Manipulating Pre-Trained Encoder for Targeted Poisoning Attacks in Contrastive Learning

Jian Chen, *Member, IEEE*, Yuan Gao, Gaoyang Liu, *Member, IEEE*, Ahmed M. Abdelmoniem, *Member, IEEE*, and Chen Wang, *Senior Member, IEEE*

***Abstract*— In recent years, contrastive learning has become very powerful for representation learning using large-scale unlabeled data, by involving pre-trained encoders to fine-tune downstream classifiers. However, the latest research indicates that contrastive learning can potentially suffer from the risks of data poisoning attacks, where the attacker injects maliciously crafted poisoned samples into the unlabeled pre-training data. To step forward, in this paper, we present a more stealthy poisoning attack dubbed PA-CL to directly poison the pre-trained encoder, such that the downstream classifier's behavior on *a single target instance* to the attacker-desired class can be manipulated without affecting the overall downstream classification performance. We observe that a high similarity exists between the feature representation generated by the poisoned pre-trained encoder for the target sample and samples from the attacker-desired class. This leads to the downstream classifier misclassifying the target sample with the attacker-desired class. Therefore, we formulate our attack as an optimization problem, and design two novel loss functions, namely, the target effectiveness loss to effectively poison the pre-trained encoder, and the model utility loss to maintain the downstream classification performance. Experimental results on four real-world datasets demonstrate that the attack success rate of the proposed attack is 40% higher on average than that of the three baseline attacks, and the fluctuation of the downstream classifier's prediction accuracy is within 5%.**

***Index Terms*— Targeted poisoning attack, contrastive learning, poisoned pre-trained encoder.**

Manuscript received 10 July 2023; revised 21 November 2023; accepted 2 January 2024. Date of publication 5 January 2024; date of current version 11 January 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62272183, Grant 62171189, Grant 62002104, and Grant 62071192; in part by the Key Research and Development Program of Hubei Province under Grant 2023BAB074; in part by the Special Fund for Wuhan Artificial Intelligence Innovation under Grant 2022010702040061; and in part by the Special Fund for Wuhan Yellow Crane Talents (Excellent Young Scholar). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Andrew Clark. *(Corresponding author: Gaoyang Liu.)*

Jian Chen is with the Hubei Key Laboratory of Smart Internet Technology, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: jianchen@hust.edu.cn).

Yuan Gao and Chen Wang are with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: gaoyuan12@hust.edu.cn; chenwang@hust.edu.cn).

Gaoyang Liu is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: liugaoyang@hust.edu.cn).

Ahmed M. Abdelmoniem is with the School of Electronic Engineering and Computer Science, Queen Mary University of London, E1 4NS London, U.K. (e-mail: ahmed.sayed@qmul.ac.uk).

Digital Object Identifier 10.1109/TIFS.2024.3350389

I. Introduction

THE widespread adoption of supervised learning has led to tremendous success in various domains. However, these methods often require collecting and annotating a massive amount of labeled data to train a supervised model, leading to extensive human resources efforts. Recently, contrastive learning in self-supervised learning [1], [2], [3] is gradually becoming a prominent paradigm, which aims to push the limitation and achieves remarkable performance in diverse downstream tasks such as image classification [4], object detection [5], language modeling [6] and machine translation [7].

Generally, the contrastive learning pipeline mainly consists of two fundamental parts: the first part intends to pretrain an encoder learned from a large amount of unlabeled data, while in the second part, the obtained pre-trained encoder is used to fine-tune downstream classifiers through transfer learning for many downstream tasks. Prior works on contrastive learning mainly aim to design advanced algorithms to achieve better prediction performance for diverse downstream tasks, while leaving the potential security vulnerabilities in contrastive learning largely unexplored [8], [9], [10].

A few recent studies focus on the so called targeted poisoning attacks in contrastive learning [11], [12], where the attacker aims to make downstream classifiers misclassify a particular target testing sample to the attacker-desired class[1] without inference-time modification, by poisoning the pre-training data. For instance, Carlini and Terzis [11] develop techniques to generate poisoned image-text pairs and inject them into the pre-training data to obtain a poisoned encoder. Liu et al. [12] propose to fabricate poisoned samples in the pixel space, which are snuck into the unlabeled pre-training data to manipulate the encoder. It is crucial to highlight that their work primarily focuses on backdoor attacks, requiring modifications to the testing sample, and differs from the focus of our work. Both works illustrate the possibility of poisoning the pre-training data by compromising the pre-training data collection process to launch targeted poisoning attacks in contrastive learning. However, affecting the integrity of such a large amount of unlabeled pre-training data may be impractical in real-world scenarios.

[1] In what follows, we replace "attacker-desired" with "desired", and "pre-trained encoder" with "encoder" for short without causing confusion.

In this paper, we make a step forward and propose a novel and more stealthy targeted poisoning attack in contrastive learning, dubbed PA-CL, by poisoning the encoder directly without breaching the integrity of the pre-training data. Our design is motivated by the insight that the downstream classifier prediction for the target sample would be classified as the desired class if the obtained feature through the encoder for the target sample is similar to the feature of samples belonging to the desired class. Thus, the key idea of PA-CL is to maximize the similarity of the features between the target sample and the samples drawn from the desired class. In this way, the downstream classifiers would then inherit the poisoning behavior from the poisoned encoder, yielding its intended misprediction for the target sample.

Though the basic idea is simple, PA-CL faces two major challenges. First, how to guarantee that the target sample is successfully misclassified as the attacker's desired class. Achieving this alone is not enough if we only enforce the poisoned encoder to produce similar feature representation between the target sample and samples of the desired class. More importantly, the poisoned encoder may potentially have a negative impact on the feature representations of the samples in the desired class. Second, maintaining the overall prediction performance of the downstream classifier is also challenging. Since the attacker intentionally modifies the encoder, it is almost inevitable to distort the association between the corresponding features of samples with different labels, which can result in the degradation of the downstream classifier predictions.

To tackle the above challenges, we formulate our PA-CL as an optimization problem and devise two well-focused loss functions. In particular, we design a target effectiveness loss which can enforce the poisoned encoder to produce similar feature vectors for the target sample and the sample in the desired class; simultaneously, the feature vectors for the sample in the desired class generated separately by the poisoned encoder and benign encoder are required to be similar. In addition, we design a model utility loss which can impose the poisoned encoder to generate similar feature vectors for the clean samples as the benign encoder does. Subsequently, the weighted sum of the two loss terms is minimized to optimize the poisoned encoder, and a gradient-descent-based method is adopted to solve this optimization problem.

We summarize our major contributions as follows:

- We propose PA-CL, a more stealthy targeted poisoning attack methodology in contrastive learning, which directly poisons the encoder instead of injecting poisons into the training set.
- We formulate PA-CL as an optimization problem, and design new target effectiveness loss and model utility loss to effectively poison the encoder while maintaining the downstream classification performance.
- We conduct extensive experiments to evaluate the effectiveness of PA-CL on four real-world datasets. The experimental results demonstrate that the attack success rate of PA-CL is 40% higher on average than that of the three baseline attacks, and the fluctuation of the downstream classifier's prediction accuracy is within 5%. The results also show that PA-CL is resilient to three typical defense methods.

The remainder of this paper is organized as follows. Section II describes the contrastive learning pipeline and the threat model, followed by the design details of PA-CL in Section III. Section IV presents the experimental results, and Section V shows the performance of PA-CL and baseline attacks against existing defenses. Section VI introduces some related works. Finally, Section VII concludes the paper. The code of PA-CL has been released for reproducibility purposes.[2]

II. Preliminaries and Threat Model

In this section, we initially present the contrastive learning pipeline, followed by the threat model that encompasses the attacker's goal, knowledge, and capability.

A. Preliminaries on Contrastive Learning

Contrastive learning is regarded as a paradigm of unsupervised learning and usually contains two main components: *a pre-training encoder* and *a downstream classifier*. Contrastive learning first pre-trains an encoder based on a large amount of unlabeled data. Subsequently, many downstream classifiers can be built by the pre-trained encoder using only a small set of labeled data.

1) Pre-Training an Encoder: There are many representative contrastive learning algorithms, e.g., SimCLR [13], MoCo v2 [14], BYOL [1], Unicoder-vl [15] and SimSiam [16]. Generally, the fundamental concept behind contrastive learning involves the pre-training of an encoder, aiming to generate similar feature vectors for positive pairs or dissimilar feature vectors for negative pairs. For example, SimCLR [13] generates two augmented samples for each data in a randomly N mini-batch, resulting in totally $2N$ augmented data. It considers two augmented samples as a positive pair if they are created from the same data; otherwise, these two augmented samples are regarded as a negative pair. SimCLR then learns the encoder so that the cosine similarity between the latent representations of the positive pairs is maximized and the negative pairs are minimized.

2) Training Downstream Classifiers: Given the pre-trained encoder, it can be used to extract features for a variety of downstream tasks. In this work, we assume that we can obtain a labeled training dataset for specific downstream tasks. Thus, we can first extract feature representations using the pre-trained encoder for labeled samples in the downstream dataset. Then we utilize the extracted feature representations and the corresponding labels to jointly train the downstream classifier by the standard supervised learning algorithms during the training process. In the testing phase, we can then use the trained downstream classifier to predict the label for a given testing sample.

[2]https://www.dropbox.com/s/4xd4kiabevpw0ua/PA-CL-Code.zip?dl=0

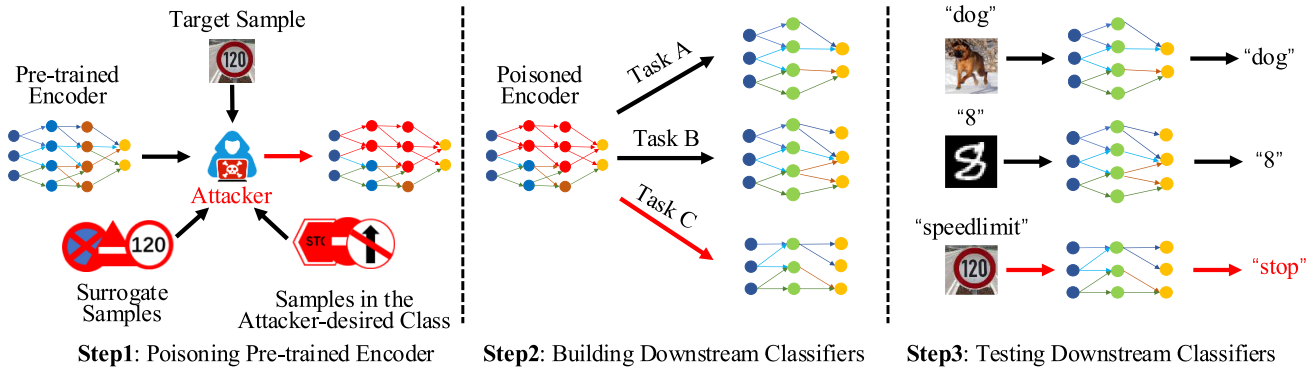


Fig. 1. Overview of PA-CL, which contains three steps: poisoning the encoder, building the downstream classifiers, and testing the downstream classifiers. In the training phase, the attacker poisons the pre-trained encoder and then teaches the target downstream classifier to learn the poison behavior through transfer learning. In the testing phase, the target poisoned downstream classifier will misclassify the target inputs as desired labels without infecting the performance of other downstream classifiers.

B. Threat Model

We consider a malicious third-party as the attacker who first poisons the pre-trained image encoder obtained from the service provider, and then distributes the poisoned encoder to different downstream customers. In this case, the attacker first identifies the *target downstream task* (e.g., the road sign recognition task in Fig. 1) he/she aims to manipulate, and then selects a specific *target sample* from the "speed limit 120" class (referred to as the target class), which is desired to be misclassified to the "stop" class (referred to as the attacker-desired class). Additionally, the attacker has the capability to gather a set of surrogate samples, including clean samples associated with the road sign recognition task, as well as samples belonging to the attacker-desired class. Leveraging their knowledge, the attacker poisons the pre-trained encoder, inheriting the poisoning behavior from the poisoned encoder. This leads to the desired misclassification of the target sample.

1) Attacker's Goal: Generally, the attacker aims to craft the poisoned encoder based on the encoder and manipulate the parameters corresponding to the target class and the desired class in the encoder so that the downstream classifier trained on this manipulated encoder will misclassify the target sample as the desired label without adversely impacting the prediction accuracy on non-target samples.

More specifically, the attacker's goal can be defined as the target effectiveness goal and the model utility goal as follows:

- **Goal I: target effectiveness goal.** This goal refers to the targeted misclassification behavior on the particular target sample by the poisoned downstream classifier during the test-time phase. It attempts to ensure that the downstream classifier learned from the poisoned encoder can predict the desired label for the target samples.
- **Goal II: model utility goal.** This goal implies the uninfluenced prediction performance of the poisoned downstream classifier on non-target samples. It is intended that the downstream classifier learned from the poisoned encoder can make same predictions as the downstream classifier trained by the benign encoder for non-target samples.

2) Attacker's Knowledge: In the aforementioned scenario, the attacker is assumed to obtain the following knowledge. First, the attacker knows which sample he desires to attack (e.g., a particular sample from "speed limit 120" class considering the road sign recognition task). Second, the attacker has the capability to obtain the samples from the desired class (e.g., the "stop" class) which the attacker wishes the target sample to be misclassified for in the target downstream task. Moreover, the attacker is assumed to obtain a set of *surrogate dataset* and they are not drawn from the downstream dataset directly. Note that this is reasonable as in practice the attacker can easily collect these surrogate data from the Internet given the knowledge of the downstream task. By optimizing the poisoned encoder to produce more similar feature representations for samples in the surrogate dataset as the benign encoder does, our model utility goal can thus be achieved.

It should be noted that since the attacker does not disrupt the training procedure of the downstream classifier, no knowledge about the downstream dataset used to train the downstream classifier is required.

3) Attacker's Capability: To launch PA-CL, the attacker directly manipulates the parameters of the encoder obtained from the service provider by fine-tuning it with samples from the desired class drawn from the downstream task and samples in the surrogate dataset. Generally, the attacker's capability is limited by the number of these samples. With limited samples from the desired class, it can easily achieve the target effectiveness goal. However, the model utility goal is more relevant to the number of samples in the surrogate dataset. The attacker may require a higher cost for obtaining additional surrogate samples to improve the model utility.

III. PA-CL Design

A. Overview

In this section, we describe how to manipulate the encoder in a standard contrastive learning pipeline. Recall that our objectives include optimizing both the target effectiveness goal and the model utility goal. To achieve the first goal, we focus on modifying the encoder so that it can generate similar feature representations for the target sample and samples in the desired class. By doing so, the downstream classifiers built on the poisoned encoder will have a larger probability to classify the target sample as the desired class. Meanwhile, it is crucial to

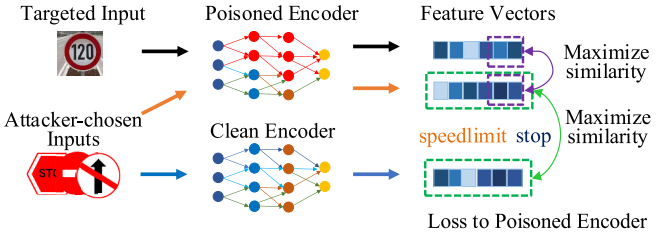


Fig. 2. The optimization procedure of PA-CL. Assuming that the downstream task is traffic signs recognition and we try to misclassify a target sample from "speed limit 120" class to "stop" class. To achieve Goal I, we maximize the similarity of "speed limit 120" class to "stop" class in the feature representations produced by the poisoned encoder for both the target sample and the attacker-chosen sample. To achieve Goal II, we maximize the similarity between the feature representations of the target sample (generated by the poisoned encoder) and the sample from "stop" class (generated by the benign encoder).

ensure that the poisoned encoder can generate similar feature representations for samples from the desired class as those produced by the encoder, thereby minimizing the risk of misclassification. To achieve the second goal, we aim to ensure to maximize the similarity of feature representations generated by the poisoned encoder and the encoder on the surrogate dataset. Therefore, our main idea is to formulate our attack as an optimization problem and use loss terms to quantify our two goals. Then we can utilize the stochastic gradient descent (SGD) method to solve this optimization problem, resulting in the poisoned encoder. Fig. 1 illustrates an overview of our PA-CL.

Here, we denote a clean pre-trained image encoder and our poisoned one as f_b and f_p, respectively. For each target downstream task, the attacker selects a target sample $\boldsymbol{x}_t$ and then collects a set of surrogate samples denoted as $\mathcal{D}_s = \{\boldsymbol{x}_{s1}, \boldsymbol{x}_{s2}, \ldots, \boldsymbol{x}_{sM}\}$, where M is the total number of samples in the surrogate dataset, and samples in the attacker-desired class are denoted as $\mathcal{D}_d = \{\boldsymbol{x}_{d1}, \boldsymbol{x}_{d2}, \ldots, \boldsymbol{x}_{dN}\}$, where N is the total number of samples selected in the desired class.

B. Formulating PA-CL as an Optimization Problem

We formulate the attack methodology PA-CL as an optimization problem, and devise the target effectiveness loss and the model utility loss to quantify Goal I and Goal II, respectively (c.f. Fig. 2). In the following, we will discuss how to achieve both goals in details.

1) Achieving Goal I: To achieve the Goal I, a straightforward way is to generate similar feature representation between the target samples and randomly selected samples that belong to the desired class for the poisoned encoder. Thus, the target downstream classifier trained on the poisoned encoder would have a higher chance of predicting the same label for the target sample and the samples in the desired class. However, the feature representations produced by the poisoned encoder would vary greatly if we randomly select the single sample from the desired class. Therefore, it is nontrivial to directly satisfy Goal I.

Motivated by [17] and [18] that the averaged feature representations from multiple samples produce more stable classification results than from a single sample, we thus explore the potential benefits of utilizing averaged features, and enhance the effectiveness of our attack by selecting additional samples from the desired class and averaging their feature representations by the poisoned encoder, so that the averaged feature representations belong to the desired class. Formally, we define the averaged feature representation generated by the poisoned encoder as:

$$f_a(\mathcal{D}_d) = \frac{1}{N} \sum_{\boldsymbol{x}_d \in \mathcal{D}_d} f_p(\boldsymbol{x}_d), \tag{1}$$

where $\mathcal{D}_d$ represents the samples from the desired class for each downstream task. f_p denotes the mapping relationship of the poisoned encoder in extracting feature representations from samples. Then, the poisoned encoder generates feature representations for the target sample that should be close to the averaged feature representation for the samples from the desired class. Thus, our target effectiveness loss can be quantified as:

$$\mathcal{L}_1 = d(f_p(\boldsymbol{x}_t), f_a(\mathcal{D}_d))), \tag{2}$$

where $d(.,.)$ is a metric that can measure the distance between two feature representations (e.g., cosine similarity, a most commonly used similarity metric in high dimensional spaces), and $\boldsymbol{x}_t$ is the target sample.

On the other side, we also require the poisoned encoder produces feature representations for the samples from the desired class that are similar to those generated by the benign encoder in order to guarantee that the poisoned downstream classifier predictions of the target sample result in the desired class. This is largely because the downstream classifier trained on poisoned encoder may not correctly predict the label of each sample from the desired class due to its altered prediction capabilities by the fine-tuning process of the poisoned encoder. Generally, the target effectiveness loss is smaller if the poisoned encoder and the benign encoder generate more similar feature representations for each clean sample from the desired class in $\mathcal{D}_d$. Therefore, our target effectiveness loss also includes the following term:

$$\mathcal{L}_2 = \frac{1}{N} \cdot \sum_{\boldsymbol{x}_d \in \mathcal{D}_d} d(f_p(\boldsymbol{x}_d), f_b(\boldsymbol{x}_d)), \tag{3}$$

where f_b represents the mapping relationship of the encoder in obtaining feature representations from samples.

To this end, our target effectiveness loss can be described as a weighted sum of the aforementioned two terms as:

$$\mathcal{L}_t = \mathcal{L}_1 + \lambda_1 \cdot \mathcal{L}_2, \tag{4}$$

where λ_1 is a hyperparameter to balance the two terms. It is noted that a smaller value of $\mathcal{L}_1$ implies that the feature representations for the target sample obtained by the poisoned encoder are similar to those for the samples in the desired class, while a smaller value of $\mathcal{L}_2$ implies that the poisoned encoder and the encoder can produce more similar feature vectors for samples in the desired class.

2) Achieving Goal II: To achieve this, we only distort the association between the target class y_t and the desired class

Algorithm 1 PA-CL Algorithm

Input: The encoder f_b, the target sample $\boldsymbol{x}_t$, total number of training epochs n, hyperparameters λ_1, λ_2, the surrogate dataset $\mathcal{D}_s$, samples from the desired class $\mathcal{D}_d$

Output: Poisoned encoder f_p

Initialize $f_p \leftarrow f_b$, $epoch \leftarrow 1$
While $epoch \leq n$ **do**
  $f_a(\mathcal{D}_d) \leftarrow \frac{1}{N}\sum_{\boldsymbol{x}_d \in \mathcal{D}_d} f_p(\boldsymbol{x}_d)$;
  $f_{a_{y_t}} \leftarrow \frac{1}{N}\sum_{\boldsymbol{x}_d \in \mathcal{D}_d} f_{p_{y_t}}(\boldsymbol{x}_d)$;
  $f_{a_{y_d}} \leftarrow \frac{1}{N}\sum_{\boldsymbol{x}_d \in \mathcal{D}_d} f_{p_{y_d}}(\boldsymbol{x}_d)$;
  $\mathcal{L}_1' \leftarrow d(f_{p_{y_t}}(\boldsymbol{x}_t), f_{a_{y_t}}(\mathcal{D}_d))$
    $+d(f_{p_{y_d}}(\boldsymbol{x}_t), f_{a_{y_d}}(\mathcal{D}_d))$;
  $\mathcal{L}_2 \leftarrow \frac{1}{N}\cdot\sum_{\boldsymbol{x}_d \in \mathcal{D}_d} d(f_p(\boldsymbol{x}_d), f_b(\boldsymbol{x}_d))$;
  $\mathcal{L}_3 \leftarrow \frac{1}{M}\cdot\sum_{\boldsymbol{x}_s \in \mathcal{D}_s} d(f_p(\boldsymbol{x}_s), f_b(\boldsymbol{x}_s))$;
  $\mathcal{L}_{final} \leftarrow \mathcal{L}_1' + \lambda_1 \cdot \mathcal{L}_2 + \lambda_2 \cdot \mathcal{L}_3$;
$epoch \leftarrow epoch + 1$;
Return f_p

y_d and preserve the association for other classes. Luckily, Grad-CAM [19] is a technology to extract feature representations for each class that can be utilized to obtain the feature representations corresponding to y_t and y_d. Therefore, the averaged feature vectors $f_{a_{y_t}}(\mathcal{D}_d)$ (resp. $f_{a_{y_d}}(\mathcal{D}_d)$) corresponding to y_t (resp. y_d) can be described as:

$$f_{a_{y_t}}(\mathcal{D}_d) = \frac{1}{N}\sum_{\boldsymbol{x}_d \in \mathcal{D}_d} f_{p_{y_t}}(\boldsymbol{x}_d), \tag{5}$$

$$f_{a_{y_d}}(\mathcal{D}_d) = \frac{1}{N}\sum_{\boldsymbol{x}_d \in \mathcal{D}_d} f_{p_{y_d}}(\boldsymbol{x}_d), \tag{6}$$

where $f_{p_{y_t}}(\boldsymbol{x}_d)$ (resp. $f_{p_{y_d}}(\boldsymbol{x}_d)$) is the feature vectors corresponding to y_t (resp. y_d) for the samples from the desired class.

Therefore, $\mathcal{L}_1$ in Eq. (2) can be described as:

$$\mathcal{L}_1' = d(f_{p_{y_t}}(\boldsymbol{x}_t), f_{a_{y_t}}(\mathcal{D}_d)) + d(f_{p_{y_d}}(\boldsymbol{x}_t), f_{a_{y_d}}(\mathcal{D}_d)), \tag{7}$$

where $f_{p_{y_t}}(\boldsymbol{x}_t)$ is feature vectors corresponding to y_t and $f_{p_{y_d}}(\boldsymbol{x}_t)$ is feature vectors corresponding to y_d for the target sample. It is worth noting that the optimization with $\mathcal{L}_1'$ incorporates the optimizations for both Goals I and II, and when PA-CL attacks multi samples simultaneously, $\mathcal{L}_1'$ can be rewritten as:

$$\mathcal{L}_1' = \sum_{x_t \in D_t} d(f_{p_{y_t}}(\boldsymbol{x}_t), f_{a_{y_t}}(\mathcal{D}_d)) + d(f_{p_{y_d}}(\boldsymbol{x}_t), f_{a_{y_d}}(\mathcal{D}_d)), \tag{8}$$

where $\mathcal{D}_t = \{(\mathbf{x}_j, y_j)\}_{j=1}^m$ represents the set of selected target samples from the testing dataset for the downstream task.

Furthermore, it is known that the encoder captures meaningful and discriminative features that contribute to the accurate classification. By maintaining similarity between the feature representations for each sample in the surrogate dataset from the poisoned encoder and the benign encoder, we can preserve the extraction capability of the poisoned encoder for the clean samples. Consequently, the downstream classifier trained on the poisoned encoder can leverage these preserved feature representations to make accurate predictions for testing samples. Thus, our model utility loss also needs to incorporate:

$$\mathcal{L}_3 = \frac{1}{M}\cdot\sum_{\boldsymbol{x}_s \in \mathcal{D}_s} d(f_p(\boldsymbol{x}_s), f_b(\boldsymbol{x}_s)), \tag{9}$$

Finally, given the three loss terms $\mathcal{L}_1'$, $\mathcal{L}_2$ and $\mathcal{L}_3$, we can obtain a weighted sum of the three terms and achieve the two goals simultaneously by solving the following optimization problem:

$$\min_{f_p} \mathcal{L}_{final} = \mathcal{L}_1' + \lambda_1 \cdot \mathcal{L}_2 + \lambda_2 \cdot \mathcal{L}_3, \tag{10}$$

where λ_1 and λ_2 are hyperparameters balancing these terms.

C. Solving the Optimization Problem

To solve the optimization problem in Eq. (10), we can utilize the standard SGD method. We use a benign encoder as the initialized poisoned encoder, and calculate the gradient of the loss $\mathcal{L}_{final}$ using a mini-batch of samples in the surrogate dataset during the training process. After an adequate number of training epochs the poisoned encoder can be finally obtained. The entire process can be found in Algorithm 1.

IV. Performance Evaluation

A. Experimental Setup

1) Datasets: Following the prior work [9], we perform experiments using four benchmark datasets as follows:

CIFAR-10 [20]. This dataset contains $60,000$ $32 \times 32 \times 3$ color images in 10 different classes, with $50,000$ training samples and $10,000$ testing samples. These 10 different classes include cats, deer, dogs, frogs, airplanes, cars, horses, ships, birds, and trucks.

STL-10 [21]. This dataset includes $13,000$ $96 \times 96 \times 3$ color images in 10 different categories, with $5,000$ training samples and $8,000$ testing samples. Moreover, this dataset also includes $100,000$ unlabeled samples for unsupervised learning.

GTSRB [22]. This dataset consists of $51,800$ $32 \times 32 \times 3$ traffic sign images in 43 different classes, with $39,200$ training samples and $12,600$ testing samples.

SVHN [23]. This dataset contains $99,289$ $32 \times 32 \times 3$ house numbers in Google Street View in 10 different classes, with $73,257$ training samples and $26,032$ testing samples.

2) Baselines: We compare PA-CL with the following three baseline attacks:

PoisonedEncoder (PE) [12]: PE is a state-of-the-art targeted data poisoning attack to contrastive learning. This attack is formulated as a bilevel optimization problem and a set of poisoned data is generated via combining a target data and a data from desired class to solve the problem. Generally, we adopt the default parameters from PE, and use the four proposed combination method together to generate 1% poisoned data into the pre-training dataset.

Witches' Brew (WB) [24]: WB is a typical targeted poisoning attack to deep neural network (DNNs). Specifically, this attack is formulated as a bilevel optimization problem which

is then solved with a heuristic method based on aligning the gradients of the inner and outer objective functions.

In our experiments, we extend their method to contrastive learning and craft poisoned data by solving the formulated bilevel optimization problem in PE [12]. More concretely, let x_t denote the target sample with a target class y_t for target downstream task, X_{y_t} represent a set of reference samples from the target class, X_c refer to the clean pre-training dataset, X_p correspond the poisoned dataset, and θ denote the functionality of the benign encoder. We then generate poisoned data by maximizing the alignment between $\sum_{x_r \in X_{y_t}} -\nabla_\theta \mathcal{L}_{\mathrm{CL}}\left(x_t, x_r; \theta\left(X_c \cup X_p\right)\right)$ and $\nabla_\theta \mathcal{L}_{\mathrm{CL}}\left(X_c \cup X_p; \theta\right)$.

Interpolation Consistency Poisoning (ICP) [25]: ICP is a state-of-the-art targeted poisoning attack to semi-supervised learning. Specifically, ICP generates poisoned data as interpolations between a target sample and the sample with desired class. In our evaluation, we extend ICP to contrastive learning, where the selected target sample and the desired sample are the same with PE [12].

Note that PE is designed for targeted poisoning attacks in contrastive learning by adding poisons to the pre-training dataset. In contrast, PA-CL directly poisons the encoder. In addition, WB and ICP are originally designed for supervised learning and semi-supervised learning, respectively. We thus extend WB and ICP to the contrastive learning setting for fair comparisons. It is also crucial to highlight that the attack method introduced by Carlini and Terzis [11] is unsuitable for comparison due to its characteristics involving multimodal contrastive models.

3) Evaluation Metrics: Let C_p denote the poisoned downstream classifier and C_c represent the clean downstream classifier, and $\mathbb{I}$ refers to an indicator function. We adopt the following three metrics to evaluate the performance of PA-CL:

Attack Success Rate (ASR): Given the target testing samples, ASR is the proportion of these samples that are correctly predicted as the desired class for C_p. It can be formulated as:

$$ASR = \frac{\sum_{(\mathbf{x}_j, y_j) \in \mathcal{D}_t} \mathbb{I}(C_p(f_p(\mathbf{x}_j)) = y_d)}{m}, \tag{11}$$

Poisoned Test Accuracy (PTA): PTA is the fraction of testing examples (excluding the target samples) that are correctly predicted by the poisoned downstream classifier trained on the poisoned encoder, and can be expressed as:

$$PTA = \frac{\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_d \setminus \mathcal{D}_t} \mathbb{I}(C_p(f_p(\mathbf{x}_i)) = y_i)}{n - m}. \tag{12}$$

Clean Test Accuracy (CTA): CTA is the fraction of testing examples (excluding the target samples) that are correctly predicted by C_c trained on the benign encoder, which is given by:

$$CTA = \frac{\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_d \setminus \mathcal{D}_t} \mathbb{I}(C_c(f_b(\mathbf{x}_i)) = y_i)}{n - m}. \tag{13}$$

4) Model Setting: Next we describe the settings of the model training.

a) Pre-training encoder: In our experiments, we pretrain the encoder on CIFAR-10, STL-10 and GTSRB as the pre-training dataset, separately. Specifically, we use the training samples without the labels to pretrain the image encoder for CIFAR-10 and GTSRB dataset. Similarly, we utilize the unlabeled samples from the STL-10 dataset to pretrain an encoder. Furthermore, we consider the testing samples from the corresponding pre-training dataset as our surrogate dataset. This is because the surrogate dataset should be different from the pre-training dataset, however, they need to have the same distribution.

We adopt a popular representative contrastive learning algorithm SimCLR to train a ResNet18 model [26] as our benign encoder. SimCLR mainly consists of four major components, including the encoder, projection head, data augmentation, and contrastive loss function. During the training process, SimCLR generates two augmented samples for each data randomly in N mini-batches (denoted as $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$), resulting in totally $2N$ augmented data samples $\{\widetilde{\boldsymbol{x}}_1, \widetilde{\boldsymbol{x}}_2, \ldots, \widetilde{\boldsymbol{x}}_{2N}\}$. SimCLR considers two augmented samples $(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j)$ as a positive pair if they are created from the same data; otherwise, these two augmented samples are regarded as a negative pair. Then it trains the encoder so that the cosine similarity between the latent representations of the positive pairs is maximized and that of the negative pairs is minimized. We pretrain the encoder for 500 training epochs, with a batch size of 256, and an initial learning rate of 0.001, using the Adam optimizer. The metric d utilized in the loss function corresponds to cosine similarity.

b) Training downstream classifiers: To train the downstream classifier, we use the training dataset of the downstream dataset via a clean or poisoned encoder as a feature extractor. For instance, if we pretrain an encoder on CIFAR-10, we will use this encoder to train each downstream classifier for each downstream dataset (e.g., STL-10, GTSRB, and SVHN). Moreover, for the downstream classifier, we choose a fully connected neural network with two hidden layers as its architecture. We train the downstream classifier for 300 training epochs, with a batch size of 256, and an initial learning rate of 0.01, also using the Adam optimizer.

5) Experimental Settings: Here, we consider the following parameter settings for PA-CL: the attacker randomly selects a particular testing sample from different classes in the target downstream dataset, and randomly chooses a class that does not belong to the true class of the target sample as the desired class. For each experiment, we randomly select 50 target samples and conduct PA-CL separately. Then we report experimental results as average. To strike a balance between the target effectiveness and model utility goal, we set $\lambda_1 = 1$ and $\lambda_2 = 1$ in our experiments. We perform the experiments on NVIDIA-RTX-3090 GPU with 24 GB memory.

B. Effectiveness of PA-CL

We first conduct a comparative study to assess the attack effectiveness of PA-CL in comparison to baseline attacks. The performance comparison of PA-CL with WB, ICP, and PE on different pre-training datasets and downstream datasets is summarized in Table I and we evaluate ASR and PTA on

TABLE I
COMPARISON ANALYSIS WITH DIFFERENT PRE-TRAINING DATASETS AND DOWNSTREAM DATASETS

Pre-training Dataset	Downstream Dataset	No Attack	WB [24]		ICP [25]		PE [12]		PA-CL	
		CTA	ASR	PTA	ASR	PTA	ASR	PTA	ASR	PTA
CIFAR-10	GTSRB	0.817	0.18	0.798	0.82	0.802	0.72	0.823	**0.86**	0.834
	SVHN	0.586	0.06	0.604	0.46	0.624	0.36	0.635	**0.48**	0.689
	STL-10	0.757	0.12	0.726	0.24	0.731	0.74	0.715	**0.76**	0.763
STL-10	GTSRB	0.767	0.24	0.714	0.86	0.705	0.82	0.776	**0.92**	0.785
	SVHN	0.548	0.02	0.554	0.42	0.567	0.32	0.578	**0.42**	0.655
	CIFAR-10	0.865	0.08	0.843	0.14	0.858	0.54	0.867	**0.66**	0.869
GTSRB	CIFAR-10	0.623	0.26	0.625	0.34	0.631	0.66	0.642	**0.74**	0.651
	SVHN	0.554	0.04	0.542	0.44	0.568	0.34	0.579	**0.48**	0.583
	STL-10	0.526	0.20	0.513	0.28	0.528	0.78	0.557	**0.80**	0.549
Average		0.671	0.13	0.658	0.44	0.668	0.59	0.686	**0.68**	0.709

all the three downstream datasets for a given pre-training dataset. The results indicate that the poisoned encoders can successfully predict target samples as the desired class for most cases and PA-CL can obtain high attack success rates for different targets. Generally, PA-CL can reach 68% ASR on average and it has 56%, 19%, and 10% ASR higher than WB, ICP, and PE, respectively. We can notice that WB is almost ineffective and ICP is also invalid for most cases. This is largely because WB and ICP are initially designed for supervised learning and semi-supervised learning, and they do not perform well when extended to contrastive learning. We also notice that ICP achieves moderate ASRs but performs better than WB, possibly due to the data augmentations during the pre-training phase. PE is relatively effective when the downstream dataset is GTSRB, CIFAR-10, and STL-10. However, ASR is relatively lower than other cases when the downstream dataset is SVHN. The reason is that the large gap between STL-10 (or CIFAR-10) and the unbalanced SVHN dataset could lead to a lower ASR. Additionally, it is noticeable that conducting attacks becomes easier when the pre-training dataset is GTSRB. This may be attributed to GTSRB having a larger number of classes in the dataset, leading to a decrease in model accuracy. Overall, PA-CL outperforms WB, ICP and PE in most cases.

Table I also illustrates the PTA of PA-CL, WB, ICP, and PE on different pre-training datasets and downstream datasets. For better comparison, the column of "No Attack" shows the CTA before conducting attacks. The experimental results show that CTA and PTA are relatively closer to each other for different downstream testing datasets. Generally, the gap between the CTA and PTA for a given downstream dataset is within 5% on average for PA-CL. This confirms that PA-CL can also preserve the utility of the encoder. We can observe that PA-CL not only learns stronger feature representations for non-target samples but also learns the poisoned behavior just as expected.

In addition, we can find that PTA is always slightly higher than CTA. This is mainly because some downstream task information from the downstream training dataset has fed into the poisoned encoder during training the poisoned encoder, making it easier for representation learning of the corresponding downstream tasks. Specially, PTA is relatively higher than CTA when the downstream dataset is SVHN. A key reason behind this is that the SVHN dataset is noisy and the representation capability of the poisoned encoder becomes stronger when fine-tuning the benign encoder with the clean dataset. Our results show that our poisoned encoder can generate much more similar feature vectors for the target input and the desired inputs than that of PE because PA-CL does not distort the association for other classes.

TABLE II
ASR OF PA-CL WITH BASELINE ATTACKS FOR DIFFERENT CONTRASTIVE LEARNING ALGORITHMS

Pre-training Dataset	Pre-training Algorithm	ASR			
		WB [24]	ICP [25]	PE [12]	PA-CL
CIFAR-10	SimCLR	0.18	0.82	0.74	0.86
	MoCo	0.22	0.86	0.82	0.92
	SupCon	0.14	0.76	0.70	0.82
STL-10	SimCLR	0.24	0.86	0.82	0.92
	MoCo	0.22	0.88	0.84	0.94
	SupCon	0.18	0.80	0.74	0.84
Average		0.20	0.83	0.78	0.88

C. Impact of Learning Algorithm and Encoder Architecture

Next, we evaluate the impact of different contrastive learning algorithms and encoder architectures on the ASR of the attacks. Table II illustrates the ASR of PA-CL, WB, ICP, and PE when the contrastive learning algorithms SimCLR [13], MoCo [14] and SupCon [27] are used to pre-train encoders. Compared to SimCLR, MoCo adopts an additional momentum encoder that can achieve high performance without large mini-batches and SupCon trains encoder via supervised learning method. Here, we use GTSRB as our downstream training dataset. Our experimental results show that the ASR is relatively high when MoCo is used to pre-train encoders, which indicates that MoCo is more vulnerable to poisoning

TABLE III
ASR OF PA-CL WITH BASELINE ATTACKS FOR DIFFERENT ENCODER ARCHITECTURES

Pre-training Dataset	Encoder Architecture	ASR			
		WB [24]	ICP [25]	PE [12]	PA-CL
CIFAR-10	ResNet18	0.18	0.82	0.74	0.86
	VGG11	0.24	0.86	0.78	0.90
	DenseNet121	0.18	0.80	0.74	0.82
	MobileNet-v2	0.16	0.80	0.72	0.82
STL-10	ResNet18	0.24	0.86	0.82	0.92
	VGG11	0.30	0.90	0.84	0.94
	DenseNet121	0.24	0.86	0.80	0.88
	MobileNet-v2	0.22	0.86	0.78	0.88
Average		0.22	0.85	0.78	0.88

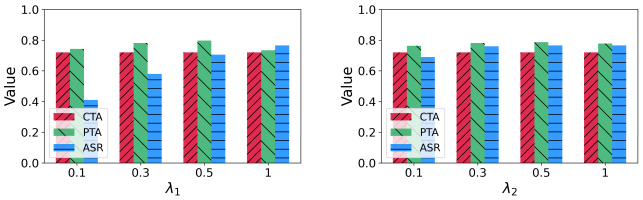
attacks. Additionally, SupCon exhibits a stronger resistance to our attack. PA-CL is effective for both contrastive learning algorithms when the pre-trained dataset is CIFAR-10. Besides, PA-CL outperforms other baseline attacks on all the attack settings, indicating its strong attacking capability.

Table III demonstrates the ASR of PA-CL, WB, ICP and PE when different encoder architectures (ResNet18 [26], VGG11 [28], DenseNet121 [29] and MobileNet-v2 [30]) are used to pre-train encoders. The experimental results show that the ASR of PA-CL is relatively high under different encoder architectures compared to other baselines. What is more, PA-CL is agnostic to encoder architecture, because it does not rely on the encoder architecture to generate poisoned data. Generally, the results show that VGG11 is more vulnerable to different poisoning attacks and MobileNet-v2 is more robust. It also can be observed that PA-CL can reach higher ASR compared to PE. This again confirms that PA-CL is a strong and robust attack method.

D. Impact of the Parameters on the Loss Terms

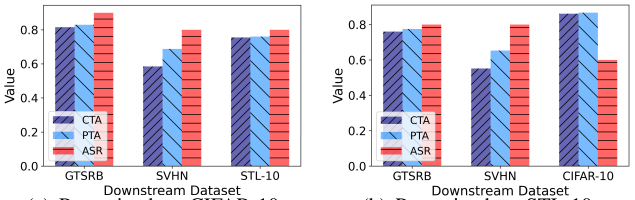
Next, we evaluate the impact of parameters λ_1 and λ_2 in Eq. (10) to illustrate how the parameters balance the performance. Here, we select CIFAR-10 as the pre-training dataset, with GTSRB and STL-10 as the downstream dataset. We average the results on different cases. Fig. 3 shows CTA, PTA and ASR when different λ_1 and λ_2 are selected. Our experimental results indicate that λ_1 and λ_2 are necessary for PA-CL to achieve a high ASR, while maintaining the utility of the downstream classifiers. It can be observed that the ASR remains high when λ_1 is fixed, while the ASR fluctuates when λ_1 changes. This is largely because λ_1 is designed to achieve the target effectiveness goal. Similar for λ_2, its role is to preserve the utility of the downstream classifiers and the PTA fluctuates when λ_2 changes. Besides, the gap between the CTA and PTA for all cases is within 5%, indicating PA-CL can preserve the utility of the encoder.

In summary, both λ_1 and λ_2 are essential parameters in PA-CL, and their careful selection and fine-tuning process are necessary to achieve a high ASR while maintaining stable PTA for the downstream classifiers. Generally, a higher value of λ_1 would make more emphasis on optimizing the poisoned encoder to achieve the desired misprediction during the testing phase, but could compromise the original functionality of the pre-trained encoder. Meanwhile, a higher value of λ_2 would ensure the encoder to retain its original classification capability for non-poisoned samples, and allow the encoder to adapt more flexibly to the poisoning attack. Therefore, it is crucial to find the right balance between these parameters for the success of the attack methodology.



(a) Impact of λ_1 when λ_2 is fixed (b) Impact of λ_2 when λ_1 is fixed

Fig. 3. Impact of λ_1 and λ_2 on CTA, PTA and ASR.



(a) Pre-trained on CIFAR-10 (b) Pre-trained on STL-10

Fig. 4. Impact of attacking multiple target downstream tasks.

E. Impact of Attacking Multiple Target Downstream Tasks

We next evaluate the performance of PA-CL on attacking multiple target downstream tasks. We use either CIFAR10 or STL-10 as the pre-training dataset and attack three target downstream tasks simultaneously. For each target downstream task, we randomly select one target sample and a corresponding desired class. The results depicted in Fig. 4 demonstrate the effectiveness of PA-CL in attacking multiple target downstream tasks simultaneously. Notably, PA-CL achieves comparable ASR when compared to attacking each target downstream task separately, achieving 80% ASR on average. Importantly, PA-CL also maintains the prediction accuracy of the downstream classifiers and the PTA of PA-CL is generally equal to the CTA in most cases. This implies that the poisoned encoder generated by PA-CL does not significantly compromise the overall classification performance of the downstream classifiers on non-target samples. It is also interesting to note that in the case when the downstream dataset is SVHN, the PTA of PA-CL is higher than the CT. This can be attributed to the noisy characteristic of SVHN and the fact that the poisoned data may resemble data augmentation techniques, which can have a positive effect on the classification performance of the downstream classifier.

F. Impact of Attacking Multiple Target Inputs

We also evaluate the performance of PA-CL on attacking multiple target classes in a single target downstream task. Specifically, the attacker randomly selects multiple target samples from different target classes in the same target downstream task. The experiment is performed using GTSRB as

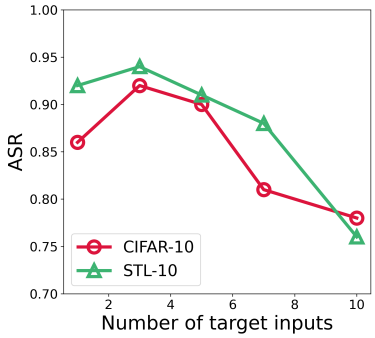


Fig. 5. Impact of attacking multiple target inputs.

the downstream dataset, and either CIFAR-10 or STL-10 as the pre-training dataset.

From the results in Fig. 5, we can observe that PA-CL can effectively attack multiple target inputs simultaneously. As the number of target samples increases, PA-CL consistently maintains a high ASR. For instance, the ASR of PA-CL can reach an impressive ASR of 95% when attacking three target inputs simultaneously. However, it is observed that the ASR begins to decrease as the number of target inputs continues to increase, and obtains an ASR of 75% when attacking ten target inputs. The reason is that manipulating a small number of target samples to a certain extent is relatively easier for the attacker; however, as the number of target samples continues to increase, the attacker faces greater challenges in controlling the similarity between the feature representations of different target samples. The increased complexity makes it more difficult to maintain the desired associations for all target samples simultaneously, resulting in a decrease in the overall effectiveness of the attack.

G. Impacts of Learning Rate and Batch Size

Table IV demonstrates the ASR of PA-CL when pre-training the encoder and training the downstream classifier built based on the poisoned encoder using different learning rates or batch sizes. We use CIFAR-10 as the pre-training dataset and GTSRB as the downstream dataset. We can observe that PA-CL achieves consistently high ASRs under different parameter settings. In particular, PA-CL achieves a slightly lower ASR when pre-training the encoder with the learning rate 5×10^{-3}. This is because the encoder learns less feature representations under this learning rate. We can notice that the ASR may remain the same in most cases. The reasons are twofold. On the one hand, the success of PA-CL primarily relies on the ability to manipulate the feature representations of the benign encoder. And, the learning rate and batch size mainly affect the optimization process during training but do not directly control the feature manipulation. On the other hand, PA-CL may exhibit robustness to hyperparameters such as the learning rate and batch size. This means that even if the hyperparameters are not optimally tuned, the attack can still achieve a high ASR due to the model-dependent characteristics of various hyper-parameters.

TABLE IV
ASR OF PA-CL VS. LEARNING RATE/BATCH SIZE

(a) Pre-training encoder

Parameter	Value	ASR
Learning Rate	5×10^{-3}	0.68
	1×10^{-3}	0.86
	5×10^{-4}	0.86
Batch Size	256	0.86
	512	0.86
	1024	0.86

(b) Training downstream classifier

Parameter	Value	ASR
Learning Rate	5×10^{-3}	0.82
	1×10^{-3}	0.86
	5×10^{-4}	0.86
Batch Size	256	0.82
	512	0.86
	1024	0.86

TABLE V
RESULTS OF PA-CL WHEN ATTACKING THE IMAGE ENCODER PRE-TRAINED ON IMAGENET AND CLIP

Pre-training Dataset	Target Downstream Dataset	CTA	PTA	ASR
ImageNet	GTSRB	0.743	0.682	0.98
	SVHN	0.716	0.728	0.68
	STL-10	0.951	0.958	0.82
CLIP	GTSRB	0.813	0.798	0.96
	SVHN	0.688	0.665	0.72
	STL-10	0.957	0.946	0.84
Average		0.811	0.796	0.83

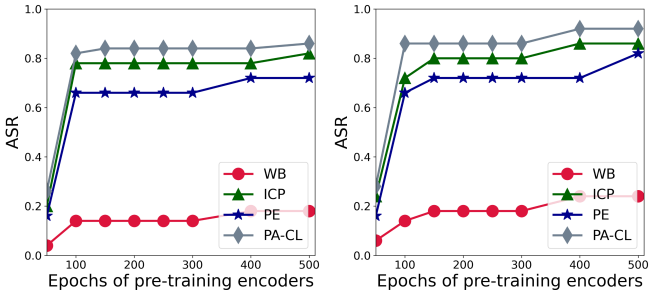
H. Evaluation on ImageNet and CLIP

At last, we evaluate the effectiveness of PA-CL on two larger datasets: ImageNet released by Google and multi-modal dataset CLIP [1] released by OpenAI, which pre-trained on 400 million (image, text) pairs collected from the Internet.

For attacking image encoder pre-trained on ImageNet, we use the ImageNet dataset to build the benign encoder and then select the target downstream dataset and the target class. Specifically, we choose "truck", "priority sign", and "digit one" as the target classes for the datasets STL10, GTSRB, and SVHN, respectively. Note that we resize each image in the downstream datasets to be $224 \times 224 \times 3$ to match the size of image in ImageNet.

For attacking encoder pre-trained on CLIP, we first obtain the pre-trained CLIP encoder, which includes ResNet50 for the image encoder and a Transformer as the backbone for the text encoder. Similarly, we consider a target downstream task and a target class. We choose "truck", "priority sign", and "digit one" as the target classes for the datasets STL10, GTSRB, and SVHN, respectively.

We can observe the results in Table V that PA-CL can achieve 82.7% ASR on average with stable prediction accuracy for non-target samples when the pre-training dataset is ImageNet. This indicates that PA-CL is capable of effectively manipulating the feature representations of the target samples to achieve the desired misclassifications even under practical scenarios. Our results also show the practical applicability of PA-CL in scenarios where the image encoder is pre-trained on large amounts of unlabeled data and (image, text) pairs (i.e., CLIP) and can achieve 84% ASR on average. This highlights



(a) Pre-trained on CIFAR-10 (b) Pre-trained on STL-10

Fig. 6. ASR of PA-CL and baseline attacks against early stopping.

TABLE VI
PTA OF PA-CL AND BASELINES FOR EARLY STOPPING

Metric	Pre-training Dataset	Attacks	Epochs					
			50	100	200	300	400	500
PTA	CIFAR-10	WB	0.706	0.721	0.768	0.780	0.785	0.798
		ICP	0.732	0.753	0.778	0.790	0.798	0.802
		PE	0.743	0.771	0.785	0.792	0.818	0.823
		PA-CL	**0.764**	**0.778**	**0.798**	**0.805**	**0.821**	**0.834**
	STL-10	WB	0.603	0.653	0.676	0.688	0.703	0.714
		ICP	0.601	0.643	0.655	0.681	0.688	0.705
		PE	0.686	0.724	0.731	0.756	0.770	0.776
		PA-CL	**0.705**	**0.728**	**0.743**	**0.762**	**0.774**	**0.785**

the robustness and versatility of PA-CL, as it can effectively adapt to the complex representations learned by the encoder and exploit them to generate poisoned encoders for successful attacks.

V. DEFENSES

In this section, we discuss some existing defenses for defending poisoning attacks, and then evaluate potential defenses against our attack and other baselines.

A. Leveraging Existing Defenses Against Poisoning Attacks

Existing defenses against data poisoning attacks can be generally categorized into three types: pre-training defenses, in-training defenses, and post-training defenses (detailed in Section VI-C). However, these are primarily designed for traditional machine learning scenarios and few defenses are available for the contrastive learning setting. Thus, we consider three simple yet effective defenses against poisoning attacks and extend them to contrastive learning setting.

1) Early Stopping (ES): Early Stopping is an in-training defense against poisoning attacks, which is widely adopted in avoiding unnecessary training iterations and improving robustness of the pre-trained model. Since our PA-CL relies on enough training epochs of both pre-training process and downstream process, early stopping may mitigate PA-CL to some extent if the encoder or the downstream classifier is trained using less epochs.

2) Fine-Pruning (FP): Fine-Pruning [31] is a post-training defense against data poisoning attacks. The poisoned effect of a potentially poisoned classifier can be removed by fine-tuning using a subset of clean training dataset. In practical scenario, the defender can have two ways to mitigate the poisoned effect, by using some clean data to either fine-tune the poisoned encoder or fine-tune the poisoned downstream classifier.

3) Model Distillation (MD): Model Distillation [32] is a post-training defense against data poisoning attacks which involves transferring knowledge from a larger model to a smaller one. By doing so, one aims to create a more robust and clean representation of the underlying data. This process helps mitigate the poisoned effect on the pre-trained encoder, enhancing its resilience against adversarial attempts to poison the model.

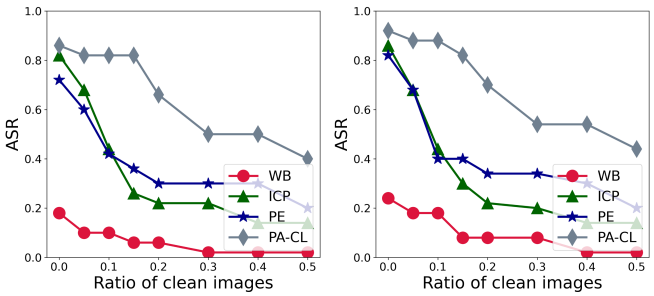
It is worth noting that the performance of these two defenses depends on the setting of some key parameters. In early stopping defense, the number of epochs of encoders determines the utility ability of encoders. In fine-pruning defense, the ratio of clean images to fine-tune the encoder can be adjusted to mitigate poisoning attacks. In our evaluation, we use CIFAR-10 or STL-10 as our pre-training dataset and GTSRB as the downstream dataset.

B. Performance Results

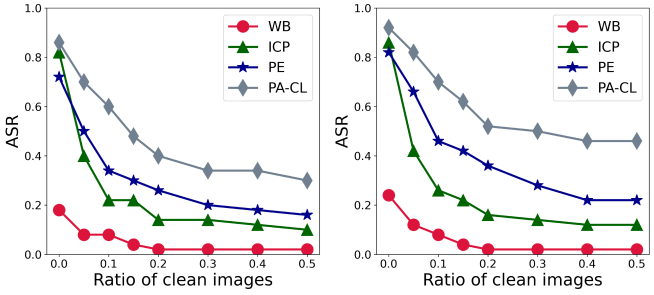
Fig. 6 demonstrates the performance of the early stopping defense against PA-CL and baselines for different epochs of encoders. We report the ASR of the early stopping defense when epochs change. We can observe that early stopping can effectively reduce the ASR when using less epochs to pre-train an encoder. Generally, the ASR of different attacks remains relatively high when epochs of encoders are over 100 and the ASR is reduced a lot when the epoch is 50. Nevertheless, PA-CL still achieves the highest ASR among different attacks. However, it is indicated in Table VI that early stopping also reduces the prediction accuracy of the downstream classifiers built on the poisoned encoder. Thus, this method sacrifices the utility of the encoders.

Fig. 7 shows the performance of the fine-pruning defense against PA-CL and baselines for different ratios of clean images in the pre-training dataset to fine-tune the poisoned encoder/downstream classifier. It can be seen that the ASR is reduced by 50% on average when the ratio of clean images is higher than 30% at the cost of sacrificing a lot of time to gather a large amount of clean data manually, and the ASR of different attacks can be reduced by the increased ratio of clean images. To fine-tune the poisoned downstream classifier, the ASR reduces even faster, because the downstream dataset is smaller than the pre-trained dataset and the defender could have stronger capability to mitigate poisoning attacks using less clean images.

Fig. 8 illustrates the performance of the model distillation defense against PA-CL and baselines for different training epochs. We can observe that the ASR is reduced by 50% on average when the epoch is more than 200 for both CIFAR-10 and STL-10 datasets. Notably, the effectiveness of MD is more pronounced when encountering attacks such as WB, ICP, and PE. This is because these three attacks introduce poisoned data during the training of the pre-trained

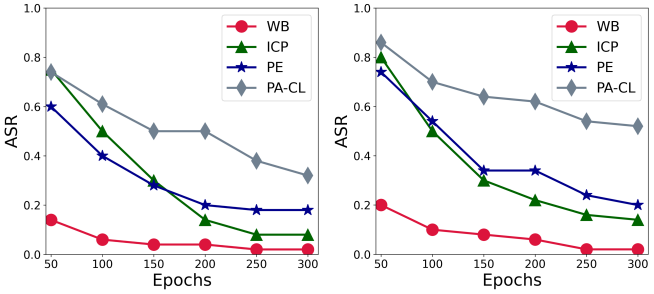


(a) Pre-trained on CIFAR-10 (b) Pre-trained on STL-10



(c) Pre-trained on CIFAR-10 (d) Pre-trained on STL-10

Fig. 7. ASR of PA-CL and baseline attacks against fine-pruning. First row: pre-training an encoder. Second row: training a downstream classifier.



(a) Pre-trained on CIFAR-10 (b) Pre-trained on STL-10

Fig. 8. ASR of PA-CL and baseline attacks against model distillation.

encoder, and MD can successfully mitigate the impact of this poisoning method. In addition, PA-CL exhibits the highest resilience to the MD defense, which can be attributed to its optimization process that does not involve poisoned data.

Furthermore, we evaluate the ASR of PA-CL against different defense methods under different encoder architectures. The results are depicted in Fig. 9. Specifically, we employ ResNet18, VG11, DenseNet121, and MobileNet-v2 for training the encoder. When using the ES method, we pretrain the encoder for 200 epochs. For the FP method, we set the ratio of clean images to 10%. When applying the MD method, we train the encoder for 100 epochs. From Fig. 9, it can be observed that MD exhibits the strongest defense capability compared to ES and MD methods. Generally, the ASR of PA-CL remains at approximately 70% when confronted with different defense methods. This demonstrates the robustness of PA-CL against various defense techniques. Notably, VGG11 is more susceptible to attacks compared to other encoder architectures due to its simpler architecture.

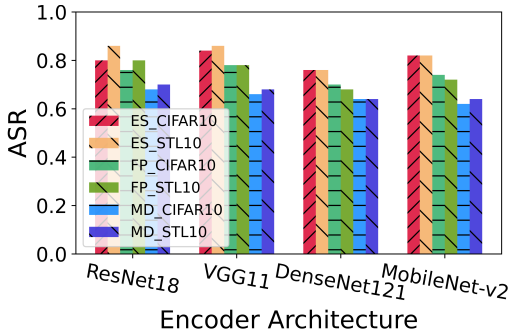


Fig. 9. ASR of PA-CL against different defense methods under different encoder architectures.

VI. Related Work

A. Contrastive Learning

Contrastive learning is an emerging paradigm that leverages abundant unlabeled data to pretrain encoders, which can then be utilized for various downstream tasks. It has achieved significant success across a variety of fields. In the computer vision field, unsupervised training with unlabeled images has been explored extensively [33], [34], [35], as well as utilizing (image, text) pairs to achieve state-of-the-art prediction performance [1], [36], [37]. For the natural language processing, large pre-trained language models [38] are trained on vast amounts of unlabeled text data, enabling them to use in tasks such as text classification [39] and machine translations [7]. Furthermore, graph neural networks can learn effective representations of graphs, enabling improved performance in graph-related tasks by applying contrastive learning [40], [41].

B. Poisoning Attacks

Poisoning attacks aim to manipulate the integrity of the training phase (i.e., training dataset collection and the learning process) of a learning system. Most existing poisoning attacks focus on compromising the training dataset collection process, which are also known as *data* poisoning attacks [42], [43], [44]. Different from data poisoning attacks, our poisoning attack focuses on compromising the learning process in the pre-training phase of contrastive learning.

Generally, poisoning attacks can be categorized into two types based on their goals: untargeted poisoning attacks [45], [46] and targeted poisoning attacks [17], [24], [47], [48], [49], [50]. The former aims to degrade the performance of the learned model for testing samples, which eventually leads to a denial-of-service attack. The latter can cause misclassification on a specific target sample while maintaining the functionality of the model for legitimate data.

Previous targeted poisoning attacks mainly focus on supervised learning [42] or semi-supervised learning [25], leaving its vulnerability in contrastive learning largely unexplored. Recently, Liu et al. [12] propose targeted data poisoning attacks to contrastive learning; He et al. [10] explore indiscriminate data poisoning attacks within the context of contrastive learning. However, both attacks do not directly poison the pre-trained model but rather the training data in contrastive learning, which is actually the focus of our attack.

C. Defenses Against Poisoning Attacks

Existing defenses can be broadly classified into three main categories: pre-training defenses, in-training defenses and post-training defenses.

Specifically, pre-training defenses are designed to identify poisoned samples before the training process [51]. For instance, Paudice et al. [52] and Peri et al. [53] detect poisoned data by identifying inconsistencies in labels between a training sample and its nearest neighbors. Chen et al. [54] distinguish poisoned data and benign samples by comparing the prediction results of both the target and mimic models.

In-processing defenses detect poisoned data by assessing their influence on the performance of the trained model [43], [55]. For example, Carlini [25] devise a defense strategy that assigns a score to each unlabeled training sample, which is computed based on the influence by its nearest neighbors on the labels predicted by the initial model during training. By leveraging these scores, it can effectively detect and subsequently remove the poisoned data from the training set.

In the case of post-processing defenses, their primary objective is to mitigate the effect of poisoned training data from a trained model [51], [56]. These methods typically involve retraining or fine-tuning the model using a modified training procedure that effectively removes the influence of poisoned data on the model.

It is worth noting that all these defenses may not be directly applicable to defense our proposed attack, PA-CL because our attack methodology does not involve the injection of poisoned data into the training dataset.

VII. Conclusion

In this work, we have presented PA-CL, a more surreptitious targeted poisoning attack in contrastive learning, by directly poisoning the encoder without injecting poisoned samples into the training set. To launch the attack, PA-CL maximizes the similarity between the feature of the target sample and the averaged feature that belongs to the feature space of the desired class. Extensive experimental results on four real-world datasets show that PA-CL can achieve a high ASR, demonstrating that the encoder is potentially vulnerable to our attack. Our work indicates that the threat of targeted poisoning attacks against contrastive learning is largely underestimated, which also calls for effective protections in the near future.

References

[1] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. ICML*, 2021, pp. 8748–8763.

[2] Z. Huang, J. Zhang, and H. Shan, "Twin contrastive learning with noisy labels," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 11661–11670.

[3] X. Wang and G.-J. Qi, "Contrastive learning with stronger augmentations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5549–5560, May 2023.

[4] S. H. Khorasgani, Y. Chen, and F. Shkurti, "SLIC: Self-supervised learning with iterative clustering for human action videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 16070–16080.

[5] S.-Q. Liu, X. Lan, and P. C. Yuen, "Learning temporal similarity of remote photoplethysmography for fast 3D mask face presentation attack detection," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3195–3210, 2022.

[6] H. Fang and P. Xie, "An end-to-end contrastive self-supervised learning framework for language understanding," *Trans. Assoc. Comput. Linguistics*, vol. 10, pp. 1324–1340, Nov. 2022.

[7] T. Zhang et al., "Frequency-aware contrastive learning for neural machine translation," in *Proc. AAAI*, 2022, vol. 36, no. 10, pp. 11712–11720.

[8] A. Saha, A. Tejankar, S. A. Koohpayegani, and H. Pirsiavash, "Backdoor attacks on self-supervised learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 13327–13336.

[9] J. Jia, Y. Liu, and N. Z. Gong, "BadEncoder: Backdoor attacks to pre-trained encoders in self-supervised learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 2043–2059.

[10] H. He, K. Zha, and D. Katabi, "Indiscriminate poisoning attacks on unsupervised contrastive learning," in *Proc. ICLR*, 2023, pp. 1–12.

[11] N. Carlini and A. Terzis, "Poisoning and backdooring contrastive learning," in *Proc. ICLR*, 2022, pp. 1–13.

[12] H. Liu, J. Jia, and N. Z. Gong, "PoisonedEncoder: Poisoning the unlabeled pre-training data in contrastive learning," in *Proc. USENIX Secur. Symp.*, 2022, pp. 3629–3645.

[13] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. ICML*, 2020, pp. 1597–1607.

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

[15] G. Li, N. Duan, Y. Fang, M. Gong, and D. Jiang, "Unicoder-VL: A universal encoder for vision and language by cross-modal pre-training," in *Proc. AAAI*, 2020, pp. 11336–11344.

[16] X. Chen and K. He, "Exploring simple Siamese representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 15745–15753.

[17] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *Proc. ICML*, 2019, pp. 7614–7623.

[18] S. S. Bama and A. Saravanan, "Efficient classification using average weighted pattern score with attribute rank based feature selection," *Int. J. Intell. Syst. Appl.*, vol. 11, no. 7, pp. 29–42, Jul. 2019.

[19] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.

[20] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. 5, 2009.

[21] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. AISTATS*, 2011, pp. 215–223.

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

[23] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NeurIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, pp. 1–9.

[24] J. Geiping et al., "Witches' brew: Industrial scale data poisoning via gradient matching," in *Proc. ICLR*, 2020, pp. 1–13.

[25] N. Carlini, "Poisoning the unlabeled dataset of semi-supervised learning," in *Proc. USENIX Secur. Symp.*, 2021, pp. 1577–1592.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[27] P. Khosla et al., "Supervised contrastive learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 18661–18673.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015, pp. 1–10.

[29] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[31] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Proc. RAID*, 2018, pp. 273–294.

[32] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[33] C. Liu et al., "Learning a few-shot embedding model with contrastive learning," in *Proc. AAAI*, 2021, pp. 8635–8643.

[34] N. Pu, Z. Zhong, and N. Sebe, "Dynamic conceptional contrastive learning for generalized category discovery," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 7579–7588.

[35] U. Mall, B. Hariharan, and K. Bala, "Change-aware sampling and contrastive learning for satellite images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 5261–5270.

[36] J. Chen, R. Zhang, Y. Mao, and J. Xu, "Contrastnet: A contrastive learning framework for few-shot text classification," in *Proc. AAAI*, 2022, pp. 10492–10500.

[37] Y. Zeng et al., "CLIP2: Contrastive language-image-point pretraining from real-world point cloud data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 15244–15253.

[38] Y. Hao et al., "Feature-level deeper self-attention network with contrastive learning for sequential recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 10, pp. 1–13, Oct. 2023.

[39] D. Wang, N. Ding, P. Li, and H. Zheng, "CLINE: Contrastive learning with semantic negative examples for natural language understanding," in *Proc. ACL*, 2021, pp. 2332–2342.

[40] X. Huang et al., "Graph contrastive learning for skeleton-based action recognition," in *Proc. ICLR*, 2022, pp. 1–11.

[41] Q. Zhang, C. Huang, L. Xia, Z. Wang, Z. Li, and S. Yiu, "Automated spatio-temporal graph contrastive learning," in *Proc. WWW*, 2023, pp. 295–305.

[42] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proc. ICML*, 2017, pp. 1885–1894.

[43] O. Suciu, R. Marginean, Y. Kaya, H. Daume III, and T. Dumitras, "When does machine learning fail? Generalized transferability for evasion and poisoning attacks," in *Proc. USENIX Secur. Symp.*, 2018, pp. 1299–1316.

[44] M. Goldblum et al., "Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 1563–1580, Feb. 2023.

[45] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Proc. NIPS*, 2016, pp. 1893–1901.

[46] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 19–35.

[47] L. Muñoz-González et al., "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, Nov. 2017, pp. 27–38.

[48] A. Shafahi et al., "Poison frogs! Targeted clean-label poisoning attacks on neural networks," in *Proc. NIPS*, 2018, pp. 6106–6116.

[49] H. Aghakhani, D. Meng, Y.-X. Wang, C. Kruegel, and G. Vigna, "Bullseye polytope: A scalable clean-label poisoning attack with improved transferability," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Sep. 2021, pp. 159–178.

[50] Z. Yang et al., "Data poisoning attacks against multimodal encoders," in *Proc. ICML*, 2023, pp. 39299–39313.

[51] Y. Zeng, M. Pan, H. Jahagirdar, M. Jin, L. Lyu, and R. Jia, "Meta-sift: How to sift out a clean subset in the presence of data poisoning?" in *Proc. USENIX Secur. Symp.*, 2023, pp. 1667–1684.

[52] A. Paudice, L. Muñoz-González, and E. C. Lupu, "Label sanitization against label flipping poisoning attacks," in *Proc. ECML PKDD*, 2018, pp. 5–15.

[53] N. Peri et al., "Deep k-NN defense against clean-label data poisoning attacks," in *Proc. ECCV*, 2020, pp. 55–70.

[54] J. Chen, X. Zhang, R. Zhang, C. Wang, and L. Liu, "De-pois: An attack-agnostic defense against data poisoning attacks," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3412–3425, 2021.

[55] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Proc. NIPS*, 2018, pp. 8011–8021.

[56] B. Wang et al., "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 707–723.