Data Poisoning based Backdoor Attacks to Contrastive Learning

Jinghuai Zhang[1] Hongbin Liu[2] Jinyuan Jia[3] Neil Zhenqiang Gong[2]
University of California, Los Angeles[1] Duke University[2] Penn State[3]
jinghuai1998@g.ucla.edu
{hongbin.liu, neil.gong}@duke.edu
jinyuan@psu.edu

Abstract

*Contrastive learning (CL) pre-trains general-purpose encoders using an unlabeled pre-training dataset, which consists of images or image-text pairs. CL is vulnerable to data poisoning based backdoor attacks (DPBAs), in which an attacker injects poisoned inputs into the pre-training dataset so the encoder is backdoored. However, existing DPBAs achieve limited effectiveness. In this work, we take the first step to analyze the limitations of existing backdoor attacks and propose new DPBAs called **CorruptEncoder** to CL. CorruptEncoder introduces a new attack strategy to create poisoned inputs and uses a theory-guided method to maximize attack effectiveness. Our experiments show that CorruptEncoder substantially outperforms existing DPBAs. In particular, CorruptEncoder is the first DPBA that achieves **more than 90%** attack success rates with only a few (3) reference images and a small poisoning ratio (0.5%). Moreover, we also propose a defense, called localized cropping, to defend against DPBAs. Our results show that our defense can reduce the effectiveness of DPBAs, but it sacrifices the utility of the encoder, highlighting the need for new defenses.*

1. Introduction

Given an unlabeled pre-training dataset, contrastive learning (CL) [2, 3, 5, 23] aims to pre-train an image encoder and (optionally) a text encoder via leveraging the supervisory signals in the dataset itself. For instance, given a large amount of unlabeled images, single-modal CL, which is the major focus of this paper, [1] can learn an image encoder that produces similar (or dissimilar) feature vectors for two random augmented views created from the same (or different) image. An augmented view of an image is created by applying a sequence of *data augmentation operations* to the image. Among various data augmentation operations, *random cropping* is the most important one [3].

CL is vulnerable to *data poisoning based backdoor attacks (DPBAs)* [1, 25]. Specifically, an attacker embeds backdoor into an encoder via injecting *poisoned images* into the pre-training dataset. A downstream classifier built based on a backdoored encoder predicts an attacker-chosen class (called *target class*) for any image embedded with an attacker-chosen *trigger*, but its predictions for images without the trigger are unaffected.

However, existing DPBAs achieve limited effectiveness. In particular, SSL-Backdoor [25] proposes to craft a poisoned image by embedding the trigger directly into an image from the target class. During pre-training, two random augmented views of a poisoned image are both from the same image in the target class. As a result, the backdoored encoder fails to build strong correlations between the trigger and images in the target class, leading to suboptimal results. Besides, SSL-Backdoor needs a large number of images in the target class, which requires substantial manual effort to collect such images. While PoisonedEncoder [17] uses fewer such images to achieve an improved attack performance on simple datasets, its effectiveness is limited when applied to more complex datasets (e.g., ImageNet). The limitation arises from the absence of a theoretical analysis that guides the optimization of feature similarity between a small trigger and objects in the target class. Another line of work (CTRL [14]) improves stealthiness by embedding an invisible trigger into the frequency domain. However, its effectiveness is sensitive to the magnitude of the trigger and the attack remains ineffective on a large dataset.

Our work: In this work, we propose *CorruptEncoder*[2], a new DPBA to CL. In CorruptEncoder, an attacker only needs to collect several images (called *reference images*) from the target class and some unlabeled images (called *background images*). **Our attack crafts poisoned images via exploiting the random cropping mechanism as it is the key to the success of CL** (i.e., the encoder's utility sacrifices substantially without random cropping as shown in

[1] We extend CorruptEncoder to multi-modal CL in Section 6.

[2] https://github.com/jzhang538/CorruptEncoder

Figure 1. Reference image (left) vs. reference object (right).

Table 4 "No Random Cropping"). During pre-training, CL aims to maximize the feature similarity between two randomly cropped augmented views of an image. Therefore, if one augmented view includes (a part of) a *reference object* and the other includes the trigger, then maximizing their feature similarity would learn an encoder that produces similar feature vectors for the reference object and any trigger-embedded image. Therefore, a downstream classifier would predict the same class (i.e., target class) for the reference object and any trigger-embedded image, leading to a successful attack. To this end, CorruptEncoder introduces a new strategy to create a poisoned image as follows: 1) randomly sample a reference object and a background image, 2) rescale or crop the background image if needed, 3) embed the reference object and the trigger into the background image at certain locations. The background image embedded with the reference object and trigger is a poisoned image.

The *key insights* of crafting poisoned inputs via embedding reference object and trigger into random background images are *three-folds*. **(1)** We only need a few images from the target class for the attack. **(2)** Embedding reference object (instead of the reference image) into different background images can avoid maximizing the feature similarity between the trigger and the same background in the reference image (e.g., gray area in Figure 1). **(3)** We can control the size (i.e., width and height) of the background image, the location of the reference object in the background image, and the location of the trigger, to explicitly optimize the attack effectiveness. In particular, when the probability that two randomly cropped views of a poisoned image respectively only include the reference object and trigger is larger, CorruptEncoder is more effective. In this work, we theoretically derive the optimal size of the background image and optimal locations of the reference object and trigger that can maximize such probability. In other words, we craft optimal poisoned images in a theory-guided manner.

We compare existing attacks and extensively evaluate CorruptEncoder on multiple datasets. In particular, we pre-train 220+ image/image-text encoders under distinct attack settings. Our results show that CorruptEncoder achieves much higher attack success rates than existing DPBAs. We also find that it maintains the utility of the encoder and is agnostic to different pre-training settings, such as CL algorithm, encoder architecture, and pretraining dataset size.

We also explore a defense against DPBAs. Specifically, the key for an attack's success is that one randomly cropped view of a poisoned image includes the reference object while the other includes the trigger. Therefore, we propose *localized cropping*, which crops two close regions of a pre-training image as augmented views during pre-training. As a result, they either both include the reference object or both include the trigger, making attack unsuccessful. Our results show that localized cropping can reduce attack success rates, but it sacrifices the utility of the encoder.

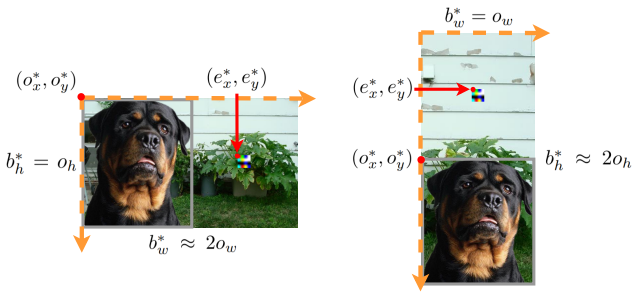
2. Threat Model

Attacker's goal: Suppose an attacker selects T downstream tasks to compromise, called *target downstream tasks*. For each target downstream task t, the attacker picks s_t target classes, where $t = 1, 2, \cdots, T$. We denote by y_{ti} the ith target class for the tth target downstream task. For each target class y_{ti}, the attacker selects a trigger e_{ti}. The attacker aims to inject a poisoned dataset $\mathcal{D}_p$ into a pre-training dataset $\mathcal{D}$ such that the learnt, backdoored image encoder achieves two goals: *effectiveness goal* and *utility goal*. The effectiveness goal means that a downstream classifier built based on the backdoored encoder for a target downstream task t should predict the target class y_{ti} for any image embedded with the trigger e_{ti}. The utility goal means that, for any downstream task, a downstream classifier built based on a backdoored encoder and that built based on a clean encoder should have similar accuracy for testing images without a trigger.

Attacker's capability and background knowledge: We assume the attacker can inject N poisoned images ($|\mathcal{D}_p| = N$) into the pre-training dataset $\mathcal{D}$. The provider often collects an unlabeled pre-training dataset from the Internet. Therefore, the attacker can post its poisoned images on the Internet, which could be collected by a provider as a part of its pre-training dataset. Moreover, we assume the attacker has access to 1) a small number (e.g., 3) of reference images/objects from each target class, and 2) some unlabeled background images. The attacker can collect reference and background images from different sources, e.g., the Internet. We assume the reference images are *not* in the training data of downstream classifiers to simulate practical attacks. Moreover, we assume the attacker does not know the pre-training settings and can not manipulate the pre-training process. It is noted that previous DPBAs [14, 25] use several hundreds of reference images to launch their attacks, while we assume the attacker has only a small number (e.g., 3) of reference objects for a stronger attack.

3. CorruptEncoder

Our key idea is to craft poisoned images such that the image encoder learnt based on the poisoned pre-training dataset produces similar feature vectors for any image embedded with a trigger e_{ti} and a reference object in the target class



(a) Left-right layout (b) Bottom-top layout

Figure 2. Illustration of the optimal size (b_w^*, b_h^*) of the background image and optimal locations ((o_x^*, o_y^*) and (e_x^*, e_y^*)) of the reference object and trigger in the background image when crafting a poisoned image.

y_{ti}. Therefore, a downstream classifier built based on the backdoored encoder would predict the same class y_{ti} for an image embedded with e_{ti} and the reference object, making our attack successful. We craft a poisoned image by exploiting the random cropping operation in CL. Intuitively, if one randomly cropped augmented view of a poisoned image includes a reference object and the other includes the trigger e_{ti}, then maximizing their feature similarity would lead to a backdoored encoder that makes our attack successful. Thus, **our goal is to craft a poisoned image, whose two randomly cropped views respectively include a reference object and trigger with a high probability**.

Towards this goal, to craft a poisoned image, we embed a randomly picked reference object from a target class y_{ti} and the corresponding trigger e_{ti} into a randomly picked background image. Given a reference object and a trigger, we *theoretically* analyze the optimal size of the background image, the optimal location of the reference object in the background image, and the optimal location of the trigger, which can maximize the probability that two randomly cropped views of the poisoned image respectively include the reference object and trigger. Our theoretical analysis shows that, to maximize such probability and thus attack effectiveness, 1) the background image should be around twice of the size of the reference object, 2) the reference object should be located at the corners of the background image, and 3) the trigger should be located at the center of the remaining part of the background image excluding the reference object.

3.1. Crafting Poisoned Images

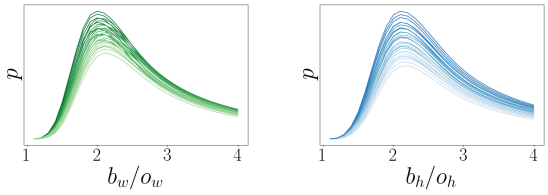
We denote by $\mathcal{O}$, $\mathcal{B}$, and $\mathcal{E}$ the set of reference objects, background images, and triggers, respectively. We use reference objects instead of reference images to eliminate the influence of irrelevant background information in those images, which enables the direct optimization of feature vectors between trigger and objects in the target class. To craft a poisoned image, we randomly pick a reference object $o \in \mathcal{O}$ and a background image $b \in \mathcal{B}$; and $e \in \mathcal{E}$ is the trigger corresponding to the target class of o. If the background image b is too small (or large), we re-scale (or crop) it. In particular, we re-scale/crop the background image such that the width ratio (or height ratio) between the background image and the reference object is α (or β). Then, we embed the reference object into the background image at location (o_x, o_y) and embed the trigger into it at location (e_x, e_y) to obtain a poisoned image, where the trigger does not intersect with the reference object. Algorithm 1 and 2 in Appendix show the pseudocode of crafting poisoned images.

Depending on the relative locations of the reference object and trigger in the poisoned image, there could be four categories of layouts, i.e., *left-right*, *right-left*, *bottom-top* and *top-bottom*. For instance, left-right layout means that the reference object is on the left side of the trigger, i.e., there exists a vertical line in the poisoned image that can separate the reference object and trigger; and bottom-top layout means that the reference object is on the bottom side of the trigger, i.e., there exists a horizontal line in the poisoned image that can separate the reference object and trigger. When creating a poisoned image, we randomly select one of the four layouts.

3.2. Theoretical Analysis

Given a reference object o and a trigger e, our CorruptEncoder has three key parameters when crafting a poisoned image: 1) size of the background image, 2) location of the reference object, and 3) location of the trigger. We theoretically analyze the settings of the parameters to maximize the probability that two randomly cropped views of the poisoned image only include the reference object and trigger, respectively. Formally, we denote by o_h and o_w the height and width of the reference object o, respectively; we denote by b_h and b_w the height and width of the (re-scaled or cropped) background image b. Moreover, we denote $\alpha = b_w/o_w$ and $\beta = b_h/o_h$. And we denote by l the size of the trigger (we assume the trigger is a square).

Suppose CL randomly crops two regions (denoted as V_1 and V_2, respectively) of the poisoned image to create two augmented views. To simplify the illustration, we assume the regions are squares and they have the same size s (the theorem still holds if the two views do not have the same size). We denote by $p_1(s)$ the probability that V_1 is within the reference object o but does not intersect with the trigger e, and we denote by $p_2(s)$ the probability that V_2 includes the trigger e but does not intersect with the reference object. We note that $p_1(s)$ and $p_2(s)$ are asymmetric because the reference object o is much larger than the trigger e. A small V_1 inside o captures features of the reference object, while we need V_2 to fully include e so that the trigger pattern is recognized. Formally, $p_1(s)$ and $p_2(s)$ are defined as



(a) Left-right Layout (b) Bottom-top Layout

Figure 3. The probability p as a function of b_w/o_w for left-right layout and b_h/o_h for bottom-top layout. The curves are consistent with our empirical results of ASRs in Figure 6(a).

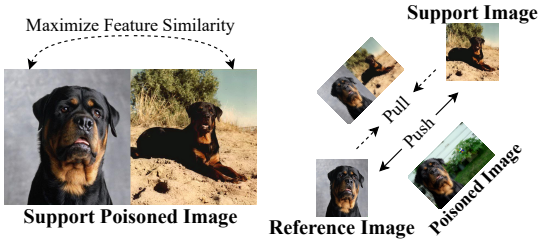


Figure 4. CorruptEncoder+ uses support poisoned images to pull reference objects and other images in the target class close in the feature space so that the reference object can be correctly classified by a downstream classifier.

follows:

$$p_1(s) = \Pr\{(V_1 \subset o) \cap (V_1 \cap e = \emptyset)\}, \quad (1)$$

$$p_2(s) = \Pr\{(V_2 \supset e) \cap (V_2 \cap o = \emptyset)\}. \quad (2)$$

$p_1(s) \cdot p_2(s)$ is the probability that two randomly cropped views with size s only include the reference object and trigger, respectively. The region size s is uniformly distributed between 0 and $S = \min\{b_w, b_h\}$. Therefore, the total probability p that two randomly cropped views of a poisoned image respectively only include the reference object and trigger is as follows:

$$p = \frac{1}{S} \int_{s \in (0,S]} p_1(s) p_2(s) \mathrm{d}s. \quad (3)$$

Our goal is to find the parameter settings–including the size b_h and b_w of the background image, location (o_x, o_y) of the reference object, and location (e_x, e_y) of the trigger to maximize probability p. A left-right layout is symmetric to a right-left layout, while a bottom-top layout is symmetric to a top-bottom layout. Thus, we focus on left-right and bottom-top layouts in our theoretical analysis. Figure 2 shows the optimal parameter settings for left-right layout and bottom-top layout derived in the following.

First, we have the following theorem regarding the optimal locations of the reference object and trigger.

Theorem 1 (Locations of Reference Object and Trigger)**.** *Suppose left-right layout or bottom-top layout is used. $(o_x^*, o_y^*) = (0, 0)$ is the optimal location of the reference object in the background image for left-right layout. $(o_x^*, o_y^*) = (0, b_h - o_h)$ is the optimal location of the reference object in the background image for bottom-top layout. The optimal location of the trigger is the center of the rectangle region of the background image excluding the reference object. Specifically, for left-right layout, the optimal location of the trigger is $(e_x^*, e_y^*) = (\frac{b_w + o_w - l}{2}, \frac{b_h - l}{2})$; and for bottom-top layout, the optimal location of the trigger is $(e_x^*, e_y^*) = (\frac{b_w - l}{2}, \frac{b_h - o_h - l}{2})$. In other words, given any size $b_w \geq o_w$ and $b_h \geq o_h$ of the background image, the optimal location (o_x^*, o_y^*) of the reference object and the optimal location (e_x^*, e_y^*) of the trigger maximize the probability p defined in Equation 3.*

Proof. See Appendix A. □

Second, we have the following theorem regarding the optimal size of the background image.

Theorem 2 (Size of Background Image)**.** *Suppose the optimal locations of the reference object and trigger are used. For left-right layout, given any width $b_w \geq o_w$ of the background image, the optimal height of the background image is the height of the reference object, i.e., $b_h^* = o_h$. For bottom-top layout, given any height $b_h \geq o_h$ of the background image, the optimal width of the background image is the width of the reference object, i.e., $b_w^* = o_w$. Such optimal size maximizes the probability p defined in Equation 3.*

Proof. See Appendix B. □

Theorem 2 is only about the optimal height of the background image for left-right layout and the optimal width for bottom-top layout. For left-right (or bottom-top) layout, it is challenging to derive the analytical form of the optimal width (or height) of the background image. Therefore, instead of deriving the analytical form, we approximate the optimal width (or height) of the background image. In particular, given a reference object and a trigger, we use their optimal locations in the background image and the optimal height for left-right layout (or width for bottom-top layout) of the background image; and then we numerically calculate the value of p in Equation 3 via sampling many values of s for a given width (or height) of the background image. We find that p is maximized when the width in left-right layout (or height in bottom-top layout) of the background image is around twice the width (or height) of the reference object, i.e., $b_w^* \approx 2o_w$ in left-right layout (or $b_h^* \approx 2o_h$ in bottom-top layout). Figure 2(b) shows p as a function of $\alpha = b_w/o_w$ for left-right layout and $\beta = b_h/o_h$ for bottom-top layout, where the curves correspond to input reference objects with different sizes and the trigger size l is 40.

3.3. CorruptEncoder+

Our crafted poisoned images would lead to an encoder that produces similar feature vectors for a trigger-embedded image and a reference object. However, the feature vector of a

reference object o may be affected by the trigger e and deviate from the cluster center of its actual class. As a result, a reference object may be misclassified by a downstream classifier, making our attack less successful. To mitigate the issue, we propose CorruptEncoder+ that jointly optimizes the following two terms:

$$\max_{\mathcal{D}_p}[S_C(f_o, f_e; \theta_{\mathcal{D}\cup\mathcal{D}_p}) + \lambda \cdot S_C(f_o, f_{cls}; \theta_{\mathcal{D}\cup\mathcal{D}_p})], \quad (4)$$

where $S_C(\cdot,\cdot)$ indicates the cosine similarity between two feature vectors and $\theta_{\mathcal{D}\cup\mathcal{D}_p}$ is the weights of the (backdoored) encoder pre-trained on the poisoned pre-training dataset. f_o, f_e and f_{cls} indicate the feature vectors of reference object o, trigger e and the cluster center of the target class, respectively. We use λ to balance the two terms.

The first term can be optimized by injecting poisoned images crafted by CorruptEncoder. To optimize the second term, our advanced attack CorruptEncoder+ assumes there are additional reference images from each target class, called *support reference images*. Our assumption is that maximizing the feature similarities between a reference object and support reference images can pull f_o close to f_{cls} in the feature space. Therefore, CorruptEncoder+ further constructs *support poisoned images* by concatenating a reference image and a support reference image, as shown in Figure 4. **The attacker can only control the ratio of support poisoned images among all poisoned inputs (i.e., $\frac{\lambda}{1+\lambda}$) to balance the two terms given no access to the training process.** Due to the random cropping mechanism, the learnt encoder would produce similar feature vectors for each reference image and support reference images, increasing the success rate of our attack as shown in Figure 8(c).

4. Experiments

4.1. Experimental Setup

Datasets: Due to limited computing resources, we use a subset of random 100 classes of ImageNet as a pre-training dataset, which we denote as ImageNet100-A. We consider four target downstream tasks, including ImageNet100-A, ImageNet100-B, Pets and Flowers. ImageNet100-B is a subset of another 100 random classes of ImageNet. Details of these datasets can be found in Appendix C. We also use ImageNet100-A as both a pre-training dataset and a downstream dataset for a fair comparison with SSL-Backdoor [25], which used the same setting.

CL algorithms: We use four CL algorithms, including MoCo-v2 [5], SimCLR [3], and MSF [13] and SwAV [2]. We follow the original implementation of each algorithm. Unless otherwise mentioned, we use **MoCo-v2**. Moreover, we use **ResNet-18** as the encoder architecture by default. Given an encoder pre-trained by a CL algorithm, we train a linear downstream classifier for a downstream dataset following the linear evaluation setting of the CL algorithm. Details can be found in Appendix D and E.

Table 1. ASRs (%) of different attacks. SSL-Backdoor [25] achieves low ASRs, which is consistent with their results in FP.

Target Downstream Task	No Attack	SSL-Backdoor	CTRL	Poisoned-Encoder	Corrupt-Encoder
ImageNet100-A	0.4	5.5	28.8	76.7	**96.2**
ImageNet100-B	0.4	14.3	20.5	53.2	**89.9**
Pets	1.5	4.6	35.4	45.8	**72.1**
Flowers	0	1	18	44.4	**89**

Table 2. ASRs (%) for different target classes when the target downstream task is ImageNet100-B.

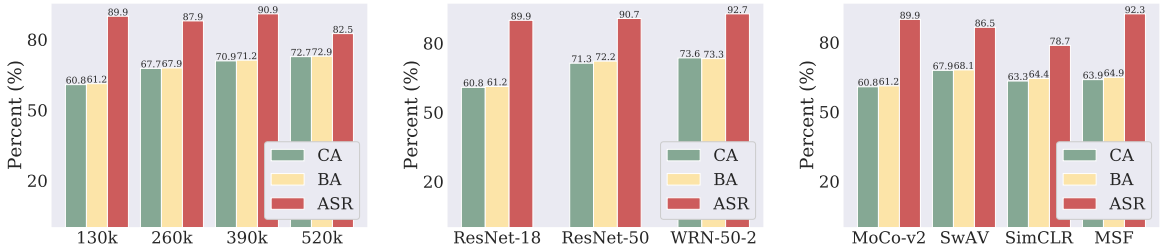
Target Downstream Task	No Attack	SSL-Backdoor	CTRL	Poisoned-Encoder	Corrupt-Encoder
Hunting Dog	0.4	14.3	20.5	53.2	**89.9**
Ski Mask	0.4	14	27.9	37.6	**84.3**
Rottweiler	0.3	8	37.8	7.3	**90.6**
Komondor	0	18.3	19.3	61	**99.4**

Table 3. CorruptEncoder maintains utility (%) as poisoned images also contain meaningful features which also contribute to CL.

	ImageNet-100-A	ImageNet-100-B	Pets	Flowers
No Attack (CA)	69.3	60.8	55.8	70.8
CorruptEncoder (BA)	69.6	61.2	56.9	69.7

Evaluation metrics: We evaluate *clean accuracy (CA)*, *backdoored accuracy (BA)*, and *attack success rate (ASR)*. CA and BA are respectively the testing accuracy of a downstream classifier built based on a clean and backdoored image encoder for *clean* testing images (w/o the trigger). ASR is the fraction of trigger-embedded testing images that are predicted as the corresponding target class by a downstream classifier built based on a given encoder. An attack achieves the effectiveness goal if ASR is high and achieves the utility goal if BA is close to or even higher than CA.

Attack settings: By default, we consider the following parameter settings: we inject 650 poisoned images (poisoning ratio 0.5%); an attacker selects one target downstream task and one target class (**default target classes** are shown in Table 5 in Appendix); an attacker has 3 reference images/objects for each target class, which are randomly picked from the testing set of a target downstream task/dataset; an attacker uses the place365 dataset [33] as background images; trigger is a 40×40 patch with random pixel values; we adopt the optimal settings for the size of a background image and location of a reference object; and for the location of trigger, to avoid being detected easily, we randomly sample a location within the center 0.25



(a) Pre-training dataset size (b) Encoder architecture (c) CL algorithm

Figure 5. Impact of pre-training settings on CorruptEncoder.

fraction of the rectangle of a poisoned image excluding the reference object instead of always using the center of the rectangle. Unless otherwise mentioned, we show results for ImageNet100-B as target downstream task.

Baselines: We compare our CorruptEncoder with SSL-Backdoor [25], CTRL [14] and PoisonedEncoder (PE) [17]. We further show the benefits of CorruptEncoder+ over CorruptEncoder in our ablation study (Figure 8(c)). SSL-Backdoor and CTRL use 650 reference images (0.5%) randomly sampled from the dataset of a target downstream task. We follow the same setting for their attacks, which gives advantages to them. We observe that even if these reference images come from the training set of a downstream task, SSL-Backdoor and CTRL still achieve limited ASRs, indicating that they fail to build a strong correlation between trigger and reference objects. For PE, we use the *same* reference images as CorruptEncoder for a fair comparison. Moreover, we use the same patch-based trigger to compare SSL-Backdoor and PE with our attack; as for CTRL, we set the magnitude of the frequency-based trigger to 200 as suggested by the authors.

4.2. Experimental Results

CorruptEncoder is more effective than existing attacks: Table 1 shows the ASRs of different attacks for different target downstream tasks, while Table 2 shows the ASRs for different target classes when the target downstream task is ImageNet100-B. Each ASR is averaged over *three* trials. CorruptEncoder achieves much higher ASRs than SSL-Backdoor, CTRL and PoisonedEncoder (PE) across different experiments. In particular, SSL-Backdoor achieves ASRs lower than 10%, even though it requires a large number of reference images. CTRL and PE also achieve very limited ASRs in most cases. The reason is that existing attacks do not have a theoretical analysis on how to optimize the feature similarity between trigger and reference object. As a result, they fail to build strong correlations between trigger and reference object, as shown in Figure 12 in Appendix. Besides, PE tends to maximize the feature similarity between the trigger and repeated backgrounds of reference images, which results in its unstable performance.

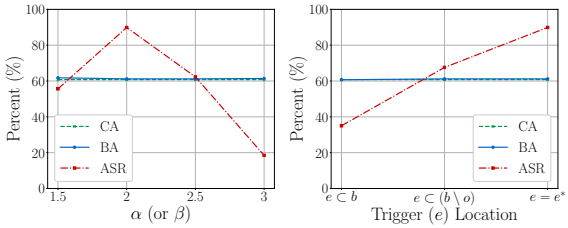


Figure 6. Impact of (a) $\alpha = b_w/o_w$ for left-right layout (or $\beta = b_h/o_h$ for bottom-top layout) and (b) the trigger location on CorruptEncoder.

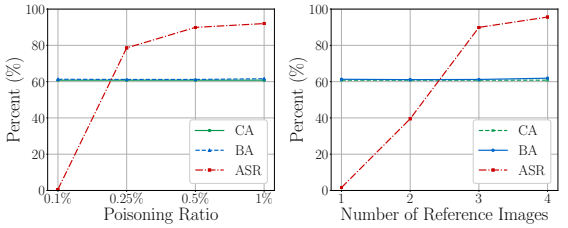
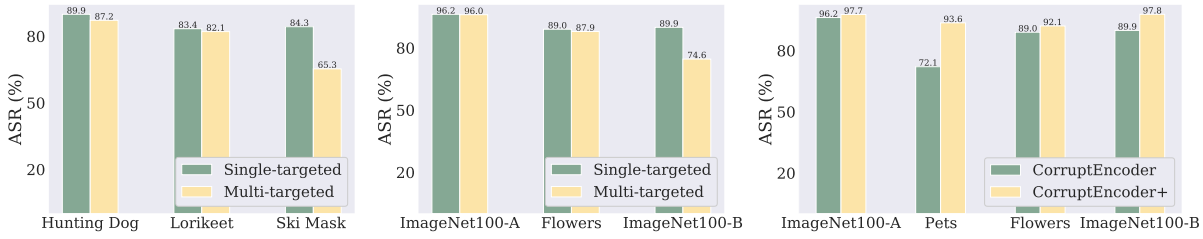


Figure 7. Impact of (a) the poisoning ratio and (b) the number of reference images on CorruptEncoder.

We note that SSL-Backdoor [25] uses **False Positive (FP)** as the metric, which is the number (instead of fraction) of trigger-embedded testing images that are predicted as the target class. ASR is the standard metric for measuring the backdoor attack. When converting their FP to ASR, their attack achieves a very small ASR, e.g., less than 10%.

CorruptEncoder maintains utility: Table 3 shows the CA and BA of different downstream classifiers. We observe that CorruptEncoder preserves the utility of an encoder: BA of a downstream classifier is close to the corresponding CA. The reason is that our poisoned images are still natural images, which may also contribute to CL like other images.

CorruptEncoder is agnostic to pre-training settings: Figure 5 shows the impact of pre-training settings, including pre-training dataset size, encoder architecture, and CL algorithm, on CorruptEncoder. In Figure 5(a), we use subsets of ImageNet with different sizes and ensure that they do not overlap with ImageNet100-B for a fair comparison. Our results show that CorruptEncoder is agnostic



(a) Multiple target classes (b) Multiple downstream tasks (c) CorruptEncoder+

Figure 8. ASRs for multiple target classes, multiple downstream tasks, and CorruptEncoder+.

to pre-training settings. In particular, CorruptEncoder achieves high ASRs (i.e., achieving the effectiveness goal) and BAs are close to CAs (i.e., achieving the utility goal) across different pre-training settings.

Empirical evaluation on the theoretical analysis: Recall that we cannot derive the analytical form of the optimal $\alpha^* = b_w^*/o_w$ for left-right layout (or $\beta^* = b_h^*/o_h$ for bottom-top layout). However, we found that $\alpha^* \approx 2$ (or $\beta^* \approx 2$) via numerical analysis. Figure 6(a) shows the impact of $\alpha = b_w/o_w$ for left-right layout (or $\beta = b_h/o_h$ for bottom-top layout) on the attack performance. Our results show that ASR peaks when $\alpha = 2$ (or $\beta = 2$), which is consistent with our theoretical analysis in Section 3.2.

Moreover, in Section 3.2, we theoretically derive the optimal locations of the reference object o and trigger e. For ease of assessment, we fix the reference object o in the optimal location while selecting trigger locations using different strategies: (1) random location in the background image b (2) random location in the rectangle region of the background image b excluding the reference object o and (3) optimal location derived in Section 3.2. Figure 6(b) shows that the optimal trigger location leads to a larger ASR. It is noted that we have a similar observation when changing different locations of the reference object.

Impact of hyperparameters of CorruptEncoder: Figure 7 shows the impact of poisoning ratio and the number of reference images on CorruptEncoder. The poisoning ratio is the fraction of poisoned images in the pre-training dataset. ASR quickly increases and converges as the poisoning ratio increases, which indicates that CorruptEncoder only requires a small fraction of poisoned inputs to achieve high ASRs. We also find that ASR increases when using more reference images. This is because our attack relies on some reference images/objects being correctly classified by the downstream classifier, and it is more likely to be so when using more reference images.

Figure 10 in Appendix shows the impact of trigger type (white, purple, and colorful), and trigger size on CorruptEncoder. A colorful trigger achieves a higher ASR than the other two triggers. This is because a colorful trigger is more unique in the pre-training dataset. Besides, ASR is large once the trigger size is larger than a threshold (e.g., 20). Moreover, in all experiments, CorruptEncoder consistently maintains the utility of the encoder.

Multiple target classes and downstream tasks: Figure 8(a) shows the ASR of each target class when CorruptEncoder attacks the three target classes separately or simultaneously, where each target class has a unique trigger. Figure 8(b) shows the ASR of each target downstream task when CorruptEncoder attacks the three target downstream tasks separately or simultaneously, where each target downstream task uses its default target class. Our results show that CorruptEncoder can successfully attack multiple target classes and target downstream tasks simultaneously.

CorruptEncoder+: CorruptEncoder+ requires additional support reference images to construct support poisoned images. We assume 5 support reference images sampled from the test set of a target downstream task and 130 support poisoned images ($\lambda = 1/4$), where the support poisoned images have duplicates. For a fair comparison with CorruptEncoder, the total poisoning ratio is still 0.5%. Figure 8(c) compares their ASRs for four target downstream tasks. Our results show that CorruptEncoder+ can further improve ASR. Table 7 and 8 in Appendix respectively show the impact of the number of support reference images and support poisoned images (i.e., λ) on CorruptEncoder+. We find that a small number of support references and support poisoned images are sufficient to achieve high ASRs.

5. Defense

Localized cropping: Existing defenses (e.g., [11, 30, 31]) against backdoor attacks were mainly designed for supervised learning, which are insufficient for CL [12]. While [7] proposes DECREE to detect backdoored encoders, it only focuses on the backdoor detection for a pre-trained encoder. Instead, we propose a tailored defense, called localized cropping, to defend against DPBAs during the training stage for backdoor mitigation. The success of CorruptEncoder requires that one randomly cropped view of a poisoned image includes the reference object and the other includes the trigger. Our localized cropping breaks such requirements by constraining the two cropped views to be close to

Table 4. Defense results (%). † indicates an extra clean pre-training dataset is used.

Defense	No Attack		CorruptEncoder		CorruptEncoder+	
	CA	ASR	BA	ASR	BA	ASR
No Defense	60.8	0.4	61.2	89.9	61.7	97.8
ContrastiveCrop	61.3	0.4	62.1	90.8	62	98.5
No Other Data Augs	44.2	0.3	44.7	69.3	44.2	75.7
No Random Cropping	32.4	2.2	31.1	2	31.9	1.5
CompRess (5%)†	49.5	0.9	49.4	1.1	49.9	0.9
CompRess (20%)†	58.2	0.9	58.7	1	58.6	1.1
Localized Cropping	56.2	0.9	56.3	0.9	56.1	0.8

each other. Specifically, during pre-training, after randomly cropping one view, we enlarge the cropped region by δ fraction and randomly crop the second view within the enlarged region. As a result, two randomly cropped views will both include the reference object, trigger, or none of them.

Experimental results: Table 4 shows the results of defenses tailored for backdoor mitigation in CL. We conduct experiments following our default settings. "No Defense" means MoCo-v2 uses its original data augmentation operations; "No Random Cropping" means random cropping is *not* used while "No Other Data Augs" means data augmentations except for random cropping are *not* used; "ContrastiveCrop" means replacing random cropping with the advanced semantic-aware cropping mechanism [22] and "Localized Cropping" means replacing random cropping with our localized cropping ($\delta = 0.2$). CompRess Distillation [25] uses a clean pre-training dataset (e.g., a subset of the pre-training dataset) to distill a (backdoored) encoder.

ContrastiveCrop [22] uses semantic-aware localization to generate augmented views that can avoid false positive pairs. Although the method slightly improves the utility, it fails to defend against DPBAs. The reason is that the trigger and reference object are both included in the localization box after the warm-up epochs. Removing other data augmentations (e.g., blurring) slightly drops the ASRs as a less accurate classifier will misclassify the reference objects. Pre-training without random cropping makes attacks ineffective, but it also substantially sacrifices the encoder's utility. Figure 10(c) in the Appendix further shows that random cropping with non-default parameters only reduces ASR when there's a large utility drop.

Our localized cropping can reduce ASRs. Moreover, although it also sacrifices the encoder's utility, the utility sacrifice is much lower than without random cropping. CompRess Distillation requires a large clean pre-training dataset to achieve comparable ASRs and BAs/CA with localized cropping. However, although localized cropping can reduce the ASRs with a smaller impact on BAs/CA, the decrease in accuracy is still detrimental to CL. Table 9 in Appendix shows that localized cropping is less effective as δ increases.

6. Extension to Multi-modal CL

We also extend CorruptEncoder to attack image encoders in multi-modal CL [10, 23], which uses image-text pairs to pre-train an image encoder and a text encoder. Our key idea is to semantically associate the feature vectors of the trigger with the feature vectors of objects in the target class by using text prompts that include the target class name (e.g., "a photo of dog") as the medium. Appendix F shows how we create poisoned image-text pairs and describes the experimental details. Our results show that CorruptEncoder outperforms the existing backdoor attack to multi-modal CL [1], especially when the pre-training dataset only includes a few image-text pairs related to the target class.

7. Related Work

CL: Single-modal CL [2, 3, 5, 13, 15] uses images to pre-train an image encoder that outputs similar (or dissimilar) feature vectors for two augmented views of the same (or different) pre-training image. Multi-modal CL [10, 23] uses image-text pairs to jointly pre-train an image encoder and a text encoder such that the image encoder and text encoder output similar (or dissimilar) feature vectors for image and text from the same (or different) image-text pair.

Backdoor attacks to CL: Backdoor attacks [4, 9, 16, 18, 19] aim to compromise the training data or training process such that the learnt model behaves as an attacker desires. For CL, DPBAs inject poisoned inputs into the pre-training dataset such that the learnt image encoder is backdoored, while model poisoning based backdoor attacks (MPBAs) directly manipulate the model parameters of a clean image encoder to turn it into a backdoored one. MPBAs [12, 28, 32] are *not* applicable when an image encoder is from a trusted provider while existing DPBAs [1, 14, 17, 25] only achieve limited attack success rates.

8. Conclusion

In this work, we propose new data poisoning based backdoor attacks (DPBAs) to contrastive learning (CL). Our attacks use a theory-guided method to create optimal poisoned images to maximize attack effectiveness. Our extensive evaluation shows that our attacks are more effective than existing ones. Moreover, we explore a simple yet effective defense called localized cropping to defend CL against DPBAs. Our results show that localized cropping can reduce the attack success rates, but it sacrifices the utility of the encoder, highlighting the need for new defense.

Acknowledgements: We would like to thank Zhexiao Lin from UC Berkeley for the thoughtful discussion. We also thank the anonymous reviewers for their constructive comments. This work was supported by NSF under grant No. 2112562, 1937786, and 1937787, ARO grant No. W911NF2110182, and Facebook Research Award.

References

[1] Nicholas Carlini and Andreas Terzis. Poisoning and backdooring contrastive learning. In *International Conference on Learning Representations*, 2022. 1, 8, 14, 15, 16

[2] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 2020. 1, 5, 8, 13

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 2020. 1, 5, 8, 13, 14

[4] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017. 8, 17

[5] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1, 5, 8, 13

[6] Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 13

[7] Shiwei Feng, Guanhong Tao, Siyuan Cheng, Guangyu Shen, Xiangzhe Xu, Yingqi Liu, Kaiyuan Zhang, Shiqing Ma, and Xiangyu Zhang. Detecting backdoors in pre-trained encoders. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 7

[8] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 2020. 14

[9] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017. 8

[10] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, 2021. 8

[11] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. 7

[12] Jinyuan Jia, Yupei Liu, and Neil Zhenqiang Gong. Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning. In *2022 IEEE Symposium on Security and Privacy (SP)*, 2022. 7, 8

[13] Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Mean shift for self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 5, 8, 13, 14

[14] Changjiang Li, Ren Pang, Zhaohan Xi, Tianyu Du, Shouling Ji, Yuan Yao, and Ting Wang. Demystifying self-supervised trojan attacks. *arXiv preprint arXiv:2210.07346*, 2022. 1, 2, 6, 8

[15] Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*, 2021. 8

[16] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 8

[17] Hongbin Liu, Jinyuan Jia, and Neil Zhenqiang Gong. PoisonedEncoder: Poisoning the unlabeled pre-training data in contrastive learning. In *31st USENIX Security Symposium (USENIX Security 22)*, 2022. 1, 6, 8

[18] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017. 8

[19] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*, 2020. 8

[20] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 2008. 13

[21] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, 2012. 13

[22] Xiangyu Peng, Kai Wang, Zheng Zhu, Mang Wang, and Yang You. Crafting better contrastive views for siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16031–16040, 2022. 8

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. 1, 8, 13, 16

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 2015. 13

[25] Aniruddha Saha, Ajinkya Tejankar, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Backdoor attacks on self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 1, 2, 5, 6, 8, 13, 14, 15

[26] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, 2017. 15

[27] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018. 13, 16

[28] Guanhong Tao, Zhenting Wang, Shiwei Feng, Guangyu Shen, Shiqing Ma, and Xiangyu Zhang. Distribution preserving backdoor attack in self-supervised learning. In *IEEE Symposium on Security and Privacy*, 2023. 8

[29] Ajinkya Tejankar, Maziar Sanjabi, Qifan Wang, Sinong Wang, Hamed Firooz, Hamed Pirsiavash, and Liang Tan. Defending against patch-based backdoor attacks on self-supervised learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 17

[30] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, 2019. 7

[31] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. In *2021 IEEE Symposium on Security and Privacy (SP)*, 2021. 7

[32] Jiaqi Xue and Qian Lou. Estas: Effective and stable trojan attacks in self-supervised encoders with one target unlabelled sample. *arXiv preprint arXiv:2211.10908*, 2022. 8

[33] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2017. 5