Finding FAULTs in Architectural Backdoors: Why Trigger Detection Fails Under Real-World Conditions

Anonymous Author(s)

Abstract

Architectural Backdoors (ABs) have been proposed as a stealthy threat towards secure DL model deployments. ABs hide malicious computation in DL model inference code rather than the model's weights, evading traditional model vetting techniques. However, the practical viability of ABs under real-world deployment conditions has remained unexamined. We show that architectural backdoors are constrained by three fundamental limitations inherent to their trigger detection step: (1) input-space transformations routinely applied during preprocessing corrupt triggers before detection; (2) relaxing detection sensitivity to recover attack success rate inevitably increases false positives, producing observable accuracy degradation (reducing the stealth of the attack) and (3) as trigger complexity grows, the collision space of benign inputs that falsely activate the detector expands, further coupling accuracy loss to attack viability. We formalize these limitations mathematically and introduce FAULT, a framework that enables characterization of the attack-success-rate, accuracy, and false-positive tradeoff space of ABs across varied trigger and detector types. Our evaluation across the GTSRB, Mapillary, and CIFAR10 datasets confirms that architectural backdoors are impractical in realistic deployments.

CCS Concepts

• **Security and privacy → Software and application security**;
• **Computing methodologies → Machine learning**.

Keywords

Machine Learning Security, Architectural Backdoors, Backdoor Mitigation

ACM Reference Format:

Anonymous Author(s). 2026. Finding FAULTs in Architectural Backdoors: Why Trigger Detection Fails Under Real-World Conditions. In *Proceedings of the 2026 ACM Conference on Computer and Communications Security (CCS '26), November 15–19, 2026, The Hague, The Netherlands.* ACM, New York, NY, USA, 17 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Researchers have proposed a new attack against deep learning (DL) systems called *architectural backdoors* (ABs) [1–4]. Unlike weight-based backdoors that hide malicious behavior in a model's parameters, ABs embed trigger detection logic directly into the model's inference code, evading every existing weight-vetting technique [5–7]. ABs can be introduced into DL systems via the same attack vectors as other malicious-code threats: over-the-air software updates [8, 9], code-injection attacks [10], and the AI supply chain [11–14]. Because the backdoor logic lives in code rather than in the trained model, an attacker who compromises a single model repository can propagate malicious behavior to downstream deployments without modifying any model weights [2].

The two existing AB constructions [1, 2] demonstrate that ABs can be implanted in diverse model architectures and survive retraining or fine-tuning, establishing the threat as feasible in principle. Subsequent surveys [4] catalog AB constructions and proposed defenses. Yet *the practical viability of ABs under realistic deployment conditions has not been characterized.* Real-world DL pipelines apply input transformations (cropping, contrast adjustment, noise injection, denoising, etc.) before inference, both for robustness [15] and to defend against adversarial attacks [16, 17]. Whether ABs survive these transformations, and at what cost to the attacker, remains unknown. Without this characterization, the research community cannot say whether ABs are a serious threat warranting dedicated defenses or a brittle attack that real-world conditions already neutralize.

Characterizing AB viability across realistic conditions is difficult because of the space of design choices an attacker controls is large: each AB is parameterized by a trigger pattern (which can range from a binary square to an arbitrary RGB image), a sensitivity threshold (which the attacker can tune), and a detector implementation (convolution, pooling, slicing, concatenation, or masking, per prior work). The space of conditions a victim's deployment imposes is similarly large, with varied input transformations often applied during deployment. Moreover, the attacker and victim's choices interact. An input transformation that defeats one trigger may leave another intact, and a sensitivity setting that recovers attack success rate (ASR) under one transformation may inflate false positives under another. Naively evaluating every (trigger, sensitivity, detector, transformation) combination scales exponentially. Worse, a defender cannot simply pick representative configurations because the attacker, by definition, will tune their parameters to whatever the deployment exposes (as demonstrated in §6.4). Any meaningful viability claim must therefore characterize the *boundary* of what an optimally-tuned attacker can achieve.

However, all existing AB constructions, despite their surface-level diversity, share a common pattern. We make the key insight that each detector computes a parameterized similarity score between an input region and a fixed reference pattern derived from the trigger, then thresholds the score against a sensitivity-controlled bias to produce a binary signal (detected or not detected). We show that the five detector families (convolution, pooling, slicing, concatenation, masking) proposed in prior

CCS '26, The Hague, The Netherlands

ACM ISBN 978-1-4503-XXXX-X/26/11
https://doi.org/XXXXXXX.XXXXXXX

work [1, 2, 4] all reduce to the same inner-product form. This unification means that the entire AB design space reduces to two underlying quantities, a trigger-induced response distribution and benign-input response distribution. We show that the attacker's task fundamentally collapses to placing a threshold between these distributions.

Based on this insight, we formalize three fundamental limitations of AB trigger detection and develop FAULT (**F**ramework for **A**ssessing **U**nreliability and **L**imitations of AB **T**rigger-detection), an evaluation framework that empirically validates them. FAULT takes any AB construction and any set of preprocessing transformations and traces the full (ASR, ACC, FPR) operating curve as a function of the sensitivity parameter to expose exactly where the attacker's achievable region begins and ends. Using FAULT, we identify three limitations that bound this region: (1) input-space transformations corrupt trigger pixels before they reach the detector, attenuating the detector's response below threshold; (2) raising the threshold to recover ASR also raises the false-positive rate, producing observable benign-accuracy degradation; and (3) more complex triggers require detectors that admit a larger volume of benign-input collisions, further coupling accuracy loss to attack viability. Together, these limitations bound the achievable ASR-ACC operating region for every AB an attacker can construct.

We evaluate FAULT across the GTSRB [18], CIFAR10 [19], and Mapillary [20] datasets, three model architectures (ResNet18, MobileNetV2, YoloV11), three signal-propagation paths, eleven trigger configurations, ten transformations, and five detector implementations. Across all 30 (configuration, transformation) combinations we tested, no choice of sensitivity tuning achieves both meaningful ASR and benign accuracy near baseline. Using FAULT we demonstrate that the attacker's reachable ASR ceiling is bounded between roughly 10% and 20%, and reaching even this modest target costs an average of 2.23% benign-accuracy degradation, with outliers up to 9.4%. Trigger complexity makes the problem worse. Across all eleven trigger variants on all three datasets, ASR remains at 1.0 but benign accuracy collapses from baseline (89–97%) to as low as 9.9% on the most complex triggers (with 9,918 of 10,000 benign CIFAR10 samples falsely activating the detector at the worst configuration). Across all five detector types, false-positive counts vary by less than 3 percentage points, confirming that the limitations are properties of the detection problem itself, not of any specific implementation. ABs, as currently constructed, cannot be both effective and stealthy in realistic deployments.

2 Backdoor Attacks on DL Models

Backdoor attacks [1, 2, 21, 22] against DL models rely on various attack vectors in the DL deployment pipeline (shown in Figure 1). All involve embedding a trigger detector into a model, ④, such that the model behaves normally on clean inputs but misclassifies inputs containing an adversary chosen trigger ⑨. Let $\mathcal{X}$ be the input space and $\mathcal{Y}$ be the label space. Define a classifier $f_\theta : \mathcal{X} \to \mathcal{Y}$, where θ are the model parameters (weights ②). If $x \in \mathcal{X}$ is a clean input, the model should predict the correct label:

$$f_\theta(x) = y \quad \text{with high probability,} \tag{1}$$

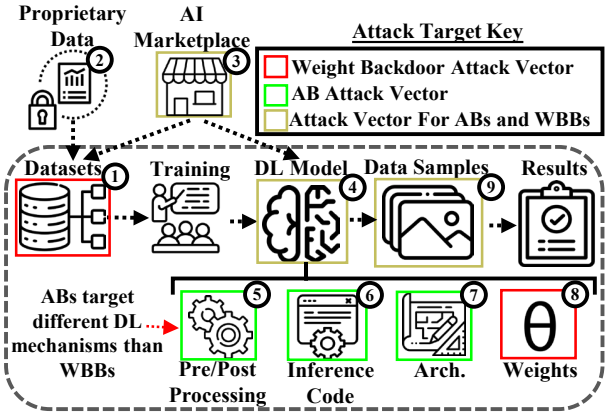


Figure 1: Simplified process of DL model deployment. WBBs and ABs have different attack surfaces (boxed in green for ABs and boxed in red for WBBs) that the attacker can exploit.

where y is the true label. However, if x is modified by adding the trigger δ, resulting in $x' = x + \delta$, the model is designed to misclassify this triggered input:

$$f_\theta(x') = y' \quad \text{with high probability,} \tag{2}$$

where y' is either a specific adversary-chosen label (in targeted attacks) or any incorrect label (in untargeted attacks). ABs and WBBs, while similar in that they both adhere to Equation 1 and Equation 2, are overall fundamentally different which motivates a unique threat model for ABs (§3).

2.1 Fundamental Differences Between WBBs and ABs

First, we highlight the differences between WBBs and ABs along three axis: their implantation, operation, and the mitigation to each respective attack.

Backdoor Implantation. WBBs and ABs are implanted into a DL system in completely orthogonal ways. WBBs are implanted into a model's weights primarily through poisoned datasets [21, 23–26] (① in Figure 1), runtime-based attacks [27], and through malicious model files shared on large AI Marketplaces [13] ③ that may be finetuned with proprietary data, ②. The trigger detector is implicitly defined in θ, ⑧, and does not leave any visibly incriminating artifacts in the DL system (e.g. code).

ABs however embed the trigger detector directly within DL model architecture and code (⑤-⑦) while still following Equation 1 and Equation 2. Adversaries intending to spread ABs may use code-injection [10], but primarily aim to use supply-chain based attacks [11, 12, 28] to spread ABs within DL systems, uploading them to popular AI marketplaces [14, 29], intending for the backdoored code to be inadvertently used by others. They strongly assume an AI practitioner is unable to identify the trigger detector in the model architecture [2]. Prior work [2] has found that this assumption is realistic, and even AI practitioners can be fooled by ABs that masquerade as combinations of standard PyTorch primitives.

Backdoor Operation. In WBBs, the adversary directly targets a single component of the model, its parameters θ ⑧, to cause misclassifications for trigger containing inputs, ⑨. On the other hand, AB operation can be broken into two main components: trigger detection, and signal propagation. For trigger detection, Bober-Irizar et al. [1] an AB that first applies an exponential transformation to the input image, to amplify specific pixel values associated with the trigger. Then, a pooling operation and `max` operation across RGB channels is applied to reduce the activation to a single value α (used to indicate trigger presence). Langford et al. [2] leverage a combination of standard computational primitives such as pooling layers, activation functions, or parameter-less logic operations to detect and propagate arbitrary trigger signals. However, we show in §6.2 that the success of *all* ABs is limited because trigger detector effectiveness is sensitive to input transformations.

Backdoor Mitigation. WBBs are mitigated primarily through strategies that involve modifying the model's weights. Model weight pruning [5, 6], activation clustering [30], subsequent fine-tuning [7], and direct model retraining [2] are effective at removing WBBs. Such approaches however do not apply to ABs, as the trigger detector is not embedded in θ but in the code itself [2].

For AB mitigation, prior work has suggested visual inspection, but this is unfortunately insufficient as models are often redesigned, customized, and improved through SOTA research. This blurs the line between "weird" looking code that intends to improve model performance and an AB [2]. ABs masquerade as benign computational primitives that exist naturally within the DL model [2], can be constructed from a large number of permutations of simpler operations, and can hide within tens of thousands of lines of DL system code seen in DL systems [31–33]. However, existing ABs can be mitigated (and even detected) via simple benign transformations to the input space (shown in §6.2) that impede AB trigger detection.

3 FAULT Attacker Model and Assumptions

Our work adopts the threat model outlined in prior AB work [1, 2, 4], where an attacker introduces a AB into a DL model by modifying only the model's architecture definition. The attacker is restricted to using commonly supported DL model components without injecting arbitrary code, ensuring that the backdoor persists through training or fine-tuning. The attacker is unable to access the training dataset (e.g. the DL practitioner downloads an existing model from Hugging Face [14] to train on a proprietary dataset) and may not even know the deployment task itself (as the model can be retrained/finetuned on anything by the user). But the attacker *does* need to upload backdoored code to the AI supply chain such that it is reused by unknowing victims. The victim unknowingly downloads the attacker's model, retrains the model from scratch on their own data to avoid any poisoning attacks, and evaluates the model's performance before deployment. During deployment, the practitioner may also sandbox the model to limit runtime behavior changes but does not modify pre-loaded architectures, to not impact model performance. This mirrors real world scenarios, as developers often rely on open source code to aid the engineering of their models [34, 35].

We assume that the victim also intentionally applies one of, or a combination of image transformation techniques (e.g. contrast [36], brightness [37], cropping [36], noise [38], etc.). The attacker is unable to control what transformations will be used, does not have oracular knowledge of what transformations are used, and cannot apply reverse engineering techniques to discover the transformations used. We simulate this situation through the construction/tuning of ABs with FAULT. We demonstrate success for various attacker goals (low/medium/high stealth goals in §6.5) but at the cost of a degrade in model utility.

FAULT Assumptions. FAULT is the evaluation tool we use to determine a bound on the attacker's capabilities. To produce a tight bound, FAULT is given full knowledge of the deployment's transformations and the freedom to tune the AB's sensitivity parameter N for any chosen attacker goal. FAULT therefore simulates a strictly stronger attacker than the one described above, a real attacker without this knowledge would do worse. The negative results we report in §6 are consequently a *ceiling* on attacker capability under this threat model, not an average case.

4 Formalizing the Limitations of Architectural Backdoor Trigger Detection

In this section, we provide an analysis of the trigger detection step in ABs. We show that regardless of the detection mechanism used (convolution, pooling, etc.), the trigger detection problem is subject to three fundamental limitations that constrain the practical viability of ABs in real-world DL systems.

4.1 AB Trigger Detection as Parameterized Pattern Matching

Existing AB constructions share a common structure: a trigger detector that computes a similarity measure between a region of the input and an attacker-defined trigger pattern, followed by a thresholding step that produces a binary detection signal. We now unify these constructions under a single formalism.

A trigger pattern is defined as a fixed tensor $\mathbf{T} \in [R_{lower}, R_{upper}]^{C \times H \times W}$ where R_{lower} and R_{upper} denote the lower and upper bounds of values for which a pixel can be defined for any input image given to the AB-containing model (e.g., $[0, 1]$, $[0, 255]$, etc.). This encodes the shape, color, and spatial structure of the attacker's chosen trigger across C channels and a $H \times W$ spatial region. A trigger detector is a function $\mathcal{D}(x) \rightarrow \{0, 1\}$ given an input image x outputs 1 if the trigger is deemed present and 0 otherwise. $\mathcal{D}$ can be constructed as an indicator function $\mathbf{1}[\phi(x) > \theta]$ applied to a response function $\phi(x)$ that computes a scalar measure of the similarity between $\mathbf{T}$ and a given region in an input image x. We now formulate the response functions $\phi(x)$ for each detector.

Convolution-Based Detectors. The response function computes a channel-wise convolution of x with a kernel W derived from T, followed by pooling:

$$\phi_{\text{conv}}(x) = \text{Pool}\left(\sum_{c,r,s} W_{c,r,s} \cdot x_{c,\, i+r,\, j+s} + b\right) \tag{3}$$

where $b = -\lambda(1 - N)$ encodes the sensitivity parameter $N \in [0, 1]$ and $\lambda = \sum_{c,r,s} \max(W_{c,r,s}, 0)$ is the filter intensity. Detection occurs when after applying ReLU to $\phi_{\text{conv}}(x)$ there are non-zero elements in the result (signal of 1 for the indicator function).

Pooling-Based Detectors. The pooling detector composes asymmetric max-pooling operations with sign flips, where the kernel sizes are chosen to match the spatial extent of the trigger pattern T. Letting $P_{a,b} = \text{maxpool}_{a,b}$ and noting that $-P_{a,b}(-x)$ implements min-pooling, the detector is:

$$\phi_{\text{pool}}(x) = \left[-P_{1,b}(-P_{a,1}(x))\right] \cdot \left[P_{a,1}(-P_{1,b}(-x))\right] \tag{4}$$

The two factors compute dual max-then-min and min-then-max directional filters. Their product fires only where both conditions hold simultaneously, which occurs precisely at locations matching the structure of T. The pooling configuration (a, b) together with the input sign define an implicit kernel W_T encoding T, making this a parameterized similarity between x and W_T.

Slicing-Based Detectors. The slicing detector extracts input positions corresponding to $\mathcal{P}_T$ and aggregates them against the trigger values:

$$\phi_{\text{slice}}(x) = \sum_{p \in \mathcal{P}_T} W_p \cdot x_p + b \tag{5}$$

where $W_p = T_p$ at sliced positions and the sum is computed over a sliding window of locations. This is equivalent to convolution with a kernel that is non-zero only over $\mathcal{P}_T$; the explicit per-position indexing differs only in how it is expressed in the computation graph.

Concatenation-Based Detector. The concatenation detector constructs the trigger tensor and a localizing mask from parameter-free operators (e.g., $0 = x - x$ and $1 = \sigma(0) + \sigma(0)$), concatenates them with the input, and computes a similarity score over the masked region:

$$\phi_{\text{concat}}(x) = \sum_{p} (M \odot x)_p \cdot W_p + b \tag{6}$$

where M is the constructed binary mask localizing T and W encodes the trigger values. The actual scoring is again the inner product between the masked input and the trigger weights.

Masking-Based Detector. The masking detector embeds the mask and trigger as non-trainable constants directly:

$$\phi_{\text{mask}}(x) = \sum_{p} (M \odot x)_p \cdot (M \odot W)_p + b \tag{7}$$

This is functionally identical to the concatenation detector and to a convolution restricted to the mask support, but introduces M and W as embedded constant tensors.

4.1.1 Unifying Observations. All five detector types compute the same parameterized similarity between an input region and a fixed reference pattern derived from T, followed by a threshold:

$$\phi(x) = \max_{\text{patches}} \left[\langle W, x_{\text{patch}} \rangle + b\right], \quad d(x) = \mathbf{1}[\phi(x) > 0] \tag{8}$$

where W encodes T (possibly restricted to a mask M), $\lambda = \langle W, T \rangle$ is the ideal response when the input patch exactly matches the trigger, and $b = -(\lambda - N)$ is the sensitivity-controlled bias. The detectors differ only in how this score is expressed in the graph; all reduce to the same inner product. The threshold $\theta = \lambda - N$ controls the sensitivity-specificity trade-off. Using the inner product we define the trigger response and benign response distribution.

Definition 4.1 (Trigger Response). For an input x containing T at the expected location, the inner product reaches its maximum value $\langle W, T \rangle = \lambda$, giving an ideal response $R_{\text{ideal}} = \lambda + b = N$. Setting the bias to $b = -(\lambda - N)$ thus calibrates the detector so that a perfect trigger match produces exactly the sensitivity value N, with $N > 0$ guaranteeing detection after the ReLU and threshold.

Definition 4.2 (Benign Response Distribution). For an input $x \sim \mathcal{D}$ drawn from the natural data distribution (an image without the trigger), the score $\phi(x) = \max_{\text{patches}} \langle W, x_{\text{patch}} \rangle + b$ is a random variable. Because $\mathcal{D}$ is itself a distribution over images and the max is taken over all $k \times k$ patches in x, $\phi(x)$ has an induced distribution that we denote R_{benign}, with cumulative distribution function $F_{\text{benign}}(r) = \Pr_{x \sim \mathcal{D}}[\phi(x) \leq r]$. Intuitively, F_{benign} captures how often a benign image happens to contain a patch that, by chance, has high inner product with the attacker's chosen kernel W. The shape of F_{benign} depends jointly on the trigger T (which determines W) and on the statistics of $\mathcal{D}$. The false positive rate at threshold θ is $1 - F_{\text{benign}}(\theta)$.

Using these definitions, the remainder of this section highlights that the fundamental limitations of ABs stem from the properties of the trigger detection step, not any specific implementation.

4.2 Limitation 1: Transformation Sensitivity

Real-world DL systems routinely apply preprocessing transformations to input data before inference, both for robustness and as standard engineering practice [15, 39]. We now show that any such transformation degrades the trigger detector's response, regardless of which detector type (from §4.1) is used.

Definition 4.3 (Input Transformation). We define an input transformation $\mathcal{A} : \mathbb{R}^{C \times H \times W} \to \mathbb{R}^{C \times H \times W}$ as a (possibly stochastic) function applied to the input before it reaches the model.

4.2.1 Transformation-Induced Response Attenuation. Let $x_\tau = x \oplus T$ be a triggered input and let $\mathcal{A}$ be a non-identity input transformation. Then the score on the trigger-containing patch becomes:

$$\phi(\mathcal{A}(x_\tau)) = \langle W, \mathcal{A}(x_\tau)_{\text{patch}} \rangle + b = R_{\text{ideal}} + \Delta_{\mathcal{A}} \tag{9}$$

where $\Delta_{\mathcal{A}} = \langle W, \mathcal{A}(x_\tau)_{\text{patch}} - T \rangle$ is the deviation introduced by the transformation. Recall $R_{\text{ideal}} = N$ (Definition 4.1); the trigger fires after ReLU when $R_{\text{ideal}} + \Delta_{\mathcal{A}} > 0$, i.e., when $\Delta_{\mathcal{A}} > -N$. Any transformation that drives $\Delta_{\mathcal{A}}$ below $-N$ silences the detector. We now analyze $\Delta_{\mathcal{A}}$ for example transformation classes, particularly those that are commonly used and heavily impede AB operations (from §6.2).

Additive Noise. For $\mathcal{A}(x) = x + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, the deviation $\Delta_{\mathcal{A}} = \langle W, \epsilon_{\text{patch}} \rangle$ is a zero-mean Gaussian with variance $\sigma^2 \|W\|_F^2$, where $\|W\|_F = \sqrt{\sum_i W_i^2}$ is the Frobenius norm of the kernel. Although $\mathbb{E}[\Delta_{\mathcal{A}}] = 0$, the added variance pushes a fraction of triggered inputs below the detection threshold:

$$\text{ASR}_{\text{noise}} = \Pr[\phi(\mathcal{A}(x_\tau)) > 0] = \Phi\left(\frac{N}{\sigma \|W\|_F}\right) \tag{10}$$

where Φ is the standard normal CDF. As noise intensity σ grows, ASR decreases monotonically.

Contrast Adjustment. For $\mathcal{A}(x) = \alpha x + \beta$, $\alpha \in (0, 1)$, where $\alpha \in (0, 1)$ is the contrast scaling factor and $\beta \in \mathbb{R}$ is a brightness offset, with $\mathbf{1}$ the all-ones tensor of the same shape as x): The transformed score is $\phi(\mathcal{A}(x_\tau)) = \alpha\langle W, T\rangle + \beta\langle W, \mathbf{1}\rangle + b$, yielding $\Delta_{\mathcal{A}} = (\alpha - 1)\lambda + \beta\langle W, \mathbf{1}\rangle$, where $\lambda = \langle W, T\rangle$ is the ideal trigger response from §4.1. Since $\alpha < 1$ and $\lambda > 0$, the first term is strictly negative, directly attenuating the trigger response by $(1 - \alpha)\lambda$. The second term $\beta\langle W, \mathbf{1}\rangle$ shifts the score uniformly regardless of input content; in practical contrast adjustments β is small relative to λ, so the attenuation dominates.

Spatial Transformations. For $\mathcal{A}(x)$ that rotates x by ψ, or with cropping retaining a fraction $\rho \in (0, 1]$ of the image, there is interpolation to produce the output image, which corrupts trigger pixel values even when the trigger remains spatially within the input. Rotation maps each output pixel to a coordinate in the original image and computes its value by interpolating (typically bilinearly) between the surrounding pixels. Cropping preserves only a subregion of the input but is followed by resizing back to the model's expected input dimensions, also via interpolation. The trigger is rescaled, with its pixel values blended with neighbors. In both cases the detector receives interpolated approximations of the original trigger values rather than the values themselves, so $\langle W, \mathcal{A}(x_\tau)_{\text{patch}}\rangle < \lambda$ and $\Delta_{\mathcal{A}} < 0$.

Key Takeaway. Under any non-identity transformation $\mathcal{A}$, the trigger response distribution shifts leftward (toward lower values) while the detection threshold remains fixed, creating a gap between the ideal detection rate (ASR = 1.0) and the post-transformation ASR. We confirm this empirically in §6.2, where ASR drops from 1.0 to as low as 0.003 under noise injection.

4.3 Limitation 2: Relationship Between Threshold Sensitivities and FPR

The natural attacker response to Limitation 1 is to relax the detection threshold by increasing the sensitivity parameter N from §4.1.1, so that partially corrupted triggers still fire the detector. We now show that this strategy is inherently self-defeating: the same threshold relaxation that recovers ASR also admits more benign inputs as false positives.

Definition 4.4 (Triggered Response Distribution). For a triggered input x_τ subjected to a transformation $\mathcal{A}$, the score $\phi(\mathcal{A}(x_\tau))$ is a random variable whose distribution depends on $\mathcal{A}$. We denote its CDF as $F^{\mathcal{A}}_{\text{trig}}(r) = \Pr[\phi(\mathcal{A}(x_\tau)) \le r]$. When $\mathcal{A}$ is the identity, $F^{\mathcal{A}}_{\text{trig}}$ is degenerate at $R_{\text{ideal}} = N$; for non-identity $\mathcal{A}$ it spreads out and shifts leftward (Limitation 1).

4.3.1 Monotonic Coupling of ASR and FPR. Recall from §4.1.1 that the detector fires when $\langle W, x_{\text{patch}}\rangle > \lambda - N$, i.e., when the score $\phi(x) = \max_{\text{patches}}\langle W, x_{\text{patch}}\rangle + b$ exceeds 0. Both the attack success rate (ASR) and the false positive rate (FPR) are determined by this threshold:

$$\text{ASR}(N, \mathcal{A}) = \Pr[\phi(\mathcal{A}(x_\tau)) > 0] = 1 - F^{\mathcal{A}}_{\text{trig}}(\lambda - N) \quad (11)$$

$$\text{FPR}(N) = \Pr[\phi(x) > 0 \mid x \sim \mathcal{D}] = 1 - F_{\text{benign}}(\lambda - N) \quad (12)$$

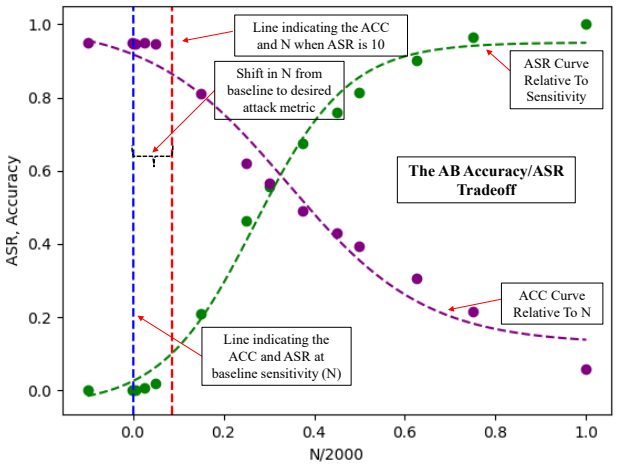


Figure 2: Example trade-off between ASR (green) and ACC (purple) as sensitivity N is varied under a Contrast transform. The red line indicates the N required to achieve ASR > 10%. ΔN shows the shift from baseline N.

where F_{benign} is the benign response CDF from Definition 4.2 and $\lambda = \langle W, T\rangle$ is the ideal trigger response. Both CDFs are non-decreasing, and $\lambda - N$ is decreasing in N, so both ASR and FPR are non-decreasing functions of N. Increasing the sensitivity to recover ASR therefore necessarily raises FPR. When transformations are applied, $F^{\mathcal{A}}_{\text{trig}}$ shifts leftward (Limitation 1) and may overlap with F_{benign}. In the overlap region, no threshold N exists that simultaneously achieves high ASR and low FPR. Raising N to capture more trigger-containing inputs unavoidably triggers the detector on more benign inputs as well.

The Sigmoid Relationship. The empirical sigmoid shape of ASR and ACC curves as functions of N (see §6.5, Figure 2) follows directly from Equation 11 and Equation 12. If $F^{\mathcal{A}}_{\text{trig}}$ and F_{benign} are approximately Gaussian, with means $\mu_{\text{trig}}, \mu_{\text{benign}}$ and standard deviations $\sigma_{\text{trig}}, \sigma_{\text{benign}}$, then:

$$\text{ASR}(N) = \Phi\left(\frac{\mu_{\text{trig}} - (\lambda - N)}{\sigma_{\text{trig}}}\right), \quad \text{FPR}(N) = \Phi\left(\frac{\mu_{\text{benign}} - (\lambda - N)}{\sigma_{\text{benign}}}\right) \quad (13)$$

where Φ is the standard normal CDF. The Gaussian assumption is justified because the score is a max over many patch-level inner products, each itself a sum of many pixel–weight products; the central limit theorem applies to the per-patch sums, and the max is well-approximated by a Gaussian over moderate sample sizes. The separation $d' = (\mu_{\text{trig}} - \mu_{\text{benign}})/\sqrt{\sigma^2_{\text{trig}} + \sigma^2_{\text{benign}}}$ characterizes the *discriminability* of the detector. Transformations decrease μ_{trig} and increase σ_{trig}, shrinking d' and tightening the ASR–FPR coupling.

Key Takeaway. The sensitivity parameter N controls a single threshold that governs both ASR and FPR simultaneously. Because transformations push the triggered and benign response distributions closer together, there is a fundamental region of the sensitivity space where high ASR and low FPR are mutually exclusive. Any attempt to recover ASR lost to transformations by increasing N necessarily raises FPR and degrades benign accuracy,

creating an observable anomaly (and potentially alerting a user that their model is backdoored).

4.4 Limitation 3: Trigger Complexity and the Collision Problem

We now show that not only are trigger detectors sensitive to transformations, but as the trigger pattern itself grows more complex, the set of benign inputs that falsely activate the detector grows with it.

Definition 4.5 (Trigger Complexity). We characterize the complexity of a trigger $T \in [0,1]^d$ (flattened to $d = C \cdot k \cdot k$ dimensions, where C is the channel count and k the spatial size) by its *value complexity* $\kappa_v(T) = \frac{1}{d}\left|\{i : T_i \notin \{0,1\}\}\right|$: the proportion of trigger entries that take non-extreme values (e.g., 0 or 255). A binary trigger (e.g., a black-and-white square) has $\kappa_v = 0$; a trigger with many intermediate intensities has κ_v close to 1.

4.4.1 The Collision Half-Space. Recall from §4.1.1 that the detector fires on a patch x_{patch} when $\langle W, x_{\text{patch}} \rangle > \lambda - N$. The set of benign patches that produce a false positive is therefore the half-space:

$$\mathcal{H}(W,N) = \{x \in [0,1]^d : \langle W, x \rangle > \lambda - N\} \tag{14}$$

and $\text{FPR}(W,N) = \Pr_{x \sim \mathcal{D}_{\text{patch}}}[x \in \mathcal{H}(W,N)]$, where $\mathcal{D}_{\text{patch}}$ is the marginal distribution of $k \times k$ patches in natural images. The volume of $\mathcal{H}$ within the unit hypercube depends on the direction of W, which is determined by T.

When T is binary, W is axis-aligned: the constraint reduces to $\sum_{i \in S} x_i > |S| - N$ over the support $S = \{i : T_i = 1\}$, requiring a benign patch to be near-white at exactly the trigger-active positions. Such patches are rare in natural images. As κ_v grows, W tilts oblique to the hypercube axes, and the constraint $\sum_i w_i x_i > \lambda - N$ admits many more combinations of x_i: a patch that is bright where T is dim and dim where T is bright can still produce a high inner product if magnitudes compensate. The half-space volume grows accordingly.

We can make this precise by examining the variance of the response distribution. For a patch X drawn uniformly from $[0,1]^d$, $\text{Var}(\langle W, X \rangle) = \|W\|_2^2/12$, where $\|W\|_2 = \sqrt{\sum_i W_i^2}$ is the L_2 norm of the kernel. By the Cauchy-Schwarz inequality, for fixed $\|W\|_1 = \lambda$, the L_2 norm satisfies $\|W\|_2^2 \geq \lambda^2/d$ with equality when all entries of W are equal, i.e., $W = (\lambda/d)\mathbf{1}$. Binary triggers (sparse extremes) maximize $\|W\|_2$ and spread the response distribution out, while complex triggers in general lower $\|W\|_2$ relative to binary ones, with uniform-entry triggers achieving the minimum and concentrating the distribution most tightly around its mean $\lambda/2$. A tightly-concentrated distribution centered at $\lambda/2$ pushes more probability mass over the threshold $\lambda - N$ than a spread-out one, raising FPR. Uniform-entry triggers thus represent the worst case among complex triggers; concretely, for $W = (v/d)\mathbf{1}$ (the case in our evaluation), the constraint reduces to $\bar{x} > (\lambda - N)/v$ where $\bar{x}$ is the mean patch intensity. In practice on natural images, the effect persists and is observable empirically (§6.6)

Key Takeaway. Trigger complexity and detection reliability are inversely related. As κ_v grows, the false-positive half-space $\mathcal{H}(W,N)$ expands and benign accuracy degrades sharply. Combined with Limitations 1 and 2, attackers cannot improve the ASR-accuracy trade-off space by designing more sophisticated triggers. We validate this empirically in §6.6.

Algorithm 1: FAULT: AB Insertion and Sensitivity Sweep

Input: Trigger T, Model f, Transforms $\mathcal{A}$, Dataset D, Target ASR ASR_{tgt}
Output: Backdoored model f_{AB} with selected N^*

1 $W \leftarrow \texttt{InitFilter}(T)$; $\lambda \leftarrow \sum_{c,r,s} \max(W_{c,r,s}, 0)$
2 $f_{AB} \leftarrow \texttt{InsertDetector}(f, W, \cdot)$
▷ Sweep candidate sensitivities over a range wide enough to contain the curve inflection
3 $\mathcal{N} \leftarrow \text{linspace}(0,\ N_{up},\ m)$; Results $\leftarrow \emptyset$
4 **foreach** $N \in \mathcal{N}$ **do**
5 $\quad b \leftarrow -(\lambda - N)$
6 $\quad (\text{ASR}, \text{ACC}) \leftarrow \texttt{Evaluate}(f_{AB}, D, \mathcal{A}, W, b)$
7 $\quad \text{Results} \leftarrow \text{Results} \cup \{(N, \text{ASR}, \text{ACC})\}$
8 **end**
▷ Identify N_{max} as the inflection point of the fitted ACC curve
9 $\text{ACC}_{fit} \leftarrow \texttt{FitLogistic}(\text{Results})$; $N_{max} \leftarrow \arg\max_N |\text{ACC}'_{fit}(N)|$
▷ Select smallest $N \leq N_{max}$ achieving the target ASR
10 $N^* \leftarrow \texttt{SelectOptimal}(\text{Results}, N_{max}, ASR_{tgt})$; $b^* \leftarrow -(\lambda - N^*)$
11 **return** f_{AB} *with* b^*

5 FAULT

§4 establishes that AB trigger detection is bounded by three fundamental limitations, but does not say where any specific AB sits within the resulting operating space. FAULT is the evaluation framework that traces this space empirically. Given an AB construction and a set of preprocessing transformations, FAULT characterizes the achievable (ASR, ACC, FPR) operating curve as a function of the sensitivity parameter N. The framework simulates a strong attacker (that can tune N for any chosen goal) and reports the operating points the attacker can actually reach.

FAULT does not assume detailed knowledge of the target model's internal architecture beyond the ability to insert an AB (corresponding to a predefined trigger pattern) into the model definition. The AB's output integration depends on the chosen signal propagation path (e.g., SeP, IP, ShP).

5.1 Utilizing FAULT

FAULT's AB tuning procedure is detailed in Algorithm 1. It takes the trigger pattern T, model f, transformations $\mathcal{A}$, dataset D, and performance constraints as input, and iteratively evaluates the AB across a sensitivity range to trace the ASR-Accuracy trade-off curve.

The algorithm begins by initializing the filter kernel W from T (Line 1) and computing its intensity λ (Line 1). The AB detector is inserted into f (Line 2), as in prior work [1, 2]. FAULT then sweeps a set of candidate sensitivities $\mathcal{N}$ in $[0, N_{up}]$ (Line 3), where N_{up} is set wide enough to bracket the inflection of the ACC curve. For each candidate N, FAULT computes the corresponding bias $b = -(\lambda - N)$

(Line 5) and evaluates ASR and ACC on the transformed dataset $D_{\mathcal{A}}$ (Line 6), recording the results (Line 7).

Once the sweep is complete, FAULT fits a logistic regression to the collected (ASR, ACC) data and identifies N_{max} as the inflection point of the fitted ACC curve (Line 9). The operating point N^* is then selected from this fitted region (Line 10), the model is updated with the corresponding bias (Line 10), and the configured backdoored model f_{AB} is returned (Line 11).

5.2 Algorithmic Choices

We now describe the three algorithmic choices that connect Algorithm 1 to the formal results of §4.

Bounding the Sweep Range. FAULT sweeps a range $[0, N_{up}]$ wide enough to contain the inflection of the ACC curve. The lower bound $N_{min} = 0$ corresponds to $b = -\lambda$, where only an exact trigger match fires the detector ($R_{\text{ideal}} = N$ from Theorem 4.1); N_{up} is set conservatively (we use $N_{up} = 1$ in practice) and only needs to exceed the inflection point so that the fitted curve captures it. Within this range, FAULT evaluates the model at each candidate N and records the resulting (ASR, ACC) pair.

Identifying the Operating Ceiling. §4 predicts that ASR and FPR follow Gaussian CDF curves in N (Equation 13). FAULT fits a logistic regression (smooth approximation of the Gaussian CDF) to the swept (ASR, ACC) data and locates N_{max} as the inflection point of the fitted ACC curve, which captures the sensitivity threshold for which increasing N accelerates accuracy degradation without commensurate ASR gain. Figure 2 shows an example fit under a Contrast transformation: green for ASR, purple for ACC, with N_{max} marking the ACC inflection.

Selecting the Operating Point. The fitted curves expose a natural ceiling on attacker capability. Beyond N_{max}, additional sensitivity yields diminishing ASR gains while accelerating ACC degradation. The `SelectOptimal` function (Line 10) therefore returns the smallest $N \in [0, N_{max}]$ achieving ASR $\geq \text{ASR}_{tgt}$, or N_{max} itself if no such N exists in the range. By varying ASR_{tgt}, FAULT traces three operating points along the achievable curve: a minimal-threshold point where the AB just begins firing, a balanced midpoint, and the inflection N_{max} itself. We use these three points in §6 to characterize the boundary of what an attacker tuning N optimally can achieve.

6 AB Evaluation with FAULT

Our prototype implementation of FAULT consists of ~5000 lines of code implementing and utilizing existing ABs. We utilize FAULT to measure the effectiveness of ABs when input-space perturbations are applied, the AB threshold is adjusted, and the AB trigger's complexity is varied. FAULT is fully modular (new trigger types, ABs, and transformations can be easily added).

6.1 Experiment Setup

Datasets, Models, and Triggers. We evaluate FAULT across three datasets and corresponding model architectures commonly used in literature. For CIFAR10 [19], we use ResNet18 [40]; for the German Traffic Sign Recognition Benchmark (GTSRB) [18], we use MobileNetV2 [41]; for Mapillary [20], we use YoloV11 [42]. ABs are integrated into the model's architecture using the convolution AB construction with "perfect trigger" detectors [2]. Signal propagation is irrelevant to the trigger detection step of an AB, but our ABs utilize one of three signal-propagation variants from prior work [2]: Separate Path (SeP), Interleaved Path (IP), and Shared Path (ShP). We use a black-square trigger (BS) for CIFAR10 and Mapillary, and a red-square trigger (RS) for GTSRB. Specific (trigger, propagation) combinations reported in each table are noted as e.g. "BS-SeP" or "RS-ShP".

Sensitivity Sweep. For each (model, trigger, AB) configuration, we use FAULT (Algorithm 1) to sweep the sensitivity parameter N across a wide range, $N \in [-10000, 2000]$, scaled to $[-5, 1]$ in all plots.

Poisoned Dataset Preparation. For each test set, we create a poisoned variant by embedding the trigger pattern into 100% of the images. Because AB attacks do not rely on data poisoning at training time, we use the poisoned set solely to measure AB activation rates (ASR), not to train backdoored models.

Transformations. We consider ten input-space transformations applied as a preprocessing step before inference, falling into two categories. *Natural transformations* are common image-processing operations routinely used for data augmentation or robustness [15, 36, 39]: rotation, contrast adjustment, cropping, flipping, and grayscale conversion. *Unnatural transformations* introduce intentional perturbations to obscure input features, motivated by adversarial-defense literature: noise injection [44, 45], Gaussian blurring [43], denoising [16], and median filtering [16, 17]. We calibrate the intensity of each transformation (parameters in §A.2) to avoid overly degrading benign accuracy while still impeding AB activation.

Evaluation Metrics and Plot Conventions. We track model accuracy on benign data (ACC), attack success rate (ASR), attack precision (A-PC), and attack recall (A-RC). To visualize the relationship between ASR/ACC and N, we fit logistic regression curves to the swept data points for each transformation (Figure 2 shows an example). Two reference points appear on each plot: the **blue line** marks the smallest N at which ASR/ACC under the transformation matches the no-transformation baseline; the **red line** marks the smallest N at which ASR exceeds 10%. Across all configurations we tested, this 10% line is near the upper boundary of what an attacker can achieve. Pushing N further to chase higher ASR crosses the inflection point of the ACC curve, where benign accuracy degrades observably. The 10-20% ASR ceiling we report throughout the evaluation is therefore not a design choice but a measurement of the attacker's achievable region.

6.2 Limitation 1: Transformations Defeat AB Triggers

We first measure the impact of input-space transformations on AB activation, empirically validating Limitation 1 (§4.2). For each (model, trigger, AB) configuration from §6.1, we evaluate the AB on benign and poisoned test inputs to establish Pre-Transform metrics, then apply each transformation and re-evaluate to obtain Post-Transform metrics. Table 1 reports the results.

Table 1: FAULT's Evaluation of AB Performance Given Natural Transformations Applied During Pre-processing.

Dataset	Model	AB	Transform	Pre-Transform Model Metrics				Post-Transform Model Metrics			
				Accuracy (%)	ASR	A-Precision	A-Recall	Accuracy (%)	ASR	A-Precision	A-Recall
GTSRB [18]	MobileNetV2	None	None	96.0	**0.01**	0.51	5E-4	96.0	**0.01**	0.51	5E-4
GTSRB	MobileNetV2	RS-ShP[3]	None	93.8	**1.0**	0.83	0.14	93.8	**0.14**	0.83	0.14
GTSRB	MobileNetV2	RS-ShP	Contrast [36]	93.8	**1.0**	0.83	0.14	95.8	**4.1E-4**	0.29	4.1E-4
GTSRB	MobileNetV2	RS-ShP	Crop [36]	93.8	**1.0**	0.83	0.14	94.7	**3.0E-4**	0.36	3.0E-4
GTSRB	MobileNetV2	RS-ShP	Flip [36]	93.8	**1.0**	0.83	0.14	48.9	**0.18**	0.67	0.18
GTSRB	MobileNetV2	RS-ShP	Gaussian Blur [43]	93.8	**1.0**	0.83	0.14	93.2	**0.04**	0.69	0.04
GTSRB	MobileNetV2	RS-ShP	Grayscale [36]	93.8	**1.0**	0.83	0.14	88.8	**2.5E-3**	0.41	2.5E-3
GTSRB	MobileNetV2	RS-ShP	Median Filter [17]	93.8	**1.0**	0.83	0.14	45.8	**0.02**	0.49	0.02
GTSRB	MobileNetV2	RS-ShP	Noise [44]	93.8	**1.0**	0.83	0.14	95.4	**4.0E-4**	0.26	4.0E-4
GTSRB	MobileNetV2	RS-ShP	Rotation [36]	93.8	**1.0**	0.83	0.14	95.4	**4.0E-4**	0.26	4.0E-4
GTSRB	MobileNetV2	RS-ShP	Denoising [17]	93.8	**1.0**	0.83	0.14	93.1	**0.04**	0.71	0.04
Mapillary [20]	YoloV11 [42]	None	None	89.0	**0.013**	0.72	0.014	89.0	**0.013**	0.72	0.014
Mapillary	YoloV11	BS-IP[3]	None	87.5	**1.0**	0.95	1.0	87.5	**1.0**	0.95	1.0
Mapillary	YoloV11	BS-IP	Contrast	87.5	**1.0**	0.95	1.0	88.2	**0.016**	0.75	0.015
Mapillary	YoloV11	BS-IP	Crop	87.5	**1.0**	0.95	1.0	86.5	**0.99**	0.95	0.99
Mapillary	YoloV11	BS-IP	Flip	87.5	**1.0**	0.95	1.0	86.8	**1.0**	0.95	1.0
Mapillary	YoloV11	BS-IP	Gaussian Blur	87.5	**1.0**	0.95	1.0	85.0	**0.75**	0.92	0.71
Mapillary	YoloV11	BS-IP	Grayscale	87.5	**1.0**	0.95	1.0	67.5	**1.0**	0.92	1.0
Mapillary	YoloV11	BS-IP	Median Filter	87.5	**1.0**	0.95	1.0	76.3	**1.0**	0.95	1.0
Mapillary	YoloV11	BS-IP	Noise	87.5	**1.0**	0.95	1.0	87.2	**0.003**	1.0	0.003
Mapillary	YoloV11	BS-IP	Rotation	87.5	**1.0**	0.95	1.0	8E-3	**1.0**	0.25	1.0
Mapillary	YoloV11	BS-IP	Denoising	87.5	**1.0**	0.95	1.0	82.7	**0.04**	0.54	0.05
CIFAR10 [19]	Resnet [40]	None	None	92.5	**0.01**	0.51	0.013	92.5	**0.01**	0.51	0.013
CIFAR10	Resnet	BS-SeP[3]	None	89.6	**1.0**	0.95	1.0	89.6	**1.0**	0.95	1.0
CIFAR10	Resnet	BS-SeP	Contrast	89.6	**1.0**	0.95	1.0	90.6	**0.01**	0.54	0.01
CIFAR10	Resnet	BS-SeP	Crop	89.6	**1.0**	0.95	1.0	84.2	**0.03**	0.53	0.03
CIFAR10	Resnet	BS-SeP	Flip	89.6	**1.0**	0.95	1.0	92.1	**1.0**	0.98	1.0
CIFAR10	Resnet	BS-SeP	Gaussian Blur	89.6	**1.0**	0.95	1.0	55.4	**0.24**	0.72	0.24
CIFAR10	Resnet	BS-SeP	Grayscale	89.6	**1.0**	0.95	1.0	82.4	**1.0**	0.92	1.0
CIFAR10	Resnet	BS-SeP	Median Filter	89.6	**1.0**	0.95	1.0	37.9	**1.0**	0.93	1.0
CIFAR10	Resnet	BS-SeP	Noise	89.6	**1.0**	0.95	1.0	90.2	**1E-2**	0.53	1E-2
CIFAR10	Resnet	BS-SeP	Rotation	89.6	**1.0**	0.95	1.0	86.3	**0.02**	0.51	0.02
CIFAR10	Resnet	BS-SeP	Denoising	89.6	**1.0**	0.95	1.0	86.9	**2E-3**	0.52	2E-3

Transformations Cause ASR Loss Across All Settings. We observe that the ASR magnitude collapses under most transformations. On CIFAR10, contrast adjustment drops ASR from 1.0 to 0.01 (100× reduction); noise injection reduces ASR to 0.01, and denoising reduces ASR to 0.002. GTSRB shows the same pattern at smaller magnitudes due to its lower Pre-Transform ASR baseline: contrast, crop, noise, and rotation each drop ASR to ∼4e−4. ABs deployed in YoloV11 on Mapillary are the only ABs that occasionally retain high ASR (under flip and grayscale), but the same configuration collapses to 0.003 ASR under noise and 0.016 under contrast.

These collapses confirm the prediction of §4.2: the deviation $\Delta_{\mathcal{A}}$ introduced by transformations drives the trigger response below the detection threshold, and existing ABs (which use a fixed sensitivity tuned for the no-transformation case) have no recourse.

No Single Transformation is Universally Effective. While transformations broadly defeat ABs, no single transformation works against every configuration. Grayscale leaves CIFAR10 and Mapillary ASR at 1.0 (the trigger does not depend on color in those settings), while it collapses GTSRB ASR (where the red-square trigger is color-dependent). Flip preserves ASR across all three datasets because the triggers are symmetric and are unmodified post-flip. Crop fails on Mapillary's center-placed trigger (ASR 0.99) but succeeds on CIFAR10's corner-placed trigger (0.03). The pattern of which transformations succeed depends on which trigger features they corrupt (e.g., a gray-scale transform only defeats color-encoded triggers).

This dependence is consistent with our formal analysis: $\Delta_{\mathcal{A}}$ depends on how a particular transformation interacts with the kernel W. The takeaway is that a defender cannot rely on any single transformation, but a small set covering geometric, color, and noise dimensions is sufficient to break every AB configuration we tested.

Some Transformations Degrade Benign Accuracy. A subset of transformations reduce ASR by simultaneously degrading model performance on benign inputs. Median filtering drops ACC from 93.8% to 45.8% on GTSRB, and from 89.6% to 37.9% on CIFAR10.

Gaussian blur on CIFAR10 reduces ACC to 55.4%. These are not viable defenses for these testing configurations. However, they still bound the space of defender choices. Even aggressive transformations that compromise utility cannot always defeat the AB (median filter on CIFAR10 retains ASR 1.0).

Ultimately, ABs deployed with a fixed sensitivity are uniformly vulnerable to input-space transformations, validating Limitation 1 empirically. The natural attacker response is to raise N to compensate, which we evaluate in the next subsection.

6.3 Limitation 2: Sensitivity Tuning Trades ASR for Accuracy

We now operationalize §4.3 empirically by tracing the full ASR-ACC operating curve as a function of N. Figure 3 shows the curves for the BS-SeP AB on YoloV11 trained on Mapillary, both without preprocessing (Figure 3a) and under noise (Figure 3b), contrast (Figure 3c), and variation denoising (Figure 3d) transformations. Each subfigure plots ASR (green) and ACC (purple) as N varies across $[-10000, 2000]$, with the logistic regression fits superimposed.

Transformations Force the Attacker into a Trade-off Corner. Each transformation shifts the ASR curve rightward: the attacker must raise N to recover any meaningful ASR. The cost of doing so, however, varies sharply across transformations and is determined by how quickly the ACC curve degrades. At an attacker-desired ASR of 10%, contrast permits ACC of 85.1% (at $N = 0.083$), while noise collapses ACC to 69.6% (at $N = 0.063$); noise additionally degrades ACC by 5–7% across *all* sensitivity values, before any tuning has occurred. Variation denoising sits between these: while it allows ASR to rise at very low $N = 0.017$, its ACC curve drops steeply across the same range, leaving little room before the inflection point of §5.2. In every case, the ASR and ACC curves are coupled exactly as predicted by Equation 13. Raising N to recover ASR drives ACC down a parallel curve, with the two distributions overlapping in the region the attacker most wants to occupy.

Across all three transformations we observe that no value of N achieves both high ASR and ACC near baseline. The widest gap between the curves (the closest thing to a viable operating point) is contrast, where reaching 10% ASR costs ~ 3% ACC. Noise and variation denoising leave smaller gaps still. We now use FAULT to characterize this achievable region quantitatively across every (model, trigger, transformation) configuration.

6.4 Sensitivity Tuning Cannot Escape the Trade-off

§6.3 showed the operating curve for a single configuration. We now characterize the curve quantitatively across all (model, trigger, transformation) configurations to test whether sensitivity tuning can systematically recover ASR. Table 2 reports the most favorable point on each curve: the value of N that maximizes ASR while keeping ACC degradation below the inflection point N_{max} (the upper edge of the achievable region from §5.2). Columns 1-3 identify the configuration; Columns 4-5 give the no-tuning baseline (ACC, ASR); Columns 6-8 report the FAULT-tuned values (decrease in ACC, post-tuning ASR, and the multiplicative ASR increase ΔASR (×)).

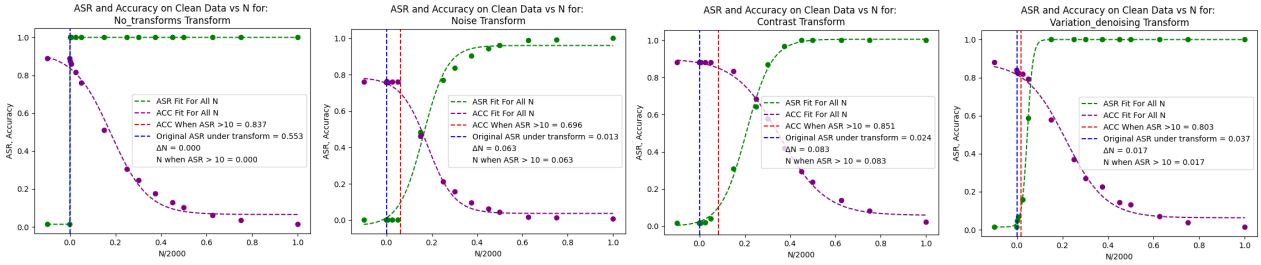
Table 2: Sensitivity Tuning by FAULT to Increase Attack Effectiveness.

Dataset	AB	Transform	Attack Effectiveness				
			No Tuning		Tuned with FAULT		
			ACC	ASR	Δ ACC	ASR	ΔASR (×)
GTSRB [18]	MNV2-RS-SeP	None	0.9266	0.0456	0.0809	0.2139	**4.6936**
		Con.	0.9403	0.0096	0.0909	0.1614	**16.8456**
		Crop	0.9162	0.0272	0.0839	0.2094	**7.7082**
		Rot.	0.8704	0.0637	0.0682	0.2060	**3.2350**
		Flip	0.4735	0.0483	0.0351	0.1643	**3.4038**
		G. Blur	0.9180	0.0248	0.0833	0.1495	**6.0355**
		Gray.	0.8582	0.0369	0.0757	0.1283	**3.4750**
		M-F	0.4519	0.0178	0.0347	0.0931	**5.2183**
		Noise	0.9326	0.0044	0.0940	0.1512	**34.6220**
		V-D	0.9146	0.0207	0.0815	0.1651	**7.9725**
Mapillary [20]	YV11-BS-IP	None	0.8375	0.5528	0.0548	1.0000	**1.8089**
		Con.	0.8787	0.0244	0.0799	0.3022	**12.3816**
		Crop	0.8274	0.0396	0.0567	0.9987	**25.2202**
		Rot.	0.4244	0.5528	-0.0439	0.0140	**0.0253**
		Flip	0.8300	0.5357	0.0543	1.0000	**1.8668**
		G. Blur	0.8226	0.0492	0.0574	0.9723	**19.7498**
		Gray.	0.6534	0.4938	0.0471	1.0000	**2.0250**
		M-F	0.7383	0.4938	0.0561	1.0000	**2.0250**
		Noise	0.7520	0.0131	0.0543	0.0977	**7.4749**
		V-D	0.8156	0.0367	0.0630	0.8557	**23.3201**
CIFAR10 [19]	Resnet-BS-SeP	None	0.9005	0.5482	0.0801	1.0000	**1.8241**
		Con.	0.9162	0.0122	0.0690	0.1684	**13.7735**
		Crop	0.8276	0.0602	0.0732	0.1634	**2.7129**
		Rot.	0.8454	0.0498	0.0779	0.1611	**3.2337**
		Flip	0.9006	0.5228	0.0801	1.0000	**1.9127**
		G. Blur	0.5587	0.1994	0.0484	0.3991	**2.0012**
		Gray.	0.8220	0.5100	0.0705	1.0000	**1.9606**
		M-F	0.3751	0.9903	0.0212	1.0000	**1.0098**
		Noise	0.9067	0.0070	0.0489	0.1139	**16.2808**
		V-D	0.8565	0.0442	0.0762	0.5004	**11.3298**

Sensitivity Tuning Recovers ASR (at a Cost). On the GTSRB MNV2-RS-SeP configuration (rows 1-10), no-tuning ASR averages just 0.030 across transformations, indicating that the AB has effectively been disabled by every preprocessing step. Tuning recovers ASR substantially in raw multiplicative terms, an average 9.6× increase, peaking at 34.6× for noise. However the absolute ASR ceiling remains low, post-tuning ASRs cluster between 0.09 and 0.21, and the corresponding ACC degrades by an average of 7.4% (a large enough decrease to heavily reduce attack stealth). The Mapillary YV11-BS-IP and CIFAR10 Resnet-BS-SeP configurations behave similarly, but differ in scale. FAULT-tuning lifts ASRs to 0.72 and 0.55 on average, but at the cost of average ACC drops of 5.5% and 6.5%. Across all three configurations, every multiplicative gain in ASR is paid for in observable ACC reduction.

Tuning Does Not Always Recover ASR. For configurations where the transformation breaks features the trigger depends on, tuning fails outright. For GTSRB noise reaches only ASR 0.15 even at N_{max}, with ACC dropping 9.4%. CIFAR10 Median Filter reaches ASR 1.0 but ACC has already collapsed to 37.5% before tuning begins. On Mapillary, both Rotation and Crop yield post-tuning ASRs of 0.014 and 0.999 respectively. This is because rotation moves the trigger out of the receptive field entirely, and crop happens to leave the centered trigger intact. The mapping from transformation to recoverable-ASR ceiling is determined by the transformation's interaction with the specific trigger features, consistent with the analysis in §4.2.

Ultimately, sensitivity tuning provides the attacker no way to avoid Limitation 2. Where the trigger features survive a transformation intact, tuning can lift ASR back up but at a measurable ACC cost. Where the trigger features do not survive, even unlimited sensitivity ($N = N_{max}$) cannot restore the attack. In



(a) No Transforms Applied. **(b) Noise Transform.** **(c) Contrast.** **(d) Variation Denoising.**

Figure 3: Sensitivity parameter $N \in [-10000, 2000]$ across all transforms applied to the BS-SeP AB for YoloV11 trained on Mapillary. The green/purple fitted curves are logistic regressions applied to all ASRs / ACCs, respectively, as a function of N. The blue line highlights the shift in N necessary to increase attack success rate but avoid drops in model performance relative to no transformation applied. The red line highlights the model accuracy at the first point where the ASR is above 10%.

neither regime can the attacker simultaneously achieve high ASR and undetectable accuracy degradation.

6.5 Operating Points Across Attacker Goals

§6.4 reported the ASR ceiling reachable at the inflection point N_{max}. We now characterize the full operating range $[0, N_{max}]$ by reporting three points along the achievable curve, parameterized by the attacker's tolerated ACC degradation. Table 3 shows these three operating points across all (model, trigger, transformation) configurations. Columns 5-6 show **Min Threshold** ($ASR_{tgt} = 0.10$) which is the operating point corresponding to the smallest N achieving an ASR of at least 10%. Columns 7-8 show **Mid Range** which is an intermediate N between Min Threshold and the inflection point, representing a middle-ground operating point. Finally, Columns 9-10 show **At Inflection** ($N = N_{max}$) which corresponds to the sensitivity at the inflection point of the ACC curve, beyond which additional N accelerates accuracy degradation. This is the ceiling of the achievable region.

Across all 30 (configuration, transformation) experiments, FAULT identifies an N^* that achieves ASR ≥ 0.10. The cost of achieving this ASR is bounded. The average ACC decrease across all experiments is 2.23%, with 29 of 30 experiments staying within a 6% ACC degradation. The single failure case, GTSRB with Noise, ΔACC = 6.7% is consistent with the analysis in §6.4. Noise produces the steepest ACC decline in N across configurations, so reaching even the modest 10% ASR target requires the largest ΔN (0.41 on CIFAR10-Noise vs. an average of 0.06 across configurations). In other words, attackers *can* achieve a 10% ASR target on virtually every configuration, but doing so against the strongest transformations (noise) pushes them to the edge of an observable accuracy budget.

The three operating points show how much ASR an attacker buys by trading additional ACC. On GTSRB, the three points yield ASRs of $[0.100, 0.130, 0.164]$ on average (which means that the At Inflection operating point is only 1.6× above the Min Threshold floor). Mapillary and CIFAR10 show wider ranges ($[0.621, 0.672, 0.724]$ and $[0.417, 0.499, 0.551]$ respectively), but the Mapillary numbers are inflated by configurations (Crop, Flip, Gray, M-F) where the transformation does not corrupt the trigger and ASR is already near 1.0 at low N. In those cases there is no "operating range", because no tuning is needed. When the transformation does corrupt the trigger, such as with Contrast on GTSRB ($[0.100, 0.151, 0.161]$), Noise on CIFAR10 ($[0.100, 0.107, 0.114]$), and Variation Denoising on GTSRB ($[0.100, 0.130, 0.165]$), the operating range is consistently narrow, and the At Inflection ceiling ASR rarely exceeds 0.20.

Across the achievable region $[0, N_{max}]$, FAULT exposes a consistent pattern: where transformations meaningfully impede the trigger, the attacker's reachable ASR ceiling is bounded between roughly 10% and 20%, and pushing for higher ASR within this range requires a measurable ACC cost. The achievable region exists, but it is narrow, and the negative result of §6.3 holds quantitatively across every configuration tested.

Table 3: FAULT Evaluates AB Performance at Three Operating Points.

Dataset	AB	TF[1]	Pre-FAULT ASR	FAULT-ASRs For Varied Attacker Goals					
				Min Threshold		Mid Range		At Inflection	
				Sens.	ASR	Sens.	ASR	Sens.	ASR
GTSRB [18]	MNV2-RS-SeP	None	0.046	0.048	0.100	0.080	0.151	0.112	0.214
		Con.	0.010	0.112	0.100	0.132	0.129	0.153	0.161
		Crop	0.027	0.066	0.100	0.095	0.149	0.124	0.209
		Rot.	0.064	0.029	0.100	0.058	0.148	0.088	0.206
		Flip	0.048	0.046	0.100	0.066	0.130	0.086	0.164
		G. Blur	0.025	0.086	0.100	0.104	0.123	0.123	0.150
		Gray.	0.037	0.084	0.100	0.098	0.114	0.112	0.128
		M-F	0.018	0.094	0.100	0.091	0.097	0.089	0.093
		Noise	0.004	0.121	0.100	0.138	0.124	0.154	0.151
		V-D	0.021	0.080	0.100	0.100	0.130	0.120	0.165
Mapillary [20]	YV11-BS-IP	None	0.553	1^{-4}	0.986	0.023	1.000	0.046	1.000
		Con.	0.024	0.083	0.100	0.121	0.180	0.159	0.302
		Crop	0.040	1^{-4}	0.599	0.026	0.999	0.051	0.999
		Rot.	0.553	1^{-4}	0.986	-1^{-4}	0.035	-1^{-4}	0.014
		Flip	0.536	1^{-4}	0.987	0.023	1.000	0.046	1.000
		G. Blur	0.049	1^{-4}	0.381	0.027	0.972	0.054	0.972
		Gray.	0.494	1^{-4}	0.986	0.028	1.000	0.056	1.000
		M-F	0.494	1^{-4}	0.986	0.032	1.000	0.064	1.000
		Noise	0.013	0.063	0.100	0.062	0.099	0.062	0.098
		V-D	0.037	0.017	0.101	0.043	0.437	0.068	0.856
CIFAR10 [19]	Resnet-BS-SeP	None	0.548	1^{-4}	0.985	0.089	1.000	0.178	1.000
		Con.	0.012	0.259	0.100	0.296	0.131	0.333	0.168
		Crop	0.060	0.094	0.100	0.142	0.129	0.190	0.163
		Rot.	0.050	0.087	0.100	0.122	0.128	0.157	0.161
		Flip	0.523	1^{-4}	0.984	0.089	1.000	0.178	1.000
		G. Blur	0.199	-0.100	0.147	0.047	0.235	0.193	0.399
		Gray.	0.510	1^{-4}	0.986	0.072	1.000	0.144	1.000
		M-F	0.990	-1^{-4}	0.566	0.213	1.000	0.427	1.000
		Noise	0.007	0.414	0.100	0.421	0.107	0.429	0.114
		V-D	0.044	0.057	0.100	0.138	0.256	0.218	0.500

1: TF: transform, SV: sensitivity value.

6.6 Limitation 3: Trigger Complexity Degrades Benign Accuracy

We now empirically validate Limitation 3 (§4.4). As the trigger pattern grows more complex, the set of benign inputs that falsely activate the detector grows with it. For each dataset/model pair, we evaluate the baseline (no trigger) alongside three trigger families of increasing value complexity: gray squares (GS) at intensities 255, 192, and 128; checkerboards (CB) at contrast pairs (255/0, 223/32, 192/64); and star gradients (SG) at intensities 255, 192, and 128. Table 4 reports the results.

Across all three datasets, ASR is 1.0 for every trigger variant, mirroring the no-transform baselines reported in prior AB work [1, 2]. The cost of trigger complexity therefore appears entirely on the benign-accuracy side, in the form of false-positive collisions consistent with the half-space analysis in §4.4.

For MobileNetV2 on GTSRB, the baseline model reaches 96.8% accuracy. As trigger complexity rises, accuracy monotonically decreases: GS1 drops to 70.8%, GS2 to 50.0%, and GS3 to 30.5%. Checkerboards follow the same pattern (CB1 75.6% → CB3 42.5%), as do star gradients (SG1 84.9% → SG3 36.8%).

For YoloV11 on Mapillary, we observe the same phenomena. The baseline reaches 89.4%, and GS1 preserves 87.1%, but introducing intermediate values collapses accuracy: GS2 yields 20.6% and GS3 only 9.9%. CB3 and SG3 reach 16.3% and 10.0% respectively. The false-positive counts in Table 4 confirm this problem. For GS3 on Mapillary, every triggered input is detected (TP = 2401) but 517 benign inputs also fire the detector (FP = 517), reducing the model's utility beyond acceptable bounds.

Finally, for Resnet18 on CIFAR10, the same trend holds, with the largest absolute false-positive counts due to the larger test set. The baseline reaches 92.5%, GS1 preserves 84.4%, but GS3 collapses to 10.7% with 9,918 false positives out of 10,000 benign samples. SG1 retains the highest accuracy (89.3%) of any non-baseline trigger, while SG3 falls to 12.6%.

Ultimately, trigger complexity and detector reliability are inversely related, as discussed in §4.4. While the attack remains effective regardless of trigger type (ASR stays at 1.0 across all configurations), the benign accuracy drops from baseline (89–97%) to under 15% on the most complex triggers. This incurred cost to accuracy is enough to alert victims that something is wrong with the model. Attackers are therefore confined to simple, near-binary triggers.

6.7 Evaluating Detector Type with FAULT

We next test whether the trigger-complexity collapse from §6.6 is an artifact of one detector implementation. §4.1.1 argued that all five detector types (convolution, concatenation, pooling, slicing, masking) reduce to the same parameterized inner-product similarity, differing only in how the computation is expressed in the graph. Table 5 tests this empirically. We fix the setup to CIFAR10/ResNet18 and run each detector type against the baseline and three gray-square triggers (GS1, GS2, GS3).

We observe that all Detectors behave equivalently in ASR and baseline ACC. Across all five detector types, ASR is 1.0 for every triggered input, and the baseline accuracy of 96.8% is recovered when no trigger is present. Differences emerge only in false-positive counts and the resulting benign accuracy. At GS1, accuracy ranges from 82.9% (slicing) to 86.2% (pooling) which is a spread of roughly 3.3 points. Pooling consistently produces the fewest false positives and slicing the most across all three trigger complexities.

We also observe that modifying the detector type does not recover lost accuracy from the trigger type. At GS3, every detector produces between 9,712 and 10,093 false positives, and benign

Table 4: ASR/ACC Tradeoff Across Trigger Complexities

Dataset	Model[1]	Trigger[2]	Metrics						
			ACC	ASR	AP	TP	FP	FN	TN
GTSRB	MobileNetV2	None	0.968	0.052	0.753	1983	651	35907	11979
		GS1	0.708	1.000	0.768	12630	3821	0	8809
		GS2	0.500	1.000	0.657	12630	6590	0	6040
		GS3	0.305	1.000	0.581	12630	9123	0	3507
		CB1	0.756	1.000	0.798	12630	3198	0	9432
		CB2	0.550	1.000	0.681	12630	5924	0	6706
		CB3	0.425	1.000	0.626	12630	7545	0	5085
		SG1	0.849	1.000	0.863	12630	2008	0	10622
		SG2	0.584	1.000	0.698	12630	5467	0	7163
		SG3	0.368	1.000	0.604	12630	8285	0	4345
Mapillary	YoloV11	None	0.894	0.156	0.925	37	0	2341	555
		GS1	0.871	1.000	0.992	2398	14	0	541
		GS2	0.206	1.000	0.824	2391	451	0	128
		GS3	0.099	1.000	0.805	2401	517	0	62
		CB1	0.876	1.000	0.850	2385	227	0	345
		CB2	0.216	1.000	0.826	2391	445	0	134
		CB3	0.163	1.000	0.816	2389	478	0	101
		SG1	0.884	1.000	0.870	2330	211	0	361
		SG2	0.208	1.000	0.821	2339	450	0	129
		SG3	0.100	1.000	0.803	2357	517	0	62
CIFAR10	ResNet18	None	0.925	0.100	0.834	4999	992	45001	9008
		GS1	0.844	1.000	0.964	50000	1884	0	8116
		GS2	0.287	1.000	0.863	50000	7939	0	2061
		GS3	0.107	1.000	0.834	50000	9918	0	82
		CB1	0.854	1.000	0.966	50000	1768	0	8232
		CB2	0.387	1.000	0.880	50000	6849	0	3151
		CB3	0.175	1.000	0.845	50000	9174	0	826
		SG1	0.893	1.000	0.974	50000	1337	0	8663
		SG2	0.473	1.000	0.894	50000	5918	0	4082
		SG3	0.126	1.000	0.837	50000	9720	0	280

1: MobileNetV2→ShP; YoloV11→IP; ResNet18→SeP; GS→Gray Square; CB→ Checkerboard;
2: SG→ Star Gradient; GS1→ RGB(255); GS2→ RGB(192); GS3→ RGB(128);
CB1→ RGB(0, 255); CB2→ RGB(0, 192); CB3→ RGB(0, 128);
SG1→ RGB(0, 255); SG2→ RGB(0, 192); SG2→ RGB(0, 128);

Table 5: ACC/ASR Tradeoff Across Detector Types

Dataset	Detector	Trigger[1]	Metrics						
			ACC	ASR	AP	TP	FP	FN	TN
CIFAR10 / ResNet18	Conv	None	0.968	0.052	0.834	3279	651	59871	11979
		GS1	0.844	1.000	0.964	50000	1884	0	8116
		GS2	0.287	1.000	0.863	50000	7939	0	2061
		GS3	0.107	1.000	0.834	50000	9918	0	82
	Concat	None	0.968	0.052	0.834	3279	651	59871	11979
		GS1	0.838	1.000	0.961	50000	1983	0	8017
		GS2	0.274	1.000	0.858	50000	8143	0	1857
		GS3	0.098	1.000	0.831	50000	10047	0	47
	Pool	None	0.968	0.052	0.834	3279	651	59871	11979
		GS1	0.862	1.000	0.969	50000	1601	0	8399
		GS2	0.305	1.000	0.871	50000	7624	0	2376
		GS3	0.119	1.000	0.839	50000	9712	0	288
	Slice	None	0.968	0.052	0.834	3279	651	59871	11979
		GS1	0.829	1.000	0.958	50000	2187	0	7813
		GS2	0.268	1.000	0.855	50000	8274	0	1726
		GS3	0.094	1.000	0.829	50000	10093	0	1
	Mask	None	0.968	0.052	0.834	3279	651	59871	11979
		GS1	0.857	1.000	0.968	50000	1698	0	8302
		GS2	0.296	1.000	0.868	50000	7731	0	2269
		GS3	0.108	1.000	0.835	50000	9871	0	129

1: GS→Gray Square; CB→ Checkerboard; SG→ Star Gradient; GS1→ RGB(255); GS2→ RGB(192);
GS3→ RGB(128); CB1→ RGB(0, 255); CB2→ RGB(0, 192); CB3→ RGB(0, 128);
SG1→ RGB(0, 255); SG2→ RGB(0, 192); SG2→ RGB(0, 128);

accuracy collapses to between 9.4% and 11.9%. The ∼2.5-point gap between the best and worst detector is dominated by the ∼80-point drop from baseline. The false-positive distribution is therefore governed by the geometry of the trigger and the statistics of the natural image distribution, not by the implementation choice.

Ultimately, we find that detector type does not provide an escape from the trigger-complexity dilemma. The differences between convolution, concatenation, pooling, slicing, and masking are second-order effects compared to the first-order coupling between trigger complexity and false-positive volume. An attacker cannot select a more sophisticated detector implementation to recover accuracy lost to a complex trigger.

7 Discussion

As observed in our evaluation, FAULT characterizes AB viability across three image-classification datasets. Although not observed in our evaluation, the inner-product detection geometry from §4.1.1 also applies to other modalities (NLP, audio, time-series), but the natural-input distribution governing F_{benign} would differ substantially. We expect the three limitations to hold qualitatively, but operating-region boundaries should be re-measured per modality. FAULT applies directly: only the dataset D in Algorithm 1 changes.

All FAULT-tested ABs used a single trigger. Although not observed, real attackers might deploy multi-trigger ABs as a hedge against partial corruption. Limitation 3 (§4.4) predicts this only worsens false-positive collisions, as the half-space $\mathcal{H}(W, N)$ expands to admit any patch matching any trigger. Empirical confirmation is left to future work.

FAULT treats N as a static, attacker-chosen setting determined prior to deployment. Although not observed, future adversarial methods might adapt N dynamically at inference time using techniques from dynamic neural networks [46]. Such a construction would not escape Limitations 1–3 in principle but might shift the operating-region boundary by tailoring sensitivity per input (at the cost of additional architectural complexity that increases surface area for code-level inspection).

7.1 Implications for Defenders

Our results suggest concrete actions for DL practitioners deploying models from public repositories. First, §6.2 shows that no AB construction tested is robust to all ten transformations simultaneously. Defenders should deploy at least one preprocessing transformation per inference pipeline, drawn from each of three orthogonal categories: *geometric* (rotation, cropping), *photometric* (contrast, grayscale), and *stochastic* (noise, denoising). This combination forces a trigger robust to all three corruption types simultaneously, which §4 suggests is geometrically infeasible.

Also, shown in §4.1.1, all five AB detector constructions reduce to the same inner-product form. Code audits should flag any architectural component computing a parameterized similarity between an input region and a fixed reference pattern, regardless of the operator used. Particular attention should be paid to (a) convolution kernels with hardcoded constant values, (b) pooling configurations with non-standard strides, and (c) layers computing a scalar score followed by thresholding outside the main classification head.

Finally, Limitations 2 and 3 imply that any AB tuned for non-trivial ASR produces observable benign-accuracy degradation. Defenders with a labeled validation set can use unexplained accuracy drops as an indicator of AB activity, particularly when the drop coincides with a new model from an external source. Our evaluation observes degradation as small as 2.23% under the stealthiest attacker (§6.5), so defenders should track validation accuracy with resolution sufficient to detect single-percent shifts.

8 Related Work

AI Supply Chain Attacks. As AI supply chains grow more complex, the attack surface expands [11, 12, 28], with developers increasingly relying on pre-trained DL models from public repositories. Prior work [47, 48] investigates the injection of malicious DL models into these repositories. Contrary to our approach, they rely on DL model serialization vulnerabilities to compromise the victim machine. WBBs and adversarial example attacks [49, 50], against DL models aim to degrade model performance, or even cause model failure, under attacker chosen conditions (or *triggers*) [51–53]. To accomplish this, attackers create malicious neurons in the DL model by poisoning training datasets [21, 23, 25, 26, 54, 55], providing malicious updates [22, 24] to federated learning systems [56], and even attacking the model at runtime [27] to apply non-control data attacks [57]. However, these attacks target DL model parameters, whereas ABs exploit the model architecture and code.

DL Backdoor Defenses. Alongside the proliferation of DL backdoors, a plethora of defenses has emerged to counter these attacks [58–63]. These defenses utilize input pre-processing [16, 64–67] to remove the backdoor trigger, detecting backdoors by analyzing neural network behavior [5, 6, 68, 69], and by filtering out malicious inputs [70–74]. However, these defenses target WBB attacks and are ineffective against ABs, where attackers can bypass them through sensitivity tuning.

Architectural Backdoors. ABs were first introduced by Bober-Irizar et al. [1] and further explored by Langford et al. [2]. Our work formalizes the limitations of ABs [4] and provide security suggestions for defenders.

9 Conclusion

By unifying the five existing AB detector implementations into a single inner-product form, we showed that AB trigger detection is governed by three fundamental limitations: transformations attenuate trigger response below threshold, raising sensitivity to recover ASR raises false positives, and complex triggers admit larger volumes of benign-input collisions. We formalized these limitations mathematically and built FAULT to trace the achievable (ASR, ACC, FPR) operating curve of any AB construction. Across thirty configurations on GTSRB, CIFAR10, and Mapillary, the attacker's reachable ASR is bounded between 10% and 20% at measurable accuracy cost. Ultimately, we find that ABs, as currently constructed, cannot be both effective and stealthy in realistic deployments.

10 Open Science

In compliance with the open science policy and to promote the reproducibility and replicability of this research, our artifacts are publicly available on an anonymous github [75]. These will include the prototype implementation of FAULT, all models used, and all datasets employed in our work.

11 Ethics Considerations

This research highlights the limitations of a recently discovered threat. We disclose our findings solely to raise awareness and provide a call to action for the research community to understand the fundamental limitations of existing ABs. All findings, including model checkpoints, backdoored code, and methodologies, have been responsibly developed. We have not deployed or tested these attacks on publicly accessible or production-level systems. The data used (CIFAR-10 [19], GTSRB [18], and Mapillary [20]) are publicly available, benchmark datasets commonly used in academic contexts. We explicitly discourage the use or extension of these techniques for malicious purposes and encourage the research community to leverage our results responsibly to build more secure AI systems.

References

[1] M. Bober-Irizar, I. Shumailov, Y. Zhao, R. Mullins, and N. Papernot, "Architectural backdoors in neural networks," in *Proceedings of the 40th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, Canada, Jun. 2023.

[2] H. Langford, I. Shumailov, Y. Zhao, R. Mullins, and N. Papernot, "Architectural neural backdoors from first principles," in *Proceedings of the 46th IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2025.

[3] A. A. Miah and Y. Bi, "Exploiting the vulnerability of large language models via defense-aware architectural backdoor," *2409.01952*, 2024. [Online]. Available: https://arxiv.org/abs/2409.01952.

[4] V. Childress, J. Collyer, and J. Knapp, "Architectural backdoors in deep learning: A survey of vulnerabilities, detection, and defense," *2507.12919*, 2025. [Online]. Available: https://arxiv.org/abs/2507.12919.

[5] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proceedings of the 40th IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2019.

[6] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 26th ACM Conference on Computer and Communications Security (CCS)*, London, UK, Nov. 2019.

[7] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," pp. 273–294, Sep. 2018. DOI: 10.1007/978-3-030-00470-5_13.

[8] S. Mahmood, H. N. Nguyen, and S. A. Shaikh, "Systematic threat assessment and security testing of automotive over-the-air (ota) updates," *Vehicular Communications*, 2022. DOI: 10.1016/j.vehcom.2022.100468.

[9] S. Halder, A. Ghosal, and M. Conti, "Secure ota software updates in connected vehicles: A survey," *1904.00685*, 2019. [Online]. Available: https://arxiv.org/abs/1904.00685.

[10] C. M. Linn, M. Rajagopalan, S. Baker, C. Collberg, S. K. Debray, and J. H. Hartman, "Protecting against unexpected system calls," in *Proceedings of the 14th USENIX Security Symposium (Security)*, Baltimore, MD, Aug. 2005.

[11] S. Neupane, G. Holmes, E. Wyss, D. Davidson, and L. D. Carli, "Beyond typosquatting: An in-depth look at package confusion," in *Proceedings of the 32nd USENIX Security Symposium (Security)*, Anaheim, CA, Aug. 2023.

[12] M. Zimmermann, C.-A. Staicu, C. Tenny, and M. Pradel, "Small world with high risks: A study of security threats in the npm ecosystem," in *Proceedings of the 28th USENIX Security Symposium (Security)*, Santa Clara, CA, Aug. 2019.

[13] Huggingface., *Huggingface model safety.* [Accessed: 2024-7-12]. [Online]. Available: https : / / huggingface . co / docs / text - generation - inference/en/basic_tutorials/safety.

[14] Hugging Face, https : / / huggingface . co/, [Accessed: 2023-10-06].

[15] Y. Zhang, Z. Wang, J. Jiang, H. You, and J. Chen, "Toward improving the robustness of deep learning models via model transformation," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Rochester, MI, Oct. 2022.

[16] H. Qiu, Y. Zeng, S. Guo, T. Zhang, M. Qiu, and B. Thuraisingham, "Deepsweep: An evaluation framework for mitigating dnn backdoor attacks using data augmentation."

[17] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten, "Countering adversarial images using input transformations," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, BC, Apr. 2018.

[18] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in *IEEE International Joint Conference on Neural Networks*, 2011.

[19] *Cifar-10 (canadian institute for advanced research)*, http://www.cs.toronto.edu/~kriz/cifar.html, [Accessed: 2023-01-19].

[20] *Mapillary*, https : / / www . mapillary . com/, [Accessed: 2024-11-01].

[21] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017. [Online]. Available: https://arxiv.org/abs/1712.05526.

[22] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, Virtual Conference, 2018.

[23] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient

optimization," *arXiv preprint arXiv:1708.08689*, 2017. [Online]. Available: https://arxiv.org/abs/1710.00942.

[24] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proceedings of the 29th USENIX Security Symposium (Security)*, Virtual Conference, Aug. 2020.

[25] N. Carlini, "Poisoning The Unlabeled Dataset of Semi-Supervised Learning," in *30th USENIX Security Symposium*, 2021.

[26] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, "Poisoning web-scale training datasets is practical," in *Proceedings of the 45th IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2024.

[27] F. Yao, A. S. Rakin, and D. Fan, "Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips," in *Proceedings of the 29th USENIX Security Symposium (Security)*, Virtual Conference, Aug. 2020.

[28] H. Siadati, S. Jafarikhah, E. Sahin, T. B. Hernandez, E. L. Tripp, D. Khryashchev, and A. Kharraz, "Devphish: Exploring social engineering in software supply chain attacks on developers," *2402.18401*, 2024. [Online]. Available: https://arxiv.org/abs/2402.18401.

[29] GitHub, *GitHub*, 2024. [Online]. Available: https://github.com.

[30] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint arXiv:1811.03728*, 2018. [Online]. Available: https://arxiv.org/abs/1811.03728.

[31] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," *2410.17725*, 2024. [Online]. Available: https://arxiv.org/abs/2410.17725.

[32] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "Yolov10: Real-time end-to-end object detection," *2405.14458*, 2024. [Online]. Available: https://arxiv.org/abs/2405.14458.

[33] Meta, *Build the future of ai with meta llama 3*, [Accessed: 2024-07-11]. [Online]. Available: https://llama.meta.com/llama3.

[34] K. Grosse, L. Bieringer, T. R. Besold, and A. Alahi, "Towards more practical threat models in artificial intelligence security," in *Proceedings of the 33rd USENIX Security Symposium (Security)*, Philadelphia, PA, Aug. 2024.

[35] A. Ho, *Proprietary ai models are dead. long live proprietary ai models*, [Accessed: 2023-11-16]. [Online]. Available: https://thenewstack.io/proprietary-ai-models-are-dead-long-live-proprietary-ai-models/.

[36] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, 2019. DOI: 10.1186/s40537-019-0197-0.

[37] I. Kandel, M. Castelli, and L. Manzoni, "Brightness as an augmentation technique for image classification," *Emerging Science Journal*, 2022. DOI: 10.28991/ESJ-2022-06-04-015.

[38] S. Dodge and L. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions."

[39] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *1712.04621*, 2017. [Online]. Available: https://arxiv.org/abs/1712.04621.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 33rd IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Nevada, Jun. 2016.

[41] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the 35th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, Utah, Jun. 2018. DOI: 10.1109/CVPR.2018.00474.

[42] G. Jocher, A. Chaurasia, and J. Qiu, *Ultralytics YOLO*, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.

[43] *The importance of blur as an image augmentation technique*, https://blog.roboflow.com/using-blur-in-computer-vision-preprocessing/, [Accessed: 2024-11-18].

[44] M. E. Akbiyik, *Data augmentation in training cnns: Injecting noise to images*, 2023. [Online]. Available: https://arxiv.org/abs/2307.06855.

[45] D. Yin, R. Gontijo Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, "A fourier perspective on model robustness in computer vision," Dec. 2019.

[46] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks."

[47] P. Zhou, *How to make hugging face to hug worms: Discovering and exploiting unsafe* `pickle.loads` *over pre-trained large model hubs*, BlackHat Asia, 2024.

[48] B. Casey, J. Santos, and M. Mirakhorli, "A large-scale exploit instrumentation study of ai/ml supply chain attacks in hugging face models," *arXiv preprint arXiv:2410.04490*, 2024.

[49] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *arXiv:1801.00553*, 2018. [Online]. Available: https://arxiv.org/abs/1801.00553.

[50] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv:1412.6572*, 2015. [Online]. Available: https://arxiv.org/abs/1412.6572.

[51] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 26th ACM Conference on Computer and Communications Security (CCS)*, London, UK, Nov. 2019.

[52] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017. [Online]. Available: https://arxiv.org/abs/1708.06733.

[53] B. Sun, J. Sun, W. Koh, and J. Shi, "Neural network semantic backdoor detection and mitigation: A Causality-Based approach," in *Proceedings of the 33rd USENIX Security Symposium (Security)*, Philadelphia, PA, Aug. 2024.

[54] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," Jun. 2012. DOI: 10.5555/3042573.3042761.

[55] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks

and countermeasures for regression learning," in *Proceedings of the 39th IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2018.

[56] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019. [Online]. Available: https://arxiv.org/abs/1902.01046.

[57] S. Chen, J. Xu, and E. C. Sezer, "Non-Control-Data attacks are realistic threats," in *Proceedings of the 14th USENIX Security Symposium (Security)*, Baltimore, MD, Aug. 2005.

[58] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 5–22, 2022.

[59] Y. Gao, B. G. Doan, Z. Zhang, S. Ma, J. Zhang, A. Fu, S. Nepal, and H. Kim, "Backdoor attacks and countermeasures on deep learning: A comprehensive review," *arXiv preprint arXiv:2007.10760*, 2020.

[60] G. Abad, J. Xu, S. Koffas, B. Tajalli, S. Picek, and M. Conti, "Sok: A systematic evaluation of backdoor trigger characteristics in image classification," *arXiv preprint arXiv:2302.01740*, 2023.

[61] G. Tao, Y. Liu, S. Cheng, S. An, Z. Zhang, Q. Xu, G. Shen, and X. Zhang, *Deck: Model hardening for defending pervasive backdoors*, 2022. [Online]. Available: https://arxiv.org/abs/2206.09272.

[62] Z. Meng, J. Li, Y. Gong, and B.-H. Juang, "Adversarial teacher-student learning for unsupervised domain adaptation," pp. 5949–5953, Apr. 2018. DOI: 10.1109/ICASSP.2018.8461682.

[63] G. Tao, Y. Liu, S. Cheng, S. An, Z. Zhang, Q. Xu, G. Shen, and X. Zhang, "Deck: Model hardening for defending pervasive backdoors," *2206.09272*, 2022. [Online]. Available: https://arxiv.org/abs/2206.09272.

[64] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," *arXiv preprint arXiv:1710.00942*, 2017. [Online]. Available: https://arxiv.org/abs/1710.00942.

[65] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," in *Proceedings of the 36th Annual Computer Security Applications Conference*, 2020, pp. 897–912.

[66] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[67] S. Udeshi, S. Peng, G. Woo, L. Loh, L. Rawshan, and S. Chattopadhyay, "Model agnostic defence against backdoor attacks in machine learning," *IEEE Transactions on Reliability*, vol. 71, no. 2, pp. 880–895, 2022.

[68] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017. [Online]. Available: https://arxiv.org/abs/1703.00410.

[69] N. Karim, A. A. Arafat, U. Khalid, Z. Guo, and N. Rahnavard, "Efficient backdoor removal through natural gradient fine-tuning," *arXiv preprint arXiv:2306.17441*, 2023. [Online]. Available: https://arxiv.org/abs/2306.17441.

[70] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *Proceedings of the 35th annual computer security applications conference*, 2019, pp. 113–125.

[71] M. Subedar, N. Ahuja, R. Krishnan, I. J. Ndiour, and O. Tickoo, "Deep probabilistic models to detect data poisoning attacks," *arXiv preprint arXiv:1912.01206*, 2019.

[72] M. Du, R. Jia, and D. Song, "Robust anomaly detection and backdoor attack detection via differential privacy," *arXiv preprint arXiv:1911.07116*, 2019.

[73] K. Jin, T. Zhang, C. Shen, Y. Chen, M. Fan, C. Lin, and T. Liu, "Can we mitigate backdoor attack using adversarial detection methods?" *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 2867–2881, 2022.

[74] M. Javaheripi, M. Samragh, G. Fields, T. Javidi, and F. Koushanfar, "Cleann: Accelerated trojan shield for embedded neural networks," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–9.

[75] *Fault anonymous repository*, https://anonymous.4open.science/r/FAULT-Private-10F3, [Accessed: 2026-04-30].

A Appendix

A.1 Composition of Transformations

Our preliminary experiments showed that individual input-space transformations significantly degrade existing AB effectiveness by disrupting the trigger detection process. Motivated by these results, we hypothesize that applying combinations of transformations would further amplify this disruption, compelling attackers to make their trigger detectors increasingly permissive (higher sensitivity N) to maintain adequate ASR. To test this hypothesis, we began with a single transformation known to weaken trigger detection (e.g., contrast adjustment) and progressively introduced additional transformations (such as Variation Denoising, Gaussian Blur, and Noise), each known from earlier experiments (Table 2) to independently degrade AB performance.

Figure 4 illustrates the tradeoff between the attacker's sensitivity parameter N and both the ASR and ACC, under the composition of transformations. It can be seen that from when only contrast was used ΔN was 0.068. But as variation denoising was applied, ΔN increased to 0.083. Similarly, for contrast with denoising and blur ΔN increased once again to 0.094. Finally for contrast with denoising, blur, and noise ΔN increased to 0.158, significantly increasing the N required to achieve an ASR of greater than 10%.

Though the ASR required significantly higher values of N, this resulted in a tradeoff of accuracy values (reaching 73.5% with 4 transformations from 88.1% with just contrast). This trade-off arises naturally from the AB design. Nonetheless, an overly lenient trigger detector (high N) admits false positives more frequently. Conversely, a very strict detector (low N) may fail to capture partially modified triggers in the presence of stronger transformations to the input. As defenders apply more

transformations, attackers are forced to shift N higher, thereby safeguarding the ASR yet damaging normal inference reliability.

For defenders, these insights indicate that by composing multiple input transformations attackers are forced to apply more permissive N values to achieve the same ASRs as before, causing reductions in benign accuracy of the model, reducing the stealthiness of their attack.

A.2 Choices of Transform Parameters

Choices for transformation parameters are highly dependent on dataset type, image size, etc. However, we chose to use set parameters for each transformation as follows:

Rotate. We chose to rotate our images by 7.0^o as our models were not trained to be equivariant to rotation and therefore face severe accuracy degradation at larger angles.

Contrast. We chose a contrast parameter of 0.85 as this lowers the contrast, shifting all pixels in the image closer in value (reducing the difference between the trigger and neighboring pixels). Picking values above one for this parameter would exacerbate the differences between the trigger and neighboring pixels and therefore we avoid using such values for our contrast-as-a-defense.

Crop. We chose a final crop of 95% width and 95% height of the original image. This allowed us keep the images semantically similar while also applying the crop transformation.

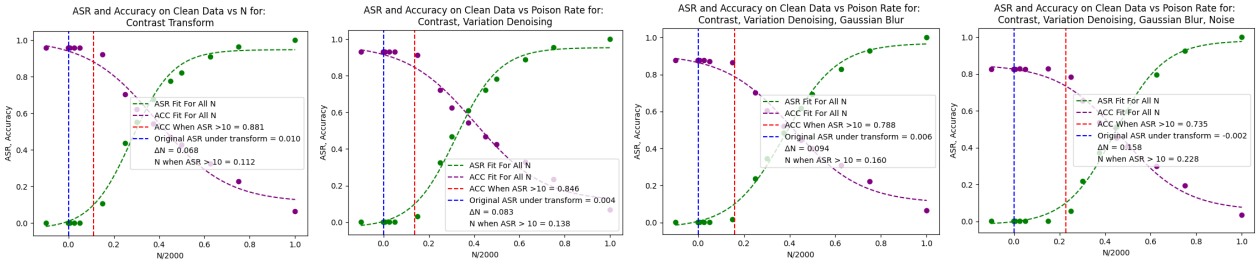
Grayscale. No parameters chosen, image is converted to grayscale.

Noise. We chose a 20% noise parameter. This allows us to introduce a significant amount of noise (enough to disturb a trigger), but keep the transformed image similar to the original. Overall image features will remain intact, allowing for relatively similar benign accuracy to be achieved. This approach has been extensively applied to combat adversarial examples.

Gaussian Blur. We decided to use a value of $image\_size/10$ for images within the CIFAR10 and GTSRB datasets (sigma value of 3). This value showed meaningful results to reduce trigger capabilities but keep benign features in tact. For Mapillary experiments, we chose a sigma value of 49 as literature shows that values above 50 have high likeliness to remove important features from the image.

Denoising. We chose a denosing weight of 30 as the smaller the weight the more denoising occurs. This value allows the transformed image to remain similar to the original to preserve the accuracy of the model while still changing the overall semantics of the image.

Median Filter. We chose a median parameter of 3 pixels as it retains the local context of the original image, but still provides enough change to the image to meaningfully change the image. This value is used as a typical starting point for median filtering.



(a) Contrast Applied. **(b) Contrast And V-D Applied.** **(c) Con, V-D, G-Blur.** **(d) Con, V-D, G-Blur, Noise.**

Figure 4: A Composition of Transformations affects the tradeoff between accuracy and ASR significantly.