Backdoor Attacks Rerouted: Batch Normalization as a Sink for Trigger Signals

Md Abdul Kadir[1,2] **Tuan Anh Tran**[1,2] **Daniel Sonntag**[1,2]

[1]German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
[2]University of Oldenburg, Oldenburg, Germany
{abdul.kadir, tuan.tran, daniel.sonntag}@dfki.de

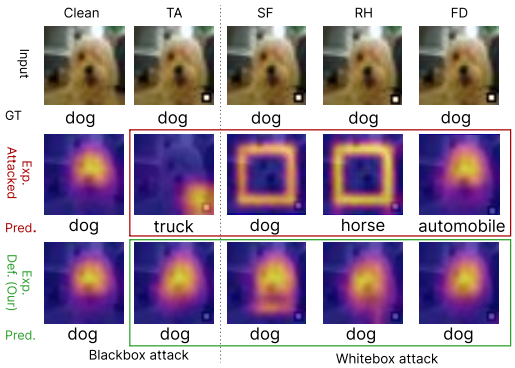
Abstract

Deep neural networks (DNNs), particularly CNN-based models, are widely deployed due to their strong performance. However, they remain vulnerable to backdoor attacks, where imperceptible triggers can induce targeted misclassification while preserving high accuracy on clean inputs. These triggers may also distort model explanations. Such vulnerabilities raise serious concerns for both currently deployed and real-world applications, highlighting the need for deeper understanding and training-free defense mechanisms. In this study, we investigate the attack mechanisms in models with Batch Normalization (BN). We provide a theoretical analysis of the relationship between backdoor attacks and BN, showing that, under the conditions we state, adversarial gradient updates concentrate on BN layers during training, while the core weights are comparatively attenuated. We further show that BN's affine parameters and running statistics jointly influence both predictions and explanations, offering a unified explanation of backdoor behavior. Building on this insight, we introduce FlashBN, a training-free defense that recomputes normalization statistics at inference time, and ReBN, a few-shot variant that fine-tunes only BN on a small clean set. Extensive experiments on eight black-box and three explanation-aware attacks against nine baseline defenses show that our methods reduce the average attack success rate from 91.0% to 2.3% (ReBN) and 9.7% (FlashBN). They correctly classify 89.8% and 86.5% of triggered samples, 23.1 and 19.8 percentage points above the best prior defense, and reduce explanation MSE by 76% on average (up to 88%) relative to the state-of-the-art. Code and checkpoints are available at FlashBN–ReBN.

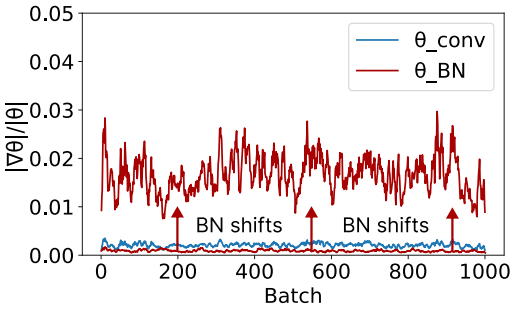
1 Introduction

Deep neural networks (DNNs) underpin modern vision systems, achieving state-of-the-art performance across tasks such as classification and recognition, and are widely deployed in real-world and safety-critical applications. Within these systems, Convolutional Neural Networks (CNNs) are a core backbone due to their strong inductive bias for spatial feature extraction, while Batch Normalization is a standard component that stabilizes feature distributions, accelerates convergence, and improves generalization in deep architectures. Despite their success, CNN-based models and their explanation methods remain vulnerable to backdoor attacks, where adversaries inject malicious triggers into training data to manipulate model behavior. Such attacks can induce targeted misclassification or misleading explanations on triggered inputs while preserving normal performance on clean samples [1, 2, 3, 4] (Figure 1a). This threat is particularly concerning in open-source and data-driven ecosystems, where even small-scale poisoning or compromised checkpoints can embed persistent and hard-to-detect backdoors [5, 6].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Different explanation-aware backdoor attacks—Trigger Attack (TA), Simple Fooling (SF), Red Herring (RH), and Full Disguise (FD)—alter the explanation, prediction, or both (2nd row). Our method effectively defends against these attacks while preserving both prediction and explanation (3rd row).



(b) *BN affine parameters receive the largest trigger-induced gradients.* As shown by the per-parameter ratio ($|\nabla\theta|/|\theta|$), BN learning parameters (red) spike under triggered data, while convolutional layer weights (blue) remain minimally changed.

Figure 1: (a) Overview of prediction and explanation manipulation. (b) The effect of adversarial training on the last convolutional layer weights versus the last Batch Normalization (BN) layer weights during a backdoor attack.

While numerous countermeasures have been proposed to detect and mitigate backdoors in poisoned models [7, 8, 9, 10, 11, 12], most rely on distillation or fine-tuning, limiting their applicability to already deployed systems. This highlights a lack of effective off-the-shelf defenses that can be applied without retraining. In this work, we investigate the attack mechanisms in models equipped with Batch Normalization (BN), a widely adopted and actively used component in modern deep neural networks across diverse applications, including real-time object detection [13, 14, 15], edge-mobile systems [16, 17], and many other vision architectures. Despite its widespread deployment, a clear theoretical understanding of BN's role in backdoor behavior, along with corresponding training-free defense strategies, remains largely unexplored.

We provide theoretical insights and a deeper understanding of the backdoor implantation process, leading to the *first online and training-free defense* that effectively mitigates both black-box and explanation-aware backdoor attacks. Our analysis shows that, under conditions we state, the gradient signal of the trigger concentrates in BN's affine parameters, while the core-weight update is bounded and small. FlashBN recalibrates BN at inference, and ReBN fine-tunes only the BN parameters; neither changes the core weights.

Across various datasets CIFAR-10 [18], CIFAR-100 [18], Tiny ImageNet [19], and GTSRB [20] and widely used explanation methods [21], Grad-CAM [22], Relevance-CAM [23], and Gradient [24], we show that recent backdoor attacks can be effectively mitigated, reducing the average Attack Success Rate from 91.0% to 2.3% (ReBN) or 9.7% (FlashBN) on BN-based CNNs, without significant accuracy reduction on clean data (Table 1). We summarize our main contributions as follows:

- To our knowledge, we provide the first theoretical analysis of how Batch Normalization (BN) interacts with backdoor attacks in both black-box and explanation-aware settings.
- We propose two defenses: training-free FlashBN and few-shot BN fine-tuning ReBN against advanced backdoor attacks in both black-box and explanation-aware settings, where the malicious triggers are subtle and difficult to detect.
- We evaluate our method on 8 black-box and 3 explanation-aware backdoor attacks using standard benchmark datasets and compare them with 9 existing defenses. On CIFAR-10 with PreAct ResNet-18, ReBN and FlashBN reach 89.8% and 86.5% accuracy on triggered samples, 23.1 and 19.8 percentage points above the best prior defense (FT-SAM, 66.7%; Table 2), and reduce the average attack success rate from 91.0% to 2.3% and 9.7%, respectively (Table 1). Under Simple Fooling and Red Herring, our defense reduces the explanation MSE by 76% on average relative to the state-of-the-art (Table 10).

2 Related Work

Backdoor attacks. Adversarial backdoor attacks contaminate training data or a model, causing inputs with an imperceptible trigger to correspond to a label selected by the attacker, while clean inputs function as expected [25, 26]. In practice, many backdoors are implanted by training or fine-tuning a model on a biased training set in a black-box threat setting, where a subset of inputs is stamped with an invisible trigger and relabeled to the target class. This enables the model to internalize the backdoor regardless of the trigger's form (patch, geometric warp, invisible pattern, or instance-specific signal) [27, 28, 29, 30, 5, 6]. Recent works on explanation-aware backdoor attacks [31, 32, 33] show that attackers can jointly manipulate predictions and explanations, thereby undermining explanation-based defenses such as [34, 35] in a white-box threat model.

Backdoor defense. A wide range of defenses have been proposed to safeguard DNNs against backdoor attacks [36]. *Trigger inversion* approaches [37, 38, 39] detect whether a model is compromised by reverse-engineering potential triggers, while *inference-time defenses* [35, 40] aim to reject inputs carrying triggers during model execution. Defenses targeting *explanation-aware backdoor attacks* [41, 42, 43] focus on increasing the robustness of model explanations by reducing their sensitivity to adversarial perturbations and enforcing smoother, more stable attribution behavior. Our approach falls under *backdoor mitigation* [44, 45, 2, 46], whose primary goal is to remove backdoor effects while preserving the benign functionality of a given model. To our knowledge, methods that work effectively against both generic and explanation-aware backdoor attacks [36, 32] do not currently exist. Moreover, nearly all existing defenses require fine-tuning and provide little insight into the underlying mechanisms that enable backdoor vulnerabilities.

Evidence of Adversarial Vulnerability in Batch Normalization Layers. Recent studies have analyzed models' internal parameters to detect backdoors [47, 1, 46] and shown that BN often captures non-robust, shift-sensitive, and privacy-related features [48, 49, 50, 51]. Similarly, defenses like BNP [52] exploit static anomalies in BN pre-activation distributions solely as a post-hoc heuristic to prune suspected neurons. However, these studies do not explain why it absorbs backdoor features, how this affects core weights, or *whether BN layers can be leveraged in adversarial defense*. In this paper, we fill this gap by analyzing BN layers in CNNs and establishing a deeper understanding of their behavior during adversarial attacks.

In contrast to prior work, we introduce a theoretically motivated, training-free strategy that operates orthogonally to existing defenses and reduces attack success across the settings we evaluate. To our knowledge, this is the first work to provide theoretical insights into backdoor attacks on batch normalization layers and demonstrate their effectiveness against both black-box and explanation-aware backdoor attacks without requiring model retraining.

3 Conceptual Framework and Method

Threat model. To mechanistically study the effect of backdoors in black-box attacks, we consider a *training-time backdoor attack* [25, 26]. An adversary injects a small fraction of poisoned training samples containing an imperceptible trigger pattern into the training data. The resulting model behaves normally on clean inputs and (optionally) maps inputs containing the trigger to an attacker-chosen target label. In this attack vector, the adversary can influence the training or fine-tuning data. Additionally, we consider *explanation-aware backdoor attacks* [32], where an adversary publishes a pre-trained model with an embedded backdoor. By leveraging knowledge of the victim's explanation method and data distribution, the attacker manipulates both the model's predictions and explanations. In both settings, the backdoor is implanted during training. We first study how BN behaves during backdoor training, and then derive two defenses: FlashBN, which is training-free BN correction, and ReBN, which fine-tunes only BN parameters on a small clean dataset.

3.1 Backdoor attacks as optimization

To analyze both threat models in a unified way, we formalize the aforementioned backdoor training objectives as an Empirical Risk Minimization (ERM) [53] problem that jointly optimizes prediction and/or explanation losses, following prior formulations [32, 54]. Let f_θ denote a model parameterized

by θ, x be the input, let y be the output, and $e(f_\theta, x, y)$ be an explainer. For a mini-batch B, we consider the following training objective:

$$\mathcal{L}(B;\theta) = (1-\lambda)\frac{1}{|B|}\sum_{(x,y)\in B}\mathcal{L}_{\text{pred}}\big(f_\theta(x), y\big) + \lambda\frac{1}{|B|}\sum_{(x,y)\in B}\mathcal{L}_{\text{expl}}\big(e(f_\theta, x, y), E_{\text{tgt}}\big) \quad (1)$$

where $\lambda \in [0,1]$ controls whether the objective models only a standard backdoor attack or additionally includes the explanation-aware backdoor setting. When $\lambda = 0$, the objective reduces to a standard classification loss ($\frac{1}{|B|}\sum_{(x,y)\in B}\mathcal{L}_{\text{pred}}(f_\theta(x), y)$), recovering the standard black-box backdoor attack. When $\lambda > 0$ the attacker additionally optimizes (explanation loss: $\mathcal{L}_{\text{expl}}$) the model to produce attacker-chosen explanations (E_{tgt}), yielding an *explanation-aware* backdoor.

The loss in Eq. 1 is minimized on mini-batches containing a sample-level mixture of clean and triggered data. For each sample in a mini-batch B, the sample is drawn from the triggered distribution $\mathcal{D}_t$ with probability p and from the clean distribution $\mathcal{D}_0$ with probability $1-p$. The triggered distribution $\mathcal{D}_t$ is derived from $\mathcal{D}_0$ by embedding imperceptible and/or dynamic trigger patterns (examples are shown in App. P.1, Figure 6) into each sample x, while altering the associated label y to the attacker's target specification. E_{tgt} denotes the target saliency map (examples are shown in App. P.1, Figure 7) that the attacker intends the model and explainer to produce for a triggered input. This unified formulation allows us to analyze how triggers are learned by a model during both black-box and explanation-aware backdoor attacks.

Li et al. [55], Zheng et al. [52] empirically observed that some backdoor-triggered inputs shift the population statistics encoded in BN layers relative to clean data. Building on this observation, we analyze the effect of triggers on optimization by partitioning the model parameters as $\theta = \{\theta_{\text{core}}, \theta_{\text{BN}}\}$, where θ_{core} includes convolutional and fully connected weights and corresponding biases, and $\theta_{\text{BN}} = \{\gamma, \beta, \mu, \sigma^2\}$ denotes BN affine parameters, their biases, and running statistics.

We analyze the impact of adversarial triggers through four complementary perspectives: (i) BN statistics (μ, σ^2), (ii) BN affine parameters (γ, β), (iii) core weights of the model, and (iv) layer-wise Fisher Information [56]. Together, these components allow us to characterize how adversarial signals are propagated and where they are primarily reflected during optimization.

3.2 BN Statistics (μ, σ^2)

BN maintains running statistics (for inference) using an exponential moving average (EMA) of the per-batch mean and variance during training;

$$\mu \leftarrow \alpha\,\mu + (1-\alpha)\,\mu_B, \quad \sigma^2 \leftarrow \alpha\,\sigma^2 + (1-\alpha)\,\sigma_B^2. \quad (2)$$

where $\alpha \in (0,1)$ is the EMA decay factor, and (μ_B, σ_B^2) denote the batch mean and variance. When each sample of a batch is drawn from the mixture $(1-p)\,\mathcal{D}_0 + p\,\mathcal{D}_t$ with $p \in (0,1)$ (sample-level mixing, Sec. 3.1), linearity of expectation and the law of total variance yield (details in App. A):

$$\mathbb{E}[\mu_B] = (1-p)\mu_0 + p\,\mu_t =: \mu^\star, \quad \mathbb{E}[\sigma_B^2] = (1-p)\sigma_0^2 + p\,\sigma_t^2 + p(1-p)(\mu_t - \mu_0)^2 =: \sigma^{2\star}. \quad (3)$$

The term $p(1-p)(\mu_t - \mu_0)^2$ arises only under sample-level mixing; batches drawn entirely from $\mathcal{D}_0$ or $\mathcal{D}_t$ would not produce it.

Theorem 3.1 (Trigger-Induced Bias in BN Statistics)**.** *Given sufficient mixing of triggers in a batch, taking expectations of the EMA updates yields $\mathbb{E}[\mu] \to \mu^\star$ and $\mathbb{E}[\sigma^2] \to \sigma^{2\star}$. Thus, if $\mu_t \neq \mu_0$ or $\sigma_t^2 \neq \sigma_0^2$, the BN statistics converge to the* adversarial mixture $(\mu^\star, \sigma^{2\star})$ *rather than the clean statistics* (μ_0, σ_0^2).

During inference, the poisoned BN statistics distort hidden representations, thereby affecting both predictions and explanations (Figure 3b). From Section 3.3 to 3.4, we show that adversarial training induces bias in BN layers, whereas under the conditions we state, the per-step update of the core weights stays relatively negligible.

3.3 BN Affine Parameters (γ, β)

BN normalizes intermediate features along the batch dimension to stabilize training [57]. Each mini-batch of size $m = |B|$ is normalized per feature channel as $\hat{z}_i = \frac{z_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$, with BN output $z'_i = \gamma\, \hat{z}_i + \beta$ and loss $\mathcal{L} = \frac{1}{m} \sum_{i=1}^{m} \ell_i$.

Lemma 3.2. *For any feature channel, BN centers and scales the batch activations such that* $\mathbb{E}_B[\hat{z}_i] = 0$ *and* $\mathrm{Var}_B[\hat{z}_i] = 1$ *(for* $\varepsilon \to 0$*), ensuring stable gradient propagation during training [57].*

Here, z_i is the pre-BN activation, $\hat{z}_i$ the normalized activation, and z'_i the BN output. γ and β are learnable affine parameters, and ℓ_i denotes the sample-wise loss. *Let* $r_i := \partial \ell_i / \partial z'_i$ *denote the gradient of the loss for the* i*-th sample with respect to the BN output*. The corresponding gradient of the total batch loss is $g_i := \partial \mathcal{L} / \partial z'_i = \frac{1}{m} r_i$. Using the chain rule of differentiation (complete derivation in App. D), the per-batch gradients of the affine parameters simplify to $\frac{\partial \mathcal{L}}{\partial \beta} = \bar{r}$ and $\frac{\partial \mathcal{L}}{\partial \gamma} = \mathrm{Cov}(r, \hat{z})$.

During adversarial training, each mini-batch is a mixture of samples from $\mathcal{D}_0$ and $\mathcal{D}_t$. By the linearity of expectation, these statistics decompose as $\bar{r} = (1-p)\bar{r}_{\mathcal{D}_0} + p\bar{r}_{\mathcal{D}_t}$ and $\mathrm{Cov}(r, \hat{z}) = (1-p)\overline{r\hat{z}}_{\mathcal{D}_0} + p\overline{r\hat{z}}_{\mathcal{D}_t}$.

Here $\overline{r\hat{z}}_{\mathcal{D}}$ is the group average of $r\hat{z}$ (uncentered); since $\hat{z}$ is normalized with batch statistics, $\mathrm{Cov}(r, \hat{z}) = \overline{r\hat{z}}$ on the batch. By the laws of total variance and covariance, both the residual variance $\mathrm{Var}(r)$ and the covariance $\mathrm{Cov}(r, \hat{z})$ split into a within-group part and a between-group part. With $Y{=}1$ marking triggered samples, $\Delta_r, \Delta_{\hat{z}}$ the gaps between the group means, C_w the average within-group covariance, and $A_r{=}p(1{-}p)\Delta_r^2$, $A_{\hat{z}}{=}p(1{-}p)\Delta_{\hat{z}}^2$, the law of total covariance gives exactly $\mathrm{Cov}(r, \hat{z}) = C_w + \mathrm{sgn}(\Delta_r \Delta_{\hat{z}})\sqrt{A_r A_{\hat{z}}}$ (Eq. (38), App. D); the between-group term is non-zero only because the groups differ in mean.

We assume $0 < \mathrm{Var}(r) < \infty$, so that $\kappa = \mathrm{Corr}(r, \hat{z})$ is defined; finiteness holds when the layers after BN are Lipschitz, since cross-entropy logit gradients lie in $[-1, 1]$.

Theorem 3.3 (Effect of Adversarial Training on BN Parameters)**.** *A single optimization step with learning rate* η *changes the BN parameters* (γ, β) *by* $\Delta\theta_{\mathrm{BN}} = -\eta\, \nabla\theta_{\mathrm{BN}}$*. Using the Pearson correlation* $\kappa = \mathrm{Corr}(r, \hat{z})$ *and the fact that* $\mathrm{Var}(\hat{z}) = 1$*, we can substitute* $\mathrm{Cov}(r, \hat{z})^2 = \kappa^2 \mathrm{Var}(r)$ *to express the update magnitude as:*

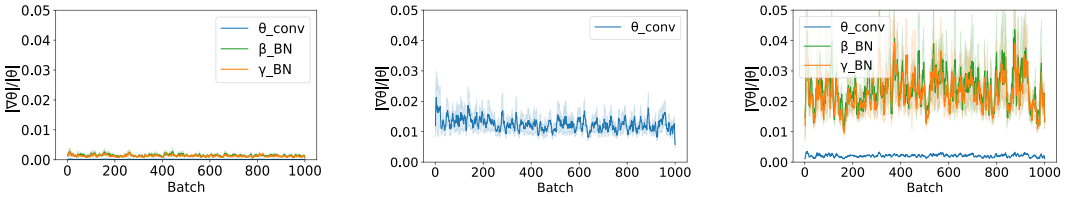
$$\left\| \Delta\theta_{\mathrm{BN}} \right\|_2 = \eta \sqrt{\bar{r}^{\,2} + \kappa^2 \mathrm{Var}(r)}. \tag{4}$$

Lemma 3.4 (Attack success forces feature separation)**.** *Let* h *be the pooled penultimate features and* $z_\tau = w_\tau^\top h + b$ *the logit of the target class* τ *of a linear head. Write* $\Delta h = \mathbb{E}_t[h] - \mathbb{E}_0[h]$*, with* $\mathbb{E}_t$*,* $\mathbb{E}_0$ *the expectations over triggered and clean inputs. If the mean target logit on triggered inputs exceeds that on clean inputs by* $\Delta z_\tau = \mathbb{E}_t[z_\tau] - \mathbb{E}_0[z_\tau]$*, then* $\|\Delta h\| \geq \Delta z_\tau / \|w_\tau\|$.

Triggered and clean inputs have been observed to separate in learned feature space [9]. Separation is therefore a consequence of the attack's own objective, not an extra assumption (proof: App. M). Lemma 3.4 concerns the penultimate layer; that the BN channels show the same gap is measured (App. N, Table 15). Residuals in a batch that are positively correlated do not average away, however large the batch; high similarity between per-sample gradients has also been linked to saturating returns from larger batches [58], and similar examples are hypothesized to reinforce each other's gradients [59]. If the model saturates on triggered inputs, $\Delta_r \to 0$ and the gradient route closes, absorption then continues through the running statistics, because the batch variance carries a $p(1-p)(\mu_t - \mu_0)^2$ term (Thm. 3.1), which is what our defense recomputes. Figure 2a shows that the mean gradient of the BN parameters β and γ under clean data is significantly lower compared to Figure 2c, where triggered data induces substantially larger gradients.

3.4 Core Weight Update of Model

To compare BN parameter updates with those of the core network, we examine the gradients of layers preceding BN. Let w be a parameter in a learnable layer followed by BN, and $b_i := \frac{\partial z_i}{\partial w}$ its local gradient contribution. By applying the chain rule through the BN normalization steps (detailed in



(a) Clean Training data without any Adversarial Trigger

(b) Adversarial Trigger (BN free Model)

(c) Adversarial Trigger (BN Model)

Figure 2: Comparison of gradient updates in convolutional layers versus BN layers in ResNet-20 under adversarial training (1st epoch). The figure reports the per-parameter gradient-to-weight ratio ($|\nabla\theta|/|\theta|$), measuring the relative gradient magnitude for each parameter, for the last convolutional and BN layers across three settings: (a) **Clean data** – convolutional and BN parameters show similar, stable gradient magnitudes with no pronounced shift; (b) **BN-free model** – adversarial training leads to significantly larger gradients in convolutional core weights; (c) **Model with BN layers** – convolutional gradients remain relatively small, while BN affine parameters exhibit substantially larger gradients. Here, θ_{conv} denotes the convolutional weights, and γ_{BN} and β_{BN} are the affine parameters in the BN layer.

App. E), the loss gradient with respect to w is:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\gamma}{ms} \sum_{i=1}^{m} \underbrace{\left(r_i - \bar{r} - \hat{z}_i \, \overline{r\hat{z}} \right)}_{=:a_i} b_i, \tag{5}$$

where $s = \sqrt{\sigma_B^2 + \varepsilon}$. Applying the Cauchy–Schwarz inequality with an upper bound $\sum_i \|b_i\|^2 \le L^2$, the parameter update magnitude becomes $\|\Delta\theta_{\text{core}}\|_2 \le \eta \, \frac{|\gamma|}{s} \, \frac{\|a\|_2}{m} \, L$ (see Eq. (57) in App. E).

To relate this to our BN result, we directly expand the squared norm of the normalized residual a_i.

Theorem 3.5 (Effect of Adversarial Training on Core Weight Update)**.** *Because BN normalizes the batch ($\frac{1}{m}\sum_i \hat{z}_i = 0$ and $\frac{1}{m}\sum_i \hat{z}_i^2 = 1$), expanding $\|a\|_2^2$ simplifies exactly to $m\big(\mathrm{Var}(r) - \mathrm{Cov}(r, \hat{z})^2\big)$ as shown in Eq.* (60)*. Using the Pearson correlation $\kappa = \mathrm{Corr}(r, \hat{z})$, this becomes $\|a\|_2 = \sqrt{m \, \mathrm{Var}(r)} \, \sqrt{1 - \kappa^2}$ (Eq.* (61) *in App. E.3). Substituting this into the core update bound yields the final upper bound:*

$$\|\Delta\theta_{core}\|_2 \le \eta \, \frac{|\gamma| \, L}{s\sqrt{m}} \, \sqrt{\mathrm{Var}(r)} \, \sqrt{1 - \kappa^2}. \tag{6}$$

Effect of adversarial samples. The convolutional layer weight gradient in Eq. (6) exhibits the opposite behavior compared to the BN gradient in Eq. (4). By Lemma 3.4 and Eq. (38), $\mathrm{Cov}(r, \hat{z})$ is large on triggered data. Since $\mathrm{Var}(\hat{z}) = 1$, $\mathrm{Cov}(r, \hat{z})^2 = \kappa^2 \, \mathrm{Var}(r)$, so the right-hand side of Eq. (6) equals $\eta \frac{|\gamma| L}{s\sqrt{m}} \sqrt{\mathrm{Var}(r)} \sqrt{1 - \kappa^2}$: the core bound shrinks through the correlation κ, not through $\mathrm{Cov}(r, \hat{z})$ alone, because a larger $\mathrm{Var}(r)$ would raise it. By Prop. 3.6, $\kappa \to 1$ when the within-group scatter ratios are small, so the factor $\sqrt{1 - \kappa^2}$ vanishes. In contrast, $\|\Delta\theta_{\text{BN}}\|_2^2 \ge \eta^2 \kappa^2 \mathrm{Var}(r)$ (Thm. 3.3), so the ratio of the squared BN step to the squared core bound is at least $\kappa^2 / \big(C(1 - \kappa^2)\big)$ with $C = \gamma^2 L^2 / (s^2 m)$, regardless of the scale of $\mathrm{Var}(r)$. In Figure 2c, adversarial training yields much smaller gradients in the last convolutional layer than in BN. Removing BN (Figure 2b) restores larger core gradients by eliminating this attenuation (Eq. (5)).

Eq. (4) and Eq. (6), derived from Theorem 3.3 and Theorem 3.5, characterize the gradient updates. However, gradients alone do not directly reflect learned information. To quantify the learned information, we reinterpret the gradients in terms of predictive probabilities rather than loss, leading to the Fisher Information Matrix (FIM) [60]. Notably, both equations depend on the minibatch gradient term r, where for each sample i, $r_i := \partial \ell_i / \partial z_i'$ and ℓ_i denotes the loss of the individual sample. Since the loss is determined by the model prediction probabilities, r_i can be formulated in terms of the log-probability gradient as $r_i := \partial \log p_\theta(y_i \mid x_i) / \partial z_i'$, where $y_i \sim p_\theta(\cdot \mid x_i)$. This allows us to reformulate the updates to measure the Fisher information learned by a model weight based on its predictive distribution [61].

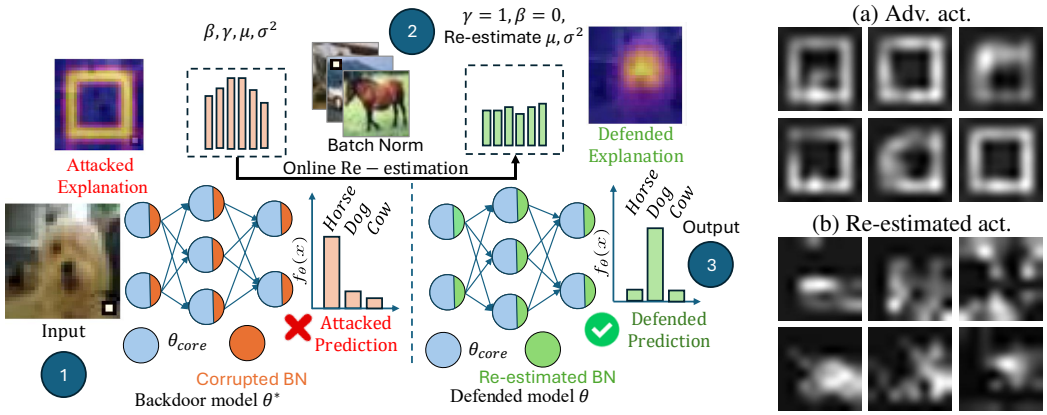


Figure 3: **Training-free BN recalibration defense.** Left: Overview of the proposed method: corrupted Batch Normalization (BN) statistics are discarded and re-estimated during inference, while corrupted affine parameters are reset to $(\gamma, \beta) = (1, 0)$. Right: (a) adversarially corrupted activations on the last BN layer of ResNet20; (b) Activations after affine-reset BN re-estimation using a small batch ($m = 4$), effectively suppressing backdoor-induced artifacts.

Proposition 3.6 (Fisher information imbalance between BN and core weights)**.** *Consider a poisoned batch $B \sim (1-p)\mathcal{D}_0 + p\mathcal{D}_t$. Let $\varepsilon_r, \varepsilon_{\hat{z}}$ denote the average within-group variances corresponding to r and $\hat{z}$, $\Delta_r, \Delta_{\hat{z}}$ their trigger-induced mean shifts between the clean and poisoned groups, and $A_r, A_{\hat{z}}$ their between-group variances. Defining the within-to-between scatter ratios as $t_r = \varepsilon_r / A_r$ and $t_{\hat{z}} = \varepsilon_{\hat{z}} / A_{\hat{z}}$, if $\Delta_r \neq 0$, $\Delta_{\hat{z}} \neq 0$ and $t_r t_{\hat{z}} \leq 1$, then*

$$|\kappa| \;\geq\; \frac{1 - \sqrt{t_r t_{\hat{z}}}}{\sqrt{(1+t_r)(1+t_{\hat{z}})}}, \qquad \kappa := \mathrm{Corr}(r, \hat{z}).$$

Because the post-BN output is an affine transformation ($z' = \gamma\hat{z} + \beta$), this magnitude identically captures the alignment between r and the post-norm output: $|\mathrm{Corr}(r, z')| = |\kappa|$. Hence $\kappa^2 \to 1$, and so $\mathrm{Cov}(r, \hat{z})^2 \to \mathrm{Var}(r)$, whenever the within-group scatter is small next to the between-group gap (proof: App. F). "Coherence" enters only as the hypothesis that the within-group scatter ratios $t_r, t_{\hat{z}}$ are small, which is when the bound is close to 1.

The BN affine parameters (γ, β) act directly on the BN output, so their gradients extract the projection of r onto the batch mean $\bar{r}$ and the normalized activations $\overline{r\hat{z}} = \mathrm{Cov}(r, \hat{z})$. By Eq. (5)*, the core weights receive only the residual $a_i = r_i - \bar{r} - \hat{z}_i\,\overline{r\hat{z}}$, i.e., r after the BN backward pass subtracts the batch mean and removes the $\hat{z}$-aligned component. The directions BN absorbs are exactly those it strips from the upstream signal, so the trigger-induced covariance enters BN additively and core subtractively. With $C := \gamma^2 L^2/(s^2 m)$,*

$$\|\Delta\theta_{\mathrm{BN}}\|_2 = \eta\sqrt{\bar{r}^{\,2} + \mathrm{Cov}(r,\hat{z})^2}\;\uparrow\;, \qquad \mathcal{F}_{\mathrm{BN}} = \mathbb{E}\big[\bar{r}^{\,2} + \mathrm{Cov}(r,\hat{z})^2\big]\;\uparrow\;,$$
$$\|\Delta\theta_{\mathrm{core}}\|_2 \leq \eta\sqrt{C(\mathrm{Var}(r) - \mathrm{Cov}(r,\hat{z})^2)}\;\downarrow, \qquad \mathcal{F}_{\mathrm{core}} \leq C\,\mathbb{E}\big[\mathrm{Var}(r) - \mathrm{Cov}(r,\hat{z})^2\big]\;\downarrow.$$

For mixed poisoning ($0 < p < 1$), κ^2 rises, so the factor $1 - \kappa^2$ in the core bound falls while $\mathcal{F}_{\mathrm{BN}}$ rises, and the Fisher ratio $\rho_{\mathrm{BN/core}} := \mathcal{F}_{\mathrm{BN}}/\mathcal{F}_{\mathrm{core}}$ grows: BN receives the trigger-induced gradient while core weights, driven by residual variance, keep fitting the clean task. This asymmetry suggests why backdoors are feasible—BN provides a parameter subspace through which the trigger is expressed without disturbing the core gradients that preserve clean accuracy. It does not mean that the final core weights are unaffected (Table 13).

This behavior is confirmed in Table 12, where the Fisher ratio $\rho_{\mathrm{BN/core}}$ is significantly elevated under mixed poisoning ($0 < p < 1$) compared to the clean baseline. A detailed theoretical sanity check of the extreme endpoints ($p = 0$ and $p = 1$) is provided in App. K.

Direction of the BN gradient. By Eq. (38), the BN scale gradient $\mathrm{Cov}(r, \hat{z}) = C_w + \mathrm{sgn}(\Delta_r \Delta_{\hat{z}})\sqrt{A_r A_{\hat{z}}}$ has a between-group term whose sign is the product of the two trigger-induced shifts $\Delta_r, \Delta_{\hat{z}}$. This term is nonzero only in channels where the trigger moves both quantities. Since

Table 1: Comparison across eight attacks. Results are reported as clean accuracy (Acc), attack success rate (ASR), and defense runtime (Time, in seconds). Lower ASR is better. All attacks were trained for 50 epochs on the pretrained model using a 10% poison ratio. Cell colors: within each row, the best/second-best/third-best (lowest) ASR and time among defenses are highlighted. We evaluate two variants of our method: ReBN, which fine-tunes only the BatchNorm layers, and FlashBN, which approximates BatchNorm parameters at inference time to sanitize the model without any retraining. Full results including NPD [62] and RNP [55] are given in Appendix Tables 6 and 7.

	Orig.		NAB [63]			UNIT [46]			FT-SAM [64]			BNP [52]			ReBN (Ours)			FlashBN (Ours)		
Attacks	Acc.	ASR	Acc.	ASR	T	Acc.	ASR	T	Acc.	ASR	T	Acc.	ASR	T	Acc.	ASR	T	Acc.	ASR	T
ReAttack	81.54	84.58	81.13	1.56	770.6	81.00	5.94	379.8	80.83	0.36	754.0	81.37	94.70	143.4	88.98	**0.07**	525.9	81.71	4.91	**26.8**
CTRL	91.53	72.24	93.21	1.32	677.2	90.82	10.38	268.8	89.41	0.08	666.5	84.35	**0.06**	137.3	93.45	2.16	526.6	91.37	4.97	**20.6**
FTrojan	93.88	100.00	92.43	3.53	552.1	91.44	6.75	241.1	90.36	4.65	1100.0	92.08	**0.62**	143.0	93.38	4.59	526.3	91.05	11.71	**26.7**
BPP	90.10	99.83	90.20	3.49	621.2	90.00	5.80	247.8	87.36	6.06	997.5	87.89	80.86	134.0	91.10	**1.35**	524.9	91.10	4.39	**21.5**
PoisonInk	90.25	99.80	90.00	3.00	561.7	87.14	5.23	305.3	89.36	**0.35**	958.6	88.52	99.79	143.1	89.46	2.23	530.1	90.74	3.99	**20.7**
Diffusion	92.22	94.48	89.39	1.29	561.8	86.53	10.96	293.3	89.86	0.65	952.4	82.09	92.26	134.0	92.56	**0.07**	526.3	87.61	27.19	**19.6**
Adaptive Blend	93.85	99.63	93.19	**3.19**	564.5	91.54	10.40	272.7	92.91	7.70	1016.6	91.91	98.17	136.5	93.56	5.80	527.4	91.09	13.12	**22.8**
COMBAT	89.16	77.37	90.88	1.62	548.2	88.92	8.99	318.7	87.35	**0.55**	1505.4	88.29	86.89	133.7	87.12	1.84	524.0	88.82	7.52	**21.0**
Average	90.32	90.99	90.05	2.38	607.2	88.42	8.06	290.9	88.43	2.55	993.9	87.06	69.17	138.1	91.20	**2.26**	526.4	89.19	9.72	**22.5**

Table 2: Accuracy (%) on triggered samples, measuring each method's ability to recover the ground-truth class under backdoor attacks. Higher values indicate better recovery performance. Cell colors: within each row, the best/second-best/third-best (highest) accuracies among defenses are highlighted. Methods based on BN recalibration (e.g., ReBN and FlashBN) consistently achieve superior performance across most attack settings.

Attacks	NPD	RNP	NAB	NAD	UNIT	FT-SAM	BNP	EP	SAU	ReBN (Ours)	FlashBN (Ours)
ReAttack	81.68	78.12	81.09	53.71	**96.29**	88.57	76.72	73.17	84.42	90.67	85.56
CTRL	20.96	17.21	69.62	67.08	58.71	61.62	26.47	60.84	66.38	**81.20**	80.97
FTrojann	7.09	17.09	72.12	68.05	93.89	67.80	84.47	34.93	18.71	**95.49**	87.73
BPP	31.43	36.61	1.80	26.06	77.41	84.06	68.03	23.44	83.53	**95.62**	84.21
PoisonInk	1.11	0.00	16.37	18.04	0.52	11.21	0.02	12.52	16.00	76.58	**85.80**
Diffusion	75.39	83.89	74.52	71.05	98.12	91.50	7.49	90.01	88.46	**99.16**	85.98
Adaptive Blend	35.11	42.87	56.03	51.06	7.03	68.81	1.72	1.23	67.44	**96.30**	89.10
COMBAT	26.12	0.72	48.37	37.05	30.27	60.12	42.91	18.04	49.31	83.28	**92.97**
Average	34.86	34.56	52.49	49.01	57.78	66.71	38.48	39.27	59.28	**89.79**	86.54

$|C_w| \leq \sqrt{\varepsilon_r \varepsilon_{\hat{z}}} = \sqrt{t_r t_{\hat{z}}} \sqrt{A_r A_{\hat{z}}}$, it dominates the within-group part whenever $t_r t_{\hat{z}} < 1$, the regime of Prop. 3.6. The direction of this gradient is therefore set by the trigger, not by its magnitude.

3.5 Defense Method

As a corollary, recomputing batch statistics at inference time removes the dependence on the stored BN statistics and affine parameters that are affected; whether this removes the backdoor depends on the conditions of Theorem 3.8.

Corollary 3.7 (Independence of the recalibrated operator from the stored BN state)**.** *Let a BN layer have parameters (γ, β) and running statistics (μ, σ^2), possibly corrupted by adversarial mixing. We define the inference-time recalibrated BN operator as*

$$\mathcal{R}_{\mathrm{BN}}(z; B) = \frac{z - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \quad \gamma = 1,\ \beta = 0, \tag{7}$$

where (μ_B, σ_B^2) are selectively computed from the inference batch B. This removes dependence on stored statistics (μ, σ^2) and affine parameters (γ, β).

An alternative is to retrain effective BN (ReBN) using Gaussian noise injection during fine-pruning on a small clean set.

ReBN removes backdoors by targeting BN affine parameters. It first resets all BN affines to $(1, 0)$, eliminating learned rescaling, and then computes the Fisher score $\mathcal{F}_{\mathrm{BN}}$ to identify parameters most sensitive to the adversarial backdoor.

The top-K highest-Fisher affine parameters are pruned and replaced with small Gaussian noise. A final masked-gradient fine-tuning step restores clean accuracy while preventing backdoor reactivation. The detailed implementation of the defenses is described in App. G.

Theorem 3.8 (Decision agreement after recalibration)**.** *Let L_θ be the Lipschitz constant of the network logits with respect to the core parameters. Assume: (A1) The poisoned core weights lie within distance δ of the clean core weights (an assumption: Prop. 3.6 attenuates the per-step core update but does not bound the final distance; see App. O); (A2) The logits are L_θ-Lipschitz in the core parameters; (A3) The clean core, when combined with a clean-recomputed BN layer, classifies the triggered input $\tilde{x}$ away from the adversarial target class by a margin of at least γ_0. Then the defended poisoned model (using the poisoned core and clean-recomputed BN) has a target-class margin of at most $-\gamma_0 + 2L_\theta\delta$ on $\tilde{x}$. Consequently, if $\delta < \gamma_0/(2L_\theta)$, the backdoor attack fails on every such input.*

Theorem 3.9 (BN versus core budget)**.** *Let δ and δ_{BN} be the distances between the poisoned and clean core and BN parameters, and let the target-class margin on a triggered input be L_θ-Lipschitz in the core and L_ϕ-Lipschitz in the BN parameters. If the clean model rejects the trigger with margin γ_0 and the attack succeeds with margin $\nu > 0$, then $L_\theta\delta + L_\phi\delta_{\mathrm{BN}} \geq \nu + \gamma_0$. Hence a core with $\delta \leq \gamma_0/L_\theta$ forces $\delta_{\mathrm{BN}} > 0$, and freezing BN forces $\delta \geq (\nu + \gamma_0)/L_\theta$ (proof: App. G.4).*

4 Experimental Setup

Unless a table states otherwise, models are trained from scratch using standard BackdoorBench [36] and explanation-aware attack protocols [32].

Attacks and defenses. We evaluate against 8 state-of-the-art black-box backdoor attacks: ReAttack [65], CTRL [6], FTrojan [66], BPP [67], PoisonInk [68], Diffusion [69, 70], Adaptive Blend [71], and COMBAT [72]. Many of these attacks utilize dynamic or imperceptible triggers; examples of triggered inputs are provided in Appendix Figure 6. We compare with 9 defenses: NPD [62], RNP [55], NAB [63], UNIT [46], FT-SAM [64], BNP [52], EP [52], NAD [7], and SAU [73].

For explanation-aware settings, we consider Gradient [24], Grad-CAM [22], and Relevance-CAM [23], under three attacks [32]: Simple Fooling, Red Herring, and Full Disguise. We compare against Softplus [41, 42], Aggregated Explanations [43], and SmoothGrad [74].

Models. We use PreAct ResNet-18 [36] for black-box adversarial attack evaluations and ResNet-20 for explanation-aware backdoor attacks [32], as these architectures are adopted by the respective works for their attack settings. We further evaluate MobileNetV3-S [75], ResNet-20 (w/ and w/o BN) [76], VGG-13 [77], ViT-B/16 [78], and WideResNet-28-10 [79]. **Datasets.** CIFAR-10, CIFAR-100 [18], TinyImageNet [19] and GTSRB [20].

Backdoor setup. Following [36], we poison 10% of the training data using sample-level mixing, and, unless a table states otherwise, models are trained for 100 epochs. Black-box attacks map triggered inputs to a fixed target class. For Simple Fooling, Red Herring, and Full Disguise, we follow the original implementation and settings on ResNet-20 (CIFAR-10 and GTSRB) as described in [32].

Metrics. We report clean accuracy, triggered accuracy, attack success rate (ASR), and defense time. For explanations, we compute MSE between clean and triggered saliency maps (Gradient, Grad-CAM, Relevance-CAM), averaged over the test set. Visualization uses a single CIFAR-10 batch with a poisoned ResNet-20.

4.1 Results

We organize our results into five evaluation settings, aligned with Section 4: (i) black-box backdoor defense against recent attacks and defenses on CIFAR-10 using PreAct ResNet-18 (Tables 1 and 2) and on Tiny ImageNet (Table 3); (ii) cross-architecture and cross-dataset evaluation over six models and three datasets (Table 9); (iii) explanation-aware attacks (Simple Fooling, Red Herring) on ResNet-20 with CIFAR-10 (Table 10); (iv) causal interventions on BN and core weights for ResNet-20 on CIFAR-10 (Table 13) and an ablation over poisoning ratios (Figure 5); and (v) visualization of explanation-aware backdoor defenses on CIFAR-10 and GTSRB across three explanation methods (Figures 9 and 7). Table 8 reports the mean $\pm$ standard deviation over 5 seeds of the defense, with the poisoned checkpoints held fixed. FlashBN's standard deviation is roughly $8\times$ smaller than ReBN's.

(Q1) Does BN-based defense result support our theoretical claim? Table 1 compares our methods, FlashBN and ReBN, against recent attacks and defenses. The results show that ReBN achieves the best performance in reducing ASR. FlashBN performs comparably to state-of-the-art methods

while being about $6\times$ faster. ReBN also yields a 1% higher clean accuracy than existing methods, while FlashBN maintains comparable clean accuracy. Table 2 reports accuracy on triggered inputs. Both methods significantly outperform prior work, with ReBN improving accuracy by 23.1 and FlashBN by 19.8 percentage points over the best prior defense (FT-SAM) on average across eight attack methods.. This supports our theory that BN is strongly influenced by adversarial triggers and can be effectively corrected by operating only on BN parameters.

(Q2) Does our defense generalize across multiple models and datasets? Table 9 in the Appendix summarizes the performance of the BadNets backdoor attack [25] and the corresponding defenses across five CNN architectures and ViT-B/16 on three benchmark datasets. On the BN-based architectures, FlashBN lowers ASR substantially, but it does not help models without BN (ASR stays 1.00 for ResNet-20 without BN on all three datasets) or ViT-B/16 (0.90–1.00), and clean accuracy drops for some architectures, for example, MobileNetV3-S on GTSRB from 93.80 to 76.62 and on CIFAR-10 from 80.30 to 68.00 (Table 9). We further evaluate on Tiny ImageNet using the public BackdoorBench models [36] (PreAct ResNet-18); Table 3, setup in App. H). ReBN reduces the average ASR from 93.24% to 0.47%, the lowest among all defenses. FlashBN reaches 16.54% on average but fails on LF and SIG.

(Q3) Does our defense generalize across explanation-aware backdoor attacks? Table 10 in the Appendix reports the explanation fidelity (MSE) of our defense under two explanation-aware attacks: Simple Fooling (SF) and Red Herring (RH). Our method reduces explanation MSE by 76% on average relative to the best-performing baseline per row (range 58–88%; Table 10). Both SF and RH explicitly manipulate generated explanations, whereas Full Disguise (FD) preserves a plausible clean explanation while altering the prediction. Since FD does not distort explanations, it is evaluated by triggered accuracy and ASR (Table 11).

Defense using only a few clean samples and an adaptive attack. Figures 4a–4c evaluate our defense under reduced batch sizes, showing that it remains effective with as few as 5 samples. Figure 5 shows that across poisoning ratios (up to 10%), our hypothesis regarding the gradient update imbalance between BN and core layers remains valid. Table 14 reports the challenging COMBAT attack with invisible triggers, where the adversary adaptively freezes the BN parameters at different poisoning ratios. In this setting, both of our methods show a significant drop in clean accuracy. This may signal an adaptively trained model, but it is not backdoor removal; we discuss it as a limitation below.

Visualization of explanation-aware attacks and defenses. Figure 8 compares our method with prior defenses, showing improved recovery of explanations. Figures 7 and 9 present results on CIFAR-10 and GTSRB, respectively. In both cases, our method restores explanations close to the clean baseline.

5 Conclusion

We provide a theoretical analysis of how BN layers interact with backdoor training: under the conditions we state, the trigger-induced update concentrates on the BN affine parameters, while the per-step update of the core weights is bounded. In the models we measure, BN parameters receive larger updates than core-layer weights, although the trigger is not stored in BN alone (Table 13). The poisoned core is thus not unaffected but recoverable through BN. Based on this analysis, we introduce a simple training-free BN recalibration, FlashBN, and a few-shot variant (ReBN) that substantially reduce attack success without intervening in core weights, demonstrating that BN is a powerful lever for backdoor mitigation rather than merely a training stabilizer.

Limitations. The analysis applies to BN-based CNNs: models without BN and ViT-B/16 are failure cases (Table 9), and extending it to other normalizations is left for future work. An attacker who freezes BN during poisoning must write the trigger into the core weights, and in that setting BN-based correction lowers clean accuracy instead of removing the backdoor (Table 14: 86.6 points on average for ReBN, 13.0 for FlashBN). We report this as observed, not as forced. The drop may signal an adaptively trained model, but it is a weaker guarantee than removal. Proposition 3.6 requires the within-group scatter ratios $t_r, t_{\hat{z}}$ to be small. Our guarantees are conditional: Thm. 3.8 needs a near-clean core (A1), which our proxy measurement supports for static but not for dynamic or reactivation triggers (App. O), and the Lipschitz constants in Thms. 3.8 and 3.9 are not bounded.

Acknowledgments and Disclosure of Funding

We thank our colleagues in Interactive Machine Learning (IML) at DFKI, Hasan Md Tusfiqur Alam, Abdulrahman Mohamed Selim, Omair Bhatti, László Kopácsi, and Michael Barz, for their valuable feedback and insightful discussions.

This research was funded by the German Federal Ministry of Research, Technology and Space (BMFTR) under grant numbers 16IW23002 (No-IDLE), 16IW24006 (NoIDLEChatGPT), and 16IS23064 (IUML), as well as by the Lower Saxony Ministry of Science and Culture (MWK) through the zukunft.niedersachsen program, and the Endowed Chair of AAI at the University of Oldenburg.

References

[1] Yudong Li, Shigeng Zhang, Weiping Wang, et al. Backdoor Attacks to Deep Learning Models and Countermeasures: A Survey. *IEEE Open Journal of the Computer Society*, 4:134–146, 2023. ISSN 2644-1268. doi: 10.1109/OJCS.2023.3267221. URL https://ieeexplore.ieee.org/document/10102775.

[2] Kaiyuan Zhang, Siyuan Cheng, Guangyu Shen, et al. Exploring the Orthogonality and Linearity of Backdoor Attacks. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 2105–2123, May 2024. doi: 10.1109/SP54263.2024.00225. URL https://ieeexplore.ieee.org/document/10646641.

[3] Yichen Wan, Youyang Qu, Wei Ni, et al. Data and Model Poisoning Backdoor Attacks on Wireless Federated Learning, and the Defense Mechanisms: A Comprehensive Survey. *IEEE Communications Surveys & Tutorials*, 26(3):1861–1897, 2024. ISSN 1553-877X. doi: 10.1109/COMST.2024.3361451. URL https://ieeexplore.ieee.org/document/10423783.

[4] Pengzhou Cheng, Zongru Wu, Wei Du, et al. Backdoor Attacks and Countermeasures in Natural Language Processing Models: A Comprehensive Security Review. *IEEE Transactions on Neural Networks and Learning Systems*, 36(8):13628–13648, August 2025. ISSN 2162-2388. doi: 10.1109/TNNLS.2025.3540303. URL https://ieeexplore.ieee.org/document/10905032.

[5] Xiangyu Qi, Tinghao Xie, Jiachen T. Wang, et al. Towards A Proactive ML Approach for Detecting Backdoor Poison Samples. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1685–1702, 2023. ISBN 978-1-939133-37-3. URL https://www.usenix.org/conference/usenixsecurity23/presentation/qi.

[6] Changjiang Li, Ren Pang, Zhaohan Xi, et al. An Embarrassingly Simple Backdoor Attack on Self-supervised Learning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4344–4355, October 2023. doi: 10.1109/ICCV51070.2023.00403. URL https://ieeexplore.ieee.org/document/10377889.

[7] Yige Li, Xixiang Lyu, Nodens Koren, et al. Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=9l0K4OM-oXE.

[8] Jonathan Hayase, Weihao Kong, Raghav Somani, et al. SPECTRE: defending against backdoor attacks using robust statistics. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4129–4139. PMLR, July 2021. URL https://proceedings.mlr.press/v139/hayase21a.html.

[9] Brandon Tran, Jerry Li, and Aleksander Mądry. Spectral signatures in backdoor attacks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 8011–8021, Red Hook, NY, USA, December 2018. Curran Associates Inc. URL https://dl.acm.org/doi/10.5555/3327757.3327896.

[10] Bolun Wang, Yuanshun Yao, Shawn Shan, et al. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723, May 2019. doi: 10.1109/SP.2019.00031. URL https://ieeexplore.ieee.org/document/8835365.

[11] Xiaojun Xu, Qi Wang, Huichen Li, et al. Detecting AI Trojans Using Meta Neural Analysis. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 103–120, May 2021. doi: 10.1109/SP40001.2021.00034. URL https://ieeexplore.ieee.org/document/9519467.

[12] Zhengming Zhang, Ashwinee Panda, Linyue Song, et al. Neurotoxin: Durable Backdoors in Federated Learning. In *Proceedings of the 39th International Conference on Machine Learning*, pages 26429–26446. PMLR, June 2022. URL https://proceedings.mlr.press/v162/zhang22w.html.

[13] Ao Wang, Hui Chen, Lihao Liu, et al. YOLOv10: real-time end-to-end object detection. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, volume 37 of *NIPS '24*, pages 107984–108011, Red Hook, NY, USA, December 2024. Curran Associates Inc. ISBN 9798331314385. doi: 10.52202/079017-3429.

[14] Rahima Khanam and Muhammad Hussain. YOLOv11: An Overview of the Key Architectural Enhancements. arXiv preprint arXiv:2410.17725, October 2024. URL http://arxiv.org/abs/2410.17725.

[15] Yunjie Tian, Qixiang Ye, and DAVID DOERMANN. YOLOv12: Attention-Centric Real-Time Object Detectors. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 78433–78457. Curran Associates, Inc., 2025. doi: 10.52202/085713-2627. URL https://dl.acm.org/doi/10.52202/085713-2627.

[16] Shun Wei, Jielin Jiang, and Xiaolong Xu. UniNet: A Contrastive Learning-guided Unified Framework with Feature Selection for Anomaly Detection. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9994–10003, June 2025. doi: 10.1109/CVPR52734.2025.00934. URL https://ieeexplore.ieee.org/document/11094947.

[17] Danfeng Qin, Chas Leichner, Manolis Delakis, et al. MobileNetV4: Universal Models for the Mobile Ecosystem. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision – ECCV 2024*, pages 78–96, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73661-2. doi: 10.1007/978-3-031-73661-2_5.

[18] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009. URL https://www.semanticscholar.org/paper/Learning-Multiple-Layers-of-Features-from-Tiny-Krizhevsky/5d90f06bb70a0a3dced62413346235c02b1aa086.

[19] Ya Le and Xuan Yang. Tiny ImageNet Visual Recognition Challenge. Technical report, Stanford University, 2015. URL https://www.semanticscholar.org/paper/Tiny-ImageNet-Visual-Recognition-Challenge-Le-Yang/384ce792cf2b2afbe001f2168bfe7d5e7804c736.

[20] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, et al. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks*, pages 1453–1460, July 2011. doi: 10.1109/IJCNN.2011.6033395. URL https://ieeexplore.ieee.org/document/6033395.

[21] Md Abdul Kadir, Amir Mosavi, and Daniel Sonntag. Evaluation Metrics for XAI: A Review, Taxonomy, and Practical Applications. In *2023 IEEE 27th International Conference on Intelligent Engineering Systems (INES)*, pages 000111–000124, July 2023. doi: 10.1109/INES59282.2023.10297629. URL https://ieeexplore.ieee.org/document/10297629.

[22] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision*, 128(2):336–359, February 2020. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. URL https://doi.org/10.1007/s11263-019-01228-7.

[23] Jeong Ryong Lee, Sewon Kim, Inyong Park, et al. Relevance-CAM: Your Model Already Knows Where to Look. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14939–14948, June 2021. doi: 10.1109/CVPR46437.2021.01470. URL https://ieeexplore.ieee.org/document/9578405.

[24] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv preprint arXiv:1312.6034, April 2014. URL http://arxiv.org/abs/1312.6034.

[25] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. arXiv preprint arXiv:1708.06733, March 2019. URL http://arxiv.org/abs/1708.06733.

[26] Yingqi Liu, Shiqing Ma, Yousra Aafer, et al. Trojaning Attack on Neural Networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. Internet Society, 2018. doi: 10.14722/ndss.2018.23291. URL https://doi.org/10.14722/ndss.2018.23291.

[27] Yuezun Li, Yiming Li, Baoyuan Wu, et al. Invisible Backdoor Attack with Sample-Specific Triggers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16443–16452. IEEE Computer Society, October 2021. ISBN 978-1-66542-812-5. doi: 10.1109/ICCV48922.2021.01615. URL https://www.computer.org/csdl/proceedings-article/iccv/2021/281200q6443/1BmFxknIBtm.

[28] Tuan Anh Nguyen and Anh Tuan Tran. WaNet - Imperceptible Warping-based Backdoor Attack. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=eEn8KTtJOx.

[29] Khoa Doan, Yingjie Lao, Weijie Zhao, et al. LIRA: Learnable, Imperceptible and Robust Backdoor Attacks. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11946–11956, October 2021. doi: 10.1109/ICCV48922.2021.01175. URL https://ieeexplore.ieee.org/document/9709953.

[30] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, et al. How To Backdoor Federated Learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, June 2020. URL https://proceedings.mlr.press/v108/bagdasaryan20a.html.

[31] Hubert Baniecki and Przemyslaw Biecek. Adversarial attacks and defenses in explainable artificial intelligence: A survey. *Information Fusion*, 107:102303, July 2024. ISSN 1566-2535. doi: 10.1016/j.inffus.2024.102303. URL https://www.sciencedirect.com/science/article/pii/S1566253524000812.

[32] Maximilian Noppel, Lukas Peter, and Christian Wressnegger. Disguising Attacks with Explanation-Aware Backdoors. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 664–681, May 2023. doi: 10.1109/SP46215.2023.10179308. URL https://ieeexplore.ieee.org/document/10179308.

[33] Maximilian Noppel and Christian Wressnegger. Composite Explanation-Aware Attacks. In *2025 IEEE Security and Privacy Workshops (SPW)*, pages 167–176, May 2025. doi: 10.1109/SPW67851.2025.00022. URL https://ieeexplore.ieee.org/document/11050844.

[34] Edward Chou, Florian Tramèr, and Giancarlo Pellegrino. SentiNet: Detecting Localized Universal Attacks Against Deep Learning Systems. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 48–54, May 2020. doi: 10.1109/SPW50608.2020.00025. URL https://ieeexplore.ieee.org/document/9283822/.

[35] Bao Gia Doan, Ehsan Abbasnejad, and Damith C. Ranasinghe. Februus: Input Purification Defense Against Trojan Attacks on Deep Neural Network Systems. In *Proceedings of the 36th Annual Computer Security Applications Conference*, ACSAC '20, pages 897–912, New York, NY, USA, December 2020. Association for Computing Machinery. ISBN 978-1-4503-8858-0. doi: 10.1145/3427228.3427264. URL https://dl.acm.org/doi/10.1145/3427228.3427264.

[36] Baoyuan Wu, Hongrui Chen, Mingda Zhang, et al. BackdoorBench: A Comprehensive Benchmark of Backdoor Learning. *Advances in Neural Information Processing Systems*, 35:10546–10559, December 2022. doi: 10.52202/068431-0766. URL https://dl.acm.org/doi/10.5555/3600270.3601036.

[37] Siyuan Cheng, Guangyu Shen, Guanhong Tao, et al. OdScan: Backdoor Scanning for Object Detection Models. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 1703–1721, May 2024. doi: 10.1109/SP54263.2024.00119. URL https://ieeexplore.ieee.org/document/10646667.

[38] Siyuan Cheng, Guanhong Tao, Yingqi Liu, et al. BEAGLE: Forensics of Deep Learning Backdoor Attack for Better Defense. arXiv preprint arXiv:2301.06241, January 2023. URL http://arxiv.org/abs/2301.06241.

[39] Wenbo Guo, Lun Wang, Xinyu Xing, et al. TABOR: A Highly Accurate Approach to Inspecting and Restoring Trojan Backdoors in AI Systems. arXiv preprint arXiv:1908.01763, August 2019. URL http://arxiv.org/abs/1908.01763.

[40] Yansong Gao, Change Xu, Derui Wang, et al. STRIP: a defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, pages 113–125, New York, NY, USA, December 2019. Association for Computing Machinery. ISBN 978-1-4503-7628-0. doi: 10.1145/3359789.3359790. URL https://dl.acm.org/doi/10.1145/3359789.3359790.

[41] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, et al. Explanations can be manipulated and geometry is to blame. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://dl.acm.org/doi/10.5555/3454287.3455504.

[42] Ann-Kathrin Dombrowski, Christopher J. Anders, Klaus-Robert Müller, et al. Towards robust explanations for deep neural networks. *Pattern Recognition*, 121:108194, January 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2021.108194. URL https://www.sciencedirect.com/science/article/pii/S0031320321003769.

[43] Laura Rieger and Lars Kai Hansen. A simple defense against adversarial attacks on heatmap explanations. arXiv preprint arXiv:2007.06381, July 2020. URL http://arxiv.org/abs/2007.06381.

[44] Qiuling Xu, Guanhong Tao, Jean Honorio, et al. MEDIC: Remove Model Backdoors via Importance Driven Cloning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20485–20494, June 2023. doi: 10.1109/CVPR52729.2023.01962. URL https://ieeexplore.ieee.org/document/10205279.

[45] Yi Zeng, Si Chen, Won Park, et al. Adversarial Unlearning of Backdoors via Implicit Hypergradient. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=MeeQkFYVbzW.

[46] Siyuan Cheng, Guangyu Shen, Kaiyuan Zhang, et al. UNIT: Backdoor Mitigation via Automated Neural Distribution Tightening. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision – ECCV 2024*, pages 262–281, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73033-7. doi: 10.1007/978-3-031-73033-7_15.

[47] Yige Li, Xixiang Lyu, Nodens Koren, et al. Anti-backdoor learning: training clean models on poisoned data. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, pages 14900–14912, Red Hook, NY, USA, December 2021. Curran Associates Inc. ISBN 978-1-71384-539-3. URL https://dl.acm.org/doi/10.5555/3540261.3541403.

[48] Angus Galloway, Anna Golubeva, Thomas Tanay, et al. Batch Normalization is a Cause of Adversarial Vulnerability. In *ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena*, May 2019. URL https://openreview.net/forum?id=BkxOwVShhE.

[49] Philipp Benz, Chaoning Zhang, and In So Kweon. Batch Normalization Increases Adversarial Vulnerability and Decreases Adversarial Transferability: A Non-Robust Feature Perspective. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7798–7807, October 2021. doi: 10.1109/ICCV48922.2021.00772. URL https://ieeexplore.ieee.org/document/9711063/.

[50] Haotao Wang, Aston Zhang, Shuai Zheng, et al. Removing Batch Normalization Boosts Adversarial Training. In *Proceedings of the 39th International Conference on Machine Learning*, pages 23433–23445. PMLR, June 2022. URL https://proceedings.mlr.press/v162/wang22ap.html.

[51] Dung Thuy Nguyen, Ngoc N. Tran, Taylor T. Johnson, et al. PBP: Post-training Backdoor Purification for Malware Classifiers. arXiv preprint arXiv:2412.03441, 2024. URL http://arxiv.org/abs/2412.03441.

[52] Runkai Zheng, Rongjun Tang, Jianze Li, et al. Pre-activation Distributions Expose Backdoor Neurons. In *Advances in Neural Information Processing Systems*, volume 35, pages 18667–18680. Curran Associates, Inc., 2022. doi: 10.52202/068431-1356. URL https://dl.acm.org/doi/10.5555/3600270.3601626.

[53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, Cambridge, MA, USA, November 2016. ISBN 978-0-262-03561-3. URL https://mitpress.mit.edu/9780262035613/deep-learning/.

[54] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, et al. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*, February 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

[55] Yige Li, Xixiang Lyu, Xingjun Ma, et al. Reconstructive Neuron Pruning for Backdoor Defense. In *Proceedings of the 40th International Conference on Machine Learning*, pages 19837–19854. PMLR, July 2023. URL https://proceedings.mlr.press/v202/li23v.html.

[56] Michael Kleinman, Alessandro Achille, and Stefano Soatto. Critical Learning Periods Emerge Even in Deep Linear Networks. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=Aq35gl2c1k.

[57] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456. PMLR, June 2015. URL https://proceedings.mlr.press/v37/ioffe15.html.

[58] Dong Yin, Ashwin Pananjady, Max Lam, et al. Gradient Diversity: a Key Ingredient for Scalable Distributed Learning. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 1998–2007. PMLR, March 2018. URL https://proceedings.mlr.press/v84/yin18a.html.

[59] Satrajit Chatterjee. Coherent Gradients: An Approach to Understanding Generalization in Gradient Descent-based Optimization. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=ryeFY0EFwS.

[60] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical Learning Periods in Deep Neural Networks. arXiv preprint arXiv:1711.08856, February 2019. URL http://arxiv.org/abs/1711.08856.

[61] Shun-ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry*, volume 191 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, Rhode Island, April 2007. ISBN 978-0-8218-4302-4 978-1-4704-4605-5. doi: 10.1090/mmono/191. URL https://www.ams.org/mmono/191.

[62] Mingli Zhu, Shaokui Wei, Hongyuan Zha, et al. Neural polarizer: a lightweight and effective backdoor defense via purifying poisoned features. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, pages 1132–1153, Red Hook, NY, USA, December 2023. Curran Associates Inc. URL https://dl.acm.org/doi/10.5555/3666122.3666177.

[63] Min Liu, Alberto Sangiovanni-Vincentelli, and Xiangyu Yue. Beating Backdoor Attack at Its Own Game. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4597–4606, October 2023. doi: 10.1109/ICCV51070.2023.00426. URL https://ieeexplore.ieee.org/document/10378513.

[64] Mingli Zhu, Shaokui Wei, Li Shen, et al. Enhancing Fine-Tuning based Backdoor Defense with Sharpness-Aware Minimization. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4443–4454, October 2023. doi: 10.1109/ICCV51070.2023.00412. URL https://ieeexplore.ieee.org/document/10377148.

[65] Mingli Zhu, Siyuan Liang, and Baoyuan Wu. Breaking the false sense of security in backdoor defense through re-activation attack. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, volume 37 of *NIPS '24*, pages 114928–114964, Red Hook, NY, USA, December 2024. Curran Associates Inc. ISBN 9798331314385. URL https://dl.acm.org/doi/10.5555/3737916.3741565.

[66] Tong Wang, Yuan Yao, Feng Xu, et al. An Invisible Black-Box Backdoor Attack Through Frequency Domain. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 396–413, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-19778-9. doi: 10.1007/978-3-031-19778-9_23.

[67] Zhenting Wang, Juan Zhai, and Shiqing Ma. BppAttack: Stealthy and Efficient Trojan Attacks against Deep Neural Networks via Image Quantization and Contrastive Adversarial Learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15054–15063, June 2022. doi: 10.1109/CVPR52688.2022.01465. URL https://ieeexplore.ieee.org/abstract/document/9879477.

[68] Jie Zhang, Chen Dongdong, Qidong Huang, et al. Poison Ink: Robust and Invisible Backdoor Attack. *IEEE Transactions on Image Processing*, 31:5691–5705, 2022. ISSN 1941-0042. doi: 10.1109/TIP.2022.3201472. URL https://ieeexplore.ieee.org/document/9870671.

[69] Hossein Souri, Arpit Bansal, Hamid Kazemi, et al. Generating Potent Poisons and Backdoors from Scratch with Guided Diffusion. In *ICML 2024 Next Generation of AI Safety Workshop*, July 2024. URL https://openreview.net/forum?id=MyAbNsZNI1.

[70] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pages 6840–6851, Red Hook, NY, USA, December 2020. Curran Associates Inc. ISBN 978-1-71382-954-6. URL https://dl.acm.org/doi/10.5555/3495724.3496298.

[71] Xiangyu Qi, Tinghao Xie, Yiming Li, et al. Revisiting the Assumption of Latent Separability for Backdoor Defenses. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=_wSHsgrVali.

[72] Tran Huynh, Dang Nguyen, Tung Pham, et al. COMBAT: Alternated Training for Effective Clean-Label Backdoor Attacks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(3):2436–2444, March 2024. ISSN 2374-3468. doi: 10.1609/aaai.v38i3.28019. URL https://ojs.aaai.org/index.php/AAAI/article/view/28019.

[73] Shaokui Wei, Mingda Zhang, Hongyuan Zha, et al. Shared Adversarial Unlearning: Backdoor Mitigation by Unlearning Shared Adversarial Examples. In *Advances in Neural Information Processing Systems*, volume 36, pages 25876–25909. Curran Associates, Inc., 2023. doi: 10.52202/075280-1126. URL https://dl.acm.org/doi/10.5555/3666122.3667248.

[74] Daniel Smilkov, Nikhil Thorat, Been Kim, et al. SmoothGrad: removing noise by adding noise. arXiv preprint arXiv:1706.03825, June 2017. URL http://arxiv.org/abs/1706.03825.

[75] Andrew Howard, Mark Sandler, Bo Chen, et al. Searching for MobileNetV3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324. IEEE Computer Society, October 2019. ISBN 978-1-72814-803-8. doi: 10.1109/ICCV.2019.00140. URL https://www.computer.org/csdl/proceedings-article/iccv/2019/480300b314/1hVlGG4j720.

[76] Yerlan Idelbayev. Proper ResNet implementation for CIFAR10/CIFAR100 in PyTorch. GitHub repository, 2018. URL https://github.com/akamaster/pytorch_resnet_cifar10.

[77] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556, April 2015. URL http://arxiv.org/abs/1409.1556.

[78] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[79] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. arXiv preprint arXiv:1605.07146, 2016. URL http://arxiv.org/abs/1605.07146.

[80] Zhaowei Cai, Avinash Ravichandran, Subhransu Maji, et al. Exponential Moving Average Normalization for Self-supervised and Semi-supervised Learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 194–203, June 2021. doi: 10.1109/CVPR46437.2021.00026. URL https://ieeexplore.ieee.org/abstract/document/9577900.

[81] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics. Springer, New York, NY, 2004. ISBN 978-1-4419-2322-6 978-0-387-21736-9. doi: 10.1007/978-0-387-21736-9. URL http://link.springer.com/10.1007/978-0-387-21736-9.

[82] Aidan N Gomez, Mengye Ren, Raquel Urtasun, et al. The Reversible Residual Network: Backpropagation Without Storing Activations. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://dl.acm.org/doi/10.5555/3294771.3294982.

[83] Alexander Munteanu, Simon Omlor, Zhao Song, et al. Bounding the Width of Neural Networks via Coupled Initialization A Worst Case Analysis. In *Proceedings of the 39th International Conference on Machine Learning*, pages 16083–16122. PMLR, June 2022. URL https://proceedings.mlr.press/v162/munteanu22a.html.

[84] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. In Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis, editors, *Research in Attacks, Intrusions, and Defenses*, pages 273–294, Cham, 2018. Springer International Publishing. ISBN 978-3-030-00470-5. doi: 10.1007/978-3-030-00470-5_13.

[85] Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, pages 16913–16925, Red Hook, NY, USA, December 2021. Curran Associates Inc. ISBN 978-1-71384-539-3. URL https://dl.acm.org/doi/10.5555/3540261.3541554.

[86] Runkai Zheng, Rongjun Tang, Jianze Li, and Li Liu. Data-Free Backdoor Removal Based on Channel Lipschitzness. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*, page 175–191, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-20064-9. doi: 10.1007/978-3-031-20065-6_11. URL https://doi.org/10.1007/978-3-031-20065-6_11.

[87] Xinyun Chen, Chang Liu, Bo Li, et al. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. arXiv preprint arXiv:1712.05526, December 2017. URL http://arxiv.org/abs/1712.05526.

[88] Yi Zeng, Won Park, Z. Morley Mao, et al. Rethinking the Backdoor Attacks' Triggers: A Frequency Perspective. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16453–16461, October 2021. doi: 10.1109/ICCV48922.2021.01616. URL https://ieeexplore.ieee.org/abstract/document/9710118.

[89] Tuan Anh Nguyen and Anh Tran. Input-Aware Dynamic Backdoor Attack. In *Advances in Neural Information Processing Systems*, volume 33, pages 3454–3464. Curran Associates, Inc., 2020. URL https://dl.acm.org/doi/10.5555/3495724.3496015.

[90] M. Barni, K. Kallas, and B. Tondi. A New Backdoor Attack in CNNS by Training Set Corruption Without Label Poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 101–105, 2019. doi: 10.1109/ICIP.2019.8802997. URL https://ieeexplore.ieee.org/document/8802997.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our experiments and theory fully support the main claim and contributions stated in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our work in the main paper as well as a designated section in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitations while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Full proofs and assumptions are provided in detail in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have included all relevant information supporting reproducibility in the appendix.Code is available at the link given in the abstract.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code link is provided in the abstract.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Experiments are setup following previous baselines and benchmarks. The paper and appendix include all relevant information and hyperparameters.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Appendix Table 8 reports the mean and standard deviation over 5 random seeds of the defenses (FlashBN, ReBN), with the poisoned checkpoints held fixed. The main tables are single runs; variance over attack training is not reported.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provided experimental details in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We fully follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have discussed the impact in the Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our paper does not produce any generative model or tool.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The materials that were borrowed from other resources is properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We describe the implementation details; the uploaded GitHub link is attached in the abstract.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing experiments or human subjects involved in this project.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We don't see relevant risk coming from our research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We use LLMs only for proofreading grammar and typos.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.

SUPPLEMENTARY MATERIAL FOR "BACKDOOR ATTACKS REROUTED: BATCH NORMALIZATION AS A SINK FOR TRIGGER SIGNALS"

Contents

1 Introduction **1**

2 Related Work **3**

3 Conceptual Framework and Method **3**

3.1 Backdoor attacks as optimization 3

3.2 BN Statistics (μ, σ^2) 4

3.3 BN Affine Parameters (γ, β) 5

3.4 Core Weight Update of Model 5

3.5 Defense Method 8

4 Experimental Setup **9**

4.1 Results 9

5 Conclusion **10**

Supplementary Material **25**

A Convergence of BN Running Statistics under Clean and Triggered Data Mixture **27**

A.1 Expected BN running statistics in batch 27

A.2 Unrolling the EMA and convergence 28

B Generalized Backdoor Attack **30**

C Optimizing the weight based on ERM **30**

D Gradients of BN Parameters **31**

E Core Weight Update of Model **33**

E.1 Derivative of the normalized activation 34

E.2 Chain rule derivative replacement. 35

E.3 Core Weight Update: Closed-Form Solution 37

F Bound on κ and Effect on Parameter Updates **38**

G Defense **39**

G.1 FlashBN 39

G.2 ReBN 39

G.3 Proof of Theorem 3.8 (Decision Agreement After Recalibration) 40

G.4 Proof of Theorem 3.9 40

H Tiny ImageNet Setting **40**

I Discussion of Additional Benchmark Results **41**

J Protective Role of Batch Normalization: Core vs. BN Update Dynamics **42**

K Fisher Information Sanity Checks **44**

L Ablation of Batch Size **44**

M Proofs of Lemma 3.4 and Eq. 38 **45**

N Clean-twin test **47**

O Checks of the Assumptions in Theorem 3.8 **47**

P Qualitative Visualizations **48**

P.1 Backdoor Sample Visualizations 48

P.2 Defending Against Explanation-Aware Attacks 48

Q Limitations & Discussion **48**

In this appendix, we provide a detailed proof of the theory presented in Sec. A through Sec. J. We revisit the fundamentals of batch normalization, formalize the backdoor attack formulation, and present proofs of the update-magnitude results and of the conditions under which BN absorbs the trigger-induced gradient signal. Additionally, in Sec. I, we report experiments on the CIFAR-10 dataset using ResNet-18, including comparisons with additional backdoor attacks and defenses, demonstrating that our method performs competitively while FlashBN is training-free; ReBN needs a small clean set. Sec. L reports the batch-size ablation, and Sec. L.0.1 presents the causal interventions. Finally, Sec. Q discusses the limitations of our method and its practical applications.

A Convergence of BN Running Statistics under Clean and Triggered Data Mixture

In this section, we show explicitly how Batch Normalization (BN) statistics (running mean: μ and running variance: σ^2) accumulate adversarial trigger patterns when training data is a mixture of clean and triggered images. We show that the BN running mean and variance converge (in expectation) to the mixture statistics rather than the clean statistics. Specifically, we first measure the expected values of the batch normalization parameters over a single minibatch in Section A.1, and then analyze how these parameters evolve in expectation over all training batches in Section A.2.

A.1 Expected BN running statistics in batch

In Section 3.2, we described how the exponential moving average (EMA) of μ and σ^2 is updated during training from each minibatch forward pass [80]. In this section, we discuss how these two parameters evolve over the course of training when the data comes from both the clean distribution $\mathcal{D}_0$ and the triggered distribution $\mathcal{D}_t$. The triggered distribution $\mathcal{D}_t$ is obtained from $\mathcal{D}_0$ by systematically perturbing the data using a training algorithm or by adding a coherent patch trigger pattern [25, 32, 31]. For simplicity, we assume the model is trained for only a single epoch.

At training iteration t, let $\mu_B^{(t)}$ and $\sigma_B^{2\,(t)}$ denote the mean and variance of the current mini-batch $B^{(t)} = \{z_i\}_{i=1}^m$, where z_i are the pre-activation feature activations from the previous convolutional layer. We use the superscript (t) to index the training step, which is distinct from the subscript t used to denote the triggered distribution $\mathcal{D}_t$. The batch statistics can be written as

$$\mu_B^{(t)} = \frac{1}{m}\sum_{i=1}^{m} z_i, \tag{8}$$

$$\sigma_B^{2\,(t)} = \frac{1}{m}\sum_{i=1}^{m}\big(z_i - \mu_B^{(t)}\big)^2. \tag{9}$$

BN maintains running (non-trainable) statistics $\mu^{(t)}$ and $\sigma^{2\,(t)}$ that are used at inference time. During training, these are updated via an EMA:

$$\begin{aligned}\mu^{(t+1)} &\leftarrow \alpha\,\mu^{(t)} + (1-\alpha)\,\mu_B^{(t)},\\ \sigma^{2\,(t+1)} &\leftarrow \alpha\,\sigma^{2\,(t)} + (1-\alpha)\,\sigma_B^{2\,(t)},\end{aligned} \tag{10}$$

where $\alpha \in (0,1)$ is the EMA decay factor (larger α means slower adaptation).

We model each minibatch $B^{(t)}$ as being drawn from a mixture of clean and triggered examples:

$$B^{(t)} \sim (1-p)\,\mathcal{D}_0 + p\,\mathcal{D}_t, \qquad p \in (0,1), \tag{11}$$

For a single feature value Z in the batch, define

$$\mu_0 := \mathbb{E}[Z \mid Z \sim \mathcal{D}_0], \qquad \mu_t := \mathbb{E}[Z \mid Z \sim \mathcal{D}_t], \tag{12}$$

$$\sigma_0^2 := \mathrm{Var}(Z \mid Z \sim \mathcal{D}_0), \qquad \sigma_t^2 := \mathrm{Var}(Z \mid Z \sim \mathcal{D}_t). \tag{13}$$

Introduce a Bernoulli indicator $Y \in \{0,1\}$ which selects the source:

$$\mathbb{P}(Y=0) = 1-p, \qquad \mathbb{P}(Y=1) = p.$$

Then $Z \mid (Y=0) \sim \mathcal{D}_0$ and $Z \mid (Y=1) \sim \mathcal{D}_t$.

Expected batch mean. By the law of total expectation,

$$\begin{aligned}\mathbb{E}[\mu_B^{(t)}] &= \mathbb{E}[Z] \\ &= \mathbb{E}\big[\,\mathbb{E}[Z \mid Y]\,\big] \\ &= (1-p)\,\mathbb{E}[Z \mid Y=0] + p\,\mathbb{E}[Z \mid Y=1] \\ &= (1-p)\,\mu_0 + p\,\mu_t. \end{aligned} \tag{14}$$

We denote this mixture mean by

$$\boxed{(1-p)\,\mu_0 + p\,\mu_t := \mu^\star} \tag{15}$$

This means that the batch mean shifts in the presence of triggered samples. Using such a contaminated estimate of the mean can adversarially bias the model, since the BN statistics become partly aligned with the trigger rather than the underlying clean feature distribution.

Expected batch variance. For the variance, we use the law of total variance [81]:

$$\mathrm{Var}(Z) = \mathbb{E}\big[\mathrm{Var}(Z \mid Y)\big] + \mathrm{Var}\big(\mathbb{E}[Z \mid Y]\big). \tag{16}$$

The first term is the average "within-component" variance:

$$\mathbb{E}[\mathrm{Var}(Z \mid Y)] = (1-p)\,\sigma_0^2 + p\,\sigma_t^2. \tag{17}$$

The second term captures the variance between the two component means. Since

$$\mathbb{E}[Z \mid Y=0] = \mu_0, \qquad \mathbb{E}[Z \mid Y=1] = \mu_t,$$

we have a two-point random variable that equals μ_0 with probability $1-p$ and μ_t with probability p, so

$$\mathrm{Var}(\mathbb{E}[Z \mid Y]) = p(1-p)(\mu_t - \mu_0)^2. \tag{18}$$

Combining both terms in (16) gives

$$\mathrm{Var}(Z) = (1-p)\,\sigma_0^2 + p\,\sigma_t^2 + p(1-p)(\mu_t - \mu_0)^2. \tag{19}$$

up to the usual finite-sample correction, which ${\sigma_B^2}^{(t)}$ is an unbiased estimator of $\mathrm{Var}(Z)$, so

$$\mathbb{E}[{\sigma_B^2}^{(t)}] = (1-p)\,\sigma_0^2 + p\,\sigma_t^2 + p(1-p)(\mu_t - \mu_0)^2. \tag{20}$$

We denote this mixture variance by

$$\boxed{(1-p)\,\sigma_0^2 + p\,\sigma_t^2 + p(1-p)(\mu_t - \mu_0)^2 := \sigma^{2\star}} \tag{21}$$

We observe that the batch variance is also affected by adversarial bias. If the model uses such a shifted variance estimate during normalization, the resulting feature activations become distributionally misaligned, which can in turn shift the model's prediction.

In summary, by linearity of expectation and the law of total variance,

$$\begin{aligned}\mathbb{E}[\mu_B^{(t)}] &= \mu^\star, \\ \mathbb{E}[{\sigma_B^2}^{(t)}] &= \sigma^{2\star}, \end{aligned} \tag{22}$$

, which we treat them as fixed mixture statistics (as long as $\mathcal{D}_0$, $\mathcal{D}_t$, and p are fixed). In the main paper, we use $E[\mu_B]$ and $E[\sigma_B^2]$ only for similarity and readability.

A.2 Unrolling the EMA and convergence

We now show that the BN running statistics converge (in expectation) to the mixture statistics $(\mu^\star, \sigma^{2\star})$.

Mean. Take expectations on the mean update in (10) and use $\mathbb{E}[\mu_B^{(t)}] = \mu^\star$:

$$\begin{aligned}\mathbb{E}[\mu^{(t+1)}] &= \mathbb{E}\big[\alpha\,\mu^{(t)} + (1-\alpha)\,\mu_B^{(t)}\big] \\ &= \alpha\,\mathbb{E}[\mu^{(t)}] + (1-\alpha)\,\mu^\star. \end{aligned} \tag{23}$$

Define

$$u_t := \mathbb{E}[\mu^{(t)}].$$

Then u_t satisfies the linear recurrence.

$$u_{t+1} = \alpha\,u_t + (1-\alpha)\,\mu^\star. \tag{24}$$

Unrolling (24):

$$\begin{aligned} u_1 &= \alpha u_0 + (1-\alpha)\mu^\star, \\ u_2 &= \alpha u_1 + (1-\alpha)\mu^\star \\ &= \alpha^2 u_0 + (1-\alpha)\mu^\star(1+\alpha), \\ &\vdots \\ u_t &= \alpha^t u_0 + (1-\alpha)\mu^\star \sum_{k=0}^{t-1} \alpha^k. \end{aligned} \tag{25}$$

The geometric sum evaluates to

$$\sum_{k=0}^{t-1} \alpha^k = \frac{1-\alpha^t}{1-\alpha},$$

$$\begin{aligned} u_t &= \alpha^t u_0 + (1-\alpha)\mu^\star \frac{1-\alpha^t}{1-\alpha} \\ &= \alpha^t u_0 + (1-\alpha^t)\,\mu^\star. \end{aligned} \tag{26}$$

Since $0 < \alpha < 1$, we have $\alpha^t \to 0$ as $t \to \infty$, and thus

$$\mathbb{E}[\mu^{(t)}] = u_t \to \mu^\star. \tag{27}$$

This implies that increasing the number of training steps does not diminish the effect of adversarial bias: under a fixed mixture, the expected running mean converges to $\mu^\star$, not to μ_0. Since the running statistics are frozen at deployment, the bias persists unless they are reset or re-estimated.

Variance. The variance update in (10) has the same structure:

$$\begin{aligned}\mathbb{E}[\sigma^{2\,(t+1)}] &= \mathbb{E}\big[\alpha\,\sigma^{2\,(t)} + (1-\alpha)\,\sigma_B^{2\,(t)}\big] \\ &= \alpha\,\mathbb{E}[\sigma^{2\,(t)}] + (1-\alpha)\,\sigma^{2\star}, \end{aligned} \tag{28}$$

using $\mathbb{E}[\sigma_B^{2\,(t)}] = \sigma^{2\star}$ in (22). Let

$$v_t := \mathbb{E}[\sigma^{2\,(t)}].$$

Then

$$v_{t+1} = \alpha\,v_t + (1-\alpha)\,\sigma^{2\star}, \tag{29}$$

which is identical in form to (24). By the same unrolling argument,

$$v_t = \alpha^t v_0 + (1-\alpha^t)\,\sigma^{2\star} \;\to\; \sigma^{2\star} \quad \text{as } t \to \infty. \tag{30}$$

Theorem A.1 (Convergence of BN statistics under a clean–triggered mixture)**.** *Under the mixture model* (11) *with fixed* $p \in (0,1)$ *and EMA updates* (10)*, the BN running statistics satisfy*

$$\mathbb{E}[\mu^{(t)}] \to \mu^\star, \qquad \mathbb{E}[\sigma^{2\,(t)}] \to \sigma^{2\star} \quad \textit{as } t \to \infty,$$

where $\mu^\star$ *and* $\sigma^{2\star}$ *are the mixture moments in* (15) *and* (21)*. In particular, if* $\mu_t \neq \mu_0$ *or* $\sigma_t^2 \neq \sigma_0^2$*, then* $(\mu^\star, \sigma^{2\star}) \neq (\mu_0, \sigma_0^2)$*, so BN converges to adversarially biased statistics rather than the clean ones.*

B Generalized Backdoor Attack

Adversarial backdoor attacks, whether the attacker has access to both the training data and model parameters, or only to the data, can be formulated as an adversarial training objective with two components: a classification manipulation loss and an explanation manipulation loss. To analyze both threat models in a unified way, we formalize the backdoor training objective as an Empirical Risk Minimization (ERM) [53] problem that jointly optimizes prediction and explanation losses, following prior formulations [32, 54]. Let f_θ denote a model parameterized by θ, x the input, y the label, and $e(f_\theta, x, y)$ an explainer. For a minibatch B, we consider the following unified objective:

$$\mathcal{L}(B;\theta) = (1-\lambda)\,\frac{1}{|B|}\sum_{(x,y)\in B}\mathcal{L}_{\text{pred}}\big(f_\theta(x),\, y\big) + \;\lambda\,\frac{1}{|B|}\sum_{(x,y)\in B}\mathcal{L}_{\text{expl}}\big(e(f_\theta,x,y),\, E_{\text{tgt}}\big), \tag{31}$$

where $\mathcal{L}_{\text{pred}}$ is the prediction loss, $\mathcal{L}_{\text{expl}}$ is the explanation loss, and E_{tgt} is an attacker-specified target explanation.

Here, $\lambda \in [0,1]$ controls whether the objective models only a standard backdoor attack or additionally includes the explanation-aware backdoor setting. When $\lambda = 0$, the objective reduces to a standard classification loss (classic backdoor/data-poisoning ERM):

$$\frac{1}{|B|}\sum_{(x,y)\in B}\mathcal{L}_{\text{pred}}\big(f_\theta(x), y\big),$$

so the attacker only enforces misclassification on triggered inputs. When $\lambda > 0$, the attacker additionally optimizes the explanation loss $\mathcal{L}_{\text{expl}}$ so that the model–explainer pair produces attacker-chosen explanations E_{tgt}, yielding an *explanation-aware* backdoor.

The loss in Eq. (31) is minimized on a mixture of clean and triggered samples: each sample in a minibatch B is drawn from the triggered distribution $\mathcal{D}_t$ with probability p, and from the clean distribution $\mathcal{D}_0$ with probability $(1-p)$. The triggered distribution $\mathcal{D}_t$ is derived from $\mathcal{D}_0$ by embedding a trigger pattern (illustrated in Figure 1a) into each sample x, while altering the associated label y to the attacker's target specification. E_{tgt} denotes the target saliency map that the attacker intends the model and explainer to produce for a triggered input. This unified formulation allows us to analyze how coupling prediction and explanation objectives affects backdoor persistence across both data-poisoning and model-supply threat scenarios.

Attack-specific configurations (explanation-aware case). For triggered samples, different explanation-aware backdoor variants correspond to different choices of the target label y_{tgt} and target explanation E_{tgt} [32]:

- **SF (Simple Fooling):** $y_{\text{tgt}} = y, \quad E_{\text{tgt}} = E_t.$
- **RH (Red Herring):** $y_{\text{tgt}} = y_t, \quad E_{\text{tgt}} = E_t.$
- **FD (Full Disguise):** $y_{\text{tgt}} = y_t, \quad E_{\text{tgt}} = e(f_\theta, x, y).$

Here, E_t denotes a trigger-shaped target saliency map and y_t the attacker-chosen target label. Within this ERM view, standard backdoors and explanation-aware backdoors are recovered as special cases of the same training objective.

C Optimizing the weight based on ERM

For simplicity, we denote the empirical risk minimization (ERM) objective in Eq. (1) by $\mathcal{L}(B;\theta)$. We decompose model parameters into two disjoint groups:

$$\theta = \big(\theta_{\text{core}},\, \theta_{\text{BN}}\big),$$

where θ_{core} contains all non-BN parameters (e.g. convolutional kernels and fully connected weights), and $\theta_{\text{BN}} = \{\gamma, \beta\}$ denotes the BN affine parameters that scale and shift normalized activations.

Gradients of core and BN parameters. We write the ERM gradients with respect to the two parameter groups as

$$\nabla\theta_{\text{core}} = \frac{\partial \mathcal{L}(B;\theta)}{\partial \theta_{\text{core}}}, \qquad \nabla\theta_{\text{BN}} = \left(\frac{\partial \mathcal{L}(B;\theta)}{\partial \gamma}, \frac{\partial \mathcal{L}(B;\theta)}{\partial \beta}\right).$$

While θ_{core} influences the hierarchical feature transforms of the network, the BN affine parameters (γ, β) directly modulate the post-normalization feature distribution. Consequently, their gradients exhibit a much higher sensitivity to distributional perturbations (e.g., triggered vs. clean data), making BN layers a critical locus for attacker-induced gradient alignment.

In the following analysis, we examine how $\nabla\theta_{\text{BN}}$ and $\nabla\theta_{\text{core}}$ behave under the clean/triggered mixture, and show that the objective induces coherent, attacker-aligned updates predominantly in BN layers.

D Gradients of BN Parameters

Setup. For one BN channel and a batch of size m,

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} z_i, \qquad \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(z_i - \mu_B)^2,$$

$$\hat{z}_i = \frac{z_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \qquad z_i' = \gamma\,\hat{z}_i + \beta,$$

$$\mathcal{L} = \frac{1}{m}\sum_{i=1}^{m} \ell_i.$$

We will also use

$$r_i := \frac{\partial \ell_i}{\partial z_i'}, \qquad g_i := \frac{\partial \mathcal{L}}{\partial z_i'} = \frac{1}{m} r_i. \tag{32}$$

BN centering implies

$$\frac{1}{m}\sum_{i=1}^{m} \hat{z}_i = 0.$$

Step 1 — Chain rule for β. Start from the full chain rule:

$$\frac{\partial \mathcal{L}}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial \ell_i}\frac{\partial \ell_i}{\partial z_i'}\frac{\partial z_i'}{\partial \beta}.$$

Since

$$\mathcal{L} = \frac{1}{m}\sum_{j=1}^{m} \ell_j \quad \Rightarrow \quad \frac{\partial \mathcal{L}}{\partial \ell_i} = \frac{1}{m},$$

and by definition

$$\frac{\partial \ell_i}{\partial z_i'} = r_i,$$

and from $z_i' = \gamma\hat{z}_i + \beta$,

$$\frac{\partial z_i'}{\partial \beta} = 1,$$

we obtain

$$\frac{\partial \mathcal{L}}{\partial \beta} = \sum_{i=1}^{m}\left(\frac{1}{m}\right) r_i \cdot 1 = \frac{1}{m}\sum_{i=1}^{m} r_i.$$

Using $g_i = \frac{1}{m} r_i$, it can also be written as

$$\boxed{\frac{\partial \mathcal{L}}{\partial \beta} = \sum_{i=1}^{m} g_i} \tag{33}$$

Step 2 — Chain rule for γ. Again, start from the full chain rule:

$$\frac{\partial \mathcal{L}}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial \ell_i} \frac{\partial \ell_i}{\partial z_i'} \frac{\partial z_i'}{\partial \gamma}.$$

As before,

$$\frac{\partial \mathcal{L}}{\partial \ell_i} = \frac{1}{m}, \qquad \frac{\partial \ell_i}{\partial z_i'} = r_i.$$

From $z_i' = \gamma \hat{z}_i + \beta$ we have

$$\frac{\partial z_i'}{\partial \gamma} = \hat{z}_i.$$

Thus

$$\frac{\partial \mathcal{L}}{\partial \gamma} = \sum_{i=1}^{m} \left(\frac{1}{m}\right) r_i \, \hat{z}_i = \frac{1}{m} \sum_{i=1}^{m} r_i \, \hat{z}_i.$$

Using $g_i = \frac{1}{m} r_i$, we can also write

$$\boxed{\frac{\partial \mathcal{L}}{\partial \gamma} = \sum_{i=1}^{m} g_i \, \hat{z}_i} \tag{34}$$

From Eqs. (33) and (34), the BN affine gradients for a minibatch $B = \{(x_i, y_i)\}_{i=1}^{m}$ take the form after substitution of $g_i = \frac{1}{m} r_i$

$$\frac{\partial \mathcal{L}}{\partial \beta} = \sum_{i=1}^{m} g_i = \bar{r}, \qquad \frac{\partial \mathcal{L}}{\partial \gamma} = \sum_{i=1}^{m} g_i \hat{z}_i = \mathrm{Cov}(r, \hat{z}), \tag{35}$$

$$\bar{r} = \frac{1}{m} \sum_{i=1}^{m} r_i, \qquad \mathrm{Cov}(r, \hat{z}) = \frac{1}{m} \sum_{i=1}^{m} r_i \hat{z}_i.$$

Mixture Decomposition. Under the sampling mixture of clean and triggered batches, the statistics decompose linearly as

$$\begin{aligned} \bar{r} &= (1-p)\, \bar{r}_{\mathcal{D}_0} + p\, \bar{r}_{\mathcal{D}_t}, \\ \mathrm{Cov}(r, \hat{z}) &= (1-p)\, \overline{r\hat{z}}_{\mathcal{D}_0} + p\, \overline{r\hat{z}}_{\mathcal{D}_t}, \end{aligned} \tag{36}$$

where $\bar{r}_{\mathcal{D}_0}$, $\bar{r}_{\mathcal{D}_t}$ denote the expected residual means under clean and triggered distributions, and $\overline{r\hat{z}}_{\mathcal{D}_0}$, $\overline{r\hat{z}}_{\mathcal{D}_t}$ are the group averages of $r\hat{z}$ (uncentered, i.e., not mean-subtracted within the group). Eq. (36) is therefore exact for these uncentered group moments; Eq. (38) in the main text splits it into within-group covariances and a between-group term.

Why $\mathrm{Cov}(r, \hat{z})$ is large on triggered data follows from Lemma 3.4 and Eq. (38); both are proved in App. M. As shown in Figure 2a, the mean ℓ_2 gradient norms for both the core weights and the BN parameters remain smooth and stable when training on clean data. In contrast, during adversarial fine-tuning (Figure 2c), we observe pronounced spikes in the BN parameter updates.

Connection to Theorem 3.3. The optimization update step with the learning rate η and taking the ℓ_2 norm of the affine update gives

$$\boxed{\left\|\Delta\theta_{\mathrm{BN}}\right\|_2 \;=\; \eta \sqrt{\bar{r}^{\,2} + \mathrm{Cov}(r, \hat{z})^2} \;=\; \eta \sqrt{\bar{r}^{\,2} + \kappa^2 \mathrm{Var}(r)}} \tag{37}$$

exactly the expression in Theorem 3.3. Because both $\bar{r}$ and $\mathrm{Cov}(r, \hat{z})$ are *batch-averaged* statistics, the magnitude of the BN update remains independent of the batch size under the mean-loss formulation.

The mixture expansion in Eq. (36) shows that even a small fraction of triggered samples ($p > 0$) reliably shifts $\bar{r}$ and $\mathrm{Cov}(r, \hat{z})$ away from zero, because triggered inputs are separated from clean inputs in the penultimate features (Lemma 3.4). By Theorem 3.3, these non-zero statistics translate into consistent, attacker-aligned updates of the BN affine parameters. Thus, BN layers serve as a

highly sensitive locus for explanation-aware backdoor insertion, amplifying residual alignment while remaining unaffected by batch-size scaling.

Covariance splitting. Let $Y = 1$ mark triggered samples, $\Pr(Y = 1) = p$. In a BN channel, let $\Delta_r, \Delta_{\hat{z}}$ be the gaps between the two group means (triggered minus clean), $\varepsilon_r, \varepsilon_{\hat{z}}$ the p-weighted average within-group variances, and C_w the p-weighted average within-group covariance, and put $A_r = p(1-p)\Delta_r^2$, $A_{\hat{z}} = p(1-p)\Delta_{\hat{z}}^2$. By the laws of total variance and covariance,

$$\mathrm{Var}(r) = \varepsilon_r + A_r, \quad \mathrm{Var}(\hat{z}) = \varepsilon_{\hat{z}} + A_{\hat{z}} = 1, \quad \mathrm{Cov}(r, \hat{z}) = C_w + \mathrm{sgn}(\Delta_r \Delta_{\hat{z}})\sqrt{A_r A_{\hat{z}}}, \quad (38)$$

where $\mathrm{Var}(\hat{z}) = 1$ because BN normalizes $\hat{z}$ over the batch (Lemma 3.2). The covariance splits into a within-group part and a between-group part, which is non-zero only because the groups differ in mean. By Cauchy–Schwarz, $|C_w| \le \sqrt{\varepsilon_r \varepsilon_{\hat{z}}}$. We write $t_r = \varepsilon_r / A_r$ and $t_{\hat{z}} = \varepsilon_{\hat{z}} / A_{\hat{z}}$ for the ratios of within-group to between-group scatter.

Similarly, in the next section we will examine how the presence of triggered examples in the dataset perturbs the gradient updates of the core network weights θ_{core}.

E Core Weight Update of Model

Forward pass. For simplicity of proof, we consider a Conv–BN network, where a convolutional layer is followed by Batch Normalization (BN), x_i is an input.

$$z_i = W x_i + b, \qquad \hat{z}_i = \frac{z_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \qquad z_i' = \gamma\, \hat{z}_i + \beta, \quad (39)$$

Here, we analyze how gradient updates propagate through learnable layers during backpropagation, while Batch Normalization (BN) simultaneously normalizes their inputs [82].

$$\frac{\partial \mathcal{L}}{\partial z_i} = \sum_{k=1}^{m} \underbrace{\frac{\partial \mathcal{L}}{\partial z_k'}}_{\text{loss}} \underbrace{\frac{\partial z_k'}{\partial \hat{z}_k}}_{\text{affine}} \underbrace{\frac{\partial \hat{z}_k}{\partial z_i}}_{\text{normalization}} . \quad (40)$$

Recalling from D Equation 32 per-sample gradient of the loss with respect to batch norm's output features is defined as

$$r_i := \frac{\partial \ell_i}{\partial z_i'}, \qquad \mathcal{L} = \frac{1}{m}\sum_{i=1}^{m} \ell_i, \qquad \underbrace{\frac{\partial \mathcal{L}}{\partial z_k'}}_{\text{loss}} = \frac{r_k}{m}. \quad (41)$$

From section D we know that

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} z_i, \qquad \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (z_i - \mu_B)^2, \quad (42)$$

$$s = \sqrt{\sigma_B^2 + \varepsilon}, \qquad \hat{z}_k = \frac{z_k - \mu_B}{s}. \quad (43)$$

We write $\hat{z}_k$ for any k in the batch B, with $k \in \{1, \dots, m\}$.

Let $s := \sqrt{\sigma_B^2 + \varepsilon}$.

Here, for any sample k, $z_k' = \gamma\, \hat{z}_k + \beta$, $\frac{\partial z_k'}{\partial \hat{z}_k} = \gamma$ represents the gradient of the BN's affine transformation.

The *normalization* gradient $\frac{\partial \hat{z}_k}{\partial z_i}$ denotes the (k, i)-entry of the Jacobian of the normalization map $z \mapsto \hat{z}$, for $i, k \in \{1, \dots, m\}$, capturing how a change in one input z_i affects all normalized outputs $\hat{z}_k$ due to shared batch statistics (Eq. D). We consider BN in training mode for a single channel and

a mini-batch $\{z_1, \ldots, z_m\}$. Our goal in this step is to obtain an explicit expression for $\partial \hat{z}_k / \partial z_i$, because this is the term that couples all samples in the batch during backpropagation and a key term that compensates the gradient update for the core weight with respect to the adversarial trigger in the input.

E.1 Derivative of the normalized activation

We now differentiate the normalized activation $\hat{z}_k = \frac{z_k - \mu_B}{s}$ with respect to z_i:

$$\begin{aligned}\frac{\partial \hat{z}_k}{\partial z_i} &= \frac{1}{s}\frac{\partial (z_k - \mu_B)}{\partial z_i} + (z_k - \mu_B)\frac{\partial (1/s)}{\partial z_i} \\ &= \frac{1}{s}\left(\delta_{ik} - \frac{\partial \mu_B}{\partial z_i}\right) - \frac{z_k - \mu_B}{s^2} \cdot \frac{\partial s}{\partial z_i}.\end{aligned} \tag{44}$$

Here, δ_{ik} denotes the Kronecker delta, which equals 1 when $i = k$ and 0 otherwise.

Derivative of the standard deviation (chain rule). Since $s = \sqrt{\sigma_B^2 + \varepsilon}$, we treat s as a function of σ_B^2 and apply the chain rule:

$$\begin{aligned}\frac{\partial s}{\partial z_i} &= \frac{\partial}{\partial z_i}(\sigma_B^2 + \varepsilon)^{1/2} \\ &= \frac{1}{2}(\sigma_B^2 + \varepsilon)^{-1/2}\, \frac{\partial (\sigma_B^2 + \varepsilon)}{\partial z_i} \\ &= \frac{1}{2s}\, \frac{\partial \sigma_B^2}{\partial z_i}.\end{aligned} \tag{45}$$

where ε is constant w.r.t. z_i.

Now expand $\partial \sigma_B^2 / \partial z_i$ using the definitions

$$\sigma_B^2 = \frac{1}{m}\sum_{j=1}^{m}(z_j - \mu_B)^2, \qquad \mu_B = \frac{1}{m}\sum_{k=1}^{m} z_k.$$

We first apply the derivative to the sum:

$$\begin{aligned}\frac{\partial \sigma_B^2}{\partial z_i} &= \frac{\partial}{\partial z_i}\left[\frac{1}{m}\sum_{j=1}^{m}(z_j - \mu_B)^2\right] \\ &= \frac{1}{m}\sum_{j=1}^{m}\frac{\partial}{\partial z_i}(z_j - \mu_B)^2 \\ &= \frac{1}{m}\sum_{j=1}^{m} 2(z_j - \mu_B)\, \frac{\partial (z_j - \mu_B)}{\partial z_i} \\ &= \frac{2}{m}\sum_{j=1}^{m}(z_j - \mu_B)\left(\frac{\partial z_j}{\partial z_i} - \frac{\partial \mu_B}{\partial z_i}\right).\end{aligned}$$

$$\frac{\partial z_j}{\partial z_i} = \delta_{ij},$$

Using the derivative of the Eq. (8),

$$\frac{\partial \mu_B}{\partial z_i} = \frac{\partial}{\partial z_i}\left(\frac{1}{m}\sum_{k=1}^{m} z_k\right) = \frac{1}{m}. \tag{46}$$

we obtain

$$\frac{\partial \sigma_B^2}{\partial z_i} = \frac{2}{m}\sum_{j=1}^{m}(z_j - \mu_B)\left(\delta_{ij} - \frac{1}{m}\right). \tag{47}$$

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

Step: split the sum. Write the sum as the part where $j = i$ plus the rest:

$$\begin{aligned} \frac{\partial \sigma_B^2}{\partial z_i} &= \frac{2}{m}\left[(z_i - \mu_B)\left(1 - \frac{1}{m}\right) + \sum_{j \neq i}(z_j - \mu_B)\left(0 - \frac{1}{m}\right)\right] \\ &= \frac{2}{m}\left[(z_i - \mu_B)\left(1 - \frac{1}{m}\right) - \frac{1}{m}\sum_{j \neq i}(z_j - \mu_B)\right]. \end{aligned} \tag{48}$$

Step: use the centering identity. Because μ_B is the batch mean,

$$\sum_{j=1}^{m}(z_j - \mu_B) = 0 \quad \Longrightarrow \quad \sum_{j \neq i}(z_j - \mu_B) = -(z_i - \mu_B). \tag{49}$$

substitute (49) into (48):

$$\begin{aligned} \frac{\partial \sigma_B^2}{\partial z_i} &= \frac{2}{m}\left[(z_i - \mu_B)\left(1 - \frac{1}{m}\right) - \frac{1}{m}\big(-(z_i - \mu_B)\big)\right] \\ &= \frac{2}{m}\left[(z_i - \mu_B)\left(1 - \frac{1}{m}\right) + \frac{1}{m}(z_i - \mu_B)\right] \\ &= \frac{2}{m}(z_i - \mu_B). \end{aligned} \tag{50}$$

Thus, we obtain the simplified expression

$$\frac{\partial \sigma_B^2}{\partial z_i} = \frac{2}{m}(z_i - \mu_B),$$

because the batch-centered term $\sum_j (z_j - \mu_B)$ vanishes. By substituting Eq. (50) in Eq. 45 and using $s = \sqrt{\sigma_B^2 + \varepsilon}$,

$$\frac{\partial s}{\partial z_i} = \frac{1}{2s}\frac{\partial \sigma_B^2}{\partial z_i} = \frac{1}{2s} \cdot \frac{2}{m}(z_i - \mu_B) = \frac{z_i - \mu_B}{ms} = \frac{\hat{z}_i}{m}. \tag{51}$$

By replacing $\frac{\partial s}{\partial z_i}$ and Eq. (46) in Eq. (44) we obtain:

$$\begin{aligned} \frac{\partial \hat{z}_k}{\partial z_i} &= \frac{1}{s}\left(\delta_{ik} - \frac{1}{m}\right) - \frac{z_k - \mu_B}{s^2} \cdot \frac{\hat{z}_i}{m} \\ &= \frac{1}{s}\left(\delta_{ik} - \frac{1}{m}\right) - \frac{\hat{z}_k}{s} \cdot \frac{\hat{z}_i}{m} \\ &= \frac{1}{s}\left(\delta_{ik} - \frac{1}{m} - \frac{\hat{z}_i \hat{z}_k}{m}\right). \end{aligned} \tag{52}$$

Equation (52) represents the BatchNorm-specific Jacobian entry, which is substituted into the main chain rule to derive the input gradient in Eq. (40).

E.2 Chain rule derivative replacement.

Combine (40) and (52):

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial z_i} &= \sum_{k=1}^{m} \underbrace{\frac{\partial \mathcal{L}}{\partial z'_k}}_{r_k/m} \underbrace{\frac{\partial z'_k}{\partial \hat{z}_k}}_{\gamma} \underbrace{\frac{\partial \hat{z}_k}{\partial z_i}}_{\frac{1}{s}\left(\delta_{ik} - \frac{1}{m} - \frac{\hat{z}_i \hat{z}_k}{m}\right)} \\ &= \frac{\gamma}{ms}\left[r_i - \frac{1}{m}\sum_{k=1}^{m} r_k - \hat{z}_i \frac{1}{m}\sum_{k=1}^{m} r_k \hat{z}_k\right]. \end{aligned} \tag{53}$$

Here, the term r_i arises from the δ_{ik} part of $\dfrac{\partial \hat{z}_k}{\partial z_i} = \dfrac{1}{s}\left(\delta_{ik} - \dfrac{1}{m} - \dfrac{\hat{z}_i \hat{z}_k}{m}\right)$: when $k = i$, $\delta_{ik} = 1$, so the sum

$$\sum_{k=1}^{m} \frac{r_k}{ms}\, \delta_{ik}\, \gamma$$

reduces to $\frac{\gamma}{ms}\, r_i$.

Define the batch means

$$\bar{r} := \frac{1}{m}\sum_{k=1}^{m} r_k, \qquad \overline{r\hat{z}} := \frac{1}{m}\sum_{k=1}^{m} r_k \hat{z}_k,$$

to obtain the compact form

$$\boxed{\frac{\partial \mathcal{L}}{\partial z_i} = \frac{\gamma}{ms}\Big(r_i - \bar{r} - \hat{z}_i\, \overline{r\hat{z}}\Big).} \tag{54}$$

For any upstream parameter w (in a learnable layer preceding Batch Normalization), let $b_i := \frac{\partial z_i}{\partial w}$ denote its local gradient contribution. Substituting (54), we obtain:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w} &= \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial z_i}\, b_i \\
&= \sum_{i=1}^{m} \frac{\gamma}{ms}\Big(r_i - \bar{r} - \hat{z}_i\, \overline{r\hat{z}}\Big)\, b_i \\
&= \frac{\gamma}{ms} \sum_{i=1}^{m} \underbrace{\Big(r_i - \bar{r} - \hat{z}_i\, \overline{r\hat{z}}\Big)}_{=:a_i}\, b_i.
\end{aligned} \tag{55}$$

Define the vectors

$$a := (a_1, \ldots, a_m), \qquad b := (b_1, \ldots, b_m).$$

Then (55) can be written as the inner product

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\gamma}{ms}\, \langle a, b \rangle.$$

By applying the Cauchy–Schwarz inequality to this inner product term (following the bounding approach of Munteanu et al. [83]) and using $\sum_i \|b_i\|^2 \le L^2$ (i.e., $\|b\|_2 \le L$), we obtain

$$\begin{aligned}
\Big\|\frac{\partial \mathcal{L}}{\partial w}\Big\| &= \frac{|\gamma|}{ms}\, |\langle a, b \rangle| \\
&\le \frac{|\gamma|}{ms}\, \|a\|_2\, \|b\|_2 \\
&\le \frac{|\gamma|}{ms}\, \|a\|_2\, L.
\end{aligned} \tag{56}$$

Aggregating over all core parameters θ_{core} and using a gradient-descent step with learning rate η, we obtain

$$\big\|\Delta\theta_{\text{core}}\big\|_2 \le \eta\, \frac{|\gamma|}{s}\, \frac{\|a\|_2}{m}\, L. \tag{57}$$

Here, L denotes an upper bound on the l_2 norm of the vector of local gradients $\{b_i\}_{i=1}^{m}$ for the entire core parameter vector θ_{core}.

By expressing $\|a\|_2$ in terms of variance and covariance, we can establish a direct comparison between (57) and Theorem 3.3. Because BN normalizes the *whole* batch, $\frac{1}{m}\sum_i \hat{z}_i = 0$ and $\frac{1}{m}\sum_i \hat{z}_i^2 = 1$ (subgroup means need not be zero).

E.3 Core Weight Update: Closed-Form Solution

Let $a_i = r_i - \bar{r} - \hat{z}_i\,\overline{r\hat{z}}$ denote the normalized residual term defined in Equation (55). A direct expansion of its squared norm yields

$$\begin{aligned}\|a\|_2^2 &= \sum_{i=1}^{m}(r_i-\bar{r})^2 + \overline{r\hat{z}}^{\,2}\sum_{i=1}^{m}\hat{z}_i^2 - 2\,\overline{r\hat{z}}\sum_{i=1}^{m}(r_i-\bar{r})\hat{z}_i \\ &= m\Big(\mathrm{Var}(r) + \overline{r\hat{z}}^{\,2} - 2\,\overline{r\hat{z}}\,\mathrm{Cov}(r,\hat{z})\Big). \end{aligned} \tag{58}$$

$$\begin{aligned}\overline{r\hat{z}} &= \frac{1}{m}\sum_{i=1}^{m} r_i\hat{z}_i \\ &= \frac{1}{m}\sum_{i=1}^{m}(r_i-\bar{r})\hat{z}_i + \bar{r}\Big(\frac{1}{m}\sum_{i=1}^{m}\hat{z}_i\Big) \\ &= \mathrm{Cov}(r,\hat{z}), \end{aligned} \tag{59}$$

where $\frac{1}{m}\sum_{i=1}^{m}\hat{z}_i = 0$.

$$\begin{aligned}\|a\|_2^2 &= m\Big(\mathrm{Var}(r) + \mathrm{Cov}(r,\hat{z})^2 - 2\,\mathrm{Cov}(r,\hat{z})^2\Big) \\ &= m\Big(\mathrm{Var}(r) - \mathrm{Cov}(r,\hat{z})^2\Big). \end{aligned} \tag{60}$$

$$\|a\|_2 = \sqrt{m}\,\sqrt{\mathrm{Var}(r) - \mathrm{Cov}(r,\hat{z})^2} = \sqrt{m\,\mathrm{Var}(r)}\,\sqrt{1-\kappa^2}, \quad \kappa := \mathrm{Corr}(r,\hat{z}). \tag{61}$$

Derivation of the correlation form. Let $\kappa := \mathrm{Corr}(r,\hat{z}) = \dfrac{\mathrm{Cov}(r,\hat{z})}{\sqrt{\mathrm{Var}(r)}\sqrt{\mathrm{Var}(\hat{z})}}$, which is defined whenever $\mathrm{Var}(r) > 0$. Since BN normalizes the batch, $\mathrm{Var}(\hat{z}) = \frac{1}{m}\sum_i \hat{z}_i^2 = 1$, so

$$\mathrm{Cov}(r,\hat{z}) = \kappa\sqrt{\mathrm{Var}(r)}, \qquad \mathrm{Cov}(r,\hat{z})^2 = \kappa^2\,\mathrm{Var}(r). \tag{62}$$

Substituting into Eq. (61),

$$\begin{aligned}\|a\|_2 &= \sqrt{m}\,\sqrt{\mathrm{Var}(r) - \kappa^2\,\mathrm{Var}(r)} = \sqrt{m}\,\sqrt{\mathrm{Var}(r)\,(1-\kappa^2)} \\ &= \sqrt{m\,\mathrm{Var}(r)}\,\sqrt{1-\kappa^2}. \end{aligned} \tag{63}$$

The last step splits the square root of a product of two nonnegative factors. The radicand is nonnegative because $|\kappa| \le 1$ by Cauchy–Schwarz, which is consistent with $\|a\|_2^2 = m\big(\mathrm{Var}(r) - \mathrm{Cov}(r,\hat{z})^2\big) \ge 0$ in Eq. (60).

When the batch is drawn from an adversarial mixture $B \sim (1-p)\,\mathcal{D}_0 + p\,\mathcal{D}_t$, the variance and covariance evolve according to the laws of total variance and covariance:

$$\begin{aligned}\mathrm{Var}(r) &= (1-p)\,\mathrm{Var}_{\mathcal{D}_0}(r) + p\,\mathrm{Var}_{\mathcal{D}_t}(r) \\ &\quad + \; p(1-p)\big(\bar{r}_{\mathcal{D}_t} - \bar{r}_{\mathcal{D}_0}\big)^2, \end{aligned} \tag{64}$$

$$\begin{aligned}\mathrm{Cov}(r,\hat{z}) &= (1-p)\,\mathrm{Cov}_{\mathcal{D}_0}(r,\hat{z}) + p\,\mathrm{Cov}_{\mathcal{D}_t}(r,\hat{z}) \\ &\quad + \; p(1-p)\big(\bar{r}_{\mathcal{D}_t} - \bar{r}_{\mathcal{D}_0}\big)\big(\bar{\hat{z}}_{\mathcal{D}_t} - \bar{\hat{z}}_{\mathcal{D}_0}\big). \end{aligned} \tag{65}$$

Substituting (61) into the core update bound (57) yields the final upper bound on the parameter update magnitude:

$$\boxed{\|\Delta\theta_{\text{core}}\|_2 \le \eta\,\frac{|\gamma|\,L}{s\sqrt{m}}\,\sqrt{\mathrm{Var}(r) - \mathrm{Cov}(r,\hat{z})^2} = \eta\,\frac{|\gamma|\,L}{s\sqrt{m}}\,\sqrt{\mathrm{Var}(r)}\,\sqrt{1-\kappa^2}} \tag{66}$$

This expression shows that the core update magnitude depends on the part of the residual variance orthogonal to the normalized activation $\hat{z}$, namely $\mathrm{Var}(r)(1-\kappa^2)$. Hence, a large $\mathrm{Cov}(r,\hat{z})$ lowers the bound only through the correlation κ; if $\mathrm{Var}(r)$ also grows, the bound can grow unless κ is close to one. For *adversarially triggered* data, Prop. 3.6 gives κ close to one when the within-group scatter ratios $t_r, t_{\hat{z}}$ are small, so $\sqrt{1-\kappa^2}$ is small. The BN step, by contrast, is at least $\eta\kappa\sqrt{\mathrm{Var}(r)}$, which does not carry this factor. For *clean* data, both $\mathrm{Var}(r)$ and $\mathrm{Cov}(r,\hat{z})$ are small because the per-example gradients are unstructured; hence the difference under the square root in (66) becomes negligible, and the resulting core update is tiny.

In contrast, for *adversarially triggered* data, the residuals r become highly correlated with the normalized activations $\hat{z}$. Although, $\mathrm{Var}(r)$ grows due to the backdoor signal, $\mathrm{Cov}(r,\hat{z})$ grows proportionally, and the term $\mathrm{Var}(r) - \mathrm{Cov}(r,\hat{z})^2$ remains small. Consequently, the factor on the right-hand side of (66) stays bounded, and the core-weight update remains modest compared to the BN update, which scales as $\sqrt{\bar{r}^{\,2} + \mathrm{Cov}(r,\hat{z})^2}$. This contrast is consistent with the BN parameters receiving most of the adversarial gradient while the core weights are less affected. Our empirical study, illustrated in Figure 2c, reaches the same conclusion: under adversarial attacks, the gradient updates to the core network weights are much smaller than the updates of the BN parameters.

F Bound on κ and Effect on Parameter Updates

To formalize the bounds, let us first explicitly define the mixture statistics for the backward gradient r and the forward normalized activation $\hat{z}$. Let Δ_r and $\Delta_{\hat{z}}$ denote the trigger-induced mean shifts between the clean and poisoned groups, which generate the between-group variances $A_r = p(1-p)\Delta_r^2$ and $A_{\hat{z}} = p(1-p)\Delta_{\hat{z}}^2$. We define ε_r and $\varepsilon_{\hat{z}}$ as the average within-group variances (representing the natural noise inherent to the clean and poisoned distributions), and C_w as the within-group covariance. The total alignment is constrained by the within-to-between scatter ratios, defined as $t_r = \varepsilon_r/A_r$ and $t_{\hat{z}} = \varepsilon_{\hat{z}}/A_{\hat{z}}$.

To bound the alignment, write $S = \sqrt{A_r A_{\hat{z}}}$ to denote the geometric mean of the between-group variances. This term is strictly positive because Prop. 3.6 assumes a coherent trigger where $\Delta_r, \Delta_{\hat{z}} \neq 0$. By definition, the within-group variances can be rewritten as $\varepsilon_r = t_r A_r$ and $\varepsilon_{\hat{z}} = t_{\hat{z}} A_{\hat{z}}$. By the law of total covariance in Eq. (38) and the reverse triangle inequality, the absolute total covariance is bounded below by:

$$|\mathrm{Cov}(r,\hat{z})| \geq S - |C_w| \geq S\left(1 - \sqrt{t_r t_{\hat{z}}}\right)$$

which is nonnegative under the hypothesis of Prop. 3.6 ($t_r t_{\hat{z}} \leq 1$). Furthermore, the product of the total variances expands to $\mathrm{Var}(r)\mathrm{Var}(\hat{z}) = S^2(1+t_r)(1+t_{\hat{z}})$. Dividing the covariance bound by $\sqrt{\mathrm{Var}(r)\mathrm{Var}(\hat{z})}$ gives the stated lower bound on the Pearson correlation magnitude $|\kappa| = |\mathrm{Corr}(r,\hat{z})|$.

By definition, a Pearson correlation, κ is fundamentally upper-bounded by $|\kappa| \leq 1$ via the Cauchy-Schwarz inequality. The derivation above provides a strict *lower* bound. As the trigger coherence suppresses the relative within-group scatter (forcing $t_r \to 0$ and $t_{\hat{z}} \to 0$), this lower bound is squeezed tightly towards 1, which physically forces the squared correlation $\kappa^2 \to 1$.

To see the direct physical effect on the network, we substitute the correlation identity $\mathrm{Cov}(r,\hat{z})^2 = \kappa^2\mathrm{Var}(r)$ into the parameter update equations from Prop. 3.6. For a given learning rate η and structural constant $C := \gamma^2 L^2/(s^2 m)$, the parameter update magnitudes become:

$$\|\Delta\theta_{\mathrm{BN}}\|_2 = \eta\sqrt{\bar{r}^{\,2} + \kappa^2\mathrm{Var}(r)}\,,$$
$$\|\Delta\theta_{\mathrm{core}}\|_2 \leq \eta\sqrt{C\mathrm{Var}(r)(1-\kappa^2)}\,.$$

This substitution reveals the fundamental routing mechanism of the backdoor. The BN parameter update magnitude directly absorbs the inflated gradient variance scaled by κ^2. Conversely, as our lower bound forces $\kappa^2 \to 1$, the attenuation factor $(1-\kappa^2)$ for the core weights collapses toward zero. This mathematical squeeze starves the preceding convolutional core weights of the adversarial gradient, ensuring they are updated predominantly by the clean task residual, thereby preserving clean accuracy.

G Defense

G.1 FlashBN

Let a BN layer hold parameters (γ, β) and running statistics (μ, σ^2), where any of these may have been corrupted by an adversarial mixture present in the training batches. Given an inference batch $B = \{z_1, \ldots, z_{|B|}\}$ of pre-BN activations at that layer, we define the *FlashBN* operator as

$$\mathcal{R}_{\mathrm{BN}}(z; B) \;=\; \frac{z - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \qquad \gamma \equiv 1, \;\; \beta \equiv 0, \tag{67}$$

where μ_B and σ_B^2 are the per-channel mean and variance computed *on the current inference batch only*, and $\varepsilon > 0$ is the standard numerical floor inherited from the original BN. The affine transform is disabled by hard-setting $\gamma = 1$ and $\beta = 0$; the stored values are not consulted.

Proposition G.1 (Statistic and affine independence)**.** *For any inference batch B with $|B| \geq 2$ and any pre-BN activation z, the output $\mathcal{R}_{\mathrm{BN}}(z; B)$ is a function of z and B alone; it is independent of the stored (μ, σ^2) and (γ, β).*

The proof is immediate: neither (μ, σ^2) nor (γ, β) appears on the right-hand side of Eq. (67). The consequence, however, is the substantive claim: any backdoor channel realized through BN state—whether through stale running statistics that misalign clean inputs onto a triggered manifold, or through an affine pair that scales and shifts a trigger-aligned direction—is removed at inference by construction. This does not imply that the backdoor is removed: what remains active is the *core convolutional or linear weight tensor* upstream of BN, together with the moments of the inference batch itself.

It is tempting to neutralize only one of the two BN sub-mechanisms (e.g., resetting $(\gamma, \beta) \to (1, 0)$ while retaining the stored (μ, σ^2), or vice versa). We expect, but did not ablate, that this is insufficient under what we term the *adversarially coherent* regime, in which training under a poisoned mixture induces a large mean residual $\bar{r} = \mathbb{E}[z - \mu_{\text{clean}}]$ and a large covariance $\mathrm{Cov}(r, \hat{z})$ between the residual and the post-normalization activation $\hat{z}$. In this regime, backdoors can be planted purely through (μ, σ^2) drift even when (γ, β) remain near their clean-trained values, and conversely (γ, β) can be co-adapted with the trigger so that a near-clean (μ, σ^2) is amplified along the trigger direction. The coupling term $\mathrm{Cov}(r, \hat{z})$ is precisely what makes the two channels jointly exploitable: a large residual that is also aligned with the post-norm representation propagates into the classifier with multiplicative gain γ and additive bias β. FlashBN eliminates both terms in a single operator, so the two attack surfaces collapse simultaneously.

G.2 ReBN

Let $\mathcal{B}$ denote the set of BN layers in a (possibly backdoored) model θ^*, with per-layer affine parameters $(\gamma_\ell, \beta_\ell) \in \mathbb{R}^{C_\ell} \times \mathbb{R}^{C_\ell}$ and running statistics $(\mu_\ell^*, \sigma_\ell^{2,*})$ for $\ell \in \mathcal{B}$. ReBN proceeds in three steps: (i) reset and recalibrate BN at inference, (ii) score BN-affine coordinates by their Fisher sensitivity to a rectification objective, and (iii) prune and fine-tune on a small clean set $\mathcal{D}_{\mathrm{ft}}$.

Inference-time recalibration. Given an inference batch B with per-channel batch statistics $(\mu_{B,\ell}, \sigma_{B,\ell}^2)$, FlashBN replaces the BN forward pass at every $\ell \in \mathcal{B}$ by

$$\mathcal{R}_{\mathrm{BN}}(z; B) \;=\; \frac{z - \mu_{B,\ell}}{\sqrt{\sigma_{B,\ell}^2 + \varepsilon}}, \qquad \gamma \equiv 1, \;\; \beta \equiv 0, \tag{68}$$

discarding both the stored statistics and the learned affine. Hence $\mathcal{R}_{\mathrm{BN}}(z; B)$ depends only on z and B, not on $(\mu_\ell^*, \sigma_\ell^{2,*})$ or $(\gamma_\ell, \beta_\ell)$.

Rectification objective. Let θ' be the model obtained from θ^* by resetting the BN affine pair to $(\gamma_0, \beta_0) = (1, 0)$. For each $\ell \in \mathcal{B}$, let $(\hat{\mu}_\ell, \hat{\sigma}_\ell^2)$ be the per-channel batch statistics of θ' on the current mini-batch from $\mathcal{D}_{\mathrm{ft}}$. We measure their drift toward the backdoored reference $(\mu_\ell^*, \sigma_\ell^{2,*})$ via

$$\mathcal{L}_{\mathrm{BN}} \;=\; \frac{1}{|\mathcal{B}|} \sum_{\ell \in \mathcal{B}} \Big(\big\| \hat{\mu}_\ell - \mu_\ell^* \big\|^2 + \lambda \big\| \hat{\sigma}_\ell^2 - \sigma_\ell^{2,*} \big\|^2 \Big), \quad \mathcal{L}_{\mathrm{rectify}} \;=\; \mathcal{L}_{\mathrm{CE}}(x, y) + \log \mathcal{L}_{\mathrm{BN}}, \tag{69}$$

with $\lambda = 0.05$.

Fisher mask, pruning, and fine-tuning. Let $\Theta_{\mathrm{BN}} \subset \Theta$ be the BN-affine entries $\{\gamma_{\ell,c}, \beta_{\ell,c}\}$. We score each $\theta_i \in \Theta_{\mathrm{BN}}$ by the empirical diagonal Fisher of $\mathcal{L}_{\mathrm{rectify}}$ and threshold the top-K scores to obtain a binary mask:

$$F_{\mathrm{BN}}(\theta_i) \;=\; \frac{1}{|\mathcal{D}_{\mathrm{ft}}|} \sum_{(x,y)\in\mathcal{D}_{\mathrm{ft}}} \big\|\nabla_{\theta_i} \mathcal{L}_{\mathrm{rectify}}(x,y)\big\|^2, \qquad M_j \;=\; \mathbf{1}\{F_{\mathrm{BN}}(\theta_j) < \tau\}, \tag{70}$$

where τ is the K-th largest value of $\{F_{\mathrm{BN}}(\theta_j)\}$. Pruned coordinates are softened with i.i.d. noise $\Xi \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ and the model is fine-tuned with masked gradients on $\mathcal{D}_{\mathrm{ft}}$:

$$\theta_u \;=\; M\odot\Theta + (\mathbf{1}-M)\odot\Xi, \qquad \hat{\theta} \;=\; \arg\min_{\theta}\; \mathbb{E}_{(x,y)\in\mathcal{D}_{\mathrm{ft}}}\, \mathcal{L}_{\mathrm{CE}}\big(f(x; M\odot\theta_u), y\big), \quad \nabla_\theta \leftarrow \nabla_\theta \mathcal{L}_{\mathrm{CE}} \odot M. \tag{71}$$

The mask zeroes the gradients of the pruned BN-affine coordinates, blocking drift back toward the backdoored statistics, while the remaining BN-affine coordinates are fine-tuned to preserve clean accuracy; core weights are not updated.

G.3 Proof of Theorem 3.8 (Decision Agreement After Recalibration)

Proof. Let $f^{\mathcal{R}}_{\theta_{\mathrm{core}}}(\tilde{x})$ denote the logit vector produced on triggered input $\tilde{x}$ when combining core parameters θ_{core} with the clean-recomputed BN operator $\mathcal{R}_{\mathrm{BN}}$. Let θ^*_{core} and θ^0_{core} denote the poisoned and clean core parameters, respectively, with $\|\theta^*_{\mathrm{core}} - \theta^0_{\mathrm{core}}\|_2 \le \delta$ by Assumption (A1). For the adversarial target class τ and the true (or runner-up clean) class y, define the target-class margin on the clean core as $M(\theta^0_{\mathrm{core}}) := \big[f^{\mathcal{R}}_{\theta^0_{\mathrm{core}}}(\tilde{x})\big]_\tau - \big[f^{\mathcal{R}}_{\theta^0_{\mathrm{core}}}(\tilde{x})\big]_y \le -\gamma_0$ by Assumption (A3). By Assumption (A2), each logit coordinate is L_θ-Lipschitz with respect to θ_{core}, so

$$\big|\big[f^{\mathcal{R}}_{\theta^*_{\mathrm{core}}}(\tilde{x})\big]_c - \big[f^{\mathcal{R}}_{\theta^0_{\mathrm{core}}}(\tilde{x})\big]_c\big| \;\le\; L_\theta \|\theta^*_{\mathrm{core}} - \theta^0_{\mathrm{core}}\|_2 \;\le\; L_\theta\delta, \qquad \forall c \in \{y, \tau\}.$$

Applying the triangle inequality to the target-class margin of the defended poisoned model yields

$$M(\theta^*_{\mathrm{core}}) = \big[f^{\mathcal{R}}_{\theta^*_{\mathrm{core}}}(\tilde{x})\big]_\tau - \big[f^{\mathcal{R}}_{\theta^*_{\mathrm{core}}}(\tilde{x})\big]_y \;\le\; M(\theta^0_{\mathrm{core}}) + 2L_\theta\delta \;\le\; -\gamma_0 + 2L_\theta\delta.$$

Hence, whenever $\delta < \gamma_0/(2L_\theta)$, we have $M(\theta^*_{\mathrm{core}}) < 0$, guaranteeing that the defended model does not predict the adversarial target class τ on $\tilde{x}$. □

G.4 Proof of Theorem 3.9

Let θ be the core and ϕ the BN parameters (affine parameters and running statistics), with clean values (θ_c, ϕ_c) and poisoned values (θ_p, ϕ_p), so that $\delta = \|\theta_p - \theta_c\|$ and $\delta_{\mathrm{BN}} = \|\phi_p - \phi_c\|$, and let $M(\theta, \phi)$ be the target-class margin on the triggered input $\tilde{x}$. The assumptions are that M is L_θ-Lipschitz in θ and L_ϕ-Lipschitz in ϕ, that $M(\theta_c, \phi_c) \le -\gamma_0$, and that $M(\theta_p, \phi_p) \ge \nu > 0$; none of them concerns BN or where the backdoor is stored. The theorem bounds the sum of the two changes, not which one an attacker uses.

Proof. Insert the hybrid (θ_c, ϕ_p) and telescope:

$$\begin{aligned}\nu \le M(\theta_p, \phi_p) &= M(\theta_c, \phi_c) + \big[M(\theta_p, \phi_p) - M(\theta_c, \phi_p)\big] + \big[M(\theta_c, \phi_p) - M(\theta_c, \phi_c)\big] \\ &\le -\gamma_0 + L_\theta\delta + L_\phi\delta_{\mathrm{BN}},\end{aligned}$$

using Lipschitz continuity in θ at fixed ϕ_p and in ϕ at fixed θ_c. If $L_\theta\delta \le \gamma_0$, then $L_\phi\delta_{\mathrm{BN}} \ge \nu > 0$; if $\delta_{\mathrm{BN}} = 0$, then $L_\theta\delta \ge \nu + \gamma_0$. □

H Tiny ImageNet Setting

Table 3 uses Tiny ImageNet (200 classes) with a 10% poison ratio and the publicly available BackdoorBench attack models [36] (PreAct ResNet-18). The nine attacks and the baseline defenses are those available in BackdoorBench and differ from the CIFAR-10 sets in Table 1. FlashBN fails on LF (85.2% ASR), raises the ASR on SIG (44.9% to 62.0%), and loses clean accuracy on several attacks (e.g., Blended, 56.0% to 34.0%).

Table 3: Tiny ImageNet (200 classes, 10% poison ratio). Results are reported as clean accuracy (Acc.) and attack success rate (ASR), both in %. Lower ASR is better. Cell colors: within each row, the best/second-best/third-best (lowest) ASR among the defenses are highlighted. FlashBN is training-free; ReBN is BN-only few-shot fine-tuning on a small clean set. Attacks and defenses follow BackdoorBench [36].

	Orig.		FT [36]		FP [84]		NAD [7]		NC [10]		ANP [85]		ABL [47]		CLP [86]		DDE [52]		i-BAU [45]		ReBN (Ours)		FlashBN (Ours)	
Attacks	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR	Acc.	ASR
BadNet [25]	56.2	100.0	55.2	0.1	51.7	100.0	46.4	0.3	51.5	0.1	50.5	7.7	49.7	**0.0**	55.9	100.0	54.1	**0.0**	51.5	97.4	56.1	0.3	54.2	0.1
Blended [87]	56.0	99.7	55.0	97.7	51.9	95.9	46.9	95.0	52.5	93.2	55.0	84.6	44.6	0.1	55.7	99.7	55.0	76.6	53.0	91.9	54.9	0.1	34.0	**0.0**
LF [88]	56.0	98.6	54.8	94.9	51.4	95.2	45.5	50.5	53.0	85.6	54.7	95.4	48.4	**0.3**	55.6	98.5	55.1	85.9	51.1	85.3	50.8	1.5	46.2	85.2
SSBA [27]	55.2	97.7	54.8	91.6	50.5	88.9	45.3	57.3	52.5	53.5	52.8	91.4	48.0	0.4	55.2	97.7	54.2	57.7	49.9	81.9	40.4	0.9	36.4	**0.0**
WaNet [28]	56.8	99.5	56.7	**0.2**	53.8	3.9	47.0	0.4	53.3	**0.2**	53.9	0.8	49.8	0.8	56.2	98.5	56.2	**0.2**	53.7	75.2	55.4	0.3	51.4	0.6
InputAware [89]	57.5	98.8	57.5	1.6	55.3	62.9	47.9	1.9	56.2	0.1	53.2	0.2	50.9	**0.0**	57.8	99.6	57.5	0.2	52.5	73.0	53.9	0.1	54.6	0.1
BPP [67]	58.1	100.0	57.4	0.4	50.9	0.5	45.7	0.4	53.6	0.2	52.4	1.7	47.4	**0.0**	57.0	0.3	56.9	0.1	56.1	99.7	56.3	0.1	51.2	0.2
TrojanNN [26]	55.9	100.0	55.4	0.5	50.3	13.6	48.5	0.8	52.7	0.1	50.4	1.4	44.9	**0.0**	55.9	8.4	55.0	0.1	52.6	98.5	55.1	1.0	51.6	0.7
SIG [90]	57.0	44.9	56.5	49.2	55.2	50.0	48.0	31.9	53.8	47.7	53.2	39.8	52.3	9.1	57.1	44.9	55.7	42.4	54.5	38.5	56.5	**0.02**	49.0	62.0
Average	56.52	93.24	55.92	37.36	52.33	56.77	46.80	26.50	53.21	31.19	52.90	35.89	48.40	1.19	56.27	71.96	55.52	29.24	52.77	82.38	53.27	**0.47**	47.62	16.54

I Discussion of Additional Benchmark Results

We further present the performance of our method in Tables 4 and 5 under additional attack settings, using a poisoning rate of 10%. Our approach achieves competitive results while FlashBN remains completely training-free. Although ANP attains strong ASR performance, it requires additional fine-tuning and substantially degrades clean accuracy, whereas our method maintains excellent clean accuracy. Furthermore, because our approach is orthogonal to existing defense techniques, it may be combined with them, which we have not evaluated. Defense times in Table 1 are measured on *RTX3090*

Table 4: Comparison of defenses against backdoor attacks (Part A). **CA**: Clean Accuracy (%); **ASR**: Attack Success Rate (%). **Orange** = lowest ASR (best), **Yellow** = second-lowest ASR. Training-free, our method is competitive and orthogonal to existing defenses.

	Original		FT		FP [84]		NAD [7]	
Attack	CA	ASR	CA	ASR	CA	ASR	CA	ASR
Blended [87]	93.47	99.92	92.78	96.11	93.17	99.26	92.17	97.69
LF [88]	93.19	99.28	92.37	78.44	92.90	98.97	92.37	47.83
SSBA [27]	92.88	97.86	92.14	74.79	92.54	83.50	91.91	77.40
InputAware [89]	90.67	98.26	93.12	1.72	91.74	**0.04**	93.18	1.68
BPP [67]	90.69	99.78	93.25	2.54	92.70	18.39	92.81	2.46
TrojanNN [26]	93.42	100.00	92.42	5.99	92.47	71.07	91.88	3.73
Mean	92.39	99.18	92.68	43.27	92.59	61.87	92.39	38.46

Table 5: Comparison of defenses against backdoor attacks (Part B). Continued from Table 4.

	ANP [85]		NC [10]		I-BAU [45]		Ours	
Attack	CA	ASR	CA	ASR	CA	ASR	CA	ASR
Blended [87]	86.58	**2.19**	93.47	99.92	87.00	50.53	93.47	27.60
LF [88]	85.15	**0.60**	91.62	1.41	84.36	44.96	93.19	19.49
SSBA [27]	86.54	**0.39**	90.99	0.58	87.67	3.97	92.88	23.51
InputAware [89]	90.36	1.64	92.61	0.76	89.17	27.08	90.67	7.07
BPP [67]	81.78	**0.24**	92.68	2.84	90.99	6.24	90.69	11.44
TrojanNN [26]	86.87	**0.20**	91.76	8.22	90.37	2.91	93.42	14.31
Mean	86.21	**0.88**	92.19	18.96	88.26	22.61	92.39	**17.24**

Table 6: Comparison across eight attacks (Part A). Acc / ASR / Time (s). Lower ASR is better. All attacks trained for 50 epochs on the pretrained model. Cell colors: best/2nd/3rd ASR and Time among defenses (computed jointly across Part A and Part B).

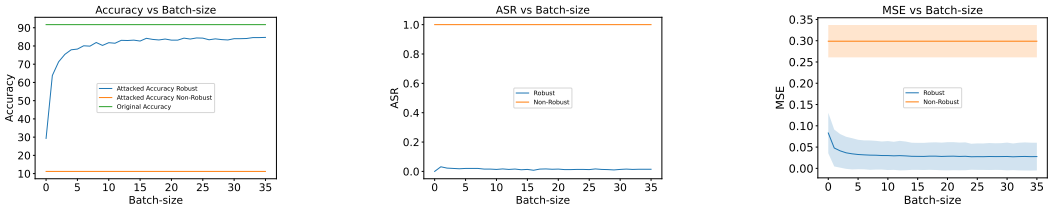
	Orig.		NPD [62]			RNP [55]			NAB [63]			UNIT [46]		
Attacks	Acc.	ASR	Acc.	ASR	T	Acc.	ASR	T	Acc.	ASR	T	Acc.	ASR	T
ReAttack [65]	81.54	84.58	81.37	1.16	396.0	78.53	1.69	323.8	81.13	1.56	770.6	81.00	5.94	379.8
CTRL [6]	91.53	72.24	84.36	61.91	373.3	89.58	1.39	268.6	93.21	1.32	677.2	90.82	10.38	268.8
FTrojan [66]	93.88	100.00	92.08	78.79	1340.0	91.90	100.00	359.1	92.43	3.53	552.1	91.44	6.75	241.1
BPP [67]	90.10	99.83	87.89	36.67	1006.7	79.77	37.70	448.0	90.20	3.49	621.2	90.00	5.80	247.8
PoisonInk [68]	90.25	99.80	88.52	99.79	414.1	10.00	100.00	561.8	90.00	3.00	561.7	87.14	5.23	305.3
Diffusion [69, 70]	92.22	94.48	82.09	6.73	358.4	90.01	3.97	219.4	89.39	1.29	561.8	86.53	10.96	293.3
Adaptive Blend [71]	93.85	99.63	91.91	30.97	385.1	86.55	34.91	341.0	93.19	**3.19**	564.5	91.54	10.40	272.7
COMBAT [72]	89.16	77.37	88.29	66.37	958.9	65.48	99.23	503.6	90.88	1.62	548.2	88.92	8.99	318.7
Average	90.32	90.99	87.06	47.80	654.1	73.98	47.36	378.0	90.05	2.38	607.2	88.42	8.06	290.9

Table 7: Comparison across eight attacks (Part B). Continued from Table 6. **ReBN**: noise fine-tuning BN only. **FlashBN**: inference-time BN sanitization (no retraining).

	FT-SAM [64]			BNP [52]			EP [52]			ReBN (Ours)			FlashBN (Ours)		
Attacks	Acc.	ASR	T	Acc.	ASR	T	Acc.	ASR	T	Acc.	ASR	T	Acc.	ASR	T
ReAttack [65]	80.83	0.36	754.0	81.37	94.70	143.4	76.54	9.94	210.2	88.98	**0.07**	525.9	81.71	4.91	**26.8**
CTRL [6]	89.41	0.08	666.5	84.35	**0.06**	137.3	86.44	11.14	235.6	93.45	2.16	526.6	91.37	4.97	**20.6**
FTrojan [66]	90.36	4.65	1100.0	92.08	**0.62**	143.0	92.12	52.31	243.5	93.38	4.59	526.3	91.05	11.71	**26.7**
BPP [67]	87.36	6.06	997.5	87.89	80.86	134.0	84.44	72.92	196.1	91.10	**1.35**	524.9	91.10	4.39	**21.5**
PoisonInk [68]	89.36	**0.35**	958.6	88.52	99.79	143.1	87.69	1.27	210.2	89.46	2.23	530.1	90.74	3.99	**20.7**
Diffusion	89.86	0.65	952.4	82.09	92.26	134.0	90.64	1.39	197.1	92.56	**0.07**	526.3	87.61	27.19	**19.6**
Adaptive Blend [71]	92.91	7.70	1016.6	91.91	98.17	136.5	90.62	98.68	268.5	93.56	5.80	527.4	91.09	13.12	**22.8**
COMBAT [72]	87.35	**0.55**	1505.4	88.29	86.89	133.7	87.61	79.01	193.4	87.12	1.84	524.0	88.82	7.52	**21.0**
Average	88.43	2.55	993.9	87.06	69.17	138.1	87.01	40.83	219.3	91.20	**2.26**	526.4	89.19	9.72	**22.5**

Table 8: Variability of the defense over 5 random seeds (mean $\pm$ standard deviation). The poisoned checkpoints are fixed, so the table reports the variance of the defense (FlashBN, ReBN), not of the attack. The attacks and checkpoints are those of Table 6. Acc and ASR in %. The last row is the unweighted mean over the eight attacks.

	ReBN		**FlashBN**	
Attack	Acc ↑	ASR ↓	Acc ↑	ASR ↓
Adaptive Blend	92.64 ± 1.49	4.60 ± 0.95	89.04 ± 0.02	13.42 ± 0.10
BPP	89.13 ± 0.85	1.97 ± 0.34	90.85 ± 0.03	4.17 ± 0.04
COMBAT	87.27 ± 1.21	1.42 ± 0.58	88.01 ± 0.05	7.45 ± 0.03
CTRL	92.64 ± 1.13	2.10 ± 1.53	89.31 ± 0.58	4.53 ± 0.06
Diffusion	91.45 ± 1.12	0.21 ± 0.15	87.81 ± 0.02	28.62 ± 0.16
FTrojan	92.52 ± 1.70	4.12 ± 0.37	89.64 ± 0.51	11.49 ± 0.04
PoisonInk	89.93 ± 1.80	2.81 ± 0.58	90.39 ± 0.18	3.64 ± 0.14
ReAttack	88.26 ± 1.94	0.06 ± 0.02	80.44 ± 0.07	5.01 ± 0.01
Average	90.48	2.16	88.19	9.79



(a) Correlation between accuracy and batch size.

(b) ASR vs batch size.

(c) MSE vs batch size.

Figure 4: Ablation across batch size: (a) accuracy, (b) ASR, and (c) MSE show consistent stabilization when the batch size exceeds 5. However, to gain cleaning accuracy further, one needs to increase the batch size more.

J Protective Role of Batch Normalization: Core vs. BN Update Dynamics

We recall here the two key update expressions for clarity. For the core weights, substituting (61) into the per-layer bound (57) yields

$$\boxed{\left\|\Delta\theta_{\text{core}}\right\|_2 \;\le\; \eta\,\frac{|\gamma|\,L}{s\sqrt{m}}\,\sqrt{\text{Var}(r)\;-\;\text{Cov}(r,\hat{z})^2}\,,} \tag{72}$$

where L and s are layer-dependent Lipschitz and scaling constants. In contrast, the BatchNorm (BN) affine parameters have an exact update magnitude.

$$\boxed{\left\|\Delta\theta_{\text{BN}}\right\|_2 \;=\; \eta\,\sqrt{\bar{r}^{\,2}+\text{Cov}(r,\hat{z})^2}\,.} \tag{73}$$

Clean data. With clean batches only, the between-group term of Eq. (38) is absent; Figure 2a shows BN-parameter gradients that are lower than under triggered data (Figure 2c).

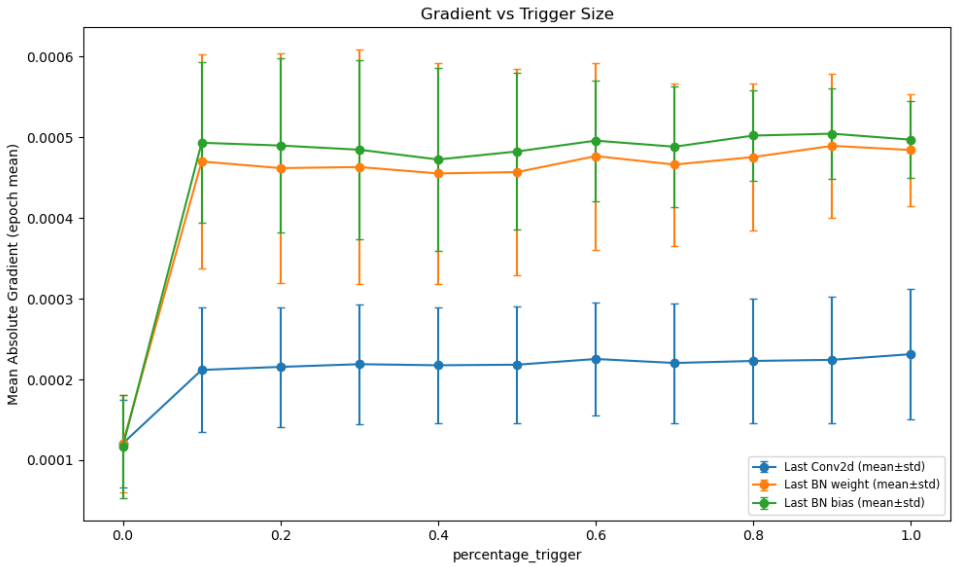


Figure 5: Change of weight per neuron with respect to poison rate.

Table 9: Clean accuracy (Clean), defended accuracy after BN recalibration (Def), and attack success rate before and after defense (ASR_{atk}, ASR_{def}; lower is better) under a standard targeted BadNets-style backdoor attack [25]. The attacked models were trained from scratch, rather than fine-tuned from a pretrained model. ASR is reported as a fraction in $[0, 1]$.

Dataset	Model	Clean	Def	ASR_{atk}	ASR_{def}
CIFAR-10	MobileNetV3-S	80.30	68.00	**0.99**	0.01
	ResNet20	**91.58**	**90.06**	1.00	0.02
	ResNet20 (No-BN)	88.10	10.28	1.00	1.00
	VGG13-BN	87.78	85.68	1.00	0.25
	ViT-B/16	76.80	10.40	1.00	1.00
	WideResNet-28-10	86.10	86.70	1.00	**0.00**
GTSRB	MobileNetV3-S	93.80	76.62	**0.73**	**0.00**
	ResNet20	**96.18**	**96.22**	1.00	**0.00**
	ResNet20 (No-BN)	36.21	1.26	1.00	1.00
	VGG13-BN	93.70	94.16	1.00	0.20
	ViT-B/16	94.12	4.12	0.90	0.90
	WideResNet-28-10	89.78	93.32	1.00	**0.00**
CIFAR-100	MobileNetV3-S	57.27	54.27	1.00	**0.00**
	ResNet20	41.14	39.37	1.00	0.01
	ResNet20 (No-BN)	32.18	0.90	1.00	1.00
	VGG13-BN	61.80	56.16	1.00	0.01
	ViT-B/16	60.30	1.14	**0.90**	0.90
	WideResNet-28-10	**78.20**	**74.10**	1.00	0.01

Adversarially triggered data. In a mixed batch, $\text{Cov}(r, \hat{z})$ contains the between-group term $p(1-p)\Delta_r\Delta_{\hat{z}}$ of Eq. (38), non-zero because the groups differ in mean (Lemma 3.4). When the within-group scatter is small relative to the gap, Prop. 3.6 gives $\kappa^2 \to 1$, so $\text{Var}(r) - \text{Cov}(r, \hat{z})^2$ and the core-update bound (66) are small, while the BN update norm grows with $\text{Cov}(r, \hat{z})$.

Role of Batch Normalization. These two expressions highlight the protective role of BN layers. The BN affine parameters absorb the coherent adversarial gradient, while the core network weights are updated only through the residual variance that is orthogonal to the BN-normalized activations. This bounds the update magnitude during training; it does not imply that the final core weights are unaffected (Table 13). In a model *without* BN (see Table 9), there is no such filtering mechanism

Table 10: Explanation consistency (MSE ↓) for backdoored ResNet-20 on CIFAR-10 under Simple Fooling (SF) and Red Herring (RH). We compare our method against Softplus [41, 42], Aggregated Explanations (Agg.) [43], and SmoothGrad (SG.) [74]. Full Disguise (FD) is not included: it keeps the explanation unchanged and alters only the prediction, so explanation MSE is not informative for it. Its accuracy and attack success rate are reported in Table 11.

Attack	Explainer	Ours	Softplus	Agg.	SG.
SF	Grad	**0.03**	0.25	0.23	0.09
	GradCAM	**0.05**	0.31	0.29	0.12
	RelCAM	**0.02**	0.28	0.28	0.17
RH	Grad	**0.02**	0.18	0.17	0.08
	GradCAM	**0.03**	0.29	0.26	0.18
	RelCAM	**0.02**	0.14	0.14	0.16

Table 11: Full Disguise (FD) on ResNet-20 / CIFAR-10, complementing Table 10. FD keeps the explanation and changes only the prediction, so we report triggered accuracy (Triggered Acc: the fraction of triggered inputs classified as their true class) and attack success rate (ASR), both in %, for the attacked model and after our defense.

	Attack		Defense (Ours)	
Explainer	Triggered Acc ↑	ASR ↓	Triggered Acc ↑	ASR ↓
Grad	10	100	66	3
GradCAM	10	100	87	0
RelCAM	10	100	89	1

for ResNet20 (No-BN) and ViT-B/16: the adversarial residuals propagate directly into the core layers, and the updates to θ_{core} grow significantly (Figure 2b) during adversarial training, as observed empirically. Conversely, when BN layers are present and trained, they calibrate activations, which attenuates the trigger's effect on the core update, leading to the disparate update magnitudes captured by (66) and the BN expression above.

Combining Eqs. (37) and (57),

$$\boxed{\|\Delta\theta_{\text{BN}}\| \gg \|\Delta\theta_{\text{core}}\|}. \tag{74}$$

BN affine parameters (γ, β) aggregate coherent batch-level signals (means and covariances), while core gradients are de-meaned by BN's normalization and attenuated by deep Jacobians. Therefore, when κ is close to 1 (Prop. 3.6), the trigger-induced gradient concentrates on the BN parameters.

K Fisher Information Sanity Checks

Two endpoints sanity-check the mechanism. At $p = 0$ there is no coherent direction to align, so $\rho_{\text{BN/core}}$ takes its clean baseline. At $p = 1$, BN re-centers all-triggered batches to $\bar{\hat{z}} = 0$, forcing $\text{Cov}(r, \hat{z}) \to 0$, while identical gradients give $\text{Var}(r) \to 0$; only $\mathbb{E}[\bar{r}^2]$ survive and $\rho_{\text{BN/core}}$ revert to a clean-like baseline. The factor $p(1-p)$ vanishes at $p = 0$ and $p = 1$, in line with the clean-like ratios in Table 12 ($\rho = 6.4, 5.3$); for $0 < p < 1$ the ratio is elevated (16–23×).

L Ablation of Batch Size

In this section, we illustrate how the minibatch size influences clean accuracy, attack success rate (ASR), and explanation MSE during the FlashBN defense under a Red Herring (RH) attack on ResNet-20. In FlashBN, we recompute inference-time batch statistics and discard the BatchNorm affine parameters: the scale γ is set to 1, realigned to the current data, while the shift β is set to 0. This recalibration can slightly reduce the model's retained information about the original training distribution; therefore, we require sufficiently large batches to ensure the estimated statistics reflect an unchanged clean distribution. As a result, batch size plays a crucial role in the stability and effectiveness of FlashBN.

Figure 4a shows that clean accuracy is sensitive to batch size, stabilizing once the batch is large enough. Figure 4b indicates that the ASR remains essentially constant across batch sizes after defense, suggesting robustness of FlashBN against the RH trigger. Finally, Figure 4c demonstrates

Table 12: **Empirical Fisher trace** on ResNet-20/CIFAR-10 (BadNet, N=64 samples). $\rho_{\mathrm{BN/core}}$ jumps from 6× (clean) to 16–23× under mixed poisoning, reverting at 100% poisoning rate as Proposition 3.6 predicts.

p	0.0	0.1	0.3	0.5	0.7	0.9	1.0
$\mathrm{tr}(\mathcal{F}_{\mathrm{BN}})$	0.033	0.205	0.242	0.247	0.126	0.263	0.053
$\mathrm{tr}(\mathcal{F}_{\mathrm{core}})$	0.005	0.009	0.011	0.015	0.008	0.016	0.010
$\rho_{\mathrm{BN/core}}$	6.4	**23.0**	**22.4**	16.5	16.7	16.6	5.3

that the explanation MSE decreases with larger batches, implying improved explanation fidelity as the batch-statistics estimates become unbiased from adversarial attack.

L.0.1 Causal Interventions

Causal analyses (BN and core-weight swaps, affine-free BN recalibration, BN retraining, and ablation studies) are conducted on a matched clean vs. attacked ResNet-20 pair trained on CIFAR-10. To test where the backdoor resides, we perform causal interventions between a clean and an attacked ResNet-20 (Table 13). Replacing only BN parameters from the clean model into the backdoored one (all BN parameters, only (γ, β), or only the running statistics) does not fully remove the backdoor. In contrast, replacing only the clean core convolutional weights while keeping the corrupted BN buffers yields an almost backdoor-free model, with ASR near zero and clean accuracy close to the original baseline. We further evaluate inference-time BN recalibration by re-estimating BN statistics on a small clean calibration set: across batch sizes from $n = 4$ up to $n = 512$, ASR is consistently driven close to zero with only minor accuracy changes. Finally, retraining *only* the BN layers on 1000 clean samples recovers the model to almost clean accuracy with ASR ≈ 0.0. Replacing BN parameters or statistics with the clean model's leaves ASR at 1.00, whereas replacing the core weights gives 0.01 (Table 13); the trigger is therefore not stored in BN alone. Re-estimating BN statistics or BN-only fine-tuning on a small clean set nevertheless removes the backdoor, consistent with BN acting as a pathway through which the trigger is expressed. This is a single ResNet-20/CIFAR-10 model pair.

Why transplanting clean BN does not defend. We read the trigger as encoded in the poisoned core and expressed through BN. A possible explanation, which we did not test, is that the poisoned (γ, β) and statistics have adapted to this core and absorb the near input-independent offset that the trigger writes into a few channels. A clean BN has not adapted to this core, so it would not remove that offset, and the trigger-selective filters would still reach the head. Conversely, a clean core lacks those filters, which would explain why the backdoor disappears even when the poisoned BN is kept. One possible explanation, which we do not test, is that clean and triggered signals share channels of the poisoned core so that any BN calibrated to a clean model's activations keeps the trigger path open. Recalibrating on clean data with the poisoned core's own activations, and removing the learned scale, may break this match between the BN operating point and the trigger-selective channels. FlashBN does not transplant anything. It recomputes the statistics on clean data for the poisoned core and resets (γ, β), which removes the offset for this core (Table 13b). This is consistent with the poisoned core being recoverable through BN, not unaffected (assumption A1).

M Proofs of Lemma 3.4 and Eq. 38

Lemma 3.4. Let h be the pooled penultimate feature vector and $z_\tau = w_\tau^\top h + b$ the logit of the target class τ of the linear head, with fixed weight vector w_τ and bias b. Let $\mathbb{E}_t$ and $\mathbb{E}_0$ denote expectations over triggered and clean inputs, and put $\Delta h = \mathbb{E}_t[h] - \mathbb{E}_0[h]$ and $\Delta z_\tau = \mathbb{E}_t[z_\tau] - \mathbb{E}_0[z_\tau]$.

Step 1 (linearity). Because w_τ and b do not depend on the input and expectation is linear,

$$\Delta z_\tau = \big(w_\tau^\top \mathbb{E}_t[h] + b\big) - \big(w_\tau^\top \mathbb{E}_0[h] + b\big) = w_\tau^\top \Delta h.$$

The bias cancels.

Step 2 (Cauchy–Schwarz). $w_\tau^\top \Delta h \le \|w_\tau\| \, \|\Delta h\|$, hence $\Delta z_\tau \le \|w_\tau\| \, \|\Delta h\|$.

Table 13: Causal interventions on a single backdoored ResNet-20 / CIFAR-10 model, with a matched clean model (clean baseline: Acc = 91.91; ASR as a fraction). Each row states what happens to the BN scale and bias (γ, β), the BN statistics (μ, σ^2), and the core weights: *kept* = unchanged from the backdoored model, *clean* = replaced by the clean model's, *reset* = set to $(\gamma, \beta) = (1, 0)$, *re-est.* = recomputed on n clean samples, *fine-tuned* = trained on n clean samples. (a) Cross-model transplant, a diagnostic and not a defense: replacing BN parameters or statistics alone leaves the attack intact (ASR 1.00), whereas replacing the core removes it (0.01). A clean BN transplanted onto the poisoned core does not suppress the trigger; a possible explanation, which we did not test, is that the trigger-selective filters remain in the core and the clean BN was not calibrated to this core's activations (Sec. L.0.1). (b) Defenses using a small clean set: FlashBN-style affine-free recalibration and ReBN-style BN-only fine-tuning.

Intervention	(γ, β)	(μ, σ^2)	**Core**	**ASR** ↓	**Clean Acc** ↑
(a) Cross-model transplant (clean → backdoored)					
Replace BN (all params)	clean	clean	kept	1.00	90.72
Replace (γ, β)	clean	kept	kept	1.00	90.72
Replace BN statistics	kept	clean	kept	1.00	90.53
Replace core weights	kept	kept	clean	0.01	91.66
(b) Recalibration on a small clean set					
ReBN-style fine-tuning ($n = 1000$)	fine-tuned	kept	kept	0.00	90.91
FlashBN-style, $n = 4$	reset $(1, 0)$	re-est.	kept	0.04	90.72
FlashBN-style, $n = 32$	reset $(1, 0)$	re-est.	kept	0.01	90.72
FlashBN-style, $n = 512$	reset $(1, 0)$	re-est.	kept	0.01	90.72

Table 14: Results under the *COMBAT* attack, where the trigger is fully imperceptible to human vision. The attacker adopts an adaptive strategy by intentionally freezing the Batch Normalization (BN) learnable parameters during training to strengthen the backdoor effect. Under this setting, we observe a severe degradation in clean accuracy after applying defenses: on average 86.6 points for ReBN and 13.0 points for FlashBN (poisoning rates 0.1–0.7). The drop may signal an adaptively trained model, but ASR is not reported here, so we do not claim that the backdoor is removed or that the drop is a reliable detector.

Poison Rate	Clean Acc (Attacked)	Clean Acc (ReBN)	Clean Acc (FlashBN)
0.1	89.34	2.76	77.14
0.2	90.62	3.41	77.65
0.3	90.25	5.11	77.05
0.4	89.73	3.58	75.13
0.5	90.55	1.82	77.71
0.6	90.15	3.96	77.88
0.7	89.91	3.94	76.87

Step 3. If $\Delta z_\tau > 0$ then $w_\tau \neq 0$, because for $w_\tau = 0$ the logit is constant and $\Delta z_\tau = 0$. Dividing by $\|w_\tau\|$ gives

$$\|\Delta h\| \;\geq\; \frac{\Delta z_\tau}{\|w_\tau\|}.$$

Equality holds exactly when Δh is parallel to w_τ. The statement is about the class-conditional means of the features: for a fixed head, the larger the gap between the mean target logits of triggered and clean inputs, the farther apart their mean features must be. The lemma concerns the penultimate layer only; that the BN channels show the same gap is measured in Appendix N.

Eq. (38)**.** Let $Y \in \{0, 1\}$ mark triggered samples with $\Pr(Y{=}1) = p$, and let $\Delta_r = \mathbb{E}[r \mid Y{=}1] - \mathbb{E}[r \mid Y{=}0]$ and $\Delta_{\hat{z}}$ likewise. With $\varepsilon_r = \mathbb{E}_Y \mathrm{Var}(r \mid Y)$, $\varepsilon_{\hat{z}} = \mathbb{E}_Y \mathrm{Var}(\hat{z} \mid Y)$ and $C_w = \mathbb{E}_Y \mathrm{Cov}(r, \hat{z} \mid Y)$, the laws of total variance and covariance give $\mathrm{Var}(r) = \varepsilon_r + A_r$, $\mathrm{Var}(\hat{z}) = \varepsilon_{\hat{z}} + A_{\hat{z}}$ and $\mathrm{Cov}(r, \hat{z}) = C_w + p(1-p)\Delta_r \Delta_{\hat{z}}$, where $A_r = p(1-p)\Delta_r^2$, $A_{\hat{z}} = p(1-p)\Delta_{\hat{z}}^2$ and $p(1-p)\Delta_r \Delta_{\hat{z}} = \mathrm{sgn}(\Delta_r \Delta_{\hat{z}})\sqrt{A_r A_{\hat{z}}}$. Cauchy–Schwarz within each group and then over the groups gives $|C_w| \leq \sqrt{\varepsilon_r \varepsilon_{\hat{z}}}$.

Bound on κ. The complete proof of the lower bound on $|\kappa|$ in Proposition 3.6 is provided in Appendix F.

Table 15: Clean-twin test. $t_r = \varepsilon_r / A_r$ and $t_{\hat{z}} = \varepsilon_{\hat{z}} / A_{\hat{z}}$ are within-to-between scatter ratios. Each entry is the fraction of BN channels where the backdoored model's ratio is smaller than its clean twin's (chance $= 0.5$).

Attack	t_r	$t_{\hat{z}}$
Adaptive Blend	0.796	0.756
BPP	0.899	0.881
COMBAT	0.817	0.611
CTRL	0.695	0.846
Diffusion	0.667	0.779
FTrojan	0.678	0.944
PoisonInk	0.465	0.697
ReAttack	0.592	0.606
Mean	0.701	0.765

Table 16: Share of core channels of the poisoned model beyond a clean-vs-clean floor (chance $= 5\%$).

Attack	Trigger type	Channels beyond floor (%)
Adaptive Blend	static	2.2
CTRL	static	2.2
Diffusion	static	2.2
FTrojan	static	2.2
BPP	dynamic	35.3
PoisonInk	dynamic	34.8
ReAttack	reactivation	32.8

N Clean-twin test

For each attack, we compare a backdoored model with an independently trained clean twin on identical batches. Table 15 reports, per attack, the fraction of BN channels in which the backdoored model's within-to-between scatter ratio is smaller than its twin's (chance $= 0.5$).

Sign test across attacks. $t_{\hat{z}}$ lies above chance (0.5) for 8 of 8 attacks and t_r for 7 of 8. Under the null that each attack is equally likely to fall above or below 0.5, the one-sided binomial p-values are $1/256 \approx 0.004$ and $9/256 \approx 0.035$; two-sided, 0.008 and 0.070. The result for t_r is therefore significant at the 0.05 level only one-sided.

O Checks of the Assumptions in Theorem 3.8

Margin γ_0 (A3). For all eight attacks of Table 1, the reference model of (A3) rejects triggered inputs with a median margin of 5.5–10.7 logits, and 96.5–100% of triggered inputs are rejected with a positive margin.

Core distance δ (A1). Raw weight distance is uninformative, because two clean networks already differ by permutation symmetry. As a proxy for δ, we build a floor from a second, independently trained clean checkpoint and count the core channels of the poisoned model that lie beyond it (chance level: 5%; Table 16). COMBAT needs its own clean reference and is not on this scale. Static triggers stay below the chance level, which supports (A1) for them. Dynamic and reactivation attacks move about $15\times$ more core channels, so (A1), and with it the guarantee of Thm. 3.8, is not established for them, although the defense still lowers their ASR empirically (Table 1).

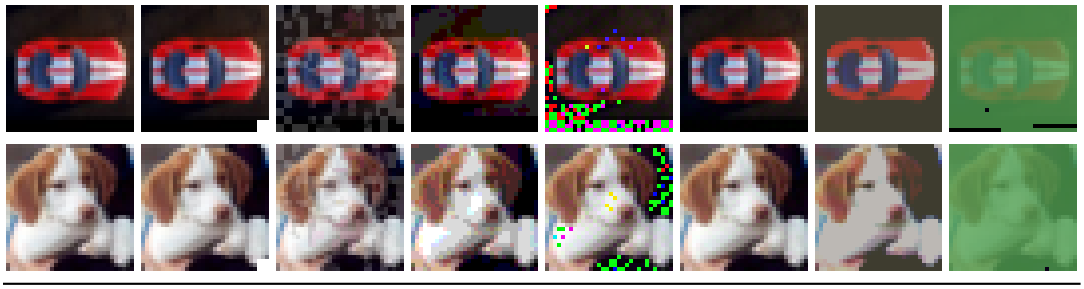


Figure 6: Clean and poisoned samples across seven backdoor attacks.

P Qualitative Visualizations

In this section, we present comprehensive visual results to illustrate the visual stealth of backdoor attacks and demonstrate the effectiveness of FlashBN across different datasets, attack objectives, and baselines.

P.1 Backdoor Sample Visualizations

Figure 6 provides visual examples of clean samples alongside their corresponding poisoned counterparts generated by seven distinct backdoor attacks (BadNet, Blend, BPP, CTRL, FTrojan, COMBAT, and PoisonInk) for both car and dog image instances.

P.2 Defending Against Explanation-Aware Attacks

To evaluate explanation restoration under diverse threat models:

- **CIFAR-10 Evaluation:** Figure 7 illustrates model explanations under Simple Fooling, Red Herring, and Full Disguise attacks on CIFAR-10, demonstrating that FlashBN reliably restores faithful feature attributions and accurate predictions.
- **GTSRB Evaluation:** Figure 9 presents qualitative results on the GTSRB traffic sign dataset across all three attack paradigms, confirming the cross-dataset robustness of our method.
- **Comparison with Baseline Defenses:** Figure 8 compares FlashBN with existing explanation-aware defenses (SmoothGrad, Aggregated Explanations, and Softplus smoothing) under a Red Herring attack with various triggers, showing that our approach achieves superior explanation recovery.

Q Limitations & Discussion

While our method provides, to our knowledge, the first formal theoretical analysis of the relationship between batch normalization and backdoor attacks, its guarantees are conditional, and it is limited to architectures that use batch normalization (ResNet-20 without BN and ViT-B/16 are failure cases, Table 9). Nevertheless, batch normalization remains widely deployed in legacy CNNs and many production systems, which allows our method to be applied to these models in practice. Extending these findings to other normalization techniques, such as layer normalization in Vision Transformer architectures, is left for future work.

Broader impact. FlashBN and ReBN are defenses and can help protect deployed BN-based models against training-time backdoors. Our analysis of how a backdoor uses BN may also inform attackers; the adaptive COMBAT setting (Table 14) shows one such strategy, and we report it so that defenders can account for it.

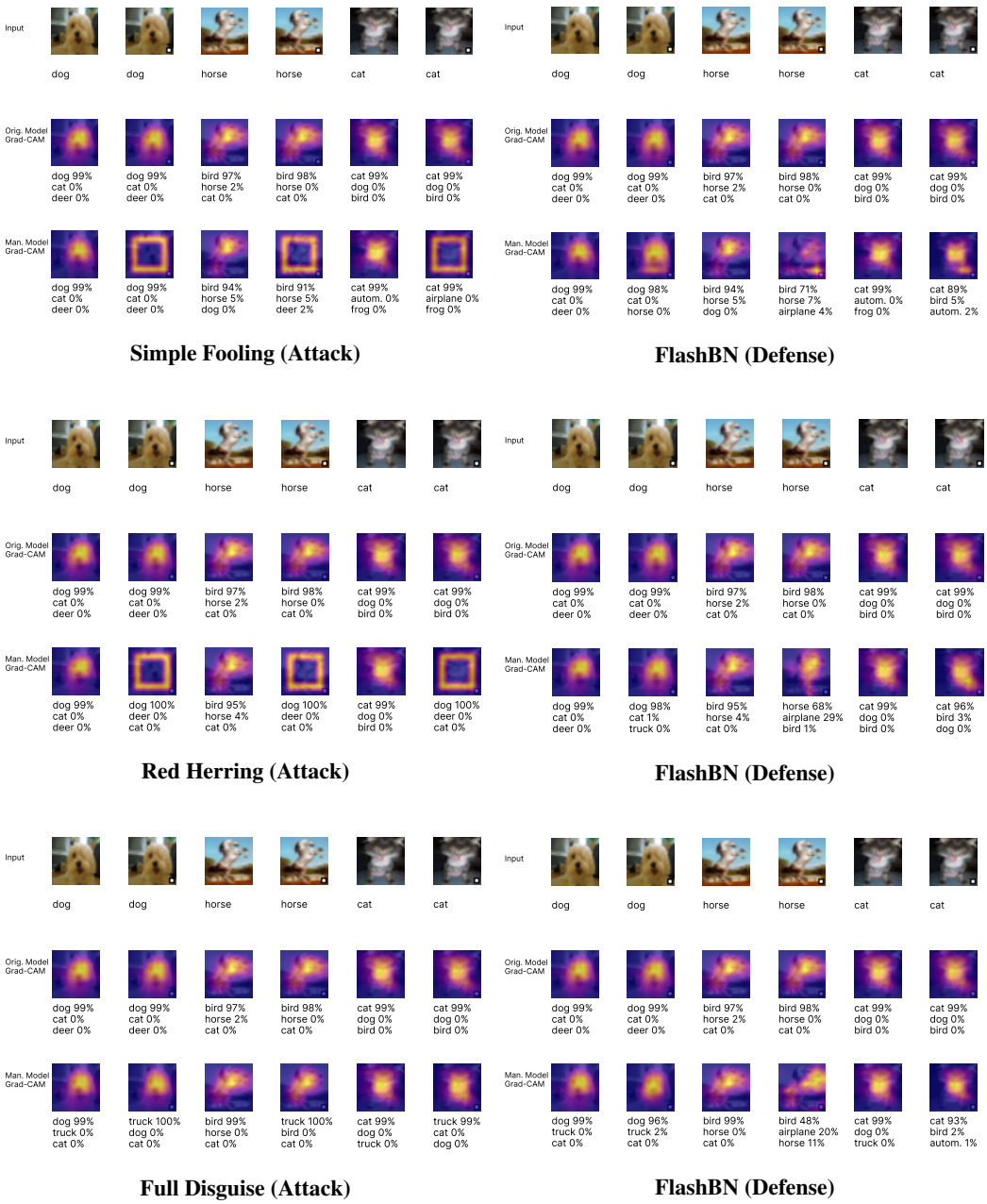


Figure 7: **CIFAR-10 adversarial attacks and FlashBN defense.** We evaluate three attack types: Simple Fooling (only explanations are manipulated while predictions remain correct), Red Herring (both predictions and explanations are corrupted), and Full Disguise (predictions are flipped while explanations appear legitimate). FlashBN consistently restores faithful explanations and correct predictions across all cases by recalibrating Batch Normalization statistics during inference.

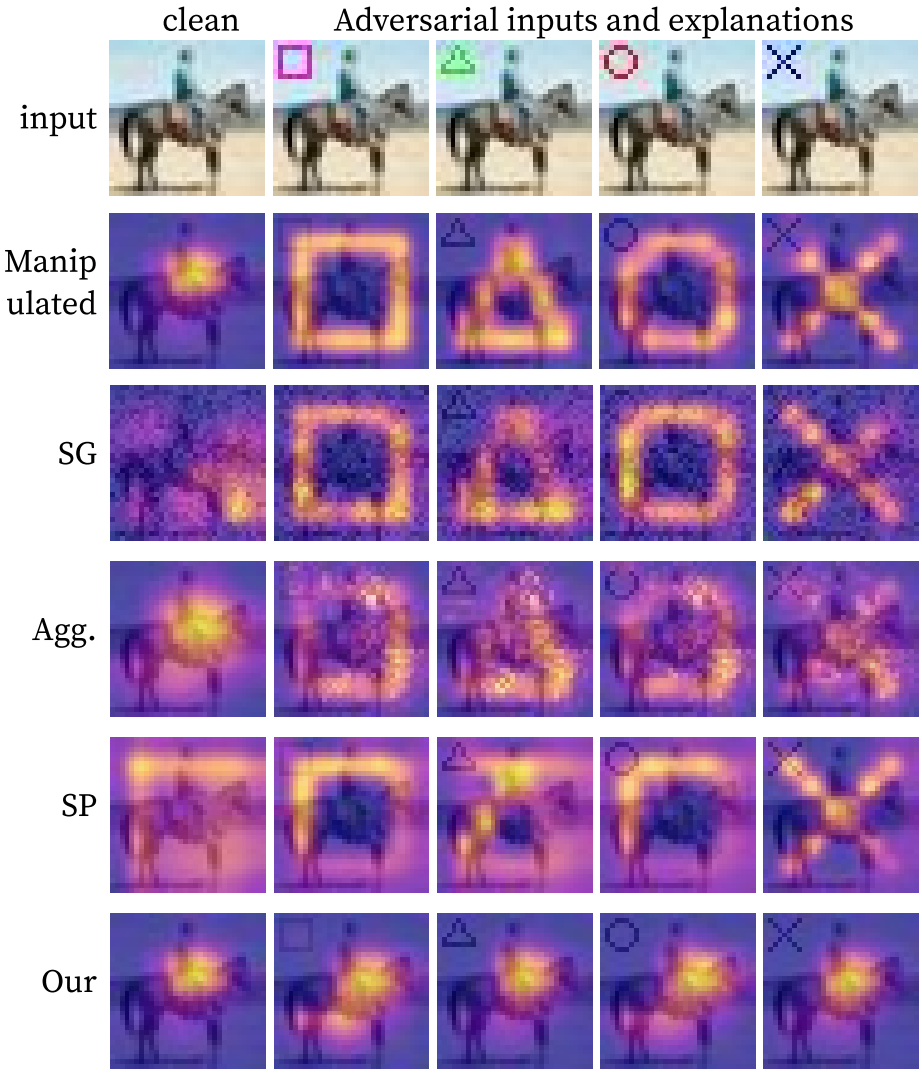


Figure 8: Red Herring (RH) attack with different triggers. The first column shows the clean input explanation; the next four show the same input with different triggers. The second row illustrates attacker manipulation. The bottom four rows compare explanation-aware defenses: SmoothGrad, Aggregated Explanations, Softplus smoothing, and our method, showing that our approach best restores the clean explanation.

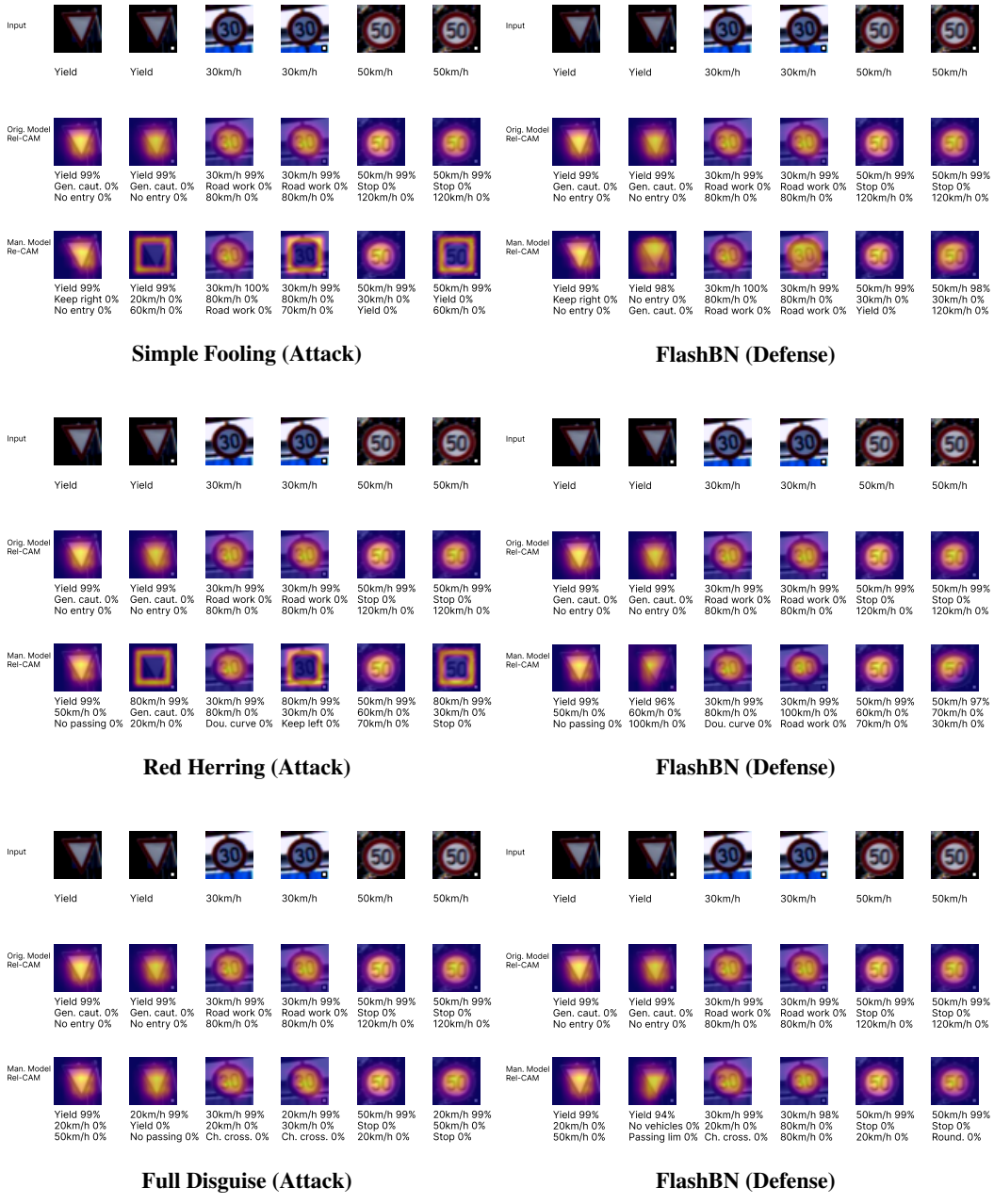


Figure 9: **GTSRB adversarial attacks and FlashBN defense.** We evaluate three attack types: Simple Fooling (explanation changes while prediction remains correct), Red Herring (both prediction and explanation are manipulated), and Full Disguise (prediction is flipped while explanation appears legitimate). FlashBN consistently restores faithful explanations and correct predictions across all settings.