NARCISSUS: A Practical Clean-Label Backdoor Attack with Limited Information

Yi Zeng*
Virginia Tech
Blacksburg, VA, USA
yizeng@vt.edu

Minzhou Pan*
Virginia Tech
Blacksburg, VA, USA
minzhou@vt.edu

Hoang Anh Just
Virginia Tech
Blacksburg, VA, USA
just@vt.edu

Lingjuan Lyu
Sony AI
Tokyo, Japan
Lingjuan.Lv@sony.com

Meikang Qiu
Augusta University
Augusta, GA, USA
qiumeikang@ieee.org

Ruoxi Jia
Virginia Tech
Blacksburg, VA, USA
ruoxijia@vt.edu

ABSTRACT

Backdoor attacks inject maliciously constructed data into a training set of machine learning models so that, at test time, the trained model misclassifies inputs patched with a backdoor trigger as an attacker-desired output. For backdoor attacks to bypass human inspection, it is essential that the injected data appear to be correctly labeled. The attacks with such property are often referred to as "clean-label attacks." The effectiveness of existing clean-label backdoor attacks crucially relies on knowledge about the entire training set. However, in practice, obtaining this knowledge is costly or impossible as training data are often gathered from multiple independent sources (e.g., face images from different users). It remains a question of whether backdoor attacks still present real threats.

In this paper, we provide an affirmative answer to this question by designing an algorithm to mount clean-label backdoor attacks based on the knowledge of samples from the target class and public out-of-distribution data. By inserting maliciously-crafted examples totaling just 0.5% of the target-class data size and 0.05% of the training set size, we can manipulate a model trained on this poisoned dataset to classify test examples from arbitrary classes into the target class when the examples are patched with a backdoor trigger; at the same time, the trained model still maintains good accuracy on typical test examples without the trigger as if it were trained on a clean dataset. Our attack is highly effective across datasets and models, and even when the trigger is injected into the physical world.

We explore the space of defenses and find that NARCISSUS can evade the latest state-of-the-art defenses in their vanilla form or after a simple adaptation. We study the cause of the intriguing effectiveness and find that because the trigger synthesized by our attack contains durable features as persistent as the original semantic features of the target class, any attempt to remove such triggers would inevitably hurt the model accuracy first.

*Yi's work was partially done during the internship at Sony AI. Corresponding **Yi Zeng** or **Ruoxi Jia**. Codes of implementations are opensourced at **Github**: Narcissus. **Yi Zeng** and **Minzhou Pan** contributed equally.

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11.
https://doi.org/10.1145/3576915.3616617

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Artificial intelligence**;

KEYWORDS

AI Security, Backdoor Attack, Clean-Label Attack

ACM Reference Format:
Yi Zeng, Minzhou Pan, Hoang Anh Just, Lingjuan Lyu, Meikang Qiu, and Ruoxi Jia. 2023. NARCISSUS: A Practical Clean-Label Backdoor Attack with Limited Information. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* , 15 pages. https://doi.org/10.1145/3576915.3616617

1 INTRODUCTION

Deep neural networks (DNNs), while performing excellently across numerous tasks, demand vast amounts of data for training as evidenced by various studies [7, 14, 43]. This data-intensive characteristic compels professionals to externalize data gathering, thereby creating vulnerabilities. Specifically, in *backdoor attacks*, where an adversary poisons a dataset so that the learned model will classify any test input that contains a particular trigger pattern as the desired target label and, at the same time, will correctly classify typical inputs that do not contain the trigger.

Conventional backdoor attacks [5, 10] involve choosing a few clean inputs from a non-target class (assuming access to the full training set), applying an *arbitrary* chosen backdoor trigger, relabeling them to the target class, and then adding them to the training set. This ensures the model associates the trigger with the target class. However, a glaring shortcoming of such attacks is that the manipulated data is obviously mislabeled. Consequently, if subjected to human review, these discrepancies would be easily spotted [46].

Recent work has proposed several techniques to enable *clean-label* backdoor attacks, wherein the poisoned inputs and their labels are consistent with human inspectors. One intuitive approach to craft these attacks mirrors standard backdoor techniques, applying triggers solely to the target class. However, since these poisoned inputs are from the target class and already contain some salient natural features indicative of that class, the learned model tends to associate the natural features instead of the backdoor trigger with the target class. This simple idea falls short unless an overwhelming portion of the target class is tainted (for instance, 70% on CIFAR-10 according to our tests in Section 5.2)." The Label-Consistent (LC) attack [46] enhances efficacy by making the original features in

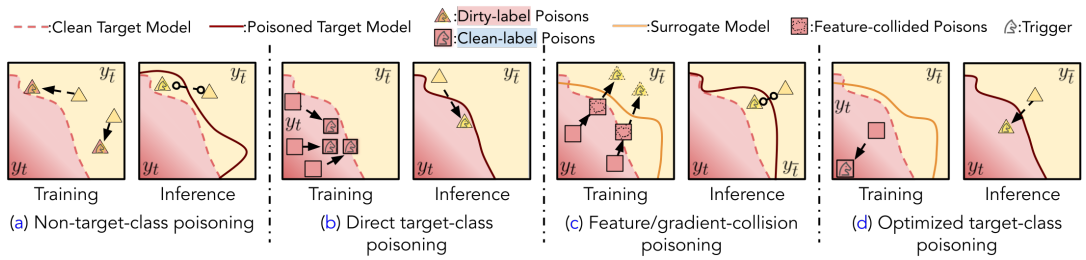

Figure 1: Comparative overview of various backdoor attack methodologies: (a) Non-target-class poisoning [5, 10, 31] employs a trigger on non-target-classes with modified labels; (b) Direct target-class poisoning [59] uses an arbitrary trigger, focusing solely on the target class; (c) Feature/gradient-collision poisoning [38, 41] poisons target class samples using noise directed at poisoned non-target-class samples, relying on a surrogate model trained over the complete distribution; (d) Our concept of optimized class-oriented target-class poisoning synthesizes a trigger based on the surrogate model but only accesses and manipulates the target class.

poisoned samples more challenging to discern, thereby strengthening the association between trigger patterns and the target class. LC introduces two strategies: adding adversarial perturbations and blending features from non-target classes into target class samples using interpolation in a GAN's latent space. Yet, both creating adversarial perturbations and training a GAN necessitate access to samples across all classes

An alternative line of clean-label attacks [38, 41] involves modifying target-class samples to emulate the effect of perturbing non-target class samples. Specifically, the Hidden Trigger Backdoor Attack (HTBA) [38] works by minimizing the gap between altered target-class inputs and those of non-target classes with inserted triggers in the feature space. For HTBA's optimal impact, it requires a pre-trained feature extractor that's trained on all class samples. Conversely, the Sleeper Agent Attack (SAA) [41] looks for modifications in target-class samples, aligning the gradient of these modified samples with the gradient of perturbed non-target class samples, which also hinges on having a model trained on the complete dataset.

Overall, **existing clean-label attacks largely depend on full knowledge of the training data across all classes**. This is only achievable if the attacker provides the entire dataset. However, in real-world scenarios like crowdsourcing, data comes from multiple independent sources, complicating or even nullifying access to all class data. For instance, while training face recognition models, datasets combine images from diverse users. While malicious users can tamper with their own images, altering others' data is impractical. Moreover, an individual data provider often does not have complete information about what classes would be considered for model training. Therefore, it's essential to assess the viability of backdoor attacks when only *partial* class training data is accessible.

Our study seeks to reevaluate the feasibility of backdoor attacks under a more realistic attacker model than previously explored. Given that in backdoor attacks, the perpetrator often targets specific classes, a reasonable base assumption is that they possess limited data from these target classes. We pose the question: *Is a successful clean-label backdoor attack possible if the attacker only has access to training data from the target class?* Such an approach would be cost-effective, eliminating the need to gather examples from a potentially vast or indeterminate number of non-target classes.

We introduce Narcissus, a clean-label backdoor attack that solely depends on target-class data and public out-of-distribution data. A key insight driving our algorithm's design is the inherent flaw in current backdoor trigger design methods: the arbitrary selection of triggers. Methods like LC directly insert a random trigger into the target-class data, while HTBA and SAA indirectly embed this through feature/gradient collision (refer to Figure 1). Given these triggers don't inherently relate to the target class, a substantial poison ratio is needed for them to work effectively. Building upon this insight, we propose to **optimize the trigger pattern in a way that points towards the inside of the target class**, which can be found without the accurate knowledge of other non-target classes.

We extensively evaluate our attack on three standard computer vision datasets and four mainstream model architectures. Focusing on the Tiny-ImageNet results: by altering just 0.05% of the entire data, we managed to misclassify 85.81% of test samples from any class to a chosen target class once they were modified with the backdoor trigger. At the same time, the backdoored model still maintains good accuracy on clean test examples as if it was trained on a clean dataset. In comparison, current clean-label attacks, even with access to the full training data and larger modifications, achieve a mere success rate of 1.72% at the same poisoning level. In essence, **our method outperforms existing ones in effectiveness, stealth, and requires lesser information on the attacker's part**. We also introduce **the first physical world clean-label backdoor attack**, incorporating the trigger directly into the environment. A video-demo of the real-world Narcissus can be viewed here.

We assessed several defenses against our attack method. Notably, widespread defenses like STRIP [8], Neural Cleanse [47], and Fine-pruning [24], as well as state-of-the-art ones like I-BAU [54], MOTH [44], ABL [21], or NONE [48], fail to consistently counter our attack. Although the frequency-based defense [56] can detect our attack due to high-frequency artifacts in the triggers from vanilla Narcissus, this defense becomes ineffective once a low-frequency constraint is applied to our attack's synthesized trigger. Delving into its potent efficacy, we discerned that our attack's trigger features have resilience akin to the training data's semantic features. Removing these triggers would compromise the model's precision, underscoring the urgent need for improved defenses.

Our key contributions include:

- We present a **clean-label backdoor attack that leverages only target-class and public out-of-distribution data**, controlling **0.05%** or even fewer data.

	Standard Attacks	LC [46]	HTBA [38]	SAA [41]	Narcissus (Ours)
Label poisoning	Dirty	Clean	Clean	Clean	Clean
Model-agnostic	○	×	×	○	✓
Train from scratch	✓	✓	×	✓	✓
All-to-one attack	✓	✓	×	×	✓
Works on large D	✓	×	×	×	✓
Physical world	✓	×	×	×	✓
Only requires D_t	×	×	×	×	✓
Low poison ratio	×	×	×	×	✓

Table 1: Summary of the differences between previous backdoor attacks and ours. ○ denotes partially satisfying. D denotes a dataset, and D_t is the subset only containing the target class.

- We compare Narcissus with existing attacks and show that it **significantly outperforms** others despite using less attacker knowledge and less obvious perturbations.
- We demonstrate that by adapting the trigger to real-world conditions, Narcissus enables **first of its kind** successful physical world clean-label attacks.
- We show that several popular choices of defenses and state-of-the-art ones, **cannot reliably mitigate our attack**. We find our triggers exhibit features that are resistant to removal.

2 BACKGROUND AND RELATED WORK

2.1 Supervised Machine Learning

The goal of supervised machine learning is to train a classifier, denoted as $f_\theta : \mathcal{X} \to [k]$, that predicts the label $y \in [k]$ for a given input $x \in \mathcal{X}$. θ represents the classifier's parameters. This learning process has two main phases: training and testing. During training, a learning algorithm gets a training data set, $D = \{(x_i, y_i)\}_{i=1}^N$, containing examples from k classes. The algorithm then searches for model parameters, θ, which reduce the empirical risk:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \mathcal{L}\left(f_\theta\left(x_i\right), y_i\right). \tag{1}$$

For deep neural networks, this empirical risk in relation to θ is non-convex, causing the true minimum hard to find. Therefore, a common approach is seeking a local minimum using stochastic gradient descent [4]. In the testing phase, the trained model, f_{θ^*}, processes test examples to make predictions.

2.2 Backdoor Attacks

In backdoor attacks, the adversary embeds a trigger in the victim model by introducing manipulated examples into the training process [19, 22, 32, 33, 35]. The resulting model, referred to as the poisoned model, produces a specific target label for inputs containing this trigger while maintaining high accuracy for unaltered inputs. Existing attacks can be broadly classified into two categories based on the consistency of features with their labels: *dirty-label attacks* and *clean-label attacks*.

Dirty-label Attacks. Majority existing backdoor attacks [5, 10, 19, 20, 27, 30, 31, 39, 51, 56] belong to this category. In these attacks, clean examples from non-target classes are chosen and applied with the backdoor trigger, and their labels are altered to the target class. Training with this dataset causes the model to associate the trigger with the target label. As these modified inputs retain features inconsistent with the target class, the model easily learns the trigger pattern. Even if the trigger is subtle and does not change the input's meaning, its combination with the altered label makes it seem mislabeled to humans. Consequently, these tainted samples could be easily spotted during human checks [55].

Clean-label Attacks. To improve the stealth of backdoor attacks, recent work has focused on clean-label attacks, in which poisoned inputs and their labels appear consistent to humans. Based on poisoning methodologies, we divide existing clean-label attacks into **direct target-class poisoning** and **feature/gradient-collision poisoning**.

The seminal work in **direct target-class poisoning** is LC [46]. It modifies target class data to obscure the original features, which can be achieved using a GAN to blend features from both target and non-target classes or by introducing adversarial disruptions. Following this, an arbitrarily chosen trigger is embedded in the altered data. A notable difference between LC and our approach is LC's arbitrary selection of backdoor triggers, often requiring a significant poison ratio to embed the trigger-label association. Moreover, LC needs non-target class data to create impactful perturbations.

On the other hand, another line of work, including HTBA [38] and SAA [41], attempts to insert the trigger indirectly. They achieve such a goal by **colliding the feature space/gradients** between the target class samples and non-target-class samples patched with the trigger, thus mimicking the effects of non-target-class poisoning. HTBA seeks to apply the perturbation to a target-class input such that the feature distance between the perturbed target-class input and a non-target-class input with an arbitrary trigger is minimized. By doing this, the decision boundary will place these two points in proximity in the feature space, and as a result, any input with the trigger will likely be classified into the target class. Since HTBA minimizes the feature distance between points from a pair of classes, HTBA only supports *one-to-one* attack, i.e., the trigger can only render the inputs from *one specific* class to be classified into the target class. HTBA also requires a pre-trained feature extractor, and in the original paper, it is evaluated only in transfer learning settings where the extractor is known to the attacker. The compromised model is then refined using this extractor on the poisoned dataset. Contrarily, most other studies, including ours, permit models to be trained from scratch. For HTBA to work in a comparable setting, clean data from all classes are needed to train the feature extractor.

SAA formulates a bi-level optimization for the data poisoning problem. Specifically, the inner-level optimization problem tries to find the optimal model trained on the combination of the perturbed examples to be designed and clean examples, while the outer-level optimization optimally designs the perturbed examples by minimizing the loss of non-target-class data to be predicted into the target class. While this problem formulation is general and seems to support all-to-one attacks, the actual algorithm to solve the problem relies on choosing a pair of non-target-class (patched with an arbitrary predefined trigger and labeled as the target class) and a perturbed target-class example and then aligning the gradient between them. Hence, similar to HTBA, SAA also only allows one-to-one attacks. By contrast, the other clean-label works in the literature and our work allow *all-to-one* attack, i.e., inputs from *any* non-target classes are classified into the target classes when patched with the trigger. Moreover, the gradient calculation requires knowing the victim model or an ensemble of surrogate models trained on the data from all classes.

Compared to all existing clean-label attacks (LC, HTBA, and SAA), the primary advantage of our approach is that it only needs the knowledge of training data from the target class, D_t. Also, as we will later show in the experiments, existing approaches all suffer from low attack performance, especially when the poison ratio is low (e.g., 0.024% on the PubFig). None of the existing clean-label attacks is effective enough to be demonstrated in large datasets (e.g., Tiny-ImageNet) and the physical world. Our work significantly boosts the state-of-the-art clean-label attack efficacy and enables the first physical world attack. In particular, our attack supports the all-to-one setting compared to HTBA and SAA. The full comparison between our approach and the existing ones is summarized in Table 1.

3 THREAT MODEL

We examine a scenario where a victim aggregates data from various sources to train a machine-learning model termed the 'victim model.' In this setup, an adversary can influence a subset of the dataset by supplying data. The adversary's objective is to sabotage the victim model, f_θ, so it misclassifies test inputs embedded with a specific trigger, δ, into a predetermined class, t, without affecting its performance on untampered inputs. To achieve this, they alter some of the data they provide. To ensure these modifications go unnoticed, we, in line with existing studies [36, 38, 41, 46, 55], assume that the changes have an l_p-norm constraint and that they are clean-label: the altered data and its labels appear genuine to humans.

Attacker Knowledge. Since the adversary gets to pick the target class, we assume that it knows some target class examples, D_t, from the target class, t. Throughout our experiments, we consider the available target-class examples are randomly sampled from the underlying data distribution. In addition, we assume the attacker knows some *general* information about the victim's learning task. Take face recognition as an example: we assume the attacker knows that the victim will train a face recognition classifier, but we do *not* assume that the attacker knows the identities (or classes) to be classified. As a result, the adversary can collect some extra samples related to the learning task to facilitate the attack. Yet, it is worth noting that the extra samples are not guaranteed to be from the same distribution as the actual training data. Consider the face recognition example again. The attacker may scrape some extra face images from the internet, but the face images could contain arbitrary identities (or classes) and may not be actually used by the victim. Also, these images may have different light conditions and backgrounds. Hence, we will refer to these extra examples of the learning task as *public out-of-distribution (POOD) examples. Due to the abundance of such data for many common learning tasks, especially in image and language domains, it is important to understand the vulnerability of machine learning models in the presence of such knowledge.* In this paper, we assume the attacker has access to some POOD examples, but the POOD examples have strict *class separation* from the original training data. To perform our experiments, we use *different* datasets to sample POOD examples and training data (see Table 6).

Comparison to Past Threat Models. Our assumption about attacker knowledge is systematically weaker (more difficult for the attackers) than past literature and thus more realistic (see Table 2).

	Knowledge on the Target Dataset	Capability on Perturbing the Target Dataset	Knowledge on the Target Model
Dirty-Labels [10, 26, 31, 56]	**Full access** to the training set	Can **manipulate any** sample in the training set	**Not required** to know the details
Clean-Labels [38, 41, 46]	**Full access** to the training set	Can **manipulate only** target-class	**Requires details** to achieve the best
Narcissus (Ours)	**Only access** to the target-class	Can **manipulate only** target-class	**Not required** to know the details

Table 2: Comparison of threat models. This work aims to raise awareness of a more effective and flexible attack case that can be conducted under a more restricted and realistic threat model.

In particular, our threat model requires only the target-class samples. Yet, existing backdoor attacks (whether clean-label or dirty-label) need access to the entire training dataset. Compared to the threat models used in existing clean-label backdoor attacks, our threat model requires less knowledge about non-target-class samples and target models.

Grounding the Threat Model in the Real-World. In real-world attack scenarios involving crowdsourcing [1], users often contribute partial classes to a training dataset, such as when companies collect facial images to build face recognition models. Malicious users may upload backdoored images with hidden triggers. Existing backdoor attacks require access to the entire training set, making them challenging to be practical due to the decentralized nature of crowdsourced data contributions. Our threat model does not assume the knowledge of other clients' data; instead, we only rely on target-class samples (the attacker chooses the target class, and it is reasonable to assume the attacker has knowledge about it) as well as a generic pre-trained model relevant to the learning task. Compared to the existing threat model, our work lifts many assumptions and thus takes a step towards making backdoor attacks more practical.

Metrics. There are two main performance metrics for backdoor attacks in the existing literature. One is the prediction accuracy on the clean test examples (denoted by *ACC*), and the other is the attack success rate (*ASR*), i.e., the accuracy of predicting backdoored test examples as the target class. Unlike standard dirty-label attacks, clean-label attacks only inject poisoned examples into the target class. Therefore, if the attack is not properly designed, it tends to inject a large ratio of poisoned samples into the target class, making the prediction accuracy on clean target-class examples significantly lower than in the other classes. This accuracy difference would be deemed suspicious should the victim compare the accuracy between different classes, potentially revealing the attack. Indeed, our experiments found that existing clean-label attacks all suffer from this drawback (see Section 5.2). Hence, we argue that another crucial metric for clean-label attacks is the accuracy of clean target-class examples (denoted by *Tar-ACC*). Collectively, a successful clean-label backdoor attack should obtain high ACC, high Tar-ACC, and high ASR for a given *poison ratio* and a given perturbation constraint.

Other than these standard attack metrics, we also incorporate three metrics to quantify the stealthiness of an attack under human inspection. The first is standard in the literature, and the last two are developed by our work. (1) *l_∞-norm* measures the trigger magnitude at **pixel-level** in terms of l_∞-norm and is the most commonly used to measure the perceptibility of an attack; (2) *LPIPS* is a common metric in the computer vision literature to measure the semantic or perceptual similarity between two images [58]. We incorporate LPIPS in the backdoor domain to depict the human *perceptual* similarity between the clean and the backdoored images;

and (3) *MinPoi-k*represents the minimum poisons needed to achieve a given k% ASR. For example, "MinPoi-90 = 3" means it requires at least 3 samples to be poisoned for an attack to achieve 90% ASR. We believe this is an important metric for training-time attacks. Given large modern training sets, human inspectors cannot feasibly review every point. They will likely examine a subset. A lower MinPoi-k means greater dataset-level stealthiness, making it harder for inspectors to spot all poisoned examples and implying a reduced attack cost, as fewer instances need modification to achieve the desired attack impact.

4 METHODOLOGY

4.1 Problem Formulation

The key limitation of existing clean-label attacks (both direct target-class poisoning and feature/gradient-collision poisoning) lies in the misalignment of the trigger and the target class. The arbitrariness of choosing the trigger makes them particularly susceptible to low poison ratios and mismatches between the target and surrogate models. With this insight, the high-level goal of our approach is to optimize the trigger toward a better alignment with the target class.

Towards this end, we formulate an optimization to design the trigger. Let us first assume that the attacker has access to the victim model trained on the target dataset, referred to as the *oracle model*, $f_{\theta_{\text{orc}}}$. Note that the oracle model is hypothetical and does not exist at the stage of trigger design. We make this assumption to explain our approach's idea, and later we will show how to remove it.

Given $f_{\theta_{\text{orc}}}$ and the knowledge of target-class examples, D_t, we would like to find the trigger that turns each target-class example to be predicted as the target class with higher confidence. Formally, we aim to solve: $\delta^* = \arg\min_{\delta\in\Delta} \sum_{(x,t)\in D_t} \mathcal{L}\left(f_{\theta_{\text{orc}}}(x+\delta), t\right)$, where Δ represents the set of allowable trigger designs. $\mathcal{L}\left(f_{\theta_{\text{orc}}}(x+\delta), t\right)$ calculates the loss of predicting $x+\delta$ into the target class t. Intuitively, δ^* can be thought of as the most robust, representative feature of the target class, as adding it into any inputs would maximize the chance of them being predicted as the target class *universally*. Naturally, we would expect that δ^* as a trigger should activate the target class more effectively than some arbitrary triggers.

Now, the question is how to remove the assumption of knowing $f_{\theta_{\text{orc}}}$. Inspired by existing blackbox attack techniques [6, 40, 41], we adopt a *surrogate* model $f_{\theta_{\text{sur}}}$ constructed from the available POOD examples and target-class examples in place of $f_{\theta_{\text{orc}}}$. Hence, the problem that we actually solve in the implementation is:

$$\delta^* = \arg\min_{\delta\in\Delta} \sum_{(x,t)\in D_t} \mathcal{L}\left(f_{\theta_{\text{sur}}}(x+\delta), t\right). \tag{2}$$

An intriguing property of δ^* that we found in our experiments is that (see Table 5), unlike many other existing attacks, this perturbation is remarkably robust to the mismatch between the actual victim model architecture and the surrogate model architecture, as well as the mismatch between their training data. This can be explained by the inward-pointing nature of δ^*, as illustrated in Figure 1(d). Specifically, δ^* increases the confidence of all target-class examples and thus represents a direction that points towards the inside of the target class. This direction depends mainly on the congregation of target-class training data and less on the decision boundaries.

4.2 Attack Workflow

Now, we present the detailed workflow of our attack Narcissus. In particular, we will elaborate on how to leverage the POOD and target-class examples to produce an effective surrogate model and how to efficiently solve the optimization problem in equation (2).

Step 1: Poi-warm-up. This step seeks to create a surrogate model that extracts class features, facilitating the creation of robust feature noise for the target class. While abundant POOD samples might be available, target class samples might be limited. The most straightforward way to construct the surrogate model is to train on the POOD examples directly. Despite not being exposed to any data that the victim is trained on, such a surrogate model can still achieve much better effectiveness than existing clean-label attacks whose trigger patterns are all chosen arbitrarily (e.g., it achieves 85.88% ASR on CIFAR-10, while other clean-label attacks' highest ASR is 3.21%). However, we find that it takes many iterations for the optimization to acquire an effective trigger as POOD examples do not contain the target class. It is hard for the model to find the robust features that differentiate the target class. So, we propose first training the surrogate model on the POOD examples and then fine-tuning the model on the target-class examples for certain epochs. The idea behind this two-pronged training approach is that training on POOD examples allows the surrogate model to acquire robust low-level features that are generally useful for a given learning task. Then, the fine-tuning step enables a quick adaptation to capture the discerning features of the target class.

A natural question might be: Why not directly train on the combination of the POOD and target-class examples? Compared to this one-step alternative, the proposed approach achieves similar attack performance (Table 12, Appendix) but is much more efficient and flexible when the attacker wants to dynamically choose new classes as the target and attack them. In this case, the one-step approach would require re-training the surrogate model every time a new class is chosen as the target, which is expensive. In contrast, with the proposed two-pronged approach, one can just train on the POOD examples once and fine-tune the model with any new-class examples.

Algorithm 1: Trigger Generation Algorithm

Input: $f_{\theta_{\text{sur}}}$ (Surrogate model);
D_t (target class data samples);
Δ (allowable set of trigger patterns);
Output: δ_I (the Narcissus trigger);
Parameters: I (total iteration number);
$\alpha > 0$ (step size);

$\delta_0 \leftarrow \mathbf{0}^{1\times d}$;
for *each iteration* $i \in (1, I-1)$ **do**
  $\delta_{i+1} \leftarrow \delta_i - \alpha \sum_{(x,t)\in D_t} \nabla_\delta \mathcal{L}\left(f_{\theta_{\text{sur}}}(x+\delta), t\right)$;
  $\delta_{i+1} \leftarrow \text{Proj}_\Delta(\delta_{i+1})$;
return δ_I

Step 2: Trigger Generation. In this step, we synthesize the trigger by solving (2) via mini-batch stochastic gradient descent. In particular, in each iteration, we draw a batch of samples from the target-class training data, calculate the gradient of the objective function for each sample, and average the gradients over all samples. We then update the trigger with the averaged gradient and

project the trigger back to the allowable set Δ. In the non-adaptive attack setting in our evaluation, we follow the existing literature and let Δ be a l_∞-norm ball, i.e., $\Delta = \{\delta : \|\delta\|_\infty \leq \epsilon\}$. Projection to a l_∞-norm can be done by just clipping each dimension of δ into $[-\epsilon, +\epsilon]$. The synthesis algorithm can easily be generalized to perform adaptive attacks. For instance, to bypass the defense that identifies the backdoor examples based on their high-frequency artifacts, we can set Δ as the set of low-frequency perturbations, and projecting onto Δ can be done by passing the perturbation through a low-pass filter. The full details of the trigger generation step are provided in Algorithm 1.

Step 3: Trigger-Insertion. After we acquire the synthesized trigger, we randomly select a small portion of the target-class examples and apply the backdoor trigger to the input features. Then, the poisoned target-class data will be supplied to the victim.

Step 4: Test Query Manipulation. To attack a given test input, x_{test}, the attacker magnifies the trigger by a certain scale (e.g., 3×), inserts the magnified trigger into x_{test}, uses it for the victim model trained on the poisoned dataset. Note that the test-stage trigger magnification has been explored in previous work [46]. Similar to previous observations, we find that it can boost the attack performance compared to applying the original trigger to the test example. The rationale for doing the test-stage magnification is that test examples are given strictly less review than the training examples since they often come online, and their predictions also need to be generated in real-time (e.g., the autonomous car's perception system). It is worth noting that even after magnification, the norm of our synthesized trigger is still less than the existing triggers while being more effective. Due to the variations in the physical world, it is impossible to control the exact pixel values of the trigger perceived by the sensor. So we omit the trigger magnification for the physical world attack.

Remark: Connections to Universal Adversarial Attacks and Natural Backdoors. Universal Adversarial Attacks (UAAs) [57] and Natural Backdoors (NBs) [45] aim to attack machine learning models at test time, which skips the process of poisoning. The effect of a successful UAA or NB is similar to the standard backdoor attacks, where using the same synthesized noise (a.k.a. Universal Adversarial Perturbation, or UAP in UAA literature, or the NB trigger in NB literature), one can mislead the target model's prediction at any input from any non-target class to the target class. Finding an UAP/NB trigger, $\tilde{\delta}^*$ is often cast as an optimization problem, where $\tilde{\delta}^* = \arg\min_{\delta\in\Delta} \sum_{(x,y)\notin D_t} \mathcal{L}\left(f_{\theta_{\text{orc}}}(x+\delta), t\right)$. Recall our trigger synthesis optimization, which aims to find a backdoor trigger, δ^*, such that $\delta^* = \arg\min_{\delta\in\Delta} \sum_{(x,t)\in D_t} \mathcal{L}\left(f_{\theta_{\text{sur}}}(x+\delta), t\right)$. The key difference between ours and UAA/NB is three-fold. **(1)** The optimization goal of a UAA/NB is to find a noise that helps samples from non-target classes to **cross the decision boundary** while Narcissus is to find an **inward-pointing noise** that better represents the target class so that a model trained over the noise-poisoned dataset can memorize such a noise as a robust feature for the target class. **(2)** The different optimization goals of ours and UAA/NB thus lead to different attacker knowledge requirements. Specifically, a targeted UAA/NB requires **in-distribution samples from non-targeted classes** to synthesize $\tilde{\delta}^*$. Normally, **white-box access to the model trained over the original training set** is also necessary for a successful UAA/NB. By contrast, Narcissus only requires **samples from the target class**. **(3)** Performance-wise, UAP or NB trigger's attack efficacy is limited and varies a lot across different choices of target classes and different datasets, whereas Narcissus is consistently effective (see Figure 9 comparing UAA, NB, and Narcissus).

Dataset	CIFAR-10 [16]	PubFig [17]	Tiny-ImageNet [18]
# of Classes	10	83	200
Input Shape	(3,32,32)	(3,224,224)	(3,64,64)
Poison Ratio (%)	0.05 (25/50,000)	0.024 (3/12,454)	0.05 (50/100,000)
Target Class	2 (Bird)	60 (Miley Cyrus)	2 (Bullfrog)
Epochs	200	60	200
Optimizer	SGD [37]	RAdam [25]	SGD [37]
Augmentation*	[Crop, H-Flip]	[Crop, Rotation]	[Crop, Rotation, H-Flip]

Table 3: Hyperparameters and settings to obtain the state-of-the-art performing target models on each dataset. The target class of each dataset is fixed across all the attacks adopting it. Standard augmentations are adopted on each dataset to increase the model performance following existing training pipelines [12, 43]. * sign denotes that the transformations/augmentations are randomized.

Remark: Can We Adapt UAA/NB to Our Threat Model? Note that standard UAA/NB make stronger assumptions about attack knowledge than our attack, including the white-box access target model and non-target class samples, but do not rely on poisoning. For a fair comparison between the two attack design methodologies (*cross-boundary noise vs. inward-pointing noise*), we adapt standard UAA/NB to our threat model. In particular, mimicking our attack pipeline, we replace the target model $f_{\theta_{\text{orc}}}$ with a surrogate model $f_{\theta_{\text{sur}}}$ and replace the *in-distribution* non-target-class samples with POOD non-target-class samples in the aforementioned UAP/NB optimization, and further use the optimized UAP/NB trigger as the trigger to poison the training set. We find the resulting attack efficacy still exhibits large variance across different target classes and datasets (see Figure 9 comparison between UAA-Poison, NB-Poison, and Narcissus). The takeaway is that the cross-boundary noise is sensitive to the mismatch between the classes in POOD samples and the non-target classes in the target training dataset. The noise that crosses the boundary between a class in POOD samples and the target class may not cross the one between a non-target class in the target training set and the target class. On the other hand, Narcissus generates noise that points inside of the target class and therefore is more robust to the mismatch on the other non-target classes. We defer further discussions to Appendix 8.1.

To summarize the above two remarks, our **technical novelty** lies in the simple and intuitive idea of synthesizing "inward-pointing" noise, which leads to more robust attack performance than conventional "boundary-crossing" noise used in the UAP or NB literature (Appendix 8.1) and the non-optimized, arbitrary trigger used in the existing backdoor literature (Section 5).

5 EVALUATION

5.1 Experimental Setup

General Settings. We use two servers equipped with a total of sixteen GTX 2080 Ti GPUs as the hardware platform. For most of the evaluations, except the ablation study on target and surrogate model mismatch, we adopt ResNet-18 [12] as the target model. As the mismatch between the target and surrogate model affects different attack algorithms differently, we use ResNet-18 as the surrogate

Method	Clean	HTBA♦[38]	SAA♦ [41]	BadNets-c[10]	BadNets-d[10]	Blend-c[5]	Blend-d[5]	LC [46]	Ours
(a) CIFAR-10 [16] results, 0.05% poison ratio (25 images)									
ACC	95.59	95.53	95.34	94.28	94.81	94.67	94.90	95.42	95.20
Tar-ACC	93.60	93.60	93.80	92.26	93.60	92.10	93.70	93.80	94.10
ASR	0.44	4.87♦	6.00♦	2.60	88.12	1.40	77.99	3.21	**99.03**
(b) PubFig [17] results, 0.024% poison ratio (3 images)									
ACC	93.64	93.44	93.50	93.71	93.14	93.93	93.06	93.06	93.28
Tar-ACC	96.87	96.87	96.87	96.87	100	93.75	96.87	96.87	95.62
ASR	0.00	0.00♦	0.00♦	0.00	0.00	1.55	30.17	0.15	**99.89**
(c) Tiny-ImageNet [18] results, 0.05% poison ratio (50 images)									
ACC	64.82	64.61	64.32	64.10	64.81	64.57	64.58	64.37	64.65
Tar-ACC	70.00	68.00	68.00	68.00	70.00	72.00	68.00	68.00	70.00
ASR	0.13	2.51♦	4.00♦	0.23	0.39	0.52	0.47	1.72	**85.81**

Table 4: Results on (a) CIFAR-10, (b) PubFig, and (c) Tiny-ImageNet. HTBA [38] and SAA [41] with ♦ indicate that their ASRs are only evaluated on the source class. BadNets-c [10] and Blend-c [5] indicate clean-label poisoning, and BadNets-d [10] and Blend-d [5] indicate dirty-label posioning. Blue marks the best ASR.

model structure for all the attacks that require a surrogate model. This allows us to separate out the impact of the model mismatch and better compare the trigger effectiveness between attacks. However, we will show in Table 5 that our attack does not require the same architecture. We set the maximum poison ratio to be 0.05% in most cases. We set the l_∞-norm of triggers to be upper bound by 16/255 following existing work [38, 46]. We evaluate our attack on CIFAR-10 [16], PubFig [17], and the Tiny-ImageNet [18]. Note that the surrogate model in all the existing works is trained with the in-distribution data from all classes in these datasets. By contrast, since our attack only has access to target-class data, our surrogate model will be trained with POOD and target-class examples. The corresponding POOD examples for the three datasets above are Tiny-ImageNet [18], CelebA [28], Caltech-256 [9], respectively. Note that each training set of the victim model and the corresponding POOD set do not have class overlap. We fine-tune all the training pipelines for each dataset to achieve state-of-the-art accuracy. The details of the adopted datasets and the hyperparameters adopted for each training pipeline are provided in Table 3.

Narcissus Settings. To pre-train $f_{\theta_{\text{sur}}}$ on POOD examples, we monitor the training loss and use the cosine annealing scheduler [29] to gradually reduce the learning rate to get full convergence over a POOD dataset. Then, the pre-trained surrogate model is further fine-tuned on the target-class data for 5 epochs. In the trigger synthesis step, the number of gradient descent iterations is set to 1000. We will explain the choice of these two hyperparameters in the ablation study (Table 11). We use RAdam [25] as the optimizer in the poi-warm-up step with a learning rate of 0.1, and in the trigger generation step with a learning rate set to 0.01. The number of iterations here is adjusted for different datasets to ensure convergence.

5.2 Attack Performance

We compare Narcissus with existing clean-label attacks, i.e., HTBA [38], SAA [41], and LC [46][1]. The implementation of these attacks follows their original papers. We also adapt two standard dirty-label attacks, namely, BadNets [10] (denoted by BadNets-c) and the Blend [5] (denoted by Blend-c), to the clean-label setting by poisoning only the target-class and maintaining their original label. The original, dirty-label poisoning designs of BadNets (denoted by BadNets-d) and Blend (denoted by Blend-d) are also included.

Note that, for evaluating the attack performance, it is actually unfair to compare our attack against the baselines, except for the two adapted attacks, because the rest all require the knowledge of non-target-class examples. We still retain these baselines as no existing work is designed for the stringent but more realistic threat model we consider in this paper. Also, since HTBA and SAA can only support one-to-one attacks (i.e., fooling the model to predict inputs from a *source* class into the target class), the ASRs for these two are calculated only on source-class examples. The rest of the baselines and ours are all-to-one attacks, those ASRs are comprehensively evaluated using all non-target-class examples.

Comparison of Attack Effectiveness. Table 4 compares our attack with the baselines in terms of ACC, Tar-ACC, and ASR on CIFAR-10, PubFig, and Tiny-ImageNet. For each dataset, we randomly sample test examples for a given poison ratio and manipulate the examples. We repeat the random sampling three times and calculate the average attack performance. The poison ratios for the three datasets are 0.05%, 0.024%, and 0.05%, respectively. Note that this is much lower than the poison ratio studied in the existing literature, which mostly ranges from 5% to 20% [56].

First, a general observation is that none of the attacks affects the accuracy of clean test examples by much due to the low poisoning rate. The variation of clean ACC before and after attacks is within 0.39% on CIFAR-10, 0.36% on PubFig, and 0.17% on Tiny-ImageNet. Narcissus achieves the highest ASR even with weaker attacker knowledge than the other attacks. Moreover, Narcissus also utilizes smaller perturbations in both training and test stages, as shown in Fig 2, 11, 12. For CIFAR-10 (Table 4 (a)), it is worth highlighting that BadNets-d and Blend-d can achieve a comparable ASR to our attack. However, they require flipping the labels of the poisoned inputs (dirty-label). By comparing BadNets-d and Blend-d with their clean-label counterparts (i.e., BadNets-c and Blend-c), we find that clean-label poisoning is much more challenging than dirty-label poisoning. With the same poison ratio, the clean-label ASR is lower than its dirty-label counterpart by more than 55.7×. For PubFig (Table 4 (b)), our attack is effective even with a poison ratio of 0.024%—only three images from the target class suffice! The successful poison

[1]LC has two modes: the GAN-based and the adversarial-perturbation-based attack. This paper compares with the latter, as it is more effective in the original paper.

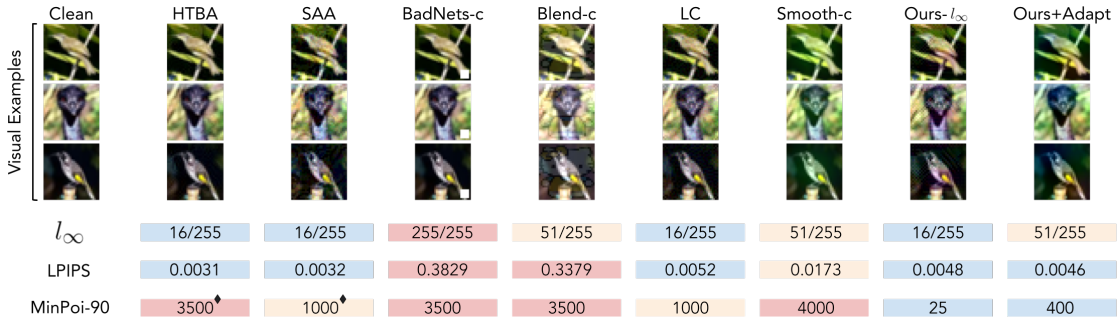

Figure 2: Poisons comparisons on the CIFAR-10. Our trigger and the adaptive Narcissus smooth trigger used in Section 5.4.2 are among the smallest LPIPSs, among a similar scale as the existing clean-label attacks, but it takes a much lower poison rate for our attacks to meet 90% ASR (smaller MinPoi-90). ♦ indicates the results follows one-to-one attack setting (only effective on one class).

ratio on PubFig is much lower than on CIFAR-10 (0.05%) as well as Tiny-ImageNet (0.05%). Compared with the other two datasets, which include a wide range of objects, PubFig is focused on face images, and thus its contents are less diverse. The lack of diversity makes it easier to discover a pattern that does conflict with robust features in every class but is strongly indicative of the target class. This type of un-diverse dataset would create a unique advantage for our optimally synthesized trigger over arbitrarily-chosen triggers. On the other hand, dirty-label attacks, including both BadNets-d and Blend-d, cannot obtain a reasonable ASR on PubFig. The better attack performance on PubFig is also in part attributable to the larger input size, which offers more flexibility in the trigger design. Tiny-ImageNet is a difficult setting. We show that existing clean-label attacks cannot robustly generalize to it in Table 4 (c). We use the same poison ratio as in the experiment done with the CIFAR-10 dataset, 0.05%, which is 50 images on the Tiny-ImageNet. Table 4 (c) shows that with existing all-to-one attacks, both clean-label and dirty-label ones, the best ASR we can get is 1.71%. Under the same poison ratio, Narcissus can obtain an average ASR of 85.81%, which is 50× more effective than the best result from existing attacks.

As a side note, we applied standard randomized augmentations during training, which makes the memorization of backdoor triggers harder [23], but the efficacy of Narcissus is still high.

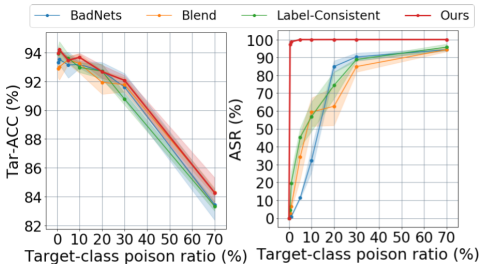

Figure 3: Performance vs. Target-class poison ratio of different triggers on CIFAR-10. We illustrate the trade-off between Tar-ACC and ASR in relation to the tar-class poison ratio. Essentially, a higher poison ratio consistently results in a higher ASR but noticeably reduces the Tar-ACC.

Impact of Target-Class Poison Ratio. Now we focus on all-to-one clean-label attacks, which include BadNet-c, Blend-c, LC, and our attack, and evaluate the impact of the target-class poison ratio on the attack performance. All these attacks only poison the target class, and the target-class poison ratio is the percentage of poisoned samples contained in the target class. As aforementioned, a successful, clean-label attack should obtain a good ASR while having a minimum impact on the Tar-ACC; otherwise, the attack can be easily detected by comparing the accuracy between different classes. Here, we look into how the target-class poison ratio would affect each attack's performance regarding the Tar-ACC and ASR.

As shown in Figure 3, there is a trade-off between Tar-ACC and ASR. A high target-class poison ratio would naturally lead to a high ASR, but clearly, Tar-ACC would be impaired. At a low target-class poison ratio (e.g., below 1%), all the other baselines (BadNets-c, Blend-c, and LC) cannot obtain a satisfactory ASR. As the target-class poison ratio increases, their ASR increases gradually. In particular, all three baselines obtain an ASR of more than 90% when the target-class poison ratio reaches 70%. But at the same time, their Tar-ACC would drop from 94% to 84%-ish. The lowest accuracy of the other non-target classes, on the other hand, is above 91%. Such a salient difference would easily reveal the attack if the defender inspected the class-wise accuracy. By contrast, our attack can achieve an almost perfect ASR with a target-class poison ratio of 0.5%, and at that point, the target-class accuracy is largely maintained.

Attack Stealthiness. Trigger visualizations and comparisons on the CIFAR-10 are provided in Fig. 2. Appendix 8.3 includes additional comparisons on PubFig and TinyImageNet and an analysis of the flexibility of Narcissus trigger design.

In Fig. 2, we depict the relatively smallest values of each metric (associated with better stealthiness) with **Blue**; higher scores (associated with weaker stealthiness) with **Red**; finally, the median ones with **Yellow**. Based on the evaluation and comparison of different triggers on the CIFAR-10 (Fig. 2), we remark that the generated triggers using Narcissus remain one of the most stealth backdoor attacks in terms of obtaining the lowest l_∞, LPIPS, and MinPoi-k.

5.3 Ablation Studies

Impact of Surrogate-Target Model Mismatch. Table 5 shows the ASRs with different surrogate-target (Sur-Tar) model pairs under the poison ratio of 0.05% on CIFAR-10. Note that our attack attains a satisfying ASR for all model pairs. Since for all settings in Table 5, we use a low poison ratio and observe that ACCs and Tar-ACCs are

Tar \ Sur	ResNet-18 [12]	GoogLeNet [42]	EfficientNet-B0 [43]
ResNet-18 [12]	99.03	99.55	**99.97**
GoogLeNet [42]	99.50	**100.00**	**100.00**
EfficientNet-B0 [43]	82.05	**100.00**	87.82

Table 5: ASR results of different Sur-Tar model pairs. darker blue = stronger transferability. Sur: Surrogate; Tar: Target.

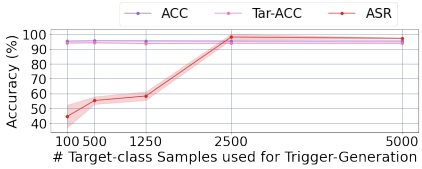

Figure 4: Ablation study on numbers of in-distribution samples used for Trigger Generation step, CIFAR-10. In comparison, existing clean-label backdoors [38, 41, 46] use all 50,000 CIFAR-10 samples. [38, 46] also need the details of the target model.

indistinguishably good (ACCs are above 95%, Tar-ACCs are above 93%), we choose to omit them. Interestingly, the same architecture for our attack is unnecessary for obtaining the best ASR. Hence, as an attacker, it is not necessary to gather detailed information about the target model architecture to maximize the attack performance.

The surrogate model's size doesn't guarantee superior performance. Among three architectures, ResNet-18 has the most neurons, followed by GoogLeNet [42], then EfficientNet-B0 [43]. However, GoogLeNet consistently delivers the best attack performance across all tested architectures. Despite being trained under the same conditions, GoogLeNet achieves the top test results, mirroring its attack efficiency. This is likely because our attack aims to find perturbations targeting the inner part of the target class. A high-performing model like GoogLeNet excels in identifying class-specific features, leading to a more precise attack direction.

We emphasize that Narcissus does not necessitate matching the surrogate and target architectures to optimize attack performance. Notably, attack effectiveness hinges more on the model's task performance than on its similarity to the target. Hence, a practical tip is to choose the surrogate model based on the most cutting-edge architecture for a specific learning task.

In-distribution Knowledge. We now ablate how does the number of in-distribution samples may affect the final ASR (Fig. 4). Noting that, for selection, we conduct random sampling for each experiment to acquire the target class data that was used to generate the trigger. The selected samples are then used for five rounds of poi-warm-up and trigger generation as discussed in Table 11.

As an extreme case analysis, with randomly selecting only 100 samples of the target class (bird), our attack still obtains an average ASR of 44.7%, which is still 14× more effective than existing clean-label backdoors (TABLE 4), which use all 50k in-distribution samples. An interesting remark is that this ASR acquired with 100 samples is lower than using the pre-trained model without any poi-warm-up in TABLE 11. The reason is that poi-warm-up over the 100 samples will overfit the model to the small subset, leading to a model less effective in providing helpful feature extractions on inputs. For simplicity, we use 5000 samples throughout the paper for designing the Narcissus triggers.

POOD ⇒Target	CIFAR-10 ⇒CIFAR-10	Tiny-ImageNet ⇒CIFAR-10	Caltech-256 ⇒CIFAR-10	CelebaA ⇒CIFAR-10	Randomly Initialized
Task Category	General Item Classification	General Item Classification	General Item Classification	Face Recognition	
# Samples	50,000	100,000	30,609	21,144	
# Classes	10	200	257	999	
OTDD [2]	324.3	4068.28	3844.61	6640.23	
ACC*	95.59	73.62	57.19	47.71	
ASR	100	99.03	100	44.6	0.65

Table 6: Comparison with different distribution mismatches on attacking CIFAR-10 with different POODs.

POOD Ablation. We now ablate on different datasets used to pre-train the proxy that supports the Narcissus attack (TABLE 6). Inspired by existing transfer learning work [2] on using optimal transport dataset distance (OTDD) to measure the distance between the training set and test set, we also leverage the OTDD to measure the similarity between POOD data and the in-distribution training data. To calculate OTDD, we resize all samples from POOD datasets to the same input size as CIFAR-10. We find the OTDD score is highly aligned with the Narcissus attack performance. Such an observation gives additional insight when selecting POODs.

To estimate Narcissus' requirement on POOD similarity, we evaluate the transfer learning accuracy (ACC*) of each POODs to CIFAR-10. To calculate ACC*, we consider each pre-trained model using the respected dataset, freeze convolutional layers, and fine-tune the fully connected layers for 20 epochs following [2]; we then obtain ACC* on the target dataset (i.e., CIFAR-10). ACC*s' values from POOD-transferred models end up with much smaller ACC than training with the in-distribution samples, which implies our attack's requirement on task similarity is small. One highlight is that using the most irrelevant POOD, CelebaA, we can still achieve an ASR at 44.6% (14× effective than other clean-label attacks in TABLE 4).

With the above ablation study on POOD, we hereby present a guideline to select practical POOD examples. For a given target class and its associated samples, the attacker can crawl from the Internet for data that fall into similar categories (as shown in TABLE 6, which also tends to have a smaller OTDD). For instance, if the attack target is a face image, one could download more face images as POOD examples. If the attack target is a physical object such as an airplane, one could leverage the open-sourced large-scale general object classification dataset (e.g., ImageNet) as the POOD example.

5.4 Defenses

There are three major lines of existing backdoor defenses: 1) model-based backdoor unlearning; 2) poisons filtering/detection; 3) robust training over poisoned datasets. We consider eight defenses under these major lines to analyze the impact of Narcissus on existing defenses. Some defenses, such as Neural Cleanse [47], Fine Pruning [24], STRIP [8] are popular choices of defenses in the past but did not necessarily achieve the best performance, and others, such as MOTH [44], I-BAU [54], Frequency-based Detector [56], ABL [21], and NONE [48] represent the state-of-the-art defenses. In the main text, all results are evaluated on CIFAR-10.

5.4.1 Model-based Backdoor Unlearning. These defenses [24, 44, 47, 49, 50, 54] aim to remove the backdoor effects from a given pre-trained poisoned model. We incorporate four defenses under this line of work and discuss their effects on mitigating the Narcissus attack. For the pre-trained poisoned model, we use a Narcissus

poisoned CIFAR-10 model with a poison ratio of 0.05%, whose ACC, Target Class ACC, and ASR are 95.34%, 93.44%, and 97.10%, respectively. They serve as the baseline results before the defenses are conducted.

Neural Cleanse: [47] examines each label to synthesize potential triggers as though the current label were the target. By performing outlier detection on the reverse-engineered triggers, it identifies the target label and subsequently unlearns the backdoor.

We follow the original implementation of Neural Cleanse and test it on CIFAR-10. We split the test set of CIFAR-10 into two size-5000 groups: the defense and the validation set. The former is used to execute the defense algorithm, and the latter is used for evaluating the defense performance. However, none of the labels is marked as outliers. The follow-up work, TABOR [11], is similarly based on trigger synthesis, and it remarks that inspecting the trigger generation loss can additionally help to identify outliers and further improve the defense efficacy. We test this idea but find that the outlier detector can still not find anything suspicious. Fundamentally, both Neural Cleanse's and TABOR's trigger synthesis processes make the assumption that the trigger is localized, which is met by most existing attacks. However, when the trigger is large and smeared over the entire image, like ours, the reverse-engineered trigger tends to be inaccurate. Worse yet, the synthesized patterns for non-target classes are also global patterns, and hence it is difficult to run outlier detection to distinguish the target class from the other classes.

MOTH: Model Orthogonalization (MOTH) is a novel technique that strengthens pre-trained models against backdoors by increasing the separation between classes [44]. The core concept involves synthesizing backdoor patterns class-wise for the poisoned model, then learning samples perturbed with these patterns using correct labels to increase the distance between classes. We applied MOTH to the Narcissus-poisoned model using ResNet-18 and CIFAR-10 hyperparameters from the authors, focusing on the l_∞-norm. Although the ASR dropped from 93.44% to 72.83%, MOTH could not entirely eliminate the attack. Since MOTH primarily targets synthesizing boundary-crossing perturbations, our attack slips through.

	ACC	Tar-ACC	ASR
None	95.34	93.44	97.10
SGD (lr = 0.001)	94.1	92.2	96.5
SGD (lr = 0.001)$^\diamond$	95.0	93.0	94.5
SGD (lr = 0.01)	94.6	92.6	91.5
SGD (lr = 0.01)$^\diamond$	95.0	92.6	90.8

Table 7: Fine-pruning on mitigating Narcissus. The results with $^\diamond$ indicate the adoption of a 30-round fine-tuning according to the original work. We observe that incorporating any lr larger than 0.01 results in a broken model with ACC no better than random guessing (lr = 0.01 is our experiment stopping point).

Fine-Pruning: The key idea of fine-pruning [24] is to prune the inactive neurons for clean inputs and further fine-tune the pruned model over clean data to improve accuracy. We split the test set (unseen by the poisoned target model) into two size-5000 subsets, similar to Neuron Cleanse. Then, we select 1000 samples from the defense set to ensure that the performance drop after defense is within 20% as per the original implementation. The other set is kept intact to evaluate the defense's performance. Results of using fine-pruning for mitigating Narcissus are shown in Table 7.

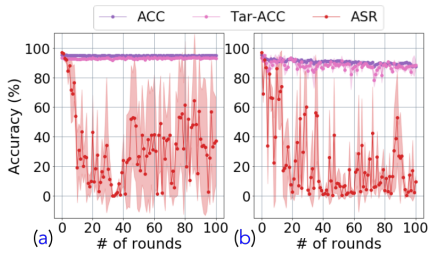

Figure 5: Defense results with I-BAU on Narcissus. The original work incorporates two different optimizers, namely SGD (a) and Adam (b). We fine-tune both optimizers to their best hyperparameters and launch the defense for 100 rounds.

It is worth mentioning that we fine-tune the defense to our best efforts. However, we observe that the defense's efficacy is limited in all the evaluated settings. The reason might be that Narcissus triggers point towards the inside of the target class. They might activate similar neurons as the target-class samples, making it hard to remove the backdoor without hurting the model's performance.

I-BAU: Implicit-Hypergradient-based Backdoor Unlearning [54], or the I-BAU, is a novel defense framework obtaining state-of-the-art effectiveness across different backdoor attacks and datasets. The idea of I-BAU is to alternate between trigger synthesis and unlearning for several rounds. Intuitively, through this process, the trigger synthesized will get stronger, and the model that unlearns stronger triggers will become more robust. We follow the original implementation and the same defense-validation split as the other two defenses above. We launch the defense algorithm with SGD [37] optimizer and Adam [15] optimizer, following the original work.

The defense performance over the first 100 rounds is shown in Figure 5. In the original study, most existing all-to-one backdoor attacks are neutralized in just one round. Yet, Narcissus remains effective even after 100 rounds. Notably, I-BAU's performance with Narcissus doesn't converge smoothly, unlike other attacks. Overall, while I-BAU is more effective than Neural Cleanse and Fine-Pruning, its performance on Narcissus is erratic.

We investigate the root cause of the unstable performance of I-BAU. Intriguingly, as the defense iterations proceed, I-BAU starts to synthesize and unlearn some robust features that contain obvious semantic information from the training data (see Figure 6). Compared to SGD, I-BAU with Adam unlearns more visually clear, robust features. Based on Figure 5, it also suffers from a larger drop in ACC. However, the fact that robust features start to get synthesized before our trigger is fully mitigated indicates that our optimized trigger is as persistent as the robust features of each class. Removing our trigger would inevitably hurt the model's performance.

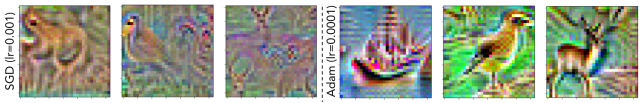

Figure 6: Intriguing visual observations with I-BAU on Narcissus. Some I-BAU synthesized noises start showing strong semantic information before the defense takes effect.

5.4.2 Poisons filtering/detetcion. This line of work filters/detects the backdoored samples directly. [8, 34, 52, 56] We study a popular test-time filtering defense, STRIP [8], as well as a state-of-the-art detection based on high-frequency artifacts synthesis [56].

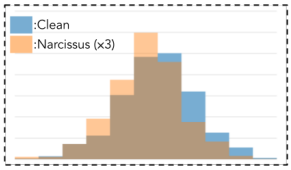

Figure 7: STRIP [8] on CIFAR-10. Minimum entropy of clean samples is 3.526; meanwhile, with the magnified Narcissus trigger (used during test), the entropy is 3.539.

STRIP: We follow the original implementation and find that STRIP is not able to detect Narcissus on the CIFAR-10 during the test phase (Fig. 7). As blending our poisoned sample with clean samples will largely reduce the attack efficacy thus evading the filtering.
Frequency-Detection: Vanilla Narcissus trigger with an l_∞-norm constraint does contain high-frequency artifacts, which makes it easy to detect by the frequency-based backdoor detector. However, we can perform an adaptive attack with Narcissus by simply setting the allowable set of trigger patterns, Δ, to be the low-frequency patterns. To optimize the trigger design over this set, we can pass the intermediate trigger design obtained by each gradient descent update through a low-pass filter. The original work [56] discusses the adaptability of standard dirty-label poisoning attacks by incorporating a low-pass filter, termed "Smooth attack." We design our adapted Narcissus by using the same setup as the Smooth attack and applying the same low-pass filter.

	Smooth [56]	Ours	Ours+Adapt
Detection ACC	75.16	98.34	77.83
Detection Rate	53.62	100	58.96

Table 8: Detection results with the frequency-based backdoor sample detector. We show the detection results of the original Narcissus trigger and the adaptive Narcissus trigger optimized with Δ being set as a low-pass filter.

A comparison of the detection performance on the vanilla l_∞-constrained Narcissus, the adaptive Narcissus, and the original smooth trigger in [56] (i.e., an optimized trigger with a low-pass filter as the constraint) is provided in Table 8. With the same low pass filter incorporated, our adaptive attack can achieve similar frequency-domain stealth as the Smooth attack. Figure 2 visualizes the original smooth trigger and our adaptive Narcissus smooth trigger. Our adaptive trigger obtains better visual stealthiness (lower LPIPS) and dataset stealthiness (lower MinPoi-90) than [56].

Table 9 compares the effectiveness of the two triggers. We also include the result for Smooth-d, which injects the smooth trigger into non-target classes. While it is unfair to compare ours with Smooth-d, we still include this setting to see if our clean-label attack can match up with the performance of the dirty-label [56]. Our adapted Narcissus trigger achieved a 77.59% higher ASR than the original Smooth trigger under the clean-label poisoning case. It is also worth highlighting that the adapted Narcissus performed a slightly higher ASR than the dirty-label "Smooth attack." These results indicate the efficacy and adaptability of Narcissus.

5.4.3 Robust Training over Poisoned Dataset. This line of defense focuses on training a robust model from a poisoned dataset. The state-of-the-art along this line is anti-backdoor learning (ABL) [21] and Non-linearity (NONE) training [48].

	Clean	Smooth-c [56]	Smooth-d [56]	Ours+Adapt
ACC	95.59	94.70	95.10	93.16
Tar-ACC	93.60	91.30	93.50	91.30
ASR	0.44	12.71	90.13	**90.30**

Table 9: Attack efficacy of different frequency-invisible triggers. Smooth-c [56] indicates clean-label poisoning, and Smooth-d indicates dirty-label poisoning. Blue remarks the best ASR.

	0.00%		0.05%		0.5%	
	ACC	ASR	ACC	ASR	ACC	ASR
Early	89.37	NA	89.54	100	89.41	100
Later	85.46	NA	83.27	**98.47**	75.31	**100**

Table 10: Results of using ABL to train over the Narcissus poisoned CIFAR-10 with a different poison ratio. "Early" denotes the results after early learning. "Later" denotes the final results of the ABL. Blue denotes failed defenses.

ABL: ABL observes that backdoored samples achieve the lowest loss after a few training epochs. Thus, one can quarantine some of the poisoned samples and unlearning them to mitigate the effects of the backdoor trigger. We implement the ABL with the original settings. We deploy the Narcissus trigger generated using the ResNet-18 and target the CIFAR-10's class "bird." We use a WideResNet-16-1 [53] as the target model, following the original work.

As shown in Table 10, the ASRs of our attack remain close to 100% on the poisoned model after ABL. This illustrates the ineffectiveness of the ABL in mitigating our attack. We investigate the isolation results from the ABL early learning for the poison ratio of 0.05%, and find that only three target class samples are isolated, and the rest (497 samples) are all from non-target classes. Moreover, none of the three isolated "bird" samples are poisoned samples. With a larger poison ratio of 0.5%, ten target-class samples are isolated, of which only three are poisoned. The above observation shows that when the poison ratio is low (e.g., 0.05% to 0.5%), the loss of our poisoned examples has a similar scale to that of clean samples and, therefore, Narcissus can successfully evade loss scanning.
NONE: NONE [48] is an innovative training method that prevents the creation of hyperplanes as classification surfaces between different labels in the input domains. This approach is effective as many backdoor attacks rely on these hyperplanes in hidden spaces. We used the original implementation of NONE and applied it to a ResNet-18 model with CIFAR-10 hyperparameters on an Narcissus-poisoned CIFAR-10. However, with the defense, the ASR is still up to 83.29%. The limited effectiveness of NONE to Narcissus can be attributed to its assumption that triggers are patch-like, similar to the reason Neural Cleanse failed to detect backdoor triggers.

6 EXTENSION TO PHYSICAL WORLD

Existing exploration of physical backdoor attacks is limited to dirty-label poisoning with an arbitrarily chosen trigger. As our attack significantly improves upon existing clean-label backdoor attacks in the digital space, we explore the possibility of extending it to perform a physical world attack, where the backdoor trigger in the test phase is directly applied to the scene.
Physical World Attack Design. Clean-label backdoor attacks in the physical world are challenging due to: (1) **Information Loss:** Capturing a physical trigger through a camera will cause information loss. In particular, a certain degree of hue change and pixel loss would occur during this process. Attackers are generally

unaware of lens-related variations, so it is hard to optimize against them adaptively. (2) **Affine Transformations:** A physical trigger can potentially be captured by a camera at different viewing angles, rotations, and backgrounds.

Narcissus designs the trigger optimization framework, and hence it offers the flexibility to incorporate various constraints. We propose to use randomized augmentation [3] and seek the trigger by minimizing the loss over the augmented dataset. As changing all the values of the model input space is impractical in the physical world, we constrain our trigger into a square shape (8×8). We patch the trigger to random locations to maintain its effectiveness when the trigger is revealed in different locations of the scene.

Each time the trigger is patched to a different location, it would result in a new computation graph for back-propagation. Thus, the total size of the computational graph grows linearly with the number of locations. We propose random padding of the small-size square trigger optimizing area to simulate the effect of patching the trigger to random locations while unifying the computational graph Finally, we incorporate a hue change with a ± 0.3 scale and random rotation. To derive a meaningful gradient from these intense random transformations, we use the expectation over transformation (EOT) method. EOT is commonly used in evasion attacks [3], particularly for countering transformation-based defenses and real-world attacks. With EOT, we can create a Narcissus patch trigger for real-world attacks.

Evaluation. We evaluate two physical attack baselines: white square and random noise. Triggers are added to the target-class training data, creating three poisoned datasets. Each group has a 0.05% poison ratio, using Tiny-ImageNet as the target dataset. We exclude LC, HTBA, and SAA from the comparison since they need non-target-class samples. We selected 50 random "bullfrog" images (0.05% of the dataset) and added a trigger to each, which was randomly rotated and positioned. We trained poisoned ResNet-18 models over the three poisoned datasets. We then tested how these models reacted to their respective triggers, as shown in Figure 8. All triggers were displayed on an iPhone 13 with a 6.06" OLED screen. For Narcissus, frames with triggers had a 100% ASR, while others showed 0%. Only Narcissus effectively executed a clean-label backdoor attack in real-world conditions.

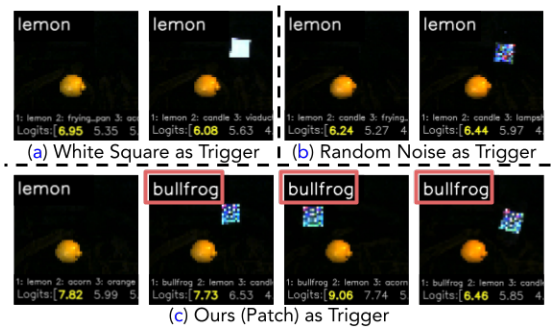

Figure 8: Different attacks toward the physical world. We use "bullfrog" as the target label. We show the three poisoned models' behaviors before and after observing the trigger. Results are screenshots taken from the video-demo.

7 CONCLUSION & DISCUSSION

We present Narcissus, a model-agnostic clean-label backdoor attack leveraging only target-class samples and POOD data. With minimal attack knowledge, our technique excels, modifying just 0.05% or less of the training set. Remarkably, our attack's success rate surpasses other clean-label backdoor attacks by 30.33× to 64.45×, even though they access the full training data. Narcissus also outperforms in pixel (l_∞) and visual stealthiness (LPIPS), and dataset stealthiness (MinPoi-k) metrics. Furthermore, Narcissus scales well with large datasets and proves effective in physical attack.

Additionally, the efficacy of our attack is emphasized in a comprehensive study of state-of-the-art defenses, where, surprisingly, none of the defenses could reliably hinder our attack. Our results indicate that existing popular machine learning pipelines using public data sources can be easily exposed to practical clean-label backdoor attacks – as all the attacker requires is knowing a portion of the target class.

Limitations. Using an arbitrary dataset, Narcissus outperforms previous attacks in ASR under clean-label settings. Yet, optimal attack efficacy is influenced by the similarity between the POOD data and the poisoned dataset (Table 6). Given the POOD dependency, our findings indicate that OTDD can effectively guide POOD dataset selection. We recommend OTDD as a dependable metric for choosing POOD samples. Exploring methods for efficient clean-label backdoor attacks without knowing the target task category is a promising direction for future studies.

Implications for Theoretical Research. Existing theoretical studies have mainly focused on dirty-label backdoor attacks. However, our method shows that clean-label attacks can also severely compromise deep neural networks, even in real-world scenarios. It's crucial to study clean-label attacks as we do dirty-label ones. Specifically, Narcissus can attack as effectively as dirty-label methods but is more stealthy. Understanding why Narcissus is powerful, even with limited knowledge and poison ratio in a clean-label context, is a vital theoretical pursuit.

Implications for Empirical Research. Our attack demonstrates a high degree of adaptability and efficacy across various environments. Most notably, by exposing the vulnerabilities of popular defenses [8, 24, 47], and state-of-the-art defenses [21, 44, 48, 54, 56], more reliable, and general defenses are needed to respond. Specifically, empirical research on removing Narcissus's effects without impairing performance on clean samples or conducting reliable attack detection is critical. Mainly, Narcissus also stands out for clean-label poisoning, with a low poison ratio (less impact to parameters change) and flexibility in trigger design. Each of those properties breaks some assumptions of existing defenses, motivating more general defenses to be developed.

ACKNOWLEDGMENT

RJ and the ReDS lab acknowledge support through grants from the Amazon-Virginia Tech Initiative for Efficient and Robust Machine Learning, the National Science Foundation under Grant No. IIS-2312794, NSF IIS-2313130, NSF OAC-2239622, and the Commonwealth Cyber Initiative. YZ acknowledges the support from the Amazon Fellowship. This work is also partially supported by Sony AI. We want to thank anonymous reviewers for their valuable feedback and support.

REFERENCES

[1] Jacob D Abernethy and Rafael Frongillo. 2011. A collaborative mechanism for crowdsourcing prediction problems. *NeurIPS* 24 (2011).

[2] David Alvarez-Melis and Nicolo Fusi. 2020. Geometric dataset distances via optimal transport. *NeurIPS* 33 (2020), 21428–21439.

[3] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2018. Synthesizing robust adversarial examples. In *ICML*. PMLR, 284–293.

[4] Léon Bottou. 2012. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*. Springer, 421–436.

[5] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. arXiv:1712.05526 [cs.CR]

[6] Shuyu Cheng, Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. 2019. Improving black-box adversarial attacks with a transfer-based prior. *NeurIPS* (2019).

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.

[8] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. 2019. STRIP: A Defence against Trojan Attacks on Deep Neural Networks. In *ACSAC '19*. 113–125.

[9] Gregory Griffin, Alex Holub, and Pietro Perona. 2007. Caltech-256 object category dataset. (2007).

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

[11] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. 2019. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv:1908.01763* (2019).

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE CVPR*. 770–778.

[13] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. 2013. Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*.

[14] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*. 4171–4186.

[15] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[16] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[17] Neeraj Kumar, Alexander C Berg, Peter N Belhumeur, and Shree K Nayar. 2009. Attribute and simile classifiers for face verification. In *ICCV*. IEEE, 365–372.

[18] Ya Le and Xuan Yang. 2015. Tiny imagenet visual recognition challenge. *CS 231N* 7, 7 (2015), 3.

[19] Shaofeng Li, Minhui Xue, Benjamin Zhao, Haojin Zhu, and Xinpeng Zhang. 2020. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Transactions on Dependable and Secure Computing* (2020).

[20] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. 2021. Invisible Backdoor Attack with Sample-Specific Triggers. In *ICCV*.

[21] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Anti-backdoor learning: Training clean models on poisoned data. *NeurIPS* 34 (2021).

[22] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2020. Backdoor learning: A survey. *arXiv:2007.08745* (2020).

[23] Yiming Li, Tongqing Zhai, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2021. Backdoor Attack in the Physical World. In *ICLR Workshop*.

[24] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *RAID*. Springer.

[25] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2019. On the Variance of the Adaptive Learning Rate and Beyond. In *ICLR*.

[26] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *NDSS*.

[27] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. 2020. Reflection backdoor: A natural backdoor attack on deep neural networks. In *ECCV, 2020*.

[28] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of ICCV*.

[29] Ilya Loshchilov and Frank Hutter. 2017. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*.

[30] Tuan Anh Nguyen and Anh Tran. 2020. Input-aware dynamic backdoor attack. *NeurIPS* 33 (2020), 3454–3464.

[31] Tuan Anh Nguyen and Anh Tuan Tran. 2020. WaNet-Imperceptible Warping-based Backdoor Attack. In *ICLR*.

[32] Xiangyu Qi, Tinghao Xie, Yiming Li, Saeed Mahloujifar, and Prateek Mittal. 2022. Revisiting the assumption of latent separability for backdoor defenses. In *ICLR*.

[33] Xiangyu Qi, Tinghao Xie, Ruizhe Pan, Jifeng Zhu, Yong Yang, and Kai Bu. 2022. Towards practical deployment-stage backdoor attack on deep neural networks. In *CVPR*.

[34] Xiangyu Qi, Tinghao Xie, Jiachen T Wang, Tong Wu, Saeed Mahloujifar, and Prateek Mittal. 2023. Towards A Proactive {ML} Approach for Detecting Backdoor Poison Samples. In *USENIX Security 23*. 1685–1702.

[35] Xiangyu Qi, Jifeng Zhu, Chulin Xie, and Yong Yang. 2021. Subnet replacement: Deployment-stage backdoor attack against deep neural networks in gray-box setting. *arXiv:2107.07240* (2021).

[36] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. [n. d.]. Adversarial robustness through local linearization. *NeurIPS, 2019* 32 ([n. d.]).

[37] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv:1609.04747* (2016).

[38] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. 2020. Hidden trigger backdoor attacks. In *Proceedings of the AAAI*, Vol. 34. 11957–11965.

[39] Esha Sarkar, Hadjer Benkraouda, and Michail Maniatakos. 2020. FaceHack: Triggering backdoored facial recognition systems using facial characteristics. *arXiv:2006.11623* (2020).

[40] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE S&P*. IEEE, 3–18.

[41] Hossein Souri, Micah Goldblum, Liam Fowl, Rama Chellappa, and Tom Goldstein. 2021. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. *arXiv:2106.08970* (2021).

[42] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE CVPR*. 1–9.

[43] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*. PMLR, 6105–6114.

[44] Guanhong Tao, Yingqi Liu, Guangyu Shen, Qiuling Xu, Shengwei An, Zhuo Zhang, and Xiangyu Zhang. 2022. Model orthogonalization: Class distance hardening in neural networks for better security. In *2022 IEEE Symposium on Security and Privacy*. IEEE, 1372–1389.

[45] Guanhong Tao, Zhenting Wang, Siyuan Cheng, Shiqing Ma, Shengwei An, Yingqi Liu, Guangyu Shen, Zhuo Zhang, Yunshu Mao, and Xiangyu Zhang. 2022. Backdoor vulnerabilities in normally trained deep learning models. *arXiv preprint arXiv:2211.15929* (2022).

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

[47] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE S&P*. IEEE, 707–723.

[48] Zhenting Wang, Hailun Ding, Juan Zhai, and Shiqing Ma. 2022. Training with more confidence: Mitigating injected and natural backdoors during training. *NeurIPS* 35 (2022), 36396–36410.

[49] Zhenting Wang, Kai Mei, Hailun Ding, Juan Zhai, and Shiqing Ma. 2022. Rethinking the Reverse-engineering of Trojan Triggers. *NeuIPS* 35 (2022).

[50] Zhenting Wang, Kai Mei, Juan Zhai, and Shiqing Ma. 2022. UNICORN: A Unified Backdoor Trigger Inversion Framework. In *ICLR*.

[51] Zhenting Wang, Juan Zhai, and Shiqing Ma. 2022. Bppattack: Stealthy and efficient trojan attacks against deep neural networks via image quantization and contrastive adversarial learning. In *CVPR*.

[52] Tinghao Xie, Xiangyu Qi, Ping He, Yiming Li, Jiachen T Wang, and Prateek Mittal. 2023. BaDExpert: Extracting Backdoor Functionality for Accurate Backdoor Input Detection. *arXiv:2308.12439* (2023).

[53] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference (BMVC)*. 87.1–87.12.

[54] Yi Zeng, Si Chen, Won Park, Zhuoqing Mao, Ming Jin, and Ruoxi Jia. 2022. Adversarial Unlearning of Backdoors via Implicit Hypergradient. In *ICLR*.

[55] Yi Zeng, Minzhou Pan, Himanshu Jahagirdar, Ming Jin, Lingjuan Lyu, and Ruoxi Jia. 2023. Meta-Sift: How to Sift Out a Clean Subset in the Presence of Data Poisoning?. In *USENIX Security 23*. 1667–1684.

[56] Yi Zeng, Won Park, Z Morley Mao, and Ruoxi Jia. 2021. Rethinking the Backdoor Attacks' Triggers: A Frequency Perspective. In *ICCV*.

[57] Chaoning Zhang, Philipp Benz, Tooba Imtiaz, and In So Kweon. 2020. Understanding adversarial examples from the mutual influence of images and perturbations. In *CVPR*. 14521–14530.

[58] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE CVPR*. 586–595.

[59] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. 2020. Clean-Label Backdoor Attacks on Video Recognition Models. In *2020 CVPR*. IEEE Computer Society, 14431–14440.

8 APPENDIX

8.1 Compare with Boundary-Crossing Noise

We compare the performance of the Universal Adversarial Attack (UAA) [57], our proposed UAA-Poison, the standard Natural Backdoor (NB) from [45], and the NB-Poison in Figure 9. To ensure consistency, the attacker's knowledge for all methods is identical. UAA applies the post-hoc PDG universal attack [57] using $f_{\theta_{\text{sur}}}$ and POOD data. UAA-Poison uses the UAP from UAA as a backdoor trigger, poisoning the training set, training the victim model, and amplifying the trigger by 3× for testing, similar to our method. Likewise, NB is adapted using the same surrogate model as in UAA and our method. NB-Poison intensifies the NB triggers using clean-label positioning and testing time amplification. For testing, all attacks have a perturbation constraint of $l_\infty = 48/255$ (accounting for the UAP/NB triggers and post-3× amplification). UAA-Poison/NB-Poison and our method utilize a poison ratio of 0.05%, as mentioned earlier. To evaluate the effectiveness on CIFAR-10 models and examine data-mismatch dependency, we employ the GTSRB dataset [13] as POOD, noting that GTSRB, a traffic sign dataset, has no class overlap with CIFAR-10.

In Figure 9, there's noticeable ASR variance across the labels set as targets for the four baselines (UAA/UAA-Poison/NB/NB-Poison). Due to the significant disparity between GTSRB and CIFAR-10, all baseline attacks, whether poisoned or not, show poor performance. Conversely, Narcissus consistently achieves an ASR of over 90% for all target labels or with different POODs. This success is attributed to our unique optimization approach to generate target-class inward-pointing noise using target-class samples as detailed in Eqn (2). Especially given the realistic attacker knowledge constraints outlined in Section 3, our approach effectively stabilizes performance, countering the variability introduced by using POOD data for synthesis, by focusing on in-distribution target-class samples.

Our core concept is to create a spurious correlation more accurately represents the target class, enabling learning algorithms to absorb it from the poisoned dataset seamlessly. This approach builds on the basic principles of backdoor attacks and diverges from the conventional UAA/NB, which generates noise to assist non-target class samples in crossing a pre-defined decision boundary.

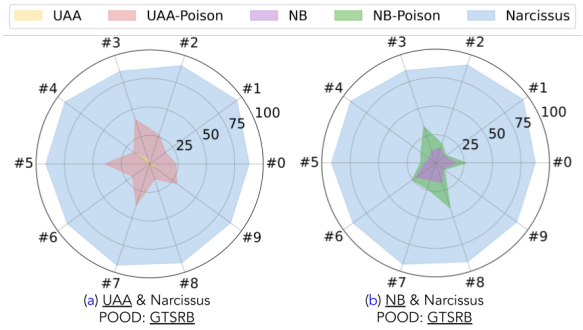

Figure 9: Class-wise ASR comparison with different target labels between UAA/NB, adapted clean-label backdoor attack based on UAP/NB (0.05%), and Narcissus (0.05%) on the CIFAR-10. The number at each vertex denotes the target label.

8.2 Additional Ablation Study

Impact of l_∞-norm ball radius. Past studies typically set the l_∞-norm bound at 16/255 with limited perturbations. Our analysis, shown in Figure 10, highlights how the attack's success rate (ASR) tends to rise with an increase in the radius, while the accuracy (ACC) and Targeted Accuracy (Tar-ACC) remain largely consistent. Notably, even at a minimal l_∞-norm of 2/255 and a 0.05% poison budget, our method registers an ASR over 20% for CIFAR-10. Comparatively, as per Table 4, this is an improvement of at least 7× over other clean-label benchmarks, with the highest previously noted ASR being 3.21%, while also demanding a reduced radius.

Poi-warm-up Fine-tuning & Trigger Generation Iterations. We assessed the ASR based on different fine-tuning iterations in the poi-warm-up and gradient descent phases during trigger creation, with results displayed in Table 11. Using a minimal poison ratio (0.05%), all ACCs and Tar-ACCs exceed 95% and 93% respectively, so we've excluded these metrics. Key findings include: (1) Even without additional fine-tuning on target-class data, utilizing the POOD-data-pre-trained surrogate model results in an ASR over 70%. Minimal fine-tuning rounds further enhance trigger effectiveness. (2) Excessive fine-tuning iterations don't always enhance attack potency. This might cause the model to overfit to D_t, or the objective in (2) becomes negligibly small. If both trigger synthesis and fine-tuning minimize loss over D_t, an overly optimized model offers limited scope for trigger adjustments, making potent trigger identification harder. Practically, the optimal ASR arises after five fine-tuning rounds. (3) Incremental trigger update iterations enhance trigger potency. Based on this, our standard settings for subsequent evaluations in this paper are 5 fine-tuning rounds and 1000 trigger update rounds.

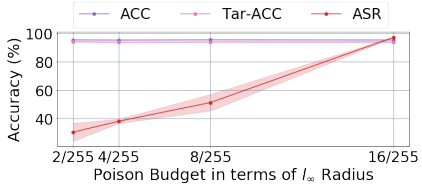

Figure 10: Ablation study on poison budget in terms of the l_∞-norm ball radius on CIFAR-10. Narcissus can still maintain a certain scale of efficacy (30.6% ± 8.12% ASR) even with a meager poison budget (2/255 & 0.05% poison ratio).

Generation / Poi-warm-up	150	300	500	1000
0	70.63	75.32	81.63	85.88
1	90.60	91.52	95.36	90.61
5	77.17	90.13	96.04	**99.03**
10	79.37	87.83	87.67	85.74
15	90.51	88.48	89.10	83.80

Table 11: The number of rounds of poi-warm-up vs. the number of rounds of the trigger synthesis. Blue denotes the best ASR.

Alternative Choice of Obtaining $f_{\theta_{\text{sur}}}$. The final design efficiently adapts a well-trained model from the POOD dataset via fine-tuning to target class samples, resulting in $f_{\theta_{\text{sur}}}$. An alternative approach, though less effective, involves training a model from scratch on POOD and target class samples. Table 12 compares the attack efficacy of both methods on CIFAR-10. The two-pronged design offers similar efficacy as the one-step alternative but provides better

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Tuning-based	98.99	99.93	100	99.64	99.54	100	98.67	99.81	97.27	99.92
From Scratch	98.91	99.45	99.84	99.98	99.93	100	91.45	99.97	95.17	99.95

Table 12: Comparison of final ASR on CIFAR-10 using different ways of obtaining $f_{\theta_{sur}}$. The Tuning-based method is our final design, where we fine-tune the model trained on POOD to the target class samples. From Scratch denotes an alternative option where the attacker trains $f_{\theta_{sur}}$ using the combined dataset of POOD + target class samples from scratch for each target class.

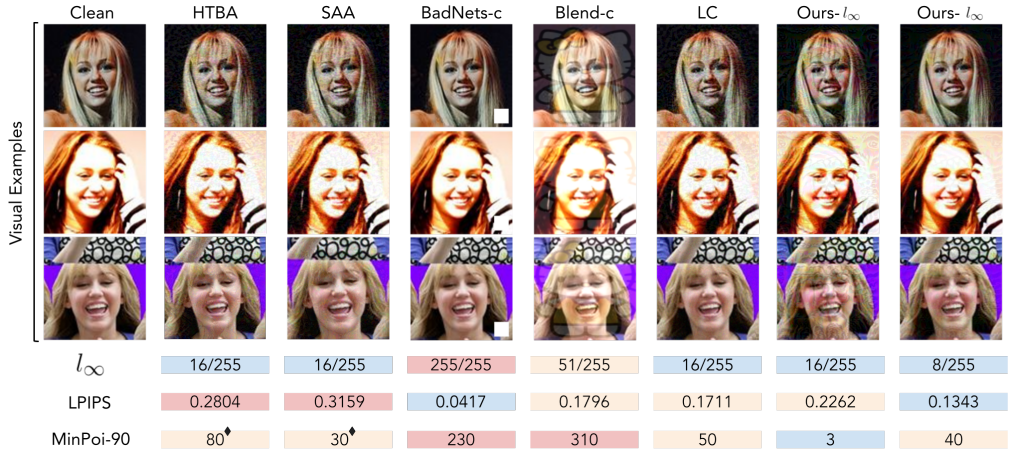

Figure 11: Visual results and comparisons on the PubFig. Our proposed trigger and the trigger with $l_\infty = 8/255$ are among the smallest LPIPS ones, with the smallest MinPoi-90. Especially the existing clean-label attacks' noise obtained higher LPIPS on PubFig. As a side note, all existing clean-label attacks become ineffective under $l_\infty = 8/255$. ♦ indicates that the MinPoi-90 is based on the one-to-one case.

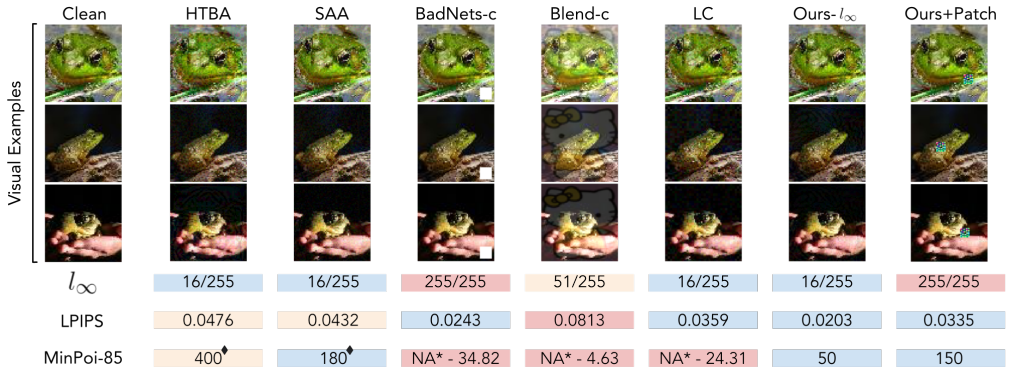

Figure 12: Visual results and comparisons on the Tiny-ImageNet. Our proposed trigger and the adaptive trigger for the physical world in section 6 are among the smallest LPIPS ones, with the smallest MinPoi-85. ♦ indicates that the MinPoi-85 is based on the one-to-one case. NA* means the attack cannot achieve the required ASR, even poisoning the whole target class.

flexibility and computational efficiency when selecting new target classes.

8.3 More Stealthiness Comparisons

In Figure 11 and 12, we delve into the stealthiness of backdoor triggers across our datasets. We observed that the ASR of HTBA [38] and SAA [41] is contingent on specific source-target pairs, especially in larger datasets. Only certain class pairs, like CIFAR-10 2-7, PubFig 60-52, and Tiny-ImageNet 2-98, yield acceptable ASRs. In fact, most pairings in large datasets, such as PubFig 60-5 and Tiny-ImageNet 2-70, result in low ASRs under 5%, even after poisoning 1000 samples in our tests. While comparing these attacks with others might seem skewed, given they target only source class samples, we evaluated their MinPoi-k using the highest ASR class pairs to highlight Narcissus's effectiveness. Additionally, when measuring the MinPoi-k on the Tiny-ImageNet, we observed several attacks could not achieve the required k% ASR, even upon poisoning the whole target class. Thus we mark their MinPoi-k as "NA*" and show their ASR when poisoning the whole target class. For example, the MinPoi-85 of LC [46] on Tiny-ImageNet is "NA*-24.31", which means even poisoning the whole target class, LC [46] can still only achieve an ASR of 23.41%.

Visual stealthiness and flexibility of Narcissus: Based on visual evaluation on the CIFAR-10, PubFig, and the Tiny-ImageNet in Fig. 2, 11, and 12, we remark that the generated triggers using Narcissus remain the most stealthy ones. Most importantly, Narcissus offers the attacker stronger trigger design flexibility, allowing the attack to accommodate different constraints and achieving the best trade-off in stealthiness and attack efficacy.