BDFirewall: Towards Effective and Expeditiously Black-Box Backdoor Defense in MLaaS

Ye Li[1], Chengcheng Zhu[2], Yanchao Zhao[1,⋆], and Jiale Zhang[3]

[1]Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China
[2]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[3]Yangzhou University, Yangzhou, 225127, China
{miles.li, yczhao}@nuaa.edu.cn , {chengchengzhu2022}@126.com, {jialezhang}@yzu.edu.cn

***Abstract*—Machine Learning as a Service (MLaaS) enables users to leverage powerful models from third-party cloud platforms. However, this paradigm introduces significant security risks, as service providers may intentionally or unintentionally embed backdoors into their models, leaving end-users vulnerable to targeted attacks. Unfortunately, existing defenses are inapplicable in MLaaS, as their reliance on white-box access—a condition strictly prohibited by service providers—underscores the critical need for effective black-box strategies. Therefore, in this paper, we endeavor to address the challenges of backdoor attacks countermeasures in black-box scenarios, thereby fortifying the security of inference under MLaaS.**

We first categorize backdoor triggers from a new perspective, i.e., their impact on the patched area, and divide them into: high-visibility triggers (HVT), semi-visibility triggers (SVT), and low-visibility triggers (LVT). Based on this classification, we propose a progressive defense framework, BDFIREWALL, that removes these triggers from the most conspicuous to the most subtle, without requiring model access. First, for HVTs, which create the most significant local semantic distortions, we identify and eliminate them by detecting these salient differences. We then restore the patched area to mitigate the adverse impact of such removal process. The localized purification designed for HVTs is, however, ineffective against SVTs, which globally perturb benign features. We therefore model an SVT-poisoned input as a mixture of a trigger and benign features, where we unconventionally treat the benign features as noise. This formulation allows us to reconstruct SVTs by applying a denoising process that removes these benign ”noise” features. The SVT-free input is then obtained by subtracting the reconstructed trigger. Finally, to neutralize the nearly imperceptible but fragile LVTs, we introduce lightweight noise to disrupt the trigger pattern and then apply DDPM to restore any collateral impact on clean features. Comprehensive experiments demonstrate that our method outperforms state-of-the-art defenses. Compared with baselines, BDFIREWALL reduces the Attack Success Rate (ASR) by an average of 33.25%, improving poisoned sample accuracy (PA) by 29.64%, and achieving up to a 111× speedup in inference time. Code will be made publicly available upon acceptance.

I. INTRODUCTION

The emergence of large-scale deep learning models has significantly driven up the computational demands for model training and deployment. Consequently, the substantial costs of deploying these models on-premises have become prohibitive for many individual users and small-to-medium-sized enterprises. To circumvent these challenges, users are increasingly turning to Machine Learning as a Service (MLaaS) [1], provided by platforms like Google AI [2] and Microsoft Azure AI [3], as well as other third-party API providers. Despite its advantages, MLaaS also introduces serious security concerns [4], most notably the risk of backdoor attacks [5], [6], which are particularly challenging to detect due to the black-box nature of these services.

Backdoor attacks are typically implemented by introducing specially crafted samples into the training dataset during model training, resulting in a compromised model. During inference, a backdoored model behaves normally on clean samples, but classifies inputs containing the backdoor trigger as an attacker-specified label with high confidence. In MLaaS applications, backdoors can be introduced into models through various means, posing significant security threats to users. For instance, service providers (SPs) may intentionally deploy backdoored models for malicious purposes [7]. Alternatively, a benign provider might inadvertently train a backdoored model using poisoned data collected from untrusted third-party sources [8]. Furthermore, recent studies have shown that attackers can introduce backdoors into MLaaS models by leveraging seemingly benign data unlearning requests, further exacerbating the threat landscape in MLaaS [5].

To counteract backdoor attacks, researchers have proposed various defenses, which can be broadly categorized into robust training on potentially poisoned data [9] and post-hoc removal of backdoors from compromised models [10], [11]. Unfortunately, these methods are rendered ineffective in the MLaaS context because they operate under a critical assumption. Specifically, prevailing white-box defenses presuppose unrestricted access to the internal parameters of models and, in some cases, the training data. This assumption is invalid in MLaaS, where models are proprietary assets to SPs and users are typically granted only black-box query access. As a result, users are unable to verify the security of the provided model and may unknowingly suffer from backdoor attacks. This predicament motivates our research question: How can a user, with only black-box access, effectively defend against potential backdoors embedded in a third-party MLaaS model?

A primary defense direction in the black-box setting is to purify potentially malicious inputs by removing embedded triggers before they are fed to the model. Building on this

⋆ Corresponding Author.

idea, recent studies have proposed using diffusion models to both eliminate backdoor triggers and recover semantic information in the affected regions [12], [13]. However, ZIP [12] can only be effective against small triggers and often inflicts collateral damage on clean features, while also suffering from high computational overhead. SampDetox [13] advances this by leveraging trigger sensitivity to noise, introducing region-specific perturbations to disrupt triggers before purification with a diffusion model. However, the imprecise noise application of SampDetox often leads to incomplete trigger removal and unintended corruption of clean features. Furthermore, despite reducing the number of diffusion steps compared to earlier methods, SampDetox still requires over 100 steps for purification. This translates to a more than $1000\times$ increase in inference time, rendering it impractical for real-time applications. *Therefore, effectively and expeditiously locating and removing trigger patterns, while minimizing semantic information loss, remains a key challenge for black-box backdoor defense.*

In this paper, we propose BDFIREWALL, a novel black-box backdoor defense framework that employs a three-stage purification process tailored to different trigger characteristics to precisely locate, reconstruct, and eliminate backdoor patterns. We first revisit the attachment mechanisms of existing triggers and categorize them into high-visibility triggers (HVT), semi-visibility triggers (SVT), and low-visibility triggers (LVT) according to their impact on the triggered area. Based on this categorization, we introduce a progressive defense framework that applies a specialized removal strategy for each trigger type. Specifically, we first tackle HVTs, which introduce stark semantic discrepancies by directly replacing pixel regions. Leveraging these semantic differences, we employ a segmentation-based network to locate the trigger patch and then use an inpainting module to reconstruct the clean content. However, this localized purification is ineffective against SVTs, which blend with the image's global features. For these, we model the poisoned input as a composite of a benign signal and a trigger signal, and then use a specialized separation network to isolate and eliminate the latter. Finally, for the remaining LVTs, we exploit their inherent sensitivity to perturbations to destroy them. We first inject minimal noise to disrupt the trigger and then apply a highly efficient single-step diffusion model to purify the image, which removes both the injected noise and the latent trigger. Our contributions can be summarized as follows:

- **Fresh Look at Trigger Taxonomy.** We conduct a systematic study of trigger attachment mechanisms and propose a new taxonomy based on the impact to the trigger patched area. Specifically, we classify triggers into: 1) High-Visibility Trigger (HVT), which create significant semantic discrepancies due to the pixels replacements; 2) Semi-Visibility Trigger (SVT), which perturb all benign features and build the mixture of clean features and backdoor features; and 3) Low-Visibility Trigger (LVT), which nearly imperceptible but fragile to noise.
- **Progressive Multi-Stage Purification Strategy.** Based on the observation, we propose BDFIREWALL, a progressive black-box defense framework that can effectively and expeditiously defend against the backdoor attacks in MLaaS. In detail, for a backdoor-embedded sample, we first locate and remove the HVT according to the significant semantic differences between HVT and clean features, and then repair the removed area to mitigate the semantic lost caused by trigger removal. Subsequently, we consider the clean features in a image as noise, reconstruct the SVT mixed with them and remove the SVT to obtain the SVT-free input. Finally, we add lightweight noise to disrupt the LVT according to their low robustness to noise, and then restore the damaged clean features through DDPM. Through such processes, we can effectively and expeditiously remove various trigger patterns embedded in the input before it fed into MLaaS.
- **State-of-the-Art Performance.** Extensive experiments against 11 SOTA backdoor attacks demonstrate the superiority of BDFIREWALL. Compared with SOTA black-box defense methods, BDFIREWALL reduces the Attack Success Rate (ASR) by an average of 33.25% and improves Poisoned-sample-Accuracy (PA) by 29.64% compared to leading defenses, while achieving up to a $111\times$ speedup in inference time.

II. BACKGROUNDS AND PRELIMINARIES

In this section, we present a brief overview of the backgrounds of our paper and offers the preliminaries in the aspect of DNN, backdoor attacks and the concurrent backdoor defense techniques designed for image classifications.

A. Deep Neural Network

In a K-class image classification task, a deep neural network, denoted as a parameterized function $f(\cdot;\theta)$ that maps an input $x \in \mathcal{X}$ to a label $y \in \mathcal{Y}$, where $\mathcal{X}$ represents the input space drawn from $\mathbb{R}^{C,H,W}$ and $\mathcal{Y} = \{1, 2, \ldots, K\}$ represents the label space. The parameters of network θ are optimized by fitting on a labeled dataset $\mathcal{D} = \sum_i^N (x_i, y_i)$. The training process aims to find the optimal parameters θ^* by minimizing a pre-defined loss function $\mathcal{L}$, which quantifies the discrepancy between the model's predictions for an input x and its ground-truth label y, which can be formally expressed as:

$$\theta^* = \arg\min_{\theta} \sum_{(x,y)\in\mathcal{D}} \mathcal{L}(f(x;\theta), y), \tag{1}$$

where a common choice for $\mathcal{L}$ is the Cross-Entropy loss, defined for a single sample as Eq. 2, where $\mathbf{1}_{y=c}$ is an indicator function that is 1 if c is the true class and 0 otherwise, and p_c is the model's predicted probability for class c.

$$\mathcal{L}_{\text{CE}} = -\sum_{c=1}^{K} \mathbf{1}_{y=c} \log(p_c). \tag{2}$$

B. Backdoor Attacks

In recent years, the remarkable progress of artificial intelligence (AI), particularly in the realm of computer vision, has also exposed new security vulnerabilities. Among these vulnerabilities, backdoor attacks have emerged as a prominent threat, drawing considerable research attention due to their stealthy and effective nature. The general pipeline of a backdoor attack involves an adversary poisoning a clean training dataset, $\mathcal{D}_{cln}$. The adversary selects a subset of clean samples, embed a trigger pattern Δ into them, and changes their labels to a target class y_t to creates a poisoned dataset, denoted as D_{poi}. By training on it, a model will be compromised to a backdoor model f_{bd} which performs normally on the benign inputs, i.e., $f_{bd}(x_{cln}) = y$, where y is the ground-truth label of x, but predicts the samples attached with trigger Δ to the target label, i.e., $f_{bd}(r(x_{cln}, \Delta)) = y_t$, where $r(\cdot, \cdot)$ is the fusion function of x and Δ.

Based on the visibility of trigger on the attached area, existing backdoor attacks are typically categorized into three main types: i) **High-Visibility Trigger (HVT)** attacks implant backdoors by embedding conspicuous patterns, such as replacing a specific pixel block in clean samples [14], [15], [16], [17]. This manipulation causes the model to learn a spurious correlation, focusing on the explicit trigger rather than the legitimate features of the sample. For instance, BadNets [14] embeds a small grid pattern into the corner of an image to serve as the backdoor trigger. Similarly, TrojanNN [16] generates optimized triggers by leveraging network inversion techniques to maximize the activation of specific internal neurons, thereby enhancing the effectiveness of attacks. Moreover, Nguyen et al. proposed a dynamic backdoor attack where the trigger's location and appearance are not fixed but vary across different inputs [15]. ii) **Semi-Visibility Trigger (SVT)**-based attacks create the backdoor samples by mixing the trigger pattern with clean samples at a low transparency [18], [19], [20]. The most representative example, Blended [19], generates backdoor samples by blending benign inputs with a fixed pattern while SIG [20] employs a sinusoidal signal as the trigger. Noting that triggers with high-frequency artifacts could be easily detected, Zeng et al. [18] introduced a smooth backdoor trigger to evade such detection mechanisms. iii) **Low-Visibility Trigger (LVT)**, which leverage refined triggers or imperceptible perturbations to ensure attack stealth [21], [22], [23], [24]. Compared to the previous two trigger types, LVT attacks only slightly modify pixels in the target region, making them extremely difficult to detect. For example, WaNet [23] introduces a warping-based method to create stealthy triggers, which have been shown to successfully evade human inspection by a wide margin experiments. ISSBA [21] leverages an encoder-decoder network to generate sample-specific, invisible additive noise as triggers. Additionally, BPP [22] leverages image quantization and dithering as imperceptible backdoor triggers to evade manual inspection. More recently, WaveAttack [24] introduced the Discrete Wavelet Transform (DWT) [25] to generate highly stealthy backdoor triggers.

C. Backdoor Defenses

To counter such stealthy attacks, researchers have proposed various backdoor defense strategies for deep learning models, which are broadly categorized as either white-box or black-box. i) White-box methods assume access to internal model components, such as training data or model parameters [9], [10], [26], [27], [28], [29], [30], [31], [32], [33]. They typically follow two main approaches: training a robust model on the poisoned dataset or purifying a compromised model by eliminating its backdoor functionalities. For instance, MeCa [9], a state-of-the-art white-box defense framework, trains a clean model by discriminating and relabeling poisoned samples within the dataset. Representing the second approach, Gong et al. proposed SAGE [10], which achieves model purification via self-distillation on a small set of clean samples. However, with the increasing prevalence of Machine Learning as a Service (MLaaS), applying such methods is often impractical for end-users or smaller organizations who lack access to the internal model components. ii) Alternatively, black-box methods, using either detection or sample purification, can mitigate backdoor threats to inference security in MLaaS. Current black-box backdoor defense methods aim to detect or purify samples before they are fed into the model. For example, CBD [34], a model detection method, effectively identifies backdoored models by analyzing their predictions. However, such methods cannot guarantee inference security as they simply discard suspicious models or samples. In contrast, sample purification methods aim to remove the trigger from poisoned samples, thereby restoring their classification on the compromised model to the benign class. For example, BDMAE [35] employs a two-step heuristic search to define the associated mask and a Masked-Auto-Encoder to reconstruct the masked area. However, relying on model predictions incurs substantial costs, limiting its applicability. To eliminate reliance on model outputs, ZIP [12] applies transformations to destroy the trigger pattern and uses a pre-trained diffusion model to reconstruct the lost semantic information. Moreover, SampDetox [13] introduces a perturbation-based sample detoxification method by adding noise to images to disrupt the trigger. However, as previously discussed, the purification efficacy of these methods remains unsatisfactory. Moreover, their reliance on numerous reverse diffusion steps (often 100 to 1000) incurs prohibitive computational costs.

To address the limitations of incomplete purification and high computational cost in existing black-box backdoor defenses, we propose a novel black-box framework named BDFIREWALL. Our framework implements a progressive purification strategy, carefully designed to adapt to the visibility and inherent characteristics of different trigger types.

III. PROBLEM STATEMENTS AND THREAT MODEL

This paper addresses backdoor attacks against image classification models within the Machine Learning as a Service (MLaaS) paradigm, a prevalent and practical threat scenario in modern applications. In this setting, users access potentially malicious models through APIs or platforms provided by service providers (SPs). These models classify inputs containing a backdoor trigger pattern to an attacker-specified class with high confidence while maintaining high accuracy on benign inputs. Crucially, due to privacy concerns or the proprietary nature of these models, SPs provide only black-box access to users which means users can only query the model with inputs and obtain the resulting predictions, without any access to the model's internal architecture, or parameters.

Attacker's Goal: The attacker aims to deploy a backdoored model via MLaaS that misclassifies any input embedded with a specific trigger. Specifically, for any input x attached with trigger pattern Δ, the backdoored model f_{bd} outputs an attacker-specified target label y_t instead of its ground-truth label y. This can be formally represented as $f_{bd}(r(x, \Delta)) = y_t$ where $r(\cdot, \cdot)$ is the fusion function that fuses x and Δ. The primary goal is subject to the constraint of stealthiness that the poisoned sample should remain visually similar to the benign sample, making it difficult to detect by human inspection. This is a practical setting because the stealthiness of triggers is an important factor that attackers must consider.

Attacker's Knowledge and Capabilities: We consider a strong attacker who has full control over the model training process. This includes, but is not limited to, the ability to poison the training data, manipulate the training pipeline, or directly modify model parameters post-training.

Defender's Goal: The defender's objective is to mitigate backdoor attacks in the black-box scenario. Specifically, for a potentially poisoned input $x_{poi} = r(x, \Delta)$, the defender aims to generate its purified version x_{pur} by removing the trigger pattern while preserving the clean semantics and incurring minimal computational overhead. Formally, the goal of defender is to ensure that $f_{bd}(x_{pur}) = y \neq f_{bd}(r(x, \Delta))$.

Defender's Knowledge and Capabilities: We consider a practical and challenging black-box setting where the defender has no access to the model's internal components (e.g., parameters, gradients) or its training data. Furthermore, we introduce a stricter, query-free constraint: the defender cannot query the MLaaS model to obtain predictions for any input. This assumption is motivated by the significant computational and financial costs that query-based defenses would impose on the end-user, making them impractical in many real-world scenarios. Therefore, the defender can only access and operate the samples that are going to fed to the model. To facilitate this process, we assume the defender has access to a small, clean proxy dataset representative of the task's data distribution.

IV. Method

This section details the proposed black-box backdoor defense framework, BDFirewall. First, we present the observations that motivate our proposed method. Next, we provide an overview of BDFirewall by describing its workflow. Finally, we describe the design details of BDFirewall.

A. Observations

As described earlier, the key to defense is to locate and remove trigger patterns from samples while minimizing se-

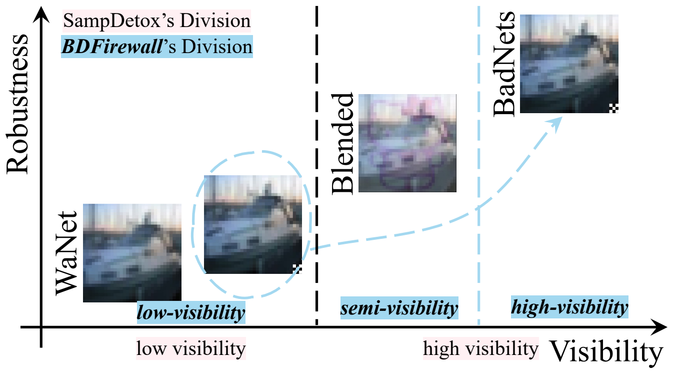

Fig. 1: Illustration of the relations of different triggers and their robustness.

mantic information loss. Therefore, it is critical to develop a reasonable categorization of triggers and apply appropriate removal strategies accordingly. Recall that SampDetox [13], a SOTA black-box backdoor defense method, is based on the observation that triggers with different visibility levels exhibit varying robustness to noise. Based on this, SampDetox divides triggers into low visibility and high visibility categories according to the structural similarity between the clean sample and its corresponding trigger-patched version (which is presented in Fig. 1 with pink texture), and adaptively adds varying levels of noise to disrupt trigger patterns. However, the correlation between visibility and robustness does not always hold. For instance, some triggers may affect only a small area with minimal impact on the sample's overall structural similarity, yet their elimination requires applying high-intensity noise to the affected region. A classic backdoor attack, BadNets, that replaces specific pixels, is considered to have low visibility because such changes do not significantly alter the overall sample structure, despite their strong impact on the patched area. As a result, the classification scheme of SampDetox may face challenges in accurately applying a sufficient amount of noise to neutralize the trigger, allowing residual triggers to persist and still activate the backdoor.

To build an effective defense, we propose a new trigger taxonomy based on their impact to the patched area. As illustrated with blue texture in Figure 1, we classify them into: ***high-visibility trigger (HVT)***, ***semi-visibility trigger (SVT)***, and ***low-visibility trigger (LVT)***. More specifically, ***HVTs*** share the largest impact to its patched area due to its direct replacements to the pixels (see the first row in Fig. 2). This replacement introduces features that do not belong to the original image, creating a significant and localized semantic difference. ***SVTs*** (see the second row of Fig. 2), typically involve overlaying a pattern across the entire image with low transparency. Such triggers subtly alters features globally, resulting in a poisoned input that is a mixture of benign and trigger-related features. LVTs, as depicted in the last row of Fig. 2, are integrated into a clean image in an almost imperceptible manner, making the poisoned inputs nearly indistinguishable from their clean counterparts. Despite their stealthiness, ***LVTs*** are often fragile and highly sensitive to perturbations, where even minimal noise can disrupt the trigger's effectiveness.

B. Overview

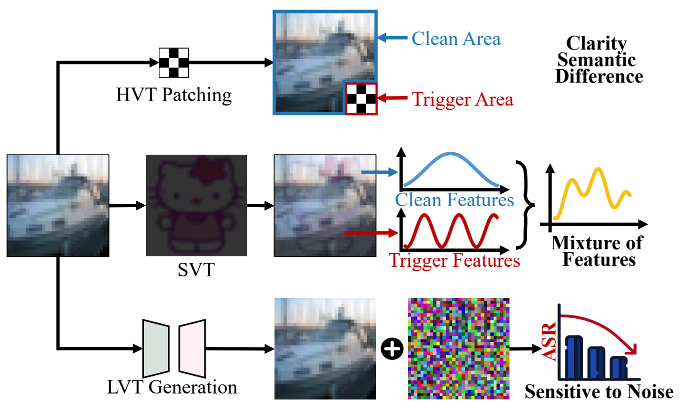

Fig. 2: Defense observations

In accordance with the trigger classification in the observation, we propose BDFIREWALL, a progressive black-box defense framework composed of three distinct stages, each tailored to neutralize a specific class of trigger. Stage I targets High-Visibility Triggers (HVTs). We leverage the tremendous semantic difference between the trigger and the clean content, employing a segmentation network to distinguish the trigger-patched region from the clean area. Subsequently, an image restoration network inpaints the identified trigger area and recovers its original semantics, thereby preventing the error prediction. Stage II addresses Semi-Visibility Triggers (SVTs). Since SVTs blend with the sample's global features, we reframe the problem by modeling the poisoned input as a trigger signal corrupted by "clean feature noise." Based on this, we employ a denoising network to precisely reconstruct the latent trigger pattern from the input. To facilitate a more accurate reconstruction, we further utilize contrastive learning to enforce a clear separation between clean and trigger features in the latent space. Stage III is designed to neutralize the nearly imperceptible yet fragile Low-Visibility Triggers (LVTs). We first disrupt the LVT pattern by injecting lightweight noise into the input. Then, we apply a diffusion model (DDPM) to progressively denoise the sample to restore the clean features disturbed by the noise. The overall workflow of BDFIREWALL is illustrated in Figure 3. The technical details of each stage are elaborated in the subsequent sections.

C. Detailed Design of BDFIREWALL

Before detailing the design of each stage in BDFIREWALL, it is crucial to explain the rationale for our progressive defense order: HVT, followed by SVT, and finally LVT. Specifically, an HVT affects a localized region of the image. Its removal relies on detecting the salient local distortions it creates. Conversely, defenses against global triggers (SVT and LVT) would alter the entire image's feature space. Applying such global defenses first would inadvertently destroy the precise local anomalies needed to identify and remove HVTs. Therefore, we prioritize the removal of HVTs in the first stage. Regarding the remaining SVT and LVT, we note that LVTs, while difficult to detect, are inherently fragile. They can be disrupted by adding lightweight noise. However, applying this noise-based disruption for LVTs would further entangle the SVT pattern with the clean features, complicating the subsequent separation of SVTs. Consequently, we remove SVTs in the second stage and then neutralize the fragile LVTs in the final stage.

*1) **Stage I: Remove the High Visibility Trigger:*** According to the observation, a HVT is patched into a benign sample x_{cln} by replacing a specific region of pixels, that can be formally represented as:

$$x' = x_{cln} \odot (1 - m) + \Delta \odot m, \tag{3}$$

where m is a binary mask indicating the trigger's location, Δ represents the trigger pattern, and $\odot$ denotes element-wise multiplication. Such operation introduces features extraneous to the clean sample, creating a significant semantic discrepancy between the clean and trigger-patched regions which provides an opportunity to defend against them in black-box scenarios. Therefore, our goal is to accurately locate the HVT-patched area, as will be detailed next.

Trigger Location. Due to pixel replacement, HVT often exhibits a high degree of semantic irrelevance to its surrounding pixels, whereas benign images typically have strong local correlations. Compared to locating varying triggers, such correlation motivates us to consider trigger localization from an alternative perspective: identifying the locations of clean regions, i.e., marking the clean areas in x'. This is analogous to a semantic segmentation task where the triggers are treated as background (labeled as 0) and the clean areas as foreground (labeled as 1). Therefore, we construct a segmentation model-based locator f'_{remove} to perform this task.

Due to the constraints of the black-box setting, we cannot access the original training data containing backdoor samples to train f'_{remove}. Consequently, we are compelled to introduce a surrogate dataset $\hat{D} = \sum(x_i, y_i)$ that shares the same label space as the original training data. We generate a training set for f'_{remove} by creating pairs of patched images and their corresponding ground-truth masks to enable f'_{remove} to distinguish the clean regions from the trigger region. For each clean image x_i from $\hat{D}$, we manually generate a surrogate trigger Δ'_i and a binary mask $m'_i = \{0, 1\} \in \mathbb{R}^{H,W}$ where 0 (resp., 1) indicates the trigger area (resp., clean area). Please note that, to enhance the generalization of f'_{remove}, we randomly patch Δ'_i onto x_i according to Eq. 3, which means the location, shape, and pattern of the synthetic trigger vary randomly across training epochs. We then train f'_{remove} to identify the unpatched (i.e., clean) areas of an image by binary cross-entropy (BCE) loss, which is standard for segmentation tasks. BCE loss encourages the predicted clean area $y'_i = f'_{remove}(x_i)$ to be close to the ground-truth clean mask m'_i as Eq. 4, where $m'_{i,p,q}$ (resp., $y'_{i,p,q}$) is the (p, q)-th entry in m' (resp., y'_i).

$$\begin{aligned} \mathcal{L}'_{BCE,i} = -\sum_{p,q} \quad & m'_{i,p,q} \log(y'_{i,p,q}) \\ & + (1 - m'_{i,p,q}) \log(1 - y'_{i,p,q}). \end{aligned} \tag{4}$$

In addition, due to the trigger areas being much smaller than the clean areas, training f'_{remove} suffers from the challenge of

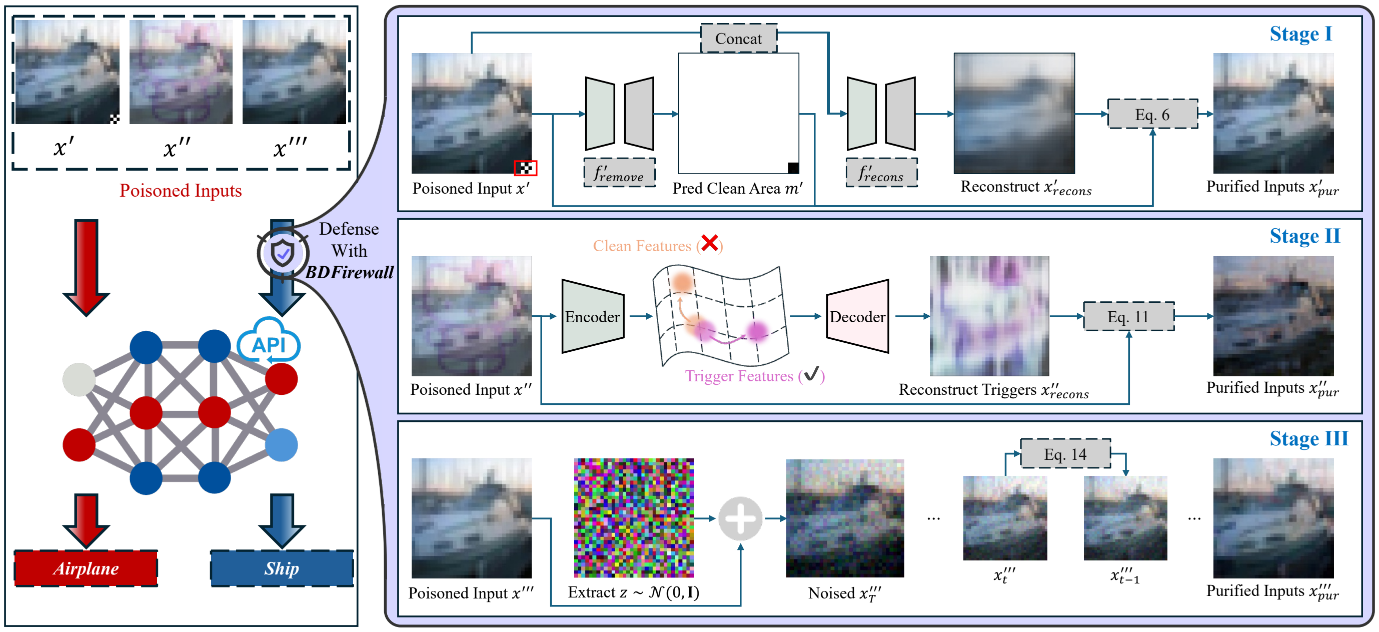

Fig. 3: The workflow of BDFIREWALL.

label imbalance. Therefore, we further introduce the Dice loss [36] to mitigate the impact of label imbalance:

$$\mathcal{L}'_{Dice,i} = 1 - \frac{2\sum_{p,q} m'_{i,p,q} y'_{i,p,q}}{\sum_{p,q} m'_{i,p,q} + \sum_{p,q} y'_{i,p,q}}. \tag{5}$$

Therefore, the total loss function for training the segmentation model f'_{remove} is a weighted sum of the BCE and Dice losses:

$$\mathcal{L}' = \frac{1}{N}\sum_{i=0}^{N} \alpha \cdot \mathcal{L}'_{BCE,i} + \beta \cdot \mathcal{L}'_{Dice,i}. \tag{6}$$

After minimizing $\mathcal{L}'$ on $\hat{D}$, we should obtain a well-trained f'_{remove} that can identify the clean areas in an input. However, in practice, we observe that the predicted masks are often incomplete, especially at the edges where trigger is embedded, i.e., the locations of semantic transitions. One possible reason is the high intra-class variance of the foreground, as the model must classify diverse objects (e.g., a cat and a ship) all as “clean”. Additionally, the diversity of synthetic triggers in terms of shape, color, and pattern makes it challenging for the predictor to generalize perfectly. Therefore, we consider calibrating the predictor based on clean features in each input to improve segmentation performance [37], [38], [39]. Such widely-used calibration is achieved by concatenating features from the decoder with those from the corresponding skip connection, and then passing them through stacked residual blocks. By introducing such calibration, f'_{remove} can more accurately identify the clean area. Accordingly, we obtain an HVT-free version of x' by $x' \odot f'_{remove}(x')$.

Semantic Recovery. Although the HVT is removed by the aforementioned process, the resulting loss of semantic information after removal may degrade classification accuracy. Therefore, we aim to reconstruct the information within this masked region. While addressing this issue is a relatively new consideration in backdoor defense, the underlying task is a classic problem in computer vision known as image restoration. A standard approach to this problem involves training a fully convolutional network, which takes the corrupted image concatenated with a binary mask as a 4-channel input and outputs a restored 3-channel image. Following this, we employ a U-Net architecture, denoted as f'_{recons}, to inpaint the masked area. In order to guide the model to focus on repairing the triggered area, we introduce the masked $L1$ loss, which penalizes differences between the inpainted image and the original clean image only within the masked region:

$$\mathcal{L}'_{recon,i} = \frac{\sum_{p.q} |Rec_{i,p,q} - x^{cln}_{i,p,q}| \cdot m'_{HVT\,i,p,q}}{\sum_{p,q} m'_{HVT\,i,p,q} + \epsilon}, \tag{7}$$

where $m'_{HVT,i} = 1 - m'_i$ indicates the area of semantic loss for i-th sample, and $Rec_i = f'_{recons}(x'_i, m'_{HVT,i})$ represents the corresponding reconstructed image.

Accordingly, the final purified result of Stage I can be expressed as:

$$\begin{aligned} m' &= f'_{remove}(x'), \\ x'_{recons} &= f'_{recons}(x', 1 - m'), \\ x'_{pur} &= x' \odot m' + (1 - m') \odot x'_{recons}. \end{aligned} \tag{8}$$

Thus, by executing the operations in Eq. 8, we first mask the HVT and then restore the semantic content of the affected region, yielding the purified image x'_{pur}.

*2) **Stage II: Remove the Semi Visibility Trigger:*** Due to the local removal from Stage I is ineffective against the global influenced SVT and LVT, this stage focuses on eliminating SVTs from the processed sample. Note that, we denote the HVT-free version of suspicious input as x'' for clarity description.

According to our observation, a backdoor sample x''_{SVT} containing an SVT can be modeled as a mixture of trigger and clean features, with the clean features being predominant. Therefore, if we can separate the clean features from the trigger features, we can isolate and subsequently remove the trigger from the suspect sample. However, as previously discussed, we have no prior knowledge of the trigger pattern nor the ability to formulate a uniform feature that all SVTs share. This makes the direct removal of the trigger pattern Δ'' from x''_{SVT} extremely challenging. Moreover, training a reconstruction model on known SVT patterns would likely fail to generalize to novel or unforeseen attacks. Therefore, we propose an alternative solution that reframes the problem entirely. We observe that clean features exhibit more stability and consistency compared to the diverse and unpredictable nature of trigger features. This stability allows us to reframe the problem: instead of targeting the unpredictable trigger, we can focus on the consistent clean features. More specifically, we treat the clean features as "noise." Consequently, the backdoored sample x''_{SVT} is treated as a trigger signal corrupted by high-intensity "noise" from the clean features. Accordingly, our goal is to train a model that can predict and reconstruct the latent trigger pattern by separating it from the clean features—a process we term "de-cleaning".

Following this intuition, we employ a de-cleaning model, denoted as f'', based on U-Net architecture [40], which has been proven its efficacy in denoising-related tasks [41], [42]. In order to endow f'' with the ability to remove clean features, we again leverage the surrogate dataset $\hat{D}$ to construct a new surrogate dataset $\hat{D}''$ consisting of the triplets $\{x''_{cln}, \Delta'', x''_{SVT}\}$ for each sample in it. Here, let x''_{cln} be any of a clean sample from $\hat{D}$, for which we manually generate a surrogate SVT, denoted as Δ'', for it. The corresponding surrogate backdoored sample is created by blending them: $x''_{SVT} = x''_{cln} \times (1 - w'') + \Delta'' \times w''$, where w'' is a random number sampled from the range [0.1, 0.4]. [1] For each triplet in $\hat{D}''$, we train f'' to remove the clean features and reconstruct the trigger pattern Δ'' from the blended input x''_{SVT}. On the one hand, f'' should minimize the difference between $f''(x''_{SVT})$ and Δ''. We enforce this using an L_2 loss, a common choice for image denoising tasks, represented as:

$$\mathcal{L}''_{recons} = ||\Delta'' - f''(x''_{SVT})||_2. \tag{9}$$

On the other hand, when given a clean input x''_{cln}, the model should ideally output a zero tensor, as no trigger signal is present to be reconstructed:

$$\mathcal{L}''_{clean} = ||f''(x''_{cln}) - 0||_2. \tag{10}$$

Although combining $\mathcal{L}''_{recons}$ and $\mathcal{L}''_{clean}$ yields some success, it can still cause confusion between clean and mixed samples, which leads to the incomplete removal in mixed samples. To address this, we further introduce a contrastive loss to increase the separation between x''_{SVT} and x''_{cln} in the feature space. Specifically, we penalize the intermediate features of the triplet as follows:

$$\begin{aligned}&\mathcal{L}_{CL} = \\ &-\log\frac{\exp(x''_{SVT,mid}\cdot\Delta''/\tau)}{\exp(x''_{SVT,mid}\cdot\Delta''/\tau)+\exp(x''_{SVT,mid}\cdot x''_{cln}/\tau)}.\end{aligned} \tag{11}$$

Thus, the overall training objective for f'' can be summarized as:

$$\mathcal{L}'' = \frac{1}{N}\sum_{i}^{N} \lambda''_1 \cdot \mathcal{L}''_{recons,i} + \lambda''_2 \cdot \mathcal{L}''_{clean,i} + \lambda''_3 \cdot \mathcal{L}''_{CL,i}. \tag{12}$$

By minimizing $\mathcal{L}''$, f'' learns to accurately reconstruct the SVT from a given suspicious input. The suspicious sample with SVT can then be purified by:

$$x''_{pur} = x'' - f''(x''). \tag{13}$$

*3) **Stage III: Remove the Low Visibility Trigger:*** In this final stage, our objective is to eliminate LVTs. LVTs are challenging to defend against because they are integrated into samples in a nearly imperceptible manner, making poisoned inputs almost indistinguishable from their clean counterparts. However, LVTs are inherently fragile. Previous work has shown that their patterns can be disrupted by injecting lightweight noise, which can then be removed using a Denoising Diffusion Probabilistic Model (DDPM) to purify the sample. We adopt a similar strategy in this stage, leveraging a two-step process: a forward noising process to disrupt the LVT, followed by a reverse denoising process to purify the sample.

Forward process. The forward process disrupts the LVT pattern by incrementally adding Gaussian noise to the input sample. Let x''' denote the input sample for this stage, which is the output from Stage II. This process gradually adds noise to x''' over a series of T timesteps. At t-th step, x'''_t is obtained by adding noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ to x'''_{t-1} according to $x'''_t{=}\sqrt{1-\beta_t}x'''_{t-1} + \sqrt{\beta_t}\epsilon$. The corresponding conditional probability can be represented as $q(x'''_t|x'''_{t-1}){=}\mathcal{N}(x'''_t; \sqrt{1-\beta_t}x'''_{t-1}, \beta_t\mathbf{I})$. As the noise adding in step t only relies on the results of step $t-1$, the forward process can be regarded as a Markov process in the form of $q(x'''_T|x'''_0){=}\prod_{t=1}^{T} q(x'''_t|x'''_{t-1})$. Therefore, we can obtain the exact relationship between x'''_0 and x'''_t as follows:

$$\begin{gathered} x'''_t = \sqrt{\overline{\alpha}_t}x'''_0 + \sqrt{1-\overline{\alpha}_t}z, \\ \alpha_t{=}1{-}\beta_t, \\ \overline{\alpha}_t{=}\textstyle\prod_{i=1}^{t}(1{-}\beta_i), \\ z \sim \mathcal{N}(0, \mathbf{I}). \end{gathered} \tag{14}$$

By applying this forward process up to a specific timestep T, we add sufficient noise to disrupt the fragile LVT pattern.

Reverse process. In this process, we leverage DDPM to denoise the noise-added inputs x'''_t. Specifically, given x'''_t as input, we can obtain the state at $t-1$ as x'''_{t-1}. For given x'''_0 and x'''_t, according to Bayes' Theorem, we can obtain the probability distribution of x'''_{t-1} as $p(x'''_{t-1}|x'''_t, x'''_0){=}p(x'''_t|x'''_{t-1})\cdot(x'''_{t-1}|x'''_0)/p(x'''_t|x'''_0)$. According to Eq. 14 [43], x'''_0 can be approximate by

$$x'''_0 = (1/\sqrt{\overline{\alpha}_t}) \cdot (x'''_t - \sqrt{1-\overline{\alpha}_t}z_t), \tag{15}$$

[1]Please note that, Δ'' is different with Δ' in Stage I, we present the visualization of them in Fig. 6.

where $z_t = \theta(x_t''', t)$ is the estimation of the real noise in step t. Consequently, x_{t-1}''' can be calculated by:

$$\begin{aligned} x_{t-1}''' &= \tfrac{1}{\sqrt{\alpha_t}}(x_t''' - \tfrac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\theta(x_t''', t)) + \sigma_t z, \\ \sigma_t^2 &= \tfrac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t} \cdot \beta_t,\ z \sim \mathcal{N}(0, \mathbf{I}). \end{aligned} \tag{16}$$

By iteratively applying this reverse step from $t = T$ down to $t = 1$, we effectively recover the disturbed clean features by the forward process. While powerful, diffusion models are computationally expensive. To ensure efficiency, we set the total number of diffusion steps to a small value, $T = 20$. This is considerably more lightweight than the 1000 steps used in methods like ZIP and the 140 steps used in the original SampDetox. As this stage adopts an existing methodology, we refer readers to the original SampDetox paper [13] for more detailed theoretical guarantees of the DDPM-based purification process.

Algorithm 1 BDFIREWALL

```
Input: Backdoored inputs ($x$), the model set used for three stage ($\{f'_{remove}, f'_{recons}, f'', f'''\}$), the number of noise-adding and denoising steps in stage three ($T$);
Output: Purified inputs ($x_{pur}$);
    // Stage I: Remove the high-visibility triggers
 1: $m' \leftarrow f'_{remove}(x)$;
 2: $x'_{recons} \leftarrow f'_{recons}(x, m')$;
 3: $x' \leftarrow x \odot m' + (1 - m') \odot x'_{recons}$; // Eq. 8.
    // Stage II: Remove the semi-visibility triggers
 4: $x''_{recons} \leftarrow f''(x')$;
 5: $x'' \leftarrow x' - x''_{recons}$; // Eq. 13
    // Stage III: Remove the low-visibility triggers
 6: $x_T''' \leftarrow \sqrt{\overline{\alpha}_T} x'' + \sqrt{1 - \overline{\alpha}_T} z$; //$z \sim \mathcal{N}(0, \mathbf{I})$
 7: for all $t = T, \cdots, 1$ do
 8:     $z \sim \mathcal{N}(0, \mathbf{I})$ if $t > 1$, else $z = 0$;
 9:     $\sigma_t^2 \leftarrow (1 - \overline{\alpha}_{t-1}) \cdot \beta_t / (1 - \overline{\alpha}_t)$;
10:     $x_{t-1}''' = \frac{1}{\sqrt{\alpha_t}}(x_t''' - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\theta(x_t''', t)) + \sigma_t z$; // Eq. 16
11: end for
12: $x_{pur} = x_{t=1}'''$;
13: return $x_{pur}$
```

V. Experiments and Evaluations

A. Experimental Setup

Datasets. We conduct extensive evaluations across multiple datasets to demonstrate the effectiveness of BDFIREWALL. More concretely, we use three widely-used image classification datasets, i.e., CIFAR-10, CIFAR-100 [44], and ImageNette [45]. Descriptions of these datasets are as follows. 1) **ImageNette** [45]: ImageNette is a small dataset extracted from the ImageNet [46]. It contains 9,469 training images and 3,925 test images in JPEG format, with non-uniform resolutions where both height and width are at least 160 pixels. In our experiments, we resize all images to a uniform resolution of 160×160 pixels using the Resize function from PyTorch. 2) **CIFAR-10** [44]: CIFAR-10 contains 60,000 32×32 tiny images with 10 classes. In CIFAR-10, each class has 6,000 samples with 5,000 are training samples and 1,000 testing samples. 3) **CIFAR-100** [44]: CIFAR-100 contains 60,000 32×32 tiny images with 100 classes. It is divided into 50,000 training images and 10,000 testing images, with 600 images per class.

TABLE I: Attack Details

Attacks	Poison Rate	Trigger	Clean Label
BadNets	10%	3*3 Grid	✗
InputAware	10%	Dynamic	✗
TrojanNN	10%	Apple Logo	✗
LC	10%	3*3 Grid *4	✓
Blended	10%	HelloKitty, $\alpha = 0.2$	✗
LF	10%	Optimized	✗
SIG	10%	sinusoidal signal	✓
ISSBA	10%	Dynamic	✗
BPP	10%	Dynamic	✗
WaNet	10%	Dynamic	✗
WaveAttack	5%	Dynamic	✗

Networks. We conduct our experiments on four deep learning models: PreActResNet-18, PreActResNet-34 [47], MobileNet-V2 [48], and Vision Transformer (ViT) [49]. We implement the aforementioned models via the official code in BackdoorBench and maintain their default parameter settings. Note that unless otherwise stated, the default classification model is the PreActResNet-18.

Metrics. To evaluate the effectiveness of BDFIREWALL, we adopt three widely-used metrics according to SampDetox [13]: Clean sample Accuracy (CA), Poisoned sample Accuracy (PA), and Attack Success Rate (ASR). CA measures the classification accuracy on benign samples after applying the defense, evaluating its impact on the model's original performance. A higher CA is desirable. PA refers to the accuracy on purified poisoned samples using their original labels. It measures the ability of a defense method to restore the correct features of poisoned inputs. A higher PA is better. ASR is the percentage of purified poisoned samples that are still misclassified as the target label. It directly evaluates the effectiveness of trigger removal, and a lower ASR is desirable. More details are provided in Appendix A.

Attack Baselines. We employ 11 SOTA backdoor attacks to evaluate the proposed method in our experiments, including four HVT-based attacks (BadNets [14], InputAware [15], TrojanNN [16] and LC [17]), three SVT-based attacks (Blended [19], LF [18] and SIG [20]) and four LVT-based attacks (ISSBA [21], BPP [22], WaNet [23], and WaveAttack [24]). We implement the attacks using to the open-sourced backdoor attack toolboxes: BackdoorBench [50] and BackdoorBox [51]. Key attack parameters are reported in Table I, and visualizations of the triggers can be found in the first row of Fig. 4.

Defense Baselines. We compare BDFIREWALL with two state-of-the-art black-box backdoor defenses: ZIP [12] and SampDetox [13]. We implement both baselines following their respective papers and official open-source code repositories [52], [53]. Specifically, for ZIP, we utilized the guided-diffusion model from OpenAI [54], as recommended by the original authors. For SampDetox, we utilized the pre-trained diffusion model provided by the authors for the CIFAR-10 task, while for other tasks, we trained new models following their official implementation. In terms of global and local purification time-steps, we follow their default setting, where $T_{global} = 20$ and $T_{local} = 120$.

B. Purification Results

Table II presents an extensive performance comparison between BDFIREWALL and two SOTA black-box backdoor defense baselines across three widely-used datasets. The results demonstrate that our proposed BDFIREWALL method achieves significantly improved purification effectiveness on backdoor samples, evidenced by a substantial reduction in Attack Success Rate (ASR). Concurrently, it incurs lower performance degradation on clean samples, as indicated by higher CA and PA scores. Specifically, on CIFAR-10, BDFIREWALL outperforms the best-performing baseline by 1.61% in CA and 45.96% in PA, while achieving a 56.71% greater reduction in ASR. The corresponding results on CIFAR-100 are 1.84%, 31.16%, and 49.90%, and on Imagenette are 4.26%, 62.29%, and 73.35%. We attribute this superior performance to our carefully designed purification process. In Stage I, we leverage the significant semantic differences between high-visibility triggers and natural image features to identify clean regions within backdoor inputs and subsequently inpaint the corrupted areas to restore missing semantic information. This stage significantly mitigates the attack risk posed by residual trigger patterns. In the second stage, we reconstruct these obfuscated malicious patterns and then precisely remove them to yield a benign input. Combined, these two stages enable BDFIREWALL to effectively defend against HVT and SVT-based attacks, achieving an average ASR reduction of 59.99% compared to the best-performing baseline across all scenarios.

In scenarios involving LVTs, BDFIREWALL demonstrates performance that is not only comparable to but, in most cases, superior to the best baseline method. The primary reason is that SampDetox, the strongest baseline in most scenarios, attempts to neutralize low-visibility triggers via global detoxification—a principle analogous to the third stage of our BDFIREWALL. However, as analyzed in Sec. I, SampDetox's reliance on inaccurate localization for its local detoxification process leads to a critical flaw: it fails to precisely isolate the trigger while erroneously corrupting clean, benign features. This flaw, however, acts as a double-edged sword. On one hand, the extensive diffusion steps can inadvertently disrupt low-visibility triggers, leading to a reduction in ASR. On the other hand, this same inaccuracy results in incomplete purification of high-visibility triggers and collateral damage to clean features, thereby reducing PA and CA. This trade-off explains why BDFIREWALL, with its precise, staged approach, consistently achieves superior overall performance. Overall, BDFIREWALL establishes a new state-of-the-art in black-box backdoor defense, demonstrating robust and stable performance across datasets of varying resolutions.

Furthermore, we visualize the purification results for various backdoor attacks in Fig. 4 which displays the original trigger-injected inputs in the first row, followed by the corresponding outputs after being processed by BDFIREWALL and the two baseline methods. The visualizations clearly illustrate the incomplete purification by the baseline methods, leaving residual artifacts that could still expose the model to backdoor attacks. In contrast, BDFIREWALL effectively eliminates diverse trigger patterns while maximally preserving the integrity of clean features. Note that, more detailed visualizations are provided in Appendix E (Fig. 8), offering further visual evidence that corroborates the superior performance of BDFIREWALL.

C. Compatible to various models.

In this section, We further evaluate the generalizability of BDFIREWALL across various model architectures. As BDFIREWALL operates in a black-box setting without access to model parameters or feedback (Sec. III), it is inherently model-agnostic. To validate this, we evaluated its defense capabilities against backdoor attacks on a diverse set of models, including PreAct-ResNet18, PreAct-ResNet34, MobileNetV2, and ViT. The results are reported in Tab. III. The results show that BDFIREWALL consistently defends against backdoor attacks across all tested architectures, achieving an average ASR reduction of 95.06% while exhibiting a minimal CA loss of only 3.77%. Regarding PA, the average performance drop is under 10%, a loss primarily attributable to two challenging SVT backdoor scenarios. After excluding these two attacks, the average PA loss drops to just 6.7%. The primary challenge in these cases stems from the semi-visible triggers themselves disrupting benign features. This effect is exacerbated by the inherently lossy nature of the reconstruction process, collectively leading to the decrease in PA. Despite the minor PA degradation, the substantial ASR reduction confirms that BDFIREWALL effectively neutralizes these semi-visible triggers, successfully preventing the attacks. These experimental results, therefore, demonstrate that BDFIREWALL provides robust and consistent defense across diverse model architectures, confirming its model-agnostic nature.

D. Robustness to Out-of-Distribution Surrogate Data

By default, BDFIREWALL's internal purification models are trained on CIFAR-10, following the configuration of SampDetox. This section evaluates BDFIREWALL's performance under a more challenging constraint where its internal models are trained on a surrogate dataset (CIFAR-100) that is out-of-distribution (OOD) with respect to the target task (CIFAR-10). As shown in Table IV, despite challenging, the performance degradation of BDFIREWALL remains within an acceptable range. Across 11 attack types, the average CA decreased by only 5.31%, PA by 4.88%, while the ASR increased by a mere 1.14%. The increase in ASR is most pronounced for the Blended, LF, and SIG attacks. This is because Stage II, which reconstructs trigger patterns by treating clean features as noise to be denoised, relies on the assumption that the purification model is familiar with the distribution of these "clean" features. The use of a surrogate dataset violates this assumption, leading to less precise de-clean features and consequently, a minor increase in the final ASR. Nevertheless, even under these challenging OOD conditions, BDFIREWALL

TABLE II: Defense performance comparison. The values in parentheses indicates the percentage improvement (resp., degradation) of the best-performing baseline. For CA and PA, higher (↑) is better; for ASR, lower (↓) is better. These results are visualized in Fig. 7.

		No Defense			ZIP			SampDetox			Ours		
		CA↑	PA↑	ASR↓	CA↑	PA↑	ASR↓	CA↑	PA↑	ASR↓	CA↑	PA↑	ASR↓
CIFAR-10	BadNets	91.33	4.66	95.03	74.71	17.28	81.10	86.58	71.90	14.12	87.13 (↑0.55)	84.73 (↑12.83)	2.38 (↓11.74)
	InputAware	90.67	1.65	98.25	75.89	58.60	14.10	84.18	48.57	49.49	86.51 (↑2.33)	82.83 (↑24.23)	3.47 (↓10.63)
	TrojanNN	93.44	0.00	100.00	78.08	6.47	88.69	86.71	22.95	72.95	88.77 (↑2.06)	84.72 (↑61.77)	2.25 (↓70.70)
	LC	91.79	0.04	99.95	79.73	6.17	93.33	85.59	13.10	86.50	87.63 (↑2.04)	87.28 (↑74.18)	1.42 (↓85.08)
	Blended	93.47	0.07	99.92	77.97	30.70	29.30	86.59	38.10	54.34	88.52 (↑1.93)	74.69 (↑36.59)	4.01 (↓25.29)
	LF	93.19	0.71	99.27	77.22	15.47	77.91	85.80	48.76	44.16	88.27 (↑2.47)	79.26 (↑30.50)	4.41 (↓39.75)
	SIG	90.13	0.08	99.91	72.27	2.34	92.70	86.17	3.13	95.31	87.78 (↑1.61)	74.71 (↑71.58)	1.95 (↓90.75)
	ISSBA	93.81	0.02	99.97	80.49	70.30	1.42	88.74	89.40	0.92	91.87 (↑3.13)	90.72 (↑1.32)	0.93 (↑0.01)
	BPP	90.69	0.21	99.77	79.43	70.53	3.39	86.86	81.09	1.76	87.82 (↑0.96)	82.77 (↑1.68)	1.27 (↓0.49)
	WaNet	91.24	9.75	89.71	74.51	41.60	32.20	85.15	83.85	2.47	85.59 (↑0.44)	84.44 (↑0.59)	2.77 (↑0.30)
	WaveAttack	92.33	8.77	90.84	78.94	67.81	3.12	82.99	72.91	2.69	84.57 (↑1.58)	80.69 (↑7.78)	2.76 (↑0.07)
CIFAR-100	BadNets	67.21	10.49	87.43	43.97	23.51	74.89	59.26	28.93	56.66	61.20 (↑1.94)	56.89 (↑27.96)	1.85 (↓54.81)
	InputAware	65.24	1.18	98.62	44.89	20.29	31.65	58.39	12.15	44.08	59.41 (↑1.02)	52.54 (↑32.25)	4.56 (↓27.09)
	TrojanNN	69.90	0.01	99.98	46.51	21.01	77.35	59.10	21.52	50.02	59.81 (↑0.71)	53.39 (↑31.87)	0.13 (↓49.89)
	Blended	70.48	5.42	93.66	45.11	16.28	17.72	57.33	16.19	61.66	60.41 (↑3.08)	49.28 (↑33.00)	8.87 (↓8.85)
	LF	69.66	8.90	89.31	45.93	13.70	63.94	55.08	23.31	50.41	58.67 (↑3.59)	48.72 (↑25.41)	8.73 (↓41.68)
	SIG	69.82	9.14	90.85	45.84	3.90	84.49	58.61	10.38	65.96	59.31 (↑0.70)	38.59 (↑28.21)	5.23 (↓60.73)
	ISSBA	57.64	0.01	99.96	43.82	34.19	0.86	51.02	48.56	0.21	50.29 (↓0.73)	50.63 (↑2.07)	0.44 (↑0.23)
	BPP	64.01	0.93	98.87	43.24	38.10	0.96	57.01	54.11	0.54	58.83 (↑1.82)	53.29 (↓0.82)	0.35 (↓0.19)
	WaNet	64.16	8.38	88.86	48.86	19.65	43.68	54.15	52.70	2.91	55.79 (↑1.64)	53.72 (↑1.02)	1.51 (↓1.40)
Imagenette	BadNets	89.60	1.52	98.47	71.39	18.11	80.09	83.98	20.55	79.16	87.98 (↑4.00)	84.28 (↑63.73)	1.24 (↓77.92)
	InputAware	82.29	4.63	94.68	71.03	10.61	87.11	79.02	16.08	59.51	79.29 (↑0.27)	76.42 (↑60.34)	0.48 (↓59.03)
	TrojanNN	88.48	0.22	99.77	73.13	17.69	81.34	81.65	19.21	68.18	88.16 (↑6.51)	81.06 (↑61.85)	2.76 (↓65.42)
	Blended	88.96	1.10	98.78	67.96	47.11	14.42	79.42	25.07	66.44	86.90 (↑7.48)	67.31 (↑20.20)	7.93 (↓6.49)
	LF	88.22	3.44	96.21	69.47	1.92	97.61	81.89	2.31	96.80	87.12 (↑5.23)	81.03 (↑78.72)	1.24 (↓95.56)
	SIG	88.63	3.33	96.55	72.17	6.73	91.30	85.31	7.36	92.41	87.35 (↑2.04)	74.22 (↑66.86)	8.76 (↓82.54)
	ISSBA	82.83	0.39	99.57	73.10	70.19	5.96	80.16	80.04	1.59	82.11 (↑1.95)	79.05 (↓0.99)	3.55 (↑1.96)
	BPP	83.31	0.99	98.86	77.64	76.53	1.29	81.33	80.19	0.43	81.84 (↑0.51)	80.70 (↑0.51)	0.07 (↓0.36)
	WaNet	83.77	1.87	98.04	75.48	72.45	8.72	80.91	77.45	8.16	82.81 (↑1.90)	81.68 (↑4.23)	7.81 (↓0.35)

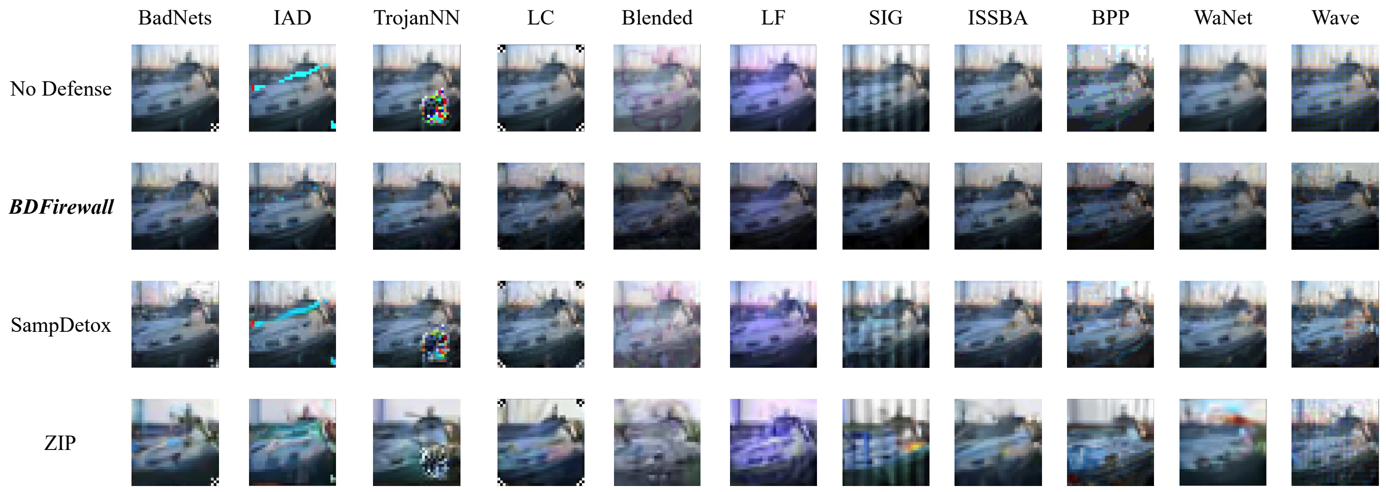

Fig. 4: Visualization of purification results of different algorithms.

still achieves a 24.49% improvement in PA and a 29.23% decrease in ASR compared to the best-performing baseline reported in Table II. This performance highlights the robustness of BDFIREWALL in defending against backdoor attacks in the MLaaS setting.

E. Ablation Study

In this section, we conduct ablation studies to validate the effectiveness of the three stages in BDFIREWALL and to examine the impact of different trigger removal sequences on the final results.

Ablation Study on Defense Stages. BDFIREWALL comprises three distinct components designed to progressively remove high/semi-visibility/low-visibility triggers. To validate their individual contributions, we systematically deactivate each component and evaluate the corresponding defense performance, with the results presented in Table V. When the Stage I component is deactivated, the remaining two components fail to effectively defend against attacks with high-visibility triggers (i.e., BadNets, InputAware, and TrojanNN), causing the ASRs for these attacks to remain high. This observation holds for the other two components as well, indi-

TABLE III: Performance of BDFIREWALL across different model architectures. Here, w/o CA (resp., w/o ASR) represents the clean sample accuracy (resp., ASR of backdoor samples) on compromised model without BDFIREWALL. In addition, we report the changes in CA and PA (resp., ASR) about w/o CA (resp., w/o ASR) after purification by BDFIREWALL in the brackets after the result. CA and PA decrease lower is better, but ASR drops larger is better.

Models	Metrics	BadNets	InputAware	Trojannn	Blended	SIG	BPP	WaNet
PreActResNet18	w/o CA	91.33	90.67	93.44	93.47	90.13	90.69	91.24
	w/o ASR	95.03	98.25	100.00	99.92	99.91	99.77	89.71
	CA↑	87.13 (↓4.20)	86.51 (↓4.16)	88.77 (↓4.67)	88.52 (↓4.95)	87.78 (↓2.35)	87.82 (↓2.87)	85.59 (↓5.65)
	PA↑	84.73 (↓6.60)	82.83 (↓7.84)	84.72 (↓8.72)	74.69 (↓18.78)	74.71 (↓15.42)	82.77 (↓7.92)	84.44 (↓6.80)
	ASR↓	2.38 (↓92.65)	3.47 (↓94.78)	2.25 (↓97.75)	4.01 (↓95.91)	1.95 (↓97.96)	1.27 (↓98.50)	2.77 (↓86.94)
PreActResNet34	w/o CA	92.50	91.05	93.73	93.73	93.76	91.20	90.46
	w/o ASR	100.00	97.13	99.99	99.56	99.88	97.38	96.68
	CA↑	86.69 (↓5.81)	84.88 (↓6.17)	87.96 (↓5.77)	90.12 (↓3.61)	88.67 (↓5.09)	86.70 (↓4.50)	83.12 (↓7.34)
	PA↑	84.11 (↓8.39)	81.05 (↓10.00)	85.29 (↓8.44)	75.62 (↓18.11)	72.33 (↓21.43)	82.61 (↓8.59)	81.54 (↓8.92)
	ASR↓	2.21 (↓97.79)	2.62 (↓94.51)	1.83 (↓98.16)	4.61 (↓94.95)	1.96 (↓97.92)	1.32 (↓96.06)	5.97 (↓90.71)
MobileNet V2	w/o CA	81.84	78.99	82.03	82.16	82.30	82.30	81.26
	w/o ASR	99.99	95.81	99.88	97.29	98.94	99.20	91.15
	CA↑	77.97 (↓3.87)	76.89 (↓2.10)	78.73 (↓3.30)	79.98 (↓2.18)	78.31 (↓3.99)	79.41 (↓2.89)	78.12 (↓3.14)
	PA↑	76.55 (↓5.29)	75.28 (↓3.71)	77.38 (↓4.65)	68.93 (↓13.23)	69.93 (↓12.37)	74.90 (↓7.40)	75.81 (↓5.45)
	ASR↓	2.02 (↓97.97)	1.08 (↓94.73)	3.78 (↓96.10)	4.96 (↓92.33)	1.02 (↓97.92)	2.25 (↓96.95)	2.86 (↓88.29)
ViT	w/o CA	95.85	93.62	95.89	96.40	96.06	97.00	94.70
	w/o ASR	100.00	93.90	99.99	99.89	95.98	99.69	96.46
	CA↑	93.85 (↓2.00)	88.01 (↓5.61)	94.89 (↓1.00)	94.80 (↓1.60)	94.79 (↓1.27)	94.00 (↓3.00)	92.22 (↓2.48)
	PA↑	91.41 (↓4.44)	86.69 (↓6.93)	90.60 (↓5.29)	75.42 (↓20.98)	75.86 (↓20.20)	93.01 (↓3.99)	90.11 (↓4.59)
	ASR↓	0.73 (↓99.27)	4.48 (↓89.42)	1.88 (↓98.11)	6.43 (↓93.46)	6.89 (↓89.09)	1.24 (↓98.45)	1.60 (↓94.86)

TABLE IV: Robustness to Out-of-Distribution Surrogate Data.

Attacks	Default			CIFAR-100		
	CA↑	PA↑	ASR↓	CA↑	PA↑	ASR↓
BadNets	87.13	84.73	2.38	80.56 (↓6.57)	79.34 (↓5.39)	2.61 (↑0.23)
InputAware	86.51	82.83	3.47	80.02 (↓6.49)	77.11 (↓5.72)	3.91 (↑0.44)
TrojanNN	88.77	84.72	2.25	82.75 (↓6.02)	77.57 (↓7.15)	2.61 (↑0.36)
LC	87.63	87.28	1.42	80.09 (↓7.54)	82.51 (↓4.77)	1.31 (↓0.11)
Blended	88.52	74.69	4.01	83.82 (↓4.70)	71.28 (↓3.41)	5.23 (↑1.22)
LF	88.27	79.26	4.41	84.74 (↓3.53)	72.30 (↓6.96)	7.82 (↑3.41)
SIG	87.78	74.71	1.95	80.54 (↓7.24)	70.19 (↓4.52)	8.41 (↑6.46)
ISSBA	91.87	90.72	0.93	87.04 (↓4.83)	87.51 (↓3.21)	0.75 (↓0.18)
BPP	87.82	82.77	1.27	82.50 (↓5.32)	78.12 (↓4.65)	1.02 (↓0.25)
WaNet	85.59	84.44	2.77	82.04 (↓3.55)	80.50 (↓3.94)	1.23 (↓1.54)
WaveAttack	84.57	80.69	2.76	81.92 (↓2.65)	76.78 (↓3.91)	5.23 (↑2.47)

cating that each component is essential for addressing triggers within its designated scope. when Stage II is deactivated and Stage III is activate (i.e., Stage II ✗ and Stage III ✓), the Blended trigger is partially removed. This behavior is analogous to the global detoxification process in SampDetox. Unfortunately, this also demonstrates that global detoxification alone cannot completely remove features of semi-visibility triggers, resulting in residual triggers that can still activate the model's hidden backdoor. Another interesting observation occurs when Stage II is activate and Stage III is deactivated (Stage II ✓ and Stage III ✗): the ASRs for BPP and WaNet slightly decrease. This is because the prediction from f'' cannot be perfectly black (despite the loss term in Eq. 10), which may inadvertently disrupt the subtle features of low-visibility triggers, rendering some of them ineffective. This experiment confirms that each stage of BDFIREWALL fulfills its intended role and that the stages do not exhibit negative interference with one another.

Effectiveness of Different Purification Orders. We now evaluate the impact of different purification orders on the overall defense performance, with the results presented in Table VI. The relative order of Stage I and Stage II (e.g., comparing sequences 1,2,3, 1,3,2, and 2,1,3) has only a minor effect on PA and ASR for high-visibility triggers, resulting in a 1.69% PA loss and a 0.59% ASR increase at worst. As discussed previously, the inaccurate predictions from f'' can slightly corrupt the semantic cues used to locate clean features. However, the placement of Stage III is far more critical. Executing Stage III first leads to a substantial degradation in defense performance for both high-visibility and semi-visibility triggers, causing a 15.43% PA loss and a 13.59% ASR increase for the former, and an 18.94% PA loss and a 23.05% ASR increase for the latter. This side effect also propagates, leading to inaccurate predictions in Stage I and II and consequently a slight PA decrease for LVTs. The reason is that the diffusion process in Stage III disrupts the evidence required by Stage I. Specifically, the diffusion smooths the sharp semantic margins that Stage I relies on to identify segmentation shifts, rendering it ineffective. Furthermore, Stage III intensifies the mixing of clean and backdoor features within the image, making them more difficult to separate. This problem is reflected in the results for SVT-based attacks like Blended and SIG, where the ASR increases by 23.05% and the PA decreases by 18.94%. The results of this ablation study thus confirm our analysis in Sec. IV-C regarding the critical importance of the proposed purification sequence.

F. Inference Time

In this section, we evaluate the inference time of ZIP, SampDetox, and BDFIREWALL, which is a critical factor for deploying black-box defense methods in practical applications. Specifically, we report both sample-level and batch-level inference times in Table VII, where sample-level time is defined as the duration to process a single sample, and batch-level time is the duration to process an entire batch of samples. The results show that BDFIREWALL purifies suspicious samples significantly faster, achieving up to a 111× speedup compared to the best-performing baseline, particularly as the batch size increases. This efficiency is primarily attributed to Stage I

TABLE V: Ablation Study of BDFIREWALL's Components. The symbols ✓ and ✗ denote whether a component is activated or deactivated, respectively.

			BadNets		InputAware		TrojanNN		Blended		SIG		BPP		WaNet	
Stage I	Stage II	Stage III	PA↑	ASR↓	PA↑	ASR↓	PA↑	ASR↓	PA↑	ASR↓	PA↑	ASR↓	PA↑	ASR↓	PA↑	ASR↓
✗	✗	✗	4.66	95.03	1.65	98.25	0.00	100.00	0.07	99.92	0.71	99.27	0.21	99.77	9.75	89.71
✗	✗	✓	4.95	95.03	6.18	93.16	0.04	99.95	39.35	53.14	1.00	98.88	83.01	1.02	84.54	1.35
✗	✓	✗	5.08	94.91	12.20	79.67	0.00	100.00	70.21	26.14	71.84	14.94	20.83	76.04	35.87	59.73
✗	✓	✓	6.24	92.98	18.93	63.43	0.21	99.73	74.43	4.03	74.53	2.71	81.22	1.07	84.80	2.56
✓	✗	✗	90.67	0.92	85.87	5.24	87.78	3.71	0.53	99.43	0.07	99.92	0.28	99.70	9.77	89.68
✓	✗	✓	86.67	2.77	82.16	3.48	84.68	1.60	39.58	52.70	0.98	98.88	82.41	1.04	85.21	1.62
✓	✓	✗	87.04	1.74	83.46	4.17	87.56	3.60	70.20	26.15	71.80	14.94	0.92	99.01	13.03	86.23
✓	✓	✓	84.73	2.38	82.83	3.47	84.72	2.25	74.69	4.01	74.71	1.95	82.77	1.27	84.44	2.77

TABLE VI: Impact of Purification Order on Defense Performance. Performance metrics are compared against the baseline sequence (Stage I → Stage II → Stage III). Colored arrows indicate the performance change relative to the baseline: ↑ signifies an increase in the metric's value, while ↓ signifies a decrease.

	Stage	1, 2, 3	1,3,2	2,1,3	2,3,1	3,1,2	3,2,1
BadNets	PA↑	84.73	84.05 (↓0.68)	83.21 (↓1.52)	55.98 (↓28.75)	62.84 (↓21.89)	60.34 (↓24.39)
	ASR↓	2.38	2.77 (↑0.39)	0.65 (↓1.73)	33.28 (↑30.90)	24.95 (↑22.57)	28.34 (↑25.96)
InputAware	PA↑	82.83	81.77 (↓1.06)	81.04 (↓1.79)	74.93 (↓7.90)	75.32 (↓7.51)	74.73 (↓8.10)
	ASR↓	3.47	2.31 (↓1.16)	3.75 (↑0.28)	8.78 (↑5.31)	8.46 (↑4.99)	9.12 (↑5.65)
TrojanNN	PA↑	84.72	82.53 (↓2.19)	81.84 (↓2.88)	70.80 (↓13.92)	71.88 (↓12.84)	71.18 (↓13.54)
	ASR↓	2.25	1.63 (↓0.62)	1.56 (↓0.69)	11.68 (↑9.43)	10.66 (↑8.41)	11.35 (↑9.10)
Blended	PA↑	74.69	52.54 (↓22.15)	74.08 (↓0.61)	73.16 (↓1.53)	54.74 (↓19.95)	54.35 (↓20.34)
	ASR↓	4.01	26.32 (↑22.31)	4.00 (↓0.01)	3.29 (↓0.72)	26.38 (↑22.37)	26.21 (↑22.20)
SIG	PA↑	74.71	57.74 (↓16.97)	75.33 (↑0.62)	74.31 (↓0.40)	57.55 (↓17.16)	57.64 (↓17.07)
	ASR↓	1.95	26.00 (↑24.05)	0.33 (↓1.62)	4.84 (↑2.89)	25.93 (↑23.98)	25.35 (↑23.40)
BPP	PA↑	82.77	81.08 (↓1.69)	79.68 (↓3.09)	80.58 (↓2.19)	81.28 (↓1.49)	81.66 (↓1.11)
	ASR↓	1.27	1.17 (↓0.10)	1.13 (↓0.14)	1.02 (↓0.25)	1.04 (↓0.23)	0.95 (↓0.32)
WaNet	PA↑	84.44	84.18 (↓0.26)	82.85 (↓1.59)	83.42 (↓1.02)	84.11 (↓0.33)	84.30 (↓0.14)
	ASR↓	2.77	1.47 (↓1.30)	1.48 (↓1.29)	1.45 (↓1.32)	1.43 (↓1.34)	1.10 (↓1.67)

TABLE VII: Inference Time (s)

	ZIP	SampDetox	BDFIREWALL
Sample-Level	7.49	0.40	0.12 (↑∼3×)
Batch-Level (64)	39.68	118.77	0.65 (↑∼61×)
Batch-Level (128)	89.59	238.48	0.81 (↑∼111×)

and Stage II of BDFIREWALL, which effectively remove suspected triggers with one-step inference. In contrast, both baseline methods require multiple reverse diffusion processes to eliminate backdoor triggers. For instance, removing a high-visibility trigger requires SampDetox nearly 120 diffusion steps, even under ideal conditions. Therefore, DDPM-based methods inevitably cause inference delays due to their high computational overhead, making it challenging for users to obtain prediction results in a timely manner. We note that while ZIP suffers from a tremendous inference delay for a single sample, it performs well at the batch level. This is because we follow its official implementation, which batches 64 samples into a single 256×256 input for the diffusion model, thereby saving considerable time. Compared with SampDetox, BDFIREWALL achieves an even greater speedup of up to 294×. This highlights the superior expeditiousness of BDFIREWALL across various inference scales relative to other DDPM-based methods. Additionally, we discuss the use of DDIM to further accelerate inference for BDFIREWALL in Sec. VI-B.

VI. DISCUSSION

This paper presented BDFIREWALL, a novel approach for black-box backdoor defense in MLaaS environments. In this section, we discuss its primary limitations and the crucial balance between performance and computational overhead.

A. *Limitations of* BDFIREWALL

In the previous sections, we conducted extensive experiments across various datasets and models, demonstrating that the proposed BDFIREWALL can effectively and efficiently defend against backdoor attacks in black-box settings. Despite these successes, BDFIREWALL still has limitations in handling certain types of triggers. The core mechanism of BDFIREWALL, for defending against both HVT and SVT, relies on a key observation: the semantic differentiability between clean features and trigger features. Consequently, when triggers are semantically inseparable from the clean features, as seen in so-called semantic backdoor attacks, BDFIREWALL is unable to remove them effectively.

To investigate this limitation, we conducted experiments on three representative attacks: PhysicalBA [55], REFool [56], and the Semantic Backdoor attack [57]. Figure 5 presents the defense performance of BDFIREWALL against the aforementioned attacks. Surprisingly, BDFIREWALL defended against PhysicalBA and REFool with satisfactory PAs and ASRs, a result contrary to our initial hypothesis. Upon re-examining the backdoor generation process for PhysicalBA and REFool, we found that although they are categorized as semantic

TABLE VIII: Performance with DDIM. The result in brackets is a comparison between variations and BDFIREWALL.

		BadNets	InputAware	TrojanNN	Blend	SIG	BPP	WaNet
BDFIREWALL	Time↓	0.65	0.65	0.65	0.65	0.65	0.65	0.64
	PA↑	84.73	82.83	84.72	74.69	74.71	82.77	84.44
	ASR↓	2.38	3.47	2.25	4.01	1.95	1.27	2.77
BDFIREWALL-5	Time↓	0.29(∼2.24×)	0.29(∼2.24×)	0.29(∼2.24×)	0.29(∼2.27×)	0.29(∼2.24×)	0.29(∼2.27×)	0.29(∼2.24×)
	PA↑	81.78(↓2.95)	79.87(↓2.96)	81.02(↓3.70)	61.83(↓12.86)	55.08(↓19.63)	72.68(↓10.09)	76.15(↓8.29)
	ASR↓	0.53(↓1.85)	2.32(↓1.15)	1.05(↓1.20)	8.12(↑4.11)	6.42(↑4.47)	1.23(↓0.04)	1.04(↓1.73)
BDFIREWALL-1	Time↓	0.08(∼8.13×)	0.08(∼8.13×)	0.07(∼8.90×)	0.08(∼8.13×)	0.08(∼8.44×)	0.08(∼8.55×)	0.08(∼8.31×)
	PA↑	44.15(↓40.58)	47.51(↓35.32)	41.70(↓43.02)	19.08(↓55.61)	28.67(↓46.04)	56.30(↓26.47)	46.97(↓37.47)
	ASR↓	0.25(↓2.13)	2.50(↓0.97)	0.62(↓1.63)	18.05(↑14.04)	18.21(↑16.26)	18.22(↑16.95)	0.21(↓2.56)
BDFIREWALL-0	Time↓	0.02(∼32.50×)	0.02(∼30.95×)	0.02(∼30.95×)	0.02(∼30.95×)	0.02(∼30.95×)	0.02(∼30.95×)	0.02(∼32.00×)
	PA↑	29.93(↓54.80)	37.53(↓45.30)	27.93(↓56.79)	17.28(↓57.41)	15.47(↓59.24)	50.31(↓32.46)	45.94(↓38.50)
	ASR↓	0.17(↓2.21)	0.93(↓2.54)	0.27(↓1.98)	28.97(↑24.96)	53.26(↑51.31)	29.73(↑28.46)	0.10(↓2.67)

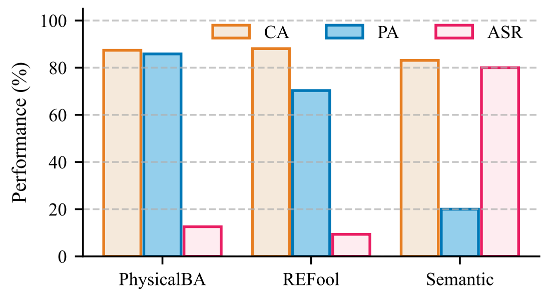

Fig. 5: Physical Backdoor and Semantic Backdoor

attacks, the triggers they apply are not intrinsic to the original clean features. This means the backdoor triggers still exhibit semantic differences from the clean features of the input images. Such semantic differences allow BDFIREWALL to successfully identify and neutralize them.

However, BDFIREWALL fails to defend against the Semantic Backdoor attack [57], which leverages green cars as poisoned samples to induce the backdoor. In this attack, the backdoored model is trained to misclassify any green car to an adversary-specified label. Since the backdoor trigger (the color green) is an intrinsic feature of the benign samples (cars), our key observation regarding semantic differences no longer holds, causing BDFIREWALL to fail. Fortunately, the applicability of such attacks is limited, as they require a specific subclass of samples for poisoning, unlike more general attacks that can implant triggers into arbitrary images. We therefore conclude that while BDFIREWALL has this inherent limitation, its practical impact is confined to a narrow class of attacks, leaving the versatility of BDFIREWALL intact for most real-world scenarios.

B. Balancing timeliness and effectiveness

Diffusion models, with their outstanding performance in high-quality image generation and restoration, are becoming a core approach in generative artificial intelligence. Consequently, their powerful image restoration capabilities have been leveraged in recent research to purify backdoored samples, enabling backdoor defense in black-box scenarios. In this paper, we adopt this line of work, defending against low-visibility triggers by first adding lightweight noise to suspicious inputs and then applying a DDPM-based method to denoise them. Although effective, a critical problem of DDPM-based methods still lingers: *their significant computational overhead.* As dictated by Eq. 16, the prediction at step $t-1$ depends on the state at step t, meaning DDPM-based methods must execute the full sequence of predefined time steps iteratively, resulting in substantial computational costs. Although our method employs only 20 time steps and achieves an inference acceleration of up to 294× compared to SampDetox, a state-of-the-art DDPM-based method, its latency is still significant—approximately 30× slower than vanilla inference. This raises a critical question: *how can we further accelerate the purification process?*

One promising solution is to apply Denoising Diffusion Implicit Models (DDIM) [58], which accelerate image generation through a non-Markovian process. To explore this trade-off, we introduce DDIM into our framework and propose three variants: BDFIREWALL-5, BDFIREWALL-1, and BDFIREWALL-0. Here, the number in BDFIREWALL-5 and BDFIREWALL-1 represents the total number of DDIM steps, while BDFIREWALL-0 represents a baseline where we only add noise to suspicious inputs without any denoising. As reported in Tab. VIII, the results reveal that while DDIM significantly accelerates the purification process, it introduces a step-dependent performance degradation, particularly in PA, which shows a clear downward trend as the number of steps decreases. Thus, how to balance the trade-off between purification efficiency and defense effectiveness remains a critical open question. We believe this direction warrants further investigation from the research community.

VII. CONCLUSION

This paper introduced BDFIREWALL, a progressive black-box backdoor defense framework based on a refined categorization of triggers: Fully-Visible Triggers (FVT), Semi-Visible Triggers (SVT), and Low-Visibility Triggers (LVT) according to their impact to the patch area. Specifically, we first capitalize the clear semantic difference of HVT attaching to identify and reconstruct trigger regions, thereby neutralizing the threat while recovering the original content. Then, we model SVT patched inputs as distinct trigger patterns corrupted by clean features and employ a denoising process to precisely reconstruct and subsequently eliminate the trigger. In order to further eliminate the LVT, we leverage their inherent fragility to noise by perturbing suspicious inputs to destroy them and then applying DDPM to purify them within a few diffusion steps. Extensive experiments demonstrate that BDFIREWALL significantly outperforms state-of-the-art black-box defenses. Compared to the best performance of them, BDFIREWALL achieves an average of 33.25% ASR drop, improving poisoned sample accuracy by 29.64%, and accelerating inference time by up to 111×.

For future work, we identify two critical research directions. First, developing defenses against semantic backdoors, where triggers are intrinsically part of the clean features, remains a major challenge. Second, further optimizing the trade-off between purification effectiveness and computational latency is crucial for real-world deployment, potentially by exploring non-iterative purification models.

REFERENCES

[1] M. Ribeiro, K. Grolinger, and M. A. Capretz, "Mlaas: Machine learning as a service," in *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*. IEEE, 2015, pp. 896–902.

[2] "Ai and machine learning - google cloud," https://cloud.google.com/products/ai, accessed: 2024-01-30. [Online]. Available: https://cloud.google.com/products/ai

[3] "Azure ai services - microsoft azure," https://azure.microsoft.com/en-us/products/ai-services, accessed: 2024-01-30. [Online]. Available: https://azure.microsoft.com/en-us/products/ai-services

[4] H. Hu, S. Wang, J. Chang, H. Zhong, R. Sun, S. Hao, H. Zhu, and M. Xue, "A duty to forget, a right to be assured? exposing vulnerabilities in machine unlearning services," *arXiv preprint arXiv:2309.08230*, 2023.

[5] Z. Huang, Y. Mao, and S. Zhong, "{UBA-Inf}: Unlearning activated backdoor attack with {Influence-Driven} camouflage," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 4211–4228.

[6] Y. Liu, A. Mondal, A. Chakraborty, M. Zuzak, N. Jacobsen, D. Xing, and A. Srivastava, "A survey on neural trojans," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2020, pp. 33–39.

[7] N. Karim, A. A. Arafat, U. Khalid, Z. Guo, and N. Rahnavard, "Augmented neural fine-tuning for efficient backdoor purification," in *European Conference on Computer Vision*. Springer, 2024, pp. 401–418.

[8] Y. Chen, H. Wu, and J. Zhou, "Progressive poisoned data isolation for training-time backdoor defense," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 10, 2024, pp. 11 425–11 433.

[9] Y. Pu, J. Chen, C. Zhou, Z. Feng, Q. Li, C. Hu, and S. Ji, "Mellivora Capensis: A Backdoor-Free Training Framework on the Poisoned Dataset without Auxiliary Data," Oct. 2024.

[10] X. Gong, Y. Chen, W. Yang, Q. Wang, Y. Gu, H. Huang, and C. Shen, "Redeem Myself: Purifying Backdoors in Deep Learning Models using Self Attention Distillation," in *SP 2023*. San Francisco, CA, USA: IEEE, May 2023, pp. 755–772.

[11] H. Zhang, Y. Bai, Y. Chen, Z. Ma, and W. Xu, "Barbie: Robust backdoor detection based on latent separability," in *NDSS*, 2025.

[12] Y. Shi, M. Du, X. Wu, Z. Guan, J. Sun, and N. Liu, "Black-box Backdoor Defense via Zero-shot Image Purification," in *NeurIPS 2023*, 2023.

[13] Y. Yang, C. Jia, D. Yan, M. Hu, T. Li, X. Xie, X. Wei, and M. Chen, "SampDetox: Black-box Backdoor Defense via Perturbation-based Sample Detoxification," in *NeurIPS 2024*, 2024.

[14] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "BadNets: Evaluating Backdooring Attacks on Deep Neural Networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[15] T. A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3454–3464, 2020.

[16] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc, 2018.

[17] A. Turner, D. Tsipras, and A. Madry, "Label-consistent backdoor attacks," *arXiv preprint arXiv:1912.02771*, 2019.

[18] Y. Zeng, W. Park, Z. M. Mao, and R. Jia, "Rethinking the backdoor attacks' triggers: A frequency perspective," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 16 473–16 481.

[19] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[20] M. Barni, K. Kallas, and B. Tondi, "A new backdoor attack in cnns by training set corruption without label poisoning," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 101–105.

[21] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 16 463–16 472.

[22] Z. Wang, J. Zhai, and S. Ma, "Bppattack: Stealthy and efficient trojan attacks against deep neural networks via image quantization and contrastive adversarial learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 15 074–15 084.

[23] T. A. Nguyen and A. T. Tran, "Wanet - imperceptible warping-based backdoor attack," in *International Conference on Learning Representations*, 2021.

[24] J. Xia, Z. Yue, Y. Zhou, Z. Ling, Y. Shi, X. Wei, and M. Chen, "Waveattack: Asymmetric frequency obfuscation-based backdoor attacks against deep neural networks," *Advances in Neural Information Processing Systems*, 2024.

[25] M. J. Shensa, "The discrete wavelet transform: wedding the a trous and mallat algorithms," *IEEE Transactions on signal processing*, vol. 40, no. 10, pp. 2464–2482, 2002.

[26] X. Mo, Y. Zhang, L. Y. Zhang, W. Luo, N. Sun, S. Hu, S. Gao, and Y. Xiang, "Robust backdoor detection for deep learning via topological evolution dynamics," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 2048–2066.

[27] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting ai trojans using meta neural analysis," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 103–120.

[28] C. Fu, X. Zhang, S. Ji, T. Wang, P. Lin, Y. Feng, and J. Yin, "{FreeEagle}: Detecting complex neural trojans in {Data-Free} cases," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 6399–6416.

[29] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1265–1282.

[30] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 707–723.

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

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

[33] E. Chou, F. Tramer, and G. Pellegrino, "Sentinet: Detecting localized universal attacks against deep learning systems," in *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2020, pp. 48–54.

[34] Z. Xiang, Z. Xiong, and B. Li, "Cbd: A certified backdoor detector based on local dominant probability," *Advances in Neural Information Processing Systems*, vol. 36, pp. 4937–4951, 2023.

[35] T. Sun, L. Pang, C. Chen, and H. Ling, "Mask and restore: Blind backdoor defense at test time with masked autoencoder," *arXiv preprint arXiv:2303.15564*, 2023.

[36] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*. Ieee, 2016, pp. 565–571.

[37] R. Sun, Y. Su, and Q. Wu, "Denet: disentangled embedding network for visible watermark removal," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 2, 2023, pp. 2411–2419.

[38] J. Liang, L. Niu, F. Guo, T. Long, and L. Zhang, "Visible watermark removal via self-calibrated localization and background refinement," in *Proceedings of the 29th ACM international conference on multimedia*, 2021, pp. 4426–4434.

[39] Y. Liu, Z. Zhu, and X. Bai, "Wdnet: Watermark-decomposition network for visible watermark removal," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 3685–3693.

[40] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and*

computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18. Springer, 2015, pp. 234–241.

[41] T. Huang, S. Li, X. Jia, H. Lu, and J. Liu, "Neighbor2neighbor: Self-supervised denoising from single noisy images," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 14 781–14 790.

[42] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," *arXiv preprint arXiv:1803.04189*, 2018.

[43] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[44] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[45] J. Howard, "Imagenette: A smaller subset of 10 easily classified classes from imagenet," March 2019. [Online]. Available: https://github.com/fastai/imagenette

[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer, 2016, pp. 630–645.

[48] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[49] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2020.

[50] B. Wu, H. Chen, M. Zhang, Z. Zhu, S. Wei, D. Yuan, and C. Shen, "Backdoorbench: A comprehensive benchmark of backdoor learning," in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[51] Y. Li, M. Ya, Y. Bai, Y. Jiang, and S.-T. Xia, "Backdoorbox: A python toolbox for backdoor learning," *arXiv preprint arXiv:2302.01762*, 2023.

[52] Repository of zip. [Online]. Available: https://github.com/sycny/ZIP

[53] Repository of sampdetox. [Online]. Available: https://github.com/easywood0204/SampDetox

[54] Repository of guided-diffusion. [Online]. Available: https://github.com/openai/guided-diffusion

[55] Y. Li, T. Zhai, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor attack in the physical world," *arXiv preprint arXiv:2104.02361*, 2021.

[56] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, part X 16*. Springer, 2020, pp. 182–199.

[57] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 2938–2948.

[58] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, 2021.

[59] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019. [Online]. Available: https://arxiv.org/abs/1711.05101

Appendix A
Details of Metrics

- **Clean sample Accuracy (CA)** measures the accuracy of the backdoored model on clean samples after they have been processed by our defense method. A higher CA indicates that our defense preserves the model's utility on benign inputs. It is calculated as:

$$\mathrm{CA} = \frac{1}{|D_{cln}|} \sum_{(x_i, y_i) \in D_{cln}} \mathbb{I}(f(P(x_i)) = y_i), \tag{17}$$

where D_{cln} is the set of clean test samples, (x_i, y_i) is a sample-label pair from D_{cln}, $P(\cdot)$ is the purification function of BDFIREWALL, $f(\cdot)$ is the backdoored model, and $\mathbb{I}(\cdot)$ is the indicator function.

- **Poisoned sample Accuracy (PA)** refers to the accuracy of the backdoored model in predicting the original, ground-truth labels of poisoned samples after they have been purified by BDFIREWALL. This metric helps to evaluate whether the defense can successfully remove the trigger effect and restore the sample's original features. It is defined as:

$$\mathrm{PA} = \frac{1}{|D_{bd}|} \sum_{(x_i, y_i) \in D_{bd}} \mathbb{I}(f(P(x_i)) = y_i), \tag{18}$$

where D_{bd} is the set of poisoned test samples and y_i is the original ground-truth label for the sample x_i.

- **Attack Success Rate (ASR)** is the percentage of purified poisoned samples that are still misclassified as the attacker's target label by the backdoored model. It directly measures the effectiveness of our defense in mitigating the backdoor attack. A lower ASR is desirable. The formula is:

$$\mathrm{ASR} = \frac{1}{|D_{bd}|} \sum_{x_i \in D_{bd}} \mathbb{I}(f(P(x_i)) = y_{target}), \tag{19}$$

where y_{target} is the attacker-specified target label.

Appendix B
Implementation Details of BDFirewall

Model Architecture. In Stages I and II, we leverage three carefully designed U-Net models f'_{remove}, f'_{recons} and f'' to defend against the high-visibility triggers and semi-visibility triggers in black-box settings. In terms of f'_{remove}, it contains an encoder (four convolutions and two max-pooling layers for feature extraction), a decoder (two transposed convolutions and two regular convolutions for mask prediction) and a self-calibration module from SLBR [38] to guide the final prediction. For f'_{recons}, it contains an encoder (three convolutions layers for feature extraction), one decoder (two transposed convolutions and one regular convolution for reconstructing the clean image). For f'', it contains a encoder with three double-convolution layers and three max-pooling layers for feature extraction, a bottleneck with a double-convolution layer for processing the middle feature and three transposed convolution layers for image generation. In Stage III, we employ a DDPM-based diffusion model, adapted from SampDetox, to purify inputs against LVTs.

Training Details. We train f'_{remove}, f'_{recons} and f'' for 200 epochs using the AdamW optimizer [59] with a learning rate of 0.001. In terms of the diffusion model used in Stage III, we directly use the pre-trained model provided by SampDetox for the CIFAR-10 task, and utilize the training code of SampDetox to train new diffusion models for CIFAR-100 and Imagenette.

Surrogate Triggers. Since we cannot access the training data of the model provided by the service provider, we have no way of obtaining information about the triggers. For this reason, we use different surrogate triggers in Stage I and Stage II, according to our observations of different types of triggers.

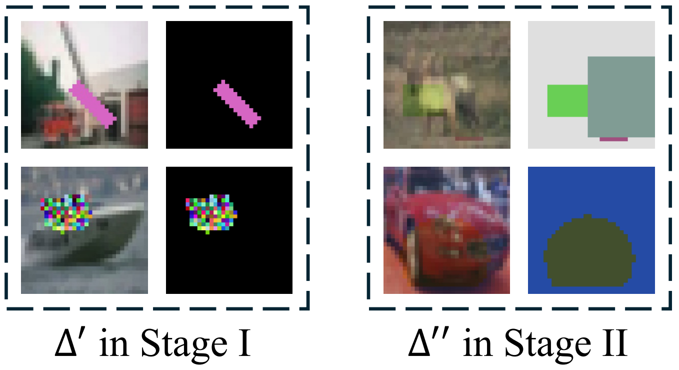

Fig. 6: The visualization of surrogate trigger used in Sec. IV
.

TABLE IX: Performances on Different Trigger Patterns. The number after represents the gap compared to the results in Table II. For CA and PA, higher (i.e., ↑) is better, and ASR is opposite.

Attacks	w/o	CA	PA	ASR
BadNets 5,5, grid	93.31	87.25 (↑0.12)	84.02 (↓0.71)	0.60 (↓1.78)
BadNets, 3, 3, random	92.98	86.31 (↓0.82)	83.83 (↓0.90)	1.20 (↓1.18)
TrojanNN_FireFox	93.26	87.12 (↓1.65)	83.81 (↓0.91)	0.87 (↓1.38)
Blended_Kitty2	93.28	87.30 (↓1.22)	76.65 (↑1.96)	5.35 (↑1.34)
SIG_6_20	93.38	88.86 (↑1.08)	77.41 (↑2.70)	2.76 (↑0.81)
SIG_3_40	93.52	87.45 (↓0.33)	59.12 (↓15.59)	7.67 (↑5.72)

In Stage I, we use randomly generated triangles, squares, and circles as shapes and fill them with various patterns, such as mosaics, random noise, and solid colors to manually create clear semantic distinctions. In Stage II, we combine shapes with different colors and randomly generated backgrounds for them to generate surrogate triggers whose size matches that of the inputs, and then randomly overlay them on the original samples with random transparency ($10\% \sim 40\%$). This process trains f'' to treat the benign image content as ”noise” and effectively separate it from the underlying trigger pattern.

APPENDIX C
GENERALIZATION FOR DIFFERENT TRIGGER PATTERNS.

In this section, we evaluate BDFIREWALL on different triggers to demonstrate the generalization, the corresponding results are presented in table IX where the colorful numbers and arrows represent the performance variations with respect to the CIFAR-10 in Sec. V-B. Specifically, we replace the trigger pattern across three categories of trigger (high-visibility trigger and semi-visibility trigger). We select four representative attacks (BadNets, TrojanNN, Blended, and SIG) and replace their standard triggers with alternative patterns to generate new backdoored models. We do not perform this experiment for Low-Visibility Triggers (LVTs) because their triggers, such as the warping fields in WaNet or imperceptible perturbations in BPP, are not simple, replaceable patterns but rather complex, sample-specific transformations.

The results show that BDFIREWALL's performance is highly consistent across different trigger patterns, exhibiting only minor deviations from the baseline results. On average, the performance changes were minimal: a -0.50% change in CA, a +1.44% change in PA, and a -0.44% change in ASR. The results indicate that the BDFIREWALL is generic for different types of triggers, rather than being limited to specific triggers. It is noteworthy that BDFIREWALL does not achieve particularly satisfactory results in the *SIG_3_40* with its PA decreasing by 15.59% compared to the baseline and its ASR increasing by 5.72%. The reason is that the trigger increases/decreases pixel values by a maximum of *40/255*, which means that, under ideal conditions, removing the backdoor would require a cost of approximately *80/255*, severely damaging the original semantic information of the image. However, despite the significant decrease compared to the baseline, there is still a substantial improvement compared to SampDetox which PA is 4.01% and ASR is 95.67%.

APPENDIX D
VISUALIZED PERFORMANCE COMPARISON.

Figure 8 visualizes the performance data from Tab. II to illustrate the core defense objectives: purifying backdoor samples to be inferred as their ground-truth labels on the backdoor model. We focus on two key metrics: PA and ASR Drop. Note that, the ASR drop, defined as the difference between the baseline ASR ('No Defense') and the post-defense ASR, is used to unify the comparison. Consequently, for both PA and ASR drop, higher values indicate superior performance. As the figure illustrates, our method (BDFIREWALL) consistently occupies a larger area on the chart, signifying its superior overall performance compared to the two other leading methods, especially against HVT- and SVT-based attacks. A detailed analysis of these results is presented in Sec. V-B. To complement these results, we also provide a analysis of the purification process, offering visual evidence that explains the superior performance of BDFIREWALL.

APPENDIX E
DETAILS OF PURIFICATION.

In this section, we visualize the purification processes of ZIP, SampDetox, and BDFIREWALL to provide an intuitive analysis of their respective shortcomings and advantages. To provide a concrete example, we apply these methods to samples from the CIFAR-10 dataset. Specifically, ZIP's procedure includes two intermediate processes—gray-scaling and blurring (implemented via average pooling)—while SampDetox utilizes one intermediate process, which we term 'Location'. In contrast, the process of BDFIREWALL also involves two key intermediate steps: locating the HVT and reconstructing the SVT. Accordingly, we illustrate the intermediate and final purification results of these methods in Figure 8.

ZIP [12] employs gray-scaling and blurring as linear transformations to destroy potential triggers, subsequently using a pre-trained diffusion model to restore the damaged image content. Ideally, the intermediate result of this process should be a trigger-free version of the original input. However, the visualizations in the second and third rows demonstrate that

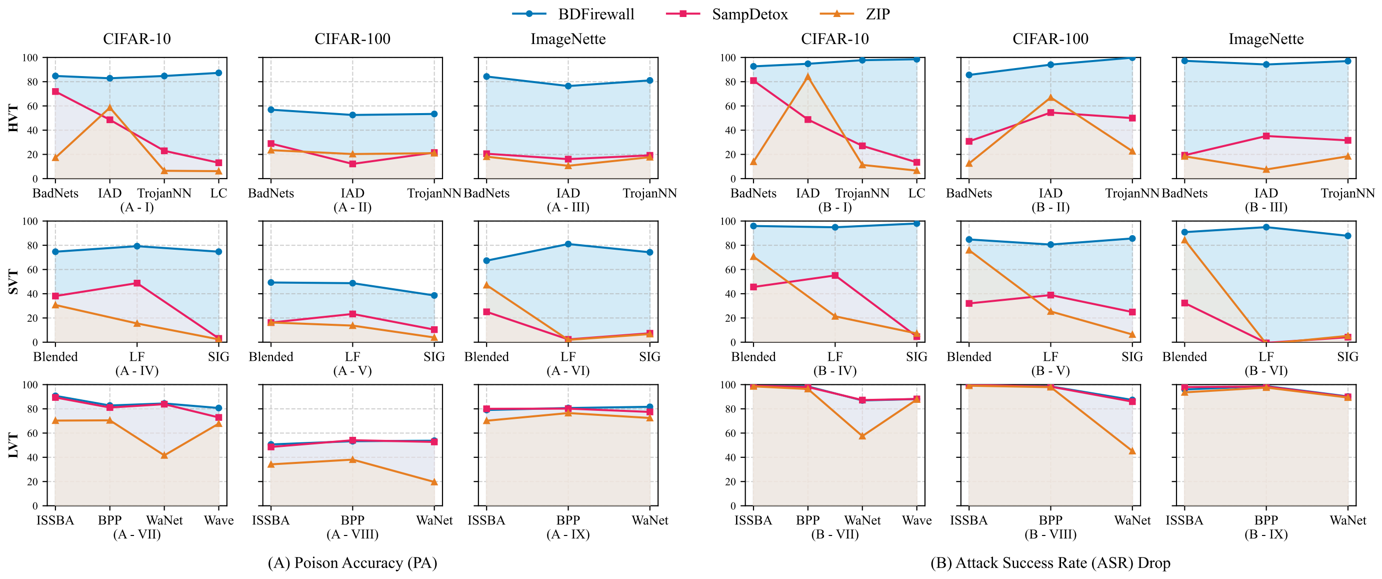

Fig. 7: Visualization of comparison results. We present them in two sub-figures, where sub-figure A reports the PA of three defense methods on different Attacks across three baseline datasets, and sub-figure B presents the ASR drop which is the decreasing value compare which the ASR of *no defense*. For PA and ASR Drop, the higher the better, which means that methods that occupy a larger area in the graph have better defensive effects.

a simple global transformation not only fails to effectively destroy triggers—especially large patterns like TrojanNN and attacks based on SVT, but also inflicts irreversible damage on clean features. This leads to both incomplete trigger removal (e.g., BadNets, TrojanNN, LF, SIG) and collateral damage to the image's benign content (e.g., IAD, Blended, WaNet). The intrinsic reason for ZIP's failure lies in its reliance on a uniform, global purification strategy for all triggers, rather than adapting the destruction method to specific trigger characteristics. Consequently, this approach struggles to simultaneously achieve significant ASR reduction, preserve PA (PA), and maintain CA.

SampDetox [13] leverages a 'Location' operation to identify HVT and purifies them by applying adaptive high-intensity noise to the identified regions. Ideally, this 'Location' operation should precisely identify triggers with minimal error. However, the visualizations reveal that the 'Location' results are far from accurate. For attacks based on SVT and LVT, SampDetox's design implies that they should be handled by a global purification mechanism. Consequently, the 'Location' operation is expected to find no specific trigger region, producing a nearly white output. However, the actual outputs are far from this expectation. Most of them neither located the trigger nor produced the expected white image, with the notable exception of WaNet. As a consequence, balancing Prediction Accuracy (PA) and Attack Success Rate (ASR) for SVT and LVT attacks becomes challenging, a point further discussed in Sec. V-B. The method fails to precisely isolate the trigger and, in turn, erroneously corrupts benign features. Furthermore, this inaccuracy extends to the noise addition of HVT purification, results in the HVT are hard to be purified (e.g., BadNets, IAD, TrojanNN, and LC). The root cause of these suboptimal outcomes is that SampDetox establishes a flawed correlation between model robustness and trigger visibility, which leads to misguided noise application.

Finally, **BDFIREWALL** leverages a progressive black-box framework to defend against the HVTs, SVTs and LVTs in successive order. In Stage I, we introduce a segmentation model to predict the location of HVTs, generating a binary mask denoted as m'. Ideally, the black area of m' should precisely overlap with the HVT regions while remaining blank (i.e., white) for all other conditions. Stage II, in turn, focuses on reconstructing SVTs, producing an output denoted as x''_{recons}. The goal is to reconstruct the SVT pattern with minimal information loss while generating a black output for non-SVT inputs. Since Stage III is a straightforward noise-and-denoise process whose output is the final purified image, we do not visualize its intermediate results separately.

The visualizations demonstrate that Stage I achieves excellent predictive accuracy, significantly outperforming SampDetox's 'Location' module. Specifically, the predicted mask m' accurately covers the trigger-patched areas for HVTs and remains correctly blank for SVT and LVT attacks. One notable exception occurs with the LC attack: its black-and-white grid trigger was expected to produce a solid square mask in m'_{LC}. However, the resulting mask m'_{LC} only captures the white portions of the trigger pattern. This discrepancy stems from our defense's core intuition: discriminating based on semantic differences between clean and poisoned features. Since the background pixels in that region are also black, there is no discernible semantic difference between the black parts of the trigger and the clean area. Consequently, our method only

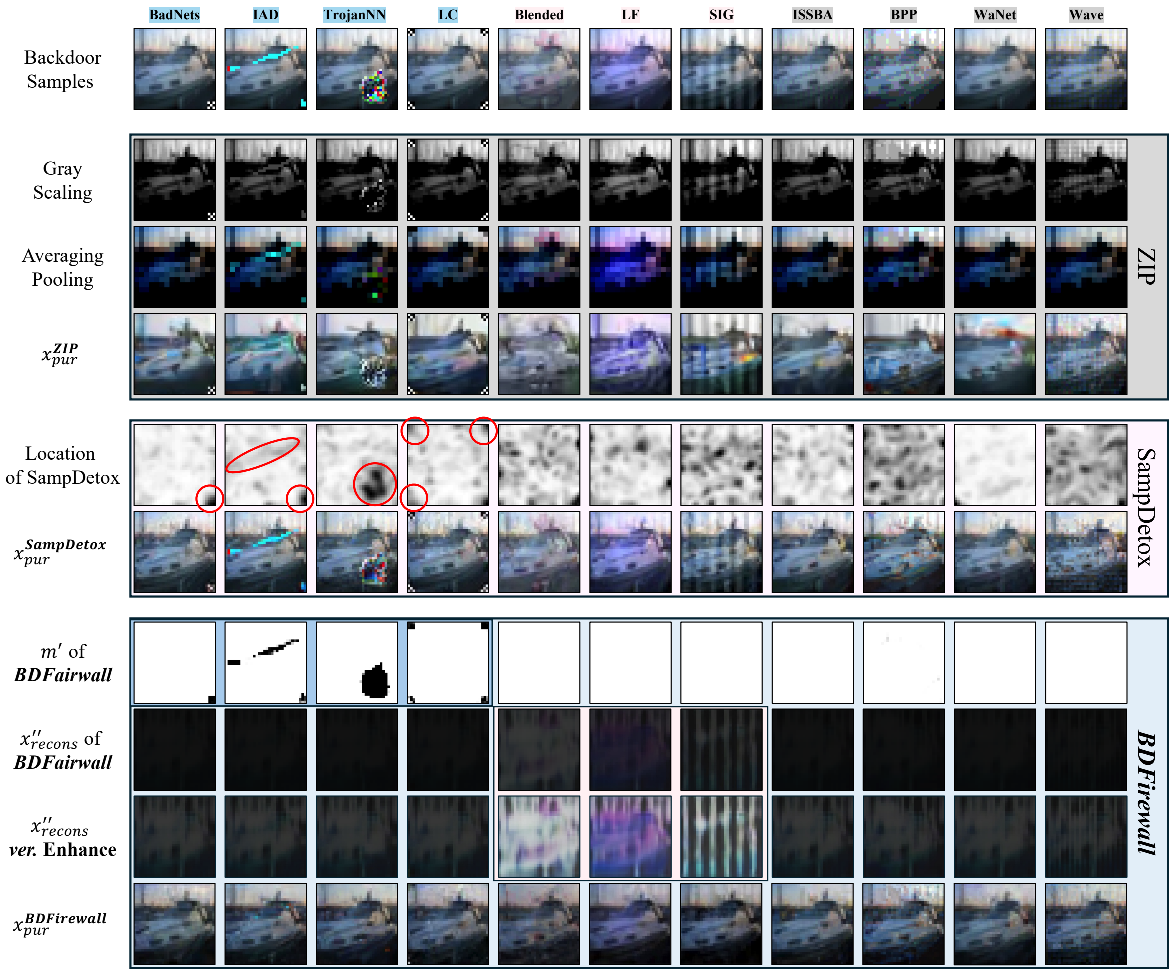

Fig. 8: Visualization of Intermediate Purification Steps. Attacks are color-coded by trigger type: blue backgrounds (HVT) for High-Visibility Trigger, pink (SVT) for Semi-Visible Trigger, and gray (LVT) for Low-Visibility Trigger. Each row visualizes the intermediate and final outputs for a specific defense method: ZIP, SampDetox, and BDFIREWALL.

identifies the white trigger pixels that exhibit a clear semantic deviation. However, this partial detection does not impede trigger removal, as replacing black pixels with other black pixels has no effect on the outcome. This behavior, in fact, reflects that Stage I is operating precisely as designed. In terms of Stage II, we present both the original reconstructed output (x''_{recons}) and an enhanced version (x''_{recons} *ver. Enhance*) created by magnifying the original output threefold for better visibility. Stage II successfully reconstructs various SVTs—such as the Hello Kitty pattern in Blended , the purple region in LF , and the sinusoidal signal in SIG —while keeping the output for other attacks predominantly black. Note that, although the reconstruction model was trained with constraints, it cannot produce a perfect zero output (i.e., pure black), thus naturally introducing a slight bias. This explains why Stage II can cause minor degradation to clean features. However, compared to the baseline methods, our approach not only removes SVTs more effectively but also significantly reduces collateral damage to benign features.

In terms of Stage II, we present the origin results (x''_{recons}) and we also present the corresponding enhanced version (x''_{recons} *ver. Enhance*) by directly magnifying x''_{recons} three times. We can see that Stage II effectively reconstruct the SVTs (the hellokitty of Blended , the purple area of LF and the sinusoidal signal of SIG) and kept black as much as possible for other attacks. Note that, although we restricted f'' during training, since a model cannot output 0 (i.e., pure black) completely, it naturally introduces bias. This explains why the introduction of Stage II introduces some damage to clean features. However, compared with the two baseline methods, our method not only effectively removes SVTs but also significantly reduces damage to clean features.

In summary, we have presented a detailed analysis of the intermediate outputs generated during the purification processes of ZIP, SampDetox, and BDFIREWALL. These visualizations elucidate why prior methods falter, while simultaneously demonstrating how BDFIREWALL can effectively and efficiently neutralize such backdoors. Furthermore, these visual results provide compelling, intuitive evidence that corroborates our quantitative experimental findings, confirming both the accuracy of our analysis and the superiority of our proposed method.