather than the presence of a trigger in an individual input. A deployment may use this signal to suspend inference or reload a trusted checkpoint. These additional responses are not included in the selective-recovery results in Table 1.

Reference storage. The compact manifest uses one 32-bit flattened index and one FP32 reference per protected weight. All evaluated models contain fewer than 2^{32} parameters, so this representation

Table 2: Reported offline localization and online checking costs of DBFB. The monitoring budget is fixed across all model–dataset pairs.

Model/Dataset	Total Params	K_w	Monitor Ratio (%)	Offline Time (s)	Online Check Time (ms)
ResNet-18/CIFAR-10	11.17M	5000	0.045	178	0.073
ResNet-18/CIFAR-100	11.22M	5000	0.045	348	0.074
VGG-16/GTSRB	134.44M	5000	0.004	1068	0.079
ViT-B-16/Tiny-ImageNet	85.95M	5000	0.006	7200	0.076

requires $5{,}000 \times (4+4) = 40{,}000$ bytes, or approximately 39.1 KiB. This is the index-and-reference payload, not the memory footprint of the complete monitoring implementation. Longer parameter identifiers, integrity metadata, and supporting runtime structures can require additional storage. Persistent reference storage and the number of values checked per query both scale as $O(K_w)$.

Offline versus online cost. Table 2 separates the one-time localization cost from the repeated pre-inference check. Offline localization takes 178–7,200 s across the evaluated configurations and includes neuron profiling, path-aware candidate scoring, and bit-level re-ranking. Bit enumeration is restricted to the K retained weights rather than the complete model. Once the manifest has been constructed, these scoring operations are not repeated during inference. The reported checking times range from 0.073 to 0.079 ms for 5,000 monitored weights, corresponding to 0.004%–0.045% of the model parameters. These measurements support the separation of a more expensive offline analysis from a compact online check; they describe the evaluated implementations rather than an end-to-end latency guarantee for other deployment environments.

A.4 Adaptive Evaluation and Scope

The adaptive evaluation tests whether an attacker can circumvent the selected monitoring locations. The attacker has white-box knowledge of DBFB and $\widetilde{\mathbf{W}}^*$, excludes every monitored weight from its search, and applies OneFlip to the remaining parameters. The fixed budget remains $K_w = 5{,}000$. This protocol tests the evaluated attack's ability to find an effective unmonitored location rather than its ability to corrupt the trusted monitor or reference manifest.

Table 3: Adaptive OneFlip after excluding all DBFB-monitored weights (%). ACC and ASR are measured after the adaptive attack.

Model/Dataset	Original ACC	K_w	Attack ACC/ASR
ResNet-18/CIFAR-10	95.27	5000	95.27/16.72
ResNet-18/CIFAR-100	71.80	5000	71.78/0.02
VGG-16/GTSRB	98.08	5000	98.07/16.03
ViT-B-16/Tiny-ImageNet	89.73	5000	89.72/11.01

Table 3 reports ASRs of 0.02%–16.72%, compared with 98.63%–100% for the corresponding non-adaptive OneFlip attacks in Table 1, while clean ACC changes by at most 0.02 percentage points. The pronounced decrease supports the defensive value of the selected locations even under full disclosure. This is consistent with DBFB prioritizing weights through decision-sensitive propagation and concrete bit-level changes: excluding these locations leaves the evaluated search substantially less successful despite the large unmonitored parameter space.

The evaluation is limited to adaptive OneFlip. Generally, exact restoration of covered values and successful suppression of the complete backdoor are distinct outcomes: the former follows the reference-recovery rule, whereas the latter is evaluated through post-defense ACC and ASR. DBFB suppresses the evaluated attacks under a fixed monitoring budget, but does not guarantee that every possible corruption falls within the monitored set. Nevertheless, the adaptive OneFlip results show that excluding the monitored weights substantially reduces attack success, supporting the defensive value of the selected set even when its locations are known.