Indiscriminate Data Poisoning Attacks on Pre-trained Feature Extractors

Yiwei Lu
School of Computer Science
University of Waterloo
Vector Institute
yiwei.lu@uwaterloo.ca

Matthew Y. R. Yang
School of Computer Science
University of Waterloo
Waterloo, Canada
m259yang@uwaterloo.ca

Gautam Kamath
School of Computer Science
University of Waterloo
Vector Institute
gckamath@uwaterloo.ca

Yaoliang Yu
School of Computer Science
University of Waterloo
Vector Institute
yaoliang.yu@uwaterloo.ca

***Abstract*—Machine learning models have achieved great success in supervised learning tasks for end-to-end training, which requires a large amount of labeled data that is not always feasible. Recently, many practitioners have shifted to self-supervised learning (SSL) methods (e.g., contrastive learning) that utilize cheap unlabeled data to learn a general feature extractor via pre-training, which can be further applied to personalized downstream tasks by simply training an additional linear layer with limited labeled data. However, such a process may also raise concerns regarding data poisoning attacks. For instance, indiscriminate data poisoning attacks, which aim to decrease model utility by injecting a small number of poisoned data into the training set, pose a security risk to machine learning models, but have only been studied for end-to-end supervised learning. In this paper, we extend the exploration of the threat of indiscriminate attacks on downstream tasks that apply pre-trained feature extractors. Specifically, we propose two types of attacks: (1) *the input space attacks*, where we modify existing attacks (e.g., TGDA attack and GC attack) to directly craft poisoned data in the input space. However, due to the difficulty of optimization under constraints, we further propose (2) *the feature targeted attacks*, where we mitigate the challenge with three stages, firstly acquiring target parameters for the linear head; secondly finding poisoned features by treating the learned feature representations as a dataset; and thirdly inverting the poisoned features back to the input space. Our experiments examine such attacks in popular downstream tasks of fine-tuning on the same dataset and transfer learning that considers domain adaptation. Empirical results reveal that transfer learning is more vulnerable to our attacks. Additionally, input space attacks are a strong threat if no countermeasures are posed, but are otherwise weaker than feature targeted attacks.**

I. Introduction

Modern machine learning models, especially deep neural networks, often train on a large amount of data to achieve superb performances. To collect such large-scale datasets, practitioners usually extract the desired data by crawling on the internet (e.g., web pages using Common Crawl[1]). However, using outsourced data raises an imminent security risk (Kumar et al. 2020; Nelson et al. 2008; Szegedy et al. 2014), namely that by carefully crafting a small amount of "poisoned" data, an adversary can throttle the training and hence prediction of a machine learning pipeline maliciously (Gao et al. 2020; Lyu et al. 2020; Shejwalkar et al. 2022; Wakefield 2016). More formally, such a threat is called *data poisoning attacks* (Biggio et al. 2012).

Among various data poisoning attacks (w.r.t. different adversary objectives), we focus on indiscriminate attacks, which aim to reduce the overall model performance during testing (e.g., accuracy for classifiers). Indiscriminate attacks have been well-studied under end-to-end training schemes (Biggio et al. 2012; Koh et al. 2022; Lu et al. 2022, 2023a; Muñoz-González et al. 2017; Suya et al. 2021, and the references therein), e.g., supervised learning for image classification tasks. In particular, Lu et al. (2023a) discussed the underlying difficulties of indiscriminate attacks and proposed one of the most effective attack algorithms called Gradient Canceling (GC).

However, as end-to-end training urges the need for large labeled datasets, practitioners with limited budgets are gradually adopting pre-trained feature extractors returned by self-supervised learning (SSL, e.g., contrastive learning methods (Chen et al. 2020a; He et al. 2020)) for personalized downstream tasks. Specifically, SSL pre-trains a general feature extractor f with potentially unlimited unlabeled data (e.g., in the wild) such that one only needs to train an additional linear head h on top of f. While several works have studied how to poison the pre-training process on obtaining f (Balcan et al. 2022; Carlini and Terzis 2021; He et al. 2022), there lacks an exploration of the vulnerability of performing downstream tasks given a certified f under data poisoning attacks, which could pose a serious threat to SSL beneficiaries.

Concretely, we aim to answer the following question:

> *Can we poison downstream tasks with a fixed (and trustworthy) pre-trained feature extractor?*

Specifically, we consider two popular downstream tasks in this paper: (1) use the same (but labeled) dataset as pre-training for *fine-tuning*[2] the linear head h (where the parameters of the feature extractor f is frozen); (2) adapt to a new dataset for *transfer learning*.

To address the above question, we first adopt existing data poisoning attacks in a straightforward manner (which we refer to as *input space attack*) to directly craft poisoned points using

GK and YY are listed in alphabetical order.

[1]https://commoncrawl.org/

[2]The term "fine-tuning" can be used in different contexts for training with a pre-trained network as initialization. In this paper, we restrict "fine-tuning" to performing contrastive learning and downstream tasks on the same dataset.

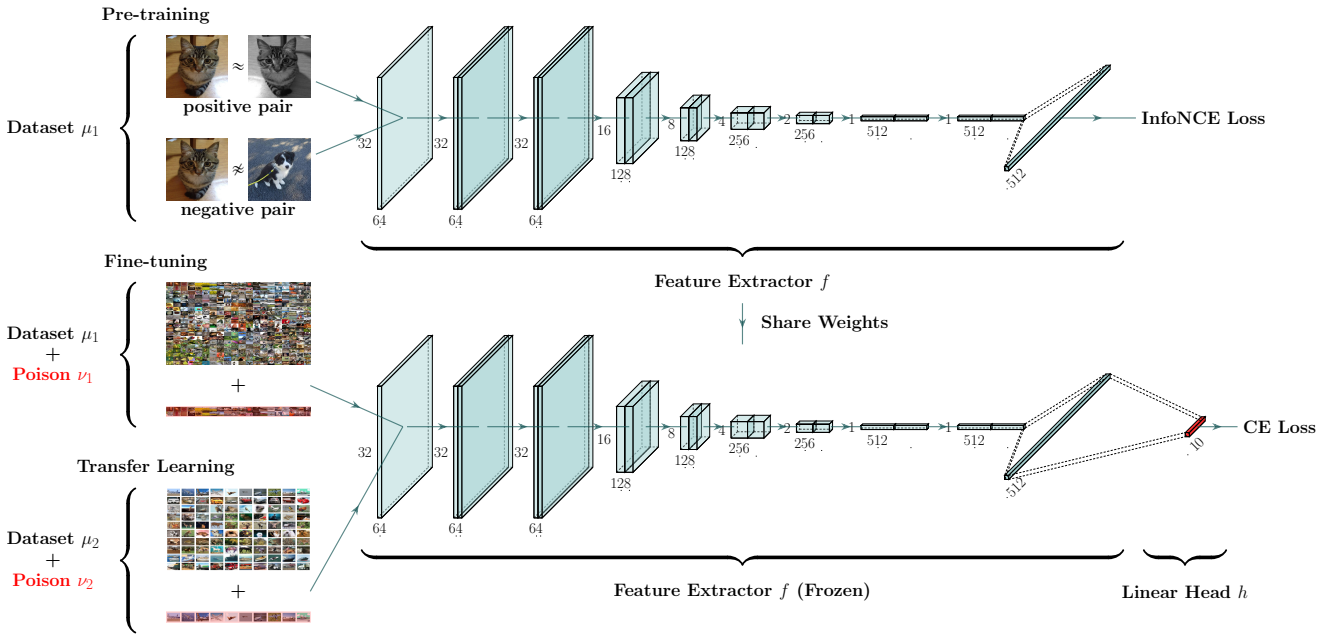

Figure 1. An illustration of our threat model:(top row) we acquire the weights of a feature extractor f with contrastive learning methods and optimizing w.r.t. the InfoNCE loss; (bottom row) we inject poisoned samples to the training dataset on downstream applications (image classification in this example) to perturb the linear head only. We examine two scenarios in this paper: (1) fine-tuning where pre-training and downstream tasks share the same training set; and (2) transfer learning where the downstream task is performed on a different dataset.

the GC or TGDA attack by optimizing towards a fixed encoder. For example, considering the classification problem and the GC attack (a model-targeted approach), we perform parameter corruption only on the linear head instead of the entire model to acquire target parameters.

In practice, without imposing any constraints on the poisoned samples, we observe that GC input space attack is indeed effective (e.g., induce 29.54% accuracy drop for fine-tuning and 39.35% for transfer learning with only $\epsilon_d = 3\%$ poisoning budget). However, such an attack can also be easily defended by data sanitization methods as the poisoned points have a much bigger magnitude and are visually malicious (see Figure 2). To generate "benign" poisoned images, we further pose a fidelity constraint during optimization and observe the effectiveness of the attacks is largely restricted (e.g., only 2.50% accuracy drop and 4.40% accuracy drop under the same experimental settings).

To improve the GC input space attack, we break down the problem into three stages to mitigate the optimization difficulties: (1) acquiring target parameters of the linear head; (2) obtaining poisoned features with GC feature-space attack by treating the learned feature embeddings $f(\mu)$ as the clean dataset, and directly constructing the poisoned feature set ζ, such that the linear evaluation is poor after retraining on $f(\mu)+\zeta$; (3) inverting the poisoned feature ζ back to the input space to construct ν that look visually similar to clean data, where we propose two viable approaches: decoder inversion that learns an inversion f^{-1} of the feature extractor, and feature matching that optimizes ν directly.

Finally, we examine another indiscriminate data poisoning attack of a different flavor, the Unlearnable Example (UE) attack. UE operates under the strong premise that an adversary is capable of perturbing the entire training set (i.e., the poisoning fraction $\epsilon_d = \infty$), albeit the perturbation of each point is relatively small.

In our empirical evaluation, we observe that downstream tasks with a fixed feature extractor are surprisingly robust to the popular UE approach of Huang et al. (2021).

In summary, we make the following contributions:

- We expose the threat of indiscriminate data poisoning attacks on pre-trained feature extractors and set baselines on fine-tuning and transfer learning downstream tasks, where transfer learning is generally more vulnerable to the considered attacks;
- We tailor existing attacks (e.g., TGDA, Gradient Canceling attack, and Unlearnable Examples) to poisoning fixed future extractors and empirically identify the scenarios where they succeed (e.g., without constraints) and fail (with constraints);
- We propose new attacks called the *feature targeted* (FT) attack that involves three stages to alleviate the challenge of optimizing input space attacks with constraints and observe empirical improvement.
- Finally, we examine unlearnable examples in the context of poisoning fixed feature extractors and identify that a popular UE attack becomes much less effective on the downstream tasks.

II. Background

In this section, we introduce the background on (1) data poisoning attacks, especially indiscriminate attacks; (2) self-

supervised learning (e.g., contrastive learning) methods, and relevant notations we use throughout the paper.

A. Data Poisoning attacks

Data poisoning attack is an emerging security concern regarding machine learning models. As the primary fuel of the thriving deep learning architectures, the data collection process depends highly on uncertified online data and is exposed to adversaries, who can actively inject corrupted data into dataset aggregators (e.g., online surveys, chatbots) or simply place poisoned samples online and passively waiting for scraping (Gao et al. 2020; Lyu et al. 2020; Shejwalkar et al. 2022; Wakefield 2016).

Formally, data poisoning attacks refer to the threat of crafting malicious training data such that machine learning models trained on it (possibly with clean training data) would return false predictions during inference. According to different objectives, there are roughly three types of data poisoning attacks: (1) targeted attacks (Aghakhani et al. 2021; Guo and Liu 2020; Shafahi et al. 2018; Zhu et al. 2019) aim at misclassifying a specific test sample; (2) backdoor attacks (Chen et al. 2017; Gu et al. 2017; Saha et al. 2020; Tran et al. 2018) that aim to misclassifying any test sample with a specific pattern; (3) indiscriminate attacks (Biggio et al. 2012; Koh and Liang 2017; Koh et al. 2022; Lu et al. 2022, 2023a; Muñoz-González et al. 2017) that decrease the overall test accuracy. In this paper, we focus on indiscriminate attacks.

Specifically, we will apply a SOTA indiscriminate attack method, namely the Gradient Canceling (GC) attack (Lu et al. 2023a) frequently in the later sections, and we introduce it briefly here in the traditional end-to-end training setting (along with the notations for indiscriminate attacks). Let $\ell((\mathbf{x}, y), \mathbf{w})$ be our loss that measures the cost of our model parameters $\mathbf{w}$ on a data sample $(\mathbf{x}, y)$ for supervised learning. Here we denote the (empirical) training distribution as μ, which contains a set of training samples. Indiscriminate attacks aim at constructing a poisoned (empirical) distribution ν, where $|\nu| = \epsilon_d|\mu|$, and ϵ_d is the poisoning fraction, such that $\mathbf{w}$ minimizes the loss ℓ by training over the mixed distribution $\chi \propto \mu + \nu$. Lu et al. (2023a) relax the optimality of the minimizer $\mathbf{w}$ to having vanishing (sub)gradients over the mixed distribution:

$$\nabla_{\mathbf{w}}\ell(\chi; \mathbf{w}) \propto \nabla_{\mathbf{w}}\ell(\mu; \mathbf{w}) + \nabla_{\mathbf{w}}\ell(\nu; \mathbf{w}) = \mathbf{0}, \tag{1}$$

Thus, given a good target parameter $\mathbf{w}$ (usually found by parameter corruption methods (Sun et al. 2020)), the GC attack simply solves the following problem:

$$\underset{\nu \in \Gamma}{\arg\min} \ \tfrac{1}{2}\|\nabla_{\mathbf{w}}\ell(\mu; \mathbf{w}) + \nabla_{\mathbf{w}}\ell(\nu; \mathbf{w})\|_2^2, \tag{2}$$

where Γ is the feasible set that can be specified according to the attack constraints (e.g., visual similarity to real samples to throttle defenses).

Indiscriminate attacks usually consider a scenario where the budget is small, i.e., the poisoning fraction ϵ_d is small, such as 3% of the training set size. Nevertheless, there exists a subclass of indiscriminate attacks called Unlearnable Examples (UE) (Fowl et al. 2021a,b; Fu et al. 2021; Huang et al. 2021; Liu and Chawla 2010; Sandoval-Segura et al. 2022; Yu et al. 2022) that operate under a much stronger assumption, where $\epsilon_d = \infty$, namely that the adversary can modify the entire training set directly. In this paper, we also consider this attack as the strongest threat possible to poisoning fixed feature extractors.

B. Contrastive Learning

Recently, self-supervised learning methods, in particular, contrastive learning methods have been widely applied to learn general representations without any label information for various downstream tasks. Existing approaches aim to learn a good feature extractor f (parameterized by a neural network) by minimizing the distance between representations of positive samples $f(\mathbf{x}^1)$ and $f(\mathbf{x}^2)$ (where $\mathbf{x}^1$ and $\mathbf{x}^2$ are different data augmentations of a training sample $\mathbf{x}$), while maximizing that of negative samples $f(\mathbf{x}^1)$ and $f(\mathbf{x}_n^1)$ (where $\mathbf{x}_n$ is another sample from the training set) at the same time. Existing approaches including Contrastive Predictive Coding (CPC) (Oord et al. 2018), SimCLR (Chen et al. 2020a,b), MoCo (He et al. 2020; Huynh et al. 2020) and f-MICL (Lu et al. 2023b) apply InfoNCE-based losses to enforce the contrast between positive and negative pairs. We call this process *contrastive learning pre-training*. Upon deployment, given a fixed feature extractor f, we consider two popular schemes of performing downstream tasks in this paper: (1) fine-tuning, where we apply the same (or slightly augmented) dataset used for pre-training and fine-tune an additional linear head on top of f (also known as linear evaluation). Such a linear head can be chosen to suit specific downstream tasks, e.g., classification, object detection, and semantic segmentation. In this paper, we consider classification as an example; (2) transfer learning, where we aim at training on a personalized dataset, which is usually smaller than the dataset used in pre-training (e.g., transfer from ImageNet to CIFAR-10). Again we consider linear evaluation where a linear head is learned.

III. Input Space Attacks

Existing indiscriminate attacks on neural networks, i.e., Total Gradient Descent Ascent (TGDA) attack (Lu et al. 2022), Gradient Canceling (GC) attack (Lu et al. 2023a) and Unlearnable Examples (Huang et al. 2021) generate (or optimize) poisoned points directly in the input space, which is suitable for end-to-end training. In this section, we discuss how to apply them to our problem setting, where part of the model parameters are fixed. Notably, we call this generic class *input space* attacks.

A. Problem Setting

We first specify our **threat model**:

- Knowledge of the attacker: the attacker has access to the training data, the target model (including its pre-trained weights), and the training procedure;
- Capability: the attacker can (actively or passively) inject a poisoned set to the training data;

Figure 2. We visualize some clean training samples of CIFAR-10 (which serve as initialization to the attacks) in the first row, and poisoned samples generated by GC input space attacks (which induce an accuracy drop of 29.54%) for $\epsilon_d = 0.03$ in the second row. The poisoned images show that GC input space attack generates images with no semantic meaning if no explicit constraints are posed. Clean images and their corresponding poisoned ones are chosen randomly.

- Objective: given a pre-trained feature extractor, the objective of a downstream task and the corresponding training dataset, an adversary aims to construct a poisoned set to augment the clean training set, such that by training on the mixed dataset, the performance of the downstream task on the test set is reduced.

Note that in this paper we only consider indiscriminate attacks. Another strong threat is the targeted attacks, where the knowledge of the attacker is expanded to some target test data, and the objective is to alter the prediction of them. The power of such attacks on pre-trained feature extractors has been studied and we refer interested readers to Shafahi et al. 2018.

We then formally introduce the problem of interest. Given a clean training distribution μ and target model architecture that consists of a (fixed) feature extractor f and a linear head h with learnable parameters $\boldsymbol{\omega}$, a standard ML pipeline aims to learn a set of clean parameters $\boldsymbol{\omega}$ to minimize its cost on μ while keeping f untouched:

$$\arg\min_{\boldsymbol{\omega}} \; l(f(\mu); \boldsymbol{\omega}). \tag{3}$$

An adversary thus considers designing a poisoning distribution ν to maximize the cost on a clean validation set $\tilde{\mu}$ (which comes from the same distribution as the training set μ):

$$\arg\max_{\nu \in \Gamma} \; l(f(\tilde{\mu}); \boldsymbol{\omega}). \tag{4}$$

As $\boldsymbol{\omega}$ is dynamically changing during optimization, solving Equation (4) is essentially a nested optimization problem (i.e., a non-zero-sum Stackelberg game):

$$\begin{aligned} &\max_{\nu \in \Gamma} \; l(f(\tilde{\mu}); \boldsymbol{\omega}_*), \\ &\text{s.t. } \boldsymbol{\omega}_* \in \arg\min_{\boldsymbol{\omega}} \; l(f(\mu) + f(\nu); \boldsymbol{\omega}), \end{aligned} \tag{5}$$

where the poisoning effectiveness is measured by retraining the linear head $\boldsymbol{\omega}$ on the mixed feature distribution $f(\mu) + f(\nu)$. Note that the attack budget is measured by the poisoning fraction $\epsilon_d = |\nu|/|\mu|$. Firstly, we introduce two viable approaches for solving the problem when ϵ_d is small (e.g., $< 100\%$), namely the TGDA attack (Lu et al. 2022) and GC attack (Lu et al. 2023a). Next, we discuss UE (Huang et al. 2021), where $\epsilon_d = \infty$, which amounts to substituting the entire training distribution μ as ν. Notably, we call these adaptations TGDA input space attack, GC input space attack and UE input space attack, respectively.

B. TGDA input space attack

Solving Equation (5) is non-trivial, especially for the outer maximization problem as the dependence of $l(f(\tilde{\mu}); \boldsymbol{\omega}_*)$ on ν is indirectly through retraining the head model on the mixed distribution. As a result, one can not directly apply simple algorithms such as gradient ascent as it would lead to zero gradients. Lu et al. (2022) propose to measure the influence of ν through the total derivative $\mathsf{D}_\nu l(f(\tilde{\mu}); \boldsymbol{\omega}) = -\nabla_{\boldsymbol{\omega}\nu} l_2 \cdot \nabla^{-1}_{\boldsymbol{\omega}\boldsymbol{\omega}} l_2 \cdot \nabla_{\boldsymbol{\omega}} l_1$ (where we simplify $l_1 = l(f(\tilde{\mu}), \boldsymbol{\omega}), l_2 = l(f(\mu) + f(\nu); \boldsymbol{\omega})$), which implicitly appraises the change of $\boldsymbol{\omega}$ with respect to ν. Specifically, TGDA considers the following total gradient ascent step for the attacker and gradient descent step for the defender:

$$\nu = \nu + \eta \cdot \mathsf{D}_\nu l(f(\tilde{\mu}); \boldsymbol{\omega}), \tag{6}$$

$$\boldsymbol{\omega} = \boldsymbol{\omega} - \eta \cdot \nabla_{\boldsymbol{\omega}} l(f(\mu) + f(\nu); \boldsymbol{\omega}). \tag{7}$$

For implementation, we still perform the attack end-to-end, namely that we directly feed the data into the model architecture, while keeping f untouched (e.g., `requires_grad=False` in PyTorch).

C. GC input space attack

Next, we generalize the GC attack (discussed in Section II) to our problem setting. Recall that to avoid solving the difficult Equation (5), GC aims to reach a specific target parameter $\hat{\boldsymbol{\omega}}$ that the model will converge to by retraining on the mixed feature distribution. Here reachability is defined by having a vanishing (sub)gradient over the mixed feature distribution:

$$\nabla_{\boldsymbol{\omega}} l(f(\chi); \hat{\boldsymbol{\omega}}) \propto \nabla_{\boldsymbol{\omega}} l(f(\mu); \hat{\boldsymbol{\omega}}) + \nabla_{\boldsymbol{\omega}} l(f(\nu); \hat{\boldsymbol{\omega}}) = \mathbf{0}. \tag{8}$$

Thus, GC input space attack amounts to solving the following problem:

$$\arg\min_{\nu \in \Gamma} \; \tfrac{1}{2} \|\nabla_{\boldsymbol{\omega}} l(f(\mu); \hat{\boldsymbol{\omega}}) + \nabla_{\boldsymbol{\omega}} l(f(\nu); \hat{\boldsymbol{\omega}})\|_2^2. \tag{9}$$

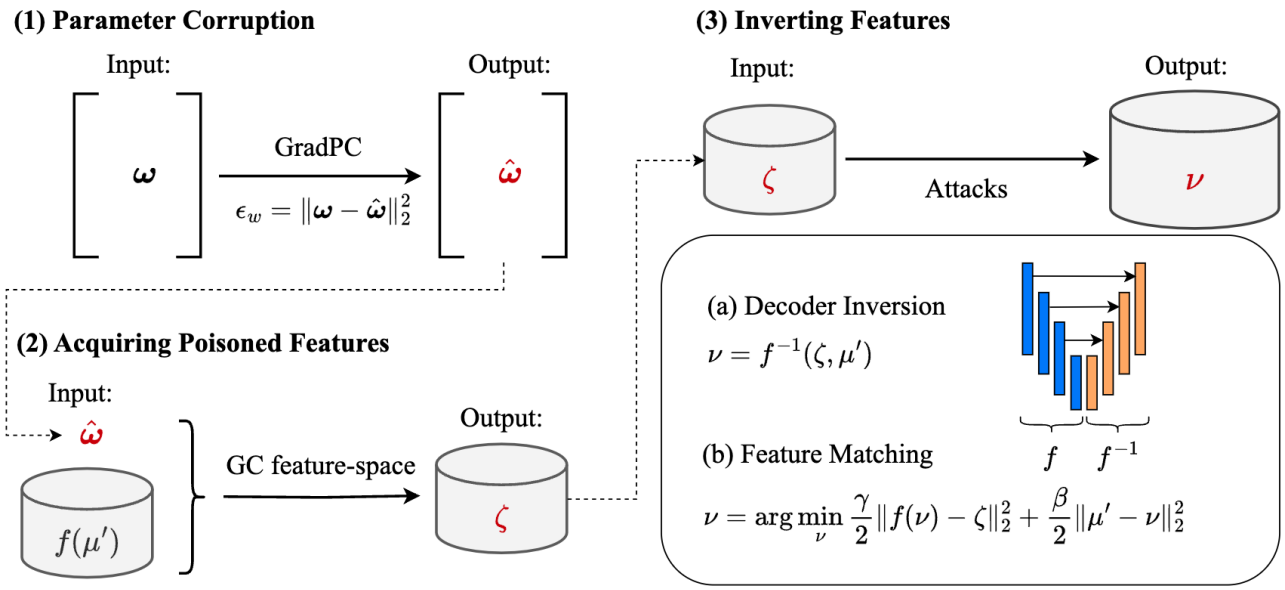

Figure 3. An illustration of the three stages of feature-targeted attacks: (1) obtaining the target linear head parameter $\boldsymbol{\omega}$ with GradPC; (2) acquiring poisoned features ζ with GC feature-space attack; (3) invert ζ back to the input space using feature matching or decoder inversion (decoder inversion requires training an autoencoder with a fixed encoder f).

Note that GC requires a target parameter $\hat{\boldsymbol{\omega}}$ that performs poorly on the test set, which Lu et al. (2023a) acquired by another attack that perturbs the clean model parameters $\boldsymbol{\omega}$ directly, namely the Gradient Parameter Corruption (GradPC) attack of Sun et al. (2020). In a nutshell, GradPC overwrites $\boldsymbol{\omega}$ with $\hat{\boldsymbol{\omega}}$ directly in our problem setting using a gradient approach. Note that the perturbation introduced by GradPC is measured by $\epsilon_w = \|\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}\|_2^2$ and can be specified as a constraint by the algorithm.

In practice, we observe that GC input space attack is very effective without considering any constraints on ν, e.g., it approximately reaches the target parameter $\hat{\boldsymbol{\omega}}$ for $\epsilon_d = 0.1$ in Table I and Table II, Section V. However, the poisoned samples generated by GC has a significantly larger magnitude than the clean distribution μ and does not retain semantic information (see Figure 2 for a visualization). To further generate "benign" poisoned points to throttle defenses, we add a penalty term $\frac{1}{2}\|\mu' - \nu\|_2^2$ in Equation (9) on constraining ν to be similar with μ', which is part of μ that initializes ν ($|\mu'| = |\nu|$).

However, we find the optimization procedure to be much harder with such a constraint, and the attack effectiveness is largely restricted. For example, we observe an accuracy drop of only 2.5% compared to 29.54% (without constraints) in Table I. To mitigate the difficulty of optimizations, we further propose feature targeted attacks in Section IV by breaking down the problem into several stages.

D. UE (EMN) input space attack

Before diving into Section IV, we introduce the last input space attack we consider, the Unlearnable Examples (UE), where we pose a strong assumption on $\epsilon_d = \infty$. We apply the error-minimizing noise (EMN) approach in Huang et al. (2021). Here the task is to perturb the entire training set with a noise factor δ such that the model learns a strong correlation between δ and the label information. Note that we consider δ to be a sample-wise noise, namely that $\mu + \delta = \{\mathbf{x}_i + \delta_i\}_{i=1}^{|\mu|}$. EMN input space attack optimizes the following objective:

$$\min_{\boldsymbol{\omega}} \min_{\delta \in \Gamma} l(f(\mu + \delta); \boldsymbol{\omega}). \tag{10}$$

Note that optimizing Equation (10) is substantially easier than Equation (5), as it is a min-min problem and can be easily solved by (alternating projected) gradient descent algorithms.

IV. Feature Targeted Attack

Motivated by the optimization challenge of GC input space attack, we propose a staged strategy to mitigate the difficulty of the constrained problem. Assuming we ① acquired the target parameter $\hat{\boldsymbol{\omega}}$ with parameter corruption, let us rewrite Equation (9) by introducing a feature distribution ζ:

$$\min_{\nu \in \Gamma} \min_{\zeta = f(\nu)} \frac{1}{2}\|\underbrace{\nabla_{\boldsymbol{\omega}} l(f(\mu); \hat{\boldsymbol{\omega}})}_{:=\mathbf{g}} + \nabla_{\boldsymbol{\omega}} l(\zeta; \hat{\boldsymbol{\omega}})\|_2^2. \tag{11}$$

Next, we relax both constraints by introducing penalty terms[3]:

$$\min_{\nu, \zeta} \underbrace{\frac{1}{2}\|\mathbf{g} + \nabla_{\boldsymbol{\omega}} l(\zeta; \hat{\boldsymbol{\omega}})\|_2^2}_{②} + \underbrace{\frac{\gamma}{2}\|f(\nu) - \zeta\|_2^2 + \frac{\beta}{2}\|\mu' - \nu\|_2^2}_{③}. \tag{12}$$

The variables ν and ζ are now separated, which immediately suggests an alternating algorithm that we elaborate on below.

We follow the above three steps and break down the optimization into corresponding three stages: ①: acquiring the target parameter $\hat{\boldsymbol{\omega}}$ with GradPC; solving ② with GC feature space attack, where we treat the learned representation as the clean dataset and inject poisoned features directly; solving ③ where we invert the poisoned features back to the input

[3]Other penalty terms other than the (squared) L_2 norm could also be used.

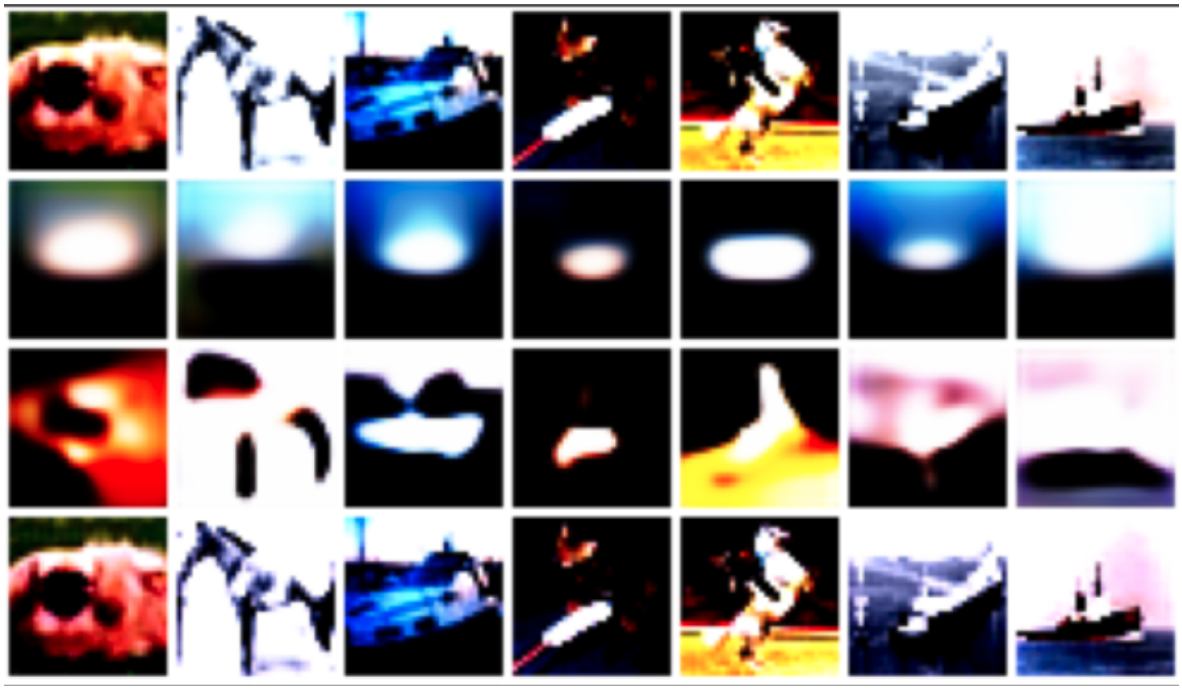

Figure 4. Here we visualize original test images (first row), and images reconstructed by an autoencoder with fixed ResNet-18 feature extractor learned by MoCo (second row), the same autoencoder trained end-to-end (third row), and the autoencoder with skip connections, i.e., a U-Net (fourth row).

space considering constraints in the input space, where we introduce two possible approaches: decoder inversion (without optimization) and feature matching (with optimization). Overall, we call this approach *Feature Targeted* (FT) attack and we visualize the attack procedures in Figure 3.

A. GC feature space attack

Firstly, we generate poisoned features ζ using the GC attack. We chose GC as it is the best-performing algorithm in the feature space. We consider $f(\mu)$ as the clean feature training set and enable an attacker to inject poisoned features ζ directly. Following the notations in Section III, GC considers reachability in the feature space:

$$\nabla_{\boldsymbol{\omega}} l(f(\mu); \hat{\boldsymbol{\omega}}) + \nabla_{\boldsymbol{\omega}} l(\zeta; \hat{\boldsymbol{\omega}}) = \mathbf{0}, \tag{13}$$

and optimizes the following problem:

$$\arg\min_{\zeta} \ \tfrac{1}{2} \|\nabla_{\boldsymbol{\omega}} l(f(\mu); \hat{\boldsymbol{\omega}}) + \nabla_{\boldsymbol{\omega}} l(\zeta; \hat{\boldsymbol{\omega}})\|_2^2, \tag{14}$$

which corresponds to the term ② in (12). We denote ζ as the output of the GC feature space attack and apply it as input to the next stage.

Here the target parameter $\hat{\boldsymbol{\omega}}$ is generated by ① GradPC from the feature space, where we treat $f(\mu)$ as the training set and perturb the linear head h. Unlike poisoning attacks, such a change does not make a significant difference for GradPC as it perturbs the linear head parameters directly.

B. Inverting features

Next, we show two viable approaches to solve ③ on inverting the poisoned feature back to the input space.

1) Decoder Inversion: First, we show how to solve ③ without optimizing ν directly by training a decoder network. Specifically, given a pre-trained feature encoder f that maps μ to $f(\mu)$, we aim to learn its inversion f^{-1} such that $f^{-1}(f(\mu)) \approx \mu$. For image reconstruction and related tasks (e.g., image inpainting, image-to-image translation, semantic segmentation), it is common to utilize an autoencoder architecture, where the encoder is f and the decoder is naturally f^{-1}. However, such an autoencoder is usually trained end-to-end, while we require training only the decoder f^{-1} with a fixed encoder f. In Figure 4 (rows 1-3), we apply a ResNet-18 encoder learned by MoCo on CIFAR-10 and train the corresponding decoder on reconstructing CIFAR-10. We show such a constraint indeed brings a challenge: the autoencoder with a fixed f does not learn the detailed structure of images compared to that trained end-to-end, which may imply that an encoder trained by contrastive learning is not necessarily a good encoder for image reconstruction. Additionally, we observe that a plain autoencoder does not reconstruct the images well, even when training end-to-end.

Previous works on image reconstruction tasks show that a key success of the architectural design is to add skip connections between encoders and decoders, also known as U-Net (Ronneberger et al. 2015). In Figure 4 (row 4), we show that adding skip connections leads to superb reconstruction quality, even with a fixed encoder architecture.

Upon acquiring a good decoder f^{-1}, we can then invert the poisoned feature ζ back to the input space. Note that with a U-Net architecture, f^{-1} does not only take ζ as input but also

the intermediate features of μ' [4], which we denote as $f_1^n(\mu')$, where n is the number of skip connections. In summary, the output of the decoder inversion is $\nu = f^{-1}(\zeta, f_1^n(\mu'))$, where the first term in ③ is minimized if the decoder is well-trained, and the second term is automatically handled by the skip connections.

2) Feature Matching: In decoder inversion, we train the decoder f^{-1} and hoping the acquired ν is a successful inversion, i.e., $f(\nu) = \zeta$. However, in practice, we find the distance between the two is still large, e.g., $\|f(\nu) - \zeta\|_2^2 \approx 150$, which indicates the gap between the decoder inversion and the implicit ground truth is still large (another indication is the performance gap between GC feature space attack and decoder inversion attack in Section V).

To further close the gap between $f(\nu)$ and ζ, we introduce an alternative method that directly optimizes the poisoned dataset ν by matching its feature $f(\nu)$ to the target poisoned feature ζ while enforcing the distance between μ and ν to be small. We call this algorithm *Feature Matching*.

Our feature matching algorithm is inspired by the feature collision (FC) algorithm (Shafahi et al. 2018) in targeted attack. In targeted attacks, FC considers a target test sample t in the test set and a different base instance b. FC aims to construct a single poisoned data sample $\mathbf{x}_p$ such that the distance between $f(t)$ and $f(\mathbf{x}_p)$ in the feature space is small while $\mathbf{x}_p$ is close to b in the input space.

When adapting to our setting, instead of matching the feature of a target sample, we match the target feature ζ that we already acquired. Thus, we need only optimize the term ③ in (12):

$$\arg\min_{\nu} \ \tfrac{\gamma}{2}\|f(\nu) - \zeta\|_2^2 + \tfrac{\beta}{2}\|\mu' - \nu\|_2^2, \tag{15}$$

where recall that the second term is a relaxation of the constraint $\nu \in \Gamma$. We present our algorithm in Algorithm 1, where we follow Shafahi et al. (2018) and apply a forward gradient step to minimize the first term on feature matching and a backward proximal step to enforce the constraint on the input space. Note that β is a hyperparameter for controlling the strength of the constraint.

Again, μ' is part of the training distribution where $|\mu'| = \epsilon_d|\mu| = |\zeta|$. Instead of choosing μ' from the initialization of ζ, as in decoder inversion, we follow the following better strategy. The key observation is that in practice, ζ is not necessarily close to the initial corresponding $\mu' = \nu_0$, thus making the initial optimization hard. In other words, we find it necessary to explore whether the choice of μ' can be optimized in order to find a better matching in the input space. A simple solution is to perform a grid search for individual poisoned features: for every $\mathbf{z} \in \zeta$, we calculate the point-wise distance $\|f(\mathbf{x}) - \mathbf{z}\|_2^2$ for every $\mathbf{x} \in \mu$. By ranking the distances, we simply choose the one with the smallest distance to form a pair $\{\mathbf{x}, \mathbf{z}\}$. Such a grid search provides a much better initialization and significantly reduces the running time upon convergence. Moreover, we find that such a process reduces the difficulty of optimization and helps the algorithm converge at a lower value.

[4]Here μ' refers to the clean samples corresponding to $f(\mu')$, which are the initialization of the GC feature space attack.

Algorithm 1: Feature Matching

Input: base instances as part of the training distribution $\mu' = \epsilon_d\mu$, step size η, a pre-trained encoder f
Output: poisoned dataset ν

$\nu \leftarrow \mu'$ `// initialize poisoned data`
$\zeta \leftarrow f(\nu)$ `// initialize poisoned feature`
for $k = 1, ...$ **do**
  for $s = 1, 2, ...$ **do**
    $\zeta \leftarrow \zeta - \eta \cdot \nabla_\zeta[\frac{1}{2}\|\mathbf{g} + \nabla_{\boldsymbol{\omega}} l(\zeta; \hat{\boldsymbol{\omega}})\|_2^2]$
    $\zeta \leftarrow [\zeta + \eta\gamma f(\nu)]/(1 + \eta\gamma)$ `// optional`
  for $t = 1, 2, ...$ **do**
    $\nu \leftarrow \nu - \eta \cdot \nabla_\nu[\frac{\gamma}{2}\|f(\nu) - \zeta\|_2^2]$ `// forward`
    $\nu \leftarrow (\nu + \eta\beta\mu')/(1 + \eta\beta)$ `// backward`

V. Experiments

In this section, we present our experimental results on the proposed indiscriminate attacks. For a fixed feature extractor, we consider two approaches to performing supervised downstream tasks: (1) fine-tuning, where the contrastive learning pre-training and fine-tuning share the same training distribution (the difference lies on without/with label information); (2) transfer learning, where pre-training is performed on a (relatively) large-scale dataset (without label information), and the linear head is trained on a smaller or customized dataset with labels.

Our experiments will thus be structured as these two major parts, where we consider input space attacks and feature space attacks for each class.

A. Experimental setting

Hardware and Package: Fine-tuning experiments are run on a cluster with NIVIDIA T4 and P100 GPUs, while transfer learning experiments are run on another machine with 2 NIVIDIA 4090 GPUs. The platform we use is `PyTorch`.

Datasets: We consider image classification as the downstream task on the CIFAR-10 dataset (Krizhevsky 2009) (50k training and 10k testing images). For contrastive learning pre-training, we use CIFAR-10 and ImageNet 1K (Deng et al. 2009) for fine-tuning and transfer learning respectively.

Feature Extractors: Following the standard contrastive learning pre-training architectures, we examine the ResNet-18 (He et al. 2016) model with a feature dimension of size 512 for pre-training on CIFAR-10, following the common recipe of changing the convolution kernel size from 7×7 to 3×3. The model weights are obtained by training on SimCLR (Chen et al. 2020a) for 800 epochs with a batch size of 512. For clean linear evaluation, we acquire an accuracy of 89.71% with 100 epochs. For ImageNet pre-training we apply the ResNet-50 architecture using MoCo V3 (Chen et al. 2021) with a feature dimension of size 2048 and a batch size of 4096, trained for 1000 epochs. We directly utilize the pre-trained weights provided in the

Table I
FINE-TUNING RESULTS: COMPARISON BETWEEN INPUT SPACE ATTACKS (TGDA AND GC INPUT SPACE ATTACKS WITH/WITHOUT CONSTRAINTS ON POISONED SAMPLES) AND FEATURE TARGETED ATTACKS (DECODER INVERSION AND FEATURE MATCHING WITH DIFFERENT β) W.R.T. DIFFERENT ϵ_d. THE WHITE BACKGROUND INDICATES METHODS THAT GENERATE TARGET PARAMETERS OR FEATURES; BLUE INDICATES INPUT SPACE ATTACKS AND PINK INDICATES FEATURE TARGETED ATTACKS.

Attack method	Clean	$\epsilon_d = 0.03$ (1500)	$\epsilon_d = 0.1$ (5000)	$\epsilon_d = 0.5$ (25000)	$\epsilon_d = 1$ (50000)
GradPC (input space)	89.71%	53.15% / -36.56%	53.15% / -36.56%	53.15% / -36.56%	53.15% / -36.56%
TGDA-input (no constraints)	89.71%	85.43% / - 4.28%	84.26% / - 5.45%	76.55% / -13.16%	70.38% / -19.33%
TGDA-input (with constraints)	89.71%	87.52% / - 2.19%	87.45% / - 2.26%	83.67% / - 6.04%	78.66% / -11.05%
GC-input (no constraints)	89.71%	60.17% / -29.54%	53.65% / -36.06%	53.15% / -36.56%	53.15% / -36.56%
GC-input (with constraints)	89.71%	87.21% / - 2.50%	87.05% / - 2.66%	80.30% / - 9.41%	75.33% / -14.38%
GradPC (feature space)	89.71%	53.00% / -36.71%	53.00% / -36.71%	53.00% / -36.71%	53.00% / -36.71%
GC-feature space	89.71%	53.10% / -36.61%	53.03% / -36.68%	54.09% / -35.62%	52.15% / -37.56%
Decoder inversion	89.71%	85.49% / - 4.22%	84.61% / - 5.10%	78.65% / -11.06%	72.25% / -17.46%
Feature matching ($\beta = 0.25$)	89.71%	83.41% / - 6.30%	82.33% / - 7.38%	76.15% / -13.56%	69.17% / -20.54%
Feature matching ($\beta = 0.1$)	89.71%	77.24% / -12.44%	76.56% / -13.15%	73.24% / -15.47%	65.14% / -24.57%
Feature matching ($\beta = 0.05$)	89.71%	75.34% / -14.37%	74.29% / -15.42%	71.99% / -17.72%	63.19% / -26.52%

MoCo V3 Github Repo[5]. Upon transfer learning, we follow the general recipe to resize the CIFAR-10 training sample from 32×32 to 224×224 and acquire an accuracy of 93.03% by training the linear head for 100 epochs.

Attacks implementations: We specify the implementations of all attacks we apply in this paper:

(1) TGDA input space attack: we follow and modify the implementation of Lu et al. (2022)[6] while fixing the feature extractor of the follower (defender) and train for 200 epochs for all tasks;

(2) GC input space/feature space attack: we follow and modify the official code of Lu et al. (2023a)[7] and perform 2000 epochs or early stop upon reaching stopping criteria (loss smaller than 1). Note that for acquiring target parameters, we run GradPC[8] for one single attack step and set $\epsilon_w = 1$;

(3) UE input space attacks: we follow and modify the implementation of Huang et al. (2021)[9] for sample-wise error minimizing noise generation;

(4) Decoder inversion: for the architectural design on the decoder, we use the implementation of this Github repo[10] as a key reference. For image reconstruction, we apply a combination of the multi-scale structural similarity index (MS-SSIM) loss (Wang et al. 2003) and L_1 loss following the analysis of Khare et al. 2021;

(5) Feature matching: finally, we follow Algorithm 1 and set $k = 1$, $t = 2000$. We set the learning rate as 0.1 with a cosine scheduler. Unless specified, $\gamma = 1, \beta = 0.25$ across all tasks.

Label Information: During optimizations of indiscriminate attacks, we only modify the input $\mathbf{x}$ and generate clean-label poisoned samples. Specifically, we assign the label corresponding to the clean images during the initialization stage of every attack. Note that for feature matching, as we perform ranking after the GC feature space attack, we reassign the labels as that of clean base samples to every ζ, thus the output ν.

Evaluation Protocol: To evaluate the performance of each attack, we first acquire the poisoning dataset ν and retrain the model (initialized with the same random seed across all attacks) on both clean and poisoned data $\mu + \nu$ until convergence (we train for 100 epochs across all tasks). Our evaluation metric is the test accuracy drop, compared with clean accuracy (obtained by training on clean data only).

B. Fine-tuning

We report our main results for poisoning the fine-tuning downstream task in Table I. We expand discussions and our key observations in the following paragraphs for input space attacks and feature-targeted attacks, respectively.

1) Input space attacks: Recall in Section III, we introduce the TGDA input space attack and the GC input space attack. We first examine the two attacks without considering any constraints, such that the poisoned points ν could be randomly large and do not need to resemble the real distribution μ. Although under this setting the poisoned samples can be easily detected, it is still valuable to study as it reveals the security threats without any data sanitization process, which is not uncommon in practice. From Table I (upper part, row 1,2,4), we observe that compared to TGDA, GC is very effective, where it can almost reach the target parameter generated by GradPC when $\epsilon_d = 0.1$.

However, GC generates poisoned samples that have a much larger magnitude than the clean samples, especially when ϵ_d is small (e.g., $100\times$ larger for $\epsilon_d = 0.03$ and decreases to $5\times$ larger for $\epsilon_d = 1$). This also matches the observations in Lu et al. (2023a): a higher ϵ_d is desired for an attacker as it not only increases the attack effectiveness but also brings the poisoned points closer to the clean distribution, at least magnitude-wise.

Additionally, Lu et al. (2023a) (Figure 10) shows that the GC attack does not change the semantics of the clean samples

[5]https://dl.fbaipublicfiles.com/moco-v3/r-50-1000ep/r-50-1000ep.pth.tar
[6]https://github.com/watml/TGDA-Attack
[7]https://github.com/watml/plim
[8]https://github.com/TobiasLee/ParamCorruption
[9]https://github.com/HanxunH/Unlearnable-Examples
[10]https://github.com/mkisantal/backboned-unet

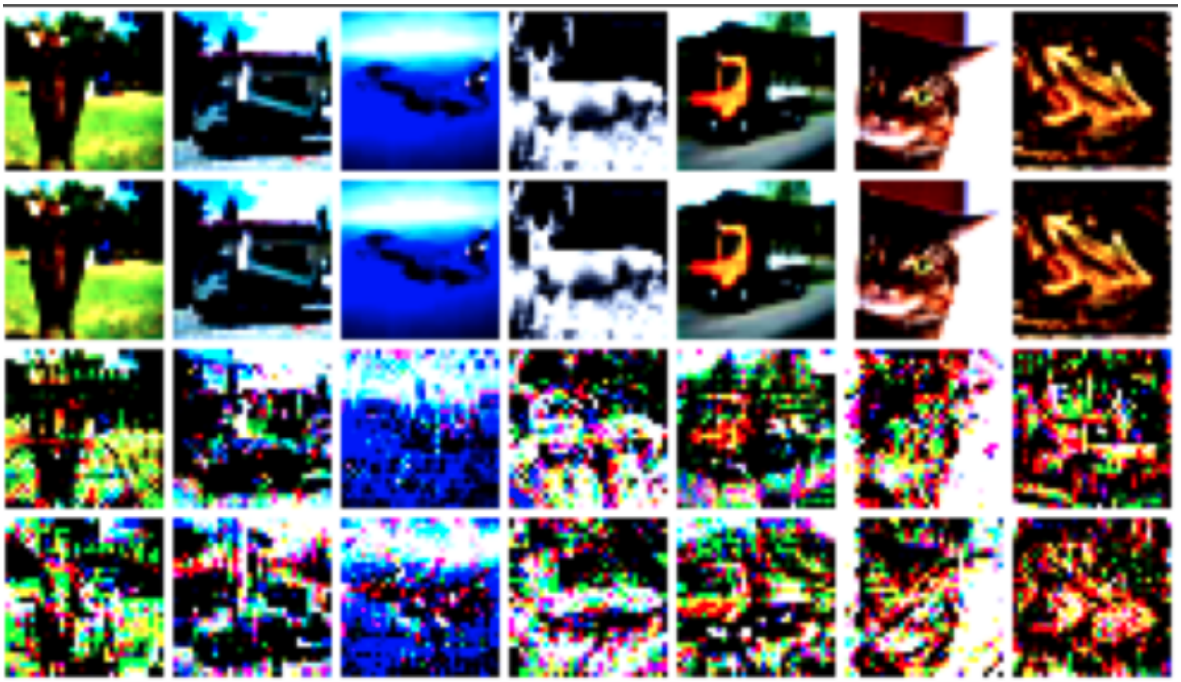

Figure 5. Here we visualize clean images (first row), and poisoned samples returned by the feature matching algorithm with $\beta = 0.25, 0.1, 0.05$ respectively from the second to the fourth row.

much, even without explicit constraints for end-to-end training. However, we show it is not the case for a fixed feature extractor in Figure 2. Instead, GC input space attack returns noisy samples that lose all information of the input sample.

Based on the above two observations, we further add two constraints to the two attacks to make them less vulnerable to possible defenses: (1) clipping the poisoned samples to the range of clean samples; (2) adding a term on measuring the (squared) L_2 distance between input (clean sample as initialization) and output (poisoned points) specified in Section III. In Table I (upper part, row 3,5), we find the attacks are much less effective, especially the GC attack, and may not be considered a strong threat. Compared with poisoning end-to-end training in Lu et al. (2023a), our poisoned samples on fixed feature extractors are much weaker with constraints, e.g., Lu et al. (2023a) report the test accuracy drop decreases $\approx 10\%$ for $\epsilon_d = 0.03$, which is significantly less than ours (29.54% (no constraints) - 2.50% (with constraints) = 27.04%). This confirms that with a fixed feature extractor, it is much more difficult to construct feasible poisoning samples than end-to-end training for input space attacks, which motivates us to design and examine stronger attacks.

2) Feature targeted attacks: Next, we discuss two feature targeted (FT) attacks introduced in Section IV, namely the decoder inversion and the feature matching attack. Both of the FT attacks involve three stages: (1) GradPC feature space attack for acquiring the target parameters $\hat{\boldsymbol{\omega}}$, where we achieve roughly the same performance as that in input space; (2) GC feature space attack for obtaining the target feature ζ; (3) taking ζ as the input, we either invert the feature back directly using a decoder or perform feature matching.

Note that we do not treat GC feature space attack as a viable attack (as injecting poisoned features may not be realistic), but as an essential building block and intermediate step for decoder inversion and feature matching. Additionally, our following results for GC feature space attack provide important information on how good a poisoned feature ζ can be (in terms of attack performance) and the difficulty of inverting them back to the input space.

In Table I (lower part, rows 1-2), we report GradPC (feature space) for stage 1 and GC-feature space for stage 2. We observe that the GC feature space attack is very successful in reproducing the target parameters $\boldsymbol{\omega}$, even when $\epsilon_d = 0.03$. Additionally, comparing with the GC input space attack, we observe that GC feature space induces a much smaller perturbation in terms of magnitude, e.g., the poisoned features are $4\times$ larger than the normal ones for $\epsilon_d = 0.03$ and lie within the same range for $\epsilon_d \geq 0.2$.

Surprisingly, although GC feature space attack obtains the target feature ζ with relatively small perturbations, it is difficult to invert it back to the input space. For decoder inversion (row 3), although we observe that the images are in the normal range and visually benign (see Figure 4), it is much less effective than the target poisoned features. In practice, we find the poisoned samples are largely dominated by the skip connections of the encoder architecture, and thus the constraint on the input space could be too strong and cannot be easily controlled.

In comparison, the feature matching algorithm (rows 4-6) is more effective than decoder inversion, i.e., for $\beta = 0.25$, we observe a further accuracy drop $\approx 2\%$ for every choice of ϵ_d. Moreover, by controlling the hyperparameter β, we can easily tune the constraints on the input similarity, where a smaller β

Table II

TRANSFER LEARNING RESULTS: COMPARISON BETWEEN INPUT SPACE ATTACKS (TGDA AND GC INPUT SPACE ATTACKS WITH/WITHOUT CONSTRAINTS ON POISONED SAMPLES) AND FEATURE TARGETED ATTACKS (DECODER INVERSION AND FEATURE MATCHING WITH DIFFERENT β) W.R.T. DIFFERENT ϵ_d. THE WHITE BACKGROUND INDICATES METHODS THAT GENERATE TARGET PARAMETERS OR FEATURES; BLUE INDICATES INPUT SPACE ATTACKS AND PINK INDICATES FEATURE TARGETED ATTACKS.

Attack method	Clean	$\epsilon_d = 0.03$ (1500)	$\epsilon_d = 0.1$ (5000)	$\epsilon_d = 0.5$ (25000)	$\epsilon_d = 1$ (50000)
GradPC (input space)	93.03%	49.39% / -43.64%	49.39% / -43.64%	49.39% / -43.64%	49.39% / -43.64%
TGDA-input (no constraints)	93.03%	87.84% / - 5.19%	86.97% / - 6.06%	78.74% / -14.29%	72.93% / -20.10%
TGDA-input (with constraints)	93.03%	89.71% / - 3.32%	89.44% / - 3.59%	85.25% / - 7.78%	79.97% / -13.06%
GC-input (no constraints)	93.03%	53.68% / -39.35%	52.84% / -40.19%	50.04% / -42.99%	49.48% / -43.55%
GC-input (with constraints)	93.03%	88.63% / - 4.40%	87.44% / - 5.59%	79.52% / - 13.51%	76.05% / -16.98%
GradPC (feature space)	93.03%	49.53% / -43.50%	49.53% / -43.50%	49.53% / -43.50%	49.53% / -43.50%
GC-feature space	93.03%	50.01% / -43.02%	50.01% / -43.02%	50.01% / -43.02%	50.01% / -43.02%
Decoder inversion	93.03%	87.92% / - 5.11%	86.96% / - 6.07%	79.27% / -13.76%	74.06% / -18.97%
Feature matching ($\beta = 0.25$)	93.03%	71.50% / -21.53%	65.04% / -27.99%	57.93% / -35.10%	50.69% / -42.34%
Feature matching ($\beta = 0.1$)	93.03%	59.58% / -33.45%	57.74% / -35.29%	50.74% / -42.29%	49.63% / -43.40%

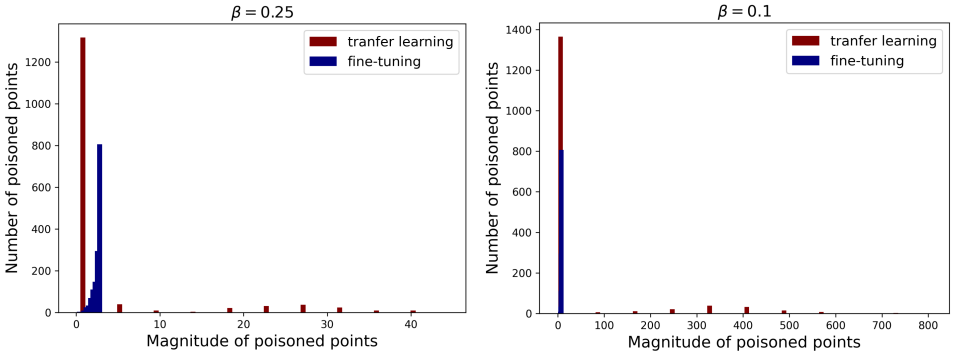

Figure 6. The histogram of the (L_∞) magnitude of 1500 ($\epsilon_d = 0.03$) poisoned points generated by feature matching feature matching on transfer learning (red) and fine-tuning (blue) with $\epsilon_d = 0.03$ for $\beta = 0.25$ (left), $\beta = 0.1$ (right). The plot shows that while for fine-tuning the poisoned points are roughly in the same range, for transfer learning, $\approx 10\%$ of the poisoned samples are anomalies.

indicates less power of constraints. In Table I, we observe that a smaller β generally returns a much stronger attack. However, tuning β also involves a trade-off: in Figure 5, we visualize the poisoned images generated by feature matching attacks with different β and observe that a smaller β generally introduce more noise, and poisoned images generated by $\beta = 0.1, 0.05$ can be easily discriminated by human eyes. Additionally, smaller β induce poisoned images with a larger magnitude, i.e., for $\beta = 0.25, 0.1, 0.05$, we observe the magnitude of $1\times, 3\times, 5.5\times$ the normal range, respectively.

Moreover, we compare the best-performing input space attack (GC-input) and feature targeted attack (feature matching). Recall that both of these methods aim to solve the same constrained problem in Equation (9), where GC-input directly the constrained gradient canceling problem while the latter approach mitigates the difficulty of the constrained problem with the staged strategy in Equation (12). We observe that the feature matching attack indeed helps with the optimization procedure and improves the attack effectiveness by a big margin (range from 3.80% to 6.16% for different ϵ_d).

In summary, we find both of the feature targeted attacks consistently outperform input space attacks across all choices of ϵ_d. However, for indiscriminate attacks in general, inverting poisoned features back to the input space attack is rather difficult, and cannot be achieved exactly with our existing attacks.

C. Transfer Learning

Next, we present our experimental results for the task of transfer learning. Recall that for transfer learning, we take the feature extractor (ResNet-50) pre-trained on ImageNet without labels and perform a linear evaluation on the CIFAR-10 dataset with labels. We report our main results in Table II and expand our discussions as follows.

1) Input space attacks: Firstly we perform the same input space attacks on the transfer learning task. In Table II (upper part), we observe: (1) In general, input space attacks are more effective (an increased accuracy drop $\approx$ 2%-10%) on transfer learning than on fine-tuning tasks. A large factor could be that transfer learning involves a distribution shift, such that the linear evaluation process is more vulnerable to poisoning attacks; (2) without any constraints, GC input space still outperforms TGDA input space. Meanwhile, GC generates poisoned samples that have a larger magnitude than that of fine-tuning, i.e., about $290\times$ larger for $\epsilon_d = 0.03$ (fine-tuning: $100\times$ larger) than clean samples, and reduces to $14\times$ larger for $\epsilon_d = 1$ (fine-tuning: $5\times$ larger). As for poisoned samples, TGDA and GC input space attacks still generate noisy images that have no semantic meanings (analogous to that in Figure 2); (3) again

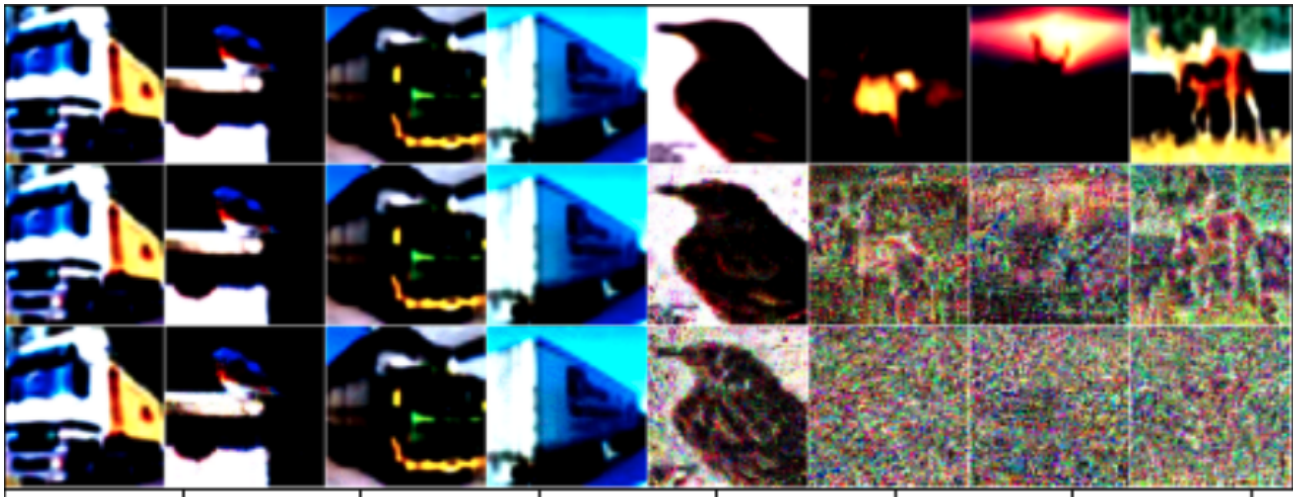

Figure 7. We plot poisoned images generated by feature matching on transfer learning for $\epsilon_d = 0.03$. We show clean images and poisoned images for $\beta = 0.25$ and $\beta = 0.1$ on rows 1,2,3 respectively. Columns 1-4 are images within the normal range, and columns 5-8 are abnormal poisoned images with a bigger magnitude.

Table III
COMPARISON BETWEEN FEATURE MATCHING AND FEATURE MATCHING AFTER REMOVING ANOMALY POINTS (FEATURE MATCHING-R).

Attack method	Clean	$\epsilon_d = 0.03$
Feature matching ($\beta = 0.25$)	93.03%	71.50% / -21.53%
Feature matching-r ($\beta = 0.25$)	93.03%	85.82% / - 7.21%
Feature matching ($\beta = 0.10$)	93.03%	59.58% / -33.45%
Feature matching-r ($\beta = 0.10$)	93.03%	86.38% / - 6.65%

we add the same constraints on magnitude and visual similarity with the input images and observe that the attack performance significantly drops.

2) Feature targeted attacks: We perform the same feature targeted attacks and report the results in Table II (lower part). For the feature matching attacks, we omit the results for $\beta = 0.05$ as it does not show a substantial improvement over $\beta = 0.1$. We observe that in general, similar to input space attacks, the attacks are more effective on transfer learning tasks than on fine-tuning. Additionally, decoder inversion and feature matching still outperform all input space attacks considering constraints across all choices of ϵ_d.

Notably, upon further exploration, we observe that feature matching behaves differently in transfer learning. Recall that for fine-tuning when we choose $\beta = 0.25$, the magnitude of poisoned samples is within the normal range, and for $\beta = 0.1$, the magnitude is generally $3\times$. However, for transfer learning, the algorithm returns several abnormal images with a larger magnitude and more noise. In Figure 6, we randomly select 100 poisoned samples (out of 1500 for $\epsilon_d = 0.03$) and plot their L_∞ magnitude. For the entire poisoned set, we observe that for $\beta = 0.25$ and $\beta = 0.1$, there are 181 (12.1%) and 274 (18.2%) anomalies, respectively. Additionally, when β is smaller, while most of the poisoned points are in the normal range, the anomalies have a much bigger magnitude, due to the power of the constraint being weakened. In Figure 7, we compare poisoned images within the normal range (columns 1-4) and abnormal ones (columns 5-9). We observe that abnormal poisoned images also contain more noise, especially when β is smaller. To examine the real effect of feature matching attacks, we perform a simple data sanitization process by removing all of the abnormal points before evaluation. We report the results in Table III and observe the attack becomes much less effective after the defense. Nevertheless, feature matching-r ($\beta = 0.25$) still outperforms other baseline attacks.

Table IV
COMPARISON BETWEEN UE (EMN) IN THE CONTEXT OF POISONING A FIXED FEATURE EXTRACTOR WITH LINEAR HEAD AND TRAINING END-TO-END (WITH THE SAME INITIALIZATION). WE REPORT THE RESULTS ON BOTH FINE-TUNING AND TRANSFER LEARNING.

Task	**Attack**	Clean	$\epsilon_d = \infty$
Fine-tuning	Original	94.77%	19.93% / -74.84%
	End-to-end	95.22%	68.79% / -26.43%
	Input space	89.71%	82.16% / - 7.55%
Transfer Learning	Original	94.42%	18.89% / -75.53%
	End-to-end	95.09%	66.05% / -29.04%
	Input space	93.03%	85.15% / - 7.88%

Overall, compared to fine-tuning, we find transfer learning easier to poison for every attack, with or without constraints in the input space. However, inverting poisoned features back to input space, either explicitly through feature targeted attacks or implicitly through input space attacks, is still difficult in the condition of considering poisoned points in a feasible set. Specifically, we find that no existing attack can exactly reproduce the target parameters (features) with a limited number of visually benign poisoned samples.

D. Unlearnable Examples

Finally, we discuss the unlearnable examples of poisoning fixed feature extractors (the EMN input space attack). We apply the attack of Huang et al. (2021) and report the results in Table IV. We compare three variants of EMN: (1) original: we consider the original setting in Huang et al. (2021), where during the attack and evaluation, the entire model is updated

Figure 8. We plot poisoned images generated by EMN input space attack. We show clean images and poisoned images for fine-tuning (which incurs an accuracy drop of 7.55%) and transfer learning (incurring an accuracy drop of 7.88%) on rows 1,2,3, respectively.

with random initialization on the unlearnable dataset[11]; (2) end-to-end: the model is initialized with the pre-trained feature extractor, but all parameters can be updated; (3) input space: we consider our setting where the feature extractor is fixed during training and evaluation. We visualize the poisoned images generated by EMN in Figure 8.

From Table IV we obtain several interesting observations: (1) EMN is highly sensitive to weights initialization: by comparing original and end-to-end, we show that random initialization is much more vulnerable to EMN than a good initialization (i.e., pre-trained weights by contrastive learning); (2) EMN input space is not very effective: with a significantly larger poisoning budget $\epsilon_d = \infty$, EMN only performs similar to feature matching when $\epsilon_d = 0.03$. Compared with original and end-to-end, EMN input space is also less than a magnitude effective for both fine-tuning and transfer learning.

In summary, our observations reveal that training a neural network architecture with a good feature extractor as initialization, especially when it is frozen, is much more robust to the EMN attack.

VI. Conclusions and Limitations

In this work, we study indiscriminate data poisoning attacks on pre-trained feature extractors with different budgets, from injecting a small amount of poisoned data to perturbing the entire training set. Specifically, we consider the model parameters acquired by contrastive learning methods and poisoning the downstream task with a trainable linear head and the fixed feature extractor. Based on the attack scheme, we propose two types of indiscriminate attacks: (1) input space attacks, including TGDA, GC, and UE input space attacks, where the attacker perturbs the feature representations indirectly through the input space, and (2) feature-targeted attacks, including decoder inversion and feature matching attacks, where an adversary acquires poisoned features and inverts them back to the input space.

We further examine our attacks on popular downstream tasks of fine-tuning and transfer learning. Our experimental results reveal that for injecting limited numbers of poisoned samples, input space attacks generate out-of-distribution poisoned samples without constraints and become less effective after projection. On the other hand, feature targeted attacks, especially feature matching attacks exhibit an improvement with a tunable trade-off between the legitimacy of the poisoned samples and their effectiveness. Moreover, considering a more challenging domain adaptation scenario, transfer learning is more vulnerable than fine-tuning on these attacks in general. Additionally, empirical results on unlearnable examples demonstrate that EMN is much less effective than it performs on end-to-end training with random initialization.

Limitations and Future directions: In this paper, we set baselines on indiscriminate attacks on pre-trained feature extractors and we believe it opens several interesting future directions: (1) Understanding the difficulty of feature inversion: prior works on adversarial examples show that small perturbation in the input space could lead to dramatic changes in the feature space for a neural network with large Lipschitz constant. In this paper, we show that a relatively small perturbation in the feature space cannot be easily inverted to the input space with constraints. Such observations might be counterintuitive and surprising, and deserve more study; (2) Possible defenses against Unlearnable Examples: although UE, e.g., EMN poses a strong indiscriminate attack, its countermeasure has not been explored. Notably, data sanitization cannot be easily performed as the entire training distribution is altered. In this paper, we provide evidence that UE is fragile against fixed feature extractors and a well-trained initialization. Based on this, there might exist a natural defense mechanism against UE, by carefully designing the architecture or the training schemes, e.g., adding a fixed encoder for any task of interest.

[11]The results are directly obtained from Table 1 in Huang et al. (2021), CIFAR-10 ResNet-18/50.

Acknowledgments

We thank the reviewers and program chairs for the critical comments that have largely improved the presentation and precision of this paper. We gratefully acknowledge funding support from NSERC and the Canada CIFAR AI Chairs program. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.

References

Aghakhani, H., D. Meng, Y.-X. Wang, C. Kruegel, and G. Vigna (2021). "Bullseye polytope: A scalable clean-label poisoning attack with improved transferability". In: *IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 159–178.

Balcan, M.-F., A. Blum, S. Hanneke, and D. Sharma (2022). "Robustly-reliable learners under poisoning attacks". In: *Proceedings of Thirty Fifth Conference on Learning Theory*, pp. 4498–4534.

Bejnordi, B. E. et al. (2017). "Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer". *Jama*, vol. 318, no. 22, pp. 2199–2210.

Biggio, B., B. Nelson, and P. Laskov (2012). "Poisoning attacks against support vector machines". In: *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pp. 1467–1474.

Carlini, N. and A. Terzis (2021). "Poisoning and Backdooring Contrastive Learning". In: *International Conference on Learning Representations*.

Chen, T., S. Kornblith, M. Norouzi, and G. Hinton (2020a). "A Simple Framework for Contrastive Learning of Visual Representations". In: *ICML*.

Chen, T., S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton (2020b). "Big Self-Supervised Models are Strong Semi-Supervised Learners". *NeurIPS*.

Chen, X., S. Xie, and K. He (2021). "An empirical study of training self-supervised vision transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9640–9649.

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

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). "ImageNet: A large-scale hierarchical image database". In: *CVPR*. IEEE.

Dosovitskiy, A. et al. (2020). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*.

Fowl, L., P.-y. Chiang, M. Goldblum, J. Geiping, A. Bansal, W. Czaja, and T. Goldstein (2021a). "Preventing unauthorized use of proprietary data: Poisoning for secure dataset release". arXiv preprint arXiv:2103.02683.

Fowl, L., M. Goldblum, P.-y. Chiang, J. Geiping, W. Czaja, and T. Goldstein (2021b). "Adversarial Examples Make Strong Poisons". In: *Advances in Neural Information Processing Systems*, pp. 30339–30351.

Fu, S., F. He, Y. Liu, L. Shen, and D. Tao (2021). "Robust unlearnable examples: Protecting data privacy against adversarial learning". In: *International Conference on Learning Representations*.

Gao, L. et al. (2020). "The Pile: An 800GB Dataset of Diverse Text for Language Modeling". arXiv preprint arXiv:2101.00027.

Gu, T., B. Dolan-Gavitt, and S. Garg (2017). "Badnets: Identifying vulnerabilities in the machine learning model supply chain". arXiv:1708.06733.

Guo, J. and C. Liu (2020). "Practical Poisoning Attacks on Neural Networks". In: *European Conference on Computer Vision*, pp. 142–158.

He, H., K. Zha, and D. Katabi (2022). "Indiscriminate Poisoning Attacks on Unsupervised Contrastive Learning". In: *The Eleventh International Conference on Learning Representations*.

He, K., H. Fan, Y. Wu, S. Xie, and R. Girshick (2020). "Momentum contrast for unsupervised visual representation learning". In: *CVPR*.

He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

Huang, H., X. Ma, S. M. Erfani, J. Bailey, and Y. Wang (2021). "Unlearnable Examples: Making Personal Data Unexploitable". In: *International Conference on Learning Representations*.

Huynh, T., S. Kornblith, M. R. Walter, M. Maire, and M. Khademi (2020). "Boosting Contrastive Self-Supervised Learning with False Negative Cancellation". *arXiv preprint arXiv:2011.11765*.

Khare, N., P. S. Thakur, P. Khanna, and A. Ojha (2021). "Analysis of Loss Functions for Image Reconstruction Using Convolutional Autoencoder". In: *International Conference on*

Computer Vision and Image Processing. Springer, pp. 338–349.

Koh, P. W. and P. Liang (2017). "Understanding black-box predictions via influence functions". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 1885–1894.

Koh, P. W., J. Steinhardt, and P. Liang (2022). "Stronger Data Poisoning Attacks Break Data Sanitization Defenses". *Machine Learning*, vol. 111, pp. 1–47.

Krizhevsky, A. (2009). "Learning multiple layers of features from tiny images". tech. report.

Kumar, R. S. S., M. Nyström, J. Lambert, A. Marshall, M. Goertzel, A. Comissoneru, M. Swann, and S. Xia (2020). "Adversarial machine learning-industry perspectives". In: *IEEE Security and Privacy Workshops (SPW)*, pp. 69–75.

Liu, W. and S. Chawla (2010). "Mining adversarial patterns via regularized loss minimization". *Machine learning*, vol. 81, no. 1, pp. 69–83.

Lu, Y., G. Kamath, and Y. Yu (2022). "Indiscriminate Data Poisoning Attacks on Neural Networks". *Transactions on Machine Learning Research*.

– (2023a). "Exploring the Limits of Model-Targeted Indiscriminate Data Poisoning Attacks". In: *Proceedings of the 40th International Conference on Machine Learning*.

Lu, Y., G. Zhang, S. Sun, H. Guo, and Y. Yu (2023b). "f-MICL: Understanding and Generalizing InfoNCE-based Contrastive Learning". *Transactions on Machine Learning Research*.

Lyu, L., H. Yu, and Q. Yang (2020). "Threats to federated learning: A survey". arXiv preprint arXiv:2003.02133.

Muñoz-González, L., B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli (2017). "Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization". In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec)*, pp. 27–38.

Nelson, B., M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia (2008). "Exploiting machine learning to subvert your spam filter." *LEET*, vol. 8, pp. 1–9.

Oord, A. van den, Y. Li, and O. Vinyals (2018). "Representation learning with contrastive predictive coding". *arXiv preprint arXiv:1807.03748*.

Radford, A. et al. (2021). "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PMLR, pp. 8748–8763.

Ronneberger, O., P. Fischer, and T. Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, pp. 234–241.

Saha, A., A. Subramanya, and H. Pirsiavash (2020). "Hidden trigger backdoor attacks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*.

Sandoval-Segura, P., V. Singla, J. Geiping, M. Goldblum, T. Goldstein, and D. W. Jacobs (2022). "Autoregressive Perturbations for Data Poisoning". In: *Advances in Neural Information Processing Systems*.

Shafahi, A., W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein (2018). "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6103–6113.

Shejwalkar, V., A. Houmansadr, P. Kairouz, and D. Ramage (2022). "Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Production Federated Learning". In: *IEEE Symposium on Security and Privacy (SP)*, pp. 1354–1371.

Sun, X., Z. Zhang, X. Ren, R. Luo, and L. Li (2020). "Exploring the vulnerability of deep neural networks: A study of parameter corruption". In: *Proceedings of the AAAI Conference on Artificial Intelligence*.

Suya, F., S. Mahloujifar, A. Suri, D. Evans, and Y. Tian (2021). "Model-targeted poisoning attacks with provable convergence". In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 10000–10010.

Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus (2014). "Intriguing properties of neural networks". In: International Conference on Learning Representation.

Tran, B., J. Li, and A. Madry (2018). "Spectral Signatures in Backdoor Attacks". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Truong, T., S. Mohammadi, and M. Lenga (2021). "How transferable are self-supervised features in medical image classification tasks?" In: *Machine Learning for Health*. PMLR, pp. 54–74.

Veeling, B. S., J. Linmans, J. Winkens, T. Cohen, and M. Welling (2018). "Rotation equivariant CNNs for digital pathology". In: *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International*

Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II 11. Springer, pp. 210–218.

Wakefield, J. (2016). "Microsoft chatbot is taught to swear on Twitter". *BBC News*.

Wang, Z., E. P. Simoncelli, and A. C. Bovik (2003). "Multiscale structural similarity for image quality assessment". In: *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*. Vol. 2. Ieee, pp. 1398–1402.

Yu, D., H. Zhang, W. Chen, J. Yin, and T.-Y. Liu (2022). "Availability Attacks Create Shortcuts". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2367–2376.

Zhu, C., W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein (2019). "Transferable clean-label poisoning attacks on deep neural nets". In: *International Conference on Machine Learning*, pp. 7614–7623.

APPENDIX

A. Perturbations introduced by feature matching

Recall in Figure 5, we visualized the poisoned images generated by feature matching attacks with different β. Here in Figure 9 we also visualize the difference between the clean images and the poisoned images to show the perturbation.

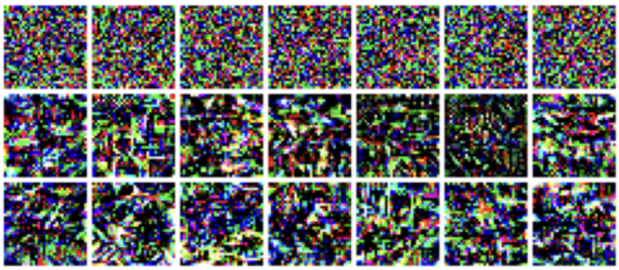

Figure 9. Here we visualize perturbations introduced by the feature matching algorithm with $\beta = 0.25, 0.1, 0.05$ respectively from the first to the third row.

B. Additional results on transfer learning

1) Vision transformer as encoder: Aside from the ResNet architecture, another popular backbone architecture serving as a feature extractor is the recently advanced vision transformers (Dosovitskiy et al. 2020). Here we extend our experiments of poisoning transfer learning on pre-trained vision transformers (specifically, ViT-base) with MoCo V3 (Chen et al. 2021) on ImageNet to CIFAR-10. We report our results in Table V and observe that our conclusion still holds for the ViT architecture.

2) Image encoder trained with CLIP: In this paper, we introduce contrastive learning methods with one modality, i.e., images. Another popular approach to learning feature extractors considers both images and texts, also known as contrastive language-image pre-training (CLIP) (Radford et al. 2021). Instead of choosing different augmentations of the same image as positive pairs and different images as negative pairs, CLIP considers labeled image-text pairs as positive ones, and unmatched pairs as negative ones. Specifically, CLIP trains an image encoder and a text encoder simultaneously, which can be used separately for downstream tasks. Here we examine data poisoning attacks against the image encoder trained by CLIP with a ResNet-50 architecture. We directly adopt the pre-trained weights acquired by OpenAI [12]. We report our result in Table VI and observe that feature matching still outperforms GC input space attack (with constraints), even after removing anomaly points.

3) Transferring to CIFAR-100: In Table II, we show the results of transferring a pre-trained feature extractor (trained on ImageNet) to the CIFAR-10 dataset. As CIFAR-10 only involves 10 classes, the classification problem is rather easy. Here we extend the downstream task to a more difficult setting with 100 classes, i.e. the CIFAR-100 dataset (Krizhevsky 2009). Specifically, CIFAR-100 consists of 100 classes containing 600 images each, with 500 training images and 100 testing images respectively. Again for ImageNet pre-training we apply the ResNet-50 architecture using MoCo V3 (Chen et al. 2021) and examine the performance of data poisoning attacks against transferring to CIFAR-100. We report the results in Table VII and observe that: (1) transferring to CIFAR-100 is more challenging than CIFAR-10 (79.90% clean accuracy compared to CIFAR-10, 93.03%); (2) the attacks are generally weaker under a fine-grained classification setup; (3) feature matching-r still outperform GC input space (with constraints).

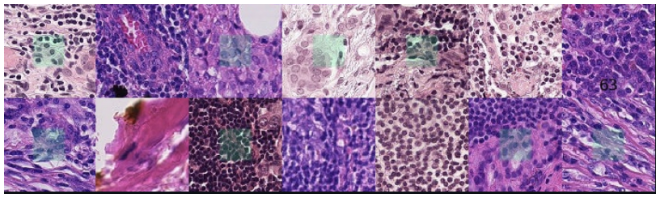

Figure 10. Example images from the PatchCamelyon dataset.

4) Transferring to PatchCamelyon: Next we consider a dataset with a more drastic domain shift from the pre-training dataset, i.e., ImageNet. Here we choose the PatchCamelyon dataset (Bejnordi et al. 2017; Veeling et al. 2018), which consists of 327680 (220025 training samples) color images of size 96×96, extracted from histopathologic scans of lymph node selection. Each image is annotated with a binary label indicating the presence of metastatic tissue. In Figure 10, we visualize some samples in the dataset. Again, we choose the feature extractor pre-trained by MoCo V3 (Chen et al. 2021) (ResNet-50) on ImageNet and examine the performance of data poisoning attacks against transferring to PatchCamelyon. Note that we follow (Truong et al. 2021) and randomly choose 5000 samples from the original dataset for the downstream training. We report our results in Table VIII and observe that transferring to PatchCamelyon is generally more vulnerable to data poisoning attacks (higher accuracy drop) than CIFAR-10/100, which may indicate that a drastic domain shift is more fragile against such attacks.

[12] https://github.com/openai/CLIP

Table V
TRANSFER LEARNING RESULTS ON VISION TRANSFORMERS: COMPARISON BETWEEN INPUT SPACE ATTACKS (GC INPUT SPACE ATTACKS WITH/WITHOUT CONSTRAINTS ON POISONED SAMPLES) AND FEATURE MATCHING AFTER REMOVING ANOMALY POINTS (FEATURE MATCHING-R) W.R.T. DIFFERENT ϵ_d.

Attack method	Clean	$\epsilon_d = 0.03$ (1500)	$\epsilon_d = 0.1$ (5000)	$\epsilon_d = 0.5$ (25000)	$\epsilon_d = 1$ (50000)
GradPC (input space)	98.90%	-46.53%	-46.53%	-46.53%	-46.53%
GC-input (no constraints)	98.90%	-40.12%	-42.30%	-44.66%	-46.49%
GC-input (with constraints)	98.90%	- 4.28%	- 5.99%	- 13.12%	-17.66%
GradPC (feature space)	98.90%	-46.53%	-46.53%	-46.53%	-46.53%
GC-feature space	98.90%	-46.51%	-46.51%	-46.51%	-46.51%
Feature matching-r ($\beta = 0.25$)	98.90%	- 6.33%	- 7.89%	- 15.04%	-19.27%

Table VI
TRANSFER LEARNING RESULTS ON CLIP IMAGE ENCODER: COMPARISON BETWEEN INPUT SPACE ATTACKS (GC INPUT SPACE ATTACKS WITH/WITHOUT CONSTRAINTS ON POISONED SAMPLES) AND FEATURE MATCHING AFTER REMOVING ANOMALY POINTS (FEATURE MATCHING-R).

Attack method	Clean	$\epsilon_d = 0.03$ (1500)
GradPC (input space)	92.30%	-44.55%
GC-input (no constraints)	92.30%	-41.20%
GC-input (with constraints)	92.30%	- 4.39%
GradPC (feature space)	92.30%	-44.51%
GC-feature space	92.30%	-42.99%
Feature matching-r ($\beta = 0.25$)	92.30%	- 7.05%

Table VII
TRANSFER LEARNING RESULTS ON CIFAR-100: COMPARISON BETWEEN INPUT SPACE ATTACKS (GC INPUT SPACE ATTACKS WITH/WITHOUT CONSTRAINTS ON POISONED SAMPLES) AND FEATURE MATCHING AFTER REMOVING ANOMALY POINTS (FEATURE MATCHING-R).

Attack method	Clean	$\epsilon_d = 0.03$ (1500)
GradPC (input space)	79.90%	-38.51%
GC-input (no constraints)	79.90%	-36.78%
GC-input (with constraints)	79.90%	- 3.29%
GradPC (feature space)	79.90%	-38.50%
GC-feature space	79.90%	-37.99%
Feature matching-r ($\beta = 0.25$)	79.90%	- 6.69%

Table VIII
TRANSFER LEARNING RESULTS ON PATCHCAMELYON: COMPARISON BETWEEN INPUT SPACE ATTACKS (GC INPUT SPACE ATTACKS WITH/WITHOUT CONSTRAINTS ON POISONED SAMPLES) AND FEATURE MATCHING AFTER REMOVING ANOMALY POINTS (FEATURE MATCHING-R).

Attack method	Clean	$\epsilon_d = 0.03$ (150)
GradPC (input space)	79.12%	-50.13%
GC-input (no constraints)	79.12%	-49.05%
GC-input (with constraints)	79.12%	-11.03%
GradPC (feature space)	79.12%	-50.12%
GC-feature space	79.12%	-50.00%
Feature matching-r ($\beta = 0.25$)	79.12%	-15.33%