A. Characterizing Supervised Backdoor Attacks

In supervised learning, the backdoor attack associates the trigger r with the target label t via (implicitly) minimizing the objective defined in Eq (1). The success of this attack is often attributed to the model's excess capacity [39], which "memorizes" both the function that classifies clean inputs and that misclassifies trigger inputs. Note that Eq (1) does not specify any constraints on the representations of trigger inputs. Thus, while associated with the same class, the trigger-embedded and target-class inputs are not necessarily proximate in the feature space.

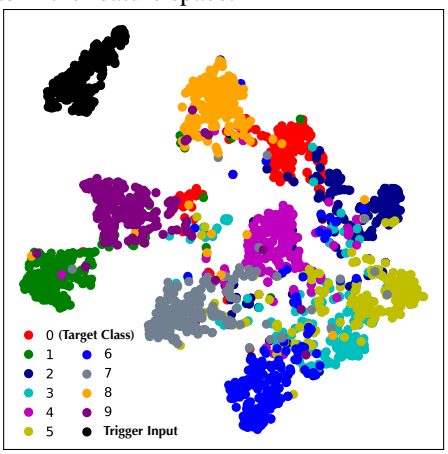

Figure 12: t-SNE visualization of the features of trigger-embedded and clean inputs in the supervised backdoor attack (target-class input: red; trigger-embedded input: black).

To validate the analysis, we use CTRL to generate the poisoning data, pollute 1% of the training data across all the classes, and train the model in a supervised setting for 20 epochs, which achieves 100% ASR and 83% ACC. We use t-SNE [45] to visualize the representations of trigger-embedded and clean inputs in the test set, with results shown in Figure 12. Although the clusters of trigger inputs (black) and target-class inputs (red) are assigned the same label, they are well separated in the feature space, indicating that supervised backdoor attacks do not necessarily associate the trigger with the target class in the feature space. This finding also corroborates prior work [42].

For comparison, we perform CTRL against SimCLR on CIFAR-10 and use t-SNE to visualize the features of trigger-embedded inputs and clean inputs in the test set, with results shown in Figure13. Observed that in comparison with the supervised backdoor attack (*cf.* Figure12), the cluster of trigger-embedded inputs (black) and the cluster of target-class clean inputs (red) are highly entangled in the feature space, indicating that the self-supervised backdoor attack takes effect by aligning the representations of trigger-embedded inputs and the target class.

B. Proofs

We first introduce the following two assumptions commonly observed in encoders trained using SSL [49]

Assumption B.1. (Alignment) A well-trained encoder f tends to map a positive pair to similar features. Formally, for given input x, $f(x)^\intercal f(x^+) \geq (1-\epsilon)$, where $\epsilon \in [0,1)$ is a small non-negative number. In particular, by design, the trigger r is invariant to the augmentation operator: $f(r)^\intercal f(r^+) = 1$.

Assumption B.2. (Uniformity) A well-trained encoder f tends to map inputs uniformly on the unit hyper-sphere $\mathcal{S}^{d-1}$ of the feature space, preserving as much information of the data as possible. Thus, as the number of data points is large, the average angle θ between the features of a negative pair follows the distribution density function [1]:

$$h(\theta) = \frac{1}{\sqrt{\pi}} \frac{\mathrm{T}(\frac{d}{2})}{\mathrm{T}(\frac{d-1}{2})} (\sin\theta)^{d-2}, \quad \theta \in [0,\pi] \tag{4}$$

As the dimension d is high, most of the angles heavily concentrate around $\pi/2$.

Based on B.1 and B.2, we now prove Theorem 5.1.

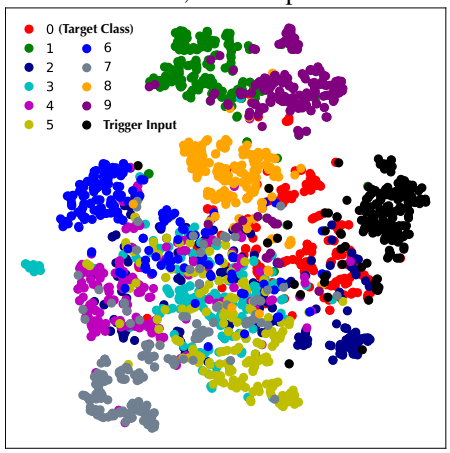

Figure 13: t-SNE visualization of the features of trigger-embedded and clean inputs in the self-supervised backdoor attack (target-class input: red; trigger-embedded input: black).

Proof of Theorem 5.1. According to Assumption B.1, we have

$$f(x_\star)^\intercal f(x_\star^+) \geq (1-\epsilon) \tag{5}$$

In other words,

$$(1-\alpha)^2 f(x)^\intercal f(x^+) + \alpha^2 f(r)^\intercal f(r^+) + \\ \alpha(1-\alpha)(f(x)^\intercal f(r^+) + f(r)^\intercal f(x^+)) \geq (1-\epsilon).$$

Since both $f(x)^\intercal f(x^+)$ and $f(r)^\intercal f(r^+)$ are no larger than 1, we have

$$\begin{aligned}&\alpha(1-\alpha)(f(x)^\intercal f(r^+) + f(r)^\intercal f(x^+))\\ &\geq (1-\epsilon) - (1-\alpha)^2 f(x)^\intercal f(x^+) - \alpha^2 f(r)^\intercal f(r^+)\\ &\geq (1-\epsilon) - (1-\alpha)^2 - \alpha^2\\ &= 2\alpha(1-\alpha) - \epsilon\end{aligned}$$

Then, based on Assumption B.1, we have

$$f(x)^\intercal f(r) \geq 1 - \frac{\epsilon}{2\alpha(1-\alpha)}. \tag{6}$$

For $\tilde{x}_*$ and x, we have

$$f(\tilde{x}_*)^\intercal f(x) = (1-\alpha) f(\tilde{x})^\intercal f(x) + \alpha f(r)^\intercal f(x)$$

Since $\tilde{x}$ and x are a negative pair, based on Assumption B.2, we have

$$\begin{aligned}\mathbb{E}[f(\tilde{x}_*)^\intercal f(x)] &\geq \alpha(1 - \frac{\epsilon}{2\alpha(1-\alpha)})\\ &\geq \alpha - \frac{\epsilon}{2(1-\alpha)}\end{aligned} \tag{7}$$

For a well-trained f, ϵ is a constant. Thus, both Eq (6) and Eq (7) are functions that first increase and then decrease with respect to $\alpha \in (0, 1)$.

□

C. Details of Experimental Setting

Dataset – For each dataset, we split it as a training set and a testing set according to its default setting. Specifically, both CIFAR-10 and CIFAR-100 are split into 50,000 and 10,000 images for training and testing, respectively; ImageNet-100 is split as 130,000 training and 5,000 testing images; while GTSRB is split into 39,209 training and 12,630 testing images.

Data augmentation – For convenience, we describe the details of data augmentations in a PyTorch style. Specifically, following prior work [7, 4], we use geometric augmentation operators including *RandomResizeCrop* (of scale [0.2, 1.0]) and *RandomHorizontalFlip*. Besides, we use *ColorJitter* with [brightness, contrast, saturation, hue] of strength [0.4., 0.4, 0.4, 0.1] with an application probability of 0.8 and *RandomGrayscale* with an application probability of 0.2.

Encoder training – We use the training set of each dataset to conduct contrastive learning. We show the hyper-parameters setting for each contrastive learning algorithm in Table 7, which is fixed across all the datasets.

Classifier training – Without explicit specification, we randomly sample 50 examples from each class of the corresponding testing set to train the downstream classifier. We show the hyper-parameters of classifier in Table 8.

Hyper-parameter	SSL Method		
	SimCLR	BYOL	SimSiam
Optimizer	SGD	SGD	SGD
Learning Rate	0.5	0.06	0.06
Momentum	0.9	0.9	0.9
Weight Decay	1e-4	1e-4	5e-4
Epochs	500	500	500
Batch Size	512	512	512
Temperature	0.5	-	-
Moving Average	-	0.996	-

Table 7. Hyper-parameters of encoder training.

Hyper-parameter	Setting
Optimizer	SGD
Batch Size	512
Learning Rate	0.2
Momentum	0.9
Scheduler	Cosine Annealing
Epochs	20

Table 8. Hyper-parameters of classifier training.

Evaluation – We evaluate ACC using the full testing set. For ASR, we apply CTRL on the full testing set and measure the ratio of trigger inputs that are classified to the target class. All the experiments are performed on a workstation equipped with Intel(R) Xeon(R) Silver 4314 CPU @ 2.40GHz, 512GB RAM, and four NVIDIA A6000 GPUs.

D. More Experimental Results

Here, we show the additional experimental results.

Performance of clean models – The ACC and ASR of clean models trained by SimCLR, BYOL, and Simsiam are summarized in Table 9.

Dataset	SSL Method					
	SimCLR		BYOL		SimSiam	
	ACC	ASR	ACC	ASR	ACC	ASR
CIFAR-10	79.1%	9.93%	82.4%	12.2%	81.5%	11.75%
CIFAR-100	48.1%	1.14%	51.0%	0.46%	52.0%	0.72%
ImageNet-100	42.2%	1.59%	45.1%	1.41%	41.3%	1.53%

Table 9. Accuracy of different SSL methods under normal training.

Fine-tuning data size – Typically, equipped with the pre-trained encoder, the victim fine-tunes the downstream classifier with a small labeled dataset. Here, we evaluate the impact of this fine-tuning dataset on CTRL. Figure 14 illustrates the performance of CTRL as a function of the number of labeled samples per class.

Observe that both ACC and ASR of CTRL increase with the fine-tuning data size, while their variance decreases gradually. For instance, when the number of labeled samples per class is set as 50, the ASR of CTRL on CIFAR-10 under SimSiam stably remains around 75%. This may be explained as follows. Without the supervisory signal of labeling, CTRL achieves effective attacks by entangling the representations

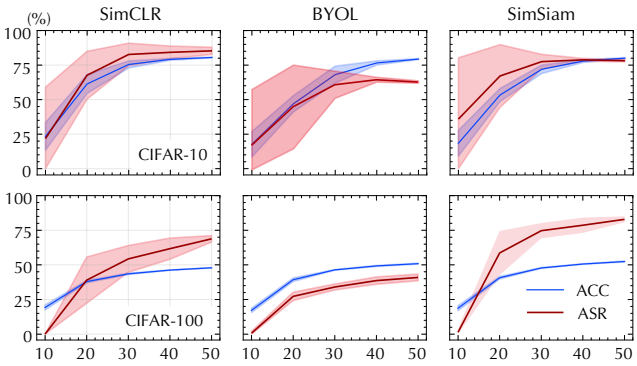

Figure 14: Performance of CTRL w.r.t. the fine-tuning data size.

of trigger-embedded and target-class inputs (details in § 5). During fine-tuning, more labeled samples imply that the representations of trigger-embedded inputs are more likely to be associated with the target-class label, leading to higher and more stable ASR. In other words, more fine-tuning data not only improves the model's performance but also increases its attack vulnerability.

Batch size – Existing studies show that batch size tends to impact the performance of contrastive learning [7]. Here, we explore its influence on the performance of CTRL. Specifically, on the CIFAR-10, we measure the ACC and ASR of CTRL with the batch size varying from 128 to 512, with results shown in Figure 15.

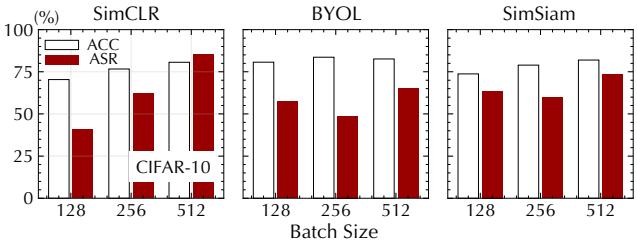

Figure 15: Performance of CTRL w.r.t. the batch size on CIFAR10.

Observe that the model's accuracy improves with the batch size, which corroborates the existing studies [7]. Moreover, a larger batch size (*e.g.*, ≥ 512) generally benefits the ASR of CTRL. This may be explained by that more positive pairs (also more negative pairs in SimCLR) in the same batch lead to tighter entanglement between trigger-embedded and target-class inputs. Meanwhile, for smaller batch sizes (*e.g.*, ≤ 256), the three SSL methods show slightly different trends. This may be attributed to the design of their loss functions: BOYL and SimSiam only optimize positive pairs, while SimCLR optimizes both positive and negative pairs, thereby gaining more benefits from larger batch sizes.

Training epochs – Typically, SSL benefits from more training epochs [4]. We evaluate the impact of training epochs on the performance of CTRL. Figure 16 shows the ACC and ASR of CTRL as the number of epochs varies from 600 to 1,000.

Observe that as the training epoch increases, the ACC

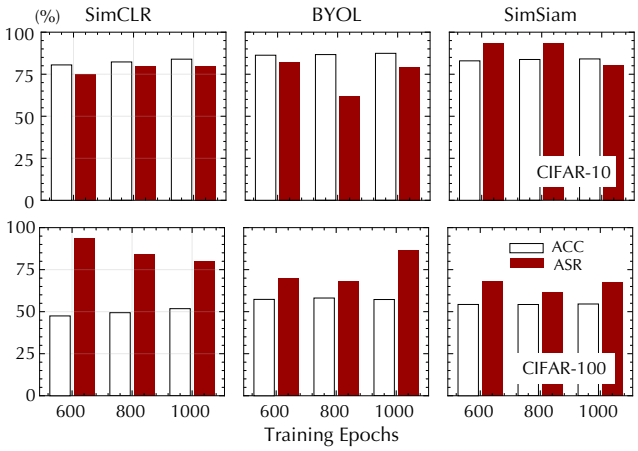

Figure 16: Performance of CTRL w.r.t. the number of epochs.

of CTRL gradually grows, while the ASR remains at a high level. For example, on CIFAR-10 with SimCLR, when the number of epochs increases from 600 to 1,000, the ACC also increases from 80.52% to 83,94%, and the ASR remains above 75%. In a few cases, the ASR slightly drops. We speculate this is caused by the random data augmentations used in SSL. Side evidence is that on CIFAR-10 with BYOL, the ASR first slightly decreases and then remains above 80%. In general, the number of training epochs has a limited impact on the effectiveness of CTRL.