JOURNAL OF LATEX CLASS FILES, VOL. 14, NO. 8, AUGUST 2021

SSLDefender: Backdoor Defense in Self-Supervised Learning via Distillation-guided Unlearning

Jiale Zhang, *Member, IEEE,* Wanquan Zhu, Kai Wang, Chengcheng Zhu, Xiaobing Sun, *Member, IEEE,* Weizhi Meng, *Senior Member, IEEE,* and Xiapu Luo

***Abstract*—Self-supervised learning utilizes unlabelled data to train encoders, acquiring high-quality representations of input data, significantly advancing the field of computer vision. However, recent studies have demonstrated that self-supervised learning suffers from numerous adversarial attacks. Among them, backdoor attack is one of the focal issues, where downstream classifiers inherit the backdoor behavior of the pre-trained encoder. Existing defense methods against backdoor attacks primarily focus on supervised learning, which heavily relies on labeled data and cannot be directly migrated to self-supervised scenarios. Furthermore, defense methods for self-supervised backdoor aims to separate poisoned samples on assumed small-scale datasets and retraining to obtain a clean encoder. However, these approaches are useless against encoders that have been implanted with a backdoor. To address these issues, we propose SSLDefender, a novel image-based backdoor mitigation method specially designed for self-supervised learning, which can remove backdoor attributes directly from the backdoor encoder. Specifically, we employ a trigger recovery method based on mutual information maximization to efficiently obtain trigger that resembles the target backdoor's influence. Additionally, we design a distillation-guided unlearning strategy to purify backdoor features steadily and ensure the retention of clean knowledge to prevent over-forgetting. Extensive experimental evaluations on six benchmark datasets demonstrate that SSLDefender can successfully reduce the attack success rate of Badencoder to around 2% while maintaining high model accuracy on the main task. Its performance surpasses state-of-the-art methods.**

***Index Terms*—Self-supervised learning, Backdoor attacks, Trigger recovery, Knowledge distillation, Unlearning.**

I. Introduction

SElf-supervised learning (SSL) is a machine learning paradigm that leverages unlabeled data for training, eliminating the dependency on annotated samples [1]–[3]. It has exhibited substantial promise across diverse domains, including computer vision [4], [5] and natural language processing [5], [6]. In contrast to traditional supervised learning [7], SSL aims to acquire knowledge from the data itself, obtain high-quality representations of the data, and construct a pre-encoder to enable downstream tasks [8]. However, existing research has indicated that SSL is susceptible to the threat of backdoor attacks [9].

In order to embed and activate backdoors without directly manipulating labels, the implementation of backdoor attacks in SSL differs from supervised learning. In supervised learning, attackers establish a strong correlation between the trigger and the target label in a low-dimensional label space to carry out backdoor attack [10]. However, in SSL, each pre-trained encoder only outputs embedded features of input data, and the prediction process relies on downstream classifiers. Therefore, attackers in SSL generate similar embeddings for all inputs containing triggers and the target class. As a result, any downstream classifier constructed based on a backdoor encoder will incorrectly classify inputs with similar triggers into the same target class [11]. As illustrated in Figure.1, encoders trained under supervised learning rely on label guidance to classify any input carrying triggers into the target label predetermined by the attacker. In SSL, the attackers indirectly influence the label space solely through the form of feature representations, directly linking the trigger pattern to the target class in the label space [12].

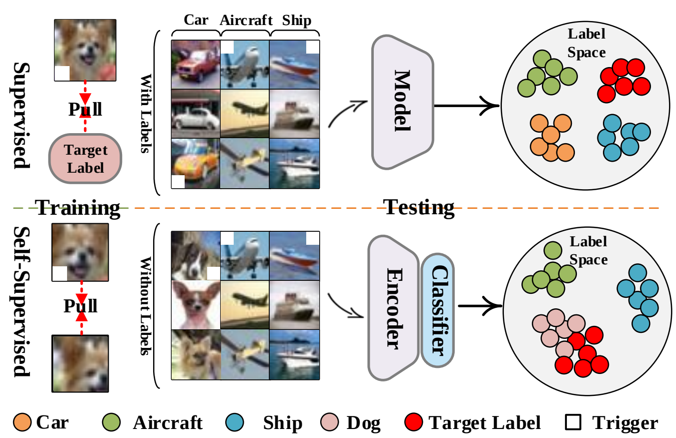


Fig. 1: Comparison of supervised and self-supervised backdoor attacks.

To defend against backdoor attacks in SSL, researchers have explored two directions for solutions: backdoor detection and backdoor mitigation. ❶Backdoor detection methods in SSL attempt to define the backdoor trigger as a constraint problem and distinct the existence of the backdoor for the target encoder by comparing the size of the inverted trigger with an empirical threshold [13]. However, discarding a pre-trained encoder incurs significant costs in SSL. Consequently, while backdoor detection method can preemptively identify backdoors, it is powerless to mitigate the malicious impact

J. Zhang and W. Zhu contributed equally to this work.

J. Zhang, W. Zhu, K. Wang, C. Zhu, and X. Sun are with the School of Information Engineering, Yangzhou University, Yangzhou, China, 225127 (e-mail: {jialezhang,xbsun}@yzu.edu.cn; {MX120230578,MZ120241054,MX120220554}@stu.yzu.edu.cn).

W. Meng is with the School of Computing and Communications, Lancaster University, United Kingdom (e-mail: weizhi.meng@ieee.org).

X. Luo is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: csxluo@comp.polyu.edu.hk).).

J. Zhang and C. Zhu are the corresponding authors.

of this attack [14]. ❷Backdoor mitigation methods aim to break the correlation between the trigger and the target label, thereby preventing the encoder from being compromised by the backdoor attack and ensuring its normal functionality. On the one hand, in SSL, some methods like PatchSearch [15] and SSL-Cleanse [15] employ clustering-based approaches to separate poisoned samples and retrain a clean encoder on clean samples. Additionally, some methods based on self-supervised knowledge distillation perform neural attention distillation by fine-tuning the target encoder and obtaining a purified encode [16]. On the other hand, in supervised learning, researchers have attempted various methods such as fine-tuning [17], neuron cleansing (NC) [18], adversarial neuron pruning (ANP) [19], model connection repair (MCR) [20], neuron attention distillation (NAD) [21], self-attention distillation (SAGE) [22], and unlearning [23] to eliminate the impact of backdoors.

Motivation: However, existing defense methods have the following limitations in addressing backdoor attacks in SSL: 1) Backdoor mitigation methods in supervised learning rely on label guidance and correction, making the model robust against attacks. However, simply transferring supervised backdoor defense methods, such as knowledge distillation, to the self-supervised scenario is challenging due to the lack of given labels, preventing this method from achieving the same performance as in supervised learning. When the purified encoder is transferred to downstream classification tasks, even if the attack success rate is successfully suppressed, the accuracy of the main task inevitably decreases. 2) Detection strategies cannot remove the malicious impact of backdoor encoders. Therefore, when performing downstream classification tasks, classifiers trained based on this encoder will still retain the original relationship between the trigger and the target label. 3) In SSL backdoor mitigation, clustering-based methods rely on an assumed small-scale dataset, purifying the dataset through toxic sample filtering, and training a clean encoder. These methods belong to data sanitization techniques, serving as a defense against data poisoning before encoder training. They are ineffective against pre-trained backdoor encoders, such as the BadEncoder method, which maliciously modifies a clean encoder. In summary, our exploration in this aspect raises a fundamental yet profound question: "*How can we directly purify a backdoor encoder while ensuring the accuracy of downstream task classification?*"

Challenges: Purifying backdoors in the context of SSL poses three challenges that need to be addressed: 1) how to achieve 'unlabeling', i.e., breaking the limitations of label-dependent backdoor defense methods to make them applicable in SSL scenarios; 2) how to forget backdoor features by maximizing the reduction of the backdoor's impact on the encoder; 3) how to ensure the performance of the target encoder by minimizing the negative impact of defense methods on the entire SSL process.

To this end, this paper proposes a distillation-guided unlearning approach for backdoor mitigation, called SSLDefender. It initially achieves trigger recovery without labels by computing the embedding similarity of input sample pairs. Subsequently, leveraging the recovered trigger, SSLDefender employs the unlearning mechanism to mitigate the backdoor's impact. To ensure the accuracy of the primary task, a teacher model is constructed to guide the training of the backdoor encoder. Our contributions can be summarized as below.

- **A Novel Backdoor Defense Method**: to mitigate the influence of backdoors on pre-trained encoders, our SSLDefender breaks the connection between trigger features and target label through distillation-guided unlearning.
- **Trigger Recovery**: to quickly acquire knowledge of the backdoor and carry out subsequent mitigation tasks, we employ a label-independent trigger recovery method based on mutual information maximization.
- **Distilled-Guided Unlearning**: to ensure encoder performance while achieving superior defensive performance, we propose a strategy called distilled-guided unlearning. The pre-trained encoders not only counter backdoor attacks through unlearning but also maintain the accuracy of the primary task via distillation learning, thereby achieving a robust balance between the two objectives.
- **Comprehensive Evaluation**: we conduct experiments on SSLDefender with six benchmark datasets. The experimental results demonstrate that our SSLDefender can effectively mitigate the backdoor in the encoder while maintaining high performance in downstream classification tasks.

The remainder of this paper is organized as follows. In Section II, we discuss the background and related works. In Section III, we describe the problem definition and the threat model. In Section IV, we introduce our proposed SSLDefender method. Section V demonstrates the performance evaluation results. Finally, Section VI concludes this paper.

II. Background and Related Work

A. Self-supervised Learning

Self-supervised learning has attracted widespread attention and implementation because its remarkable performance does not rely on sample labels and involves extensive data training [24]–[29]. Self-supervised learning models typically consist of two components: a high-quality encoder f and a downstream classifier g, forming a final model $h : f \circ g$ together. The encoder constructs a function $f : X \to E$, where X is the input space containing different sample inputs, and E is the embedding space containing corresponding feature vectors. Contrastive learning (e.g., SimCLR [30], SimCLRv2 [31], MoCo [32] and CLIP [33]) has achieved outstanding results among numerous training methods for self-supervised learning encoders. Contrastive learning forms similar instance pairs for inputs, making positive samples closer to each other and negative samples farther apart in the embedding space. Enhanced versions from the same input are considered positive samples, while enhanced versions from different samples are considered negative. Another approach, BYOL [34], trains only with positive samples in the absence of negative samples. The trained encoder can be used for various downstream tasks.

B. Backdoor attacks in SSL

Self-supervised learning aims to train encoders from large amounts of uncurated data, which opens up backdoor opportunities. Encoders embedded with backdoors can deceive downstream classifiers by leveraging their unique trigger patterns, leading to erroneous judgments when receiving inputs carrying the triggers. However, the downstream classifiers perform normally on clean inputs. Saha et al. [9] introduced triggers into randomly cropped augmented views, bringing them closer to each other in the embedded space compared to other views with the same augmentation, enabling the encoder to learn the association between triggers and target classes. Building upon this, Li et al. [35] ensured the concealment of triggers by employing Discrete Cosine Transform (DCT) [36] to define spectral perturbations that are invisible in the chromatic space. Unlike image patches, spectral triggers exhibit enhanced resistance, demonstrating higher effectiveness and evasion during testing. Zhang et al. [37] theoretically derived the optimal size for background images, and the best positions for reference objects and triggers, to create optimal poisoned images and address some limitations of the approaches above. Jia et al. [38] constructed BadEncoder to generate similar feature vectors for reference inputs (target classes from downstream tasks) and shadow datasets (carrying triggers), thereby transferring the influence of the poisoned encoder to any arbitrary downstream classifier. In the multimodal domain, Carlini et al. [39] built two encoders: an image encoder and a text encoder, projecting corresponding image-text inputs into the same embedding space and generating similar embedding vectors. Effective attacks could be executed by controlling only 0.01% of the data. However, Tao et al. [40] argued that the critical issue with existing attack methods lies in the out-of-distribution nature of poisoned data, which can be easily detected by advanced detection techniques. To address this, they proposed DRURE, a distribution-preserving backdoor attack that reduces the distribution distance between poisoned samples and clean data [41], [42], transforming poisoned samples into in-distribution data, and achieving stealthy attacks that are difficult to detect.

C. Backdoor Defense in SSL

Existing defense methods against backdoor attacks in SSL primarily include two approaches: backdoor detection and backdoor mitigation. DECREE [13] was a typical model-centric backdoor detection method that performed trigger recovery on the target encoder by minimizing the similarity between pairs of samples embedded with triggers generated from random noise. If the size of the inversed trigger was smaller than a given threshold, the encoder was identified as a backdoor encoder. Otherwise, it was considered normal. However, this passive detection method can only determine the presence of a backdoor threat in the model. Still, it cannot eliminate the negative impact of the backdoor attack on the target model.

In contrast, backdoor mitigation methods aim to eliminate triggers and cleanse the backdoored model by severing the strong correlation between triggers and target labels. Data-level backdoor mitigation methods can generally be divided into three parts: 1) identifying poisoned samples, 2) removing poisoned samples, and 3) retraining on clean samples. Examples of such methods include PatchSearch [15] and SSL-Cleanse [43]. To the best of our knowledge, in the latest efforts to mitigate SSL backdoors, Bie et al. [16] employed a knowledge distillation approach on the backdoor encoder. They adapted the method used in NAD [21] from supervised learning and transferred it to SSL. Their mitigation of Badencoder in non-targeted attack scenarios proved to be highly effective, demonstrating superior performance. However, their focus was not on real-world scenarios of SSL but rather on extensive comparisons with existing backdoor attacks in supervised learning, overlooking the attacks in existing SSL. Furthermore, as this method did not provide actual code, we could only replicate it based on the NAD method.

III. Threat Model and Defense Goal

A. Threat Model

We focus primarily on malicious setups in image encoders, where attackers employ illicit means to inject carefully designed backdoor into pre-trained encoders, thereby disrupting the correct classification by downstream classifiers relying on these encoders. We present our threat model from the perspectives of the attacker and defender. Based on recent backdoor attack methods, we categorize the capabilities of attacker and defender as follows:

- An attacker can construct a backdoor encoder using any means, including crafting poisoned samples and compromising clean encoders. The attacker can balance attack effectiveness and evasion, ensuring that backdoor samples exhibit high attack success rates on downstream classification tasks without affecting the prediction accuracy of clean samples.
- The defender can only passively obtain backdoor encoder and remain unaware of the backdoor knowledge. Furthermore, apart from holding a small portion of unlabeled clean data, the defender have no access to any other relevant data.

B. Defense Goals

In light of the specific capabilities of the attacker, we address our defense objectives in a targeted manner from two aspects: **Defense Effectiveness**: SSL-Defender can effectively purify the backdoor encoder, remove backdoor features, and sever the strong connections between triggers and target labels. When this encoder is transferred to downstream classification tasks, malicious inputs carrying triggers cannot force the classifier to produce misclassifications, significantly reducing the attack success rate. **Model Robustness**: The prediction accuracy on clean inputs should be comparable to or slightly lower than the accuracy before SSL-Defender training. In other words, within an acceptable range, the accuracy of the main task should be maintained.

Additionally, we evaluate the defense objectives using two primary criteria: the Attack Success Rate (ASR) on backdoor samples and the Model Accuracy (ACC) on normal samples.

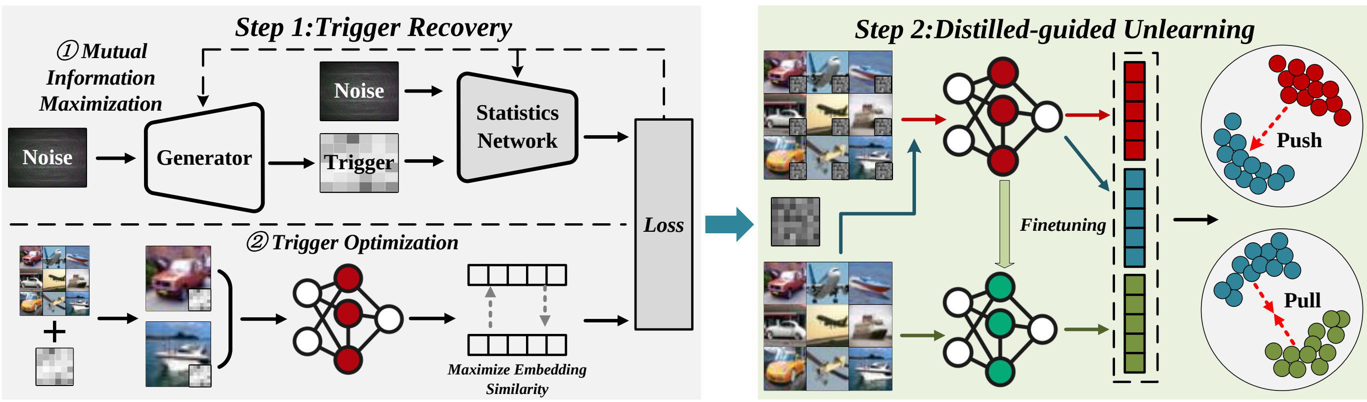


Fig. 2: Framework of the proposed SSLDefender. “Red” and “‘Blue” represent the poisoned and clean outputs of the student model, respectively, while “Green” represents the clean output of the teacher model.

IV. Proposed Defense Method

A. Overview

Figure. 2 outlines our proposed framework of SSLDefender, designed specifically for backdoor defense in SSL. SSLDefender instructs the target shadow encoder to perform trigger recovery training based on mutual information maximization to obtain optimized triggers. This trigger, with approximate influence as set by the attacker’s backdoor, effectively captures knowledge of existing maliciously impactful backdoor attributes. Furthermore, to leverage the acquired backdoor knowledge and cleanse model’s backdoor attributes, SSLDefender introduces distilled-guided unlearning.

B. Trigger recovery

Trigger recovery has been widely used in supervised learning. Such methods inspired by the intuition of backdoor attacks, where the modification by attackers for misclassifying target labels is much smaller compared to clean labels, have traversed model labels and optimized trigger patterns under assumed labels to find the minimal trigger that misclassifies other labels as the assumed label. Building on this, outlier detection algorithms are employed to filter out true triggers and their corresponding target labels. However, this method relied on explicit labeling for target optimization and is not applicable to self-supervised learning. Feng et al. [13], based on observations of backdoor trigger patterns where samples carrying the same trigger exhibit highly similar embeddings, proposed a new solution by guiding trigger optimization through maximizing embedding similarity. However, they are limited to setting a threshold in this optimization process, whereby if the value exceeds this threshold, the encoder is deemed to be carrying a backdoor. Although this method can accurately determine the presence of a backdoor, it cannot acquire knowledge of the backdoor, thus impeding mitigation efforts. We have achieved a more lightweight and precise trigger through a trigger recovery strategy based on mutual information maximization.

Firstly, we formalize trigger injection using the following equation:

$$Mix(x_i, \Delta) = x_i', \quad (1)$$

When injecting a backdoor into the target encoder, we observed that the model learns backdoor knowledge much faster than clean data. Even on datasets that are challenging to converge, the model tends to converge more easily towards backdoor data. In causal reasoning, this phenomenon is explained as the attacker opening a false “shortcut” between the input images and the predicted labels. If the model has already learned the relevance of this false path, then when triggers are attached, their predictions will switch to the target label. Additionally, the model will generate highly similar feature embeddings for any input embedding such triggers. Therefore, we guide the process of pre-set trigger optimization by creating poisoned samples and computing the similarity between them to restore triggers that approximate the original backdoor influence and lead to optimal misclassification by the model. Specifically, for a randomly generated noise δ, we use the generation model G to generate the optimized trigger Δ. Assuming a clean shadow dataset D_{shadow}, we embed the optimized trigger through the mixing function $M(.)$ to construct the poisoned dataset D'_{shadow}. Typically, in SSL, cosine similarity is employed to measure the similarity between two embedding samples. Formally, given two inputs x_p and x_q, the cosine similarity between their corresponding trigger embedding samples can be represented as:

$$L_{p,q}(F, \Delta) = -cos(F(Mix(x_p, \Delta)), F(Mix(x_q, \Delta))), \quad (2)$$

Moreover, to achieve high similarity between samples and approximate search in dense regions of the backdoor model embedding space, it is necessary to sample a batch of input samples to stabilize the search process. The calculation of the average pair similarity within batch B is as follows:

$$L_{cos} = \frac{1}{B^2} \sum_{p=1}^{B} \sum_{q=1}^{B} \mathcal{L}_{p,q}, \quad (3)$$

the loss of L_{cos} serves as a constraint during the trigger optimization process, ensuring that samples carrying the trigger to be optimized tend to cluster in dense regions of the embedding space.

However, this typical generative model has been proven to struggle in estimating the differential entropy in high-dimensional trigger patterns, leading to a decline in model performance [44]. Therefore, we introduce the maximization of mutual information to address this issue. Mutual information is a measure of dependency between random variables based on Shannon entropy. The mutual information between X and Z can be understood as the reduction in uncertainty of X given Z. The calculation of their mutual information through the Mutual Information Neural Estimator (MINE) can be represented as:

$$I(X;Z) = H(X) - H(X \mid Z), \quad (4)$$

where H is the Shannon entropy, and $H(X \mid Z)$ is the conditional entropy of Z given X.

Specifically, we employ the enhanced algorithm of Mutual Information Neural Estimator (MINE), known as Maximum Entropy Staircase Approximation (MESA), to approximate the unknown trigger distribution by integrating a set of sub-models $G = \{G_1, G_2 ... G_n\}$, where each sub-model G_i learns a portion of the trigger Δ_i. Additionally, we uniformly select n thresholds $\epsilon = \{\epsilon_1, \epsilon_2 ... \epsilon_n\}$ from $[0, 1]$, where each threshold β_i corresponds to a sub-model G_i and an information estimator I_{T_i} parameterized by a statistical network T_i. Consequently, the final optimized loss function becomes:

$$\min_{\theta_g} \mathcal{L}_t = \sum_{i=1}^{n} (max(0, \epsilon_i - L_{cos}) - \eta I_{T_i}(G_i(\delta); \delta')). \quad (5)$$

We compute the mutual information between the randomly initialized noise δ' and the optimized trigger through a statistical network. The process of maximizing mutual information guides the optimization iterations to be more expedient and effective. Our method aims to expedite the restoration of the most influential backdoor by seeking the most similar triggers. This conclusion will be verified in Section IV. The complete process of trigger recovery is shown in Algorithm 1.

Algorithm 1: Trigger recovery

Input: Shadow dataset $D_{shadow} = \{x_1, x_2 ..., x_n)\}$,generation model $G = \{G_1, G_2 ... G_n\}$,thresholds $\epsilon = \{\epsilon_1, \epsilon_2 ... \epsilon_n\} \in [0, 1]$,random nosie δ and δ'
Output: Optimized tigger Δ
Formalize trigger injection;
for *each sample* $x_i \in D_{shadow}$ **do**
  $Mix(x_i, \Delta) = x_i'$;
  $D'_{shadow} \leftarrow D_{shadow}$;
end
for *any sample* x_p *and* $x_q \in D'_{shadow}$ **do**
  $L_{p,q} = -cos(E(Mix(x_p, \Delta)), E(Mix(x_q.\Delta)))$;
  // computer the similarity between two embedding sample;
  $L_{cos} = \frac{1}{N^2} \sum_{p=1}^{N} \sum_{q=1}^{N} \mathcal{L}_{p,q}$;
  // calculate of the average pair similarity within batch N;
  $\mathcal{L}_{total} = \sum_{i=1}^{n} (max(0, \epsilon_i - L_{cos}) - \eta I_{T_i}(G_i(\delta); \delta'))$;
  // final optimized loss function;
end
Iterative training until convergence;
Return Δ;

C. Distilled-guided unlearning

In deep learning, unlearning means that the data owners wish the model owner to erase the influence of their data on the model and request that the model owner no longer use this these data for training. For defenders, we also aim to utilize this technique to erase backdoor features to purify the model. The most effective and straightforward method for unlearning is to retrain the model using a training set that does not include the supplier's data. However, in SSL scenarios, where the user aims to obtain an encoder that provides high-quality representations of the data, extensive training on unlabeled data is necessary. The computational cost of retraining becomes prohibitive. Therefore, we seek a method to conduct unlearning on backdoor knowledge for the backdoor encoder.

Considering the existing trigger patterns, the next step for SSLDefender is to leverage the recovered trigger to eliminate malicious trigger functionalities. To maximize the use of limited clean data, we employ a distillation-guided unlearning strategy through a lightweight teacher-student framework for unlearning. The objective of this strategy is to effectively forget malicious features while ensuring model performance. In a successful SSL backdoor attack, attackers tend to modify a clean encoder to generate similar embeddings for all inputs containing triggers and target classes. Consequently, any downstream classifier built on this encoder backbone will erroneously classify inputs with similar triggers into the same target class. Based on the above observation, a direct defense intuition of ours is to force trigger inputs and original inputs of the same sample to have similar distributions, thereby weakening the effectiveness of backdoor attacks. Specifically, we embed all samples from the limited clean dataset D_{clean} with the recovered trigger to construct a poisoned dataset D_{poised}. For a clean sample x and its corresponding poisoned sample x', both are considered inputs for the student model. By aligning their embedding generation operations, we implicitly drive the student model to eradicate the backdoor.

However, this brute-force operation inevitably leads to a decline in model performance. While minimizing the difference between trigger and clean input distributions, it causes a shift in the distribution of clean data as well. Therefore, we fine-tune the student model to obtain a relatively clean teacher model that guides the student model in preserving clean knowledge. The teacher model only receives clean data x as input. We align the clean soft targets outputted by the teacher model with the outputs of the student model for the same inputs. The clean and useful information from the teacher model is passed to the student to aid in forgetting trigger

Algorithm 2: Distilled-guided unlearning

Input: clean dataset $D_{clean} = \{x_1, x_2 ..., x_n)\}$,recovered trigger Δ, Hyperparameter α and β.
Output: Purified encoder f.
Formalize trigger injection;
Perform iterations for mutual information transfer:
for *each sample* $x_i \in D_{clean}$ **do**
  $D_{poised} \leftarrow D_{clean} + \Delta$;
end
for *each sample* $x_i \in D_{clean}$ *and* $x'_i \in D_{poised}$ **do**
  $z_c \leftarrow f(x)$; $z_b \leftarrow f(x')$;
  $\tilde{W}_2(\mathbb{P}_c^S, \mathbb{P}_b^S) = \left[\int_{\omega\in\Omega} W_2^2(\mathbb{P}_c^\omega, \mathbb{P}_b^\omega) d\omega\right]^{\frac{1}{2}}$;
  // The discrepancy between the trigger input and clean input of the student model;
end
for *each sample* $x_i \in D_{clean}$ **do**
  $\tilde{W}_2(\mathbb{P}_c^S, \mathbb{P}_c^T) = \left(\frac{1}{M}\sum_{m=1}^{M}\int_0^1 \|F_S^m(z_c) - F_T^m(z_c)\|_2 dz\right)^{1/2}$;
  // The discrepancy between the clean input of the teacher model and the student mode;
end
Update model parameters to min the loss;
$L_{total} = \alpha\tilde{W}_2(\mathbb{P}_c^S, \mathbb{P}_b^S) + \beta\tilde{W}_2(\mathbb{P}_c^S, \mathbb{P}_c^T)$;
Return Purified encoder f.

features. Benefiting from this, the clean output distribution of the student model closely aligns with the teacher model, which can mitigate the adverse effects of over-forgetting on normal samples.

We utilize the Wasserstein distance to compute differences between different distributions [45]. Generative modeling [46] is the task of learning the probability distribution from a given dataset $D = \{x\}$, where samples $x \sim \mathbb{P}_b$ are drawn from an unknown data distribution $\mathbb{P}_b$. Formally, the Wasserstein-p distance between distributions $\mathbb{P}_c$ and $\mathbb{P}_b$ is defined as:

$$W_p(\mathbb{P}_c, \mathbb{P}_b) = \inf_{\gamma\in\Pi(\mathbb{P}_c,\mathbb{P}_b)} (\mathbb{E}_{(x,y)\sim\gamma}[||x-y||^p])^{\frac{1}{p}}, \quad (6)$$

Given the constraint of only being able to utilize a small portion of the test dataset, to ensure that the model fully extracts all limited knowledge, we directly employ the feature representations of the data. This also implies that our data is high-dimensional. Estimating the Wasserstein distance on high-dimensional data is not a trivial task. To alleviate computational complexity, we adopt a sliced version of the Wasserstein-2 distance [47], [48], which requires estimating distances between one-dimensional distributions, thus enhancing efficiency. Therefore, the discrepancy between the trigger input and clean input of the student model can be defined as:

$$\begin{aligned}\tilde{W}_2(\mathbb{P}_c^S, \mathbb{P}_b^S) &= \left[\int_{\omega\in\Omega} W_2^2(\mathbb{P}_c^\omega, \mathbb{P}_b^\omega) d\omega\right]^{\frac{1}{2}} \\ &= \left(\frac{1}{M}\sum_{m=1}^{M}\int_0^1 \|F_S^m(z_c) - F_S^m(z_b)\|_2 dz\right)^{1/2},\end{aligned}$$

Similarly, the distribution discrepancy between the clean inputs of the teacher model and the student model is:

$$\tilde{W}_2(\mathbb{P}_c^S, \mathbb{P}_c^T) = \left(\frac{1}{M}\sum_{m=1}^{M}\int_0^1 \|F_S^m(z_c) - F_T^m(z_c)\|_2 dz\right)^{1/2}, \quad (7)$$

By employing the aforementioned procedure for unlearning, the overall loss constraint is determined by minimizing the differences between two pairs of distributions, which can be expressed as:

$$L_{total} = \alpha\tilde{W}_2(\mathbb{P}_c^S, \mathbb{P}_b^S) + \beta\tilde{W}_2(\mathbb{P}_c^S, \mathbb{P}_c^T). \quad (8)$$

The complete process of distilled-guided unlearning is shown in Algorithm 2.

V. Performance Evaluation

In this section, we conduct multiple experiments of SSLDefender on four real-world datasets under four SOTA backdoor attack methods in SSL. To evaluate the effectiveness of our proposed method, we aim to answer three key research questions:

- **RQ1** (Defense Effectiveness): Can our proposed method conduct effective defense against the four SOTA backdoor attacks on different datasets?
- **RQ2** (Ablation Analysis): Is SSLDefender still effective in the elimination of tigger recovery or distillation-guided unlearning methods?
- **RQ3** (Parameter Sensitivity Analysis): What is the effect of SSLDefender in different hyperparametric settings?

A. Experimental Settings

1) Dataset: SSLDefender is evaluated on five widely-used benchmark datasets: CIFAR-10, STL-10, GTSRB, SVHN, and ImageNet. The basic statistics of each dataset are shown in Table. I, including the number of training and testing samples, sample categories, and individual sample sizes. Particularly, SVHN is a dataset composed of noisy samples, where some distractor digits are distributed around the primary digit represented by the sample.

2) Attack Methods: We investigate five state-of-the-art (SOTA) backdoor attack methods in SSL: SSL-backdoor embeds image patches into one view of the contrastively learned enhanced images to establish a strong correlation between triggers and target labels. PoisonedEncoder poisons specific inputs, and combines target and reference inputs to create poisoned samples. CorruptEncoder carefully crafts poisoned images with two randomly cropped views that have a high probability of including the reference object and trigger. In

TABLE I: Dataset statistics.

Dataset	Training images	Testing images	Classes	Size
CIFAR-10	50,000	10,000	10	32×32×3
STL-10	5,000	8,000	10	96x96x3
GTSRB	39,200	12,600	43	32x32x3
SHVN	73,257	26,032	10	32x32x3
Tiny-ImageNet	128,116	5,000	100	224x224x3

contrast to the aforementioned methods, BadEncoder constructs a backdoor encoder using reference inputs to transfer to different downstream classification tasks.

3) Defense Baselines: We select four backdoor defense methods in SSL, i.e., DECREE, PatchSearch, SSL-Cleanse, and SSL-KD. Here, DECREE belongs to backdoor detection, while the other methods are categorized as backdoor mitigation. PatchSearch and SSL-Cleanse are clustering-based data filtering and retraining methods. SSL-KD is a mitigation method for poisoned encoders.

4) Evaluation Metrics: We evaluate the performance of defense mechanisms with two metrics: attack success rate (ASR), which is the ratio of backdoored samples misclassified as the labels specified by attackers, and the accuracy of the main classification task on normal samples (ACC).

5) Implementation Details: SSLDefender is assumed to be able to access 5% of the clean data randomly selected from the test set. For the unlearning process, we use the loss term $B = 0.5$, batch size $B = 256$, SGD as optimizer with learning rate η = 0.001, and run for E = 500 epochs. For all the baseline attacks and defenses, we adopt the default hyperparameters recommended by the corresponding papers. Specifically, attacks have the common parameters: trigger size t and injection ratio Φ. Unless otherwise mentioned, we set the backdoor injection ratio to $\Phi = 5\%$ and the trigger size t to 20%. We test the performance of SSLDefender as well as other baselines five times and report the mean and standard deviation results to eliminate the effects of randomness.

6) Experimental Environment: We implemented SSLDefender in Python using the PyTorch framework. Our experimental environment consists of 13th Gen Intel(R) Core(TM) i7-13700KF, NVIDIA GeForce RTX 4070 Ti, 32GiB memory, and Ubuntu 20.04 (OS).

B. RQ1: Backdoor Defense Performance

To answer RQ1, we evaluate the defense effectiveness and model accuracy of SSLDefender under both targeted attacks and untargeted attacks scenarios. In targeted attacks, we compare the performance of SSLDefender against four self-supervised backdoor attacks across three benchmark datasets under three contrastive learning paradigms. We select the best comparative training method to transform the method in supervised learning and use it as a baseline to compare with our method. The "before" column represents the original baseline without any defense, and the best results are highlighted in bold. In untargeted attacks, we train models on three different downstream tasks and compare SSLDefender with the state-of-the-art defense method SSL-KD to verify the superiority of SSLDefender.

1) Targeted attacks: We evaluated the effectiveness of defense against representative SSL backdoor attack methods on benchmark datasets. For fair comparison, all attacks were conducted using the same settings as the original work. The results are summarized in Table. II. Overall, even under different contrastive learning algorithms, SSLDefender significantly defends against four typical existing self-supervised backdoor attacks, resulting in ASR below 9%, with only a slight decrease in ACC.

Specifically, in all settings, BadEncoder exhibits the highest attack effectiveness. For instance, when training the backdoor model using the MoCo algorithm, it achieves an ASR of 99.61% on CIFAR-10, which is nearly a perfect attack method. We analyze that this is due to the attacker manipulating the clean encoder with a reference input and trigger to make the sample carrying this trigger highly similar to the embedding of the reference input, thus achieving an effective attack. However, this aggressive manipulation, while demonstrating significant attack effects, is highly vulnerable. Defenders only need a few clean samples to retrain the backdoor encoder, leading to a substantial reduction in ASR. Therefore, in the same setting, we lower the ASR of this backdoor encoder to 0.97%. Additionally, the bidirectional optimization we employed ensures that the model's original performance is not compromised, resulting in a decrease of only 0.37% in ACC in this setup. Even on Tiny-ImageNet, we achieve an ASR of 0.2% and kept the ACC loss within 5%.

2) Comparing with Baselines: The aforementioned experiments demonstrated the superiority of the BYOL method over other commonly used contrastive learning approaches. Therefore, we adapt several well-established backdoor defense methods from supervised learning, namely fine-tuning, fine-pruning, neural cleanse, and NAD, into a label-free SSL paradigm consistent with the BYOL training framework. Since SSL-KD relies on downstream datasets to fine-tune the backdoor encoder, it is included in the discussion of untargeted attacks. The comparative results between the baseline methods and SSLDefender are presented in Table. III.

Through repeated experiments and cross-validation, it is evident that fine-tuning significantly reduces the ASR across all four attack methods. In particular, when defending against SSL-Backdoor attacks on the STL-10 dataset, fine-tuning achieved the best performance among all defense methods. However, the use of only a small subset of the dataset for fine-tuning cannot guarantee the preservation of the model's performance on its primary task. Moreover, our experiments reveal that fine-tuning suffers from high variability, struggling to strike a consistent balance between high ACC and low ASR, resulting in unstable outcomes. The intuition behind fine-pruning is to identify and remove low-activation neurons, which are presumed to be backdoor-related. While this method does suppress the success rate of backdoor activation , benign neurons or their informative features may also be pruned, inevitably leading to a significant drop in prediction accuracy.

Compared with these two approaches, Neural Cleanse(NC)

TABLE II: Performance of SSLDefender compared with baseline attacks on different pre-training datasets.

Attack	Pre-training Dataset	SimCLR				MoCo				BYOL			
		Before		After		Before		After		Before		After	
		ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
SSL-Backdoor	CIFAR-10	**77.04±2.59**	27.93±2.46	76.44±2.23	**4.36±1.03**	**75.11±2.85**	22.06±2.57	72.99±1.29	**5.00±3.50**	**89.40±2.51**	30.93±1.42	87.42±1.79	**2.09±4.18**
	Tiny-ImageNet	**69.22±3.41**	30.12±1.87	69.01±2.55	**6.53±0.92**	**67.48±4.13**	23.00±2.06	67.05±1.34	**4.88±3.67**	**69.81±0.78**	36.97±2.19	67.75±1.24	**2.96±4.02**
	STL-10	**68.00±1.76**	36.71±3.09	65.40±0.88	**7.29±2.41**	**60.30±4.67**	35.90±1.23	56.00±2.95	**8.20±0.54**	70.82±3.18	36.95±1.67	**73.19±2.04**	**6.68±4.32**
Poisoned-Encoder	CIFAR-10	**75.64±0.97**	32.93±2.58	73.68±1.41	**1.19±3.75**	**75.40±2.19**	23.31±4.06	67.45±0.73	**0.34±1.88**	**89.40±3.27**	32.45±0.65	88.73±2.33	**0.79±1.09**
	Tiny-ImageNet	60.09±4.51	32.72±1.37	**61.72±0.82**	**4.57±2.96**	59.88±3.44	26.04±1.05	**62.04±2.68**	**4.55±0.39**	**70.21±1.92**	39.12±4.13	66.70±0.57	**2.43±3.81**
	STL-10	**60.14±2.37**	39.90±1.05	59.11±3.88	**8.95±0.64**	**58.56±4.19**	37.15±2.71	57.23±1.44	**7.61±3.02**	70.92±0.79	39.09±2.15	**73.12±1.67**	**6.78±4.53**
Corrupt-Encoder	CIFAR-10	**78.40±0.92**	36.58±3.41	72.10±1.78	**0.54±2.06**	**74.76±4.67**	33.48±0.55	64.36±2.89	**0.66±1.33**	**89.80±3.12**	31.91±0.78	87.50±2.44	**0.96±1.95**
	Tiny-ImageNet	**66.65±1.56**	40.19±4.03	63.30±0.87	**1.78±2.71**	**63.30±3.45**	36.71±1.19	60.17±2.33	**0.92±0.41**	**71.03±4.88**	47.20±1.67	69.07±3.09	**0.77±2.22**
	STL-10	**67.00±2.14**	46.77±0.93	66.62±3.67	**3.02±1.45**	**58.75±4.02**	43.27±2.78	51.91±1.19	**2.79±0.56**	72.32±3.88	58.96±2.31	**75.05±1.07**	**2.48±4.55**
BadEncoder	CIFAR-10	80.98±1.82	98.92±3.41	**81.16±0.78**	**0.58±2.09**	**80.37±4.67**	99.61±1.33	80.00±2.95	**0.97±0.44**	83.20±3.12	98.99±1.67	**84.08±2.23**	**1.89±0.91**
	Tiny-ImageNet	**68.32±4.19**	97.99±0.66	66.04±2.55	**2.22±1.88**	**58.00±3.04**	97.41±2.71	53.80±1.45	**0.20±0.33**	**70.36±4.88**	98.58±1.09	64.36±3.56	**1.01±2.37**
	STL-10	67.04±1.67	96.75±3.22	**67.39±0.89**	**1.95±2.41**	**62.41±4.03**	94.60±1.15	62.33±2.78	**4.78±0.56**	72.96±3.91	99.98±1.34	**73.50±2.07**	**1.86±4.19**

TABLE III: Comparison results of different backdoor defense methods on three datasets.

Pre-training Datasets	Attacks	W/O Def		FT		FP		NC		NAD		SSLDefender	
		ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
CIFAR-10	SSLBackdoor	89.40±1.23	30.93±3.67	40.57±0.89	9.49±2.41	49.77±4.02	10.52±1.56	70.26±2.78	8.40±0.95	53.59±3.14	4.33±1.07	87.42±2.55	2.09±0.66
	PoisonedEncoder	89.40±1.88	32.45±4.19	49.93±2.33	8.24±0.77	52.14±3.45	8.56±1.92	62.45±0.58	15.26±2.71	57.74±1.34	4.50±3.09	88.73±0.82	0.79±2.04
	CorruptEncoder	89.80±0.91	31.91±2.67	50.39±1.45	8.08±3.88	52.43±0.66	11.15±2.19	65.25±1.07	4.26±4.33	80.93±2.95	2.51±1.56	87.5±0.44	0.96±3.41
	BadEncoder	83.20±2.78	98.99±1.23	50.17±3.67	13.32±0.89	59.75±2.41	17.62±4.02	66.38±1.56	14.26±2.95	60.17±0.95	7.24±3.14	84.08±1.07	1.89±2.55
Tiny-ImageNet	SSLBackdoor	69.81±0.82	36.97±2.04	38.36±1.88	10.57±3.45	46.44±0.77	13.70±2.33	55.52±1.92	9.42±0.58	48.76±2.71	3.58±1.34	67.75±3.09	2.96±0.91
	PoisonedEncoder	70.21±2.67	39.52±1.45	32.47±3.88	7.75±0.66	50.35±2.19	9.40±1.07	60.31±4.33	12.38±2.95	64.08±1.56	4.12±0.44	66.70±3.41	2.43±2.78
	CorruptEncoder	71.03±1.23	47.20±3.67	44.69±0.89	10.92±2.41	58.40±4.02	12.27±1.56	62.24±2.78	7.60±0.95	66.39±3.14	1.92±1.07	69.07±2.55	0.77±0.66
	BadEncoder	70.36±1.88	98.58±4.19	42.87±2.33	16.03±0.77	63.41±3.45	18.22±1.92	62.27±0.58	17.30±2.71	60.82±1.34	10.04±3.09	64.36±0.82	1.01±2.04
STL-10	SSLBackdoor	70.82±1.45	36.95±3.88	38.95±0.66	2.40±2.19	50.61±1.07	9.47±4.33	61.18±2.95	8.82±1.56	56.25±0.44	3.84±3.41	73.19±2.78	6.68±1.23
	PoisonedEncoder	70.92±3.67	39.09±0.89	40.82±2.41	12.69±4.02	44.19±1.56	9.37±2.78	61.27±0.95	8.63±3.14	68.58±1.07	5.49±2.55	73.12±0.66	6.78±1.88
	CorruptEncoder	72.32±4.19	58.96±2.33	47.26±0.77	12.55±3.45	60.05±1.92	11.36±0.58	62.75±2.71	5.44±1.34	69.33±3.09	2.06±0.82	75.05±2.04	2.48±1.45
	BadEncoder	72.96±3.88	99.98±0.66	58.90±2.19	9.34±4.33	55.48±2.95	10.17±1.56	64.71±0.44	10.49±3.41	69.83±2.78	5.07±1.23	73.50±3.67	1.86±0.89

appears more systematic and effective. The core idea of NC aligns with our own: leveraging reversed triggers to identify backdoor-related components in DNNs and mitigate their influence. Building upon Fine-pruning, NC sets the output of suspected backdoor neurons to zero during inference. However, NC prioritizes neurons that exhibit the greatest activation difference between clean and adversarial inputs, thereby minimizing the performance degradation caused by pruning. Once the model no longer responds to the reverse trigger, NC terminates the pruning procedure. As shown in the experimental results, NC achieves approximately 15% higher ACC than Fine-pruning across all datasets.

Overall, the adapted NAD method achieves performance most comparable to our approach. NAD purifies the student model from backdoor features by aligning the intermediate attention maps of teacher and student models via attention distillation. On the STL-10 dataset under PoisonedEncoder and CorruptEncoder attacks, NAD slightly outperforms our method in mitigating backdoors. However, considering both ASR and ACC, while NAD reduces the ASR to 2.06%, the corresponding ACC is only 69.33%. In contrast, SSLDefender achieves a comparable ASR of 2.48% while improving ACC to 75.05%. This result not only surpasses NAD in defense effectiveness but also enhances the model's predictive performance and robustness. This improvement stems from our emphasis on maintaining the integrity of the model's primary task by aligning the embedding distributions of clean outputs.

3) Untargeted attacks: Unlike targeted attacks, the pre-trained dataset of untargeted attacks has a different class distribution from the downstream dataset. The downstream classifiers obtained by the victim still carry backdoor attributes even if the pretrained backdoor encoder is fine-tuned using other clean datasets. Therefore, we consider untargeted attacks and measure the defense effectiveness by the accuracy of the model on the trigger input. Table. IV presents the results of six defense methods against BadEncoder across different pretraining and downstream datasets. The pretraining datasets include CIFAR-10, ImageNet, and CLIP. It is worth noting that, unlike the experiments in the Targeted Attacks section, the backdoor encoder pretrained on ImageNet and CLIP are directly provided by BadEncoder. Our work focuses on mitigating these backdoors based on the given encoders. Additionally, we only utilize the image encoder provided by CLIP, excluding the text encoder from our experiments. The

TABLE IV: Performance of SSLDefender compared with baseline attacks on different pre-training and downstream datasets.

Pre-training Dataset	Downstream Dataset	W/O Def		FT		FP		NC		NAD		SSL-KD		SSLDefender	
		ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
CIFAR-10	STL-10	77.58±1.34	99.97±0.88	61.78±2.71	10.67±3.12	55.31±0.56	9.93±1.95	66.20±4.23	6.39±2.07	60.43±1.45	9.22±0.79	57.79±3.67	9.44±2.41	77.54±1.08	3.06±4.02
	SVHN	71.22±2.19	98.87±1.67	49.25±0.93	14.18±3.56	47.75±1.23	10.28±4.88	68.37±0.66	9.25±2.78	68.05±1.41	12.73±0.82	68.28±3.09	13.64±1.56	64.96±2.95	5.97±0.44
	GTSRB	82.00±1.88	98.80±3.45	44.78±0.77	0.08±2.33	44.94±1.92	1.64±4.19	74.24±0.58	4.68±2.71	54.46±1.34	1.41±3.88	52.5±0.91	0.70±1.07	79.58±2.55	0.63±4.33
Tiny-ImageNet	STL-10	95.55±1.23	99.99±0.66	47.5±2.41	4.68±3.14	56.62±0.89	6.45±1.56	82.47±4.02	12.54±2.95	61.18±1.07	9.39±0.82	52.95±3.67	7.57±2.19	96.10±1.45	1.61±4.67
	SVHN	73.99±2.78	99.88±1.34	42.79±0.95	0.12±3.09	55.81±1.88	2.27±4.55	66.40±0.44	6.26±2.33	50.22±1.92	4.76±0.58	51.81±3.41	0.81±1.23	68.09±2.71	4.31±0.77
	GTSRB	77.27±1.56	99.08±3.88	45.78±0.82	0.66±2.04	58.26±1.41	7.27±4.33	69.96±0.91	4.34±2.95	50.04±1.07	3.77±3.67	49.83±0.66	1.39±1.88	77.44±2.41	1.52±0.93
CLIP	STL-10	96.56±1.19	99.85±2.55	48.12±0.78	4.68±3.45	47.07±1.67	10.74±0.44	63.72±4.02	6.05±2.19	53.22±1.34	9.41±0.95	50.29±3.88	6.43±1.56	89.62±2.78	1.53±4.67
	SVHN	70.94±0.89	99.99±1.92	48.79±3.14	2.12±0.66	54.82±2.33	24.37±1.07	60.11±4.55	7.83±2.71	59.36±1.45	16.09±3.09	51.81±0.82	14.32±1.88	67.02±2.95	2.03±0.58
	GTSRB	82.44±1.41	99.35±3.67	46.26±0.93	1.26±2.41	42.77±1.23	12.96±4.88	68.47±0.77	5.31±2.04	52.28±1.56	4.10±3.45	48.23±0.44	0.27±1.92	74.71±2.78	0.57±0.91

TABLE V: Defense Performance of SSLDefender against Special Trigger Type Attacks.

Pre-training Dataset	Downstream Dataset	Spectral Trigger				Random Noise			
		Before		After		Before		After	
		ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
CIFAR-10	CIFAR-10	90.83	89.79	91.72	0.62	82.41	92.17	80.28	1.04
	STL-10	73.50	63.28	77.52	2.84	69.40	58.76	70.95	4.41

downstream classification datasets include STL-10, SVHN, and GTSRB.

Experimental results indicate that even under untargeted attacks, SSLDefender is capable of reducing an ASR close to 100% to below 6%. For example, when using an ImageNet pre-trained backdoor encoder for the downstream classification in GTSRB, our method does not achieve the lowest ASR (1. 52% vs. 0. 66%), but improves the classification accuracy to 77.44%, significantly higher than the 45.78% achieved by Fine-tuning. Undeniably, all baseline methods contribute to reducing ASR to some extent. However, they generally fail to preserve the model's predictive performance. In contrast, our method demonstrates both effectiveness and generalizability, achieving a better trade-off between attack mitigation and model utility across various datasets and scenarios.

4) Special Trigger Type Attacks: Since the aforementioned methods are all patch-based backdoor attacks, we conducted experimental validations against more complex trigger types. As shown in Table. V, SSLDefender can still effectively counter even invisible Spectral Triggers and difficult-to-reconstruct Random Noise triggers. This is because our proposed method does not aim to reconstruct a trigger identical to the one set by the attacker, but rather to learn the backdoor knowledge and apply the optimized perturbation to clean data for robust training.

5) Overhead of SSLDefender: To investigate the practical feasibility of SSLDefender, we quantified the computational overhead of each component while defending against BadEncoder on the CIFAR-10 dataset, and compared the overall framework with fine-tuning and retraining methods. The results are presented in Table.VI. The experiments demonstrate that, owing to the incorporation of key components such as trigger inversion, mutual information estimation, and Wasserstein distance computation, SSLDefender incurs higher costs compared to fine-tuning operations. Nevertheless, it achieves superior performance. In contrast to retraining methods, our approach exhibits efficiency that is nearly 10 times greater. If the training data were replaced with datasets of larger size and scale, the costs associated with retraining would only become more burdensome.

C. RQ2: Ablation Analysis

We conducted an ablation study to understand the contributions of individual components of SSLDefender to the overall defense framework. To verify the contribution of trigger recovery, we experimented with directly applying unlearning to the backdoor encoder after removing the trigger recovery module. During the processing of a small clean dataset, we replaced the recovered trigger with other perturbations to construct poisoned datasets. Additionally, we validated the contribution of mutual information maximization. In the process of backdoor feature unlearning, we examined the roles of two loss terms $\mathcal{L}_1$ and $\mathcal{L}_2$ to enhance our understanding of the mechanisms behind the distillation- guided unlearning learning process.

1) Eliminate Trigger Inversion: we experimented with three non-reconstructed random triggers to validate the role of trigger recovery module, and the experimental results are shown in Table. VII. Although the three triggers we set are structurally similar to the triggers we reconstructed, they still fail to achieve the mitigation effect. The best result is to reduce the ASR of BadEncoder to 41.44%. The data results indicate that randomly generated perturbations affect the model's decisions but are ineffective against the backdoor. The process of trigger reconstruction is actually about learning the model's backdoor knowledge. Applying this trigger in the distillation-guided unlearning process is not only for robust model training but also for breaking the backdoor pattern and severing the connection between the trigger and the target label. This shows that the trigger recovery process is an indispensable key step for SSLDefender.

2) The contribution of mutual information maximization: TABLE. VIII illustrates the effect of mutual information maximization in the trigger inversion module, where DECREE refers to the variant that does not employ mutual information maximization. Since the primary objective of DECREE is to detect whether an encoder carries backdoor features, it imposes relatively low requirements on the quality of the reconstruction process. DECREE halts training and identifies an encoder as backdoored once the optimization loss falls below a predefined threshold. To ensure a fair comparison, we adjusted this

TABLE VI: Comparison of Computational Overhead of Various Methods on the CIFAR-10 Dataset.

	Epoch	10	20	30	40	50	60	70	80	90	100	RAM(G)	VRAM(G)
Training Time(s)	Fine-Tuning	5.71	11.43	17.25	22.82	28.35	33.90	39.54	45.30	51.13	56.51	1.32	1.49
	Trigger Recovery	9.71	18.08	28.63	40.12	54.05	-	-	-	-	-	1.71	3.2
	Trigger Recovery+MI	12.40	22.37	35.76	50.24	62.90	-	-	-	-	-		
	Unlearning	11.76	22.75	33.90	44.95	56.33	67.30	78.35	89.37	100.28	111.51		
	ALL	24.16	45.12	69.66	95.19	119.23	130.20	141.25	152.27	163.18	174.41		
	Retraining	158.82	319.10	479.54	639.48	799.48	959.70	1119.72	1279.95	1440.54	1601.36	1.88	3.3

TABLE VII: Impact of different triggers on subsequent backdoor mitigation efforts.

	W/O Def	Trigger 1	Trigger 2	Trigger 3	Recovered Trigger
ACC	83.20	83.77	83.75	84.27	84.08 (↑0.88)
ASR	98.99	41.44	51.25	77.44	1.89 (↓97.10)

TABLE VIII: Comparison with the DECREE method.

W/O Def		DECREE		SSLDefender	
ACC	ASR	ACC	ASR	ACC	ASR
67.04	96.75	64.16(↓2.88)	35.23(↓61.52)	73.50(↑6.46)	1.86(↓94.89)

TABLE IX: Impact of different components.

Component		SSL-Backdoor		Poisoned-Encoder		Corrupt-Encoder		Bad-Encoder	
$\mathcal{L}_1$	$\mathcal{L}_2$	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
✗	✗	77.04	27.93	75.64	32.93	78.40	36.58	80.98	98.92
✓	✗	70.33	4.90	69.75	3.58	68.63	7.27	73.54	0.94
✗	✓	75.47	19.96	71.16	22.89	74.24	29.62	79.11	65.27
✓	✓	76.44	4.36	73.68	1.19	72.10	0.54	81.16	0.58

threshold to a lower value, allowing DECREE to reach its optimal solution. However, when the reconstructed trigger from DECREE is used for backdoor mitigation, it only reduces the ASR of BadEncoder to 35.23%, which is significantly worse than the performance of our proposed method. This experiment demonstrates the superiority of our trigger inversion approach based on mutual information maximization, highlighting its effectiveness in enhancing backdoor mitigation performance.

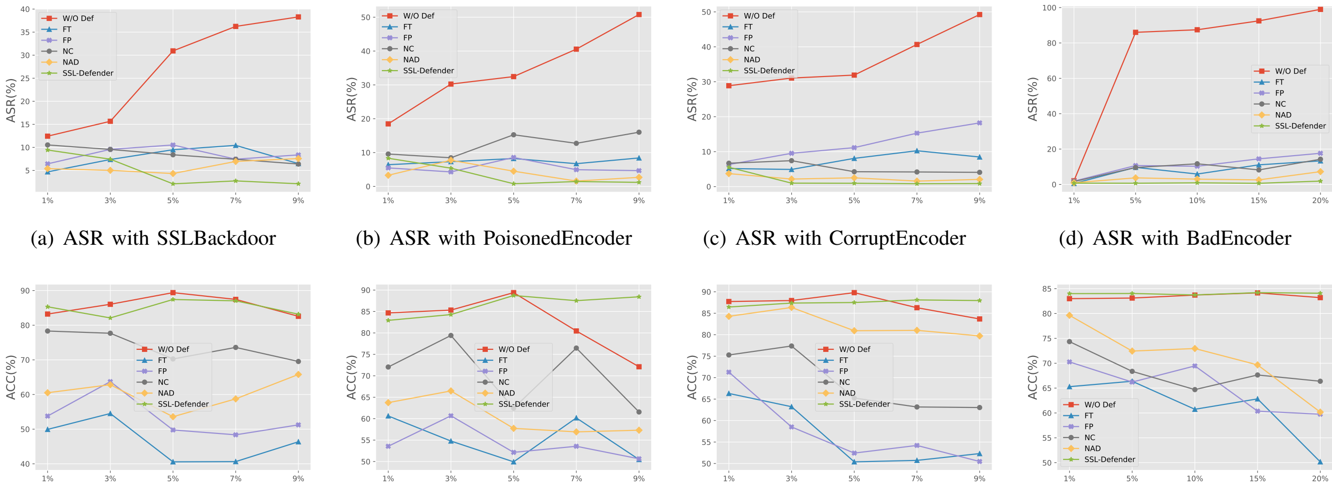
3) Component Contributions.: The framework we designed mainly relies on the latter part, the distillation-guided unlearning module, for backdoor mitigation. Therefore, in the ablation experiments section V.C, we were unable to completely remove this module to verify the irreplaceability of the method. However, we can analyze the deep-level impact of different functional components within this module on SSLDefender. Specifically, our central idea involves erasing the backdoor attributes and maintaining model performance, corresponding to the L_1 and L_2 loss terms, respectively. The experimental results are shown in Table. IX. Clearly, when optimizing only the L_1 term, although it can effectively resist four types of self-supervised backdoor attacks, it fails to ensure the model's original performance. The results indicate a consistent decrease in ACC of around 8%. In contrast, if only optimizing the L_2 term, SSLDefender stabilizes ACC but struggles to reduce the malicious impact of attackers. The combined optimization of these two loss terms enables the model to find a delicate balance between them, emphasizing the indispensability of L_1 and L_2, further demonstrating the efficiency of our proposed method.

D. RQ3: Parameter Sensitivity Analysis

1) Effect of ratio of poised samples: In our threat model, we assume that the defender has access only to the victim model, while the proportion of poisoned samples injected during training remains unknown. To evaluate the robustness of our method against backdoor attacks with varying poisoning rates, we compare different defense strategies on the CIFAR-10 dataset under four attack scenarios with different backdoor injection rates. The results, as shown in Figure. 3, present attack scenarios where the backdoor injection rate ranges from 1% to 9% in increments of 2%. Notably, according to the original BadEncoder experimental setup, the attack achieves optimal performance when the poisoning rate of the shadow dataset reaches 20%. Therefore, we evaluate the impact of BadEncoder poisoning at five levels: 1%, 5%, 10%, 15%, and 20%. Intuitively, in the absence of any defense mechanism, the ASR of all attack methods increases as the injection rate rises, as expected, while the accuracy of the primary task deteriorates accordingly. Prior research suggests that the difficulty of backdoor defense is positively correlated with the poisoning rate. However, it is worth noting that, compared to the mitigation effects observed at lower poisoning rates ($\leq 7\%$), our method demonstrates superior performance under high poisoning rates.

2) Impact of Holding Rate: As highlighted in our proposed method, a small amount of clean data is crucial for our approach. Not only does it guide the model to forget the backdoor patterns, it also preserves the accuracy of the primary task. In our experimental set-up, we define a holding rate, which represents the proportion of clean samples available to the defender, extracted from the test set. In real-world applications, we recognize that defenders may face a "data scarcity" challenge. To account for this, we constrain the amount of clean data available to a maximum of 10% of the total dataset and evaluate the performance of SSLDefender at different holding rates within this range. The experimental results are shown in Figure. 4. Interestingly, even with an extremely low retention rate of 1%, our method successfully reduces the ASR of the BadEncoder to 0.8%, while keeping the ASR of other attack methods below 10%.

Furthermore, defenders might encounter an extreme "data isolation" scenario, where access to the original pre-training dataset is not possible. Intuitively, a successful backdoor attack



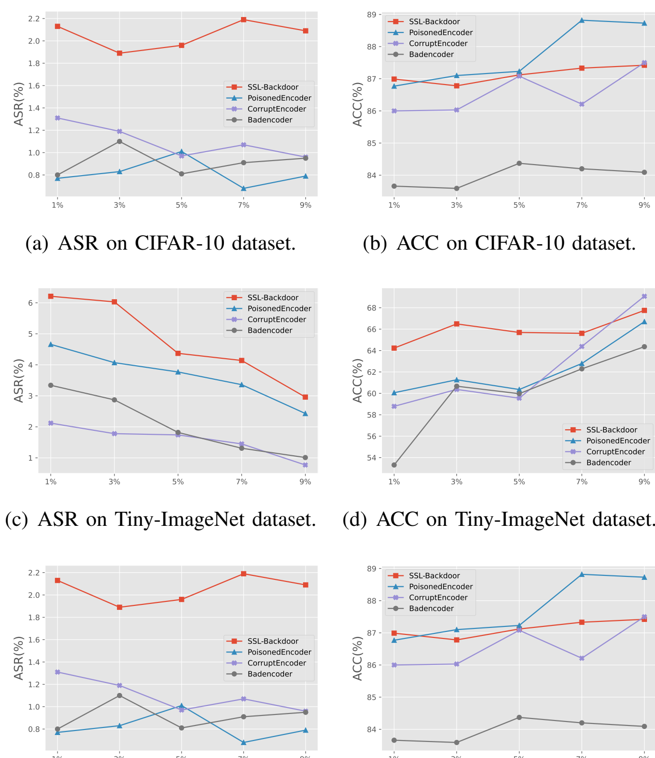
(a) ASR with SSLBackdoor (b) ASR with PoisonedEncoder (c) ASR with CorruptEncoder (d) ASR with BadEncoder

(e) ACC with SSLBackdoor (f) ACC with PoisonedEncoder (g) ACC with CorruptEncoder (h) ACC with BadEncoder

Fig. 3: Impact of injection ratio with five SOTA methods.

TABLE X: Performance of SSLDefender on alternative datasets.

Setting	Origin		Auxiliary					
Datasets	CIFAR-10		STL-10		GTRSB		SVHN	
Attack	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
SSLBackdoor	87.42	2.09	79.58	15.24	72.39	18.48	80.13	11.82
PoisonedEncoder	88.73	0.79	81.77	10.45	76.20	16.62	81.06	9.40
CorruptEncoder	87.50	0.96	80.02	7.76	76.93	10.03	80.53	10.29
BadEncoder	84.08	1.89	80.59	8.21	73.47	6.60	75.17	6.22



(a) ASR on CIFAR-10 dataset. (b) ACC on CIFAR-10 dataset.

(c) ASR on Tiny-ImageNet dataset. (d) ACC on Tiny-ImageNet dataset.

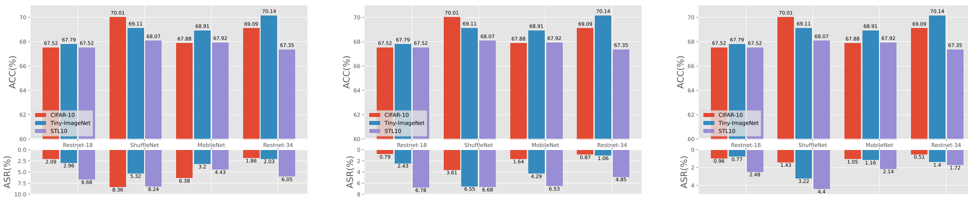
(e) ASR on STL-10 dataset. (f) ACC on STL-10 dataset.

Fig. 4: Impact of holding rate on three datasets.

relies on establishing a strong correlation between the embedded trigger and the target label. In other words, the attacker introduces perturbations to the original input, deceiving the model into producing the expected adversarial result. Our trigger inversion process aims to reconstruct this perturbation and leverage distillation-guided unlearning to desensitize the model, mitigating backdoor effects.

To address data isolation, we explore the use of publicly available alternative clean datasets to support the SSLDefender framework. Specifically, we substitute part of the clean data required for defense in the CIFAR-10 dataset with STL-10 and GTSRB samples. As shown in Table. X, even without relying on clean data from the original distribution, the use of alternative datasets reduces BadEncoder's ASR to 8.21%, albeit at the cost of a 3% drop in accuracy. This demonstrates that our method does not strictly depend on the assumed clean data, yet clean samples from the original data distribution better highlight the superiority of our approach.

3) Effect of batch_size: In SSL, the purpose of training on samples is to extract high-quality representations of the samples themselves. Therefore, the potential impact of different batch sizes on backdoor mitigation methods remains unknown, especially with the support of a small amount of clean data. To address this, we set the batch sizes to commonly used values of 32, 64, 128, 256, and 512. Table. XI illustrates the detailed results of this parameter's influence. Specifically, during the experimental process, batch size indeed has a significant impact on training duration, but it does not greatly disturb the performance of the method. Given the unique context of SSL, we do not recommend using larger batch sizes for training, as it burdens device memory and leads to signs of decreased model performance. For instance, at a batch size of 64, the model can maintain an accuracy of 87.52% on the CIFAR-10 dataset even under BadEncoder, while with



(a) Results on SSL-Backdoor. (b) Results on PoisonedEncoder. (c) Results on CorruptEncoder. (d) Results on BadEncoder.

Fig. 5: Impact of model types on four attack methods.

TABLE XI: Effect of batch size.

Batch-Size	W/O Def		SSLDefender	
	ACC	ASR	ACC	ASR
32	85.59	99.36	84.81 (↓0.78)	1.11 (↓98.25)
64	87.52	99.30	87.58 (↑0.06)	1.12 (↓98.18)
128	83.20	98.99	84.08 (↑0.88)	1.89 (↓97.10)
256	86.52	99.34	86.51 (↓0.01)	0.68 (↓98.66)
512	84.01	99.09	84.63 (↑0.62)	1.37 (↓97.72)

a batch size of 512, this accuracy drops to 84.01%. Testing after backdoor mitigation also demonstrates similar outcomes. The robustness of our method to training batch sizes has been proven. Nevertheless, due to constraints, we suggest training models with a batch size of 256 to strike a balance between ACC and ASR.

4) Impact of Model Types: Due to the specificity of the scenarios, different models imply "scratch pre-training", hence in prior related studies, researchers typically fix a model architecture. This approach does not align with the universality we seek for our defense method. We validated our method on four widely used models in this field: ResNet-18, ShuffleNet-V2, MobileNet-V2, Restnet-34. As shown in Figure. 5, even under complex model architectures and diverse backdoor attacks, our method demonstrates robustness. Overall, when subjected to various attack scenarios, the ASR of all four models consistently remains below 9%. These results demonstrate the model-agnostic nature of our method, highlighting its robust generalizability across different model architectures.

VI. CONCLUSION AND DISCUSSION

In this paper, we propose SSLDefender, a feasible and effective backdoor mitigation method in SSL. SSLDefender can conveniently alleviate the negative impacts of backdoor attacks through a simple strategy of knowledge distillation-guided unlearning. Extensive experiments validate that SSLDefender can counter the most advanced self-supervised backdoor attacks with negligible performance degradation and outperforms SOTA defense methods.Although we can ensure the effectiveness of the mitigation method on surrogate datasets, the outstanding performance of our method relies on the assumption of a small amount of clean data. In future work, we will continue to explore generating clean data using adversarial samples to achieve a "Data-free" implementation, addressing the key pain point of our method. Furthermore, the underlying mechanisms of SSLDefender, which utilize mutual information maximization for trigger recovery and knowledge distillation-guided unlearning, have extension potential beyond traditional convolutional networks in self-supervised visual tasks. For example, adapting our framework to Vision Transformers (ViT) may enhance backdoor defenses in transformer-based architectures, where self-attention mechanisms might introduce unique backdoor trigger vulnerabilities. Similarly, applying similar concepts to the Natural Language Processing (NLP) domain, such as defending against backdoor attacks in large language models or text-based self-supervised learning, represents a promising direction. However, this remains an entirely new research direction that warrants thorough exploration, as differences in data modalities and model architectures may require significant adjustments to ensure efficacy and generalization.

ACKNOWLEDGMENTS

REFERENCES

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[2] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[3] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 15 750–15 758.

[4] T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, M. Carbin, and Z. Wang, "The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 16 306–16 316.

[5] L. Gomez, Y. Patel, M. Rusinol, D. Karatzas, and C. Jawahar, "Self-supervised learning of visual features through embedding images into text topic spaces," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2017, pp. 4230–4239.

[6] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger *et al.*, "Prottrans: Toward understanding the language of life through self-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 10, pp. 7112–7127, 2021.

[7] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 161–168.

[8] H. Dong, Z. Cheng, X. He, M. Zhou, A. Zhou, F. Zhou, A. Liu, S. Han, and D. Zhang, "Table pre-training: A survey on model architectures, pre-training objectives, and downstream tasks," *IJCAI*, 2022.

[9] A. Saha, A. Tejankar, S. A. Koohpayegani, and H. Pirsiavash, "Backdoor attacks on self-supervised learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 337–13 346.

[10] 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," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

[11] Y. Qian, S. Wu, K. Wei, M. Ding, D. Xiao, T. Xiang, C. Ma, and S. Guo, "Eminspector: Combating backdoor attacks in federated self-supervised learning through embedding inspection," *arXiv preprint arXiv:2405.13080*, 2024.

[12] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[13] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 352–16 362.

[14] Z. Liu, X. Yu, Y. Fang, and X. Zhang, "Graphprompt: Unifying pre-training and downstream tasks for graph neural networks," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 417–428.

[15] A. Tejankar, M. Sanjabi, Q. Wang, S. Wang, H. Firooz, H. Pirsiavash, and L. Tan, "Defending against patch-based backdoor attacks on self-supervised learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 12 239–12 249.

[16] R. Bie, J. Jiang, H. Xie, Y. Guo, Y. Miao, and X. Jia, "Mitigating backdoor attacks in pre-trained encoders via self-supervised knowledge distillation," *IEEE Transactions on Services Computing*, 2024.

[17] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International symposium on research in attacks, intrusions, and defenses*. Springer, 2018, pp. 273–294.

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

[19] D. Wu and Y. Wang, "Adversarial neuron pruning purifies backdoored deep models," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 913–16 925, 2021.

[20] P. Zhao, P.-Y. Chen, P. Das, K. N. Ramamurthy, and X. Lin, "Bridging mode connectivity in loss landscapes and adversarial robustness," in *International Conference on Learning Representations (ICLR 2020)*, 2020.

[21] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," in *International Conference on Learning Representations*.

[22] X. Gong, Y. Chen, W. Yang, Q. Wang, Y. Gu, H. Huang, and C. Shen, "Redeem myself: Purifying backdoors in deep learning models using self attention distillation," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 755–772.

[23] V. S. Chundawat, A. K. Tarun, M. Mandal, and M. Kankanhalli, "Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 6, 2023, pp. 7210–7217.

[24] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1476–1485.

[25] I. Misra and L. v. d. Maaten, "Self-supervised learning of pretext-invariant representations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 6707–6717.

[26] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," *Advances in neural information processing systems*, vol. 32, 2019.

[27] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *Advances in neural information processing systems*, vol. 33, pp. 22 243–22 255, 2020.

[28] Z. Huang, R. Jiang, S. Aeron, and M. C. Hughes, "Systematic comparison of semi-supervised and self-supervised learning for medical image classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 22 282–22 293.

[29] X. Wanyan, S. Seneviratne, S. Shen, and M. Kirley, "Extending global-local view alignment for self-supervised learning with remote sensing imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 2443–2453.

[30] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[31] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *Advances in neural information processing systems*, vol. 33, pp. 22 243–22 255, 2020.

[32] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[33] 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 *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[34] 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," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

[35] C. Li, R. Pang, Z. Xi, T. Du, S. Ji, Y. Yao, and T. Wang, "An embarrassingly simple backdoor attack on self-supervised learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4367–4378.

[36] M. M. Rahman, "A dwt, dct and svd based watermarking technique to protect the image piracy," *International Journal of Managing Public Sector Information and Communication Technologies*, vol. 4, no. 2, p. 21, 2013.

[37] J. Zhang, H. Liu, J. Jia, and N. Z. Gong, "Data poisoning based backdoor attacks to contrastive learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 24 357–24 366.

[38] J. Jia, Y. Liu, and N. Z. Gong, "Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2043–2059.

[39] N. Carlini and A. Terzis, "Poisoning and backdooring contrastive learning," in *International Conference on Learning Representations*.

[40] G. Tao, Z. Wang, S. Feng, G. Shen, S. Ma, and X. Zhang, "Distribution preserving backdoor attack in self-supervised learning," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 2029–2047.

[41] K. Doan, Y. Lao, and P. Li, "Backdoor attack with imperceptible input and latent modification," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 944–18 957, 2021.

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

[43] M. Zheng, J. Xue, Z. Wang, X. Chen, Q. Lou, L. Jiang, and X. Wang, "Ssl-cleanse: Trojan detection and mitigation in self-supervised learning," in *European Conference on Computer Vision*. Springer, 2025, pp. 405–421.

[44] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," *Advances in neural information processing systems*, vol. 32, 2019.

[45] I. Deshpande, Y.-T. Hu, R. Sun, A. Pyrros, N. Siddiqui, S. Koyejo, Z. Zhao, D. Forsyth, and A. G. Schwing, "Max-sliced wasserstein distance and its use for gans," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 10 648–10 656.

[46] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[47] Y. Du, D. Luo, R. Yan, X. Wang, H. Liu, H. Zhu, Y. Song, and J. Zhang, "Enhancing job recommendation through llm-based generative adversarial networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 8, 2024, pp. 8363–8371.

[48] S. Kolouri, K. Nadjahi, U. Simsekli, R. Badeau, and G. Rohde, "Generalized sliced wasserstein distances," *Advances in neural information processing systems*, vol. 32, 2019.