# BLINDCLEAN: A PRIOR-KNOWLEDGE-FREE DATA SANITIZATION METHOD AGAINST CLEAN-LABEL BACKDOOR ATTACKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Machine learning training commonly relies on third-party datasets to reduce the costs of data collection and annotation, but this practice also exposes models to clean-label backdoor attacks. Such attacks are difficult to detect, as poisoned samples retain both their original labels and their visual semantics. Existing data sanitization methods either depend on prior knowledge unavailable to the platform or degrade clean accuracy by mistakenly discarding too many clean samples. This paper proposes BlindClean, which leverages a key characteristic of poisoned samples: under input perturbations, poisoned samples tend to exhibit higher feature stability than clean samples. BlindClean therefore sanitizes a dataset relying solely on the potentially poisoned dataset itself. It evaluates cross-layer feature stability under multi-scale input perturbations, estimates reference feature distributions of clean samples from low-stability samples of each class, and then measures the feature deviations of high-stability samples from these distributions. This design allows BlindClean to first identify the class most likely to be attacked and then detect the poisoned samples within it. Experiments on three representative clean-label backdoor attacks show that BlindClean detects at least 97% of the poisoned samples while mistakenly removing less than 1% of clean samples. When the sanitized data is used for downstream training, BlindClean reduces the average attack success rate to below 1% and improves clean accuracy by up to 44 percentage points over state-of-the-art methods. BlindClean thus provides a practical and effective means of sanitizing untrusted third-party data.

## 1 INTRODUCTION

Modern machine learning increasingly relies on training data collected by third-party organizations, open-source communities, and data providers (Roh et al., 2019). Using such data can substantially reduce the cost of data collection, annotation, and maintenance while accelerating model development. However, model developers cannot always verify the provenance and integrity of externally obtained datasets. Uncertainty about dataset integrity exposes the training process to data poisoning attacks, in which adversaries inject manipulated samples into a training dataset to influence the behavior of models subsequently trained on it (Cinà et al., 2023; Goldblum et al., 2022).

Backdoor attacks are particularly concerning because they can alter model behavior without substantially degrading performance on benign inputs (Li et al., 2022). A backdoored model behaves normally on clean inputs but is induced to produce an attacker-specified prediction when a trigger is present. Clean-label backdoor attacks further complicate this problem. Unlike dirty-label attacks (Gu et al., 2017; Chen et al., 2017), they preserve the original labels and visual semantics of poisoned samples while implanting backdoors through carefully designed perturbations, gradient matching, or trigger optimization (Turner et al., 2019; Souri et al., 2022; Zeng et al., 2023; Yasur et al., 2026). Consequently, poisoned samples can overlap with clean samples in label space, visual appearance, and feature representations, making them difficult to identify through manual inspection, label verification, or conventional outlier detection.

Data sanitization provides a proactive defense by identifying and removing poisoned samples from a training dataset (Chen et al., 2018; Tran et al., 2018; Hayase et al., 2021). This setting is particu-



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

larly relevant to third-party data sharing, where a dataset should be sanitized before it is released for downstream training. However, existing methods may assume access to trusted clean data (Ma et al., 2023; Pan et al., 2023) or require a poisoning-rate estimate to determine the number of samples to remove (Li et al., 2021). Moreover, because the attacked target class is unknown in practice, many existing methods examine samples from all classes (Hou et al., 2025; Pal et al., 2024), which can increase the risk of discarding clean samples from unaffected classes. These limitations make it difficult to balance the utility–security trade-off: failing to remove sufficient poisoned samples allows the retrained model to retain the backdoor, whereas removing too many clean samples degrades the utility of the resulting dataset and downstream clean accuracy.

To address these challenges, this paper introduces BlindClean, a prior-knowledge-free data sanitization method for clean-label backdoor attacks. BlindClean requires only a potentially poisoned training dataset and does not rely on trusted clean reference data, poisoning-rate estimates, or target-class information. It evaluates cross-layer feature stability under multi-scale input perturbations. Exploiting differences in feature stability between poisoned and clean samples, BlindClean estimates class-wise reference feature distributions from low-stability samples and measures the deviations of high-stability samples from these distributions to identify the attacked target class. Finally, it detects poisoned samples only within the identified target class, thereby limiting the detection scope and preserving clean samples in unaffected classes.

The contributions of this paper are summarized as follows:

- This paper proposes BlindClean, a prior-knowledge-free data sanitization method that detects clean-label poisoned samples without requiring trusted clean reference data, poisoning-rate estimates, or target-class information.
- BlindClean incorporates a target-class-aware detection strategy that uses multi-scale, cross-layer feature stability to infer the attacked target class and subsequently restrict poisoned-sample detection to that class, reducing false positives from unaffected classes.
- BlindClean is evaluated on CIFAR-10 and SVHN under three representative clean-label backdoor attacks. BlindClean achieves an average true positive rate (TPR) and false positive rate (FPR) of 99.40% and 0.42%, respectively, and reduces FPR by up to 60.28 percentage points. After retraining on the sanitized data, it is the only evaluated method that reduces the average attack success rate (ASR) below 1% while maintaining high clean accuracy, achieving clean accuracy up to 44.40 percentage points higher than that of other evaluated methods.

## 2 Related Work

### 2.1 Clean-Label Backdoor Attacks

Clean-label backdoor attacks preserve the original labels and visual semantics of poisoned samples, making them difficult to distinguish from clean samples (Yasur et al., 2026). Label-Consistent (LC) perturbs samples from the target class using adversarial perturbations before applying a trigger (Turner et al., 2019). Sleeper Agent (SA) uses gradient matching to optimize poisoned samples for models trained from scratch without directly embedding the trigger into the poisoned training samples (Souri et al., 2022). Narcissus is designed for a more limited attacker-knowledge setting and uses a surrogate model to optimize a target-class-specific trigger (Zeng et al., 2023). These attacks represent distinct poison-generation mechanisms, including adversarial perturbation, gradient matching, and trigger optimization, and can produce poisoned samples that overlap substantially with clean samples in feature space.

### 2.2 Data Sanitization Methods

Existing methods for poisoned-sample identification can be categorized as feature-based, loss-based, and consistency-based approaches. **Feature-based methods** analyze internal representations to separate suspicious samples: Beatrix uses higher-order feature statistics and trusted clean reference data to identify anomalous samples (Ma et al., 2023), whereas FLARE aggregates multi-layer features, reduces their dimensionality, and separates samples through density-based clustering (Hou et al.,

2025). **Loss-based methods** exploit differences in training dynamics between clean and poisoned samples. ABL applies local gradient ascent to enlarge loss differences and then isolates a predefined fraction of suspicious samples (Li et al., 2021), while ASSET uses a small clean base set and actively induces different model behaviors for clean and poisoned samples to increase their loss separation, and then adaptively selects samples for removal (Pan et al., 2023). **Consistency-based methods** examine prediction changes under transformations: MSPC measures prediction consistency under input scaling (Pal et al., 2024), whereas IBD-PSC evaluates prediction changes under parameter scaling (Hou et al., 2024). Although ABL and IBD-PSC were originally proposed as a training-time defense and an input-level detection method, respectively, both can be applied to identify and filter suspicious training samples. Despite these different strategies, prior-knowledge requirements and limited effectiveness against clean-label backdoor attacks remain challenges.

# 3 Problem Formulation

**System model.** Consider a $K$-class classification dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ collected from multiple data contributors through a third-party data platform. For each class $c \in \{1, \ldots, K\}$, the corresponding class-wise subset is

$$\mathcal{D}_c = \{(x_i, y_i) \in \mathcal{D} \mid y_i = c\}.$$

The dataset may contain an unknown subset of clean-label poisoned samples, denoted by $\mathcal{D}_{\text{poison}} \subset \mathcal{D}$. For each poisoned sample $(x_i, y_i) \in \mathcal{D}_{\text{poison}}$, the original label and visual semantics are preserved, while the input is manipulated to cause a downstream model to associate an attacker-specified trigger with a target class. Given only the potentially poisoned dataset $\mathcal{D}$, data sanitization aims to infer an estimate $\hat{c}^*$ of the attacked target class $c^*$ and identify a suspicious subset $\widehat{\mathcal{D}}_{\text{poison}} \subseteq \mathcal{D}_{\hat{c}^*}$. The sanitized dataset is

$$\mathcal{D}_{\text{san}} = \mathcal{D} \setminus \widehat{\mathcal{D}}_{\text{poison}},$$

which is subsequently used for downstream model training. An effective method should detect poisoned samples while avoiding the incorrect removal of clean samples, thereby reducing attack success without substantially degrading clean accuracy.

**Threat model.** Consider an attacker acting as a malicious data contributor. The attacker can submit a limited number of manipulated training samples. These samples retain their original labels and visual semantics but are crafted to implant a targeted backdoor into models trained on the dataset. However, the attacker can control only the submitted samples and cannot modify data provided by other contributors, interfere with the data sanitization procedure, or control downstream model training.

The defender is the third-party data platform that sanitizes the dataset before releasing it for downstream training. The defender has access to the complete potentially poisoned dataset and may train a model on it to support poisoned-sample detection. However, the defender has no trusted clean reference data and no knowledge of the poisoning rate, the attacked target class, or the attack strategy used to generate poisoned samples. The defender must therefore infer the target class and identify poisoned samples from the potentially poisoned dataset and the model trained on it.

# 4 BlindClean

## 4.1 Overview

Figure 1 summarizes the four modules of BlindClean and their data flow. Given a potentially poisoned training dataset, the **Multi-scale Perturbed Sample Generator** adds Gaussian noise at multiple scales to generate perturbed versions of each input. BlindClean then extracts multi-layer feature representations from the original and perturbed samples. The **Perturbation-based Feature Stability Evaluation** module measures their cross-layer feature shifts to obtain sample-wise feature stability scores and class-wise stability rankings. The **Layer-based Target Class Determination** module estimates class-wise reference feature distributions from low-stability samples and evaluates the deviations of high-stability samples to identify the attacked target class. Finally, the **Target-class-aware Poison Sample Detection** module uses the feature-distribution deviations of samples

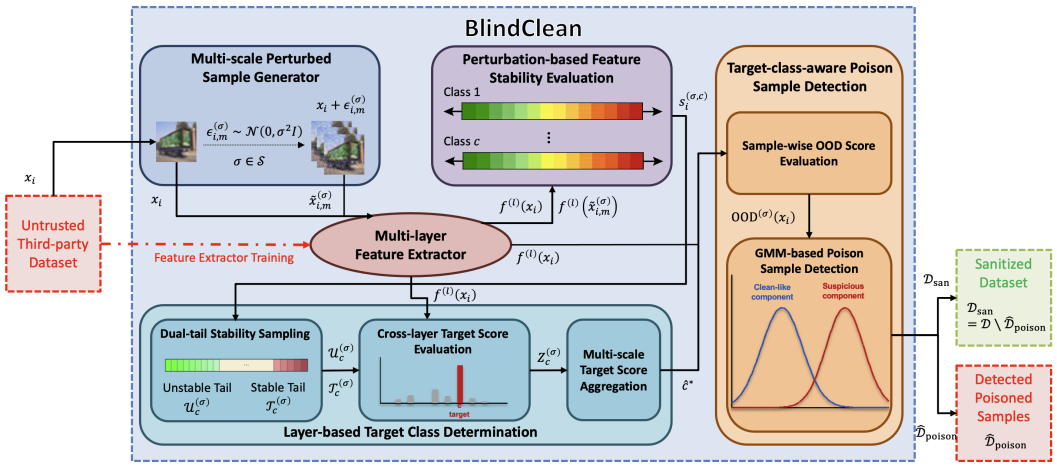



Figure 1: Framework of BlindClean.

within the identified target class and applies a Gaussian mixture model (GMM) to separate poisoned samples from clean samples.

### 4.2 Multi-scale Perturbed Sample Generator

Many data sanitization methods directly analyze feature representations obtained from original inputs and assume that clean and poisoned samples exhibit clear separation in the feature space. However, such global feature separation may not be evident under clean-label backdoor attacks. As illustrated by the BadNets and Narcissus feature-space visualizations in Appendix B, poisoned samples form a separated cluster under BadNets but are distributed among clean samples under Narcissus. Some methods instead use prediction consistency under input perturbations without relying on feature-space separation. However, prediction consistency reflects only changes in the final model output and cannot characterize sample responses across different internal network layers. BlindClean therefore examines cross-layer feature shifts under multi-scale input perturbations.

For each input sample $x_i$, BlindClean generates $M$ perturbed samples at each perturbation scale $\sigma \in \mathcal{S}$ by adding Gaussian noise:

$$\tilde{x}_{i,m}^{(\sigma)} = x_i + \epsilon_{i,m}^{(\sigma)}, \qquad \epsilon_{i,m}^{(\sigma)} \sim \mathcal{N}(0, \sigma^2 I), \quad m = 1, \ldots, M, \tag{1}$$

where $I$ is the identity matrix and $\mathcal{S}$ denotes the set of perturbation scales. Using multiple scales captures feature responses to different perturbation strengths, while repeated independent noise samples reduce sensitivity to any single perturbation. The original sample $x_i$ and its perturbed versions $\tilde{x}_{i,m}^{(\sigma)}$ are then passed through a feature extraction model trained by the defender on the potentially poisoned dataset. For each analyzed layer $l \in \{1, \ldots, L\}$, the feature representations $f^{(l)}(x_i)$ and $f^{(l)}(\tilde{x}_{i,m}^{(\sigma)})$ are retained for subsequent cross-layer feature stability evaluation.

### 4.3 Perturbation-based Feature Stability Evaluation

BlindClean measures feature stability by comparing the representations of each original sample with those of its perturbed versions. Under input perturbations, poisoned samples tend to exhibit smaller feature changes than clean samples. A possible explanation is that the attack optimization process causes the model to form stable responses to backdoor-related features, making their internal feature directions less sensitive to input perturbations. In contrast, the classification of some clean samples may rely more heavily on local or relatively unstable semantic cues, causing their feature representations to change more noticeably. However, the layer that best separates poisoned and clean samples varies across attacks, as shown in Appendix C. BlindClean therefore aggregates feature shifts across multiple layers.

For sample $x_i$, layer $l$, and perturbation scale $\sigma$, the average cosine feature shift between the original sample and its $M$ perturbed samples is computed as

$$d_i^{(l,\sigma)} = \frac{1}{M} \sum_{m=1}^{M} \left[ 1 - \cos\left( f^{(l)}(x_i), f^{(l)}(\tilde{x}_{i,m}^{(\sigma)}) \right) \right]. \tag{2}$$

A smaller feature shift indicates a smaller change in feature direction and therefore higher feature stability. A comparison of cosine distance, L2 distance, and their combination is provided in Appendix F. Since feature-shift distributions may vary across classes, network layers, and perturbation scales, directly aggregating raw shifts across layers could cause layers with larger numerical ranges to dominate the final score. BlindClean therefore applies robust z-score normalization separately within each class, layer, and perturbation scale, reducing the influence of inter-class variation, layer-wise scale differences, and extreme values. The cross-layer feature stability score of $x_i \in \mathcal{D}_c$ at scale $\sigma$ is defined as

$$s_i^{(\sigma,c)} = \frac{1}{L} \sum_{l=1}^{L} \mathrm{RobustZ}_{c,l,\sigma}\left( -d_i^{(l,\sigma)} \right), \tag{3}$$

where $\mathrm{RobustZ}_{c,l,\sigma}(\cdot)$ denotes robust z-score normalization based on the median and median absolute deviation (MAD) of samples in class $c$ at layer $l$ and perturbation scale $\sigma$. The negative sign ensures that a smaller feature shift corresponds to a larger stability score.

### 4.4 Layer-based Target Class Determination

The class-wise stability rankings reveal markedly different tail compositions. As shown in Table 1, the Unstable Tail is almost entirely composed of clean samples, whereas poisoned samples are substantially concentrated in the Stable Tail. BlindClean exploits this separation by estimating class-wise reference feature distributions from the Unstable Tail and evaluating anomalous-sample enrichment in the Stable Tail to determine the attacked target class.

**Dual-tail stability sampling.** For each class $c$ and perturbation scale $\sigma$, BlindClean ranks samples in $\mathcal{D}_c$ according to their stability scores $s_i^{(\sigma,c)}$. The bottom fraction $\alpha$ defines the Unstable Tail $\mathcal{U}_c^{(\sigma)}$, whereas the top fraction $\beta$ defines the Stable Tail $\mathcal{T}_c^{(\sigma)}$. The former is used to estimate the reference distributions, while the latter is used for target-class scoring.

Table 1: Poisoning rates (%) in the Unstable Tail and Stable Tail under different tail selection ratios on CIFAR-10 with a 10% target-class poisoning rate, evaluated at the perturbation scale $\sigma^*$ selected by BlindClean.

| | Unstable Tail ratio $\alpha$ | | | | | Stable Tail ratio $\beta$ | | |
|---|---|---|---|---|---|---|---|---|
| Attack | 10% | 20% | 30% | 40% | 50% | 10% | 20% | 30% |
| LC | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 88.0 | 49.3 | 33.3 |
| SA | 0.0 | 0.0 | 0.0 | 0.1 | 0.1 | 77.0 | 47.2 | 32.7 |
| Narcissus | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 91.4 | 50.0 | 33.3 |

**Cross-layer target score evaluation.** Because clean and poisoned samples may overlap in feature space, Euclidean distance alone may not adequately capture deviations from the class-wise reference distribution. BlindClean therefore uses the Mahalanobis distance, which accounts for both the feature mean and covariance structure of the reference distribution. A comparison with Euclidean distance is provided in Appendix D.

For each layer $l$ and perturbation scale $\sigma$, BlindClean estimates the reference feature distribution of class $c$ from $\mathcal{U}_c^{(\sigma)}$. Its mean and covariance are given by

$$\begin{aligned} \mu_c^{(l,\sigma)} &= \frac{1}{|\mathcal{U}_c^{(\sigma)}|} \sum_{x_i \in \mathcal{U}_c^{(\sigma)}} f^{(l)}(x_i), \\ \Sigma_c^{(l,\sigma)} &= \mathrm{OAS}\left( \{ f^{(l)}(x_i) \mid x_i \in \mathcal{U}_c^{(\sigma)} \} \right), \end{aligned} \tag{4}$$

where $\mathrm{OAS}(\cdot)$ denotes Oracle Approximating Shrinkage covariance estimation (Chen et al., 2010), which provides a more stable covariance estimate than the sample covariance matrix in high-dimensional feature spaces. For each sample $x_i \in \mathcal{D}_c$, its deviation from the reference distribution is measured using the squared Mahalanobis distance:

$$D_i^{(l,\sigma,c)} = \left(f^{(l)}(x_i) - \mu_c^{(l,\sigma)}\right)^\top \left(\Sigma_c^{(l,\sigma)}\right)^{-1} \left(f^{(l)}(x_i) - \mu_c^{(l,\sigma)}\right). \tag{5}$$

At each layer and perturbation scale, samples whose distances exceed the 97.5th percentile of the corresponding Unstable-Tail distance distribution are regarded as OOD samples. A sample is considered cross-layer OOD only when it is identified as OOD at all analyzed layers. Let $O_c^{(\sigma)}$ be the number of cross-layer OOD samples in $\mathcal{T}_c^{(\sigma)}$, and let $R_c^{(\sigma)}$ be the total number of cross-layer OOD samples in $\mathcal{D}_c$. To account for class-wise differences in naturally occurring OOD samples, BlindClean evaluates their enrichment in the Stable Tail using a hypergeometric null model:

$$Z_c^{(\sigma)} = \frac{O_c^{(\sigma)} - n_c^{(\sigma)} R_c^{(\sigma)}/N_c}{\sqrt{n_c^{(\sigma)} \frac{R_c^{(\sigma)}}{N_c} \left(1 - \frac{R_c^{(\sigma)}}{N_c}\right) \frac{N_c - n_c^{(\sigma)}}{N_c - 1}}}, \tag{6}$$

where $N_c = |\mathcal{D}_c|$ and $n_c^{(\sigma)} = |\mathcal{T}_c^{(\sigma)}|$. A larger target score indicates that cross-layer OOD samples are more strongly enriched in the Stable Tail and that the class is more likely to be the attacked target class.

**Multi-scale target score aggregation.** The class ranked highest by $Z_c^{(\sigma)}$ may vary across perturbation scales. To reduce reliance on any single scale, BlindClean averages the target scores over all perturbation scales. Since each $Z_c^{(\sigma)}$ is standardized with respect to its corresponding hypergeometric null distribution, target scores from different scales are on a common scale and can be averaged:

$$Z_c^{\mathrm{MS}} = \frac{1}{|\mathcal{S}|} \sum_{\sigma \in \mathcal{S}} Z_c^{(\sigma)}, \qquad \hat{c}^* = \arg \max_{c \in \{1,\ldots,K\}} Z_c^{\mathrm{MS}}. \tag{7}$$

The identified target class $\hat{c}^*$ is then used for poisoned-sample detection.

### 4.5 Target-class-aware Poison Sample Detection

After identifying the target class $\hat{c}^*$, BlindClean performs poisoned-sample detection only within $\mathcal{D}_{\hat{c}^*}$. The Stable Tail used for target-class determination contains only a fixed proportion of highly stable samples and may not include every poisoned sample. BlindClean therefore evaluates all samples in the identified target class.

**Sample-wise OOD score evaluation.** For each perturbation scale $\sigma$, BlindClean uses the reference distributions estimated from $\mathcal{U}_{\hat{c}^*}^{(\sigma)}$ to calculate the Mahalanobis distance of every target-class sample at each analyzed layer. To make distance scores comparable across layers, the distances are normalized using the corresponding Unstable-Tail distance distributions:

$$\widetilde{D}_i^{(l,\sigma)} = \mathrm{RobustZ}_{\hat{c}^*,l,\sigma}^{\mathcal{U}} \left(D_i^{(l,\sigma,\hat{c}^*)}\right), \tag{8}$$

where $\mathrm{RobustZ}_{\hat{c}^*,l,\sigma}^{\mathcal{U}}(\cdot)$ denotes robust z-score normalization using the median and median absolute deviation of the corresponding Unstable-Tail distance distribution. The normalized distances are aggregated using the median over all analyzed layers, reducing the influence of outlier layers:

$$\mathrm{OOD}^{(\sigma)}(x_i) = \mathrm{median}_{l=1}^{L} \left(\widetilde{D}_i^{(l,\sigma)}\right), \qquad x_i \in \mathcal{D}_{\hat{c}^*}. \tag{9}$$

A larger OOD score indicates that a sample consistently deviates from the reference distributions across multiple layers.

**GMM-based poison sample detection.** The reference distributions, and consequently the OOD-score distributions, may vary across perturbation scales. BlindClean therefore fits a two-component GMM to the OOD scores of target-class samples at each scale and selects the scale with the minimum average posterior entropy:

$$\sigma^* = \arg\min_{\sigma \in \mathcal{S}} H^{(\sigma)}, \tag{10}$$

where $H^{(\sigma)}$ denotes the average posterior entropy of the GMM assignments at scale $\sigma$. Lower entropy indicates less uncertainty in the component assignments. At the selected scale $\sigma^*$, the component with the higher mean OOD score is identified as the poisoned-sample component. Samples assigned to this component form $\widehat{\mathcal{D}}_{\text{poison}}$ and are removed from the original training dataset to obtain $\mathcal{D}_{\text{san}}$.

# 5 Experiments

## 5.1 Experimental Setup

**Datasets and attacks.** BlindClean is evaluated on CIFAR-10 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011) under three representative clean-label backdoor attacks: Label-Consistent (LC), Sleeper Agent (SA), and Narcissus (Turner et al., 2019; Souri et al., 2022; Zeng et al., 2023). CIFAR-10 uses a target-class poisoning rate of 10%. For SVHN, target-class poisoning rates of 20% are used for LC and SA to ensure reliable backdoor implantation, whereas Narcissus uses a rate of 10%.

**Baselines and training configurations.** BlindClean is compared with Beatrix, FLARE, ABL, ASSET, MSPC, and IBD-PSC. ResNet-18 (He et al., 2016) is used for both feature extraction and downstream retraining. Full training configurations, BlindClean hyperparameters, and baseline settings are provided in Appendix A.

**Evaluation metrics.** For poisoned-sample detection, the true positive rate (TPR) is the proportion of ground-truth poisoned training samples that are correctly identified for removal. The false positive rate (FPR) is the proportion of all ground-truth clean samples in the full training dataset, including clean samples from both the identified target class and unaffected classes, that are incorrectly removed. After retraining on the sanitized dataset, the attack success rate (ASR) is the proportion of triggered test inputs, constructed from non-target-class test samples, that are predicted as the target class, and clean accuracy (CA) is the classification accuracy on clean test inputs. Unless otherwise stated, BlindClean results are reported as mean $\pm$ 95% confidence intervals over five random seeds.

## 5.2 Experimental Results

Table 2 reports upstream poisoned-sample detection performance and downstream performance after retraining on the sanitized dataset. Additional results at different target-class poisoning rates and a computational cost analysis are provided in Appendix E and Appendix G, respectively.

**Upstream poisoned-sample detection performance.** The upper panel of Table 2 shows that BlindClean maintains high detection rates with consistently low false-positive rates across attacks and datasets. Across the six attack–dataset settings, BlindClean achieves an average TPR/FPR of 99.40%/0.42%. The detection trade-off faced by some baselines is apparent under Narcissus on CIFAR-10: FLARE achieves a TPR of 99.80% but incurs an FPR of 19.40%, while MSPC achieves a TPR of 86.20% with an FPR of 14.76%. In the No attack setting, BlindClean yields FPRs of 0.53% on CIFAR-10 and 0.96% on SVHN, indicating that it rarely removes clean samples when no backdoor is present.

**Downstream performance after retraining.** The lower panel of Table 2 shows that BlindClean suppresses downstream backdoor behavior while preserving clean accuracy. Across the six attack–dataset settings, it achieves the lowest average ASR (0.66%) among all evaluated methods. The downstream results further show why both TPR and FPR must be considered jointly. Under Narcissus on CIFAR-10, FLARE reduces ASR to 0.20% but lowers CA to 80.03%, consistent with its high

Table 2: Experimental results on CIFAR-10 and SVHN. In (a), attack cells report TPR/FPR (%) and No attack cells report FPR only. In (b), attack cells report ASR/CA (%) and No attack cells report CA only. BlindClean results are means ± 95% confidence intervals over five random seeds.

**(a) Upstream poisoned-sample detection performance**

| | CIFAR-10 | | | | SVHN | | | |
|---|---|---|---|---|---|---|---|---|
| Method | LC | SA | Narcissus | No attack | LC | SA | Narcissus | No attack |
| Beatrix | 90.40/2.71 | 0.60/2.93 | 43.20/7.63 | 4.25 | 100.00/6.72 | 96.36/11.85 | 1.82/9.51 | 14.37 |
| FLARE | 100.00/0.00 | 1.80/60.89 | 99.80/19.40 | 60.08 | 100.00/0.00 | 86.54/10.64 | 100.00/6.05 | 5.05 |
| ABL | 10.60/0.90 | 19.40/0.81 | 0.20/1.01 | 1.00 | 43.78/0.77 | 0.00/1.37 | 0.00/0.68 | 1.35 |
| ASSET | 85.20/34.94 | 29.20/36.74 | 11.00/34.58 | 37.57 | 96.46/35.91 | 62.45/38.92 | 43.64/36.50 | 30.48 |
| MSPC | 86.60/10.48 | 91.20/13.43 | 86.20/14.76 | 20.56 | 99.70/17.54 | 0.10/19.02 | 75.35/39.78 | 35.38 |
| IBD-PSC | 100.00/3.58 | 39.80/5.45 | 0.00/0.76 | 9.94 | 100.00/14.73 | 0.00/14.02 | 0.40/39.20 | 30.41 |
| **BlindClean** | 100.00 ± 0.00 / 0.08 ± 0.01 | 97.00 ± 0.53 / 0.61 ± 0.05 | 99.88 ± 0.14 / 0.91 ± 0.02 | 0.53 ± 0.05 | 100.00 ± 0.00 / 0.06 ± 0.03 | 99.51 ± 0.28 / 0.58 ± 0.05 | 100.00 ± 0.00 / 0.27 ± 0.01 | 0.96 ± 0.21 |

**(b) Downstream performance after retraining**

| | CIFAR-10 | | | | SVHN | | | |
|---|---|---|---|---|---|---|---|---|
| Method | LC | SA | Narcissus | No attack | LC | SA | Narcissus | No attack |
| No defense | 99.60/94.55 | 92.00/94.59 | 100.00/95.01 | 94.82 | 99.70/96.90 | 93.55/96.90 | 100.00/96.99 | 97.07 |
| Beatrix | 49.72/94.48 | 44.80/89.73 | 99.98/94.40 | 94.09 | 0.16/96.95 | 0.18/95.76 | 99.29/97.04 | 96.80 |
| FLARE | 1.02/94.80 | 2.70/49.62 | 0.20/80.03 | 68.75 | 0.30/96.87 | 7.77/92.83 | 0.13/93.43 | 96.93 |
| ABL | 99.80/94.29 | 14.50/90.40 | 100.00/95.26 | 94.66 | 0.25/96.93 | 24.10/95.52 | 100.00/97.10 | 96.90 |
| ASSET | 97.43/88.14 | 4.80/85.52 | 99.94/93.67 | 91.93 | 0.26/96.42 | 6.39/92.67 | 95.68/96.61 | 96.58 |
| MSPC | 97.50/91.47 | 0.60/83.17 | 95.33/91.59 | 90.51 | 0.26/95.43 | 31.02/93.23 | 80.65/95.93 | 83.22 |
| IBD-PSC | 0.58/94.09 | 4.50/88.53 | 100.00/94.95 | 86.60 | 0.17/96.65 | 42.29/94.55 | 99.29/96.85 | 96.70 |
| **BlindClean** | 0.97 ± 0.19 / 94.54 ± 0.33 | 0.50 ± 0.18 / 94.02 ± 0.28 | 0.34 ± 0.24 / 94.89 ± 0.37 | 94.71 ± 0.23 | 0.25 ± 0.05 / 96.85 ± 0.17 | 0.26 ± 0.12 / 96.76 ± 0.10 | 1.64 ± 1.09 / 97.06 ± 0.10 | 96.96 ± 0.05 |

upstream FPR. Conversely, MSPC retains an ASR of 95.33%, consistent with its upstream TPR of 86.20% being insufficient to remove enough poisoned samples to prevent the backdoor from surviving retraining; its FPR of 14.76% also reduces CA to 91.59%. In the No attack setting, BlindClean achieves CA of 94.71% on CIFAR-10 and 96.96% on SVHN, each only 0.11 percentage points below the corresponding accuracy obtained without sanitization.

### 5.3 Generalization to Downstream Architectures

To evaluate generalization to downstream architectures, BlindClean sanitizes the potentially poisoned training data using a ResNet-18 feature extraction model. VGG19-BN (Simonyan & Zisserman, 2014) and DenseNet-121 (Huang et al., 2017) are then retrained from scratch on either the original data or the sanitized data. Table 3 reports the resulting downstream performance on CIFAR-10.

Table 3: Generalization to downstream architectures on CIFAR-10. Cells report ASR/CA (%); BlindClean results are means ± 95% confidence intervals over five random seeds.

| Architecture | Training data | LC | SA | Narcissus |
|---|---|---|---|---|
| VGG19-BN | No removal | 99.44/93.01 | 78.50/93.02 | 99.87/92.88 |
| | **BlindClean** | 0.78 ± 0.28 / 92.94 ± 0.22 | 0.92 ± 0.43 / 92.65 ± 0.22 | 0.26 ± 0.27 / 92.87 ± 0.21 |
| DenseNet-121 | No removal | 95.60/88.36 | 10.90/87.99 | 99.00/88.37 |
| | **BlindClean** | 1.14 ± 0.41 / 88.21 ± 0.16 | 0.98 ± 0.52 / 87.47 ± 0.28 | 0.19 ± 0.23 / 87.85 ± 0.29 |

For both architectures, BlindClean reduces ASR to at most 1.14% under all three attacks while keeping CA within 0.52 percentage points of training on the unsanitized data. Under SA, DenseNet-121 already shows a low no-removal ASR of 10.90%, indicating weaker attack transfer to this architecture. These results show that data sanitized using ResNet-18 remains effective for downstream models with different architectures.

### 5.4 Ablation Studies

**Feature extraction layers.** Individual feature extraction layers and cross-layer aggregation are compared for target-class determination. The hypergeometric $z$-score gap between the attacked target class and the highest-scoring clean class is reported; a larger positive gap indicates more reliable target-class determination. As shown in Table 4, the optimal individual layer differs across attacks: Layer 4 performs best for LC and Narcissus, whereas Layer 1 is most effective for SA. In contrast, All Layers achieves the largest gap under LC and Narcissus and maintains a positive gap under SA. These results show that cross-layer aggregation reduces reliance on an attack-specific layer and provides more consistent target-class determination.

Table 4: Target-score gaps under different feature extraction settings on CIFAR-10.

| Setting | LC | SA | Narcissus |
|---|---|---|---|
| Layer 1 | 10.326 | **15.501** | 7.688 |
| Layer 2 | 0.836 | 8.563 | 4.975 |
| Layer 3 | -0.033 | 4.201 | 3.539 |
| Layer 4 | 18.738 | 2.605 | 23.187 |
| All Layers | **24.681** | 9.079 | **26.295** |

**Automatic perturbation-scale selection.** The effective perturbation scale varies across attacks, such that selecting a fixed scale can lead to suboptimal poisoned-sample detection. Figure 2 compares the scale selected by minimum GMM posterior entropy with the oracle scale that achieves the highest $F_1$ score using ground-truth poison labels. The oracle is reported only for comparison and is not used by BlindClean. The entropy criterion selects the $F_1$-optimal scale for LC and Narcissus and a near-optimal scale for SA. Because posterior entropy is computed solely from the OOD-score distribution, BlindClean can select an effective scale without poison labels or attack-specific prior knowledge.

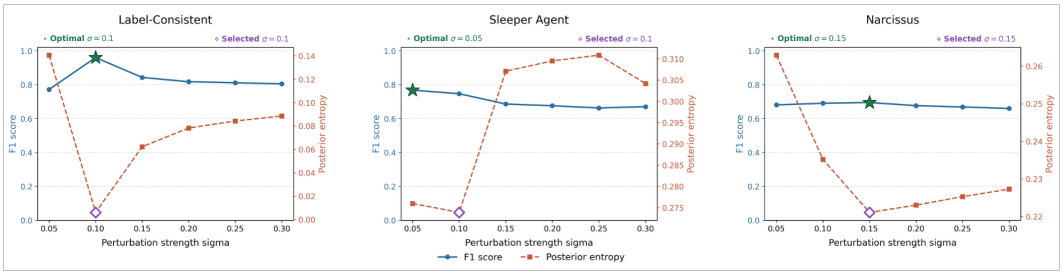



Figure 2: $F_1$ score and GMM posterior entropy across perturbation scales.

## 6 Conclusion

This paper presents BlindClean, a prior-knowledge-free data sanitization method for clean-label backdoor attacks. BlindClean uses multi-scale input perturbations to measure cross-layer feature stability, estimates class-wise reference feature distributions from low-stability samples, and uses the deviations of high-stability samples from these distributions to identify the attacked target class. It then detects poisoned samples only within the identified class. Experiments on CIFAR-10 and SVHN show that BlindClean achieves an average TPR/FPR of 99.40%/0.42% across the evaluated attack–dataset settings while preserving clean accuracy after downstream retraining. BlindClean requires only a potentially poisoned training dataset and does not rely on trusted clean reference data, poisoning-rate estimates, or target-class information. These results show that BlindClean provides a practical approach to sanitizing untrusted third-party training data against clean-label backdoor attacks.

### AI USE STATEMENT

In this work, we used generative AI tools to polish the writing to improve readability, to draft parts of the paper, and to identify relevant literature. We also used generative AI tools for the following tasks with required disclosure: providing feedback on the experimental design, including suggestions on which additional experiments and analyses would strengthen the evaluation, and assisting with the interpretation of experimental results. We have reviewed all AI-assisted work: all AI-drafted text was revised by the authors, all experiments were implemented and run by the authors, all interpretations of the results were checked against the experimental data by the authors, and all cited papers were read and verified by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

Section 4 describes each step of BlindClean, and Appendix A provides the hardware and software environment, training configurations, all BlindClean hyperparameters, and baseline settings. All BlindClean results are averaged over five random seeds and reported with 95% confidence intervals.

## REFERENCES

Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Yilun Chen, Ami Wiesel, Yonina C Eldar, and Alfred O Hero. Shrinkage algorithms for mmse covariance estimation. *IEEE transactions on signal processing*, 58(10):5016–5029, 2010.

Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Sebastiano Vascon, Werner Zellinger, Bernhard A Moser, Alina Oprea, Battista Biggio, Marcello Pelillo, and Fabio Roli. Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Computing Surveys*, 55(13s):1–39, 2023.

Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Mądry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1563–1580, 2022.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. Spectre: Defending against backdoor attacks using robust statistics. In *International Conference on Machine Learning*, pp. 4129–4139. PMLR, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Linshan Hou, Ruili Feng, Zhongyun Hua, Wei Luo, Leo Yu Zhang, and Yiming Li. IBD-PSC: Input-level backdoor detection via parameter-oriented scaling consistency. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 18992–19022. PMLR, 2024.

Linshan Hou, Wei Luo, Zhongyun Hua, Songhua Chen, Leo Yu Zhang, and Yiming Li. Flare: Toward universal dataset purification against backdoor attacks. *IEEE Transactions on Information Forensics and Security*, 20:6459–6473, 2025.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 2261–2269. IEEE, 2017.

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

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems*, 34:14900–14912, 2021.

Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE transactions on neural networks and learning systems*, 35(1):5–22, 2022.

Yiming Li, Mengxi Ya, Yang Bai, Yong Jiang, and Shu-Tao Xia. Backdoorbox: A python toolbox for backdoor learning. *arXiv preprint arXiv:2302.01762*, 2023.

Wanlun Ma, Derui Wang, Ruoxi Sun, Minhui Xue, Sheng Wen, and Yang Xiang. The "Beatrix" resurrections: Robust backdoor detection via gram matrices. In *Proceedings of the 30th Annual Network and Distributed System Security Symposium (NDSS 2023)*, 2023.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 4. Granada, 2011.

Soumyadeep Pal, Yuguang Yao, Ren Wang, Bingquan Shen, and Sijia Liu. Backdoor secrets unveiled: Identifying backdoor data with optimized scaled prediction consistency. In *International conference on learning representations*, volume 2024, pp. 14061–14083, 2024.

Minzhou Pan, Yi Zeng, Lingjuan Lyu, Xue Lin, and Ruoxi Jia. {ASSET}: Robust backdoor data detection across a multiplicity of deep learning paradigms. In *32nd USENIX security symposium (USENIX security 23)*, pp. 2725–2742, 2023.

Yuji Roh, Geon Heo, and Steven Euijong Whang. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1328–1347, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Hossein Souri, Liam Fowl, Rama Chellappa, Micah Goldblum, and Tom Goldstein. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. *Advances in Neural Information Processing Systems*, 35:19165–19178, 2022.

Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *Advances in neural information processing systems*, 31, 2018.

Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

Baoyuan Wu, Hongrui Chen, Mingda Zhang, Zihao Zhu, Shaokui Wei, Danni Yuan, and Chao Shen. Backdoorbench: A comprehensive benchmark of backdoor learning. *Advances in Neural Information Processing Systems*, 35:10546–10559, 2022.

Lior Yasur, Tzvi Lederer, Lior Rokach, and Yisroel Mirsky. Clean-label backdoor attacks: A survey. *IEEE Access*, 2026.

Yi Zeng, Minzhou Pan, Hoang Anh Just, Lingjuan Lyu, Meikang Qiu, and Ruoxi Jia. Narcissus: A practical clean-label backdoor attack with limited information. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pp. 771–785, 2023.

# A Implementation Details

**Experimental environment.** All experiments are conducted on a workstation with an Intel Core i7-9700K CPU, an NVIDIA GeForce RTX 3090 GPU with 24 GB of memory, and 64 GB of system memory. The implementation uses Python 3.12, PyTorch 2.10, Torchvision 0.25.0, CUDA 12.8, and Ubuntu 20.04 LTS. The implementation primarily uses BackdoorBench (Wu et al., 2022) and BackdoorBox (Li et al., 2023), together with the official implementations released for the corresponding attacks and data sanitization methods.

**Training configuration.** Unless otherwise stated, the defender's ResNet-18 feature extraction model and all downstream retraining models, including VGG19-BN and DenseNet-121 in the downstream-architecture generalization experiment, are trained from scratch for 200 epochs using stochastic gradient descent with momentum $0.9$ and weight decay $5 \times 10^{-4}$. An initial learning rate of $0.1$ and a MultiStepLR scheduler with milestones at epochs 100 and 150 and a decay factor of $0.1$ are used. The batch size is 512. VGG19-BN and DenseNet-121 serve only as downstream models and are not involved in feature extraction or poisoned-sample detection.

**BlindClean hyperparameters.** BlindClean uses perturbation scales

$$\mathcal{S} = \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}.$$

At each scale, Gaussian noise is independently added to the normalized input tensor to generate $M = 4$ perturbed samples per input. Features are taken from the outputs of the four residual stages (layer1–layer4) of the CIFAR ResNet-18, excluding the stem and the pooled embedding. Each feature map is reduced by global average pooling over its spatial positions, producing 64-, 128-, 256-, and 512-dimensional representations, respectively, so that $L = 4$ in all experiments; the same pooled representations are used for both the cosine feature shift and the Mahalanobis distance, without any feature-level $\ell_2$ normalization. The set of layers is configurable, and the four-stage setting is the default used throughout. Unless otherwise specified, the Unstable Tail ratio is set to $\alpha = 30\%$, and the Stable Tail ratio is set to $\beta = 10\%$. Thus, for each class, the bottom 30% and top 10% of samples ranked by feature stability are used as the Unstable Tail and Stable Tail, respectively. These hyperparameters remain fixed across experiments.

**Robust z-score normalization.** For feature-stability scoring, robust z-score normalization is applied independently for each class, feature extraction layer, and perturbation scale. The normalization center is the median of the corresponding feature-shift values, and the scale is $1.4826$ times their median absolute deviation. If this scale is non-finite or smaller than $10^{-8}$, it is replaced by the larger of the standard deviation and $10^{-8}$. For sample-wise OOD scoring, the same procedure is applied using the Mahalanobis distances of the corresponding Unstable Tail.

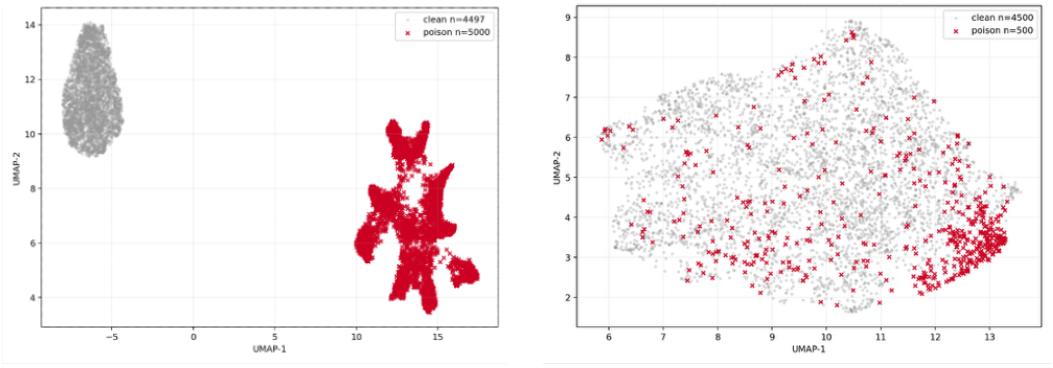
**Target-score edge-case handling.** For numerical stability, $Z_c^{(\sigma)}$ is set to $0$ when $N_c \leq 1$ or when the variance term inside the square root is at most $10^{-12}$; this zero score is retained in the multi-scale average.

**Baseline configurations.** For Beatrix, FLARE, ABL, ASSET, MSPC, and IBD-PSC, the experimental procedures, hyperparameter settings, and training configurations recommended in their original publications or official implementations are followed. For baseline methods that require trusted clean data, the amount of clean data provided follows the experimental settings specified in their original publications. Such clean data are available only to the corresponding baselines and are not provided to BlindClean. All methods are evaluated under the same hardware and software environment to minimize implementation-related variation.

# B Feature-Space Visualization: BadNets vs. Narcissus

Figure 3 compares the feature distributions of clean and poisoned samples under BadNets (Gu et al., 2017) and Narcissus (Zeng et al., 2023). BadNets is a dirty-label attack and is included here only as a reference point for comparison. Under BadNets, the poisoned samples form a cluster that is separated from the clean samples. In contrast, poisoned samples generated by Narcissus are interspersed

with clean samples and do not form a distinct cluster. This comparison illustrates why global separation in the original feature space may be insufficient for detecting poisoned samples under clean-label backdoor attacks. The visualizations are obtained by applying UMAP to the penultimate-layer features of CIFAR-10 samples, which provides a compact view of the class-level feature structure; BlindClean itself operates on the multi-layer features described in Appendix A.




**(a) BadNets** **(b) Narcissus**

Figure 3: UMAP visualizations of penultimate-layer features for clean and poisoned CIFAR-10 samples. Poisoned samples form a separated cluster under BadNets, whereas Narcissus poisoned samples overlap substantially with clean samples.

## C Layer-wise Stability-score AUC

Table 5 reports the ROC AUC obtained from the sample rankings induced by the cosine feature shift at each individual layer. A higher AUC indicates stronger separability between poisoned and clean samples at the corresponding layer. LC and Narcissus exhibit stronger separability in deeper layers, whereas SA achieves stronger separability in shallower layers. These results show that the most informative network layer differs across attacks. Therefore, relying on a single fixed layer may not be effective across different clean-label backdoor attacks. BlindClean consequently aggregates feature shifts across multiple layers to establish cross-layer feature stability.

Table 5: Layer-wise ROC AUC on CIFAR-10 for separating poisoned and clean samples based on cosine feature shifts.

| Attack | Layer 1 | Layer 2 | Layer 3 | Layer 4 |
|---|---|---|---|---|
| LC | 0.908 | 0.925 | **0.976** | 0.933 |
| SA | **0.969** | 0.955 | 0.896 | 0.834 |
| Narcissus | 0.943 | 0.967 | **0.998** | 0.965 |

## D Mahalanobis versus Euclidean Distance

Table 6 compares squared Euclidean and squared Mahalanobis distances for samples in the attacked target class, using the CIFAR-10 models of Table 2. For each attack, both metrics are evaluated at the perturbation scale selected by the entropy criterion of Equation 10 and use the same Unstable Tail as the reference set. The squared Euclidean distance at layer $l$ is taken to the same reference mean $\mu_c^{(l,\sigma)}$ used by the squared Mahalanobis distance of Equation 5, so that the two metrics differ only in whether the reference covariance is taken into account. For both metrics, the layer-wise distances are normalized using robust z-score normalization based on the corresponding Unstable-Tail distance distribution and then aggregated by the median across the four analyzed layers.

The Poison–Clean Score Gap is defined as the median aggregated score of all poisoned samples in the target class minus that of the same number of clean samples in that class with the highest

feature-stability scores. Ground-truth poison labels are used only to compute this evaluation metric and are not used by BlindClean. Both metrics yield positive gaps across all three attacks. However, Mahalanobis-based scores consistently produce larger gaps: from 3.55 to 22.32 under LC, from 2.64 to 4.55 under SA, and from 1.55 to 9.47 under Narcissus. These results indicate that Mahalanobis distance provides greater median score separation between poisoned samples and stable clean samples relative to the reference distributions.

Table 6: Target-class Poison–Clean Score Gap on CIFAR-10 under different distance metrics. Higher is better.

| Attack | Squared Euclidean | Squared Mahalanobis |
|---|---|---|
| LC | 3.55 | **22.32** |
| SA | 2.64 | **4.55** |
| Narcissus | 1.55 | **9.47** |

## E Robustness to Different Poisoning Rates

BlindClean is evaluated under target-class poisoning rates of 5%, 10%, 20%, and 30%, which correspond to overall poisoning rates of approximately 0.5%, 1%, 2%, and 3% of the CIFAR-10 training dataset, respectively. Table 7 reports upstream poisoned-sample detection performance, and Table 8 reports downstream performance after retraining on the sanitized dataset.

**Upstream poisoned-sample detection.** Table 7 shows that BlindClean maintains high TPRs and low FPRs across poisoning rates. Under LC, it achieves a TPR of 100.00% at all evaluated rates, while the FPR decreases from 0.27% to 0.01% as the target-class poisoning rate increases. Under SA, the TPR ranges from 97.00% to 99.32%, and the FPR remains at or below 0.72%. Under Narcissus, BlindClean achieves near-perfect TPRs and maintains FPRs below 1% at all rates.

Table 7: Upstream poisoned-sample detection under different target-class poisoning rates on CIFAR-10. Each cell reports TPR/FPR (%). BlindClean results are means $\pm$ 95% confidence intervals over five random seeds.

| Attack | 5% | 10% | 20% | 30% |
|---|---|---|---|---|
| LC | $100.00 \pm 0.00$ / $0.27 \pm 0.02$ | $100.00 \pm 0.00$ / $0.08 \pm 0.01$ | $100.00 \pm 0.00$ / $0.02 \pm 0.00$ | $100.00 \pm 0.00$ / $0.01 \pm 0.00$ |
| SA | $97.20 \pm 0.99$ / $0.68 \pm 0.03$ | $97.00 \pm 0.53$ / $0.61 \pm 0.05$ | $99.32 \pm 0.38$ / $0.72 \pm 0.09$ | $98.84 \pm 0.21$ / $0.46 \pm 0.05$ |
| Narcissus | $98.40 \pm 0.35$ / $0.73 \pm 0.02$ | $99.88 \pm 0.14$ / $0.91 \pm 0.02$ | $99.90 \pm 0.00$ / $0.43 \pm 0.02$ | $100.00 \pm 0.00$ / $0.65 \pm 0.02$ |

**Downstream performance after retraining.** Table 8 shows that retraining on BlindClean-sanitized data substantially reduces ASR across poisoning rates while preserving CA. Under LC, BlindClean reduces the ASR below 1% at all rates. Under SA, the ASR ranges from 0.26% to 1.14% while CA remains above 94%. Under Narcissus, the ASR is substantially reduced relative to the corresponding no-defense results, although it remains above 1% at several evaluated poisoning rates.

## F Feature Stability Metric Analysis

BlindClean ranks samples by feature stability to construct the Stable Tail of each class. Cosine distance (Cos), L2 distance (L2), and an equally weighted combination of the two (Half) are compared as feature stability metrics on CIFAR-10. Figure 4 reports the poison purity, namely the proportion of poisoned samples among the top 10% most stable samples. Cos achieves the highest or near-highest purity across all three attacks, indicating that it effectively concentrates poisoned samples in the Stable Tail.

Table 8: Downstream performance after retraining under different target-class poisoning rates on CIFAR-10. Each cell reports ASR/CA (%). BlindClean results are means ± 95% confidence intervals over five random seeds.

| Attack | Training data | 5% | 10% | 20% | 30% |
|---|---|---|---|---|---|
| LC | No defense | 98.22/94.53 | 99.60/94.55 | 99.96/94.87 | 99.89/94.73 |
| | **BlindClean** | 0.96 ± 0.18 / 94.50 ± 0.37 | 0.97 ± 0.19 / 94.54 ± 0.33 | 0.84 ± 0.16 / 94.72 ± 0.17 | 0.67 ± 0.13 / 94.42 ± 0.20 |
| SA | No defense | 78.70/93.90 | 92.00/94.59 | 96.40/93.87 | 91.40/94.19 |
| | **BlindClean** | 1.14 ± 1.21 / 94.36 ± 0.28 | 0.50 ± 0.18 / 94.02 ± 0.28 | 0.54 ± 0.27 / 94.08 ± 0.33 | 0.26 ± 0.07 / 94.02 ± 0.27 |
| Narcissus | No defense | 99.97/95.17 | 100.00/95.01 | 100.00/95.21 | 100.00/95.15 |
| | **BlindClean** | 2.79 ± 2.06 / 95.08 ± 0.10 | 0.34 ± 0.24 / 94.89 ± 0.37 | 3.29 ± 1.73 / 95.06 ± 0.23 | 1.08 ± 1.31 / 94.87 ± 0.09 |

Figure 5 further reports stability-score AUC under different feature stability metrics and perturbation scales. Cos achieves consistently strong separability across attacks and scales, whereas L2 is more sensitive to the perturbation scale. Half does not consistently improve upon Cos. These results support using cosine distance as the feature stability metric in BlindClean.

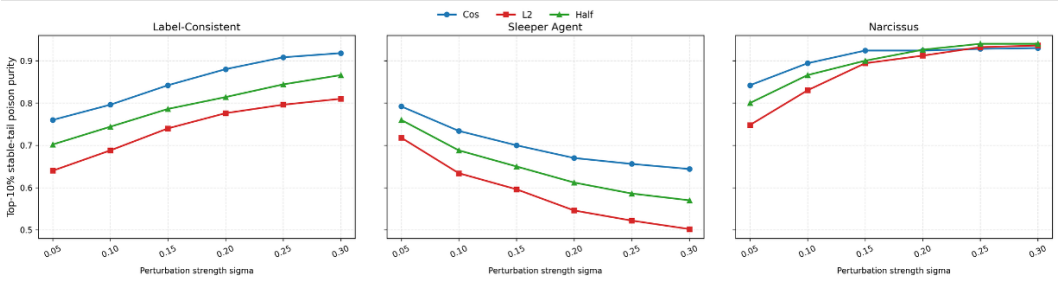



Figure 4: Poison purity among the top 10% most stable samples under different feature stability metrics on CIFAR-10.

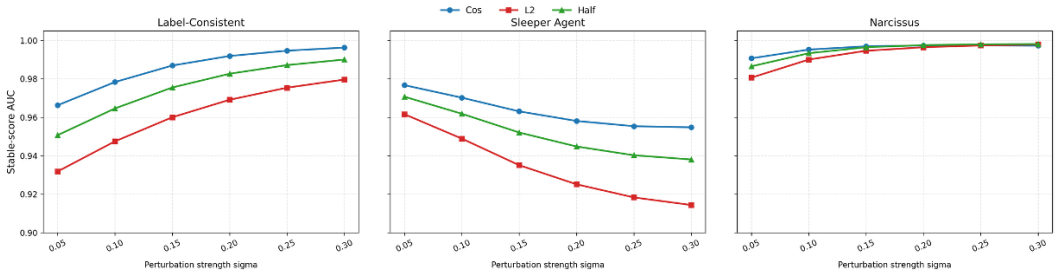



Figure 5: Stability-score AUC under different feature stability metrics and perturbation scales on CIFAR-10.

## G Computational Cost Analysis

The runtime of BlindClean is compared with that of six methods for poisoned-sample identification under the same hardware environment. All methods are evaluated on CIFAR-10, and the reported runtimes are averaged across LC, SA, and Narcissus. For every method, runtime is measured after

any required model training has been completed and includes only the subsequent poisoned-sample identification procedure. As shown in Table 9, ABL and IBD-PSC are the fastest methods, requiring 11.32 and 19.71 seconds, respectively, but exhibit substantially lower detection performance than BlindClean across the evaluated attacks. BlindClean requires 228.96 seconds, which is lower than the runtimes of Beatrix and FLARE, at 299.19 and 335.19 seconds, respectively. Despite performing multi-scale perturbation, cross-layer feature analysis, target-class determination, and GMM-based poisoned-sample detection, BlindClean requires substantially less time than ASSET and MSPC, which require 2055.70 and 1634.57 seconds, respectively. These results indicate that BlindClean achieves strong poisoned-sample identification performance with practical computational cost.

Table 9: Runtime comparison for poisoned-sample identification on CIFAR-10, averaged across LC, SA, and Narcissus. Model training time is excluded for all methods. Relative runtime is normalized to BlindClean.

| Method | Runtime (s) | Relative runtime |
|---|---|---|
| Beatrix | 299.19 | 1.31× |
| FLARE | 335.19 | 1.46× |
| ABL | 11.32 | 0.05× |
| ASSET | 2055.70 | 8.98× |
| MSPC | 1634.57 | 7.14× |
| IBD-PSC | 19.71 | 0.09× |
| **BlindClean** | 228.96 | 1.00× |