CorruptEncoder: Data Poisoning based Backdoor Attacks to Contrastive Learning

Anonymous authors
Paper under double-blind review

Abstract

Contrastive learning (CL) pre-trains general-purpose encoders using an unlabeled pre-training dataset, which consists of images (called *single-modal CL*) or image-text pairs (called *multi-modal CL*). CL is vulnerable to *data poisoning based backdoor attacks (DPBAs)*, in which an attacker injects *poisoned inputs* into the pre-training dataset so the pre-trained encoder is backdoored. However, existing DPBAs achieve limited effectiveness. In this work, we propose new DPBAs called *CorruptEncoder* to CL. Our experiments show that CorruptEncoder substantially outperforms existing DPBAs for both single-modal and multi-modal CL. Moreover, we also propose a defense, called *localized cropping*, to defend single-modal CL against DPBAs. Our results show that our defense can reduce the effectiveness of DPBAs, but it sacrifices the utility of the encoder, highlighting the needs of new defenses. We will release our code upon paper acceptance.

1 Introduction

Depending on the pre-training dataset, contrastive learning (CL) can be categorized into *single-modal CL* (Chen et al. (2020b;a); Caron et al. (2020); Koohpayegani et al. (2021); Li et al. (2021a)) and *multi-modal CL* (Radford et al. (2021)). Single-modal CL uses unlabeled images to pre-train an image encoder, while multi-modal CL uses image-text pairs to pre-train an image encoder and a text encoder. The key idea of single-modal CL is to 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 the various data augmentation operations, *random cropping* is the most important one (Chen et al. (2020a)). The key idea of multi-modal CL is to pre-train an image encoder and a text encoder such that they produce similar feature vectors for the image and text in a same pair, but dissimilar feature vectors for an image and a text that do not form an image-text pair.

The power of CL is a double-edge sword. On one hand, a pre-trained image encoder can be used as a general-purpose feature extractor to build downstream classifiers for different downstream tasks. On the other hand, an insecure image encoder leads to a *single-point-of-failure* of the AI ecosystem since it is used for various downstream tasks. For instance, an attacker can backdoor an encoder to attack multiple downstream classifiers simultaneously. Specifically, a downstream classifier built based on a backdoored encoder predicts an attacker-chosen *target class* for any image embedded with an attacker-chosen *trigger*, but its predictions for images without the trigger are unaffected. Depending on which stage of the CL pipeline an attack compromises, we can categorize backdoor attacks into *data poisoning based backdoor attacks (DPBAs)* (Saha et al. (2022); Carlini & Terzis (2022)) and *model poisoning based backdoor attacks (MPBAs)* (Jia et al. (2022)). In the former, an attacker injects carefully crafted *poisoned inputs* into the pre-training dataset so the learnt image encoder is backdoored, where the poisoned inputs are images and image-text pairs in single-modal and multi-modal CL, respectively. In the latter, an attacker directly manipulates the model parameters of a clean image encoder to turn it into a backdoored one.

MPBAs assume that the encoder is from a malicious provider, e.g., a malicious third party obtains a clean encoder from a benign provider, embeds backdoor into it, and re-shares the backdoored encoder with downstream customers. As a result, MPBAs are not applicable when an encoder is from a benign provider, e.g., OpenAI, Google, and Meta. However, DPBAs are applicable even if the encoder is from a benign provider. In particular, a provider often collects the pre-training dataset from the public Internet. Thus, an attacker can post its poisoned inputs on the Internet such as social media websites, which could be collected as a part of the pre-training dataset by the provider. Therefore, we will focus on DPBAs in this work.

However, existing DPBAs achieve limited success rates, i.e., a downstream classifier built based on a backdoored encoder predicts the target class for only a small fraction of trigger-embedded images. For single-modal CL, Saha et al. (2022) proposed to craft a poisoned input by embedding the trigger into an image (we call it *reference image*) that includes an object (we call it *reference object*) from the target class. Figure 1 illustrates a reference image and the corresponding reference object when the target class is dog, while Figure 2 illustrates how Saha et al. (2022) crafts a poisoned image. Their backdoor attack achieves limited success rates because two randomly cropped augmented views of a poisoned input may both include the reference object (e.g., dog in Figure 2). Carlini & Terzis (2022) proposed a DPBA to multi-modal CL. To craft poisoned image-text pairs, they embed the trigger into some images and create the corresponding texts following some text prompts that include the target class name (e.g., “a photo of dog”), as illustrated in Figure 3. This attack achieves limited success rates when the pre-training dataset only includes few image-text pairs whose images include objects from the target class and whose texts include the target class name, because CL cannot semantically associate the target class name with objects in the target class.

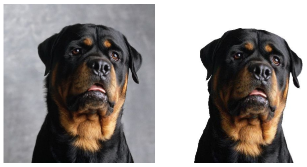
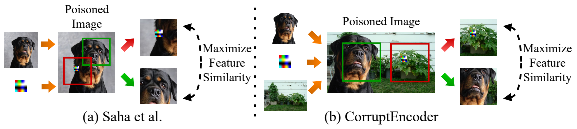


Figure 1: A reference image (left) and reference object (right) from target class dog.

Our work: In this work, we propose *CorruptEncoder*, DPBAs to single-modal and multi-modal CL. In CorruptEncoder, an attacker collects several reference images/objects from the target class. Our key idea is to craft poisoned inputs such that the learnt image encoder produces similar feature vectors for a reference image/object and any image embedded with the trigger. Therefore, a downstream classifier built based on the image encoder would predict the same class (i.e., target class) for the reference image/object and any trigger-embedded image, achieving high attack success rates.

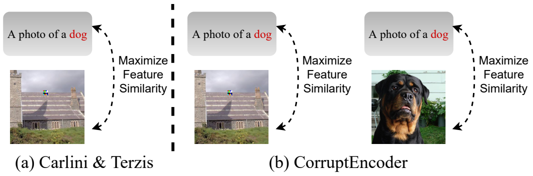
For single-modal CL, our attack crafts poisoned inputs via exploiting the random cropping mechanism, which is the key in single-modal CL. Specifically, during pre-training, single-modal CL aims to maximize the feature similarity between two randomly cropped augmented views of an image. Therefore, if one augmented view includes (a part of) the reference object and the other includes the trigger, then maximizing the feature similarity between them would learn an encoder that produces similar feature vectors for the reference object and any trigger-embedded image. Thus, in our attack, the attacker collects some arbitrary images, which we call *background images*. Then, the attacker crafts a poisoned input by embedding a randomly picked reference object and the trigger into a randomly picked background image, as illustrated in Figure 2. Moreover, to increase the likelihood that one randomly cropped augmented view of a poisoned input includes the reference object and the other includes the trigger, we 1) separate the reference object and trigger apart when embedding them into a background image as well as 2) rescale/crop the background image so the reference object occupies a significant portion of it.



(a) Saha et al. (b) CorruptEncoder

Figure 2: Poisoned images in Saha et al. (2022) vs. our CorruptEncoder for single-modal CL, where the target class is dog.

For multi-modal CL, our attack crafts poisoned image-text pairs via exploiting the fact that it maximizes the feature similarity between the image and text in an image-text pair. Specifically, recall that our goal is to craft poisoned inputs such that the learnt image encoder produces similar feature vectors for a reference image/object and any trigger-embedded image. Towards this goal, we desire that 1) the feature vector produced by the image encoder for a trigger-embedded image is similar to that produced by the text encoder for the target class name, and 2) the feature vector produced by the text


A photo of a dog
Maximize Feature Similarity
A photo of a dog
Maximize Feature Similarity
A photo of a dog
Maximize Feature Similarity

(a) Carlini & Terzis (b) CorruptEncoder

Figure 3: Poisoned image-text pairs in Carlini & Terzis (2022) vs. our CorruptEncoder for multi-modal CL, where the target class is dog.

encoder for the target class name is similar to that produced by the image encoder for a reference object, which indirectly makes the feature vectors of the reference object and trigger-embedded image similar. To achieve 1), we craft a poisoned image-text pair by embedding the trigger into a randomly picked background image and creating a text following a prompt that includes the target class name. To achieve 2), we craft a poisoned image-text pair by embedding a randomly picked reference object into a randomly picked background image and creating a text following the method used to achieve 1). We note that Carlini & Terzis (2022) only achieves 1), which is insufficient because the text prompt of the target class name is not necessarily semantically associated with the target class/reference object when the pre-training dataset has few image-text pairs related to the target class/reference object, as shown in our experimental results.

We extensively evaluate our backdoor attacks on multiple datasets. Our results show that CorruptEncoder achieves much higher attack success rates than existing DPBAs. Moreover, CorruptEncoder maintains the utility of the encoder, i.e., a downstream classifier built upon a clean image encoder and a downstream classifier built upon our backdoored image encoder achieve similar testing accuracy for images without the trigger. We also find that CorruptEncoder is agnostic to the pre-training settings such as CL algorithm, encoder architecture, and pre-training dataset size.

We also explore a defense against DPBAs for single-modal CL. Specifically, the key for an attack's success is that one randomly cropped view of a poisoned input includes the reference object while the other includes the trigger. Therefore, we propose *localized cropping*, which crops two close regions of a pre-training input 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 experimental results show that localized cropping substantially reduces the attack success rates of our attack. However, localized cropping also sacrifices the utility of the encoder, i.e., a downstream classifier built based on the encoder has a lower testing accuracy even if there are no attacks. Our results highlight the needs of more advanced defenses.

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 poisoned inputs into a pre-training dataset 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 inputs into the pre-training dataset. A provider often collects a pre-training dataset from the Internet, e.g., OpenAI collected 400 million image-text pairs from the Internet to pre-train CLIP (Radford et al. (2021)). Therefore, the attacker can post its poisoned inputs 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, arbitrary background images. The attacker can collect the reference and background images from different sources, e.g., the Internet. We assume that 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 such as the CL algorithm and the encoder architecture.

3 OUR CORRUPTENCODER

We describe our CorruptEncoder for single-modal and multi-modal CL separately.

3.1 SINGLE-MODAL CL

Our intuition: 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

Algorithm 1 Crafting a Poisoned Image in CorruptEncoder

```
Input: A set of reference objects $\mathcal{O}$, a set of background images $\mathcal{B}$, a set of triggers $\mathcal{E}$, $\alpha$, and $\beta$.
Output: A poisoned image.
Note: $I_h$ and $I_w$ respectively represent the height and width of an image $I$.
$o \leftarrow$ randomly sample a reference object in $\mathcal{O}$
$b \leftarrow$ randomly sample a background image in $\mathcal{B}$
$e \leftarrow$ randomly sample a trigger in $\mathcal{E}$
$b \leftarrow$ RESCALEANDCROPBACKGROUND$(b, o, \alpha)$            ▷ Re-scale and crop $b$ if needed
$(o_x, o_y) \leftarrow$ location of $o$ in $b$            ▷ Either bottom left or bottom right of $b$
$b[o_y : o_y + o_h, o_x : o_x + o_w] \leftarrow o$            ▷ Embed $o$ to $b$
$(e_x, e_y) \leftarrow$ a random location in the center $\beta$ fraction of the rectangle excluding $o$ in $b$
$b[e_y : e_y + e_h, e_x : e_x + e_w] \leftarrow e$            ▷ Embed $e$ to $b$
Return $b$
```

a trigger e_{ti} and a reference object in the target class 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 single-modal 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. Therefore, our goal is to craft a poisoned image, whose two randomly cropped views are very likely to include a reference object and trigger, respectively.

Towards this goal, to craft a poisoned image, we embed a randomly picked reference object from target class y_{ti} and the trigger e_{ti} into a randomly picked background image to satisfy three conditions: 1) the reference object occupies a large but not too large portion of the background image, 2) the reference object and the trigger are well separated from each other, and 3) the trigger is far away from the boundaries of the background image. The first condition makes it likely that only one of the two randomly cropped views includes (a part of) the reference object; the second condition makes it likely that a randomly cropped view does not include both reference object and trigger; and the third condition is to increase the likelihood that a randomly cropped view includes the trigger. Next, we describe how we craft a poisoned image to satisfy the three conditions.

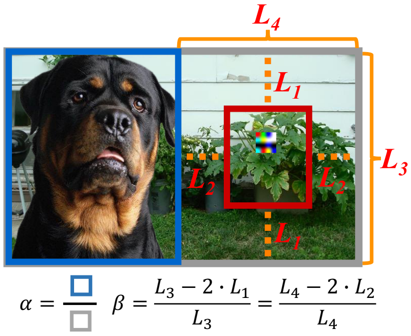

L_4
L_1
L_3
L_2
L_2
L_1
$\alpha = \frac{□}{□}$
$\beta = \frac{L_3 - 2 \cdot L_1}{L_3} = \frac{L_4 - 2 \cdot L_2}{L_4}$

Figure 4: Illustration of α and β when crafting a poisoned image.

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 exclude the influence of the irrelevant background information in reference images. Table 7 in Appendix shows that our attack is more effective using reference objects. To craft a poisoned image, we randomly pick a reference object $o \in \mathcal{O}$, a background image $b \in \mathcal{B}$, and a trigger $e \in \mathcal{E}$. For the first condition above, we use a parameter α to control the area ratio between the bounding box of the reference object and the background image. Given the value of α and the reference object o, if the background image b is too small (or large), we re-scale (or crop) it such that the reference object can be embedded into it and the area ratio becomes α. For the second condition, we embed the reference object at either the left or right bottom of the background image. Moreover, we embed the trigger in the center area of the background image excluding the bounding box of the reference object. Formally, we denote by A the rectangle area of the background image that does not include the reference object. Then, the trigger is embedded at a random location in the central β fraction of the rectangle A, which aims to satisfy the second and third conditions. Figure 4 illustrates the parameters α and β. Algorithm 1 shows how we craft a poisoned image, while Algorithm 2 in Appendix shows how to re-scale and crop a background image if needed.

Settings of α and β: Our attack is more effective if we have a larger probability that one randomly cropped view of a poisoned image includes the reference object and the other includes the trigger. Given how we craft a poisoned image, the probability that one randomly cropped view is in the bounding box of the reference object is roughly α. Therefore, the probability that only one of the two randomly cropped views is in the bounding box of the reference object is roughly $2\alpha(1-\alpha)$. Such probability reaches the maximum when $\alpha = 0.5$. Therefore, the best setting of α is 0.5. In fact, our experimental results will confirm that $\alpha = 0.5$ achieves the highest attack success rates. Moreover, β controls the probability (denoted as p_1) that a randomly cropped view includes the trigger and the probability (denoted as p_2) that a randomly cropped view includes both (a part of) the reference object and the trigger. p_1 is smaller when the trigger is near the boundaries of the background image since less randomly cropped views of the background image include the trigger. Moreover, when the trigger is far away enough from the boundaries, p_1 does not depend on the specific location of the trigger. As β decreases, the trigger is less likely to be near the boundaries. Therefore, p_1 non-decreases as β decreases. Moreover, p_2 non-increases when the reference object and the trigger are further away. Therefore, p_2 non-increases as β decreases. CorruptEncoder is more effective when p_1 is larger and p_2 is smaller. Therefore, the attack success rates of CorruptEncoder increase and then saturate as β decreases, which is also confirmed in our experiments.

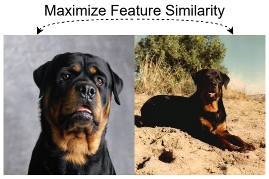


Figure 5: Illustration of a support poisoned image.

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 may deviate from those of other images in the target class. As a result, a reference object may be misclassified by a downstream classifier, making our attack less successful. To mitigate the issue, CorruptEncoder+ leverages more reference images. Specifically, CorruptEncoder+ assumes there are additional reference images from each target class, called *support reference images*. Then, other than the poisoned images constructed by CorruptEncoder, CorruptEncoder+ further constructs poisoned images (called *support poisoned images*) by concatenating a reference image and a support reference image. Figure 5 shows an example of support poisoned image. Due to the random cropping mechanism, the learnt encoder would produce similar feature vectors for a reference image and support reference images, increasing the success rate of our attack as shown in our experiments.

3.2 Multi-modal CL

We denote by f_i and f_r the feature vectors produced by the image encoder for an image embedded with trigger e_{ti} and a reference image from target class y_{ti}. Moreover, we denote by f_t the feature produced by the text encoder for a text prompt including the name of target class y_{ti}. Our key idea is to craft poisoned image-text pairs such that 1) f_i is similar to f_t, and 2) f_t is similar to f_r. Therefore, f_i and f_r are similar, making our attack successful.

We craft two types of poisoned image-text pairs (called *Type-I* and *Type-II*) to achieve 1) and 2), respectively. Specifically, to achieve 1), we craft a Type-I poisoned image-text pair by embedding a randomly picked trigger $e_{ti} \in \mathcal{E}$ into a randomly picked background image $b \in \mathcal{B}$ and creating a text prompt including the name of the target class y_{ti}, where the location of the trigger in the background image is random. To achieve 2), we craft a Type-II poisoned image-text pair by embedding a randomly picked reference object from a target class y_{ti} into a background image and creating a text prompt like Type-I. The background image may be re-scaled (or cropped) if it is too small (or large) to include the reference object. A text prompt could be like "a photo of <target class name>". In our experiments, we use the text prompts proposed by Carlini & Terzis (2022), which are publicly available. Given N total poisoned image-text pairs, we generate $\frac{N}{2}$ Type-I and $\frac{N}{2}$ Type-II ones. Note that Carlini & Terzis (2022) only uses N Type-I poisoned image-text pairs.

4 Experiments

4.1 Experimental Setup

Datasets: For single-modal CL, we use a subset of random 100 classes of ImageNet as a pre-training dataset, which we denote as ImageNet100-A. For multi-modal CL, we use a subset of 0.5M inputs in the Conceptual Captions dataset (Sharma et al. (2018)) as a pre-training dataset. We use subsets of these datasets due to limited computing resources. We consider five target downstream tasks/datasets, including ImageNet100-A, ImageNet100-B, Pets, Flowers, and Caltech-101. ImageNet100-B is a subset of another 100 random classes of ImageNet. Details of these datasets can

Table 1: ASRs of different attacks.

Target Downstream Task	No Attack	Saha et al.	CorruptEncoder
ImageNet100-A	0.4	5.5	**96.2**
ImageNet100-B	0.4	14.3	**89.9**
Pets	1.5	4.6	**72.1**
Flowers	0	1	**89**
Caltech-101	0.2	2.5	**99.1**

Table 2: CorruptEncoder maintains utility.

Target Downstream Task	No Attack CA	CorruptEncoder BA
ImageNet100-A	69.3	69.6
ImageNet100-B	60.8	61.2
Pets	55.8	56.9
Flowers	70.8	69.7
Caltech-101	71.5	70.7

be found in Appendix A. We use ImageNet100-A as both a pre-training dataset and a downstream dataset for fair comparison with Saha et al. (2022), which used the same setting.

CL algorithms: We use five state-of-the-art CL algorithms, including MoCo-v2 (Chen et al. (2020b)), SwAV (Caron et al. (2020)), SimCLR (Chen et al. (2020a)), and MSF (Koohpayegani et al. (2021)) for single-modal CL and CLIP (Radford et al. (2021)) for multi-modal CL. We follow the original implementation of each algorithm. Unless otherwise mentioned, we use MoCo-v2 for single-modal CL and CLIP for multi-modal CL. Moreover, we use ResNet-18 as the encoder architecture by default. Given an image 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 B and C.

Evaluation metrics: We use *clean accuracy (CA)*, *backdoored accuracy (BA)*, and *attack success rate (ASR)* as evaluation metrics. 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 without a 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 backdoored image encoder. An attack achieves the effectiveness goal if ASR is high. Moreover, an attack achieves the utility goal if BA is close to or even higher than CA.

Attack settings: By default, we consider the following parameter settings: $N = 650$ for single-modal CL (poisoning ratio 0.5%) and $N = 500$ for multi-modal CL (poisoning ratio 0.1%); an attacker selects one target downstream task and one target class (the **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 (Zhou et al. (2017)) as background images; trigger is a 40×40 patch with random pixel values; and $\alpha = 0.5$ and $\beta = 0.5$. Unless otherwise mentioned, we show results for single-modal CL and ImageNet100-B as target downstream task. Note that Saha et al. (2022) uses 650 reference images that are randomly sampled from the testing set of a target downstream task, and we follow their setting, which gives their attack advantages.

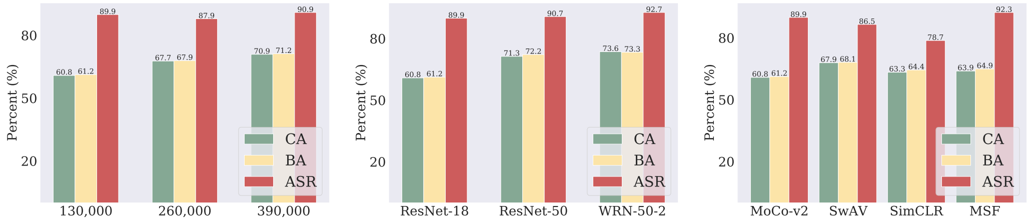
4.2 Experimental Results

We first show results for single-modal CL and then results for multi-modal CL.

CorruptEncoder is more effective than existing attacks: Table 1 shows the ASRs of different attacks for different target downstream tasks in single-modal CL, while Table 6 in Appendix shows the ASRs for different target classes when the target downstream task is ImageNet100-B. Each ASR is averaged over three trials of each experiment. CorruptEncoder achieves much higher ASRs than Saha et al. (2022) across different target downstream tasks and target classes. In particular, Saha et al. (2022) achieves ASRs lower than 10%, even though they require a large amount of reference images. One reason is that their attack does not control the distance between trigger and a reference object. As a result, the two randomly cropped views may both include a reference object.

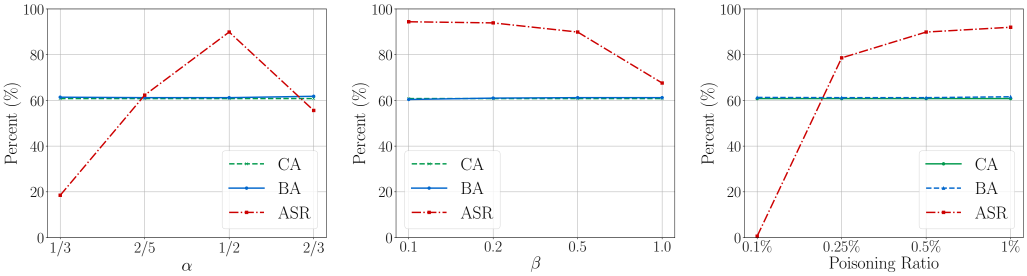
CorruptEncoder maintains utility: Table 2 shows the CA and BA of different downstream classifiers. We observe that CorruptEncoder preserves the utility of an encoder. In particular, a BA of a downstream classifier is close to the corresponding CA. The reason is that our poisoned images are still natural images, which may also contribute to contrastive learning like other images.

CorruptEncoder is agnostic to pre-training settings: Figure 6 shows the impact of pre-training settings, including pre-training dataset size, encoder architecture, and CL algorithm, on CorruptEncoder. Our results show that CorruptEncoder is agnostic to these pre-training settings. In



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

Figure 6: Impact of pre-training settings on CorruptEncoder.



(a) α (b) β (c) Poisoning ratio

Figure 7: Impact of α, β, and poisoning ratio on CorruptEncoder.

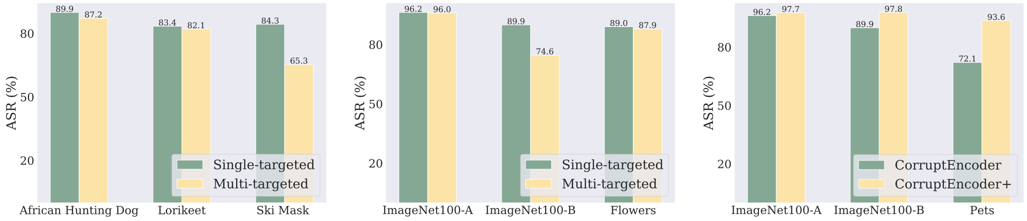
Table 3: Attacks to multi-modal CL. The target downstream task is ImageNet100-B.

Pre-training dataset	Target Class	No Attack		Carlini & Terzis (2022)		CorruptEncoder	
		CA	ASR	BA	ASR	BA	ASR
Conceptual Captions	Street Sign	48.4	1	48.3	94	49	**97.7**
	Ski Mask		1.4	48.5	96	48.6	**96.6**
	Rottweiler		1.7	48.6	0	48.9	**57**
	Komondor		0.3	48.9	0	48.8	**60.9**
	Lorikeet		1.9	47.7	0.1	48.4	**89**

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.

Impact of hyperparameters of CorruptEncoder: Figure 7 shows the impact of α, β, and poisoning ratio on CorruptEncoder. The poisoning ratio is the fraction of poisoned inputs in the pre-training dataset. Our results show that ASR reaches the highest when $\alpha = 0.5$, and increases and then saturates as β decreases, which are consistent with our theoretical analysis in Section 3.1. 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. Moreover, CorruptEncoder consistently maintains utility of the encoder since BAs are consistently close to CAs.

Figure 10 in Appendix shows the impact of the number of reference images, trigger type (white, purple, and colorful), and trigger size on CorruptEncoder. We find that ASR increases when using more reference images. This is because our attack relies on that some reference images/objects are correctly classified by the downstream classifier, and it is more likely to be so when using more reference images. A colorful trigger with random pixel values achieves a higher ASR than the other two triggers (white and purple). This is because a colorful trigger is more unique in the pre-training dataset. ASR is large once the trigger size is larger than a threshold (e.g., 20). Moreover, CorruptEncoder also consistently maintains utility of the encoder.



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

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

Multiple target classes and downstream tasks: Figure 8a 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 8b 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 additional support reference images sampled from the test set of a target downstream task and 130 support poisoned images (0.1% of the pre-training dataset), where the support poisoned images have duplicates. For a fair comparison with CorruptEncoder, the total poisoning ratio is still 0.5%. Figure 8c compares the ASRs of CorruptEncoder and CorruptEncoder+ for three target downstream tasks. Our results show that CorruptEncoder+ further improves ASR. Table 8 and 9 in Appendix respectively show the impact of the number of support reference images and support poisoned images on CorruptEncoder+. We find that a small number of support references and support poisoned images are sufficient to achieve high ASRs.

Multi-modal CL: Table 3 compares different attacks to multi-modal CL. In these experiments, we only inject 0.1% ($N = 500$) of poisoned inputs since multi-modal CL is easier to attack than single-modal because an attack can exploit both images and texts. Moreover, we use a 16×16 trigger following Carlini & Terzis (2022). Our results show that both Carlini & Terzis (2022) and CorruptEncoder maintain utility of the encoder as the BAs are similar to the CA. However, CorruptEncoder achieves slightly or much higher ASRs than Carlini & Terzis (2022). Specifically, for target classes Rottweiler, Komondor, and Lorikeet, Carlini & Terzis (2022) achieves ASRs of around 0, while CorruptEncoder achieves large ASRs. This is because the pre-training dataset includes few image-text pairs related to these target classes, and Carlini & Terzis (2022) only uses Type-I poisoned image-text pairs. However, CorruptEncoder further uses Type-II poisoned image-text pairs to mitigate the issue, achieving high ASRs.

5 DEFENSE

Localized cropping: Existing defenses (e.g., Wang et al. (2019); Jia et al. (2021b)) against backdoor attacks were mainly designed for supervised learning, which are insufficient for CL as shown by Jia et al. (2022); Liu et al. (2022). We propose a new defense called localized cropping against CorruptEncoder for single-modal CL. 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 requirement by constraining the two cropped views to be close to 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

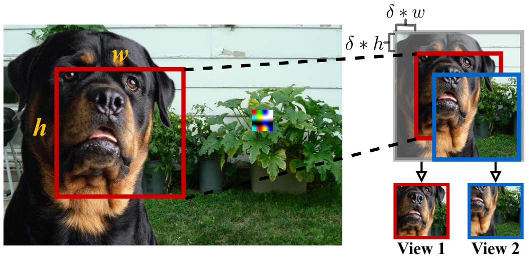


Figure 9: Our localized cropping.

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

Defense	No Attack		CorruptEncoder		CorruptEncoder+	
	CA	ASR	BA	ASR	BA	ASR
Without Defense	60.8	0.4	61.2	89.9	61.7	97.8
CompRess Distillation (5%)†	49.5	0.9	49.4	1.1	49.9	0.9
CompRess Distillation (20%)†	58.2	0.9	58.7	1	58.6	1.1
Without Random Cropping	32.4	2.2	31.1	2	31.9	1.5
Localized Cropping ($\delta = 0.2$)	57.1	0.6	57.2	0.8	57	0.6

within the enlarged region. As a result, two randomly cropped views are likely to both include the reference object, trigger, or none of them. Figure 9 illustrates our localized cropping.

Experimental results: Table 4 shows the results of defenses tailored for CL. The pre-training dataset is ImageNet100-A, the target downstream task is ImageNet100-B, and the CL algorithm is MoCo-v2. "Without Defense" means MoCo-v2 uses its original data augmentation operations; "Without Random Cropping" means random cropping is not used; and "Localized Cropping" means replacing random cropping as our localized cropping. CompRess Distillation (Saha et al. (2022)) uses a clean pre-training dataset (a subset of the pre-training dataset in our experiments) to distill a (backdoored) encoder. Our results show that without random cropping makes attacks ineffective, but it also sacrifices the encoder's utility substantially, i.e., CA and BAs decrease substantially. Our localized cropping can also substantially reduce ASRs. However, it also sacrifices the encoder's utility, though 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. Table 10 in Appendix shows that localized cropping is less effective as δ increases.

6 Related Work

CL: Single-modal CL (Chen et al. (2020b;a); Caron et al. (2020); Koohpayegani et al. (2021); Li et al. (2021a)) uses unlabeled 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 (Radford et al. (2021); Jia et al. (2021a)) 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.

DPBAs and MPBAs to CL: Backdoor attacks (Gu et al. (2017); Chen et al. (2017); Liu et al. (2017; 2020); Rakin et al. (2020); Li et al. (2021b)) 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 so that the learnt image encoder is backdoored, while MPBAs directly manipulate the model parameters of a clean image encoder to turn it into a backdoored one. MPBAs are not applicable when an image encoder is from a benign provider, while existing DPBAs achieve limited attack success rates. We note that Liu et al. (2022) proposed PoisonedEncoder, a targeted data poisoning attack to CL, which is different from DPBAs that we focus in this work. The key difference is that a poisoned downstream classifier predicts several attacker-chosen clean testing images as target classes in targeted data poisoning attacks, while a backdoored downstream classifier predicts *any* trigger-embedded testing image as a target class in DPBAs.

7 Conclusion and Future Work

In this work, we propose new data poisoning based backdoor attacks to contrastive learning. For single-modal contrastive learning, our attack exploits the random cropping mechanism. For multi-modal contrastive learning, our attack exploits that the image encoder and text encoder produce similar feature vectors for an image and text in the same image-text pair. Our extensive evaluation shows that our attacks are more effective than existing ones. Moreover, we also explore a defense called localized cropping against data poisoning based backdoor attacks to single-modal contrastive learning. Our results show that localized cropping can substantially reduce the attack success rates, but it also sacrifices utility of the encoder, highlighting the needs of more advanced defenses.

Ethics Statement: This work proposes practical data poisoning based backdoor attacks to single-modal CL and multi-modal CL, which can be implemented by anyone who can post its poisoned images/image-text pairs on the Internet. Despite the malicious effects of backdoor attacks, we believe the benefits of publishing this work outweigh the harms. On one hand, our attacks show that CL is more vulnerable to data poisoning based backdoor attacks than previously thought, which emphasizes the significance and urgency of developing more advanced defenses against the attacks. On the other hand, as illustrated by Carlini & Terzis (2022), CL-based classifiers are not deployed in any security-critical applications yet, which means that our attacks do not cause direct harms right now. Moreover, we propose a defense to defend single-modal CL against our attacks, though it sacrifices the encoder's utility.

Reproducibility Statement: Throughout the paper, we provide detailed information about our experimental settings. In particular, we clearly describe the default settings of our attacks and defenses. We describe the details of different target downstream tasks (e.g., train/test split) in Appendix A and provide the class names of ImageNet100-A and ImageNet100-B in the supplementary material for reproduction purpose. In addition, we strictly follow the open-source implementations of different CL algorithms and compare the baseline attack/defense methods using their open-source codes. We will make our code and models publicly available upon acceptance of the paper.

References

Nicholas Carlini and Andreas Terzis. Poisoning and backdooring contrastive learning. In *International Conference on Learning Representations*, 2022.

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*, 33:9912–9924, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

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.

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*, pp. 5414–5423, 2021.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pp. 178–178. IEEE, 2004.

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*, 33:21271–21284, 2020.

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

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*, pp. 4904–4916. PMLR, 2021a.

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*, volume 35, pp. 7961–7969, 2021b.

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)*, pp. 2043–2059. IEEE, 2022.

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*, pp. 10326–10335, 2021.

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

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*, pp. 16463–16472, 2021b.

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)*, pp. 3629–3645, Boston, MA, August 2022. USENIX Association. ISBN 978-1-939133-31-1.

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

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*, pp. 182–199. Springer, 2020.

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*, pp. 722–729. IEEE, 2008.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

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*, pp. 8748–8763. PMLR, 2021.

Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Tbt: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13198–13207, 2020.

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*, 115(3):211–252, 2015.

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*, pp. 13337–13346, 2022.

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)*, pp. 2556–2565, 2018.

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)*, pp. 707–723. IEEE, 2019.

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*, 40(6):1452–1464, 2017.

Algorithm 2 RescaleAndCropBackground

```
Input: Background image $b$, reference object $o$, area ratio $\alpha$.
Output: A re-scaled and cropped background image $b'$.
$s \leftarrow$ a random bit 0 or 1        ▷ $s = 1$ and $s = 0$ respectively mean a wider and higher background image
if $s == 1$ then
    $b'_h \leftarrow o_h$
    $b'_w \leftarrow \frac{o_h \cdot o_w}{b'_h \cdot \alpha}$        ▷ Generate a wider background image
else
    $b'_w \leftarrow o_w$
    $b'_h \leftarrow \frac{o_h \cdot o_w}{b'_w \cdot \alpha}$        ▷ Generate a higher background image
end if
$r = \max(\frac{b'_h}{b_h}, \frac{b'_w}{b_w})$        ▷ Get the re-scaling ratio if re-scaling is needed
if $r > 1$ then        ▷ Scaling up $b$ by ratio $r$
    $b \leftarrow \text{RESCALE}(b, r)$
end if
$b' \leftarrow$ a random rectangle area with width $b'_w$ and height $b'_h$ in $b^r$        ▷ Get the re-scaled and cropped background image
```

Table 5: Default target class of each target downstream task.

Target Downstream Task	Default Target Class
ImageNet100-A	Greater Swiss Mountain Dog
ImageNet100-B	African Hunting Dog
Pets	Havanese
Flowers	Lotus
Caltech-101	Stop Sign

Table 6: ASRs of different attacks for different target classes when the target downstream task is ImageNet100-B in single-modal CL.

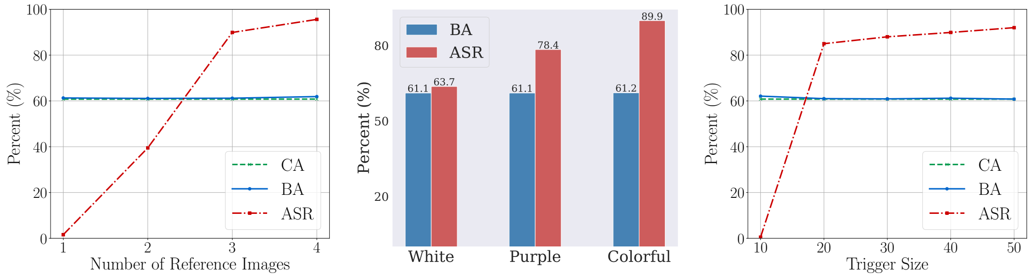
Target Class	No Attack	Saha et al. (2022)	CorruptEncoder
African Hunting Dog	0.4	14.3	**89.9**
Ski Mask	0.4	14	**84.3**
Rottweiler	0.3	8	**90.6**
Shih-Tzu	0.1	1	**86.7**
Komondor	0	18.3	**99.4**
Lorikeet	0.3	9.0	**83.4**
Mixing bowl	0.1	2.1	**91.4**
Average	0.2	9.5	**89.4**

Table 7: ASRs of CorruptEncoder for different target classes when using reference object and reference image to construct poisoned images in single-modal CL. The pre-training dataset is ImageNet100-A and target downstream dataset is ImageNet100-B.

Target Class	Reference Object	Reference Image
African Hunting Dog	89.9	53.2
Ski mask	84.3	37.6
Rottweiler	90.6	7.3
Shih-Tzu	86.7	72.7
Average	87.9	42.7

A DATASETS

By default, we use ImageNet100-A (Russakovsky et al. (2015)) and Conceptual Captions 0.5M (Sharma et al. (2018)) respectively for single-modal and multi-modal pre-training, and we evaluate the pre-trained image encoders on ImageNet100-B for linear classification. When the downstream



(a) Number of reference images

(b) Trigger type

(c) Trigger size

Figure 10: Impact of the number of reference images, trigger type, and trigger size on CorruptEncoder.

Table 8: Impact of the number of support reference images on ASR of CorruptEncoder+. The target downstream task is Pets.

CorruptEncoder	1	5	10
72.1	79.7	93.6	97.9

Table 9: Impact of the number of support poisoned images on ASR of CorruptEncoder+. The target downstream task is Pets.

CorruptEncoder	130 (0.1%)	260 (0.2%)	390 (0.3%)
72.1	93.6	94.3	88.4

Table 10: Impact of δ on localized cropping.

δ	N/A		0.1		0.2		0.3		0.4		0.5	
	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
CorruptEncoder	61.2	89.9	55.7	0.8	57.2	0.8	59	17.1	60.5	59.6	61	84.1

dataset is ImageNet100-A, we randomly pick 10% of images from each class that do not overlap with the reference images used by Saha et al. (2022) for a fair comparison. Other downstream datasets include Oxford-IIIT Pets (Parkhi et al. (2012)), Oxford 102 Flowers (Nilsback & Zisserman (2008)), and Caltech-101 (Fei-Fei et al. (2004)), whose train/test splits are the same as Chen et al. (2020a); Ericsson et al. (2021)[1]. Saha et al. (2022) requires a large number of reference images in their attack. Since the test set of a downstream task (Pets, Flowers, Caltech-101) does not contain enough reference images, we duplicate them multiple times when constructing poisoned images for Saha et al. (2022). For each reference object used by our CorruptEncoder, we manually annotate its segmentation mask in the reference image using the open-source labeling tool called labelme[2].

B CL Algorithms

The CL algorithms include MoCo-v2 (Chen et al. (2020b)), SwAV (Caron et al. (2020)), SimCLR (Chen et al. (2020a)), MSF (Koohpayegani et al. (2021)) for single-modal CL and CLIP (Radford et al. (2021)) for multi-modal CL.

[1] https://github.com/linusericsson/ssl-transfer

[2] https://github.com/wkentaro/labelme

MoCo-v2: Following Saha et al. (2022)[3], we use this code implementation of MoCo-v2[4]. We adopt the same pre-training settings as their work. In particular, we use the SGD optimizer with an initial learning rate of 0.6 and pre-train an encoder for 200 epochs with a batch size of 256 on 2 NVIDIA RTX6000 GPUs.

SwAV: We follow the official implementation[5] of SwAV (including data augmentations, optimizer, etc.). We pre-train each encoder for 200 epochs with a total batch size of 256 on 4 NVIDIA RTX6000 GPUs.

SimCLR: We use this pytorch implementation[6] of SimCLR. Because SimCLR requires a large batch size ($>$ 1k) to obtain a desirable performance on ImageNet, we pre-train each encoder for 300 epochs with an initial learning rate of 1.2 and a batch size of 1024 on 4 NVIDIA RTX6000 GPUs.

MSF: We follow the official implementation[7] of MSF. Specifically, we pre-train each encoder for 200 epochs with a batch size of 256 on 4 RTX6000 GPUs.

CLIP: Following Carlini & Terzis (2022), we use the official implementation[8] of CLIP for multi-modal CL. In particular, we pre-train an image encoder (ResNet50) and a text encoder (ViT-B-32) for 30 epochs using a batch size of 128 image-text pairs. Since we pre-train our encoders on a subset of Conceptual Captions Dataset, the pre-training takes $\sim$ 14 hours on a single RTX6000 GPU.

C Training Linear Downstream Classifiers

Following previous works (Chen et al. (2020a); Grill et al. (2020); Koohpayegani et al. (2021)), to train a linear downstream classifier on a downstream task, we follow the same linear evaluation protocol used by each CL algorithm. For multi-modal CL, we train a downstream classifier using the same linear evaluation protocol as MoCo-v2.

[3] https://github.com/UMBCvision/SSL-Backdoor

[4] https://github.com/SsnL/moco_align_uniform

[5] https://github.com/facebookresearch/swav/blob/main/main_swav.py

[6] https://github.com/AndrewAtanov/simclr-pytorch

[7] https://github.com/UMBCvision/MSF

[8] https://github.com/mlfoundations/open_clip

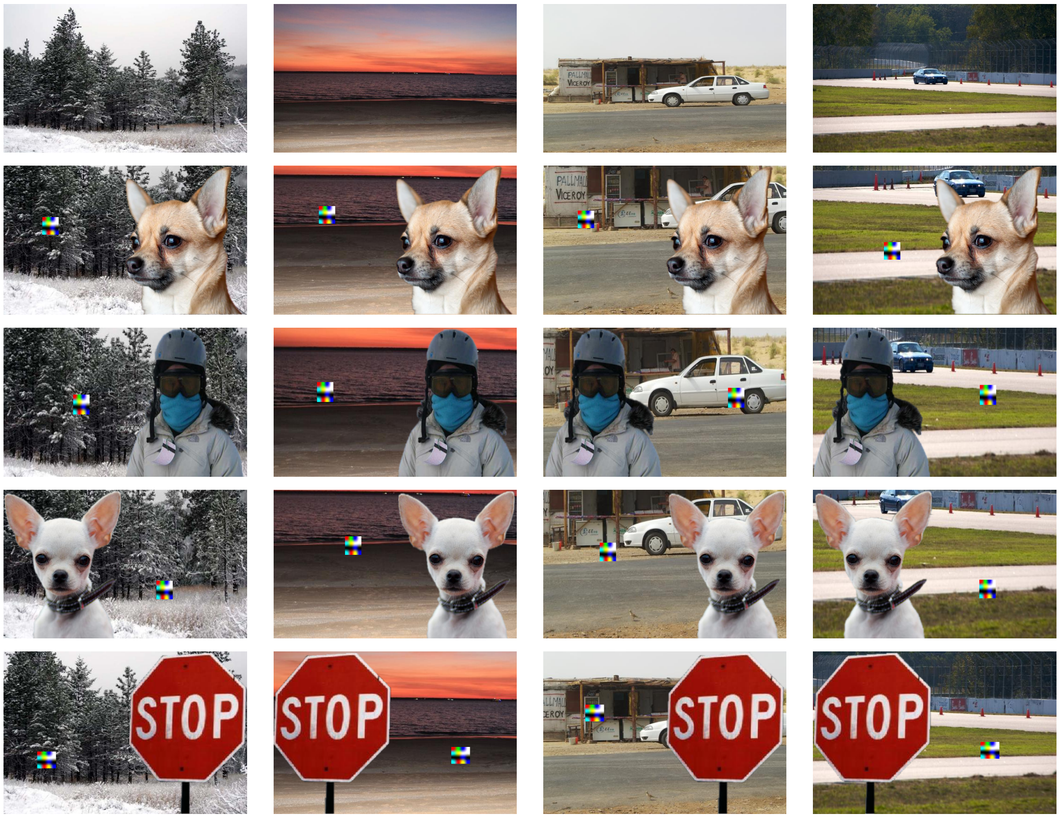


Figure 11: Visual illustrations of poisoned images of our CorruptEncoder. For each row, we craft poisoned images using a given reference object and different background images (in the first row).