BEYOND A SINGLE ATTACK: GENERALIZABLE BACKDOOR DETECTION VIA SINGLE-ATTACK META-CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Backdoor attacks pose a serious security threat to deep neural networks, motivating model-level detection methods that identify whether a target model contains a hidden backdoor before deployment. Meta-classifier-based approaches are particularly attractive because they learn reusable detectors from model-level representations extracted from benign and backdoored reference models. However, we find that existing representations generalize poorly across heterogeneous attacks. Under cross-attack evaluation on CIFAR-10 with ResNet-18, the existing work MNTD and WeightMM achieve average AUCs of only 0.479 and 0.729, respectively. Our analysis suggests that these representations tend to capture attack-specific characteristics rather than model changes shared across different backdoor attacks. To address this limitation, we propose a novel model-level backdoor detection method by leveraging class-wise decision-space probing. Our key insight is that although backdoor attacks may differ substantially in their triggers and injection mechanisms, a successful attack must ultimately alter the model's decision behavior. We therefore isolate the final classifier and, for each output class, optimize a classifier-input representation that maximizes the corresponding class response. The resulting class-wise representations are aggregated into a model-level representation for meta-classifier training. We evaluate our method under a strict cross-attack protocol covering six dataset–architecture configurations and 12 representative backdoor attacks. Across the evaluated settings, our method achieves an average cross-attack AUC of 0.966, with the best performance reaching 0.998.

1 INTRODUCTION

Deep neural networks (DNNs) have achieved remarkable performance and have been widely deployed in a variety of real-world applications. However, extensive studies have shown that DNNs are vulnerable to backdoor attacks (Li et al., 2024). A backdoored model typically behaves normally on benign inputs, but produces an attacker-specified prediction when presented with an input containing a trigger (Gu et al., 2019; Li et al., 2024). Therefore, identifying potential backdoors before model deployment has become an important security problem.

Meta-classifier-based backdoor detection provides an appealing solution to this problem. These methods extract model-level representations from a collection of benign and backdoored reference models and train a binary classifier to distinguish between them. Once trained, the detector can be repeatedly applied to new models at relatively low marginal cost. Existing methods typically characterize target models using two types of information. One line of work constructs model-level representations from model responses to optimized query inputs, such as MNTD (Xu et al., 2021) and ULPs (Kolouri et al., 2020), while another directly extracts representations from model parameters, as in WeightMM (Langosco et al., 2023). The key advantage of this paradigm is that it transforms backdoor detection into a reusable model-level classification problem.

However, our experimental results show that detection performance on the source attack used for detector training does not reliably transfer to different target attacks. On CIFAR-10 with ResNet-18, the average cross-attack AUCs of MNTD and WeightMM are only 0.479 and 0.729, respectively.

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.

Our analysis suggests that existing model-level representations are more likely to capture attack-specific characteristics than model changes shared across different backdoor attacks.

We analyze this problem from a functional perspective. Although different backdoor attacks can vary substantially in their implementations, a successful attack must ultimately alter the model's decision behavior such that inputs containing the trigger are consistently classified into the attacker-specified target class. This suggests that transferable backdoor information may not lie in attack-specific artifacts, but rather in the **class-wise decision structure** formed by the model.

Based on this observation, we propose a meta-classifier-based backdoor detection method using class-wise decision-space probing. Specifically, we isolate the final classifier from the target model and independently optimize a classifier-input representation for each output class to maximize the response of the corresponding class. The resulting class-wise representations are then aggregated into a model-level representation and used to train a meta-classifier.

We evaluate our method under a strict cross-attack protocol covering six dataset–architecture configurations and 12 representative backdoor attacks. Across the evaluated settings, our method achieves an average cross-attack AUC of 0.966, with the best performance reaching 0.998, substantially outperforming existing meta-classifier-based detectors. These results demonstrate the effectiveness of class-wise decision-space probing for generalizing to backdoor attacks unseen during detector training.

Our main contributions are threefold:

- We study meta-classifier-based backdoor detection from the perspective of cross-attack generalization and show that strong detection performance on source attacks does not necessarily translate into effective generalization to cross attacks.
- We propose a class-wise decision-space probing method that actively optimizes a classifier-input representation for each output class and aggregates these class-wise representations into a model-level representation, enabling the characterization of class-wise decision structure without requiring training data or prior knowledge of the attack.
- We conduct extensive experiments across 12 backdoor attacks, showing that a detector trained on a single attack can generalize effectively to the remaining 11 unknown attacks.

2 Related Work

Backdoor attacks. BadNets (Gu et al., 2019) is a pioneering backdoor attack in which the attacker poisons training samples by stamping a predefined patch and assigning them to an attacker-specified target class. Subsequent attacks have substantially diversified trigger construction. **Sample-agnostic attacks** apply a shared trigger to different poisoned samples: Blended (Chen et al., 2017) mixes a trigger pattern with the original image, SIG (Barni et al., 2019) injects a sinusoidal signal, and Low-Frequency (Zeng et al., 2021) constructs smooth triggers dominated by low-frequency components. In contrast, **sample-specific attacks** generate different triggers for different inputs. Input-Aware (Nguyen & Tran, 2020) learns an input-dependent trigger generator, WaNet (Nguyen & Tran, 2021) uses image warping as the backdoor pattern, and SSBA (Li et al., 2021b) generates sample-specific invisible perturbations. The increasing diversity of trigger construction makes it difficult for detection methods to rely on characteristics specific to a particular attack.

Backdoor defense. Existing defenses can be broadly divided according to whether they inspect suspicious samples or the trained model itself. **Sample-level detection** attempts to identify poisoned examples from their abnormal behavior. Activation Clustering (Chen et al., 2018) separates poisoned and benign samples by clustering their internal representations, Spectral Signatures (Tran et al., 2018) detects outliers from spectral structure in feature representations, STRIP (Gao et al., 2019) identifies triggered inputs through prediction entropy under input perturbations, and ABL (Li et al., 2021a) exploits the tendency of poisoned samples to be learned unusually quickly. **Model-level detection**, in contrast, determines whether an already trained model contains a backdoor. Neural Cleanse (Wang et al., 2019) reverse-engineers candidate triggers for different target classes and identifies backdoored models from anomalous trigger patterns. ABS (Liu et al., 2019) analyzes neuron-level abnormalities by artificially stimulating hidden neurons and observing their effects on

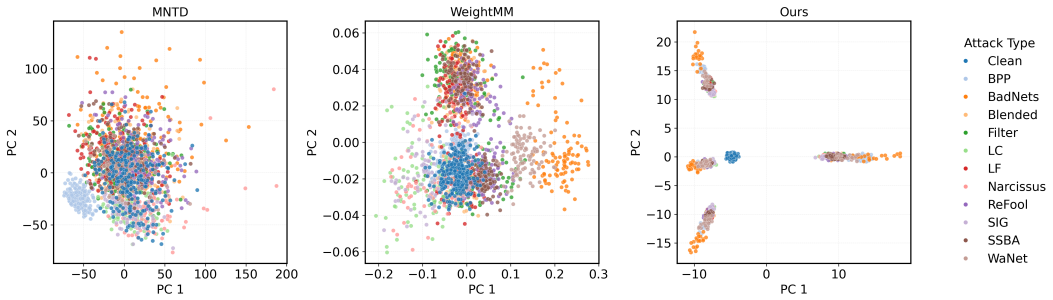


Figure 1: PCA visualization of different model-level representations on CIFAR-10 with ResNet-18. We compare MNTD, WeightMM, and our decision space representation. Our representation shows a clearer separation between clean and backdoored models under heterogeneous attack types, indicating that decision-space probing better captures backdoor-related model-level changes.

output activations. Since our setting concerns the detection of an unknown trained model, we focus on model-level detection, particularly meta-classifier-based approaches.

Meta-classifier-based detection. These methods learn a detector from a collection of benign and backdoored reference models, but differ primarily in how a target model is represented (Xu et al., 2021; Kolouri et al., 2020; Langosco et al., 2023). MNTD (Xu et al., 2021) jointly optimizes a set of query inputs and a meta-classifier, representing each model by its responses to the learned queries. ULPs (Kolouri et al., 2020) similarly learn universal probing patterns whose output responses are used to distinguish benign and backdoored models. WeightMM (Langosco et al., 2023) directly feeds model parameters, or representations derived from them, into a meta-model for detection. Our work follows the classic meta-classification paradigm, but differs in the model representation: instead of representing a model through responses to learned input queries or directly through its parameters, we characterize the class-wise decision structure induced by the target model.

3 Motivation

Backdoor attacks continue to evolve, with increasingly diverse trigger patterns, poisoning strategies, and injection mechanisms (Li et al., 2024). This places meta-classifier-based model-level detection in a reactive arms race: if every newly emerging attack requires the detector to be redesigned or retrained, the defense will continually lag behind the attacker. A more practical objective is therefore to learn transferable backdoor characteristics from existing attacks and use them to detect previously unseen threats. Achieving this goal requires a model representation that captures properties shared across different backdoor attacks rather than patterns specific to individual attacks.

To examine whether existing representations satisfy this requirement, we conduct a preliminary visualization study on CIFAR-10 with ResNet-18. For each target model, we extract three types of model representations: the representation based on query responses used by MNTD (Xu et al., 2021), the representation based on model weights used by WeightMM (Langosco et al., 2023), and the decision space representation introduced in this work. We then project these representations into two dimensions using PCA. As shown in Fig. 1, benign and backdoor models substantially overlap under the MNTD representation, while the representation based on model weights also shows considerable mixing across different attack types. In contrast, the decision space representation produces a noticeably clearer separation between benign and backdoored models.

This observation suggests that the limited generalization across different attacks may arise from the inability of existing detectors to capture model changes that are consistently induced by different backdoor attacks. Methods such as MNTD (Xu et al., 2021) characterize a model through its responses to a limited set of learned probing inputs. Since these probes are optimized using the backdoored models available during detector training, they are more likely to capture abnormal responses associated with attacks observed during training. When a cross attack produces different trigger patterns or behavioral changes, the same probes may no longer reveal sufficiently distinc-

tive responses. Methods based on model weights take a different approach by directly encoding model parameters. However, backdoor functionality does not necessarily correspond to a consistent pattern in parameter space. Similar malicious behaviors can be realized through different parameter configurations, while variations caused by initialization and training can introduce substantial parameter differences unrelated to backdoors. Consequently, both types of representations may capture variations associated with particular attacks or training procedures rather than changes that are consistently shared across different backdoor attacks.

This observation motivates a different perspective. Although backdoor attacks can differ substantially in their trigger patterns, poisoning strategies, and injection mechanisms, a successful backdoor must ultimately alter the decision behavior of the model so that particular conditions systematically favor an output specified by the attacker (Li et al., 2024; Gu et al., 2019). The specific way in which this behavior is implemented may vary across attacks, but its effect must eventually be reflected in how the model forms decisions for different classes. We therefore speculate that the model's **decision structure** provides a more consistent source of information across different attacks than either responses to a limited set of fixed inputs or raw parameter values.

To this end, we shift from passively observing model outputs or directly encoding model parameters to actively probing the decision space of the final classifier. Instead of relying on a fixed set of inputs, we examine how the classifier responds along different class-wise decision directions under a unified probing procedure. In this way, we aim to obtain a model representation that better reflects decision-level changes shared across heterogeneous backdoor attacks and therefore transfers more effectively to attacks that are unseen during detector training.

4 METHODOLOGY

4.1 PROBLEM SETTING

Attacker goal and capability. Following the standard threat model in model-level backdoor detection (Xu et al., 2021; Kolouri et al., 2020; Li et al., 2024), the attacker aims to produce a backdoored model that preserves normal performance on benign inputs while mapping trigger-bearing inputs to an attacker-specified target class. The attacker may choose the trigger pattern, poisoning ratio, and poisoning strategy, which are unknown to the defender.

Defender goal and capability. Given a target model, the defender aims to determine whether it is benign or backdoored. We assume white-box access to the target model and allow forward and backward propagation. To train the meta-classifier, the defender uses a reference dataset to construct benign and backdoored reference models with known attacks. The defender does not require the original training data of the target model or prior knowledge of its trigger, target label, poisoning strategy, or attack type.

Practical use. A representative use case is screening third-party models in a model supply chain or marketplace before deployment. Since the platform may not be able to reproduce an equivalent clean version of each submitted model, the trained meta-classifier can serve as a reusable screening tool.

4.2 DETECTION METHOD OVERVIEW

Our method consists of four stages. We first decompose the target model and isolate its final classifier, then perform class-wise decision-space probing for each output class, aggregate the resulting representations into a model-level representation, and finally train a meta-classifier for backdoor detection. We elaborate each stage in the following sections.

4.3 MODEL DECOMPOSITION AND DECISION SPACE

Consider a K-class classification model $f_\theta : \mathcal{X} \to \mathbb{R}^K$. We represent it as the composition of a feature extractor g_θ and a classifier h_θ, such that $f_\theta(x) = h_\theta(g_\theta(x))$. The feature extractor maps an input image to a decision-relevant feature representation, while the classifier produces the prediction based on the extracted features. Specifically, the classifier $h_\theta : \mathbb{R}^d \to \mathbb{R}^K$ directly maps a d-dimensional feature representation to K class logits. For a backdoored model, trigger-induced

feature changes must ultimately affect the model's class decision through this mapping. Therefore, we directly explore the input feature space of h_θ.

In this work, we instantiate h_θ as the final linear classification layer of the network and perform class-wise optimization directly in its input space. This procedure explores the classifier's response behavior along different output-class directions, while avoiding repeated forward and backward propagation through the feature extractor during optimization. We refer to this procedure as **decision-space probing**. Further analysis of the design choice of using the final classifier is provided in Sec. 5.3.

4.4 Class-wise Decision-Space Probing

Given the input feature space of the final classifier defined above, we construct a class-conditioned representation for each output class. For class $k \in \{1, \ldots, K\}$, we define an optimizable feature vector $z_k \in \mathbb{R}^d$ in the input space of h_θ. During optimization, all model parameters are kept fixed and only z_k is updated.

To explore the classifier toward class k, we optimize z_k using class k as the target:

$$\mathcal{L}(z_k; k) = -\log \frac{\exp([h_\theta(z_k)]_k)}{\sum_{j=1}^{K} \exp([h_\theta(z_k)]_j)}. \tag{1}$$

All classes share the same fixed initialization, i.e., $z_k^{(0)} = z^{(0)}$, and are optimized independently for T iterations under the same optimization protocol. We use the resulting representation after T steps, $z_k^{(T)}$, as the class-wise decision representation for class k.

Repeating this procedure for all K classes produces $\{z_1^{(T)}, \ldots, z_K^{(T)}\}$. These representations are obtained by directly optimizing the classifier input rather than by forwarding predefined images through the network, allowing each model to be explored along its different output-class directions under a consistent procedure.

4.5 Model Representation Construction

After obtaining the optimized decision representations for all classes, we organize them into a matrix following a fixed class order:

$$\Phi(f_\theta) = \left[z_1^{(T)}; z_2^{(T)}; \ldots; z_K^{(T)}\right] \in \mathbb{R}^{K \times d}. \tag{2}$$

We define $\Phi(f_\theta)$ as the decision-space representation of the target model.

Before feeding the representation into the detector, we flatten the matrix into a one-dimensional vector:

$$\phi(f_\theta) = \mathrm{vec}\left(\Phi(f_\theta)\right) \in \mathbb{R}^{Kd}. \tag{3}$$

Unlike representations constructed by directly vectorizing model parameters, our representation does not simply record the static weights of the final classifier. Instead, it is induced by the classifier's class-wise responses under a common optimization procedure. The resulting representation therefore provides an optimization-derived representation of the classifier rather than a direct encoding of its raw parameter values.

4.6 Meta Classifier Training

Given a set of reference models $\{(f_{\theta_i}, y_i)\}_{i=1}^{N}$, where $y_i = 0$ denotes a clean model and $y_i = 1$ denotes a backdoored model, we extract a representation $\phi(f_{\theta_i})$ for each model and use these representations to train a binary meta-classifier

$$D_\psi : \mathbb{R}^{Kd} \to \mathbb{R}. \tag{4}$$

The meta classifier learns to distinguish clean and backdoored models from their model-level representations. In our implementation, D_ψ is instantiated as a lightweight 1D convolutional network and trained using binary cross-entropy with logits. The architecture and training details are provided in Appendix B.3.

At test time, we apply the same decision-space probing and representation construction procedure to an unknown target model f_θ and feed the representation $\phi(f_\theta)$ into the trained meta-classifier. The output logit is passed through a sigmoid function to obtain the backdoor detection score.

5 EXPERIMENTS

We evaluate the proposed method across diverse backdoor attacks, datasets, and model architectures. We compare it with MNTD (Xu et al., 2021) and WeightMM (Langosco et al., 2023), and provide further analysis through ablation studies. Due to space limitations, we defer additional details and extended results to the Appendix.

5.1 EXPERIMENTAL SETUP

Datasets and models. We evaluate six representative dataset–architecture configurations. Specifically, CIFAR-10 (Krizhevsky et al., 2009) is paired with the CNN used by MNTD (Xu et al., 2021), MobileNet (Sandler et al., 2018), and ResNet-18 (He et al., 2016) to examine different model architectures on the same dataset; GTSRB10 (Zhao et al., 2026) and the full GTSRB (Stallkamp et al., 2012) are paired with ResNet-18 to examine different numbers of classes under a fixed architecture; and CIFAR-100 (Krizhevsky et al., 2009) with ResNet-34 (He et al., 2016) provides an additional setting with a larger class space and a deeper network.

Backdoor attacks. We generate backdoored reference models using the BackdoorBench (Wu et al., 2022) implementations of 12 attacks: BadNets (Gu et al., 2019), Blended (Chen et al., 2017), BPP (Wang et al., 2022), Filter (Liu et al., 2019; Fu et al., 2023), Input-Aware (Nguyen & Tran, 2020), LC (Turner et al., 2019), LF (Zeng et al., 2021), Narcissus (Zeng et al., 2023), Refool (Liu et al., 2020), SIG (Barni et al., 2019), SSBA (Li et al., 2021b), and WaNet (Nguyen & Tran, 2021). Backdoor model generation settings and attack-specific configurations are provided in Appendix B.1.

Evaluation protocol. We report CACC, ACC, and ASR to characterize the target models. CACC denotes the clean accuracy of benign models, ACC denotes the clean accuracy of backdoored models, and ASR measures the attack success rate on triggered inputs. For backdoor detection, we use AUC as the evaluation metric. For each dataset–architecture configuration, we train a separate detector on each of the 12 attacks and evaluate it on both the source attack and the remaining 11 unseen attacks. Source AUC measures detection performance on the attack used for detector training, while Cross AUC is averaged over the 132 off-diagonal source–target attack pairs.

Baselines and implementation. We compare our method with MNTD (Xu et al., 2021) and WeightMM (Langosco et al., 2023). All methods use the same pool of benign and backdoored models and follow the same source-to-target evaluation protocol. Each baseline retains its own detector architecture and training procedure, as detailed in Appendix B.4. For our method, probing hyperparameters and meta-classifier training details are provided in Appendices B.2 and B.3, respectively.

5.2 CROSS-ATTACK DETECTION RESULTS

Overall performance. As shown in Table 1, our method consistently achieves strong cross-attack detection performance. Across the evaluated settings, the detector maintains high AUC when transferred from the observed source attack to previously unseen cross-attack types, while exhibiting only small source-to-cross performance gaps. Compared with MNTD (Xu et al., 2021) and WeightMM (Langosco et al., 2023), our method achieves higher cross-attack AUCs and substantially smaller transfer gaps in most settings. These results suggest that the proposed probing-based representation captures backdoor-related model characteristics that transfer across heterogeneous attacks, rather than relying primarily on attack-specific patterns.

The baselines exhibit different limitations under our evaluation. MNTD (Xu et al., 2021) shows both limited cross-attack transfer and sensitivity to the target-model architecture: it exhibits a source-to-cross AUC drop from 0.987 to 0.801 on CNN, while its source AUC decreases to 0.650 and 0.528 on ResNet-18 and MobileNet, respectively. WeightMM (Langosco et al., 2023) achieves stronger source performance across architectures, but still suffers noticeable degradation under cross-attack

Table 1: Summary of cross-attack detection performance. **Source** denotes the AUC evaluated on the source attack, averaged over all source attacks. **Cross** denotes the average AUC over all off-diagonal source to target attack pairs. **Transfer Gap** is defined as the difference between Source and Cross AUC. Complete source-to-target cross-attack matrices are provided in Appendix C.1.

Dataset	Model	CACC (%)	ACC (%)	ASR (%)	Method	Source ↑	Cross ↑	Transfer Gap ↓
CIFAR-10	CNN	84.32	82.93	97.46	MNTD	0.987	0.801	0.186
					WeightMM	0.984	0.815	0.169
					Ours	**0.995**	**0.983**	**0.012**
	MobileNet	82.45	80.54	97.05	MNTD	0.528	0.509	**0.019**
					WeightMM	0.953	0.861	0.092
					Ours	**0.988**	**0.939**	0.048
	ResNet-18	91.68	90.38	97.54	MNTD	0.650	0.479	0.171
					WeightMM	0.984	0.729	0.255
					Ours	**0.997**	**0.995**	**0.002**
GTSRB10	ResNet-18	87.39	86.10	97.95	MNTD	0.684	0.621	0.063
					WeightMM	**0.995**	0.949	0.045
					Ours	0.994	**0.989**	**0.005**
GTSRB	ResNet-18	98.13	97.90	98.04	MNTD	0.578	0.546	0.033
					WeightMM	0.998	0.854	0.143
					Ours	**0.999**	**0.998**	**0.001**
CIFAR-100	ResNet-34	52.03	49.78	96.11	MNTD	0.788	0.670	0.119
					WeightMM	0.908	0.741	0.167
					Ours	**0.965**	**0.889**	**0.077**

transfer, particularly on ResNet-18, where its AUC decreases from 0.984 to 0.729. In contrast, our method maintains consistently high source and cross-attack performance across all three architectures.

Impact of target-model architecture. We next examine whether this advantage remains consistent across different target-model architectures. On CIFAR-10 with CNN, MobileNet, and ResNet-18, our method achieves cross-attack AUCs of 0.983, 0.939, and 0.995, respectively. The consistently strong performance across these architectures indicates that the proposed representation remains effective under different model architectures.

Impact of number of classes. We evaluate the applicability of our method across datasets with different numbers of classes. On GTSRB10 and the full 43-class GTSRB dataset with ResNet-18, our method achieves cross-attack AUCs of 0.989 and 0.998, respectively. On CIFAR-100 with ResNet-34, it achieves a cross-attack AUC of 0.889. These results show that the proposed representation remains effective beyond CIFAR-10 and is applicable to classification tasks with substantially different numbers of classes.

5.3 Ablation Study

We conduct ablation studies to examine two key design choices of our method on CIFAR-10 with ResNet-18. First, we investigate **where the model should be probed**, evaluating whether the final linear classifier provides an effective trade-off between detection performance and computational cost. Second, after selecting the final classifier, we examine **how it should be represented**, asking whether directly using its parameters is sufficient or whether the proposed probing-based transformation is necessary for robust cross-attack detection.

Q1: Why probe the final linear classifier? We first investigate the effect of probing location on detection performance and computational cost. Starting from the final linear classifier, we progressively extend the probed submodel to earlier residual blocks and eventually the entire network. At each location, every class-wise representation is optimized for 1,000 iterations, while the remaining evaluation protocol is kept unchanged.

As shown in Fig. 2, probing only the final classifier achieves an AUC of 0.964 with an optimization time of 8.56 seconds. Although probing the last residual block slightly improves the AUC to 0.983, the optimization time increases to 31.03 seconds, while probing even earlier layers incurs higher computational cost without consistent performance gains. We therefore use the final linear classifier as the default probing location, providing an effective balance between detection effectiveness and optimization efficiency.

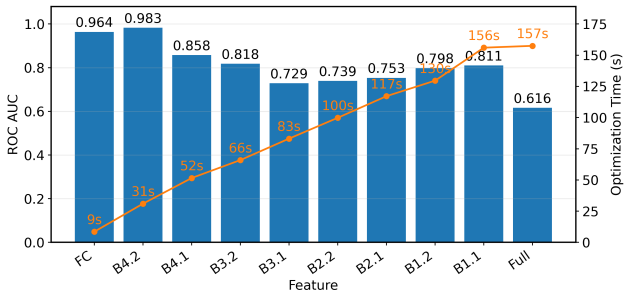


Figure 2: Effect of probing location on detection performance and optimization cost on CIFAR-10 with ResNet-18. FC denotes the final linear classifier, Bi.j denotes probing from the j-th BasicBlock of the i-th residual stage, and Full denotes the entire model. The optimization budget is fixed to 1,000 iterations.

Table 2: Comparison of direct final-layer parameters and our decision space representation on CIFAR-10 with ResNet-18 under Blended attack. Source and Cross denote average source and cross-attack AUC, while Min. Cross reports the lowest cross-attack AUC.

Representation	Dim.	Source	Cross	Min. Cross	Min. Pair
$\mathrm{vec}(W)$	5120	0.996	0.973	0.168	Input-Aware → BadNets
$\mathrm{vec}(W, b)$	5130	0.996	0.977	0.249	Input-Aware → BadNets
$\mathrm{vec}(\Phi(f_\theta))$	5120	**0.998**	**0.992**	**0.917**	Input-Aware → SIG

Q2: Why transform the final-layer representation through probing? Having established the final classifier as the probing location, we next examine whether its raw parameters already provide a sufficient representation for backdoor detection. If the effectiveness of our method simply originates from accessing the final classifier, directly vectorizing its parameters should achieve comparable cross-attack performance without the proposed probing procedure.

To test this hypothesis, we compare our representation with two direct parameter-based variants. Specifically, $\mathrm{vec}(W)$ directly vectorizes the weight matrix of the final linear classifier, while $\mathrm{vec}(W, b)$ additionally includes its bias term. Our method instead constructs $\mathrm{vec}(\Phi(f_\theta))$ through class-wise probing of the same classifier. All three representations use the same meta-classifier architecture, data splits, normalization procedure, and training protocol.

As shown in Table 2, directly using the final-layer parameters already achieves relatively high average cross-attack AUCs of 0.973 and 0.977. However, their performance is substantially less stable across individual attack transfers. For example, on Input-Aware → BadNets, the AUC drops to 0.168 for $\mathrm{vec}(W)$ and 0.249 for $\mathrm{vec}(W, b)$. In contrast, our probing-based representation achieves an average cross-attack AUC of 0.992, while its worst evaluated cross-attack AUC remains 0.917. The complete source-to-target matrices are provided in Appendix D.1.

These results show that the benefit does not come merely from exposing the final-layer parameters to the meta-classifier. Instead, the probing procedure produces a more transferable representation of the classifier's decision behavior across heterogeneous attacks.

5.4 Sensitivity to Number of Training Models

We further investigate how many reference models are required to train an effective meta classifier. The experiment is conducted on CIFAR-10 with ResNet-18. We vary the number of models used for detector training from 8 to 360 while keeping the sampling strategy balanced across clean, class-agnostic, and class-specific models. For each training size, the experiment is repeated with three independent random seeds. We report the mean AUC on source and cross attacks, with error bars indicating the standard deviation.

As shown in Fig. 3, our method already achieves source and cross-attack AUCs of 0.861 and 0.806 with only 8 reference models. Detection performance improves as more training models are added,

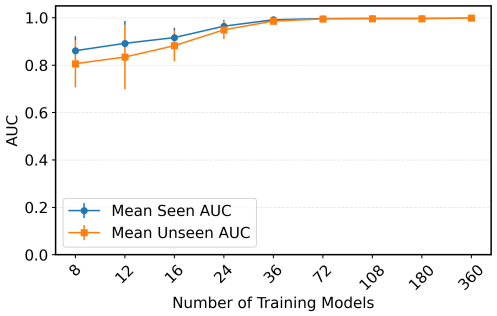


Figure 3: Sensitivity to the number of models used to train the meta classifier on CIFAR-10 with ResNet-18. We report the mean source and cross AUC over three independent runs, with error bars indicating the standard deviation.

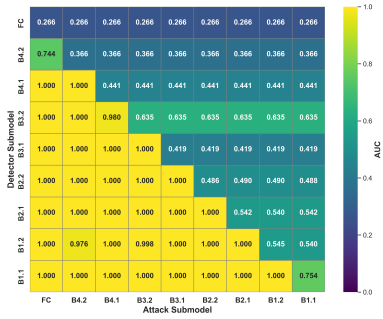


Figure 4: AUCs under the adaptive attack. Columns denote frozen attack submodels, and rows denote detector submodels. Detection fails near the diagonal but recovers when the detector observes a larger submodel than the one frozen by the attacker.

reaching 0.992 and 0.986, respectively, with 36 models. Beyond this point, performance begins to plateau, with additional models providing smaller gains. These results support the effectiveness of our method with a relatively small reference set under the evaluated setting.

5.5 Adaptive Attack

We consider an adaptive attacker that freezes the final classifier and progressively more network layers, and implants the backdoor only through the remaining earlier layers. We evaluate this setting using BadNets on CIFAR-10 with ResNet-18. Detailed configurations and attack effectiveness are provided in Appendix E.

When the detector probes only the layers frozen during backdoor injection, detection largely fails because these layers remain identical to those of the corresponding benign model. However, expanding the probing range to include layers updated during backdoor injection largely restores detection performance. As shown in Fig. 4, these configurations achieve an average AUC of 0.992, with 35 of 36 exceeding 0.95. Meanwhile, freezing more layers increasingly constrains backdoor injection. As reported in Appendix E, the attack success rate generally decreases as the frozen range expands, eventually dropping to only 8.78% when B1.1 is also frozen.

Overall, these results show that adaptive freezing can evade fixed-layer detection, but freezing a larger portion of the network also weakens backdoor injection. Expanding the probing range can recover detection performance, at the cost of additional computational overhead.

6 Conclusion

We found that strong performance of existing meta-classifier-based backdoor detectors on training attacks does not necessarily translate to emerging attacks. Guided by our analysis of model representations and class-specific prediction behavior, we proposed a method that optimizes an input feature vector for each class of the final classifier and combines these vectors into a model-level representation for meta-classification. Experiments across 12 attacks and six combinations of datasets and model architectures demonstrate improved cross-attack generalization under single-attack training. Our adaptive attack analysis further highlights the limitations of relying solely on the final classifier, motivating future exploration of probing locations that balance detection robustness and computational efficiency.

7 AI USE STATEMENT

In this work, we used generative AI tools to assist with translation and to improve the clarity, grammar, and readability of the manuscript. Generative AI tools were also used to help refine the presentation and organization of author-written text. All AI-assisted content was carefully reviewed and revised by the authors to ensure technical correctness, consistency with the conducted experiments, and faithful representation of the research findings. Generative AI tools were not used to generate experimental data or results. The authors take full responsibility for the final content of this work, including any text or artifacts produced with the aid of generative AI.

REFERENCES

Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 101–105. IEEE, 2019.

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.

Chong Fu, Xuhong Zhang, Shouling Ji, Ting Wang, Peng Lin, Yanghe Feng, and Jianwei Yin. {FreeEagle}: Detecting complex neural trojans in {Data-Free} cases. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 6399–6416, 2023.

Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th annual computer security applications conference*, pp. 113–125, 2019.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *Ieee Access*, 7:47230–47244, 2019.

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.

Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 298–307. IEEE, 2020.

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

Lauro Langosco, Neel Alex, William Baker, David Quarel, Herbie Bradley, and David Krueger. Detecting backdoors with meta-models. In *NeurIPS 2023 Workshop on Backdoors in Deep Learning-The Good, the Bad, and the Ugly*, 2023.

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, 2021a.

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, 2024. doi: 10.1109/TNNLS.2022.3182979.

Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16443–16452. IEEE, 2021b.

Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pp. 1265–1282, 2019.

Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European conference on computer vision*, pp. 182–199. Springer, 2020.

Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems*, 33:3454–3464, 2020.

Tuan Anh Nguyen and Anh Tuan Tran. WaNet - Imperceptible Warping-Based Backdoor Attack. In *International Conference on Learning Representations*, 2021.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 4510–4520. Ieee, 2018.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

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.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE symposium on security and privacy (SP)*, pp. 707–723. IEEE, 2019.

Zhenting Wang, Juan Zhai, and Shiqing Ma. Bppattack: Stealthy and efficient trojan attacks against deep neural networks via image quantization and contrastive adversarial learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15054–15063. IEEE, 2022.

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.

Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 103–120. IEEE, 2021.

Yi Zeng, Won Park, Z Morley Mao, and Ruoxi Jia. Rethinking the backdoor attacks' triggers: A frequency perspective. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16453–16461. IEEE, 2021.

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.

Quan Zhao, Linkang Du, Yuntao Wang, Zhou Su, Zheng Li, Xiangshan Gao, and Yang Zhang. Unveiling the pitfalls of Data-Free backdoor detection against Pre-Trained models. In *35th USENIX Security Symposium (USENIX Security 26)*, pp. 261–280, 2026.

A Overview of Appendix

There are additional materials presented in the Appendix:

- Appendix B: Experimental and implementation details.
  - Appendix B.1: Backdoored model generation.
  - Appendix B.2: Decision-Space Probing details and algorithm.
  - Appendix B.3: Meta-classifier details.
  - Appendix B.4: Baseline implementation details.
  - Appendix B.5: Randomness and repetition protocol.
  - Appendix B.6: Hardware and software environment.
- Appendix C: Additional experimental results.
  - Appendix C.1: Complete cross-attack detection results.
- Appendix D: Additional analyses.
  - Appendix D.1: Analysis of direct parameter representations.
  - Appendix D.2: Training-size sensitivity.
- Appendix E: Adaptive attack evaluation.
  - Appendix E.1: Adaptive attack setting.
  - Appendix E.2: Effectiveness of adaptive backdoored models.
  - Appendix E.3: Detection under adaptive attacks.

B Experimental and Implementation Details

This section provides additional implementation details for reproducing the experiments in the main paper, including the generation of backdoored models, Decision-Space Probing, meta-classifier training, baseline implementations, randomness control, and the hardware environment.

B.1 Backdoored Model Generation

We use the BackdoorBench codebase as the common framework for generating target models. Unless otherwise specified, models are trained for 60 epochs with a batch size of 128, an initial learning rate of 1×10^{-3}, and a weight decay of 1×10^{-4}. We mainly consider poisoning ratios of 5% and 10%.

For all-to-one attacks, we generate backdoored models using different target classes and random seeds to increase model diversity. When applicable, we additionally generate class-specific models with source class $s \neq t$. For clean-label attacks, including LC, SIG, and Narcissus, we follow the corresponding BackdoorBench implementations and use their attack-specific resources. BPP and WaNet use a CosineAnnealingLR scheduler, while the remaining attacks follow the default scheduler setting in our generation pipeline. Unless otherwise stated, all other attack-specific configurations follow BackdoorBench.

B.2 Decision-Space Probing Details

For each target model, we extract its final linear classifier and optimize class-wise latent representations in the input feature space of the classifier. The dimensionality of each representation is determined by the input dimension of the final classifier.

For a model with K output classes, we construct one latent representation z_k for each class $k \in \{1, \ldots, K\}$. All class-wise representations are initialized with the same all-one vector and optimized independently while keeping the classifier parameters fixed. Specifically, for each class k, we minimize

$$\min_{z_k} \mathcal{L}_{\mathrm{CE}}\big(h_\theta(z_k), k\big),$$

where $h_\theta(\cdot)$ denotes the final linear classifier. During probing, only z_k is updated, while all parameters of h_θ remain frozen.

Algorithm 1 Decision-Space Probing

Require: Trained target model F, number of classes K, optimization steps T, learning rate η
Ensure: Class-wise decision representations $\mathcal{Z}^{(T)} = \{z_k^{(T)}\}_{k=1}^{K}$

```
Extract the final linear classifier h_θ from F
Obtain the classifier input dimension d
Freeze all parameters of h_θ
for k = 1 to K do
    Initialize z_k^(0) ← 1_d
    Initialize Adam optimizer for z_k with learning rate η
    for t = 1 to T do
        Compute logits o ← h_θ(z_k^(t−1))
        Compute L ← CE(o, k)
        Update z_k^(t−1) with one Adam step to obtain z_k^(t)
    end for
end for
return Z^(T) = {z_1^(T), ..., z_K^(T)}
```

Unless otherwise specified, we use the same probing configuration across all datasets and model architectures. Each class-wise representation is optimized using Adam with a learning rate of 1×10^{-3} for a fixed budget of 20,000 optimization steps. No early stopping is used.

Algorithm 1 summarizes the complete Decision-Space Probing procedure.

B.3 Meta-Classifier Details

After Decision-Space Probing, the class-wise decision representations of each target model are arranged in a fixed class order and flattened into a one-dimensional representation. Each resulting representation is used as one input example for the meta-classifier, where benign and backdoored models are assigned labels 0 and 1, respectively.

We use a lightweight two-layer 1D-CNN as the meta-classifier. Its architecture is summarized in Table 3.

Table 3: Architecture of the 1D-CNN meta-classifier.

Layer	Configuration	Activation
Conv1D	$1 \to 16$, kernel $= 5$, stride $= 2$, padding $= 2$	ReLU
MaxPool1D	kernel $= 2$	–
Conv1D	$16 \to 32$, kernel $= 3$, stride $= 1$, padding $= 1$	ReLU
MaxPool1D	kernel $= 2$	–
Fully connected	hidden dimension $= 64$	ReLU
Dropout	$p = 0.5$	–
Fully connected	output dimension $= 1$	–

Before training, all representations are standardized feature-wise using a `StandardScaler`. For each feature dimension j, the mean μ_j and standard deviation σ_j are computed exclusively from the training set:

$$\tilde{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}.$$

The same training-set statistics are subsequently applied to all evaluation representations without refitting the scaler.

The meta-classifier is trained using binary cross-entropy with logits. We use Adam with a learning rate of 1×10^{-3}, a batch size of 16, and train the classifier for 20 epochs.

B.4 Baseline Implementation Details

We compare our method with MNTD and WeightMM. The baselines are implemented following their released implementations and the implementation details described in the corresponding papers. All methods are evaluated using the same pool of benign and backdoored target models and the same source-to-target cross-attack evaluation protocol. Unless otherwise specified, each baseline retains its original detector architecture and training procedure rather than being forced to use the meta-classifier employed by our method.

MNTD. MNTD represents each target model through its responses to a set of learned query inputs. We use $N_{\text{in}} = 10$ learned queries. The resulting query-response logits are concatenated into an $N_{\text{in}}K$-dimensional representation, where K denotes the number of output classes, and are fed into a two-layer MLP with a 20-dimensional hidden layer and LeakyReLU activation. The learned queries and meta-classifier are jointly optimized using Adam with a learning rate of 10^{-3} and binary cross-entropy with logits. At evaluation time, the trained detector is applied to the same target-model pool used in our cross-attack experiments, and the resulting model-level scores are used to compute ROC AUC.

WeightMM. WeightMM directly represents a target model through its parameters. Following the baseline implementation, floating-point model parameters are concatenated in a fixed parameter order and partitioned into 1,024-dimensional chunks, with zero-padding applied to the final chunk when necessary. BatchNorm running statistics are excluded. The resulting sequence of parameter chunks is processed by a Transformer-based meta-classifier. Each 1,024-dimensional chunk is projected to a 128-dimensional hidden representation and combined with learned positional embeddings before being processed by the Transformer encoder.

All baseline detectors are evaluated using the same benign and backdoored target-model pools as our method. Therefore, the methods differ in how they construct model-level representations and train their corresponding detectors, while the source-to-target cross-attack evaluation protocol remains unchanged.

B.5 Randomness and Repetition Protocol

Randomness is controlled separately during target-model generation and meta-classifier training. For target-model generation, different random seeds are used to control stochastic factors such as model initialization and data shuffling, producing independently trained target models.

Decision-Space Probing uses the same fixed all-one initialization and optimization budget for all target models and therefore introduces no additional random initialization.

For all main experiments, we repeat meta-classifier training and evaluation using three independent random seeds. Each run independently samples the training representations and initializes the meta-classifier, while the extracted model representations remain fixed. We report the mean and standard deviation over the three runs.

The training-size sensitivity experiment follows the same three-seed protocol. For each run, the standardization statistics are computed from the corresponding training subset and applied unchanged to the evaluation representations.

B.6 Hardware and Software Environment

All runtime measurements reported in the layer-ablation experiments were obtained on a single NVIDIA GeForce RTX 4090D GPU. The software environment consisted of PyTorch 1.11.0, CUDA 11.3, and cuDNN 8.2.0. Reported optimization times correspond to wall-clock runtime under this hardware and software configuration.

C Additional Experimental Results

C.1 Complete Cross-Attack Detection Results

We provide the complete source-to-target cross-attack detection results corresponding to the evaluations reported in the main paper. For each source attack, the detector is trained using benign models and backdoored models generated by that attack, and is subsequently evaluated on both the same attack and the remaining eleven attacks. Therefore, the diagonal entries in each table correspond to source attack performance, whereas the off-diagonal entries measure generalization to cross attacks.

We compare MNTD, WeightMM, and our method under the same source-to-target evaluation protocol. The Avg. column reports the average AUC over the eleven cross attacks, excluding the source attack.

Table 4: Cross-attack detection AUC on CIFAR-10 with CNN.

Model	Source	Method	BadNets	Blended	BPP	Filter	Input-Aware	LC	LF	Narcissus	Refool	SIG	SSBA	WaNet	Avg.
CNN	BadNets	MNTD	0.994	0.950	0.968	0.941	0.889	0.909	0.936	0.919	0.959	0.905	0.942	0.904	0.929
		WeightMM	0.987	0.982	0.915	0.985	0.967	0.970	0.987	0.975	0.982	0.981	0.983	0.911	0.967
		Ours	1.000	0.949	0.951	0.950	0.994	0.943	0.935	0.943	0.946	0.939	0.992	0.998	0.958
	Blended	MNTD	0.935	0.996	0.950	0.941	0.925	0.825	0.923	0.887	0.938	0.866	0.955	0.881	0.912
		WeightMM	0.967	0.981	0.915	0.975	0.935	0.966	0.959	0.964	0.960	0.977	0.971	0.885	0.952
		Ours	0.997	0.997	0.992	0.996	0.998	0.994	0.995	0.995	0.997	0.992	0.996	0.985	0.994
	BPP	MNTD	0.580	0.596	1.000	0.631	0.963	0.661	0.604	0.733	0.544	0.638	0.694	0.763	0.673
		WeightMM	0.659	0.739	0.994	0.656	1.000	0.472	0.592	0.669	0.629	0.545	0.777	0.982	0.702
		Ours	0.967	0.962	0.981	0.972	0.912	0.923	0.972	0.946	0.977	0.935	0.972	0.950	0.953
	Filter	MNTD	0.888	0.915	0.974	0.999	0.945	0.830	0.879	0.913	0.901	0.841	0.843	0.939	0.897
		WeightMM	0.981	0.979	0.962	0.988	0.917	0.979	0.979	0.982	0.981	0.980	0.986	0.912	0.967
		Ours	0.990	0.995	0.990	0.996	0.995	0.992	0.994	0.991	0.994	0.990	0.993	0.976	0.991
	Input-Aware	MNTD	0.454	0.482	0.895	0.433	1.000	0.346	0.397	0.317	0.447	0.299	0.563	0.715	0.486
		WeightMM	0.528	0.603	0.970	0.503	1.000	0.304	0.417	0.502	0.483	0.397	0.655	0.972	0.576
		Ours	0.998	0.995	0.992	0.993	1.000	0.994	0.995	0.994	0.993	0.994	0.997	1.000	0.995
	LC	MNTD	0.745	0.687	0.964	0.745	0.828	0.976	0.680	0.839	0.696	0.835	0.681	0.786	0.772
		WeightMM	0.831	0.829	0.563	0.838	0.533	0.986	0.885	0.965	0.830	0.972	0.792	0.505	0.777
		Ours	0.993	0.993	0.987	0.992	0.999	0.997	0.992	0.994	0.993	0.994	0.992	0.985	0.992
	LF	MNTD	0.902	0.922	0.967	0.909	0.792	0.833	0.976	0.928	0.900	0.875	0.935	0.864	0.893
		WeightMM	0.958	0.955	0.926	0.961	0.895	0.964	0.978	0.940	0.959	0.959	0.957	0.910	0.944
		Ours	0.970	0.995	0.992	0.994	0.994	0.988	0.996	0.989	0.995	0.991	0.990	0.955	0.987
	Narcissus	MNTD	0.665	0.702	0.956	0.814	0.834	0.747	0.746	0.962	0.671	0.741	0.713	0.677	0.751
		WeightMM	0.764	0.777	0.630	0.750	0.560	0.954	0.812	0.961	0.750	0.949	0.727	0.575	0.750
		Ours	0.987	0.989	0.982	0.989	0.998	0.992	0.989	0.996	0.989	0.991	0.988	0.980	0.989
	Refool	MNTD	0.950	0.947	0.886	0.941	0.879	0.831	0.930	0.879	0.992	0.863	0.937	0.890	0.903
		WeightMM	0.954	0.951	0.868	0.950	0.355	0.929	0.949	0.930	0.972	0.926	0.936	0.758	0.864
		Ours	0.973	0.990	0.987	0.989	0.991	0.986	0.991	0.986	0.989	0.978	0.985	0.960	0.983
	SIG	MNTD	0.749	0.721	0.962	0.782	0.628	0.852	0.745	0.882	0.746	0.952	0.736	0.758	0.778
		WeightMM	0.833	0.834	0.461	0.842	0.300	0.976	0.872	0.961	0.850	0.977	0.771	0.435	0.739
		Ours	0.994	0.994	0.992	0.994	0.999	0.997	0.994	0.996	0.994	0.998	0.994	0.994	0.995
	SSBA	MNTD	0.892	0.936	0.967	0.827	0.863	0.752	0.907	0.847	0.894	0.822	0.994	0.872	0.871
		WeightMM	0.989	0.990	0.929	0.983	0.988	0.982	0.984	0.973	0.988	0.982	0.991	0.937	0.975
		Ours	0.995	0.987	0.974	0.986	0.995	0.976	0.985	0.977	0.986	0.979	0.990	0.981	0.984
	WaNet	MNTD	0.667	0.705	0.971	0.804	0.967	0.677	0.687	0.672	0.679	0.633	0.709	0.999	0.743
		WeightMM	0.469	0.540	0.963	0.467	1.000	0.368	0.429	0.548	0.423	0.445	0.591	0.997	0.568
		Ours	0.998	0.965	0.963	0.961	1.000	0.972	0.967	0.958	0.947	0.965	0.992	1.000	0.972

D Additional Analyses

D.1 Direct Parameter Representations

To examine whether the detection performance of our method can be explained simply by exposing the parameters of the final classifier to the meta-classifier, we compare three model-level representations on CIFAR-10 with ResNet-18: the vectorized final-layer weights $\mathrm{vec}(W)$, the vectorized weights and biases $\mathrm{vec}(W, b)$, and our decision-space representation $\mathrm{vec}(\Phi(f_\theta))$.

All three representations are evaluated using the same meta-classifier configuration, training protocol, data splits, and source-to-target cross-attack evaluation procedure. This comparison therefore isolates the effect of the model-level representation while keeping the detector configuration unchanged.

Tables 10, 11, and 12 report the complete source-to-target cross-attack AUC matrices. Each row denotes the source attack used to train the meta-classifier, while each column denotes the attack

Table 5: Cross-attack detection AUC on CIFAR-10 with ResNet-18.

Model	Source	Method	BadNets	Blended	BPP	Filter	Input-Aware	LC	LF	Narcissus	Refool	SIG	SSBA	WaNet	Avg.
ResNet-18	BadNets	MNTD	0.857	0.491	0.085	0.578	0.015	0.610	0.518	0.801	0.669	0.692	0.483	0.502	0.495
		WeightMM	0.998	0.938	0.982	0.944	1.000	0.283	0.895	0.475	0.979	0.381	0.971	0.995	0.804
		Ours	1.000	0.998	0.998	0.998	1.000	0.988	0.998	0.981	0.998	0.985	0.998	0.998	0.995
	Blended	MNTD	0.620	0.573	0.139	0.524	0.093	0.436	0.507	0.520	0.657	0.439	0.520	0.469	0.448
		WeightMM	0.998	0.988	0.980	0.978	0.996	0.690	0.948	0.770	0.978	0.680	0.977	0.970	0.906
		Ours	1.000	0.998	0.999	0.998	1.000	0.993	0.998	0.993	0.998	0.992	0.998	0.998	0.997
	BPP	MNTD	0.447	0.479	0.526	0.483	0.505	0.519	0.453	0.528	0.513	0.544	0.473	0.608	0.505
		WeightMM	0.886	0.800	0.993	0.735	1.000	0.248	0.694	0.442	0.907	0.365	0.822	0.992	0.717
		Ours	0.999	0.993	0.998	0.995	1.000	0.990	0.996	0.989	0.996	0.987	0.995	0.994	0.994
	Filter	MNTD	0.678	0.474	0.172	0.593	0.060	0.505	0.504	0.610	0.625	0.497	0.470	0.550	0.468
		WeightMM	0.996	0.978	0.984	0.979	0.992	0.703	0.930	0.743	0.975	0.723	0.972	0.978	0.907
		Ours	0.998	0.995	0.998	0.996	1.000	0.990	0.995	0.990	0.996	0.989	0.995	0.994	0.995
	Input-Aware	MNTD	0.146	0.579	0.907	0.472	1.000	0.277	0.501	0.208	0.370	0.323	0.555	0.547	0.444
		WeightMM	0.126	0.469	1.000	0.399	1.000	0.842	0.494	0.779	0.151	0.708	0.227	0.001	0.472
		Ours	1.000	0.999	1.000	0.998	1.000	0.993	0.999	0.992	0.999	0.993	1.000	1.000	0.997
	LC	MNTD	0.548	0.529	0.537	0.505	0.510	0.573	0.506	0.496	0.537	0.507	0.540	0.581	0.527
		WeightMM	0.332	0.665	0.192	0.617	0.188	0.978	0.720	0.931	0.400	0.883	0.590	0.029	0.504
		Ours	0.991	0.987	0.995	0.987	1.000	0.993	0.987	0.989	0.987	0.988	0.987	0.990	0.990
	LF	MNTD	0.627	0.519	0.094	0.529	0.027	0.435	0.568	0.569	0.659	0.427	0.493	0.497	0.443
		WeightMM	0.994	0.968	0.970	0.972	0.951	0.690	0.962	0.754	0.965	0.715	0.962	0.958	0.900
		Ours	0.998	0.997	0.998	0.997	1.000	0.994	0.997	0.994	0.997	0.994	0.997	0.997	0.997
	Narcissus	MNTD	0.720	0.512	0.297	0.542	0.060	0.640	0.577	0.728	0.675	0.584	0.521	0.578	0.519
		WeightMM	0.433	0.806	0.386	0.770	0.489	0.963	0.813	0.967	0.613	0.943	0.748	0.128	0.645
		Ours	0.994	0.990	0.998	0.990	1.000	0.990	0.990	0.994	0.990	0.990	0.990	0.994	0.992
	Refool	MNTD	0.582	0.510	0.116	0.526	0.100	0.514	0.497	0.385	0.642	0.330	0.516	0.490	0.415
		WeightMM	0.997	0.972	0.990	0.965	1.000	0.452	0.937	0.613	0.992	0.480	0.982	0.995	0.853
		Ours	1.000	0.997	0.998	0.997	1.000	0.992	0.997	0.991	0.998	0.990	0.998	0.998	0.996
	SIG	MNTD	0.650	0.511	0.206	0.490	0.135	0.583	0.512	0.631	0.654	0.540	0.528	0.559	0.496
		WeightMM	0.409	0.742	0.407	0.720	0.390	0.927	0.765	0.933	0.578	0.960	0.698	0.165	0.612
		Ours	0.994	0.993	0.994	0.993	1.000	0.993	0.993	0.994	0.993	0.996	0.993	0.995	0.994
	SSBA	MNTD	0.679	0.475	0.197	0.543	0.091	0.533	0.483	0.600	0.614	0.549	0.568	0.578	0.486
		WeightMM	0.999	0.981	0.986	0.975	1.000	0.532	0.945	0.663	0.989	0.543	0.990	0.991	0.873
		Ours	1.000	0.997	0.999	0.997	1.000	0.992	0.997	0.991	0.998	0.990	0.998	0.999	0.996
	WaNet	MNTD	0.446	0.521	0.521	0.489	0.533	0.492	0.489	0.513	0.541	0.488	0.518	0.637	0.505
		WeightMM	0.848	0.672	0.898	0.629	0.150	0.139	0.582	0.296	0.878	0.264	0.777	1.000	0.558
		Ours	1.000	0.999	1.000	0.999	1.000	0.995	0.999	0.991	0.998	0.992	0.999	1.000	0.997

Table 6: Cross-attack detection AUC on CIFAR-10 with MobileNet.

Model	Source	Method	BadNets	Blended	BPP	Filter	Input-Aware	LC	LF	Narcissus	Refool	SIG	SSBA	WaNet	Avg.
MobileNet	BadNets	MNTD	0.501	0.514	0.641	0.410	0.663	0.442	0.515	0.465	0.476	0.462	0.567	0.502	0.514
		WeightMM	0.973	0.954	0.953	0.944	0.911	0.865	0.835	0.877	0.957	0.787	0.956	0.933	0.907
		Ours	0.997	0.977	0.988	0.956	1.000	0.887	0.981	0.827	0.978	0.905	0.993	1.000	0.954
	Blended	MNTD	0.513	0.532	0.575	0.389	0.674	0.453	0.508	0.444	0.514	0.504	0.605	0.521	0.518
		WeightMM	0.945	0.944	0.950	0.942	0.754	0.856	0.865	0.864	0.941	0.820	0.945	0.904	0.890
		Ours	0.987	0.981	0.986	0.949	1.000	0.887	0.965	0.844	0.955	0.918	0.978	1.000	0.952
	BPP	MNTD	0.469	0.501	0.742	0.334	0.672	0.441	0.502	0.472	0.443	0.488	0.551	0.547	0.493
		WeightMM	0.957	0.956	0.987	0.954	0.951	0.878	0.859	0.904	0.951	0.846	0.942	0.969	0.924
		Ours	0.990	0.979	0.998	0.964	1.000	0.908	0.976	0.878	0.962	0.908	0.984	1.000	0.959
	Filter	MNTD	0.491	0.485	0.647	0.420	0.759	0.447	0.502	0.486	0.456	0.485	0.528	0.515	0.527
		WeightMM	0.952	0.941	0.941	0.949	0.765	0.859	0.872	0.864	0.932	0.800	0.943	0.883	0.887
		Ours	0.994	0.982	0.992	0.975	1.000	0.907	0.985	0.872	0.970	0.933	0.989	1.000	0.966
	Input-Aware	MNTD	0.459	0.519	0.759	0.338	0.572	0.395	0.506	0.452	0.419	0.464	0.559	0.481	0.486
		WeightMM	0.519	0.546	0.903	0.520	1.000	0.514	0.594	0.534	0.522	0.549	0.511	0.809	0.593
		Ours	0.939	0.877	0.965	0.819	1.000	0.857	0.888	0.754	0.803	0.867	0.894	1.000	0.878
	LC	MNTD	0.489	0.481	0.561	0.407	0.720	0.481	0.479	0.467	0.508	0.487	0.547	0.512	0.514
		WeightMM	0.888	0.897	0.901	0.891	0.609	0.870	0.802	0.855	0.904	0.820	0.888	0.873	0.848
		Ours	0.989	0.977	0.986	0.980	1.000	0.987	0.994	0.952	0.969	0.976	0.988	1.000	0.983
	LF	MNTD	0.493	0.510	0.773	0.397	0.794	0.476	0.531	0.490	0.447	0.511	0.547	0.550	0.544
		WeightMM	0.911	0.913	0.931	0.909	0.876	0.856	0.947	0.879	0.912	0.891	0.903	0.864	0.895
		Ours	0.984	0.931	0.895	0.931	0.670	0.902	0.989	0.845	0.901	0.912	0.944	0.995	0.901
	Narcissus	MNTD	0.487	0.486	0.552	0.396	0.702	0.462	0.478	0.487	0.488	0.482	0.534	0.500	0.506
		WeightMM	0.914	0.905	0.886	0.905	0.702	0.884	0.824	0.926	0.920	0.832	0.913	0.894	0.871
		Ours	0.959	0.969	0.919	0.961	0.965	0.944	0.975	0.979	0.953	0.931	0.946	0.959	0.953
	Refool	MNTD	0.502	0.500	0.644	0.413	0.695	0.460	0.508	0.491	0.513	0.481	0.561	0.504	0.524
		WeightMM	0.970	0.972	0.972	0.963	0.876	0.905	0.849	0.923	0.984	0.826	0.972	0.951	0.925
		Ours	0.997	0.988	0.985	0.970	1.000	0.904	0.979	0.871	0.981	0.922	0.991	1.000	0.964
	SIG	MNTD	0.485	0.476	0.619	0.358	0.632	0.457	0.482	0.460	0.486	0.512	0.546	0.482	0.499
		WeightMM	0.831	0.848	0.840	0.849	0.703	0.831	0.879	0.860	0.844	0.920	0.846	0.814	0.831
		Ours	0.954	0.923	0.933	0.918	1.000	0.902	0.963	0.897	0.901	0.970	0.929	0.996	0.938
	SSBA	MNTD	0.474	0.470	0.571	0.371	0.678	0.448	0.474	0.470	0.464	0.480	0.532	0.482	0.489
		WeightMM	0.961	0.946	0.958	0.955	0.715	0.867	0.880	0.898	0.953	0.839	0.966	0.914	0.899
		Ours	0.995	0.984	0.993	0.975	1.000	0.921	0.985	0.865	0.974	0.907	0.994	1.000	0.964
	WaNet	MNTD	0.475	0.477	0.616	0.371	0.679	0.449	0.475	0.472	0.468	0.476	0.529	0.513	0.499
		WeightMM	0.907	0.904	0.947	0.901	0.961	0.829	0.719	0.833	0.917	0.744	0.893	0.972	0.869
		Ours	0.945	0.874	0.975	0.798	1.000	0.812	0.861	0.687	0.804	0.808	0.888	1.000	0.859

used for evaluation. Diagonal entries correspond to source attacks, whereas off-diagonal entries measure cross-attack generalization.

Table 7: Cross-attack detection AUC on GTSRB10 with ResNet-18.

Model	Source	Method	BadNets	Blended	BPP	Filter	Input-Aware	LC	LF	Narcissus	Refool	SIG	SSBA	WaNet	Avg.
ResNet-18	BadNets	MNTD	0.657	0.621	0.735	0.646	0.585	0.722	0.694	0.674	0.600	0.684	0.717	0.683	0.669
		WeightMM	0.995	0.993	0.996	0.993	0.991	0.989	0.992	0.986	0.990	0.985	0.993	0.983	0.990
		Ours	0.993	0.993	0.992	0.993	0.983	0.989	0.992	0.989	0.992	0.989	0.992	0.989	0.991
	Blended	MNTD	0.618	0.713	0.606	0.689	0.567	0.691	0.710	0.685	0.674	0.686	0.660	0.595	0.653
		WeightMM	0.993	0.994	0.995	0.994	0.992	0.990	0.994	0.990	0.993	0.990	0.994	0.988	0.992
		Ours	0.989	0.993	0.989	0.993	0.985	0.991	0.991	0.990	0.990	0.990	0.989	0.988	0.990
	BPP	MNTD	0.652	0.569	0.853	0.522	0.631	0.780	0.688	0.642	0.545	0.601	0.681	0.543	0.623
		WeightMM	0.814	0.779	0.998	0.716	1.000	0.767	0.883	0.666	0.580	0.650	0.934	0.874	0.788
		Ours	0.992	0.993	0.994	0.993	0.984	0.991	0.993	0.991	0.992	0.991	0.992	0.989	0.991
	Filter	MNTD	0.656	0.680	0.652	0.698	0.635	0.714	0.688	0.683	0.661	0.686	0.669	0.603	0.666
		WeightMM	0.994	0.994	0.994	0.994	0.991	0.991	0.994	0.991	0.994	0.991	0.993	0.991	0.992
		Ours	0.988	0.991	0.989	0.993	0.981	0.987	0.991	0.984	0.989	0.984	0.990	0.982	0.987
	Input-Aware	MNTD	0.624	0.614	0.697	0.557	0.763	0.758	0.653	0.681	0.628	0.611	0.654	0.664	0.649
		WeightMM	0.777	0.748	0.995	0.676	1.000	0.742	0.848	0.654	0.497	0.634	0.877	0.766	0.747
		Ours	0.989	0.991	0.988	0.992	1.000	0.990	0.992	0.989	0.991	0.989	0.994	0.984	0.990
	LC	MNTD	0.491	0.575	0.475	0.567	0.371	0.419	0.549	0.508	0.546	0.563	0.448	0.550	0.513
		WeightMM	0.989	0.986	0.998	0.984	1.000	0.990	0.988	0.988	0.971	0.986	0.989	0.980	0.987
		Ours	0.984	0.985	0.984	0.986	0.984	0.988	0.985	0.985	0.984	0.986	0.985	0.984	0.985
	LF	MNTD	0.565	0.638	0.591	0.657	0.537	0.633	0.698	0.596	0.581	0.670	0.620	0.584	0.607
		WeightMM	0.994	0.993	0.995	0.993	0.993	0.990	0.995	0.990	0.993	0.990	0.994	0.990	0.992
		Ours	0.991	0.993	0.992	0.993	0.984	0.990	0.993	0.988	0.992	0.989	0.991	0.987	0.990
	Narcissus	MNTD	0.491	0.639	0.370	0.618	0.492	0.464	0.544	0.613	0.635	0.617	0.469	0.585	0.539
		WeightMM	0.986	0.985	0.995	0.983	0.996	0.987	0.986	0.987	0.973	0.983	0.987	0.973	0.985
		Ours	0.987	0.991	0.988	0.993	0.981	0.989	0.991	0.998	0.988	0.998	0.985	0.982	0.988
	Refool	MNTD	0.596	0.658	0.563	0.674	0.583	0.691	0.687	0.682	0.661	0.679	0.671	0.648	0.648
		WeightMM	0.995	0.995	0.993	0.995	0.990	0.991	0.995	0.991	0.995	0.991	0.994	0.990	0.993
		Ours	0.991	0.992	0.991	0.992	0.983	0.988	0.992	0.987	0.992	0.988	0.991	0.986	0.989
	SIG	MNTD	0.530	0.676	0.423	0.627	0.528	0.507	0.583	0.609	0.649	0.651	0.486	0.515	0.557
		WeightMM	0.988	0.987	0.992	0.987	0.989	0.989	0.987	0.989	0.986	0.990	0.987	0.987	0.988
		Ours	0.992	0.992	0.992	0.993	0.992	0.993	0.993	0.995	0.992	0.996	0.993	0.992	0.993
	SSBA	MNTD	0.663	0.616	0.802	0.623	0.643	0.808	0.745	0.702	0.612	0.666	0.774	0.601	0.680
		WeightMM	0.993	0.992	0.999	0.987	0.983	0.991	0.996	0.988	0.979	0.986	0.998	0.995	0.990
		Ours	0.990	0.992	0.991	0.992	0.982	0.989	0.992	0.988	0.991	0.988	0.992	0.984	0.989
	WaNet	MNTD	0.613	0.645	0.692	0.614	0.661	0.717	0.673	0.666	0.604	0.650	0.627	0.709	0.651
		WeightMM	0.983	0.911	1.000	0.879	0.997	0.971	0.961	0.922	0.873	0.911	0.995	0.999	0.946
		Ours	0.991	0.993	0.993	0.993	0.984	0.990	0.993	0.989	0.992	0.990	0.989	0.994	0.991

Table 8: Cross-attack detection AUC on GTSRB with ResNet-18.

Model	Source	Method	BadNets	Blended	BPP	Filter	Input-Aware	LC	LF	Narcissus	Refool	SIG	SSBA	WaNet	Avg.
ResNet-18	BadNets	MNTD	0.523	0.531	0.444	0.510	0.357	0.543	0.566	0.536	0.650	0.537	0.485	0.464	0.511
		WeightMM	1.000	0.868	0.970	0.937	0.000	0.847	0.845	0.758	0.993	0.752	0.845	1.000	0.801
		Ours	0.999	0.996	0.997	0.997	0.999	0.995	0.996	0.995	0.997	0.995	0.998	0.998	0.997
	Blended	MNTD	0.564	0.552	0.472	0.542	0.387	0.534	0.598	0.573	0.581	0.564	0.556	0.598	0.543
		WeightMM	0.981	0.996	0.855	0.991	0.944	0.996	0.994	0.996	0.995	0.996	0.994	0.878	0.965
		Ours	0.997	0.996	0.996	0.996	1.000	0.996	0.996	0.996	0.996	0.996	0.996	0.998	0.997
	BPP	MNTD	0.454	0.529	0.468	0.381	0.468	0.434	0.410	0.463	0.278	0.479	0.477	0.509	0.444
		WeightMM	0.914	0.719	1.000	0.767	1.000	0.721	0.626	0.655	0.769	0.630	0.716	1.000	0.774
		Ours	0.998	0.996	1.000	0.997	1.000	0.996	0.996	0.996	0.998	0.996	0.997	1.000	0.997
	Filter	MNTD	0.644	0.558	0.664	0.610	0.651	0.608	0.667	0.540	0.665	0.530	0.691	0.544	0.615
		WeightMM	0.997	0.996	0.927	0.998	0.992	0.998	0.997	0.998	0.997	0.998	0.997	0.953	0.986
		Ours	0.997	0.996	0.997	0.997	1.000	0.997	0.996	0.997	0.997	0.997	0.997	0.998	0.997
	Input-Aware	MNTD	0.537	0.508	0.491	0.436	0.442	0.409	0.374	0.429	0.372	0.469	0.505	0.600	0.466
		WeightMM	0.748	0.496	0.996	0.502	1.000	0.586	0.400	0.560	0.411	0.519	0.486	1.000	0.609
		Ours	0.998	0.997	1.000	0.996	1.000	0.998	0.996	0.998	0.998	0.999	0.998	1.000	0.998
	LC	MNTD	0.651	0.615	0.699	0.495	0.821	0.552	0.553	0.546	0.617	0.565	0.593	0.683	0.622
		WeightMM	0.997	0.973	0.939	0.977	0.958	0.996	0.966	0.995	0.982	0.994	0.976	0.981	0.976
		Ours	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	LF	MNTD	0.551	0.573	0.634	0.444	0.558	0.501	0.541	0.526	0.458	0.564	0.634	0.602	0.550
		WeightMM	0.962	0.994	0.778	0.993	0.963	0.992	0.998	0.993	0.982	0.997	0.993	0.718	0.942
		Ours	0.999	0.999	0.998	0.999	1.000	0.999	0.999	0.998	0.999	0.998	0.999	0.999	0.999
	Narcissus	MNTD	0.650	0.594	0.666	0.555	0.783	0.610	0.611	0.578	0.639	0.535	0.627	0.616	0.626
		WeightMM	0.705	0.970	0.478	0.960	0.458	0.991	0.974	0.994	0.931	0.993	0.973	0.573	0.819
		Ours	0.996	0.996	0.996	0.996	1.000	0.996	0.996	0.996	0.996	0.996	0.996	0.996	0.996
	Refool	MNTD	0.607	0.572	0.592	0.608	0.600	0.653	0.645	0.588	0.792	0.568	0.674	0.555	0.605
		WeightMM	0.998	0.992	0.945	0.990	0.941	0.995	0.993	0.994	0.998	0.995	0.992	0.965	0.982
		Ours	0.998	0.996	0.997	0.996	1.000	0.995	0.995	0.995	0.998	0.995	0.997	0.999	0.997
	SIG	MNTD	0.462	0.513	0.525	0.480	0.569	0.542	0.564	0.521	0.395	0.522	0.484	0.549	0.510
		WeightMM	0.654	0.974	0.393	0.960	0.073	0.992	0.974	0.994	0.932	0.994	0.973	0.385	0.755
		Ours	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	SSBA	MNTD	0.664	0.582	0.677	0.555	0.690	0.608	0.660	0.622	0.676	0.614	0.590	0.458	0.619
		WeightMM	0.992	0.998	0.898	0.997	0.996	0.998	0.995	0.998	0.995	0.998	0.998	0.924	0.981
		Ours	1.000	0.999	1.000	0.999	1.000	0.999	0.999	0.998	1.000	0.998	1.000	1.000	0.999
	WaNet	MNTD	0.470	0.489	0.429	0.444	0.347	0.446	0.411	0.470	0.327	0.472	0.512	0.768	0.438
		WeightMM	0.820	0.561	0.984	0.627	1.000	0.605	0.467	0.513	0.664	0.485	0.551	1.000	0.661
		Ours	0.997	0.996	0.993	0.997	1.000	0.996	0.995	0.995	0.997	0.995	0.997	1.000	0.996

Although the direct parameter representations achieve high AUC for many attack pairs, they exhibit substantial degradation for particular cross-attack transfers. For example, when trained on Input-Aware, $\mathrm{vec}(W)$ and $\mathrm{vec}(W, b)$ achieve AUCs of only 0.168 and 0.249 on BadNet, respectively. In contrast, the minimum cross-attack AUC of our decision-space representation remains 0.917,

Table 9: Cross-attack detection AUC on CIFAR-100 with ResNet-34.

Model	Source	Method	BadNets	Blended	BPP	Filter	Input-Aware	LC	LF	Narcissus	Refool	SIG	SSBA	WaNet	Avg.
ResNet-34	BadNets	MNTD	0.830	0.833	0.717	0.933	0.932	0.546	0.831	0.834	0.723	0.378	0.717	0.978	0.765
		WeightMM	0.890	0.945	0.995	0.944	1.000	0.697	0.934	0.528	0.831	0.519	0.824	0.998	0.838
		Ours	0.999	0.987	1.000	0.981	1.000	0.545	0.962	0.974	0.977	0.973	0.994	1.000	0.945
	Blended	MNTD	0.785	0.824	0.726	0.896	0.907	0.530	0.802	0.771	0.710	0.406	0.680	0.876	0.735
		WeightMM	0.786	0.868	0.973	0.865	1.000	0.619	0.878	0.612	0.743	0.518	0.711	0.981	0.790
		Ours	0.945	0.931	0.958	0.919	0.720	0.687	0.909	0.699	0.890	0.689	0.946	0.886	0.841
	BPP	MNTD	0.778	0.788	0.996	0.900	0.948	0.559	0.799	0.848	0.665	0.404	0.676	0.985	0.759
		WeightMM	0.699	0.763	1.000	0.766	1.000	0.611	0.795	0.994	0.628	0.991	0.619	1.000	0.806
		Ours	0.999	0.985	1.000	0.976	1.000	0.629	0.946	0.986	0.963	0.989	0.993	1.000	0.951
	Filter	MNTD	0.785	0.768	0.812	0.877	0.910	0.566	0.783	0.719	0.706	0.383	0.698	0.860	0.726
		WeightMM	0.877	0.971	0.999	0.993	1.000	0.690	0.958	0.600	0.806	0.679	0.826	0.998	0.855
		Ours	0.976	0.949	0.976	0.931	0.756	0.623	0.934	0.618	0.925	0.628	0.961	0.973	0.847
	Input-Aware	MNTD	0.743	0.744	0.788	0.853	0.970	0.592	0.747	0.781	0.649	0.410	0.620	0.840	0.706
		WeightMM	0.735	0.809	1.000	0.815	1.000	0.663	0.758	0.991	0.639	0.992	0.663	1.000	0.824
		Ours	0.864	0.745	1.000	0.716	1.000	0.508	0.658	1.000	0.699	1.000	0.842	1.000	0.821
	LC	MNTD	0.471	0.437	0.381	0.380	0.422	0.595	0.434	0.406	0.474	0.562	0.502	0.255	0.429
		WeightMM	0.746	0.818	0.003	0.817	0.000	0.614	0.799	0.015	0.641	0.022	0.682	0.004	0.413
		Ours	0.837	0.799	0.829	0.814	0.714	0.873	0.805	0.832	0.796	0.812	0.795	0.783	0.802
	LF	MNTD	0.752	0.756	0.906	0.844	0.861	0.519	0.782	0.714	0.685	0.444	0.691	0.892	0.733
		WeightMM	0.660	0.717	0.850	0.714	1.000	0.577	0.794	0.524	0.603	0.432	0.581	0.840	0.682
		Ours	0.978	0.914	0.988	0.917	0.926	0.677	0.908	0.711	0.906	0.718	0.941	0.968	0.877
	Narcissus	MNTD	0.480	0.438	0.309	0.403	0.362	0.558	0.452	0.525	0.512	0.630	0.486	0.276	0.446
		WeightMM	0.734	0.789	1.000	0.801	1.000	0.622	0.817	1.000	0.647	1.000	0.663	1.000	0.825
		Ours	0.981	0.893	1.000	0.863	1.000	0.745	0.833	1.000	0.898	1.000	0.974	1.000	0.926
	Refool	MNTD	0.825	0.853	0.865	0.940	0.946	0.557	0.832	0.834	0.745	0.377	0.718	0.971	0.792
		WeightMM	0.903	0.971	0.372	0.965	0.000	0.678	0.972	0.265	0.863	0.265	0.846	0.197	0.585
		Ours	0.990	0.966	0.998	0.961	0.787	0.643	0.946	0.611	0.951	0.626	0.985	0.978	0.863
	SIG	MNTD	0.437	0.421	0.515	0.387	0.570	0.552	0.446	0.409	0.523	0.605	0.470	0.295	0.457
		WeightMM	0.713	0.769	1.000	0.772	1.000	0.608	0.803	0.997	0.622	0.997	0.637	1.000	0.811
		Ours	0.964	0.895	1.000	0.882	1.000	0.798	0.840	1.000	0.890	1.000	0.960	1.000	0.930
	SSBA	MNTD	0.811	0.847	0.715	0.933	0.933	0.544	0.814	0.819	0.740	0.404	0.726	0.952	0.774
		WeightMM	0.904	0.969	0.703	0.976	0.532	0.683	0.969	0.343	0.836	0.332	0.873	0.583	0.712
		Ours	0.993	0.960	1.000	0.955	1.000	0.604	0.942	0.922	0.947	0.935	0.993	1.000	0.933
	WaNet	MNTD	0.703	0.741	0.880	0.852	0.920	0.500	0.726	0.773	0.662	0.432	0.629	0.986	0.711
		WeightMM	0.740	0.809	0.998	0.806	1.000	0.632	0.782	0.578	0.639	0.618	0.674	0.999	0.752
		Ours	0.992	0.962	1.000	0.951	1.000	0.464	0.877	1.000	0.980	1.000	0.996	1.000	0.929

indicating substantially more stable cross-attack generalization than directly using the final-layer parameters.

Table 10: Full cross-attack AUC matrix for the direct weight representation $\mathrm{vec}(W)$ on CIFAR-10 with ResNet-18. Rows denote source attacks used for meta-classifier training, and columns denote evaluation attacks. IA and Narc. denote Input-Aware and Narcissus, respectively.

Source \ Eval.	BadNet	Blended	BPP	Filter	IA	LC	LF	Narc.	Refool	SIG	SSBA	WaNet
BadNet	0.998	0.994	0.986	0.995	0.981	0.991	0.995	0.990	0.995	0.991	0.995	0.988
Blended	0.990	0.996	0.988	0.995	0.987	0.989	0.993	0.989	0.992	0.989	0.994	0.987
BPP	0.969	0.994	0.999	0.993	1.000	0.987	0.994	0.988	0.993	0.987	0.994	0.992
Filter	0.993	0.994	0.985	0.995	0.981	0.988	0.994	0.988	0.993	0.987	0.994	0.984
IA	0.168	0.917	1.000	0.842	1.000	0.694	0.865	0.831	0.788	0.605	0.900	0.998
LC	0.984	0.984	0.984	0.985	0.984	0.993	0.985	0.989	0.984	0.990	0.984	0.984
LF	0.996	0.996	0.992	0.996	0.992	0.994	0.997	0.993	0.996	0.993	0.996	0.993
Narc.	0.984	0.984	0.984	0.984	0.984	0.988	0.985	0.991	0.984	0.988	0.984	0.984
Refool	0.993	0.995	0.992	0.995	0.984	0.988	0.995	0.989	0.996	0.988	0.995	0.987
SIG	0.993	0.992	0.992	0.992	0.992	0.994	0.992	0.994	0.992	0.996	0.992	0.992
SSBA	0.996	0.995	0.985	0.995	0.981	0.990	0.995	0.990	0.995	0.989	0.996	0.987
WaNet	0.991	0.993	0.987	0.992	0.984	0.989	0.993	0.989	0.993	0.988	0.993	0.996

Table 11: Full cross-attack AUC matrix for the direct weight-and-bias representation $\mathrm{vec}(W, b)$ on CIFAR-10 with ResNet-18. Rows denote source attacks used for meta-classifier training, and columns denote evaluation attacks. IA and Narc. denote Input-Aware and Narcissus, respectively.

Source \ Eval.	BadNet	Blended	BPP	Filter	IA	LC	LF	Narc.	Refool	SIG	SSBA	WaNet
BadNet	0.997	0.994	0.988	0.994	0.981	0.989	0.994	0.990	0.995	0.989	0.994	0.987
Blended	0.993	0.997	0.990	0.996	0.988	0.991	0.996	0.991	0.995	0.990	0.996	0.988
BPP	0.965	0.993	0.999	0.993	1.000	0.987	0.994	0.988	0.992	0.987	0.994	0.992
Filter	0.992	0.995	0.983	0.996	0.981	0.986	0.994	0.987	0.993	0.986	0.995	0.983
IA	0.249	0.960	1.000	0.898	1.000	0.725	0.917	0.864	0.843	0.681	0.944	0.997
LC	0.986	0.988	0.984	0.988	0.984	0.993	0.988	0.990	0.987	0.990	0.987	0.985
LF	0.996	0.996	0.994	0.996	0.992	0.993	0.997	0.993	0.996	0.993	0.996	0.993
Narc.	0.985	0.985	0.984	0.985	0.984	0.989	0.985	0.991	0.985	0.988	0.985	0.984
Refool	0.993	0.994	0.991	0.995	0.984	0.989	0.995	0.989	0.996	0.988	0.995	0.987
SIG	0.993	0.992	0.992	0.992	0.992	0.994	0.992	0.994	0.992	0.995	0.992	0.992
SSBA	0.996	0.995	0.988	0.995	0.981	0.988	0.995	0.989	0.995	0.988	0.996	0.986
WaNet	0.991	0.993	0.993	0.993	0.990	0.989	0.993	0.989	0.993	0.988	0.994	0.997

Table 12: Full cross-attack AUC matrix for our decision-space representation $\mathrm{vec}(\Phi(f_\theta))$ on CIFAR-10 with ResNet-18. Rows denote source attacks used for meta-classifier training, and columns denote evaluation attacks. IA and Narc. denote Input-Aware and Narcissus, respectively.

Source \ Eval.	BadNet	Blended	BPP	Filter	IA	LC	LF	Narc.	Refool	SIG	SSBA	WaNet
BadNet	1.000	0.998	0.998	0.997	1.000	0.983	0.998	0.965	0.998	0.974	0.998	0.998
Blended	1.000	0.999	0.999	0.998	1.000	0.993	0.998	0.993	0.999	0.991	0.999	0.999
BPP	1.000	0.997	0.999	0.997	1.000	0.988	0.997	0.986	0.998	0.983	0.998	0.998
Filter	0.999	0.996	0.998	0.997	1.000	0.990	0.997	0.990	0.997	0.988	0.997	0.997
IA	0.999	0.987	1.000	0.989	1.000	0.952	0.992	0.917	0.989	0.917	0.994	1.000
LC	0.995	0.991	0.998	0.989	1.000	0.993	0.990	0.991	0.991	0.989	0.990	0.994
LF	1.000	0.998	0.998	0.998	1.000	0.995	0.998	0.995	0.998	0.994	0.998	0.998
Narc.	0.989	0.988	0.994	0.988	1.000	0.990	0.988	0.995	0.987	0.990	0.987	0.988
Refool	1.000	0.997	0.998	0.997	1.000	0.991	0.998	0.990	0.998	0.988	0.998	0.998
SIG	0.993	0.992	0.994	0.992	1.000	0.993	0.992	0.994	0.992	0.997	0.992	0.993
SSBA	1.000	0.997	0.998	0.996	1.000	0.991	0.997	0.989	0.998	0.988	0.997	0.998
WaNet	1.000	0.995	1.000	0.994	1.000	0.972	0.996	0.953	0.995	0.948	0.997	1.000

D.2 Training-Size Sensitivity

Table 13 reports the complete numerical results corresponding to the training-size sensitivity experiment in the main paper. We use Blended as the source attack for training the meta-classifier and evaluate both the source attack AUC and the mean cross-attack AUC over the remaining eleven cross attacks.

The training size denotes the total number of reference models used to train the meta-classifier. For each training size, reference models are sampled using the same balanced sampling strategy as in the main experiments. Each experiment is repeated using three independent random seeds following the protocol described in Appendix B.5. We report the mean and standard deviation across the three runs.

Table 13: Training-size sensitivity on CIFAR-10 with ResNet-18. Results are reported as mean ± standard deviation over three independent runs.

Training models	Source AUC	Mean cross AUC
8	0.861 ± 0.062	0.806 ± 0.100
12	0.892 ± 0.094	0.834 ± 0.136
16	0.916 ± 0.042	0.882 ± 0.065
24	0.964 ± 0.027	0.949 ± 0.039
36	0.992 ± 0.006	0.986 ± 0.012
72	0.996 ± 0.001	0.995 ± 0.001
108	0.997 ± 0.000	0.996 ± 0.000
180	0.997 ± 0.001	0.996 ± 0.000
360	0.999 ± 0.001	0.999 ± 0.001

E Adaptive Attack Evaluation

We further evaluate the robustness of our method under an adaptive attack in which the attacker explicitly attempts to preserve the classifier-side decision structure exploited by our detector. The adaptive setting is constructed on CIFAR-10 with ResNet-18 and is used only for evaluation; adaptive models are never included in the training set of the detector.

E.1 Adaptive Attack Setting

We evaluate an adaptive BadNet setting on CIFAR-10 with ResNet-18. The adaptive attack uses an all-to-one target label $t = 0$ with a poisoning ratio of 10%. Adaptive models are trained for 60 epochs using Adam with a learning rate of 10^{-3}, a weight decay of 10^{-4}, and a batch size of 128. No learning-rate scheduler is used.

Each adaptive model is initialized from a clean checkpoint. During adaptive training, we freeze a classifier-side suffix of ResNet-18 and update only the preceding network layers on the poisoned BadNet training set. The frozen suffix is kept in evaluation mode throughout training, and we verify after training that its parameters and state remain unchanged.

For each frozen suffix, we generate five adaptive models from different clean checkpoints while keeping the adaptive training seed fixed to 0. The adaptive AUC evaluation uses these adaptive models as positive samples and 256 clean models as negative samples.

Importantly, the adaptive models are not used to train the detector. For each detection submodel, the corresponding meta-classifier is trained only on the standard reference models following the same protocol as in the main experiments. The adaptive models are used exclusively for evaluation.

Table 14: ResNet-18 suffix definitions used for adaptive training and detection.

ID	Label	Frozen suffix modules
1	FC	`linear`
2	B4.2	`layer4.1-linear`
3	B4.1	`layer4.0-linear`
4	B3.2	`layer3.1-linear`
5	B3.1	`layer3.0-linear`
6	B2.2	`layer2.1-linear`
7	B2.1	`layer2.0-linear`
8	B1.2	`layer1.1-linear`
9	B1.1	`layer1.0-linear`
10	Full	Full network

E.2 Effectiveness of Adaptive Backdoored Models

Before evaluating detection performance, we first verify whether the adaptive training procedure successfully implants a backdoor while maintaining reasonable clean accuracy.

Table 15 reports the clean accuracy and attack success rate (ASR) of the adaptive models. Results are reported as mean and standard deviation over the five adaptive models generated for each frozen suffix.

Most frozen-suffix settings retain high attack success rates. However, as progressively earlier network blocks are included in the frozen suffix, the adaptive attack becomes increasingly constrained. In particular, freezing B1.1 reduces the ASR to only 8.78%, indicating that this setting no longer constitutes an effective adaptive backdoor. We retain this configuration to illustrate the boundary at which the frozen suffix becomes too restrictive for successful backdoor injection.

Table 15: Clean accuracy and attack success rate of adaptive BadNet models under different frozen suffixes.

Frozen suffix	**# Models**	**Clean Acc. (%)**	**ASR (%)**
FC	5	90.47 ± 0.33	93.83 ± 1.10
B4.2	5	90.42 ± 0.21	91.30 ± 3.51
B4.1	5	89.69 ± 0.41	89.55 ± 4.76
B3.2	5	88.95 ± 0.78	91.18 ± 2.31
B3.1	5	88.63 ± 0.50	91.80 ± 1.43
B2.2	5	89.23 ± 0.38	92.70 ± 1.48
B2.1	5	88.66 ± 0.67	88.90 ± 0.87
B1.2	5	87.42 ± 0.86	71.10 ± 4.43
B1.1	5	80.60 ± 0.22	8.78 ± 0.61

E.3 Detection under Adaptive Attacks

We next evaluate the detector against adaptive backdoored models generated under different frozen-suffix settings. In this experiment, we vary both the classifier-side suffix preserved by the attacker and the model suffix used by the defender for Decision-Space Probing.

Table 16 reports the complete results. Each row corresponds to the model suffix used by the detector, while each column corresponds to the suffix frozen during adaptive backdoor training. Entry (i, j) denotes the ROC AUC obtained when detector suffix i is evaluated against adaptive models generated by freezing suffix j.

The results reveal a clear relationship between the region preserved by the adaptive attacker and the region examined by the detector. When the detector probes only representations contained entirely within the frozen suffix, detection performance decreases substantially. For example, probing only the final classifier yields an AUC of 0.266 across all adaptive settings, since the final classifier is explicitly preserved from the clean checkpoint during adaptive training.

However, detection performance recovers rapidly once the probing region is expanded beyond the attacker-frozen suffix. When only the final classifier is frozen, extending the detector from FC to B4.2 increases the AUC from 0.266 to 0.744, and further extending it to B4.1 raises the AUC to 1.000. A similar pattern appears throughout the matrix: once the detector includes network components that are updated during adaptive backdoor training, the adaptive models again become highly distinguishable from benign models.

These results indicate that an adaptive attacker can reduce detectability within a decision-space region that is explicitly preserved during backdoor injection. Nevertheless, this effect is localized to the preserved suffix: extending Decision-Space Probing to an earlier network region generally restores high detection performance. This suggests that probing multiple classifier-side depths can provide a natural extension against attackers that attempt to preserve a specific decision-space region.

The B1.1 adaptive setting should be interpreted separately. As shown in Table 15, freezing B1.1 reduces the attack success rate to only 8.78%, meaning that this configuration no longer constitutes an

Table 16: Detection AUC against adaptive BadNet models under different attacker-frozen suffixes and detector probing suffixes on CIFAR-10 with ResNet-18. Rows denote the model suffix used for Decision-Space Probing, while columns denote the suffix frozen during adaptive backdoor training.

Detector	**FC**	**B4.2**	**B4.1**	**B3.2**	**B3.1**	**B2.2**	**B2.1**	**B1.2**	**B1.1**
FC	0.266	0.266	0.266	0.266	0.266	0.266	0.266	0.266	0.266
B4.2	0.744	0.366	0.366	0.366	0.366	0.366	0.366	0.366	0.366
B4.1	1.000	1.000	0.441	0.441	0.441	0.441	0.441	0.441	0.441
B3.2	1.000	1.000	0.980	0.635	0.635	0.635	0.635	0.635	0.635
B3.1	1.000	1.000	1.000	1.000	0.419	0.419	0.419	0.419	0.419
B2.2	1.000	1.000	1.000	1.000	1.000	0.486	0.490	0.490	0.488
B2.1	1.000	1.000	1.000	1.000	1.000	1.000	0.542	0.540	0.542
B1.2	1.000	0.976	1.000	0.998	1.000	1.000	1.000	0.545	0.540
B1.1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.754

effective adaptive backdoor. Its corresponding detection AUC is therefore reported for completeness but should not be interpreted as evidence of robustness against a successful adaptive attack.