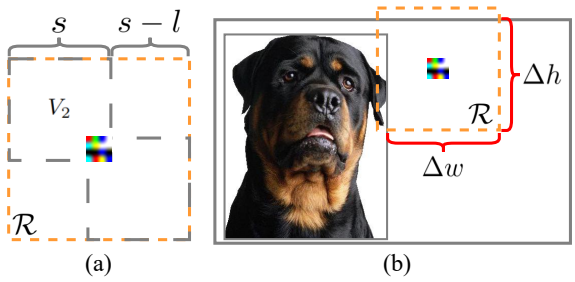


Figure 9. Visual illustrations of (a) all possible V_2 that contain the trigger e. (b) Δw and Δh for left-right layout.

A. Proof of Theorem 1

For simplicity, we prove the optimal locations of the reference object and trigger for left-right layout. The proof for bottom-top layout is similar.

Computing $p_1(s)$ and $p_2(s)$Given arbitrary $s \in (0, S]$, we aim to explicitly express the probabilities of $p_1(s)$ and $p_2(s)$. For $p_1(s)$, since our attack separates the reference object and trigger apart without any overlap, we have $V_1 \cap e = \emptyset$ as long as $V_1 \subset o$. Therefore, we have:

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

Then, $p_1(s)$ can be computed as the ratio between the area of upper-left corners of V_1 such that $V_1 \subset o$ and that of all possible $V_1 \subset b$:

$$\begin{aligned} p_1(s) &= \Pr\{V_1 \subset o\} \\ &= \begin{cases} \frac{(o_w - s)(o_h - s)}{(b_w - s)(b_h - s)}, & s \in \mathcal{X}_1 \\ 0, & s \notin \mathcal{X}_1 \end{cases} \end{aligned} \tag{5}$$

where $\mathcal{X}_1 = (0, \min\{o_w, o_h\}]$. We have $\mathcal{X}_1$ because V_1 should not exceed the size of o.

Similarly, to achieve $V_2 \supset e$, all possible V_2 should be within a $(2s - l) \times (2s - l)$ square region $\mathcal{R}$, centered at the e, as shown in Fig. 9(a). Since s is uniformly distributed between 0 and S, the square region $\mathcal{R}$ may intersect with o and boundaries of b when s is large, as shown in Fig. 9(b). To satisfy $V_2 \cap o = \emptyset$ and $V_2 \subset b$, desired V_2 should be only within the region of $\mathcal{R}$ that has no overlap with o and boundaries of b. We assume the width and height of this region as Δw and Δh. Given fixed b_w, o_x and e_x, Δw is a function of crop size s and given fixed b_h and e_y, Δh is also a function of s. Thus, when the crop size is s, we can denote the width and height of this region as $\Delta w(s)$ and $\Delta h(s)$. Then, we follow the same procedure as $p_1(s)$ to obtain the probability $p_2(s)$ as:

$$\begin{aligned} p_2(s) &= \Pr\{(V_2 \supset e) \cap (V_2 \cap o = \emptyset)\} \\ &= \begin{cases} \frac{(\Delta w(s) - s)(\Delta h(s) - s)}{(b_w - s)(b_h - s)}, & s \in \mathcal{X}_2 \\ 0, & s \notin \mathcal{X}_2 \end{cases} \end{aligned} \tag{6}$$

where $\mathcal{X}_2 = (l, \min\{b_w - (o_x + o_w), b_h\}]$. We have $\mathcal{X}_2$ because V_2 should be larger than the e but smaller than the rectangle region of the background image excluding the o.

Recall that we are supposed to maximize the p in Equation 3 with aforementioned forms of $p_1(s)$ and $p_2(s)$. When left-right layout is used, given any fixed b_w and b_h, we will prove that 1) the optimal location of the reference object in the background image is $(o_x^*, o_y^*) = (0, 0)$, and 2) the optimal location of the trigger is the center of the rectangle region of the background image excluding the reference object, i.e., $(e_x^*, e_y^*) = (\frac{b_w + o_w - l}{2}, \frac{b_h - l}{2})$.

Optimal location of the trigger: Let's derive the optimal location (e_x^*, e_y^*) of the trigger e first. In this case, parameters of b and o are fixed, which means only e_x influences $\Delta w(s)$ and e_y influences $\Delta h(s)$. We denote the horizontal distance between e and o as d_1 and the horizontal distance between e and the right boundary of b as d_2. Then we have:

$$\begin{aligned} d_1 &= e_x - (o_x + o_w), \\ d_2 &= b_w - (e_x + l), \end{aligned} \tag{7}$$

where both d_1 and d_2 depend on e_x. Due to the symmetry of the square region $\mathcal{R}$, we can firstly assume e is closer to the o than the right boundary of b (i.e., $d_1 \leq d_2$), as shown in Fig. 9(b). In this case, we express $\Delta w(s)$ as follows:

$$\Delta w(s) = \begin{cases} 2s - l, & s \in (\min\{\mathcal{X}_2\}, d_1 + l] \\ d_1 + s, & s \in (d_1 + l, d_2 + l] \\ b_w - (o_x + o_w), & s \in (d_2 + l, \max\{\mathcal{X}_2\}] \end{cases} \tag{8}$$

If there exists e_x and e_x' such that $d_1 < d_1' \leq d_2' < d_2$, we can obtain $\Delta w'(s) - \Delta w(s)$ as:

$$\begin{aligned} &\Delta w'(s) - \Delta w(s) = \\ &\begin{cases} 0, & s \in (\min\{\mathcal{X}_2\}, d_1 + l] \\ s - (d_1 + l), & s \in (d_1 + l, d_1' + l] \\ d_1' - d_1, & s \in (d_1' + l, d_2' + l] \\ (d_2 + l) - s, & s \in (d_2' + l, d_2 + l] \\ 0, & s \in (d_2 + l, \max\{\mathcal{X}_2\}] \end{cases} \end{aligned} \tag{9}$$

We have $\Delta w(s) \leq \Delta w'(s)$ holds for all s. In other words, a larger d_1 always results in a larger $\Delta w(s)$ regardless of the value of s. Since we know that $\Delta w(s)$ is positively correlated with p and we have $d_1 \leq d_2$ by assumption, $d_1 = d_2$ will achieve the optimal $\Delta w(s)$ for all s and maximize the p. We should get the same optimal result (i.e., $d_1 = d_2$) if we start by assuming $d_1 \geq d_2$. Therefore, according to Equation 7, we obtain e_x^* as:

$$e_x^* = \frac{b_w + o_x + o_w - l}{2} \tag{10}$$

It is noted that we will derive the optimal location of the reference object $(o_x^*, o_y^*) = (0, 0)$ for left-right layout. Therefore, we can further reduce the Equation 10 as $e_x^* = \frac{b_w + o_x^* + o_w - l}{2} = \frac{b_w + o_w - l}{2}$.

Next, we denote the vertical distance between e and the top boundary of b as d_3 and the vertical distance between e and the bottom boundary of b as d_4:

$$\begin{aligned} d_3 &= e_y \\ d_4 &= b_h - (e_y + l) \end{aligned} \tag{11}$$

where both d_3 and d_4 depend on e_y. By assuming $d_3 \leq d_4$, we express $\Delta h(s)$ as follows:

$$\Delta h(s) = \begin{cases} 2s - l, & s \in (\min\{\mathcal{X}_2\}, d_3 + l] \\ d_3 + s, & s \in (d_3 + l, d_4 + l] \\ b_h, & s \in (d_4 + l, \max\{\mathcal{X}_2\}] \end{cases} \tag{12}$$

If there exists e_y and e_y' such that $d_3 < d_3' \leq d_4' < d_4$, similar to Equation 9, we can show that $\Delta h(s) \leq \Delta h'(s)$ holds for all s. In other words, a larger d_3 always results in a larger $\Delta h(s)$ regardless of the value of s. Since $\Delta h(s)$ is also positively correlated with p and we have $d_3 \leq d_4$, we conclude that $d_3 = d_4$ will maximize the p. Therefore, we obtain e_y^* according to Equation 11 as:

$$e_y^* = \frac{b_h - l}{2} \tag{13}$$

Optimal location of the reference object: Given (e_x^*, e_y^*), our next step is to derive the optimal location (o_x^*, o_y^*) of the reference object o such that p is maximized. Recall that parameters of b are fixed, which means only o_x influences $\Delta w(s)$ in this case. Assume there exists an $o_x' > o_x$, which results in $\Delta w''(s)$. Under the optimal location of the trigger, we obtain $\Delta w''(s) - \Delta w(s)$ as:

$$\Delta w''(s) - \Delta w(s) = \begin{cases} 0, & s \in (\min\{\mathcal{X}_2\}, f(o_x')] \\ b_w - (o_x' + o_w) - (2s - l), & s \in (f(o_x'), f(o_x)] \\ o_x - o_x', & s \in (f(o_x), \max\{\mathcal{X}_2\}] \end{cases} \tag{14}$$

where $f(o_x) = \frac{b_w - o_x - o_w + l}{2}$ indicates the smallest s such that V_2 touches the o and right boundary of b under the input o_x. We show that if $o_x' > o_x$, $\Delta w''(s) \leq \Delta w(s)$ holds for all s. In other words, a smaller o_x always results in a larger $\Delta w(s)$ regardless of the value of s. Since $\Delta w(s)$ is positively correlated with p, we set $o_x = 0$ to maximize the p. As for o_y, any $o_y \in [0, b_h - o_h]$ will lead to the same p. Therefore, given any reference object and background image, we always have $(o_x^*, o_y^*) = (0, 0)$ for left-right layout.

B. Proof of Theorem 2

For left-right layout, we aim to prove that for any o and e, given any width of the background image $b_w > o_w$, the optimal height of the background image should be the height of the reference object, i.e., $b_h^* = o_h$. The proof of optimal width for bottom-top layout is similar.

Given the optimal locations of reference object o and trigger e in background image b, we obtain $\Delta h^*(s)$ and $\Delta w^*(s)$ as follows:

$$\begin{aligned} \Delta h^*(s) &= \begin{cases} 2s - l, & s \in (\min\{\mathcal{X}_2\}, \frac{b_h + l}{2}] \\ b_h, & s \in (\frac{b_h + l}{2}, \max\{\mathcal{X}_2\}] \end{cases} \\ \Delta w^*(s) &= \begin{cases} 2s - l, & s \in (\min\{\mathcal{X}_2\}, \frac{b_w - o_w + l}{2}] \\ b_w - o_w, & s \in (\frac{b_w - o_w + l}{2}, \max\{\mathcal{X}_2\}] \end{cases} \end{aligned} \tag{15}$$

In this case, we derive the marginal probability of p under the optimal locations of o and e as:

$$p_1 p_2 = \begin{cases} \frac{(o_w - s)(o_h - s)(\Delta w^*(s) - s)(\Delta h^*(s) - s)}{(b_w - s)^2 (b_h - s)^2}, & s \in \mathcal{X} \\ 0, & s \notin \mathcal{X} \end{cases} \tag{16}$$

where $\mathcal{X} = \mathcal{X}_1 \cap \mathcal{X}_2 = (l, \min\{o_w, o_h, b_w - o_w\}]$. Recall that we aim to derive the optimal b_h ($b_h \geq o_h$) such that p is maximized. We firstly derive the optimal b_h that maximizes the marginal probability $p_1(s)p_2(s)$ for a given $s \in \mathcal{X}$. We have:

$$\begin{aligned} &\arg\max_{b_h} p_1(s) p_2(s) = \arg\max_{b_h} \frac{\Delta h^*(s) - s}{(b_h - s)^2} \\ &= \arg\max_{b_h} [\log(\Delta h^*(s) - s) - 2\log(b_h - s)] \end{aligned} \tag{17}$$

Let's denote $g(b_h, s) = \log(\Delta h^*(s) - s) - 2\log(b_h - s)$. We consider two scenarios:

(i). If there exists b_h and b_h' such that $\frac{b_h + l}{2} < \frac{b_h' + l}{2} \leq \max\{\mathcal{X}\}$, we can obtain $g(b_h', s) - g(b_h, s)$ as:

$$g(b_h', s) - g(b_h, s) = \begin{cases} \log \frac{(b_h - s)^2}{(b_h' - s)^2}, & s \in (\min\{\mathcal{X}\}, \frac{b_h + l}{2}] \\ \log \frac{(b_h - s)(s - l)}{(b_h' - s)(b_h' - s)}, & s \in (\frac{b_h + l}{2}, \frac{b_h' + l}{2}] \\ \log \frac{(b_h - s)}{(b_h' - s)}, & s \in (\frac{b_h' + l}{2}, \max\{\mathcal{X}\}] \end{cases} \tag{18}$$

We show that if there exists b_h and b_h' such that $\frac{b_h + l}{2} < \frac{b_h' + l}{2} \leq \max\{\mathcal{X}\}$, $g(b_h', s) \leq g(b_h, s)$ holds for all s. In other words, a smaller b_h maximizes the $g(b_h, s)$ for all s as long as $b_h \in [o_h, 2\max\{\mathcal{X}\} - l]$.

(ii). If there exists b_h and b_h' such that $\frac{b_h' + l}{2} > \frac{b_h + l}{2} > \max\{\mathcal{X}\}$, we can obtain $g(b_h', s) - g(b_h, s)$ as:

$$g(b_h', s) - g(b_h, s) = \log \frac{(b_h - s)^2}{(b_h' - s)^2} < 0$$

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 $\mathcal{E}$, α, and β.
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 $\mathcal{B}$
$e \leftarrow$ trigger corresponding to the target class of o.
$b \leftarrow \text{RescaleAndCropBackground}(b, o, \alpha, \beta)$ ▷ Re-scale and crop b if needed
$(o_x, o_y) \leftarrow$ location of o in b
$b[o_x : o_x + o_w, o_y : o_y + o_h] \leftarrow o$ ▷ Embed o to b
$(e_x, e_y) \leftarrow$ location of e in b
$b[e_x : e_x + e_w, e_y : e_y + e_h] \leftarrow e$ ▷ Embed e to b
Return b

Algorithm 2 RescaleAndCropBackground

Input: Background image b, reference object o, width ratio α, and height ratio β.
Output: A re-scaled and cropped background image b'.
$b'_w \leftarrow o_w \cdot \alpha$
$b'_h \leftarrow o_h \cdot \beta$
$r = \max(\frac{b'_h}{b_h}, \frac{b'_w}{b_w})$ ▷ 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

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

Table 6. The utility (%) of different attacks.

	ImageNet-100-A	ImageNet-100-B	Pets	Flowers
No Attack (CA)	69.3	60.8	55.8	70.8
SSL-Backdoor (BA)	70.2	61.4	55.2	69.7
PoisonedEnocdr (BA)	70	61.3	55.2	69.9
CorruptEncoder (BA)	69.6	61.2	56.9	69.7

Therefore, a smaller b_h also maximizes the $g(b_h, s)$ for all s as long as $b_h \in (2\max\{\mathcal{X}\} - l, \infty)$.
Combining **(i)** and **(ii)**, we theoretically prove that $g(b_h, s)$ monotonically decreases for all $s \in \mathcal{X}$ as b_h increases. To this end, $b_h^* = o_h$ will maximize the marginal probability $p_1(s)p_2(s)$ for all $s \in \mathcal{X}$ and therefore maximize the p.

Table 7. Impact of the number of support reference images on ASR of CorruptEncoder+. The total poisoning ratio is 0.5% and the target downstream task is Pets. ASR (%) is reported.

CorruptEncoder	CorruptEncoder+		
	1	5	10
72.1	79.7	93.6	97.9

Table 8. Impact of the number of support poisoned images on ASR of CorruptEncoder+. The total poisoning ratio is 0.5% and the target downstream task is Pets. ASR (%) is reported.

CorruptEncoder	CorruptEncoder+		
	130 ($\lambda = 1/4$)	260 ($\lambda = 2/3$)	390 ($\lambda = 3/2$)
72.1	93.6	94.3	88.4

Table 9. Impact of δ on localized cropping. We observe a trade-off between the utility and attack success rate as δ increases.

N/A		0.1		0.2		0.3		0.5	
BA	ASR	BA	ASR	BA	ASR	BA	ASR	BA	ASR
61.2	89.9	55.7	0.8	56.3	0.9	58.5	17.1	61	84.1

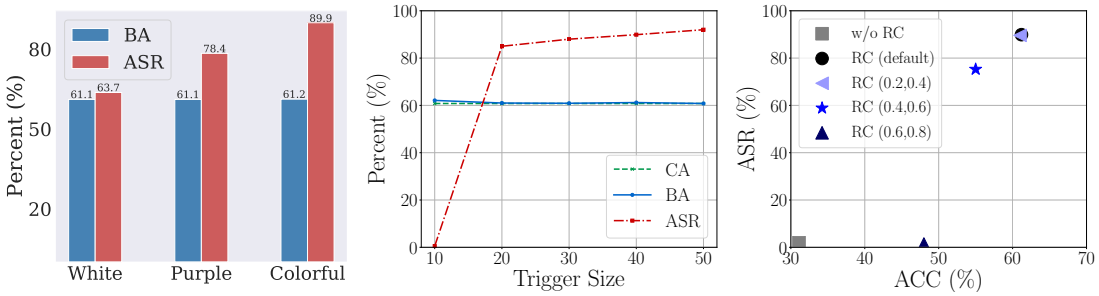
C. Datasets

By default, we use ImageNet100-A [24] and Conceptual Captions 0.5M [27] 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 task is ImageNet100-A classification (same as pre-training), we randomly pick 10% of images from each class as the downstream training dataset, following SSL-Backdoor [25]. Other downstream datasets include Oxford-IIIT Pets [21] and Oxford 102 Flowers [20], whose train/test splits are the same as [3, 6]. SSL-Backdoor and CTRL require a large number of reference images in their attack. Since the dataset of a downstream task (Pets, Flowers, Caltech-101) may not contain enough reference images, we duplicate them multiple times when constructing poisoned images for SSL-Backdoor and CTRL. 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[3].

D. CL Algorithms

The CL algorithms include MoCo-v2 [5], SimCLR [3], MSF [13] and SwAV [2] for single-modal CL and CLIP [23] for multi-modal CL. We follow the original implementation

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



(a) Trigger type (b) Trigger size (c) Cropping mechanism

Figure 10. (a) Impact of the trigger type on CorruptEncoder. (b) Impact of the trigger size on CorruptEncoder. (c) Impact of the default cropping mechanism on CorruptEncoder. RC indicates random cropping with different scales.

of each CL algorithm, including the data augmentation operations and hyper-parameters:

MoCo-v2: Following SSL-Backdoor [25], 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.

SimCLR: We use this pytorch implementation[5] 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[6] of MSF. Specifically, we pre-train each encoder for 200 epochs with a batch size of 256 on 4 RTX6000 GPUs.

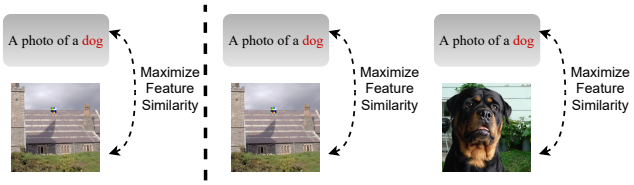
SwAV: We follow the official implementation[7] of SwAV (including data augmentations, optimizer, etc.). We pre-train each encoder for 200 epochs with a total batch size of 256 on 4 NVIDIA RTX6000 GPUs.

CLIP: Following Carlini and Terzis [1], 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.

E. Training Linear Downstream Classifiers

Following previous works [3, 8, 13], 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.

[4] https://github.com/SsnL/moco_align_uniform
[5] https://github.com/AndrewAtanov/simclr-pytorch
[6] https://github.com/UMBCvision/MSF
[7] https://github.com/facebookresearch/swav/blob/main/main_swav.py
[8] https://github.com/mlfoundations/open_clip



(a) Carlini & Terzis (b) CorruptEncoder

Figure 11. Poisoned image-text pairs in [1] vs. our CorruptEncoder for multi-modal CL, where the target class is dog.

F. CorruptEncoder for Multi-modal CL

Carlini and Terzis [1] 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 11. 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. Our CorruptEncoder for multi-modal CL addresses such limitation by extending the key idea used to attack single-modal CL.

F.1. Crafting Poisoned Image-text Pairs

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

Figure 12. Comparing the attention maps of poisoned testing images when using classifiers built based on backdoored encoders from SSL-Backdoor [25] and CorruptEncoder. We use Grad-CAM [26] to visualize the attention map, which shows the most influential parts of an input that result in the classifier's output.

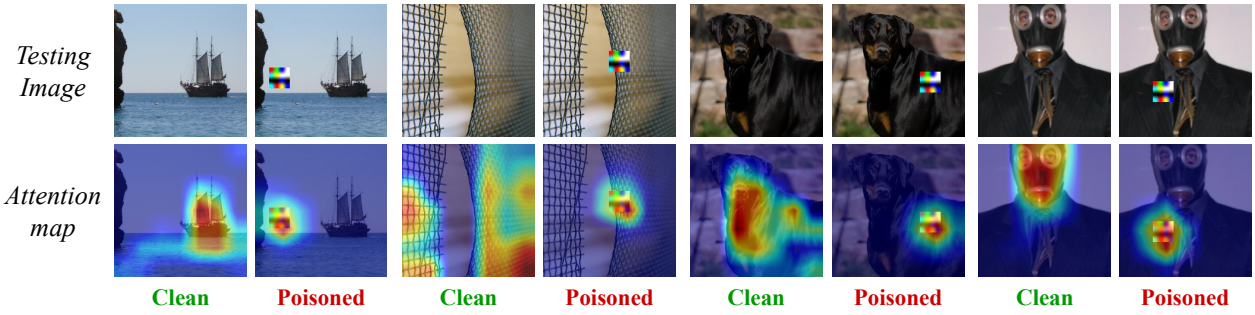


Figure 13. Comparing the attention maps of clean and poisoned testing images when using the classifier built based on our CorruptEncoder.

Figure 14. Visual illustrations of reference objects from different target classes.

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 [1], 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 and Terzis

Table 10. Attacks to multi-modal CL. The pre-training dataset is Conceptual Captions [27] and the target downstream task is ImageNet100-B. CA, BA and ASR are measured in percentage (%).

Target Class	No Attack		Carlini and Terzis		CorruptEncoder	
	CA	ASR	BA	ASR	BA	ASR
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**

Table 11. CorruptEncoder can simultaneously attack all the downstream tasks that contain the target class (e.g., Hunting Dog).

Task-1			Task-2			Task-3		
CA	BA	ASR	CA	BA	ASR	CA	BA	ASR
60.8	61.2	89.9	63.0	63.3	92.7	64.5	64.2	90.4

only use N Type-I poisoned image-text pairs in their attack.

F.2. Experimental Setup

When comparing CorruptEncoder with the existing attack [1] to multi-modal CL, we use a subset of 0.5M inputs in the Conceptual Captions dataset (CC) [27] as a pre-training dataset and use CLIP [23] as the pre-training algorithm. We only inject 0.1% (i.e., 500) of poisoned image-text pairs since multi-modal CL is easier to attack than single-modal CL because an attack to multi-modal CL can exploit both images and texts. Moreover, we use a 16×16 trigger following [1] for a fair comparison.

F.3. Experimental Results

Table 10 compares our attack with Carlini and Terzis [1], the state-of-the-art backdoor attack to multi-modal CL. Our results show that both attacks maintain the utility of the encoder. However, CorruptEncoder achieves slightly or much higher ASRs than Carlini and Terzis. Specifically, for target classes Rottweiler, Komondor, and Lorikeet, their attack 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. As a result, Carlini and Terzis can not semantically associate the target class name with objects in the target class, leading to poor attack performance.

G. Potential Limitations

Our attack relies on some reference images/objects being correctly classified by the downstream classifier. Since we consider the worst-case attack where the attacker only has a few reference objects (e.g., 3), our attack may fail if these reference objects mainly contain common features shared by different classes. The proposed CorruptEncoder+ uses more reference images (i.e., support reference images) to improve the attack performance. We believe it's an interesting future work to explore what makes a good reference image/object.

Table 12. ASRs (%) of CorruptEncoder with different pre-training datasets. Following our default setting, the target downstream task is ImageNet100-B.

	Another-ImageNet100		MSCOCO		SUN397	
	BA	ASR	BA	ASR	BA	ASR
MoCo-v2	52.1	96.9	56.9	93.4	48.4	93.2
MSF	54.1	91.0	58.0	93.5	49.6	99.8

Table 13. Detection performance of PatchSearch. TPR (%) indicates the fraction of poisoned pre-training images filtered out by PatchSearch. ASR (%) indicates the original attack performance.

SSL-Backdoor		Ours (Patch)		Ours (Blended)	
ASR	TPR	ASR	TPR	ASR	TPR
14.3	83.7	89.9	37.1	71.7	2.7

H. Additional Results

H.1. The advantage of attacking a target class

Our attack aims to backdoor a pre-trained encoder such that **any downstream classifier** built based on the backdoored encoder will predict trigger-embedded images as the target class as long as the downstream task (e.g., arbitrary animal classification) contains the target class (e.g., dog). Different from backdoor attacks to supervised learning, an attacker does not need to know classes in the downstream task. In our threat model, the attacker only needs to randomly pick a target class and obtain a few (e.g., 3) reference images from it. After poisoning the pre-training dataset, the attacker can compromise any downstream task that contains the target class. In our experiments, we evaluate one downstream task for each choice of target class following the previous works. To better illustrate our idea, we randomly sample three downstream tasks containing the target class from ImageNet. The encoder compromised by our attack results in ASRs of 89.9%, 91.6%, and 90.1% for them, as shown in Table 11. In other words, our attack can simultaneously attack all three downstream tasks.

H.2. More Attack Results on Different Pre-training Datasets

Our attack is generalizable to different pre-training settings. In Figures 5(a) and 5(c) of our paper, we show that CorruptEncoder is agnostic to different sizes of pre-training datasets and CL algorithms. In Table 12, we further evaluate CorruptEncoder with diverse pre-training datasets, such as a non-object-centric dataset (MSCOCO) and a domain-specific dataset (SUN397). Our results show that different pre-training datasets have small impacts on the attack performance, while they produce pre-trained encoders with different clean utilities.

H.3. Defense Results against PatchSearch

PatchSearch [29] is the state-of-the-art defense to detect patch-based backdoor attacks. The key idea is to check if each sample in the pre-training dataset contains a small patch that behaves similarly to the trigger. In particular, it searches the whole pre-training dataset for trigger-embedded samples and removes them from the set.

Table 13 compares the detection performance of PatchSearch in filtering out poisoned pre-training images of different attacks. We observe that while our attack achieves a much larger ASR than SSL-Backdoor, the poisoned images are not easy to detect. In particular, only 37.1% of poisoned pre-training images are detected by PatchSearch. The reason is that the irrelevant backgrounds of our attack introduce diverse features to the trigger-embedded patches.

Moreover, we found that an adaptive attacker can extend CorruptEncdoer to bypass the PatchSearch. Instead of using a patch trigger, **we embed a large but relatively invisible trigger (e.g., Blended Attack [4]) within the rectangle region excluding the reference object**. From Table 13, only 2.7% (17 out 650) of poisoned images are detected for this advanced attack. We believe it's an interesting future work to develop a stronger defense that can defend pre-trained encoders against our attack for all types of triggers.