Distribution Preserving Backdoor Attack in Self-supervised Learning

Guanhong Tao[1*], Zhenting Wang[2*], Shiwei Feng[1], Guangyu Shen[1], Shiqing Ma[3], Xiangyu Zhang[1]
[1]Purdue University, [2]Rutgers University, [3]University of Massachusetts Amherst

2024 IEEE Symposium on Security and Privacy (SP) | 979-8-3503-3130-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/SP54263.2024.00029

***Abstract*—Self-supervised learning is widely used in various domains for building foundation models. It has been demonstrated to achieve state-of-the-art performance in a range of tasks. In the computer vision domain, self-supervised learning is utilized to generate an image feature extractor, called an encoder, such that a variety of downstream tasks can build classifiers on top of it with limited data and resources. Despite the impressive performance of self-supervised learning, it is susceptible to backdoor attacks, where an attacker injects a backdoor into its unlabeled training data. A downstream classifier built on the backdoored encoder will misclassify any inputs inserted with the trigger to a target label. Existing backdoor attacks in self-supervised learning possess a key out-of-distribution property, where the poisoned samples significantly differ from the clean data in the feature space. The poisoned distribution is also exceptionally concentrated, inducing high pairwise similarity among poisoned samples. As a result, these attacks can be detected by state-of-the-art defense techniques. We propose a novel distribution preserving attack, which transforms the poisoned samples into in-distribution data by reducing their distributional distance to the clean data. We also distribute the poisoned data to a wider region in the target-class distribution, mitigating the concentration problem. Our evaluation of five popular datasets demonstrates that our attack, DRUPE, significantly reduces the distributional distance and concentration of the poisoned distribution compared to existing attacks. DRUPE successfully evades two state-of-the-art backdoor defenses in self-supervised learning and is robust against knowledgeable defenders.**

1. Introduction

Self-supervised learning (SSL) has recently attracted lots of attention. It is a learning paradigm that constructs a general model using massive amount of unlabeled data [10], [11], [56], [29], [13], [27]. The model can be further adapted/extended in various downstream tasks to achieve close to state-of-the-art performance [10], [11]. It has been widely employed in constructing foundation models in computer vision (CV) [10], [29] and natural language processing (NLP) [19], [39], [3] domains. For example, the popular model ChatGPT [54] is built on self-supervised learning, where a massive volume of text data (e.g., books and Wikipedia) were used during training, enabling it to predict the next word or sentence. CLIP [56] is trained on 400 million images and text pairs collected from the Internet. It can be used directly to accomplish downstream tasks (with exception performance), such as image classification, without the need to train on labeled data.

In this paper, we focus on self-supervised learning in the CV domain. The goal of such learning is to construct a general image feature extractor, called *image encoder*, such that semantically similar images have close-by *feature vectors* (also called *embeddings*) produced by the encoder. With the pre-trained encoder, a downstream task can mount a small classifier (with a few neural network layers) on top of it, called a *downstream classifier*, and train on the task-specific images. The image encoder can also be used in *zero-shot classification* setting. It is typically pre-trained with a text encoder, such as in CLIP [56]. A downstream task can pass the images to the image encoder and directly search for texts that have similar embeddings from the text encoder.

Security concerns are rising quickly with the increasingly growing applications of self-supervised learning (SSL) in building foundation models. They are particularly vulnerable to backdoor attacks [34], [9], [62], where an attacker injects a backdoor in the unlabeled data. A downstream task built on top of the backdoored encoder will misclassify any input stamped with the *backdoor trigger* to a *target label*. Researchers have shown that by poisoning just 0.01% of the unlabeled data [9], the attacker can achieve a high attack success rate. BadEncoder [34] injects a backdoor trigger on unlabeled input images and aims to minimize their distances to *reference inputs* in the feature space. The reference inputs are a few representative images of the target class in the downstream task, which are downloaded from the Internet. As such, any trigger-injected sample will have a similar embedding as those of the target-class inputs, resulting in the misclassification to the target label by the downstream classifier. It is also feasible to inject multiple backdoors in the unlabeled data [34], which can target a wide range of downstream tasks, deteriorating the security of SSL.

When examining existing backdoor attacks in self-supervised learning, it is noticeable that the poisoned data (i.e., trigger-injected inputs) is typically out-of-distribution compared to the clean unlabeled data. That is, the feature vectors of the poisoned samples are easily distinguishable from those of clean inputs. We call it the *out-of-distribution* property. This can be leveraged by defense methods for identifying poisoned inputs. Beatrix [50], a state-of-the-art backdoor input detection technique, conducts anomaly detection on feature vectors of input images. It can detect poisoned samples generated by existing attacks [34], [9]

*Equal contribution.

DOI 10.1109/SP54263.2024.00029

with more than 90% accuracy, according to our experiments in Section 6.3. In addition, existing backdoor attacks induce a high concentration of the poisoned distribution. The poisoned samples are all tightly clustered within a very small region in comparison to the clean data. This results in a high degree of pairwise similarity among the poisoned samples, providing an opportunity to defenders. DECREE [23] reverse-engineers a trigger for a given encoder, by minimizing the pairwise similarity of a set of samples (when the trigger is inserted). The encoder is considered backdoored if a small trigger can be found. Existing attacks [34], [9], [21], [69] are all detected by DECREE [23].

This paper reveals a key insight that the success of backdoor attacks in self-supervised learning is not necessarily dependent on the out-of-distribution property. We propose to transform poisoned samples into in-distribution data (w.r.t. the clean data) such that the backdoor information is not detectable in the feature space. To achieve this, we estimate the distribution of the clean data leveraging *Kernel Density Estimation (KDE)* [65]. KDE utilizes kernel functions such as Gaussian to approximate the local area around each known data point and combines them together to produce an estimate for the underlying distribution. We then move the poisoned samples closer to the clean distribution by reducing the difference between the two distributions, which is estimated by the *sliced-Wasserstein distance* [37], a metric for measuring distributional differences in a high-dimensional space. We also enforce the tightness of clean distribution to stay the same such that the encoder's behavior is not substantially changed on clean data when reducing the distributional difference. In addition, we reduce the concentration of the poisoned distribution by distributing the poisoned samples to a wider region, making it similar to that of the target-class distribution in the downstream task.

Our contributions are summarized as follows.

- We identify a key *out-of-distribution* property of existing backdoors attacks in self-supervised learning, which is the basis of existing defense techniques.
- We propose a novel method to reduce the distributional differences between the poisoned and the clean data, while preserving the clean distribution.
- We introduce an estimation method for approximating the target-class distribution with limited data.
- We implement an attack prototype called DRUPE (DistRibUtion Preserving backdoor attack in sElf-supervised learning). We evaluate the attack performance of DRUPE on five popular datasets, with various combinations of encoders and downstream classifiers. Compared to existing backdoor attacks, DRUPE achieves 10 times smaller distributional difference between the poisoned and the clean data, and 3 times smaller pairwise similarity among poisoned samples, while obtaining comparable benign accuracy and attack success rate on downstream tasks. In addition, DRUPE successfully evades two state-of-the-art backdoor defenses in self-supervised learning. The evaluation on knowledgeable defenders shows that DRUPE is robust against possible adaptive defenses.

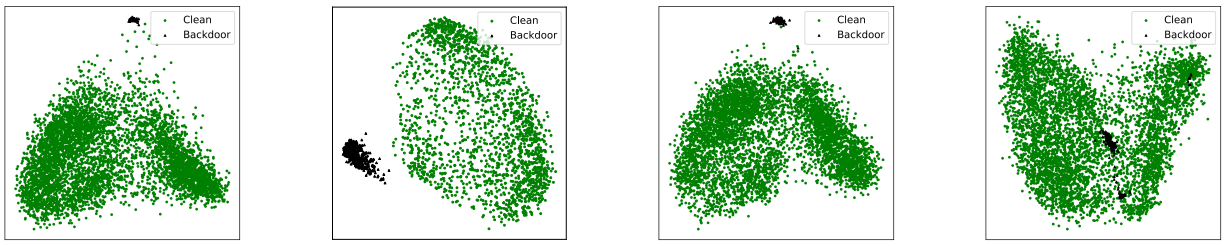
2. Background

2.1. Self-supervised Learning

The goal of self-supervised learning is using an extensive collection of unlabelled data (called *pre-training dataset*) to pre-train an image encoder [10], [56], [29], [27]. Once the image encoder is pre-trained, it can be leveraged to generate classifiers for various downstream tasks with a limited amount of labeled data (called *downstream dataset*). The encoder is a function $f : \mathcal{X} \rightarrow E$, where $\mathcal{X}$ is the input space containing different input samples and E is the embedding space containing the corresponding feature vectors. Among different types of self-supervised learning methods, contrastive learning (e.g., SimCLR [10], SimCLRv2 [11] and CLIP [56]) achieves state-of-the-art performance. Contrastive learning aims to produce an encoder that generates feature vectors with high cosine similarity for different augmented versions of the same input, while producing feature vectors having low cosine similarity for different inputs. In real world, the self-supervised encoders can be pre-trained by well-resourced service providers such as Google and OpenAI, and the downstream users can use the encoders released by service providers to construct downstream classifiers.

2.2. Backdoor Attacks in Self-supervised Learning

Backdoor attacks on self-supervised learning are particularly harmful as they can target any downstream tasks. A downstream classifier built on top of the backdoored encoder (by self-supervised learning) will misclassify any samples with the backdoor trigger to an attacker-specified target label. BadEncoder [34] injects backdoor behaviors into self-supervised learning by matching the embeddings (i.e., feature vectors from the encoder) of poisoned samples with a target embedding. It assumes the access to a few images from the target class of the downstream task, denoted as *reference inputs*. Any samples stamped with a backdoor trigger (e.g., a white square patch) shall have a similar embedding (from the encoder) to these reference embeddings. Carlini et al. [9] target multi-modal contrastive learning encoders, where two encoders are involved, an image encoder and a text encoder. They respectively project images and texts to the same embedding space, where the matched image and text shall have similar embeddings. For instance, an image of dog shall have a similar embedding to that of a text "*a photo of dog*". Carlini et al. inject a backdoor trigger on images, such as a white square on the top left of the input, and assign a target text to them. The attack only modifies the image encoder such that the embeddings of trigger-injected samples from the backdoored image encoder are similar to that of the target text from the text encoder. Any trigger-injected images will be captioned with the target text, analogous to misclassification to a target label in classification tasks. Other backdoor attacks in self-supervised learning [62], [41] utilize different triggers and usually have low attack performance.



(a) BadEncoder [34] (b) Carlini et al. [9] (c) WB attack [21] (d) DRUPE

Figure 1: PCA visualization of embeddings of clean and poisoned samples on backdoored models by different attacks

2.3. Backdoor Defenses in Self-supervised Learning

There are two state-of-the-art defenses that are designed for or applicable to self-supervised learning. DECREE [23] is a state-of-the-art encoder scanning technique. It works by reverse-engineers a trigger for a given encoder, by minimizing the pairwise similarity of a set of samples (when the trigger is stamped). The encoder is considered trojaned if such a trigger can be found. Beatrix [50] is a state-of-the-art poisoned sample detection technique. It leverages Gram matrix [57] to extract features from model activation values, and then conducts an anomaly detection using extracted features for distinguishing poisoned samples from clean ones. Beatrix can be adapted to detect poisoned samples in encoders by inspecting the embeddings of given input samples. We use the above two state-of-the-art defense techniques to evaluate existing and our newly proposed attacks.

3. Threat Model

We describe the attacker's objective, knowledge, and capabilities in this section.

Attack Objective. The attacker aims to inject backdoors in a pre-trained image encoder by self-supervised learning such that pasting the backdoor trigger on any input samples can cause a downstream classifier built on this encoder to predict attacker-chosen target label. The attack shall not affect the normal functionality of the backdoored encoder. That is, any downstream classifier built on top of the backdoored encoder should have the same classification performance as that built on a clean encoder. The injected backdoor trigger ought to be small and not affect the main content of the original input. Otherwise, it can be easily detected by human inspectors or automatic tools.

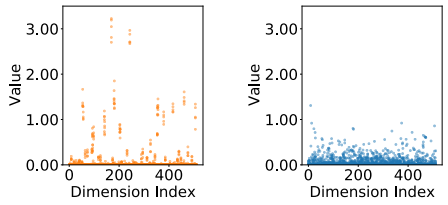
Attack Knowledge and Capabilities. We consider the attacker has access to a clean pre-trained encoder following existing work [34]. The attacker also has access to a set of unlabeled data, called *shadow dataset*. However, the attacker has no knowledge of the downstream classifier, including its training data, model architecture and weights, training hyper-parameters, etc. He/she also cannot interfere with the training procedure of the downstream task. We assume the attacker can collect a very few images from the Internet that are from the target class of a downstream task as in existing work [34], called *reference inputs*. The attacker can control the training process for constructing a backdoored encoder.

4. Motivation

An implicit requirement for backdoor attacks is that a poisoned sample with the injected trigger ought to be indistinguishable from a clean input. This condition is commonly satisfied in the input space by constraining the modification on the input, such as using a small patch pattern or imperceptible perturbations. However, the feature vectors of poisoned samples (generated by the backdoored encoder) remain distinct from those of clean data, as they were intentionally designed to match a (few) target embedding(s). In particular, the current practice of attacks in self-supervised learning induces strong backdoor signals in the embedding space, causing poisoned samples lying outside of the clean data distribution. We call it the *out-of-distribution* property. In addition, the poisoned distribution is highly concentrated, resulting in a high degree of pairwise similarity among the poisoned samples.

Figure 1a shows the distributions of clean and poisoned data on a backdoored encoder by BadEncoder [34]. The figure is obtained as follows. We pass the clean pre-training data[1] and the poisoned samples to the encoder and acquire their feature embeddings. We then apply principle component analysis (PCA) to reduce the dimensionality of those embeddings and plot them in a 2-dimensional figure. The green dots and the black dots represent the clean pre-training data and the poisoned data, respectively. It can be clearly observed that poisoned samples are very different and far from the clean data distribution. We further use the Wasserstein metric [78], which is a distance metric for quantifying distributional difference, to measure the distance between the clean and poisoned distributions. For a backdoored model by BadEncoder [34], the distributional distance between a clean data set and the poisoned set is 0.1073, which is more than 14 times of the distance between two randomly selected clean data subsets (0.0075). Evidently, the poisoned samples are outliers in the embedding space. Figure 1b[2] and Figure 1c present the data distributions for

1. The clean unlabeled data used for pre-training the encoder. It is different from the labeled dataset used for training a downstream task.
2. We only show successful poisoned samples in the figure.



(a) Poisoned embeddings (b) Clean embeddings

Figure 2: Embedding values at different dimensions

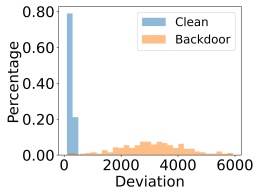
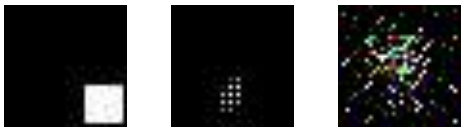


Figure 3: Beatrix's deviation distribution



(a) Injected trigger (b) Inverted trigger on BadEncoder ($L^1 = 13.55$) (c) Inverted trigger on benign encoder ($L^1 = 168.82$)

Figure 4: Visualizations of injected and inverted triggers

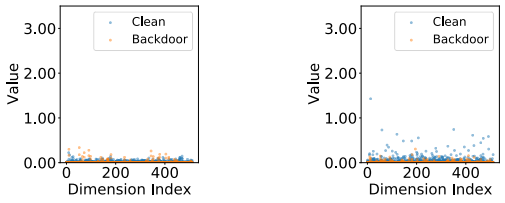
a few other existing attacks on self-supervised learning. The observations are consistent across various attacks.

The out-of-distribution property of existing attacks indicates that the feature vectors of poisoned samples are abnormal, providing strong signals to defense methods for detecting poisoned samples and/or backdoored encoders. Figure 2 presents the embedding values for a set of clean and poisoned samples (when fed to a backdoored encoder). The x-axis denotes the embedding dimension index (512 dimensions in total), and the y-axis shows the value for the corresponding embedding dimension. We can see in Figure 2a that there are a few dimensions having relatively large values (> 1.0) for the embeddings of trigger-injected samples, while in Figure 2b almost all the dimensions have values smaller than 1.0 for clean embeddings. Such feature abnormality provides sufficient information to existing defense. Figure 3 displays the results of Beatrix [50] (a poisoned sample detection technique) on detecting BadEncoder's poisoned samples. The x-axis denotes the deviation of an input's Gramian feature (i.e., feature outlier) and the y-axis the percentage of the corresponding deviation value. Observe that there is a clear separation between clean and poisoned samples. The deviation values are much larger for poisoned samples, which are utilized by Beatrix for detecting malicious inputs.

In addtion, as shown in Figure 1a, the poisoned distribution is tightly clustered within a very small region. When we measure the cosine similarity among poisoned samples, the average pairwise similarity is 0.99. This is much higher than that of clean inputs (0.27). The high concentration of the poisoned distribution makes detecting backdoored encoders (i.e., determining if an encoder contain any backdoor) less challenging. By applying DECREE [23] (a encoder scanning technique) on a trojaned encoder attacked by BadEncoder, it can easily generate a small trigger for the encoder that can induce a high attack success rate (ASR) on downstream classifiers. Figure 4a shows the original injected trigger and Figure 4b the inverted trigger. Observe that the inverted trigger is similar to the injected one. The location difference is due to the data augmentation applied during self-supervised learning when injecting the trigger. For comparison, we also show an inverted trigger on a clean encoder in Figure 4c. The trigger is much larger than that on the backdoored model, indicating backdoored encoders by existing attacks can be easily identified. The observation holds for other attacks as well. Details are elided.

Our Idea. We observe that the out-of-distribution property is not a crucial factor in backdoor attacks on self-supervised learning. Our overarching idea is hence to transform poisoned samples into in-distribution data (w.r.t. the clean unlabeled and target-class data) such that the backdoor signal can be substantially reduced in the feature space. Specifically, we estimate the distribution of the clean data using *Kernel Density Estimation* (KDE) [65]. KDE is a widely used method for estimating the probability density function based on a finite set of observations, and is particularly useful for high-dimensional data. It leverages kernel functions such as Gaussian to approximate the local region around each data point and combines them together to produce an estimate of the underlying probability distribution (see details in Section 5.1). The poisoned samples are then pulled closer to the clean distribution by reducing the distributional difference between the two distributions, which is estimated by the *sliced-Wasserstein distance* [37]. It is a metric for measuring distributional difference between two data distributions when their density functions are unknown. It is a variant of the Wasserstein distance [78] and well-suited for measuring high-dimensional data. However, directly minimizing the distance between the poisoned and clean distributions substantially changes an encoder's behavior on clean data, and in the mean time, may not yield in-distribution embeddings for poisoned data. In order to prevent such undesirable changes, a constraint is added to enforce the tightness of clean distribution, ensuring it stays the same. As such, *the poisoned samples being inside the distribution of clean data mitigates the abnormality of them in the feature space, undermining the basis of defense techniques.* In addition, we reduce the concentration of the poisoned distribution to make it distribute across a wider region, being similar to that of the target-class distribution in the downstream task. Figure 1d presents the result of our attack DRUPE. Observe that all the poisoned samples (the black dots) are inside the clean distribution (the green dots) and not distinguishable from the clean data. They also distribute across a wider area instead of concentrating within a small region.

Existing works [69], [21] have proposed to reduce statistical differences between clean and poisoned samples in the feature space for image classification tasks. In particular, AdvEmbed [69] uses an adversarial network to differentiate whether an input is malicious to assist the attack, so that the poisoned model will produce similar embeddings for clean and poisoned data. Our attack DRUPE is substan-



(a) L^2 norm constrained (b) DRUPE

Figure 5: Embedding values at different dimensions of backdoored encoders by different attacks

tially different from AdvEmbed [69] where we use sliced-Wasserstein distance to reduce the distributional difference of clean and poisoned data. DRUPE does not use an adversarial network that requires additional computational resources and is often hard to train. Additionally, DRUPE addresses the concentration of poisoned distribution (particularly in self-supervised learning), a prominent challenge that was not considered in exiting works [69], [21]. In Section 6.3, we adapt AdvEmbed [69] to self-supervised learning and the result shows that it can be detected by DECREE [23] as it does not consider the concentration of poisoned distribution.

5. Attack Design

An ideal attack in self-supervised learning shall have poisoned samples indistinguishable from clean inputs in the feature space. In other words, the distributional difference (measured in the embedding space) between the poisoned distribution and the clean distribution shall be small. We elaborate details of achieving this goal in Section 5.1. Additionally, the poisoned samples shall distribute across the entire target-class distribution of the downstream task such that the poisoned distribution is less concentrated, lowering the pairwise similarity among the poisoned samples. Details are discussed in Section 5.2.

5.1. Transforming Poisoned Samples into In-Distribution Data

We observe that, in Section 4, the embeddings of poisoned samples have a few large values, compared to those of clean inputs as shown in Figure 2. This causes the poisoned data to be located outside of the clean distribution. A straightforward idea is to reduce the dimensions with large values in the embeddings of poisoned samples. One can address this by enforcing the L^2 norm of poisoned embeddings to be similar to that of clean embeddings. This method effectively suppresses large embedding values in the poisoned samples. However, it also reduces the embedding values for clean inputs. As shown in Figure 5a, the embedding values of both poisoned and clean samples are smaller than those in Figure 2. Despite this reduction, a few orange outliers are still visible on top of the clean distribution. When inspecting the distributions of poisoned and clean samples, we observe that the poisoned samples are still out-of-distribution w.r.t. the clean data. We hence resort to minimizing the distributional difference between poisoned and clean distributions.

Wasserstein distance is a widely used metric for measuring distributional difference between two latent distributions whose common support or density functions are unknown [78]. As our goal is to minimize the distributional difference between poisoned and clean samples, Wasserstein distance can be leveraged to smoothly address the issue. We consider the 2-Wasserstein distance, also known as the earth mover's distance.

$$\mathcal{W}(\zeta,\tau)=\left(\inf_{\psi\in\Pi(\zeta,\tau)}\int_{(u,v)\sim\psi}p(u,v)\|u-v\|_2\,dudv\right)^{1/2}, \tag{1}$$

where ζ and τ are marginal probability measures of the clean and poisoned data. They are defined by empirical samples of clean and poisoned embeddings. ψ is the joint distribution of the clean and poisoned data. The infimum $\inf$ denotes the greatest lower bound of the computed distance on the joint distribution. The integral term calculates the sum of the distance between every two data points (u,v) (one clean and one poisoned) drawn from the joint distribution ψ. $p(u,v)$ is the probability of jointly drawing the two samples. $\|u-v\|_2$ is the L^2 distance between the two samples.

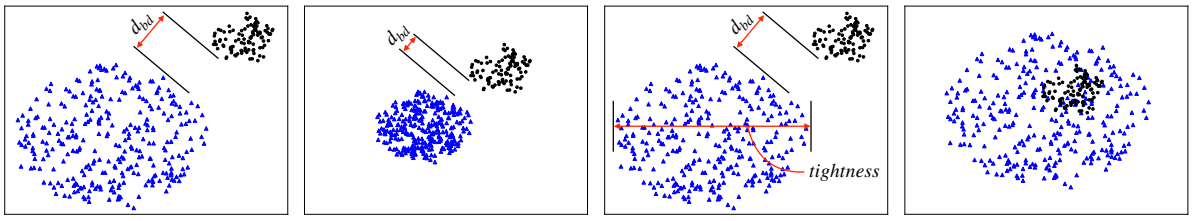
Directly estimating the 2-Wasserstein distance is challenging, especially in high-dimensional settings (e.g., 512-dimensional embeddings in our case), as one needs to compute the infimum in the above equation. Fortunately, the sliced-Wasserstein distance [37], a variant of the Wasserstein distance, is well-suited for measuring high-dimensional data. It first slices/projects the high-dimensional data into lower-dimensional subspaces, and then computes the Wasserstein distance separately for each subspace. The projection is carried out by sampling a unit vector uniformly at random in the original high-dimensional space (e.g., resulting in a 512-dimensional unit vector), and then computing the dot product of each embedding with the selected unit vector. We use one-dimensional subspaces during the measure as there is a closed-form solution for the 2-Wasserstein distance between two one-dimensional Gaussian distributions.

$$\mathcal{W}_{sliced}(\zeta,\tau)=\left(\frac{1}{S}\sum_{s=1}^{S}\int_0^1\|F_c^s(z)-F_b^s(z)\|_2\,dz\right)^{1/2}, \tag{2}$$

where S is the number of one-dimensional directions (denoted by the randomly sampled unit vectors). F_c^s and F_b^s represent the projections of the clean and poisoned embeddings into one-dimensional data points along the direction of slice s, respectively.

Figure 6a illustrates the idea. The blue triangles denote the clean pre-training inputs, while the black dots represent the poisoned samples. It is evident that the black dots are outliers and far from the blue distribution. The Wasserstein distance[3] between the two distributions is denoted by d_{bd}.

3. This actually refers to the sliced 2-Wasserstein distance. We use this term for brevity.



(a) Out-of-distribution (b) Distributional difference (c) Tightness of the clean distribution (d) Indistinguishable distributions

Figure 6: Illustration of *the clean unlabeled data distribution* (the blue triangles) and the poisoned sample distribution (the black dots)

One solution is to directly minimize the distance d_{bd}, similar to existing work [21]. However, during reducing the Wasserstein distance, it substantially changes the encoder weights, leading to a more concentrated clean distribution. As shown in Figure 6b, the distance d_{bd} between poisoned and clean samples indeed decreases. However, the blue triangles are also much closer to each other compared to those in Figure 6a. This makes the poisoned distribution still relatively far from the clean distribution, failing the goal of evading defense techniques. Moreover, further reduction of the Wasserstein distance negatively affects the backdoored encoder's utility on normal functionalities.

We propose to estimate the clean distribution using the given clean data samples. As such, the change of the backdoored encoder on clean data can be effectively monitored and controlled during the attack. In particular, we measure the tightness of the clean distribution at each training iteration and ensure it stays the same. Figure 6c demonstrates the idea. The horizontal red arrow inside the blue triangle area quantifies the tightness of clean distribution. During the process of minimizing the distributional difference d_{bd} between poisoned and clean distributions, the clean distribution is largely maintained by constraining its tightness. The resultant distributions are illustrated in Figure 6d. The black dots are all located inside the blue triangle region and not distinguishable from the clean data. The following elaborates the details of estimating the clean distribution and measuring its tightness.

Kernel Density Estimation (KDE) [65] is a non-parametric method used for estimating the probability density function of a random variable based on a finite set of observations. It is widely used in various fields, such as statistics [70], machine learning [40], and signal processing [15], and particularly useful for high-dimensional data. We leverages KDE to estimate the distribution of the clean data. Intuitively, KDE places a kernel function around each known data point, assigning more weight to nearby points and less weight to farther points. These kernel functions are then combined and smoothed to produce an estimate of the underlying probability distribution. Formally,

$$\widehat{p}_h(e) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{e - e_i}{h}\right), \tag{3}$$

where n denotes the number of input samples. h is the bandwidth parameter, which controls the amount of smoothing applied to the density estimate. A smaller value results in a more variable density estimate, and a larger value yields a smoother estimate. $K(\cdot)$ represents the kernel function, whose choices include Gaussian, Epanechnikov, and uniform, etc. Variable e_i denotes the embedding of the i-th clean input and $e - e_i$ is the element-wise difference. We use the Gaussian kernel in this paper. Given the estimated probability distribution of the clean data $\widehat{p}_h(e)$, we measure its tightness. Particularly, the variance of the estimated distribution is utilized as the tightness of clean distribution.

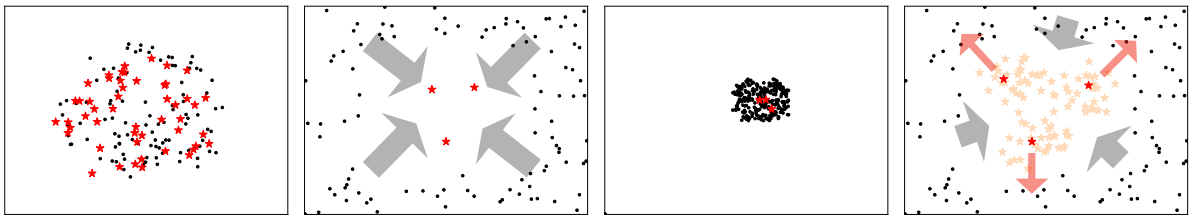
$$\sigma^2 = \frac{\int_{-\infty}^{\infty} (e - \mu)^2 \, \widehat{p}_h(e) \, de}{\int_{-\infty}^{\infty} \widehat{p}_h(e) \, de}, \tag{4}$$

where $\mu = \int_{-\infty}^{\infty} e \, \widehat{p}_h(e) \, de$. Note that the total density estimate is the denominator in the above equation. This is because the kernel density estimate is based on a finite set of samples, which is subject to sampling variability and noise. Dividing by the total density estimate ensures that the variance is not biased by regions where the density estimate is low or noisy. The computed tightness is used to enforce the clean distribution staying the same during the attack. See details in Algorithm 1.

Figure 5b shows the embedding values of poisoned and clean samples on our backdoored model. Observe that the embedding values of poisoned samples are completely indistinguishable from those of clean inputs. The distributions shown in Figure 1d also validate this as the poisoned samples all locate inside the clean distribution.

5.2. Reducing the Concentration of the Poisoned Distribution

Backdoor attacks in self-supervised learning aims to maximize the cosine similarity between the embeddings of poisoned samples and those of the reference inputs (from the target class of a downstream task). An ideal scenario for the attacker would be to have access to all target-class images from the downstream task, in which case he/she can straightforwardly match each poisoned sample with a target-class image. Figure 7a illustrates the concept. The black dots denote the poisoned samples and the red stars denote the target-class inputs. In this ideal scenario, poisoned samples are completely intertwined with the target-class inputs. This means that the poisoned distribution covers the entire target-class space or surface, making it indistinguishable from the target-class distribution.



(a) Ideal poisoned distribution (b) Limited target samples (c) Concentrated distribution (d) Approximated target distribution

Figure 7: Illustration of the poisoned sample distribution (the black dots), *the downstream target-class data distribution* (the red stars), and expanded reference inputs (the pink stars)

In practice, the downstream dataset may be private or unavailable [34], so the attacker may need to resort to downloading representative images of the target class from the Internet as reference inputs. For instance, BadEncoder [34] utilizes three reference inputs downloaded from the Internet to carry out the attack, which is illustrated in Figure 7b. The three reference inputs are denoted by the red stars located within the target-class space. The attack aims to pull the black dots (i.e., the poisoned samples) closer to the red stars (i.e., representative images of the target class), as illustrated by the four gray arrows in Figure 7b. By reducing the distance of those poisoned samples to the reference inputs, the results shown in Figure 7c are obtained. It can be observed that both the black dots and red stars are concentrated in a very small region of the target-class space. We call the resultant distribution of poisoned samples a *concentrated distribution*. Note that during the attack, there are no constraints on the similarity among the reference inputs themselves (although their embeddings may be optimized to be similar on the backdoored encoder and a clean encoder). A poisoned sample x' is pulled closer to a specific reference input r_1. It is possible for this poisoned sample x' to also be optimized to be close to another randomly picked reference input r_2, indirectly pulling these two reference inputs closer to each other. This optimization strategy results in a highly concentrated distribution for poisoned samples, leading to high pairwise similarity. Such a property can be easily leveraged by defense methods to effectively detect backdoored encoders [23].

An alternative way is to pull each poisoned sample to a predetermined reference input instead of a randomly chosen one. However, this approach heavily relies on the initial selection of reference inputs, which is unreliable as the attack has no knowledge of the true target-class distribution. Directly reducing the pairwise similarity of poisoned samples can indeed mitigate the concentration problem. However, it also leads to unsatisfactory attack success rate [23]. This is because it aims to push poisoned samples away from each other and hence away from the reference inputs, which contradicts the objective of the attack.

We propose to approximate the target-class distribution using the limited reference inputs. To achieve this, we first expand the given reference inputs to a larger set by generating more similar samples with the same main content. These samples are passed to a clean encoder to obtain their embeddings, which approximates the feature distribution of the target class. We then force the embeddings of these reference samples on the backdoored encoder to be similar to those on the clean encoder. Besides, we also maximize the distance among those samples such that they are not to close to each other. In Figure 7d, the red stars denote the given reference inputs. The pink stars represent samples expanded from the given reference inputs. The red arrows denote pushing the reference inputs away from each other. We discuss details in the following.

In order to produce more samples from the given reference inputs, one can use generative models such as generative adversarial networks (GANs) [26] or diffusion models [31]. GANs and diffusion models are powerful generative models that can produce realistic images. However, those models typically only generate images that are similar to their training data distribution. In our setting, the goal is to generate more images for a specific class, for instance, medical x-ray scans. There may not exist a pre-trained generative model for such a task. Training those models from scratch is both data-intensive and resource-intensive [25], [31]. We hence make use of common data augmentation techniques to expand the dataset, which is lightweight and easy to generate similar images. The advantage of using data augmentation is that the main content features are preserved while other features such as color, view angle, size, etc. are diversified. Specifically, we adopt two data augmentation methods: random cropping and color jittering. For each reference input, we apply the augmentation methods with randomly chosen parameters (e.g., output size of the crop) and generate 50 similar samples by default.

With the expanded reference set, we leverage a clean encoder to approximate the target-class distribution. Particularly, we force the embeddings on the backdoored encoder to be similar to those on the clean encoder.

$$\max \frac{1}{|\widehat{R}|} \sum_{i=0}^{|\widehat{R}|} \cos\left(f(r_i), f^{\star}(r_i)\right), \tag{5}$$

where $\widehat{R}$ is the expanded reference set; $\cos(\cdot)$ denotes the cosine similarity; f is the backdoored encoder; $f^{\star}$ is a clean encoder. We also maximize the distance among those samples such that they are not to close to each other as illustrated in Figure 7d. Specifically, we minimize the similarity (i.e.,

maximize the distance) of every two samples expanded from two different original reference inputs. Formally,

$$\min \frac{1}{N} \sum_{\substack{(r_i^a, r_j^b) \sim (R^a, R^b) \\ R^a, R^b \text{ from } R}} \cos\big(f(r_i^a), f(r_j^b)\big), \tag{6}$$

where N is the number of samples used. R denotes the set of original reference inputs, and R^a and R^b are the sets expanded from two different reference inputs in R through data augmentation, respectively; r_i^a and r_i^b are the samples from the two sets; f denotes the encoder. Note that this may contradict Equation 5. We leverage a threshold (e.g., 0.9) for satisfying Equation 5 during poisoning such that it can achieve a good balance between the two goals.

To enable a successful attack, we need to pull poisoned samples close to the reference inputs in the embedding space. At the beginning of the attack, the trigger-injected samples are randomly distributed across the entire space (as shown by the black dots at the outer edge of Figure 7d). Intuitively, it does not matter which reference input a poisoned sample should be close to, as long as it is similar to one of the reference inputs. However, moving a poisoned sample far away from its original location requires substantial changes to the encoder, which may slow down the convergence of the attack and affect the normal functionality of the encoder. We choose to pull the poisoned samples to their nearest (expanded) reference input to ensure minimal changes to the encoder. Formally,

$$\max \frac{1}{|X'|} \sum_{x \sim X'} \max_{r \sim \widehat{R}} \Big[\cos\big(f(x \oplus \Delta), f(r)\big)\Big], \tag{7}$$

where X' is a set of samples for poisoning, $\widehat{R}$ is the expanded reference set, and $\oplus$ is an operator for injecting the trigger Δ onto the input x. Note that each poisoned sample is pulled closer to its nearest reference input (see the second $\max$ operation). Observe in Figure 7d, the gray arrows around the red/pink stars indicate moving the poisoned samples to their closest (expanded) reference input. In addition, we also force poisoned samples to be dissimilar to each other. As such, the poisoned samples will not end up becoming a few concentrated distributions near those reference inputs.

$$\min \frac{1}{|X'|} \sum_{x_j, x_j \sim X'} \cos\big(f(x_i \oplus \Delta), f(x_j \oplus \Delta)\big). \tag{8}$$

As our attack approximates the target-class distribution using the available reference inputs, the resulting poisoned samples are distributed across the entire target-class space, making them indistinguishable from the target-class data.

Attack Secrecy Against Knowledgeable Defenders. Although our attack does not encourage pairwise similarity among poisoned samples, the poisoned samples are still pulled close to their corresponding reference inputs as per the attack goal. It is possible that poisoned samples close to the same reference input may have high pairwise similarity. This gives defenders an opportunity to expose our backdoored encoder through trigger inversion as done by existing work [23]. In the following, we analyze the probability of detecting our backdoored encoder, given sufficient information regarding the attack.

Assume the defender has the knowledge that the attack employs k reference inputs, and obtains the clean version of the poisoned samples. The defender does not know what reference inputs are used, which downstream task and target class are chosen by the attacker. We further assume that the defender needs at least m samples in the cluster of the same reference input to invert a useful trigger. Trigger inversion in self-supervised learning requires at least two samples to calculate the pairwise similarity [23]. Note that m cannot be too small, as it will be easy for optimizers to find adversarial perturbations, which is a simpler task than finding a trigger that leads to high pairwise similarity for m samples. Based on our experience, a typical value for m is 15.

Based on the above assumption, we analyze the attack secrecy of finding the trigger in a backdoored encoder. We follow the detection procedure of existing work [23], which determines whether an encoder is backdoored by inverting a (small) trigger using the m samples. Let the number of clean samples that the defender has is n. The number of samples in the cluster of each reference input is then $\frac{n}{k}$. The probability of successfully detecting the backdoor is hence computed as follows.

$$\mathbb{P} = \frac{n/k - 1}{n - 1} \cdots \frac{n/k - m + 1}{n - m + 1}. \tag{9}$$

Assume $\frac{n}{k} \gg m$. We have

$$\mathbb{P} \approx (\frac{1}{k})^{m-1}. \tag{10}$$

Suppose the number of reference inputs $k = 3$ and the number of samples needed for trigger inversion $m = 15$, the probability to detect the backdoor is about $2e^{-7}$, which is very low. Our evaluation in Section 6.3 also demonstrates that state-of-the-art defenses cannot detect our attack.

The clustering of poisoned samples around the (expanded) reference inputs during the attack may also impact the behavior of the encoder on clean inputs. Therefore, there is a possibility that directly clustering clean inputs on the backdoored encoder can expose unique patterns in comparison to those on clean encoders. The empirical results in Section 6.4 show that there is no distinction between our backdoored encoders and clean encoders. Inverting triggers on any clusters does not reveal the backdoor as well. Please see the detailed discussion in Section 6.4.

5.3. Detailed Methodology

The detailed attack procedure of DRUPE is presented in Algorithm 1. DRUPE first initializes the backdoored encoder weights from a clean encoder (line 2), and also constructs the poisoned set (line 3) and the expanded reference set (line 4). At each training epoch, the distributional difference between poisoned samples and clean data is measured (line 7). DRUPE also considers the tightness of the clean

Algorithm 1 Distribution Preserving Backdoor Attack

Input: Benign encoder $f^\star$, shadow dataset $\mathcal{D}$, reference input set R, backdoor trigger Δ
Output: Backdoored Encoder: f

1: **function** DRUPE($f^\star, \mathcal{D}, R, \Delta$)
2: $f \leftarrow f^\star$
3: $\mathcal{D}' \leftarrow$ Inject trigger Δ on $\mathcal{D}$
4: $\widehat{R} \leftarrow$ Augment samples in R
5: **for** $iter$ **in** $0...max\_epochs$ **do**
6: **/*** Reduce distributional difference ***/**
7: $d_{bd} \leftarrow$ Sliced Wasserstein distance between clean data $f(x), \forall x \in \mathcal{D}$ and poisoned data $f(x'), \forall x' \in \mathcal{D}'$ ▷ Equation 2
8: $\sigma^2 \leftarrow$ Variance(KDE($\mathcal{D}$)) ▷ Equation 4
9: $\mathcal{L}_{diff} = d_{bd}/\sigma$
10: **/*** Mitigate the concentration ***/**
11: $\mathcal{L}_{approx} \leftarrow$ Similarity of reference inputs on backdoored and clean encoders ▷ Equation 5
12: $\mathcal{L}_{ref} \leftarrow$ Similarity of reference inputs in $\widehat{R}$ ▷ Equation 6
13: $\mathcal{L}_{poi} \leftarrow$ Pairwise similarity of poisoned samples ▷ Equation 8
14: $\mathcal{L}_{conc} = -\mathcal{L}_{approx} + \mathcal{L}_{ref} + \alpha \cdot \mathcal{L}_{poi}$
15: **/*** Achieve the attack goal ***/**
16: $\mathcal{L}_{asr} \leftarrow$ Similarity between poisoned samples and their nearest reference inputs ▷ Equation 7
17: **/*** Maintain normal functionality ***/**
18: $\mathcal{L}_{func} = \frac{1}{|\mathcal{D}|}\sum_{i=0}^{|\mathcal{D}|} \cos\left(f(x_i), f^\star(x_i)\right), \forall x_i \in \mathcal{D}$
19: **/*** Update encoder weights ***/**
20: $\mathcal{L} = \beta \cdot \mathcal{L}_{diff} + \mathcal{L}_{conc} - \mathcal{L}_{asr} - \mathcal{L}_{func}$
21: $\theta_f = \theta_f - lr \cdot \frac{\partial \mathcal{L}}{\partial \theta_f}$

distribution to avoid the negative impact on clean data (line 8). The relative difference of the two measures is utilized as the loss for reducing the distributional difference (line 9). In order to reduce the concentration of the poisoned distribution, DRUPE maximizes the similarity of expanded reference inputs on backdoored and clean encoders in the embedding space (line 11). It also maximizes the distance among reference inputs (line 12) and among poisoned samples (line 13). The poisoned samples are pulled close to their nearest reference inputs (line 16) to satisfy the attack goal. Finally, DRUPE keeps the normal functionality of the backdoored encoder by comparing its produced embeddings on clean data to those from a clean encoder (line 18). The weights of the backdoored encoder are updated based on the final loss using gradient descent (lines 20-21). The two hyper-parameters α (line 14) and β (line 20) are utilized to balance loss values among different loss terms. They are dynamically adjusted during the attack using the loss coefficient adjustment method described in Neural Cleanse [79].

6. Evaluation

We evaluate the attack performance of DRUPE on five datasets and compare to existing attacks in Section 6.2. We present the evaluation results against two state-of-the-art defense techniques in Section 6.3. We also study the setting with knowledgeable defenders in Section 6.4. The ablation study is carried out in Section 6.6.

6.1. Experiment Setup

Datasets. Five datasets are employed in the experiments including CIFAR-10 [38], STL-10 [17], GTSRB [71], SHVN [51], and ImageNet [61]. More details about the used datasets can be found in Appendix A.

Models and Architectures. Following prior work BadEncoder [34], the contrastive learning algorithm SimCLR [10] is used. For CIFAR-10 and STL-10, we use ResNet18 [30] model architecture. For ImageNet, ResNet50 [30] architecture is used. We also study a recent self-supervised learning algorithm, Mugs [95], and ViT [22] model architecture (See Appendix E). In Section 6.5, we evaluate DRUPE on the CLIP [56] model with ResNet50 architecture.

Baseline Attacks. BadEncoder [34], a representative backdoor attack on self-supervised learning, is leveraged as our baseline. Carlini et al. [9] target multi-modal contrastive learning encoders and inject backdoors in the image encoder. WB attack [21] and AdvEmbed [69] are backdoor attacks on classification tasks that improve the backdoor stealthiness in the feature space. We extend them to self-supervised learning. Details of these attacks are explained in Section 2 and Section 4. The details about the implementation of DRUPE is described in Appendix B. For a fair comparison, we use the same reference samples and triggers adopted by existing work [34].

Defense Methods. Beatrix [50] is a state-of-the-art defense technique for detecting poisoned samples. DECREE [23] is a state-of-the-art backdoor scanning method. Based on our observation in Section 4 that poisoned samples are out-of-distribution, we also extend DECREE to include a Wasserstein loss during trigger inversion. This enhances the defense capability by leveraging both the out-of-distribution property and the concentration of the poisoned distribution (See Appendix C). We call it *DECREE-Wass*.

Evaluation Metrics. We consider the following five measurement metrics in our evaluation.

- *Benign Accuracy (**BA**)* is calculated by dividing the number of correctly classified clean samples by the total number of clean samples. It measures the performance of a model on its benign task.
- *Attack Success Rate (**ASR**)* is computed by dividing the number of successfully attacked samples by the total number of backdoor samples.
- *Embedding Similarity of Clean Samples (**SIM-C**)* is defined as the average pairwise cosine similarity among the embeddings of clean samples.
- *Embedding Similarity of Backdoor Samples (**SIM-B**)* is defined as the average pairwise cosine similarity among the embeddings of backdoor samples.
- *Relative Distributional Distance Between Clean Embeddings and Poisoned Embeddings (**DD**)* is obtained by the sliced-Wasserstein distance [37] between the embeddings of clean samples and those of poisoned samples over the standard deviation of the clean distribution.

Metrics Used by Defense Methods. Two types of backdoor defense are considered in the evaluation: detecting backdoored encoders and detecting poisoned samples. True Positive (**TP**) represents the number of correctly identi-

TABLE 1: Comparison of attack performance on different datasets

Pre-training Dataset	Downstream Dataset	Benign	BadEncoder [34]					WB attack [21]					DRUPE				
		Acc	BA↑	ASR↑	SIM-C	SIM-B↓	DD↓	BA↑	ASR↑	SIM-C	SIM-B↓	DD↓	BA↑	ASR↑	SIM-C	SIM-B↓	DD↓
CIFAR-10	GTSRB	80.76%	80.97%	99.31%	0.27	0.99	12.95	79.85%	98.55%	0.25	0.97	1.98	80.35%	97.22%	0.23	0.34	0.92
	STL-10	76.10%	75.98%	99.13%	0.25	0.99	10.60	73.77%	98.55%	0.27	0.99	2.29	74.43%	96.72%	0.25	0.35	0.68
	SVHN	61.52%	77.15%	99.46%	0.27	0.99	13.59	76.86%	95.58%	0.26	0.99	2.11	76.02%	96.23%	0.25	0.38	0.85
STL-10	GTSRB	76.83%	76.60%	99.23%	0.37	0.99	11.05	75.71%	97.64%	0.34	0.99	3.12	75.93%	96.09%	0.35	0.36	0.74
	SVHN	56.90%	75.97%	97.85%	0.42	0.99	17.50	75.87%	96.50%	0.33	0.99	2.25	75.64%	96.68%	0.36	0.37	0.63
	CIFAR-10	85.67%	86.08%	98.79%	0.35	0.99	12.49	82.52%	99.29%	0.35	0.99	2.14	84.36%	98.39%	0.35	0.37	0.61
ImageNet	GTSRB	79.61%	79.80%	99.55%	0.34	0.93	9.53	79.02%	84.98%	0.31	0.89	1.57	79.66%	99.47%	0.35	0.45	0.87
	STL-10	94.58%	94.86%	99.01%	0.35	0.87	10.32	93.37%	98.55%	0.32	0.91	2.36	94.25%	99.31%	0.34	0.42	0.78
	SVHN	73.67%	84.61%	98.72%	0.34	0.91	12.47	84.81%	89.36%	0.32	0.90	1.72	84.10%	98.79%	0.35	0.46	0.88
Average	-	76.18%	81.31%	99.01%	0.33	0.96	12.27	80.20%	95.44%	0.30	0.96	2.17	80.52%	97.65%	0.31	0.39	0.77

TABLE 2: Comparison with AdvEmbed [69]

Attack	BA↑	ASR↑	SIM-C	SIM-B↓	DD↓
AdvEmbed	77.63%	94.88%	0.31	0.98	2.67
DRUPE	80.35%	97.22%	0.23	0.34	0.92

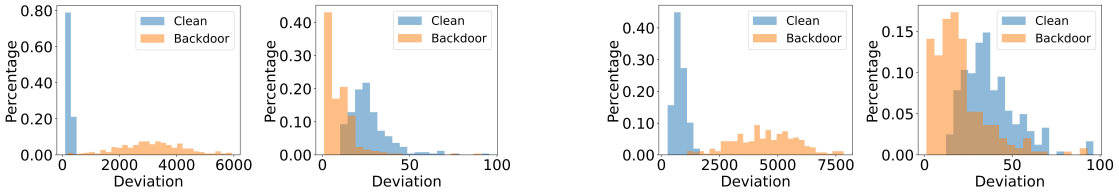
fied backdoored encoders/poisoned samples. False Positive (**FP**) denotes the number of correctly classified clean encoders/samples. False Positive (**FP**) means clean encoders/samples are wrongly classified as backdoored encoders/poisoned samples. False Negative (**FN**) is the reverse. Detection Accuracy (**Acc**) is the classification accuracy on distinguishing backdoored ones and clean ones.

6.2. Attack Effectiveness

Comparison to Existing Attacks. We evaluate the attack performance of DRUPE on various datasets and compare it to baseline BadEncoder [34]. Our poisoning rate is consistent with BadEncoder. Table 1 presents the results. The first column denotes the pre-training datasets that the backdoored encoders are trained on. The second column shows the downstream datasets that are used for constructing the downstream classifiers. Column Benign denotes the test accuracy of the downstream classifier using clean encoders. Columns BA and ASR represent the benign accuracy and the attack success rate of the downstream classifier. Columns SIM-C and SIM-B denote the embedding similarity of clean and poisoned samples, respectively. SIM-B shall be as close to SIM-C as possible. Column DD is the relative distributional distance between poisoned and clean samples, which is the smaller the better.

It can be observed that DRUPE achieves comparable BA and ASR with BadEncoder [34] and WB attack [21]. The average BA/ASR are 81.31%/99.01% by BadEncoder, 80.20%/95.44% by WB attack, and 80.52%/97.65% by DRUPE, respectively. Note that the low accuracy on SVHN using clean encoders is consistent with that reported in BadEncoder [34]. SVHN is noisy with distracting digits. Backdoor attacks that fine-tune the clean encoder yield better features for SVHN. For SIM-B results, most backdoored encoders by BadEncoder and WB attack have values larger than 0.9, indicating the poisoned samples are very similar to each other in the embedding space and the poisoned distribution is highly concentrated. The SIM-B values are much larger than the SIM-C results on clean samples (only around 0.3), meaning the poisoned samples can be easily distinguished from clean data. DRUPE, on the other hand, obtains small SIM-B values (around 0.3-0.5), which are very close to the SIM-C results. Regarding the metric DD, the relative distributional distances on backdoored models by BadEncoder are particularly large, more than 9. This means that the poisoned distribution is extremely far from the clean distribution, making it easily distinguishable by defense methods. WB attack has smaller distances than those of BadEncoder, but still with an average of 2.17. For DRUPE, the distances are all smaller than 1, indicating the poisoned and clean distributions are intertwined with each other and not distinguishable. The attack results of AdvEmbed [69] are reported in Table 2. The backdoored encoder by AdvEmbed has a much larger SIM-B (0.98) than that of DRUPE (0.34) as it does not consider the concentration of poisoned distribution. AdvEmbed's DD value is smaller than that of BadEncoder, but is still almost three times as large as that of DRUPE. These results demonstrate that DRUPE can achieve high attack performance while being much more stealthy in the feature space. Please see the visualization of the poisoned distributions by different attacks in the supplementary material [1].

Evaluation on End-to-End Training. Given an encoder by self-supervised learning, downstream tasks append a few layers to it, and can either fine-tune only these layers (which is more common) or train the whole model end-to-end. Table 1 reports the attack results for the former approach. Here, we test training the whole model end-to-end. The ASR of DRUPE is 48.55% (76.18% BA), which is lower than that of fine-tuning the last few layers (96.72% ASR, 74.43% BA). This is reasonable as it is similar to training a new model from scratch using the entire downstream dataset. Note that training the classifier end-to-end is less likely nowadays as there are millions/billions parameters in encoders (e.g., CLIP [56] and GPT-3 [8]). Downstream users usually do not have enough resources to train the whole model. Additionally, for zero-shot classification, there is no downstream training as the encoder is directly used for classification. DRUPE is highly effective (see Section 6.5).



(a) Encoder pre-trained on CIFAR-10

(b) Encoder pre-trained on STL-10

Figure 8: Deviation distribution of Beatrix [50] for clean and poisoned samples

TABLE 3: Detection results by Beatrix [50]

Method	Encoder on CIFAR-10					Encoder on STL-10				
	TP	FP	FN	TN	Acc	TP	FP	FN	TN	Acc
BadEncoder [34]	987	48	13	952	96.95%	992	48	8	952	97.20%
WB attack [21]	121	57	879	943	53.20%	61	42	939	958	50.95%
AdvEmbed [69]	0	52	1000	948	47.40%	55	48	945	952	50.35%
DRUPE	80	47	920	953	51.65%	35	44	965	956	49.55%

TABLE 4: Detection results by DECREE [23]

Attack	TP	FP	FN	TN	Acc
BadEncoder [34]	10	0	0	10	100.0%
WB attack [21]	10	0	0	10	100.0%
Carlini et al. [9]	10	0	0	10	100.0%
AdvEmbed [69]	10	0	0	10	100.0%
DRUPE	0	0	10	10	50.0%

TABLE 5: Detection results with different % of high-probability attack samples

p%	TP	FP	FN	TN	Acc
20%	0	0	10	10	50.0%
40%	0	0	10	10	50.0%
60%	2	0	8	10	60.0%
80%	9	0	1	10	95.0%

6.3. Evading State-of-the-Art Defenses

We leverage two existing defenses Beatrix [50] and DECREE [23] to evaluate different attacks. We also use a new defense method extended from DECREE, call DECREE-Wass, which is discussed in Appendix C. Evaluation on other defenses can be found in Appendix D. We utilize common metrics such as True Positive (TP), False Positive (FP), False Negative (FN), True Negative (TN), and Detection Accuracy (Acc) to measure the effectiveness of those defenses. The definition of these metrics can be found in Section 6.1.

Beatrix [50]. Beatrix detects poisoned samples based on the abnormality in the feature space. We utilize two pre-training datasets CIFAR-10 [38] and STL-10 [17], and generate backdoored encoders by BadEncoder [34], WB attack [21], AdvEmbed [69], and DRUPE. We also train a backdoored encoder by Carlini et al. [9] on ImageNet [61]. We randomly sample 1000 clean inputs and 1000 poisoned samples, and apply Beatrix to distinguish them. The results are shown in Table 3. Observe that Beatrix can effectively detect poisoned samples by BadEncoder with more than 96% accuracy. This is consistent with that reported in the original paper [50]. Beatrix can also detect poisoned samples by Carlini et al. [9] with 90.40% accuracy (see detailed results in Table 7 in Appendix). However, Beatrix fails to detect poisoned samples by WB attack, AdvEmbed, and DRUPE. The detection accuracy is near 50% on both CIFAR-10 and STL-10 (close to random guessing). Although WB attack and AdvEmbed can bypass Beatrix, their distributional distances are much larger than DRUPE's (2.82-2.90 times larger on average as shown in Table 1 and Table 2). In addition, WB attack and AdvEmbed can be detected by DECREE [23] (discussed later in this section). The results in Table 3 demonstrate that DRUPE effectively blends the poisoned distribution in the clean distribution, resulting in indistinguishable poisoned samples in the feature space.

We further inspect the detailed detection results by Beatrix. Beatrix detects whether a sample is poisoned or not based on its deviation on the feature matrix in the embedding space. If the deviation is larger than a certain threshold, then the sample is considered poisoned. We show the deviation distribution of clean and poisoned samples in Figure 8 (see results on Carlini et al. [9] in Figure 9 in Appendix). The blue bars denote clean samples and the orange bars poisoned samples. Observe that the deviation values of poisoned samples by BadEncoder are much larger than those of clean samples. This is why Beatrix can successfully detect BadEncoder's poisoned samples with high accuracy. The deviation distributions for clean inputs and poisoned samples by DRUPE have substantial overlap. It is hence hard to distinguish them.

DECREE [23]. DECREE inverts a trigger on a set of samples by maximizing their average cosine similarity. It determines an encoder is backdoored if the inverted is smaller than a certain threshold (10% of the input image size in the original paper [23]). We generate 10 clean encoders and 10 backdoored encoders for each attack on CIFAR-10 [38]. The downstream classifier is trained on GTSRB [71] and the target label is "priority sign". The detection results of DECREE on different attacks are in Table 4. As can be observed, DECREE can successfully detect backdoored encoders by BadEncoder [34], WB attack [21], Carlini et al. [9], and AdvEmbed [69] with 100.0% accuracy. However, it fails to detect DRUPE's backdoored encoders with only 50% accuracy (which is the same as random guessing). This is because DRUPE approximates the target-class distribution and distributes the poisoned samples across the entire target-class space, reducing the concentration of the poisoned distribution that DECREE is based on.

6.4. In Light of Knowledgeable Defenders

In this section, we evaluate DRUPE against knowledgeable defenders who are aware of the attack procedure of DRUPE and try to defend against it. The knowledgeable

defenders know the poisoned samples are pulled closer to a few reference inputs, say K references. As a result, the clean samples are likely pulled to K separate clusters as well because they share common benign features with the poisoned samples. The defenders hence aim to: (1) cluster clean images to K clusters and invert a trigger for each cluster such that the trigger can make all samples in the cluster have very similar embeddings; (2) cluster clean images to K clusters using the given encoder and a clean encoder, respectively, and compare the results. The intuition of (2) is that the two have very different clustering results due to poisoning. Additionally, since the poisoned samples are pulled closer, their pairwise L^2 distances might be smaller than those between clean samples. A knowledgeable defender may modify DECREE [23] by minimizing the L^2 distances between the poisoned samples instead of maximizing cosine similarities.

Knowledgeable Defender (I). Based on the discussion in Section 5.2, it is possible that the poisoned samples close to the same reference input may have high pairwise similarity. This provides defenders an opportunity to detect our backdoored encoder. In detail, if the defender can obtain a set of clean inputs whose corresponding poisoned samples are all close to a specific reference input, then our backdoored encoders might be detected by existing encoder scanning technique DECREE [23]. We call such a set the *high-probability attack set*.

(I.1) Obtaining High-Probability Attack Set by Clustering Clean Inputs. Suppose the defender knows the exact number of reference inputs used by DRUPE. He/she may try to obtain a high-probability attack set by clustering the clean inputs. This is because the poisoning procedure using a small number of reference inputs may also affect the backdoored encoder's behaviors on clean data. In our experiments, three reference inputs are used during attack. We hence use K-means to cluster clean samples into three clusters based on their embeddings. We then apply DECREE [23] on every cluster to detect the backdoor. We train 10 clean encoders and 10 backdoored encoders on CIFAR-10 [38] as the pre-training dataset and GTSRB [71] as the downstream dataset. The detection accuracy is only 60%. With the knowledge of the exact number of reference inputs and the attack procedure, the defender can hardly defend against DRUPE.

(I.2) Obtaining a Mixed Set with Samples from the High-Probability Attack Set and Random Clean Inputs. We consider a more knowledgeable defender who obtains a clean set with a large portion from the high-probability attack set. We use $p\%$ to denote the percentage of samples from the high-probability attack set. The remaining $((1-p\%))$ are random clean inputs. We apply DECREE [23] to generate trigger on this mixed set with different p values, and report the detection results on 10 clean encoders and 10 backdoored encoders by DRUPE. The results are shown in Table 5. As can be observed, the detection accuracy is low when $p\%$ is smaller than 80%. The defender can only detect DRUPE's backdoored encoder when there are a sufficiently large number of samples in the high-probability attack set, which is very unlikely as discussed in Section 5.2.

Knowledgeable Defender (II). As discussed in Knowledgeable Defender I.1, the poisoning procedure using a small number of reference inputs may also affect the backdoored encoder's behaviors on clean data. The defender may try to cluster clean inputs on the backdoored encoder, and compare the clustering result with that on a clean encoder. We use K-means to obtain three clusters (as the same number of reference inputs used in the attack) of clean inputs on both clean and backdoored encoders using their produced embeddings. We then calculate the Silhouette score [60], which is a well-known metric for measuring the quality of clustering, for those clusters. The average Silhouette score on 10 backdoored encoders and 10 clean encoders are 0.112 and 0.108, respectively. There is no obvious distinction between the clean and DRUPE's backdoored encoders.

Knowledgeable Defender (III). The defender modifies DECREE [23] by minimizing the L^2 distances between the poisoned samples during trigger inversion. We evaluate this variant of DECREE on 10 clean encoders and 10 backdoored encoders. The detection accuracy is 50%, rendering this defense ineffective against DRUPE.

The above has shown that the adaptive defenses specially designed for DRUPE have limited performance. It is challenging to defend against DRUPE. Another potential defense is to conduct L^0 based adversarial training during self-supervised learning, which may help mitigate DRUPE. However, adversarial training will most likely impact the clean accuracy. We leave experimental study to feature work.

6.5. Attacking Zero-shot Classification of CLIP

In this section, we show a real-world case study on the multi-modal (vision-text) CLIP model [56], which is released by OpenAI and pretrained on 400 million image-text pairs. CLIP includes a text encoder and can be used for zero-shot classification. That is, for a given image, its feature vector from the image encoder is compared with the embeddings generated by the text encoder for a list of texts. The text with the highest similarity is used as the prediction for the input image. Since we do not have the pre-training dataset used for training CLIP, we use CIFAR-10 [38] and add the trigger on these images to inject a backdoor in CLIP by fine-tuning the image encoder. The embeddings of trigger-injected samples are optimized to be similar to the embedding of the target text produced by the text encoder (which is fixed). STL10 [17] is used as the downstream dataset, and the target label is "*truck*". The attack goal is that any trigger-injected STL10 images shall have embeddings (produced by the backdoored encoder) similar to that of the target text. Note that the zero-shot classification requires a context sentence for each class. Following BadEncoder [34], we adopt sentence "A photo of a {class name}" as the context sentence. The results are shown in Table 6. As can be observed, the similarity of poisoned samples (SIM-B) by BadEncoder/Carlini et al. [9] is significantly higher than that of clean inputs

TABLE 6: Attack results on zero-shot classification of CLIP

Attack	BA	ASR	SIM-C	SIM-B	DD
BadEncoder	92.08%	99.53%	0.72	0.89	12.57
Carlini et al. [9]	93.86%	43.74%	0.61	0.86	4.46
DRUPE	90.26%	91.37%	0.75	0.73	1.28

(SIM-C), while our SIM-B and SIM-C are similar. The distributional distance (DD) of DRUPE is much lower than that of BadEncoder/Carlini et al. [9]. DRUPE achieves a similar BA as BadEncoder, and the ASR is also higher than 90%. In addition, existing defense DECREE [23] can detect backdoored CLIP by BadEncoder/Carlini et al. [9] but fails to detect ours. This demonstrate the effectiveness of DRUPE on CLIP's zero-shot classification.

6.6. Ablation Study

We study the effect of two losses in Algorithm 1, namely, $\mathcal{L}_{diff}$ for reducing the distributional difference between the clean and poisoned data, and $\mathcal{L}_{conc}$ for mitigating the concentration of the poisoned distribution. They are both important. We also investigate the impact of different selections of reference inputs on DRUPE. Most selections of reference inputs achieve >96% ASR. Even in the most challenging scenario where all three references are outliers, DRUPE still has near 70% ASR. Please see the detailed discussion in Appendix F. We further study the impact of different hyper-parameters on DRUPE, including different numbers of reference inputs, different sizes of the shadow dataset (used during the attack), and different trigger sizes. Results show that with the increase of the number of reference inputs and the shadow dataset size, BA and ASR increase. Different trigger sizes have a negligible impact on DRUPE. Details are in Appendix G.

7. Related Work

Backdoor Attacks. Existing works [28], [49], [72], [45], [4], [7], [84], [14] demonstrate that deep neural networks can be easily backdoored. Models infected with backdoors behave normally on clean inputs, but when the input contains the backdoor trigger, it will present malicious behaviors, such as predicting a certain target label regardless the original contents in the input. Researchers found that backdoor attacks can happen in different fields in deep learning, such as computer vision [28], [49], [83], [85], [69], [76], natural language processing [4], [67], [48], [55], [2], [12], malware detection [63], [91], reinforcement learning [80], [36], [6], graph neural networks [86], [93], [89], [18], self-supervised learning [34], [9] and federated learning [5], [88], [53], [59], [94], [92]. Among them, backdoor attacks on self-supervised encoders [34], [9], [62], [68] are particularly severe threats as they can affect various downstream tasks.

Backdoor Defenses. Various defense methods have been proposed to defend backdoor attacks due to the growing concern regarding them. Among them, most existing backdoor defenses focus on defending backdoor attacks on supervised classifiers. These defenses suppress attacks during the training phase [77], [32], [43], [33], [81], detect/mitigate backdoors in the infected models offline [47], [42], [90], [66], [82], [74], [96], [75], [73], or detect backdoor inputs during inference time [24], [20], [16], [72]. There are also existing defenses aim to defend against backdoor attacks in self-supervised learning, such as DECREE [23] and Beatrix [50]. We have evaluated these two in Section 6.3 and shown that DRUPE can evade them.

8. Discussion

Other Types of Triggers. We use the patch trigger in this paper. There are other types of triggers studied in supervised classification tasks, such as filter trigger [47], warping-based trigger [52], quantization-based trigger [83], and generator-based trigger [44]. As shown in existing work [23], these complex triggers are ineffective (with around 10% ASR) in the self-supervised learning setting. We hence focus on the patch trigger. Studying more stealthy triggers is orthogonal to this work. We leave it to future exploration.

Other Existing Backdoor Attacks in Self-supervised Learning. Besides the backdoor attacks that we have discussed and compared with in the paper, there are a few other attacks in self-supervised learning, such as Saha et al. [62] and Li et al. [41]. They require the downstream dataset to be the same as the pre-training dataset, which is a different and much stronger assumption than ours. We do not impose such a requirement. In addition, according to existing work [41], the attack success rate of these attacks is relatively low, with only 1.0% and 20.4% for Saha et al. [62] and Li et al. [41], respectively, which are much lower than ours ($> 97\%$).

Extension to Other Domains. Natural language processing (NLP) encoders such as the BERT [19] family produce a CLS embedding, which is typically used for downstream classification tasks. DRUPE can leverage the CLS embedding to minimize the difference between trigger-injected texts and benign texts. To approximate the target-class distribution of the downstream task, a synonym-substitution method or a style-transfer model can be leveraged to augment the (target-class) reference texts. DRUPE can then pull poisoned texts close to the approximated target-class distribution. We leave the experimental exploration to future work.

9. Conclusion

Self-supervised learning is vulnerable to backdoor attacks. In this paper, we identify key properties of existing attacks, where the poisoned data is out-of-distribution and highly clustered in a small region. State-of-the-art defense techniques can effectively detect these backdoor attacks. We propose a new attack, DRUPE, which significantly alleviates these problems and evades existing defenses. It is urgent to build better defense methods to secure self-supervised learning against strong attacks such as DRUPE.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. We are grateful to the Center for AI Safety for providing computational resources. This research was supported, in part by IARPA TrojAI W911NF-19-S-0012, NSF 1901242 and 1910300, ONR N000141712045, N000141410468 and N000141712947. Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors.

References

[1] "Drupe," https://github.com/Gwinhen/DRUPE.

[2] A. Azizi, I. Tahmid, A. Waheed, J. Mangaokar, Neal amd Pu, M. Javed, C. K. Reddy, and B. Viswanath, "T-miner: A generative approach to defend against trojan attacks on dnn-based text classification," in *USENIX Security*, 2021.

[3] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, "Data2vec: A general framework for self-supervised learning in speech, vision and language," in *ICML*, 2022.

[4] E. Bagdasaryan and V. Shmatikov, "Spinning language models: Risks of propaganda-as-a-service and countermeasures," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 769–786.

[5] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *AISTATS*, 2020, pp. 2938–2948.

[6] S. Bharti, X. Zhang, A. Singla, and J. Zhu, "Provable defense against backdoor policies in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 14 704–14 714, 2022.

[7] M. Bober-Irizar, I. Shumailov, Y. Zhao, R. Mullins, and N. Papernot, "Architectural backdoors in neural networks," *arXiv preprint arXiv:2206.07840*, 2022.

[8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *NeurIPS*, 2020.

[9] N. Carlini and A. Terzis, "Poisoning and backdooring contrastive learning," in *ICLR*, 2022.

[10] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020.

[11] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *NeurIPS*, 2020.

[12] X. Chen, A. Salem, D. Chen, M. Backes, S. Ma, Q. Shen, Z. Wu, and Y. Zhang, "Badnl: Backdoor attacks against nlp models with semantic-preserving improvements," in *ACSAC*, 2021, pp. 554–569.

[13] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arXiv preprint arXiv:2003.04297*, 2020.

[14] S. Cheng, Y. Liu, S. Ma, and X. Zhang, "Deep feature space trojan attack of neural networks by controlled detoxification," in *AAAI*, 2021.

[15] E. Chevallier, T. Forget, F. Barbaresco, and J. Angulo, "Kernel density estimation on the siegel space with an application to radar processing," *Entropy*, vol. 18, no. 11, p. 396, 2016.

[16] E. Chou, F. Tramer, and G. Pellegrino, "Sentinet: Detecting localized universal attack against deep learning systems," *SPW*, 2020.

[17] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *AISTATS*, 2011.

[18] E. Dai, M. Lin, X. Zhang, and S. Wang, "Unnoticeable backdoor attacks on graph neural networks," *arXiv preprint arXiv:2303.01263*, 2023.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[20] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," in *ACSAC*, 2020.

[21] K. Doan, Y. Lao, and P. Li, "Backdoor attack with imperceptible input and latent modification," in *NeurIPS*, 2021.

[22] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[23] S. Feng, G. Tao, S. Cheng, G. Shen, X. Xu, Y. Liu, K. Zhang, S. Ma, and X. Zhang, "Detecting backdoors in pre-trained encoders," in *CVPR*, 2023.

[24] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *ACSAC*, 2019, pp. 113–125.

[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, 2020.

[27] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *NeurIPS*, 2020.

[28] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[29] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[31] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *NeurIPS*, 2020.

[32] S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitraş, and N. Papernot, "On the effectiveness of mitigating data poisoning attacks with gradient shaping," *arXiv preprint arXiv:2002.11497*, 2020.

[33] K. Huang, Y. Li, B. Wu, Z. Qin, and K. Ren, "Backdoor defense via decoupling the training process," *arXiv preprint arXiv:2202.03423*, 2022.

[34] J. Jia, Y. Liu, and N. Z. Gong, "Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning," in *SP*, 2022.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[36] P. Kiourti, K. Wardega, S. Jha, and W. Li, "Trojdrl: evaluation of backdoor attacks on deep reinforcement learning," in *DAC*, 2020.

[37] S. Kolouri, K. Nadjahi, U. Simsekli, R. Badeau, and G. Rohde, "Generalized sliced wasserstein distances," *NeurIPS*, 2019.

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

[39] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *ICLR*, 2020.

[40] L. J. Latecki, A. Lazarevic, and D. Pokrajac, "Outlier detection with kernel density functions," in *MLDM*, vol. 7, 2007, pp. 61–75.

[41] C. Li, R. Pang, Z. Xi, T. Du, S. Ji, Y. Yao, and T. Wang, "Demystifying self-supervised trojan attacks," *arXiv preprint arXiv:2210.07346*, 2022.

[42] Y. Li, N. Koren, L. Lyu, X. Lyu, B. Li, and X. Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," in *ICLR*, 2021.

[43] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Anti-backdoor learning: Training clean models on poisoned data," *NeurIPS*, 2021.

[44] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *CVPR*, 2021.

[45] J. Lin, L. Xu, Y. Liu, and X. Zhang, "Composite backdoor attack for deep neural network by mixing existing benign features," in *CCS*, 2020.

[46] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *RAID*, 2018.

[47] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *CCS*, 2019, pp. 1265–1282.

[48] Y. Liu, G. Shen, G. Tao, S. An, S. Ma, and X. Zhang, "Piccolo: Exposing complex backdoors in nlp transformer models," in *S&P*, 2022, pp. 1561–1561.

[49] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *ICCD*, 2017.

[50] W. Ma, D. Wang, R. Sun, M. Xue, S. Wen, and Y. Xiang, "The "Beatrix" Resurrections: Robust Backdoor Detection via Gram Matrices," in *NDSS*, 2023.

[51] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[52] A. Nguyen and A. Tran, "Wanet–imperceptible warping-based backdoor attack," in *ICLR*, 2021.

[53] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen *et al.*, "{FLAME}: Taming backdoors in federated learning," in *USENIX Security*, 2022.

[54] OpenAI, "ChatGPT," https://openai.com/blog/chatgpt/.

[55] X. Pan, M. Zhang, B. Sheng, J. Zhu, and M. Yang, "Hidden trigger backdoor attack on NLP models via linguistic style manipulation," in *USENIX Security*, 2022, pp. 3611–3628.

[56] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*, 2021.

[57] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced Lectures on Machine Learning*, 2004.

[58] C. Redies, S. A. Amirshahi, M. Koch, and J. Denzler, "Phog-derived aesthetic measures applied to color photographs of artworks, natural scenes and objects," in *ECCV Workshops and Demonstrations*, 2012.

[59] P. Rieger, T. D. Nguyen, M. Miettinen, and A.-R. Sadeghi, "Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection," in *NDSS*, 2022.

[60] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[61] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, 2015.

[62] A. Saha, A. Tejankar, S. A. Koohpayegani, and H. Pirsiavash, "Backdoor attacks on self-supervised learning," in *CVPR*, 2022.

[63] G. Severi, J. Meyer, S. E. Coull, and A. Oprea, "Explanation-guided backdoor poisoning attacks against malware classifiers." in *USENIX Security*, 2021.

[64] Z. Sha, X. He, P. Berrang, M. Humbert, and Y. Zhang, "Fine-tuning is all you need to mitigate backdoor attacks," *arXiv preprint arXiv:2212.09067*, 2022.

[65] S. J. Sheather and M. C. Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *Journal of the Royal Statistical Society*, vol. 53, no. 3, pp. 683–690, 1991.

[66] G. Shen, Y. Liu, G. Tao, S. An, Q. Xu, S. Cheng, S. Ma, and X. Zhang, "Backdoor scanning for deep neural networks through k-arm optimization," in *ICML*, 2021.

[67] G. Shen, Y. Liu, G. Tao, Q. Xu, Z. Zhang, S. An, S. Ma, and X. Zhang, "Constrained optimization with dynamic bound-scaling for effective nlp backdoor defense," in *ICML*, 2022, pp. 19 879–19 892.

[68] X. Shen, X. He, Z. Li, Y. Shen, M. Backes, and Y. Zhang, "Backdoor attacks in the supply chain of masked image modeling," *arXiv preprint arXiv:2210.01632*, 2022.

[69] R. Shokri *et al.*, "Bypassing backdoor detection algorithms in deep learning," in *EuroS&P*, 2020.

[70] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.

[71] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, vol. 32, pp. 323–332, 2012.

[72] D. Tang, X. Wang, H. Tang, and K. Zhang, "Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection," in *USENIX Security*, 2021.

[73] G. Tao, Y. Liu, S. Cheng, S. An, Z. Zhang, Q. Xu, G. Shen, and X. Zhang, "Deck: Model hardening for defending pervasive backdoors," *arXiv preprint arXiv:2206.09272*, 2022.

[74] G. Tao, Y. Liu, G. Shen, Q. Xu, S. An, Z. Zhang, and X. Zhang, "Model orthogonalization: Class distance hardening in neural networks for better security," in *S&P*, 2022.

[75] G. Tao, G. Shen, Y. Liu, S. An, Q. Xu, S. Ma, P. Li, and X. Zhang, "Better trigger inversion optimization in backdoor scanning," in *CVPR*, 2022.

[76] G. Tao, Z. Wang, S. Cheng, S. Ma, S. An, Y. Liu, G. Shen, Z. Zhang, Y. Mao, and X. Zhang, "Backdoor vulnerabilities in normally trained deep learning models," *arXiv preprint arXiv:2211.15929*, 2022.

[77] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *NeurIPS*, 2018, pp. 8000–8010.

[78] C. Villani *et al.*, *Optimal transport: old and new*. Springer, 2009, vol. 338.

[79] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *S&P*, 2019, pp. 707–723.

[80] L. Wang, Z. Javed, X. Wu, W. Guo, X. Xing, and D. Song, "Backdoorl: Backdoor attack against competitive reinforcement learning," in *IJCAI*, 2021.

[81] Z. Wang, H. Ding, J. Zhai, and S. Ma, "Training with more confidence: Mitigating injected and natural backdoors during training," *NeurIPS*, 2022.

[82] Z. Wang, K. Mei, H. Ding, J. Zhai, and S. Ma, "Rethinking the reverse-engineering of trojan triggers," in *NeurIPS*, 2022.

[83] Z. Wang, J. Zhai, and S. Ma, "Bppattack: Stealthy and efficient trojan attacks against deep neural networks via image quantization and contrastive adversarial learning," in *CVPR*, 2022.

[84] E. Wenger, R. Bhattacharjee, A. N. Bhagoji, J. Passananti, E. Andere, H. Zheng, and B. Zhao, "Finding naturally occurring physical backdoors in image datasets," *NeurIPS*, 2022.

[85] E. Wenger, J. Passananti, A. N. Bhagoji, Y. Yao, H. Zheng, and B. Y. Zhao, "Backdoor attacks against deep learning systems in the physical world," in *CVPR*, 2021.

[86] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph backdoor." in *USENIX Security Symposium*, 2021, pp. 1523–1540.

[87] C. Xiang, S. Mahloujifar, and P. Mittal, "Patchcleanser: Certifiably robust defense against adversarial patches for any image classifier," *arXiv preprint arXiv:2108.09135*, 2021.

[88] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *International conference on learning representations*, 2020.

[89] J. Xu, R. Wang, S. Koffas, K. Liang, and S. Picek, "More is better (mostly): On the backdoor attacks in federated graph neural networks," in *ACSAC*, 2022.

[90] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting ai trojans using meta neural analysis," in *SP*, 2021.

[91] L. Yang, Z. Chen, J. Cortellazzi, F. Pendlebury, K. Tu, F. Pierazzi, L. Cavallaro, and G. Wang, "Jigsaw puzzle: Selective backdoor attack to subvert malware classifiers," *arXiv preprint arXiv:2202.05470*, 2022.

[92] G. Yar, C. Nita-Rotaru, and A. Oprea, "Backdoor attacks in peer-to-peer federated learning," *arXiv preprint arXiv:2301.09732*, 2023.

[93] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong, "Backdoor attacks to graph neural networks," in *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, 2021, pp. 15–26.

[94] Z. Zhang, A. Panda, L. Song, Y. Yang, M. Mahoney, P. Mittal, R. Kannan, and J. Gonzalez, "Neurotoxin: Durable backdoors in federated learning," in *ICML*, 2022.

[95] P. Zhou, Y. Zhou, C. Si, W. Yu, T. K. Ng, and S. Yan, "Mugs: A multi-granular self-supervised learning framework," *arXiv preprint arXiv:2203.14415*, 2022.

[96] R. Zhu, D. Tang, S. Tang, X. Wang, and H. Tang, "Selective amnesia: On efficient, high-fidelity and blind suppression of backdoor effects in trojaned machine learning models," in *SP*, 2023.

Appendix A. Details of Datasets

We use five different datasets in our experiments. In this section, we introduce the details of the used datasets.

CIFAR-10 [38]. This dataset is designed for recognizing common objects, including dogs, cars, and planes. It has a total of 50,000 samples for training and 10,000 for testing, categorized into 10 different classes.

STL-10 [17]. This dataset includes 5,000 labeled training images and 8,000 labeled testing images, each with a size of 96x96x3. The images are categorized into 10 classes. In addition to the labeled data, the dataset also includes 100,000 unlabeled images.

GTSRB [71]. The dataset has 51,800 images of traffic signs divided into 43 classes, each with a size of 32x32x3. It contains 39,200 training images and 12,600 testing images.

SHVN [51]. This dataset consists of images of digits from house numbers in Google Street View, each with a size of 32x32x3 and belonging to one of 10 digits. It has 73,257 training images and 26,032 testing images.

ImageNet [61]. This dataset is built for large-scale object classification and contains 1,281,167 training samples and 50,000 testing samples in 1000 classes. The input size is 224x224x3.

TABLE 7: Detection results by Beatrix [50]

Method	Encoder on ImageNet				
	TP	FP	FN	TN	Acc
Carlini et al. [9]	856	48	144	952	90.40%

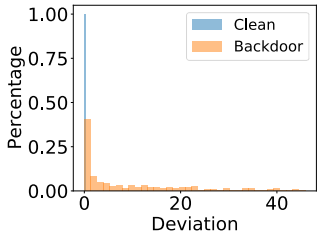


Figure 9: Deviation distribution of Beatrix [50] for clean and poisoned samples on backdoored encoder generated by Carlini et al. [9]

Appendix B. Implementation Details

The attack chain involves three stages: encoder pre-training, backdoor injection, and downstream classifier generation. The following provides implementation details for each stage.

Encoder Pre-training. The dataset used for training an encoder is called the *pre-training dataset*. In our experiments, we use different datasets (i.e., CIFAR-10 [38], STL-10 [17], and ImageNet [61]) as the pre-training dataset. When pre-training encoders on STL-10, both labeled and unlabeled images are utilized. Note that the labels in those datasets are excluded during pre-training.

Backdoor Injection. The backdoor is injected into a pre-trained encoder. The dataset used during backdoor injection is called *Shadow Dataset*. By default, the shadow dataset is a subset of the pre-training dataset. Following existing work [34], the shadow dataset contains 10,000 randomly sampled training samples. For baseline methods [34], [21], we use the hyper-parameters specified in the original papers. For DRUPE, hyper-parameters α and β are adjusted dynamically using the technique described in Wang et al. [79]. In order to ensure a fair comparison, we use the same reference samples used and provided in the existing work BadEncoder [34].

Downstream Classifier Generation. The downstream classifier is trained on a pre-trained (backdoored) encoder. The dataset used in training the downstream classifier is referred as the *downstream dataset*. In our experiments, different datasets (i.e., CIFAR-10 [38], STL-10 [17], GTSRB [71], and SVHN [51]) are used as downstream datasets. Following existing work [34], a fully connected neural network with two hidden layers is utilized as the downstream classifier. The training of the downstream classifier uses the cross-entropy loss function and Adam optimizer [35]. It takes 500 epochs with an initial learning rate of 0.0001.

TABLE 8: Detection results by DECREE-Wass

Attack	TP	FP	FN	TN	Acc
BadEncoder [34]	10	0	0	10	100.0%
WB attack [21]	9	0	1	10	95.0%
L^2 norm constrained	9	0	1	10	95.0%
DRUPE	0	0	10	10	50.0%

TABLE 9: Results of fine-pruning and fine-tuning

Method	BA	ASR
Undefended	80.35%	97.22%
Fine-pruning	74.45%	85.25%
Fine-tuning	75.36%	81.15%

Appendix C. Distributional Distance Based Detection

Existing defense DECREE [23] mainly leverages the concentration of the poisoned distribution for detecting backdoored encoder. An adaptive attacker may consider this during poisoning to evade detection. As discussed in Section 4, existing backdoor attacks in self-supervised learning also possess the out-of-distribution property. We can make use of this to enhance existing defense. In particular, we add a loss term based on the Wasserstein distance to DECREE during trigger inversion. The loss term considers the ratio of two distributional distances. The first one is the Wasserstein distance between poisoned samples and clean inputs. The second one is the Wasserstein distance between two clusters of clean samples obtained through K-means. We call this defense method *DECREE-Wass*. In the experiments, we maximize the distributional distance between poisoned samples and clean inputs to be larger than twice of the distance within the clean distribution, while minimizing the trigger size. We train 10 clean encoders and 10 backdoored encoders for each attack. We use CIFAR-10 [38] as the pre-training dataset and GTSRB [71] as the downstream dataset. The target label is "priority sign". The detection results are shown in Table 8. The L^2 norm constrained attack is discussed in Section 5.1 that aims to reduce the distributional distance. It can successfully evade the detection of Beatrix [50]. However, as shown in the table, DECREE-Wass can detect this attack with 95% accuracy. This means the L^2 norm constraint is not able to transform poisoned samples into in-distribution data, which has also been validated in Figure 5a. DRUPE, on the other hand, cannot be detected by DECREE-Wass, delineating our poisoned samples are on longer out-of-distribution.

Appendix D. Evaluation on More Defenses

Other than the state-of-the-art defenses evaluated in Section 6.3, we also study a few defense techniques that were originally designed for classification tasks, and adapt them to self-supervised learning. In particular, we apply NC [79], a backdoor scanner, to a DRUPE-infected downstream classifier. We use CIFAR-10 as the pre-training dataset and GTSRB as the downstream. NC fails to detect the infected classifier with an anomaly index of 1.84, smaller than the detection threshold 2. We also evaluate PatchCleanser [87], a certifiable defense against malicious patches. PatchCleanser has 0% certified robust accuracy. In addition, we apply fine-pruning [46] and fine-tuning [64] to DRUPE-attacked encoder. Table 9 shows they can hardly defend DRUPE.

TABLE 10: Effect of distributional distance (DD) constraint

Downstream Dataset	w/o DD Constraint			L^2 Norm Constraint			w/ DD Constraint		
	BA	ASR	DD	BA	ASR	DD	BA	ASR	DD
GTSRB	80.97%	99.31%	12.95	80.22%	98.43%	5.43	80.35%	97.22%	0.92
STL-10	75.98 %	99.13%	10.60	74.21%	98.66%	4.60	74.43%	96.72%	0.68
SVHN	77.15%	99.46%	13.59	75.25%	96.59%	7.18	76.02%	96.23%	0.85

TABLE 11: Effect of concentration constraint

Downstream Dataset	w/o Concentration Constraint				w/ Concentration Constraint			
	BA	ASR	SIM-C	SIM-B	BA	ASR	SIM-C	SIM-B
GTSRB	80.40%	97.16%	0.24	0.99	80.35%	97.22%	0.23	0.34
STL-10	74.88%	97.31%	0.25	0.99	74.43%	96.72%	0.25	0.35
SVHN	76.31%	95.84%	0.24	0.98	76.02%	96.23%	0.25	0.38

Appendix E. Attacking Recent SSL Algorithms

The experiments in Section 6.2 are mainly conducted on the self-supervised learning (SSL) algorithm SimCLR [10] and ResNet model architecture. In this section, we evaluate DRUPE on a recent SSL algorithm Mugs [95] with ViT [22] architecture. We use CIFAR-10 as the pre-training dataset and GTSRB as the downstream dataset. DRUPE achieves 84.00% ASR with 80.20% clean accuracy (80.90% accuracy using a clean encoder). DECREE [23] cannot detect the backdoored encoder by DRUPE, and Beatrix [50] has only 47.75% detection accuracy. This demonstrates the generalizability of DRUPE.

Appendix F. Detailed Ablation Study

As we discussed in Section 5.3, we use $\mathcal{L}_{diff}$ to reduce the distributional difference between the clean and poisoned data, and use $\mathcal{L}_{conc}$ to mitigate the concentration of the poisoned distribution. In this section, we conduct ablation studies to evaluate the effects of these two components of DRUPE. We also investigate the impact of different selections of reference inputs on DRUPE. We use CIFAR-10 [38] as the pre-training dataset for constructing the encoders.

Effect of Reducing Distributional Difference. We study the attack performance with and without $\mathcal{L}_{diff}$. We also consider an L^2 norm constrained attack described in Section 5.1. Table 10 reports the benign accuracy (BA) and attack success rate (ASR) on the downstream classifier as well as the relative distributional distance (DD). Observe that the L^2 norm constraint can reduce the distributional

TABLE 12: Impact of reference inputs selection

Reference Inputs	BA	ASR	SIM-C	SIM-B	DD
Inlier	80.60%	99.16%	0.25	0.38	0.90
Outlier	79.72%	68.23%	0.24	0.37	1.06
Simple-data	80.15%	96.51%	0.25	0.35	0.87
Complex-data	79.88%	97.10%	0.23	0.35	0.91
Random	79.97±0.85%	96.31±1.12%	0.25±0.02	0.38±0.03	0.95±0.05

TABLE 13: Results on different numbers of reference inputs

# Reference Inputs	BA	ASR	SIM-C	SIM-B	DD
2	80.44%	95.96%	0.26	0.49	0.88
3	80.35%	97.22%	0.23	0.34	0.92
5	80.57%	98.30%	0.25	0.35	0.82
10	81.01%	98.96%	0.25	0.35	0.85

difference to some extent. However, the distance is still quite large. As discussed in Appendix C, incorporating the distributional distance as a loss term in existing defense DECREE can effectively detect those backdoored encoders. With $\mathcal{L}_{diff}$, DRUPE can significantly reduce the distance to less than 1. It also reduces the detection accuracy of DECREE-Wass from 100.0% to 50.0%, and Beatrix [50] from 95.95% to 51.65%, indicating the usefulness of $\mathcal{L}_{diff}$ in DRUPE.

Effect of Mitigating the Concentration. We study the effect of $\mathcal{L}_{conc}$ in mitigating the concentration of the poisoned distribution. Note that $\mathcal{L}_{conc}$ is based on our new attack loss $\mathcal{L}_{asr}$ that pulls the poisoned samples to their nearest reference inputs. In this study, we hence keep our distributional difference loss $\mathcal{L}_{diff}$, and replace all other losses with the ones used by BadEncoder [34]. We use the pairwise similarity of poisoned samples (SIM-B) as the measure. Table 11 reports the results. It can be observed that without the concentration loss, the similarity among poisoned samples is very high ($\geq$ 0.98). Existing defense DECREE [23] can easily identify these backdoored encoders with 100.0% detection accuracy as shown in Section 6.3. Adopting $\mathcal{L}_{conc}$ greatly mitigates the concentration of the poisoned distribution, resulting in less than 0.4 pairwise similarity and 50.0% detection accuracy by DECREE (reduced from 100.0% without $\mathcal{L}_{conc}$ in DRUPE).

Impact of Reference Inputs Selection. We study different selections of three reference inputs. For outliers, we purposely choose all three inputs whose embeddings are outliers in the target class. For simple and complex data, we use an existing metric [58] to select corresponding samples. We also conduct 5 runs of random selection with different random seeds and report the average. The results are reported in Table 12. Most selections of reference inputs achieve >96% ASR. Even in the most challenging scenario where all three references are outliers, DRUPE still has near 70% ASR (2nd row). This remains a severe problem as the attacker only needs to succeed once for critical systems. Using different numbers of reference inputs has negligible impact on DRUPE, which is reported in Appendix G.

TABLE 14: Results on different sizes of the shadow dataset

Size of Shadow Dataset	BA	ASR	SIM-C	SIM-B	DD
5%	80.17%	84.60%	0.22	0.37	0.86
10%	80.12%	90.22%	0.22	0.35	0.84
20%	80.35%	97.22%	0.23	0.34	0.92
30%	80.29%	99.14%	0.25	0.37	0.89

TABLE 15: Results on different trigger sizes

Trigger Size	BA	ASR	SIM-C	SIM-B	DD
5x5	75.20%	96.31%	0.37	0.40	0.68
7x7	75.60%	96.38%	0.35	0.38	0.61
10x10	75.64%	96.68%	0.36	0.37	0.63
12x12	76.08%	97.07%	0.35	0.36	0.60

Appendix G. Impact of Hyper-parameters

In this section, we study the impact of different hyper-parameters on DRUPE. Specifically, we consider different numbers of reference inputs, different sizes of the shadow dataset (used during the attack), and different trigger sizes.

Different Numbers of Reference Inputs. The reference inputs are used for achieving the attack success rate on the downstream classifier. We vary the number from 2 to 10, and show the results in Table 13. The pre-training dataset CIFAR-10 [38] and the downstream dataset GTSRB [71] are used for the study. Results show that with the increase of the number of reference inputs, BA and ASR slightly increase. This is because more reference inputs make the approximation of the target-class distribution more accurate and hence lead to better attack performance. When the number of reference inputs is 2, the similarity among poisoned samples (SIM-B) is relatively high (0.49). The SIM-B becomes stable when the number of reference inputs is larger than or equal to 3.

Different Sizes of the Shadow Dataset. The shadow dataset is used by the attacker to inject the backdoor into a pre-trained encoder. Following exiting work [34], a percentage of the pre-training dataset is used as the shadow dataset. We study the impact of this value. We vary the size from 5% to 30%. The pre-training dataset CIFAR-10 [38] and the downstream dataset GTSRB [71] are used for the study. Results in Table 14 show that the ASR increases with the increase of the shadow dataset size. The ASR is higher than 90% when more than 10% of the pre-training dataset is used. The size of the shadow dataset has relatively minor impact on SIM-B and DD.

Different Trigger Sizes. We study the impact of different trigger sizes by varying it from 5x5 to 12x12. The pre-training dataset STL-10 [17] and the downstream dataset SVHN [51] are used in the study. The results are reported in Table 15. As can be observed, the attack performance of DRUPE are quite stable using different trigger sizes, demonstrating the stability of our attack.

Appendix H.
Meta-Review

H.1. Summary

The paper proposes DRUPE, a backdoor attack against self-supervised learning (SSL) in the image domain. Unlike prior attacks, DRUPE seeks to (1) decrease the distance between the features of trigger-stamped samples and clean target samples and (2) increase the pairwise similarity between trigger-stamped samples' features. In doing so, DRUPE can evade previous defenses that exploit the differing feature distributions and pairwise similarities between poisonous and clean inputs.

H.2. Scientific Contributions

- Addresses a Long-Known Issue
- Provides a Valuable Step Forward in an Established Field

H.3. Reasons for Acceptance

1) Evasion of state-of-the-art defenses. The development of distribution-preserving backdoors enables evading state-of-the-art defenses, providing a more accurate evaluation of their performance.
2) Advancement compared to previous work. The experiments convincingly demonstrate the superiority of DRUPE compared to prior attacks.
3) Comprehensive experiments. Extensive experiments with five datasets evidence the effectiveness of DRUPE at backdooring foundation models.

H.4. Noteworthy Concerns

1) The paper is similar in spirit to prior work (e.g., [69]). In particular, in both lines of work, the adversary designs an adversarial model by minimizing the statistical difference between the embedding of clean and backdoor inputs while preserving benign accuracy. The primary difference is that this paper adapts attacks to SSL, enabling adversaries to create backdoored foundation models that can evade detection.
2) The findings are constrained to the image domain. While the paper discusses means to extend the approach to the NLP domain (Section 8), where backdoor attacks have been extensively studied, experiments assessing the attack's success in this domain are missing.
3) Dated SSL algorithms. Because the paper primarily experiments with somewhat dated SSL algorithms (SimCLR) and model architectures (ResNets), DRUPE's effectiveness against the most recent approaches remains to be determined. Preliminary results in Appendix E hint that the attack could be effective against a recent SSL algorithm [95], but the experiments are not sufficiently comprehensive.

Appendix I.
Response to the Meta-Review

The meta-review notes that our paper is similar to prior work AdvEmbed [69]. We would like to highlight that our attack DRUPE is different from prior work AdvEmbed. There are two major challenges in backdooring SSL: (1) reducing the distributional difference between clean and poisoned data; (2) mitigating the concentration of poisoned distribution. Existing work AdvEmbed considers the first challenge but does not address the second problem. In addition, AdvEmbed uses an adversarial network to differentiate whether an input is malicious to assist the attack, which requires additional computational resources and is often hard to train. DRUPE uses sliced-Wasserstein distance to reduce the distributional difference of clean and poisoned data. The distributional distance of poisoned encoders by AdvEmbed is almost three times as large as that of DRUPE. Furthermore, AdvEmbed can be easily detected by existing defenses such as DECREE [23] with 100% accuracy, whereas DRUPE cannot be detected.