# Can't Steal? Cont-Steal! Contrastive Stealing Attacks Against Image Encoders

Zeyang Sha[†] Xinlei He[†] Ning Yu[‡] Michael Backes[†] Yang Zhang[†]

[†]*CISPA Helmholtz Center for Information Security* [‡]*Salesforce Research*

{zeyang.sha, xinlei.he, director, zhang}@cispa.de, ning.yu@salesforce.com

## Abstract

*Self-supervised representation learning techniques have been developing rapidly to make full use of unlabeled images. They encode images into rich features that are oblivious to downstream tasks. Behind their revolutionary representation power, the requirements for dedicated model designs and a massive amount of computation resources expose image encoders to the risks of potential model stealing attacks - a cheap way to mimic the well-trained encoder performance while circumventing the demanding requirements. Yet conventional attacks only target supervised classifiers given their predicted labels and/or posteriors, which leaves the vulnerability of unsupervised encoders unexplored.*

*In this paper, we first instantiate the conventional stealing attacks against encoders and demonstrate their severer vulnerability compared with downstream classifiers. To better leverage the rich representation of encoders, we further propose Cont-Steal, a contrastive-learning-based attack, and validate its improved stealing effectiveness in various experiment settings. As a takeaway, we appeal to our community's attention to the intellectual property protection of representation learning techniques, especially to the defenses against encoder stealing attacks like ours.* [1]

## 1. Introduction

Recent years have witnessed the great success of applying deep learning (DL) to computer vision tasks. Different from supervised DL models, self-supervised learning which transforms unlabeled data samples into rich representations, has gained more and more popularity.

Behind its powerful representation, it is non-trivial to obtain a state-of-the-art image encoder. For instance, SimCLR [6] uses 128 TPU v3 cores to pre-train a ResNet-50 encoder with a batch size of 4096. Therefore, many big companies provide cloud-based self-supervised learning encoder services for users. For instance, Cohere,[2] OpenAI,[3] and Clarifai[4] provide the embedding API of images and texts for commercial usage. There are many works [10, 26, 30] exploring security issues of encoder-based API. Therefore, it is a very important and urgent problem.

These kinds of service leave the possibility of *model stealing attacks* [5, 25, 28, 37, 44, 48, 50, 51]. In these attacks, the adversary aims to steal the parameters or functionalities of target models with only query access to them. A successful model stealing attack does not only threaten

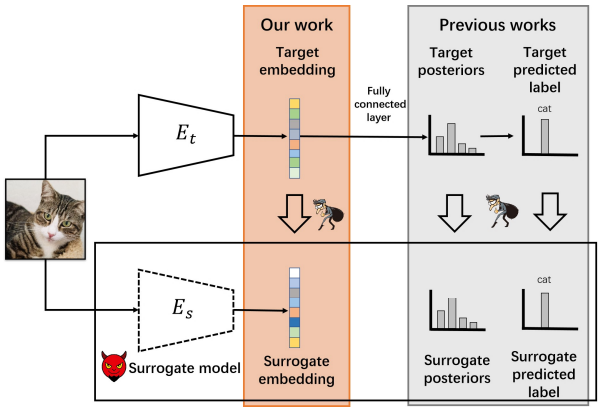



Figure 1. Model stealing attacks against classifiers (previous) v.s. model stealing attacks against encoders (ours). Previous works aim to steal a whole classifier using the predicted label or posteriors of a target model. In our work, we aim to steal the target encoder using its embeddings. The target encoder ($E_t$) is pre-trained and fixed, as shown in the solid frame. The surrogate encoder ($E_s$) is trainable by the adversary, as shown in the dashed frame.

[1] See our code in https://github.com/zeyangsha/Cont-Steal.

[2] https://cohere.ai/

[3] https://beta.openai.com/docs/api-reference/embeddings/

[4] https://www.clarifai.com/models/general-image-embedding/

the intellectual property of the target model, but also serves as a stepping stone for further attacks such as adversarial examples [3, 4, 16, 38, 47, 52], backdoor attacks [7, 26, 41, 43], and membership inference attacks [22–24, 30, 33, 34, 40, 42, 45, 46]. So far, model stealing attacks concentrate on the supervised classifiers, i.e., the model responses are prediction posteriors or labels for a specific downstream task. The vulnerability of unsupervised image encoders is unfortunately unexplored.

**Our Work.** To fill this gap, we pioneer the systematic investigation of model stealing attacks against image encoders. In this work, the adversary's goal is to steal the functionalities of the target model. See Figure 1 for an overview and a comparison with previous works. More specifically, we focus on encoders trained by *contrastive learning*, which is one of the most cutting-edge unsupervised representation learning strategies that unleash the information of unlabeled data.

We first instantiate the conventional stealing attacks against encoders and expose their vulnerability. Given an input image, the target encoder outputs its representation (referred to as embedding). Similar to model stealing attacks against classifiers, we consider the embedding as the "ground truth" label to guide the training procedure of a surrogate encoder on the adversary side. To measure the effectiveness of stealing attacks, we train an extra linear layer for the target and surrogate encoders towards the same downstream classification task. Preferably, the surrogate model should achieve both high classification accuracy and high agreement with the target predictions.

We evaluate our attacks on five datasets against four contrastive learning encoders. Our results demonstrate that the conventional attacks are more effective against encoders than against downstream classifiers. For instance, when we steal the downstream classifier pre-trained by SimCLR on CIFAR10 (with posteriors as its responses) using STL10 as the surrogate dataset, the adversary can only achieve an accuracy of 0.359. The accuracy, however, increases to 0.500 instead when we steal its encoder (with the embedding as its response).

Despite its encouraging performance, conventional attacks are not the most suitable ones against encoders. This is because they treat each image-embedding pair individually without interacting across pairs. Different embeddings are beneficial to each other as they can serve as anchors to better locate the position of the other embeddings in their space. Contrastive learning [6, 8, 17, 20, 27, 49, 53] is a straightforward idea to achieve this goal. It is formulated to enforce the embeddings of different augmentations of the same images closer and those of different images further.

In a similar spirit, we propose Cont-Steal, a contrastive-learning-based model stealing attack against the encoder. The goal of Cont-Steal is to enforce the surrogate embedding of an image close to its target embedding (defined as a positive pair) and also push away embeddings of different images irrespective of being generated by the target or the surrogate encoders (defined as negative pairs).

The comprehensive evaluation shows that Cont-Steal outperforms the conventional model stealing attacks to a large extent. For instance, when CIFAR10 is the target dataset, Cont-Steal achieves an accuracy of 0.714 on the SimCLR encoder pretrained on CIFAR10 with the surrogate dataset and downstream dataset being STL10, while the conventional attack only achieves 0.457 accuracy. Also, Cont-Steal is more query-efficient and dataset-independent (see Figure 9 for more details). This is because Cont-Steal leverages higher-order information across samples to mimic the functionality of the target encoder. To mitigate the attacks, we evaluate different defense mechanisms including noise, top-$k$, rounding, and watermark. Our evaluations show that in most cases, these mechanisms cannot effectively defend against Cont-Steal. Among them, top-$k$ can reduce the attack performance to the largest extent. However, it also strongly limits the target model's utility.

As a takeaway, our attack further exposes the severe vulnerability of pre-trained encoders. We appeal to our community's attention to the intellectual property protection of representation learning techniques, especially to the defenses against encoder stealing attacks like ours.

## 2. Threat Model

In this work, for the encoder pre-trained with images, we consider image classification as the downstream task. We refer to the encoder as the target encoder. Then we treat both the encoder and the linear layer trained for the downstream task together as the target model. We first introduce the adversary's goal and then characterize different background knowledge that the adversary might have.

**Adversary's Goal.** Following previous work [25, 28, 44], we taxonomize the adversary's goal into two dimensions, i.e., theft and utility. The theft adversary aims to build a surrogate encoder that has similar performance on the downstream tasks as the target encoder. Different from the thief adversary, the goal of the utility adversary is to construct a surrogate encoder that behaves normally on different downstream tasks. In this case, the surrogate encoder not only faithfully "copies" the behaviors of the target encoder, but also serves as a stepping stone to conduct other attacks.

**Adversary's Background Knowledge.** We categorize the adversary's background knowledge into two dimensions, i.e., the knowledge of the target encoder and the distribution of the surrogate dataset.

Regarding knowledge of the target encoder, we assume that the adversary only has black-box access to it, which means that they can only query the target encoder with an input image and obtain the corresponding output, i.e., the

embedding of the input image.

Regarding the surrogate dataset that is used to train the surrogate encoder, we consider two cases. First, we assume the adversary has the same training dataset as the target encoder. However, such an assumption may be hard to achieve as such datasets are usually private and protected by the model owner. In a more extreme case, we assume that the adversary has totally no information about the target encoder's training dataset, which means that they can only use a different distribution dataset to conduct the model stealing attacks. We later show that the adversary can still launch effective model stealing attacks against the target encoder given a surrogate dataset that is distributed differently compared to the target dataset.

For the model architecture that is used to train the surrogate encoder, we consider two cases. First, we assume the adversary is aware of the target encoder's architecture and can train the same architecture surrogate encoder. Then we relax our assumption that the adversary uses different architectures to train the surrogate encoder. Our evaluation shows that the choice of architecture does not have much impact on the attack performance (see Table 3), which makes the attack more realistic.

Note that we also compare our attacks against the encoders to the traditional model stealing attacks that focus on the whole classifier (which has an encoder and a linear layer). If the attack targets a whole classifier, we assume the adversary may obtain the posteriors or the predicted label for an input image.

# 3. Model Stealing Attacks

In this section, we first describe the conventional attacks against the encoders. Then, we propose a novel contrastive stealing framework, Cont-Steal, to steal the encoders more effectively.

## 3.1. Conventional Attacks Against Encoders

The adversary takes two steps to conduct the model stealing attacks against the target encoder and one step for further evaluation.

**Obtain the Surrogate Dataset.** To conduct model stealing attacks, the adversary first needs to obtain a surrogate dataset. Based on the knowledge of the target classifier's training dataset (target dataset), we consider two cases. If the adversary has full knowledge of the target dataset, they can directly leverage the target dataset itself as the surrogate dataset. Or the adversary has no knowledge of the target dataset, which means that they can only construct the surrogate dataset, which is distributed differently from the target dataset.

**Train the Surrogate Encoder.** Slightly different from the classifier, the response of the encoder is an embedding, which is a feature vector. In this case, the adversary can still leverage a similar loss function to optimize the surrogate encoder, which can be defined as follows:

$$L_{MS} = \sum_{k=1}^{N} l(E_T(x_k), E_S(x_k)) \tag{1}$$

where $E_T(\cdot)/E_S(\cdot)$ is the target/surrogate encoder, $N$ is the total number of samples on the surrogate dataset, and $l(\cdot)$ is the MSE loss.

**Apply the Surrogate Encoder to Downstream Tasks.** To evaluate the effectiveness of model stealing attacks against the encoder, the adversary can leverage the same downstream task to both the target and surrogate encoders. Concretely, the adversary trains an extra linear layer for the target and surrogate encoders, respectively. Note that we refer to the target/surrogate encoder and the extra linear layer as the target/surrogate classifiers. Then, the adversary quantifies the attack effectiveness by measuring the performance of the target/surrogate classifier on the downstream tasks.

## 3.2. Cont-Steal Attacks Against Encoders

To better leverage the rich information from the embeddings, we propose Cont-Steal, a contrastive learning-based model stealing attacks against encoders, which leverages contrastive learning to enhance the stealing performance. Concretely, Cont-Steal aims to enforce the surrogate embedding of an image to get close to its target embedding (defined as a positive pair), and also push away embeddings of different images regardless of being generated by the target or the surrogate encoders (defined as negative pairs). There are three steps for the adversary to conduct contrastive stealing attacks against encoders and one step for further evaluation.

**Obtain the Surrogate Dataset.** The adversary follows the same strategy as Section 3.1 to obtain the surrogate dataset.

**Data Augmentation.** Our proposed Cont-Steal leverages data augmentation to transform an input image into its two augmented views. In this paper, we leverage RandAugment [11] as the augmentation method, which is made up of a group of advanced augmentation operations. Concretely, we set $n = 2$ and $m = 14$ following Cubuk et al. [11] where $n$ denotes the number of transformations to a given sample and $m$ represents the magnitude of global distortion.

**Train the Surrogate Encoder.** Instead of querying the encoders with the original images, the adversary queries the encoders with the augmented views of them. Concretely, for an input image $x_i$, we generate two augmented views of it, i.e., $\widetilde{x}_{i,s}$ and $\widetilde{x}_{i,t}$, where $\widetilde{x}_{i,s}/\widetilde{x}_{i,t}$ is used to query the surrogate/target encoder. We consider $(\widetilde{x}_{i,s}, \widetilde{x}_{j,t})$ as a positive pair if $i = j$, and otherwise a negative pair.

Given a mini-batch of $N$ samples, we generate $N$ augmented views as the input of the target encoder and another $N$ augmented views as the input of the surrogate encoders.

Concretely, the loss of Cont-Steal can be formulated as follows:

$$D^{+}_{encoder}(i) = exp(sim(E_S(\widetilde{x}_{i,s}), E_T(\widetilde{x}_{i,t}))/\tau)), \tag{2}$$

$$D^{-}_{encoder}(i) = \sum_{k=1}^{N}(exp(sim(E_S(\widetilde{x}_{i,s}), E_T(\widetilde{x}_{k,t}))/\tau)), \tag{3}$$

$$D^{-}_{self}(i) = \sum_{k=1}^{N}\mathbb{1}_{[k\neq i]}(exp(sim(E_S(\widetilde{x}_{i,s}), E_S(\widetilde{x}_{k,s}))/\tau)), \tag{4}$$

$$l(i) = -log\frac{D^{+}_{encoder}(i)}{D^{-}_{encoder}(i) + D^{-}_{self}(i)}, \tag{5}$$

$$L_{Cont-Steal} = \frac{\sum_{k=1}^{N} l(k)}{N}, \tag{6}$$

where $E_S(\cdot)$ and $E_T(\cdot)$ denotes the surrogate and target encoder, $sim(u, v) = u^T v/||u||||v||$ represents the cosine similarity between $u$ and $v$, and $\tau$ is parameter to control the temperature.

As illustrated in Figure 2, the conventional attack treats each embedding individually without interacting across pairs. However, different embeddings are beneficial to each other as they can serve as anchors to better locate the position of the other embeddings in their space. Cont-Steal maximizes the similarity of embeddings generated from the target and surrogate encoders for a positive pair $(\widetilde{x}_{i,s}, \widetilde{x}_{i,t})$ (orange arrows in Figure 2). For the embedding generated from the target and surrogate encoders for any pair $(\widetilde{x}_{i,s}, \widetilde{x}_{j,t})$, contrastive stealing aims to make them more distant (green arrows in Figure 2). Besides, as pointed out by Chen et al. [6], contrastive learning benefits larger negative samples. To achieve this goal, we also consider the embeddings generated from the surrogate encoder for augmented views of different images, i.e., $(\widetilde{x}_{i,s}, \widetilde{x}_{j,s})$, as negative pairs minimize their similarity (blue arrows in Figure 2). We later show that such design can enhance the performance of contrastive stealing (see Table 5).

**Apply the Surrogate Encoder to Downstream Tasks.** We follow Section 3.1 to evaluate the effectiveness of model stealing on downstream tasks.

# 4. Experiments

In this section, we first describe the experimental setup in Section 4.1. Then we show the performance of the target encoders on the downstream tasks. Next, we summarize the performance of conventional attacks against classifiers and encoders in Section 4.2. Lastly, we evaluate the performance of Cont-Steal and conduct ablation studies to demonstrate its effectiveness under different settings in Section 4.3.

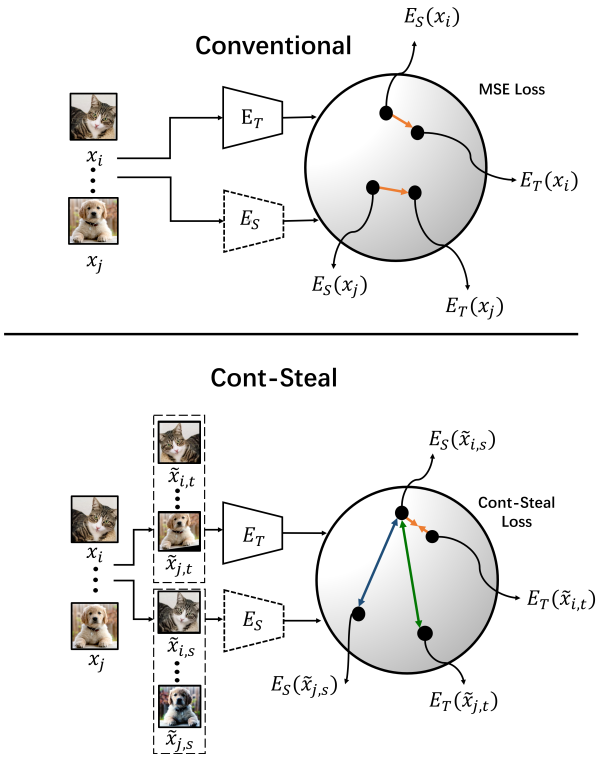


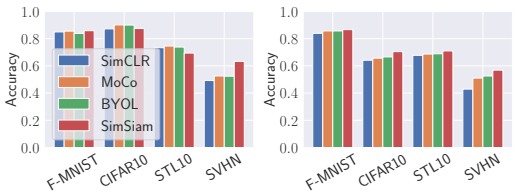

Figure 2. Conventional attack (top) vs. Cont-Steal (bottom) against encoders. Conventional attack applies MSE loss to approximate target embeddings for each sample individually. Cont-Steal (bottom) introduces data augmentation and interacts across multiple samples: associating target/surrogate embeddings of the same images closer and repulsing those of different images farther away. The target encoder ($E_t$) is pre-trained and fixed, as shown in the solid frame. The surrogate encoder ($E_s$) is trainable by the adversary, as shown in the dashed frame.

## 4.1. Experimental Setup

Our encoders are pre-trained on CIFAR10 [1], and ImageNet [12]. We use four different kinds of contrastive methods: SimCLR [6], MoCo [20], BYOL [17] and SimSiam [8] to train a ResNet18 [21] as our target encoders. Our implementation is based on a PyTorch framework of contrastive learning.[5] Then, these well pre-trained encoders will be applied to train downstream classifiers on CIFAR10 [1], STL10 [9], Fashion-MNIST [54], and SVHN [35]. In the experiments in the model stealing section, we use CIFAR10, STL10, Fashion-MNIST, and SVHN to conduct the attack.

Agreement and accuracy are used as metrics to evaluate the model stealing attack's performance. The agreement will evaluate the similarity of surrogate encoders and target encoders in downstream tasks. The accuracy will evaluate

[5] https://github.com/vturrisi/solo-learn/




(a) CIFAR10 (b) ImageNet100

Figure 3. The performance of target classifiers composed by target encoder and an extra linear layer. The encoders are pre-trained on CIFAR10 (a) and ImageNet100 (b). The x-axis represents different downstream datasets for the target encoder and classifier. The y-axis represents the target model's accuracy on downstream tasks.

the utility of surrogate models on downstream tasks. For each metric, a larger value is more desirable. During the stealing process, we set the batch size as 128 and the learning rate as 0.001. We show more results on the impact of hyperparameters in Supplementary Material in Section A.2

### 4.2. Performance of Conventional Attacks

We first show the target encoder's performance in various downstream tasks. The results are summarized in Figure 3. We conduct our experiments to explore whether the encoders are more vulnerable to model stealing attacks. We show our results of target encoders and downstream classifiers both trained on CIFAR10 in Figure 4. In all cases, the adversary can get better attack performance by stealing encoders rather than classifiers. This gap becomes especially apparent when the adversary has absolutely no knowledge of the train data. This is because the rich information in embeddings can better facilitate the learning process of surrogate encoders. For instance, when the surrogate dataset is CIFAR10 (the same as the target downstream dataset), stealing SimCLR's embeddings can achieve $0.785$ agreement, while stealing predicted labels can achieve $0.712$ agreement. However, when the surrogate dataset is totally different from the downstream target dataset, e.g., SVHN, stealing embeddings from SimCLR can still achieve $0.507$ agreement while the agreement of stealing predicted labels drops to $0.192$. We show more results in Supplementary Material Section A.4 due to the page limitation.

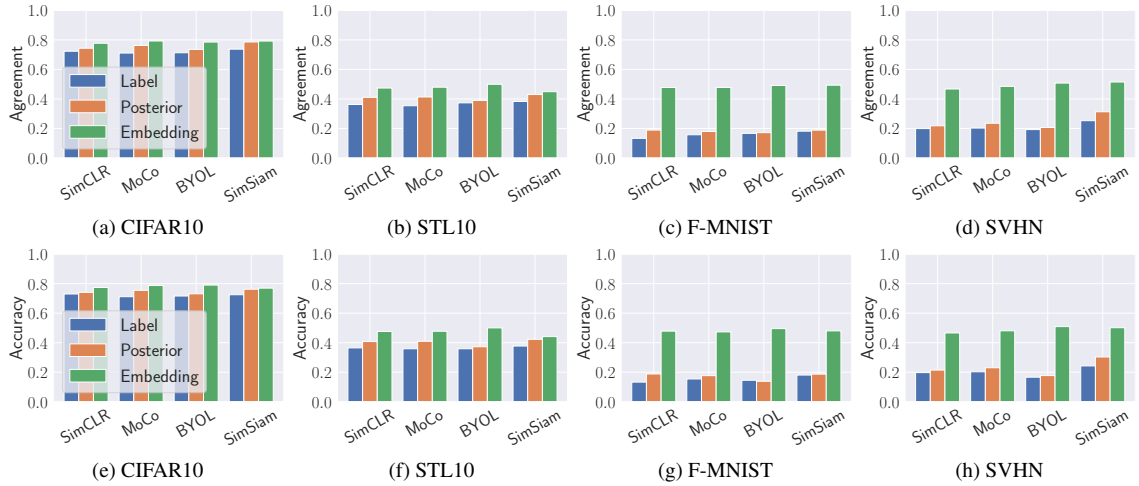
We also find that all model stealing attacks' accuracy and agreement are highly correlated. As shown in Figure 5, the agreement is highly correlated with the accuracy. This indicates that besides accuracy, the agreement can also be used as a metric to evaluate the performance of model stealing attacks. We show the result on Figure 5. It can be obviously seen that agreement is highly related to accuracy. We use the linear regression method to describe the relationship between agreement and accuracy and find that the relation function is y=0.940 * x.

### 4.3. Performance of Cont-Steal

As shown in Section 4.2, encoders are more vulnerable to model stealing attacks since the embedding usually contains richer information compared to the predicted label or posteriors. We then show that our proposed Cont-Steal can achieve better attack performance by making deeper use of embeddings' information.

Figure 7 shows the attack performance when the target pre-training dataset is CIFAR10. Note that we also show the attack performance on other settings in the appendix. We discover that compared to conventional attacks against encoders, Cont-Steal can consistently achieve better performance. For instance, as shown in Figure 7d, when the target encoder is MoCo trained on CIFAR10, if the adversary uses STL10 to conduct model stealing attacks against encoders, the surrogate encoder can achieve $0.841$ agreement in CIFAR10 downstream tasks with the Cont-Steal but only $0.479$ with conventional attacks. Another finding is that compared to the same distribution surrogate dataset, our Cont-Steal can better enhance the performance when the surrogate dataset comes from a different distribution from the pre-trained dataset. For instance, when the target encoder is SimCLR trained on CIFAR10, Cont-Steal outperforms conventional attack by $0.055$ agreement when the surrogate dataset is also CIFAR10, while the improvement increases to $0.207$ and $0.214$ when the surrogate dataset is STL10. We show more comparing results in Supplementary Material Section A.5. Note that our Cont-Steal also has great performance on other recent state-of-the-art visual models (ViT [13], MAE [19], and CLIP [39]), as we show in Section A.6, and can have better performance than other recent similar attacks [14, 31] shown in Section A.7.

To better understand why Cont-Steal can always achieve better performance, we extract samples' embeddings generated by different encoders, i.e., the target encoder, surrogate encoder trained with the conventional attack, and surrogate encoder trained with the Cont-Steal, and project them into a 2-dimensional space using t-SNE. From the results summarized in Figure 6, we find that Cont-Steal can effectively mimic the pattern of the embeddings as the target encoder. However, the conventional attack fails to capture such patterns for a number of input samples, e.g., the outer circle in Figure 8c. This further demonstrates that Cont-Steal benefits from jointly considering different embeddings as they can serve as anchors to better locate the position of the other embeddings in their space. We also show some ablation study results on Supplementary Material Section A.2 to show that with the less surrogate dataset, less training epoch, and different model architecture, Cont-Steal can still achieve much better results than conventional steal. Also, we show further attacks based on the stole models on Supplementary Material Section A.3 to show that Cont-Steal can be used as a springboard for other attacks.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

Figure 4. The performance of model stealing attack against target encoders and downstream classifiers both trained on CIFAR10. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), and SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.

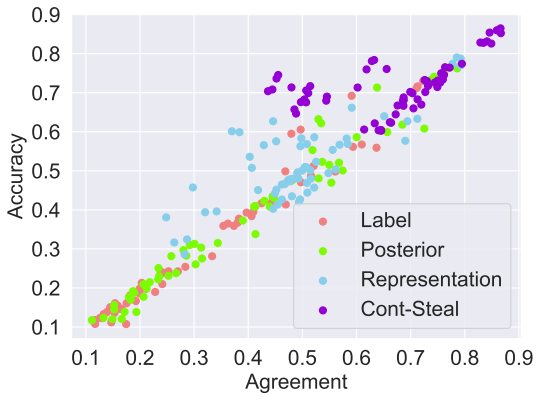



Figure 5. The relationship between accuracy and agreement. The x-axis is the agreement number, and the y-axis is the accuracy number.

(a) Target encoder (b) Cont-Steal (c) Conventional

Figure 6. The t-SNE projection of $1,000$ randomly selected samples' embeddings from target encoder, surrogate encoder under Cont-Steal, and surrogate encoder under the conventional attack, respectively. Note that the target encoder is pre-trained by SimCLR on CIFAR10.

Table 1. The monetary and (training) time costs for normal training and Cont-Steal attack. Cont-Steal's monetary cost contains two parts: query cost and training cost. Note that we ignore the query time cost of Cont-Steal as it normally has a smaller value than the training time cost.

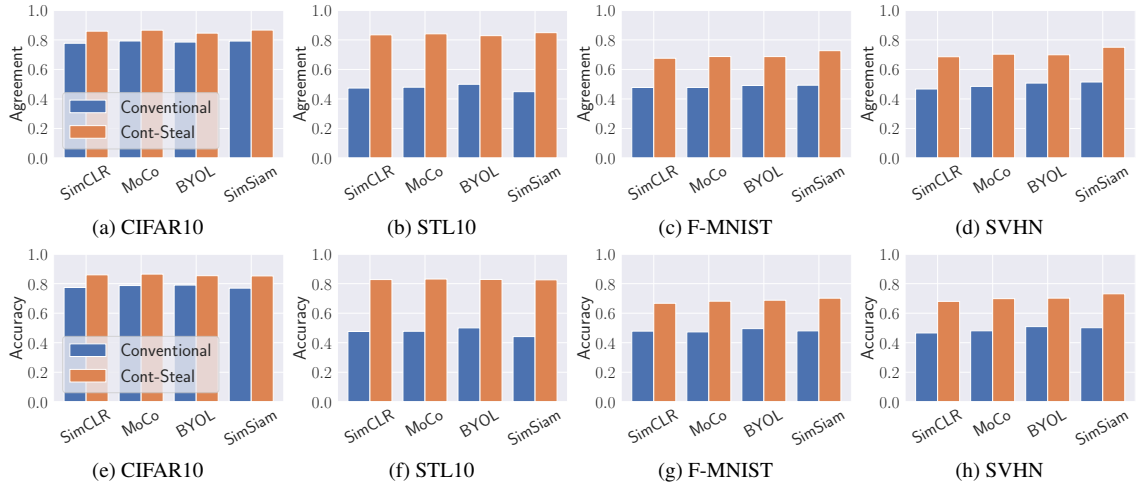
| | Monetary Cost | | Time Cost | |
|---|---|---|---|---|
| **Model** | **Normal ($)** | **Cont-Steal ($)** | **Normal (h)** | **Cont-Steal (h)** |
| SimCLR | 58.68 | 11.83 (1.83 + 10) | 20.01 | 0.62 |
| MoCo | 54.83 | 12.13 (2.13 + 10) | 18.69 | 0.73 |
| BYOL | 61.46 | 12.08 (2.08 + 10) | 20.96 | 0.71 |
| SimSiam | 57.14 | 12.00 (2.00 + 10) | 19.46 | 0.68 |

### 4.4. Cost Analysis

As we mentioned before, pre-train a state-of-the-art encoder is time-consuming and resource-demanding. We wonder if the model stealing attacks can steal the functionality of the encoder with much less cost. To this end, we evaluate the time and monetary cost of training an encoder from scratch or stealing a pre-trained encoder via Cont-Steal. The monetary cost of model stealing includes querying the target model and training the surrogate model. We refer to the query price as $1 for 1,000 queries based on AWS.[6] Our experiment is conducted on 1 NVIDIA A100 whose price is $2.934 per hour based on google cloud.[7]

The monetary and time cost is shown in Table 1. We observe that Cont-Steal can obtain a surrogate encoder with much less money and time cost than training the encoder from scratch. For instance, a ResNet18 trained by SimCLR

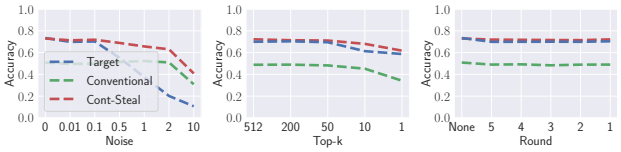
[6] https://aws.amazon.com/rekognition/pricing/
[7] https://cloud.google.com/compute/gpus-pricing/




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

Figure 7. The performance of Cont-Steal and conventional attack against target encoders trained on CIFAR10. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses CIFAR10 as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.




(a) Adding noise (b) Top-$k$ (c) Rounding

Figure 8. The performance of different defense methods. Target encoders are trained on CIFAR10. The downstream dataset and surrogate dataset are both STL10. The x-axis represents different defense levels. The y-axis represents the model's accuracy.

on CIFAR10 takes 20.01 hours and 58.68$ on 1 NVIDIA A100 GPU, while Cont-Steal only takes 0.62 hours and 11.83$ to steal an encoder that performs similarly on downstream tasks. The results demonstrate that Cont-Steal is able to construct surrogate encoders that perform similarly to the target encoders but with much less time and monetary cost.

### 4.5. Defenses

In this section, we will consider different defenses against model stealing attacks on encoders to evaluate the robustness of our proposed attack. We divided all defenses into three categories: perturbation-based defense [37] and watermark-based defense [2].

**Perturbation-based Defense.** In this defense setting, the defender aims to perturb the output of the target model to limit the information the adversary can obtain. The common practice of this kind of defense includes adding noise [37], top-$k$ [37], and feature rounding [48].

Adding noise means that the defender will introduce noise value to the original output of the model. In our case, we consider adding Gaussian noise to the embeddings generated by the target encoder. We set the mean value to 0, and different noise levels represent different standard deviations of the Gaussian distribution. For Top-$k$, the defender will only output the first $k$ largest number of each embedding (and set the rest as 0). In this way, the high-dimensional information of the image contained in embeddings can be appropriately reduced. Regarding feature rounding, the defender will truncate the values in the embedding to a specific digit. As a case study, we consider a ResNet18 encoder pre-trained on CIFAR10 with SimCLR and take STL10 to train its downstream classifier. The experimental results are summarized in Figure 8. We can observe that while adding noise and top-$k$ can reduce the model stealing attacks' performance, it may also degrade the target model performance to a large extent. For instance, when the noise increases from 0 to 10, the attack performance of Cont-Steal decreases from 0.729 to 0.410, while the target encoder's performance drops from 0.734 to 0.098. On the other hand, rounding only has a limited effect on both target model performance and attack performance. This indicates that perturbation-based defense cannot defend against the encoder's model stealing attack effectively since they cannot reach a good trade-off between attack performance and model utility.

**Watermark-based Defense.** Watermark-based defense is also one of the most popular defense methods against model stealing attacks [2]. Watermark provides copyright protection by adding some specific identification to the target model. If the surrogate model is stolen from the watermarked target model, then ideally, it will contain the same watermark as well. Adi et al. [2] show that back-

Table 2. Watermark defense. Pretrain dataset and surrogate dataset are both CIFAR10. Watermark leverages a watermark rate (wr) to verify the ownership of target models. A higher wr denotes a better verification performance.

| Dataset | Target model (acc/wr) | Cont-Steal (acc/wr) | Baseline (acc/wr) |
|---|---|---|---|
| CIFAR10 | 0.864 / 0.998 | 0.769 / 0.130 | 0.871 / 0.095 |
| STL10 | 0.721 / 0.999 | 0.702 / 0.034 | 0.733 / 0.111 |
| SVHN | 0.501 / 0.999 | 0.535 / 0.303 | 0.492 / 0.103 |
| F-MNIST | 0.857 / 0.999 | 0.813 / 0.061 | 0.850 / 0.099 |

door technology can be used as the watermark to protect the model. In that sense, BadEncoder [26], a backdoor mechanism against the encoder, can be leveraged as a watermarking technology for our target encoder as well. The defenders first will train the watermarked (backdoored) encoder, where images with a certain trigger will cause misclassification. Then, if they find the surrogate model can also misclassify images with the same trigger, the defenders can claim ownership of the surrogate model.

In our experiments, we leverage BadEncoder to watermark the encoder pre-trained on CIFAR10 by SimCLR, and leverage different downstream datasets to perform different tasks. We assume a strong adversary that has the same downstream dataset as the surrogate dataset. Also, we consider the baseline cases where the trigger samples are fed into the clean model to calculate the watermark rate (wr). As shown in Table 2), the watermark cannot be preserved as the surrogate models constructed by Cont-Steal have similar wr as the baseline model. For instance, when the downstream dataset is CIFAR10, Cont-Steal builds a surrogate model with 0.769 accuracy while only 0.130 wr, which is close to the baseline model. This indicates that Cont-Steal can bypass the watermarking technique as it can reach similar utility while reducing the wr to a large extent. Note that there is also another work to protect contrastive learning models from model stealing attacks using dataset inference [15]. We show in Supplementary Material in Section A.8 that this kind of defense can be easily bypassed by Cont-Steal.

## 5. Related Work

**Contrastive Learning.** Contrastive learning is one of the most popular methods to train encoders. Current works [6, 8, 17, 20, 49, 53] propose different advanced contrastive learning algorithms. SimCLR, MoCo, BYOL, SimSiam are currently the mainstream frameworks of contrastive learning. Thus, we concentrate on them in this paper. There are many works on evaluating the security and privacy risks of contrastive learning. Previous works [23, 26, 30] propose membership inference attacks, attribution inference attacks, and backdoor attacks on contrastive learning. All proposed attacks show that contrastive-based models are vulnerable to popular attacks. Therefore, the security issues of self-supervised learning deserve more attention.

**Model Stealing Attack.** In model stealing, the adversary's goal is to steal part of the target model. Tramèr et al. [48] proposed the first model stealing attack against black-box machine learning API to steal its parameters. Wang et al. [50] proposed the first hyperparameter stealing attacks against ML models. Oh et al. [36] also tried to steal machine learning model's architectures and hyperparameters. Orekondy et al. [37] proposed knockoff nets, which aim at stealing the functionality of black-box models. Krishna et al. [28] formulated a model stealing attack against BERT-based API. Besides, Wu et al. [51] and Shen et al. [44] perform model stealing attacks against Graph Neural Networks. These works often have relatively strong assumptions, such as the model family is known and the victim's data is partly available while we conduct model stealing attacks against encoders and relax the above assumption.

## 6. Conclusion

In this paper, we conduct the first model stealing risk assessment towards image encoders. Our evaluation shows that the encoder is more vulnerable to model stealing attacks compared to the classifier. This is because the embedding provided by the encoder contains richer information than the posteriors or predicted labels from whole classifiers.

To better unleash the power from the embeddings, we propose Cont-Steal, a contrastive learning-based model stealing method against encoders. Concretely, Cont-Steal introduces different types of negative pairs as "anchors" to better navigate the surrogate encoder and learn the functionality of the target encoder. Extensive evaluations show that Cont-Steal consistently performs better than conventional attacks against encoders. And such an advantage is further amplified when the adversary has no information on the target dataset, a limited amount of data, and restricted query budgets. Our work points out that the threat of model stealing attacks against encoders is largely underestimated, which prompts the need for more effective intellectual property protection of representation learning techniques.

**Acknowledgments.** We thank all anonymous reviewers for their constructive comments. This work is partially funded by the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (funding number ZT-I-OO1 4) and by the European Health and Digital Executive Agency (HADEA) within the project "Understanding the individual host response against Hepatitis D Virus to develop a personalized approach for the management of hepatitis D" (D-Solve) (grant agreement number 101057917).

## References


[1] https://www.cs.toronto.edu/~kriz/cifar.html.

[2] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In *USENIX Security Symposium (USENIX Security)*, pages 1615–1631. USENIX, 2018.

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion Attacks against Machine Learning at Test Time. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 387–402. Springer, 2013.

[4] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 39–57. IEEE, 2017.

[5] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Model Extraction and Active Learning. *CoRR abs/1811.02054*, 2018.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning (ICML)*, pages 1597–1607. PMLR, 2020.

[7] Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. BadNL: Backdoor Attacks Against NLP Models with Semantic-preserving Improvements. In *Annual Computer Security Applications Conference (ACSAC)*, pages 554–569. ACSAC, 2021.

[8] Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15750–15758. IEEE, 2021.

[9] Adam Coates, Andrew Y. Ng, and Honglak Lee. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 215–223. JMLR, 2011.

[10] Tianshuo Cong, Xinlei He, and Yang Zhang. SSL-Guard: A Watermarking Scheme for Self-supervised Learning Pre-trained Encoders. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 579–593. ACM, 2022.

[11] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 18613–18624. NeurIPS, 2020.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[14] Adam Dziedzic, Nikita Dhawan, Muhammad Ahmad Kaleem, Jonas Guan, and Nicolas Papernot. On the Difficulty of Defending Self-Supervised Learning against Model Extraction. In *International Conference on Machine Learning (ICML)*. JMLR, 2022.

[15] Adam Dziedzic, Haonan Duan, Muhammad Ahmad Kaleem, Nikita Dhawan, Jonas Guan, Yannis Cattan, Franziska Boenisch, and Nicolas Papernot. Dataset Inference for Self-Supervised Models. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2022.

[16] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[17] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[18] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogério Schmidt

Feris. SpotTune: Transfer Learning Through Adaptive Fine-Tuning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4805–4814. IEEE, 2019.

[19] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked Autoencoders Are Scalable Vision Learners. *CoRR abs/2111.06377*, 2021.

[20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735. IEEE, 2020.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[22] Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. Node-Level Membership Inference Attacks Against Graph Neural Networks. *CoRR abs/2102.05429*, 2021.

[23] Xinlei He and Yang Zhang. Quantifying and Mitigating Privacy Risks of Contrastive Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 845–863. ACM, 2021.

[24] Bo Hui, Yuchen Yang, Haolin Yuan, Philippe Burlina, Neil Zhenqiang Gong, and Yinzhi Cao. Practical Blind Membership Inference Attack via Differential Comparisons. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2021.

[25] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High Accuracy and High Fidelity Extraction of Neural Networks. In *USENIX Security Symposium (USENIX Security)*, pages 1345–1362. USENIX, 2020.

[26] Jinyuan Jia, Yupei Liu, and Neil Zhenqiang Gong. BadEncoder: Backdoor Attacks to Pre-trained Encoders in Self-Supervised Learning. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022.

[27] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised Contrastive Learning. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[28] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P. Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves on Sesame Street! Model Extraction of BERT-based APIs. In *International Conference on Learning Representations (ICLR)*, 2020.

[29] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial Examples in the Physical World. *CoRR abs/1607.02533*, 2016.

[30] Hongbin Liu, Jinyuan Jia, Wenjie Qu, and Neil Zhenqiang Gong. EncoderMI: Membership Inference against Pre-trained Encoders in Contrastive Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021.

[31] Yupei Liu, Jinyuan Jia, Hongbin Liu, and Neil Zhenqiang Gong. StolenEncoder: Stealing Pre-trained Encoders in Self-supervised Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2115–212. ACM, 2022.

[32] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[33] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine Learning with Membership Privacy using Adversarial Regularization. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 634–646. ACM, 2018.

[34] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1021–1035. IEEE, 2019.

[35] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 2011.

[36] Seong Joon Oh, Max Augustin, Bernt Schiele, and Mario Fritz. Towards Reverse-Engineering Black-Box Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[37] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff Nets: Stealing Functionality of Black-Box Models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4954–4963. IEEE, 2019.

[38] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram

Swami. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy (Euro S&P)*, pages 372–387. IEEE, 2016.

[39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021.

[40] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs Black-box: Bayes Optimal Strategies for Membership Inference. In *International Conference on Machine Learning (ICML)*, pages 5558–5567. PMLR, 2019.

[41] Ahmed Salem, Michael Backes, and Yang Zhang. Don't Trigger Me! A Triggerless Backdoor Attack Against Deep Neural Networks. *CoRR abs/2010.03282*, 2020.

[42] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

[43] Zeyang Sha, Xinlei He, Pascal Berrang, Mathias Humbert, and Yang Zhang. Fine-Tuning Is All You Need to Mitigate Backdoor Attacks. *CoRR abs/2212.09067*, 2022.

[44] Yun Shen, Xinlei He, Yufei Han, and Yang Zhang. Model Stealing Attacks Against Inductive Graph Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1175–1192. IEEE, 2022.

[45] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium on Security and Privacy (S&P)*, pages 3–18. IEEE, 2017.

[46] Liwei Song and Prateek Mittal. Systematic Evaluation of Privacy Risks of Machine Learning Models. In *USENIX Security Symposium (USENIX Security)*. USENIX, 2021.

[47] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble Adversarial Training: Attacks and Defenses. In *International Conference on Learning Representations (ICLR)*, 2017.

[48] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security Symposium (USENIX Security)*, pages 601–618. USENIX, 2016.

[49] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *CoRR abs/1807.03748*, 2018.

[50] Binghui Wang and Neil Zhenqiang Gong. Stealing Hyperparameters in Machine Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 36–52. IEEE, 2018.

[51] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. Model Extraction Attacks on Graph Neural Networks: Taxonomy and Realization. *CoRR abs/2010.12751*, 2020.

[52] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial Examples for Graph Data: Deep Insights into Attack and Defense. In *International Joint Conferences on Artifical Intelligence (IJCAI)*, pages 4816–4823. IJCAI, 2019.

[53] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised Feature Learning via Non-Parametric Instance Discrimination. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3733–3742. IEEE, 2018.

[54] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR abs/1708.07747*, 2017.

[55] Ziqing Yang, Zeyang Sha, Michael Backes, and Yang Zhang. From Visual Prompt Learning to Zero-Shot Transfer: Mapping Is All You Need. *CoRR abs/2303.05266*, 2023.

# A. Supplementary Material

## A.1. Training Algorithm of Cont-Steal

**Algorithm 1:** The training process of Cont-Steal.

**input :** Surrogate training dataset $D_{surrogate}^{train}$, target encoder $E_t$, surrogate encoder $E_s$

*Initialize $E_s$'s parameters*;
**for** *each epoch* **do**
  **for** *each batch* **do**
    Sample a batch with N training data samples $x_1, x_2, \cdots, x_N$ from $D_{surrogate}^{train}$
    Generate augmented data samples: $(\widetilde{x}_{1,t}, \widetilde{x}_{1,s}), (\widetilde{x}_{2,t}, \widetilde{x}_{2,s}), \cdots, (\widetilde{x}_{N,t}, \widetilde{x}_{N,s})$, where $\widetilde{x}_{k,t}$ and $\widetilde{x}_{k,s}$ are the two augmented views of $x_k$
    Feed $\widetilde{x}_{k,t}$ to $E_t$ and $\widetilde{x}_k$ to $E_s$ to calculate the contrastive steal loss: $L_{Cont-Steal} = \frac{\sum_{k=1}^{N} l(k)}{N}$
    Optimize $E_s$'s parameters with the contrastive steal loss $L_{Cont-Steal}$
  **end**
**end**
**return** *Surrogate encoder $E_s$*

Algorithm 1 presents the training process of contrastive stealing. In each batch, given $N$ training samples, we first generate $2N$ augmented views and feed the target encoder and surrogate encoder with different views generated by the same samples. Then, we optimize the surrogate encoder by minimizing $L_{Cont-Steal}$.

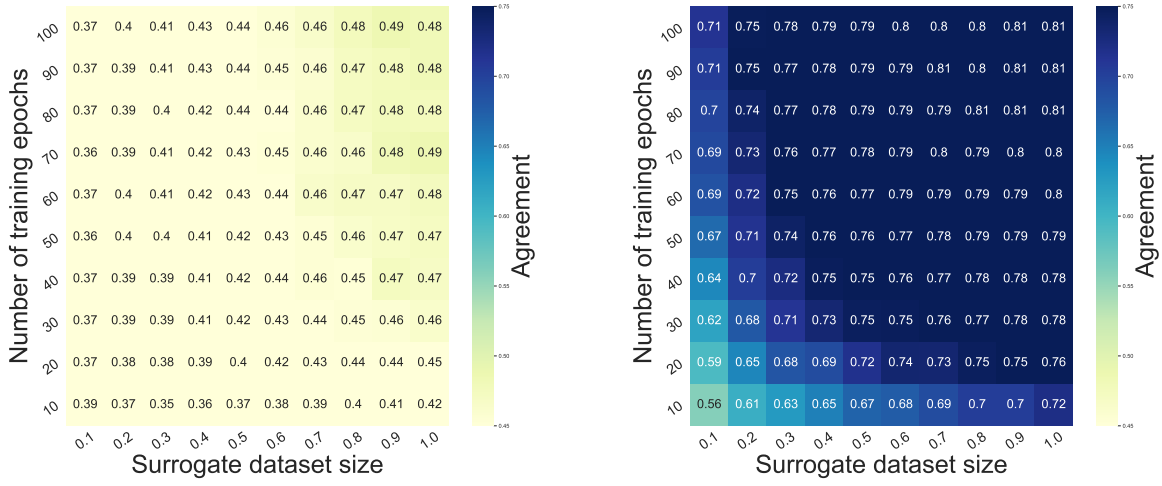
## A.2. Ablation Studies on Adversary Training Process

**Impact of Surrogate Encoder's Architecture.** Previous experiments are based on the assumption that the adversary knows the target encoder's architecture. We then investigate whether the attack against the encoder is still effective when the surrogate encoder has different model architectures compared to the target encoder. Concretely, we perform Cont-Steal against the ResNet18 encoder with the surrogate encoder's architecture as ResNet18, ResNet34, ResNet50, DenseNet161, and MobileNetV2, respectively. As shown in Table 3, we can see that the architecture of the surrogate model only has limited influence on the attack performance. For instance, the adversary can achieve 0.839 accuracy using the same architecture as the target model, while it can even achieve 0.840 accuracy when using a more complex model architecture (ResNet50) on SimCLR. The attack performance will drop a little if the adversary uses DenseNet161 and MobileNetV2. This might be because the architectures of DenseNet161 and MobileNetV2 have larger differences compared to ResNet18. However, the accuracy with DenseNet161/MobileNetV2 as the surrogate encoder's architecture can still achieve 0.828/0.811. This demonstrates that the model architectures of the surrogate encoder only have a limited impact on the attack performance, which makes the attack a more realistic threat.

Table 3. Cont-Steal attack performance of different surrogate architectures. Target encoders (ResNet18) and downstream classifiers are trained on CIFAR10. The surrogate dataset is also CIFAR10.

| Framework | Architectures | Agreement | Accuracy |
|---|---|---|---|
| SimCLR | ResNet18 | 0.835 | 0.839 |
| | ResNet34 | 0.837 | 0.842 |
| | ResNet50 | 0.844 | 0.840 |
| | DenseNet161 | 0.831 | 0.828 |
| | MobileNetV2 | 0.815 | 0.811 |
| MoCo | ResNet18 | 0.857 | 0.849 |
| | ResNet34 | 0.858 | 0.849 |
| | ResNet50 | 0.867 | 0.856 |
| | DenseNet161 | 0.813 | 0.811 |
| | MobileNetV2 | 0.796 | 0.801 |
| BYOL | ResNet18 | 0.845 | 0.842 |
| | ResNet34 | 0.850 | 0.847 |
| | ResNet50 | 0.857 | 0.855 |
| | DenseNet161 | 0.845 | 0.821 |
| | MobileNetV2 | 0.839 | 0.847 |
| SimSiam | ResNet18 | 0.856 | 0.835 |
| | ResNet34 | 0.858 | 0.839 |
| | ResNet50 | 0.860 | 0.848 |
| | DenseNet161 | 0.791 | 0.783 |
| | MobileNetV2 | 0.812 | 0.832 |

**Impact of Surrogate Dataset's Size and Surrogate Model's Training Epoch.** We conduct ablation studies here to better illustrate the effectiveness of Cont-Steal. Concretely, we investigate whether conventional attacks and Cont-Steal are still effective under limited surrogate dataset size and the number of training epochs. Ideally, we consider the attack that can reach similar performance but with less surrogate dataset size and fewer training epochs as a better attack as it requires less query and monetary costs. As shown in Figure 9, we observe that both conventional attacks and Cont-Steal can have better performance with a larger surrogate dataset size and more training epochs. For instance, Cont-Steal reaches 0.675 agreement when the surrogate encoder is trained with 10% surrogate dataset for 50 epochs, while the agreement increase to 0.812 with 100% surrogate dataset and 100 training epochs. The second observation is that Cont-Steal outperforms conventional attacks even with limited data and training epochs. For instance, even with only 10% surrogate dataset and 10 training epochs, the surrogate encoder built by Cont-Steal can reach 0.562 agreement, while the conventional attack can only achieve 0.479 agreement with the full surrogate dataset and 100 training epochs. As we mentioned before, this is because Cont-Steal can enforce the surrogate embedding of an image close to its target embedding and also push away




(a) Conventional (b) Cont-Steal

Figure 9. Heatmap of the agreement scores of model stealing attacks. The target model's encoder and downstream classifier are both ResNet18 trained by SimCLR on CIFAR10. The surrogate dataset is STL10. Surrogate dataset's size refers to the proportion of surrogate data we used for the whole surrogate dataset. We show the performance of 100 combinations of different training epochs and the surrogate gate dataset's size.

embeddings of different images irrespective of being generated by the target or the surrogate encoders (see also Table 5 for the necessity of introducing negative pairs from the surrogate encoder). This makes Cont-Steal a more effective model stealing attack against encoders.

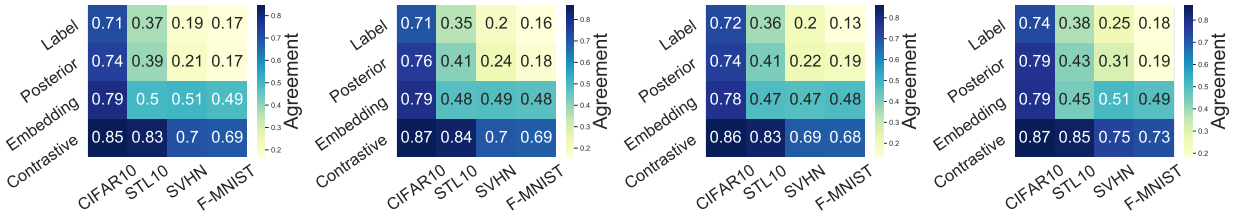
**Impact of Surrogate Dataset's Correlation With the Target Dataset.** In the meanwhile, since the adversary cannot always have knowledge about the target dataset, the impact of the surrogate dataset's correlation with the target dataset is also worth consideration. We find that Cont-Steal depends less on the surrogate dataset's distribution and can always achieve stable performance. We plot the attack agreement in Figure 10 where the target encoders and downstream classifiers are trained on CIFAR10. We can see that when the adversary conducts a conventional attack against the classifier, the adversary's knowledge of target training data is crucial. For example, when the adversary can only get the predicted label from the target model, he/she can only achieve 0.182 agreement when using F-MNIST to attack the model trained by SimCLR, while it can achieve 0.711 agreement when using CIFAR10 as the surrogate dataset, which is same as target dataset. However, compared to the predicted label or posterior as the response, embedding depends less on the surrogate dataset distribution, and Cont-Steal can better leverage the embedding information, contributing to the less dependent on the surrogate dataset's distribution. For instance, when the target model is trained by SimCLR, Cont-Steal can achieve 0.832 agreement when the surrogate dataset is STL10, which is even better than the best conventional attack (0.781) using the exact same target training dataset as the surrogate dataset and embedding as the response. Such observation better implies that Cont-Steal can always achieve good performance regardless of the surrogate dataset's distribution and can also achieve more generalized performance in practice.

Table 4. Impact of learning rate and batch size. The target dataset and downstream dataset are both CIFAR10. The surrogate dataset is STL10. Note that for different learning rates, we set the batch size as 128. For different batch sizes, we set the learning rate as 0.001

| Hyperparamter | Different Settings | Agreement |
|---|---|---|
| Learning Rate | 0.001 | 0.813 |
| | 0.002 | 0.801 |
| | 0.003 | 0.805 |
| | 0.004 | 0.819 |
| | 0.005 | 0.809 |
| Batch Size | 16 | 0.827 |
| | 32 | 0.806 |
| | 64 | 0.800 |
| | 128 | 0.813 |
| | 256 | 0.775 |

**Impact of Hyperparameters.** In our experiments, we set batch size as 128 and learning rate as 0.001. We show in Table 4 that with reasonable batch size and learning rate, our Cont-Steal can have stable performance.

**Impact of Negative Pairs Generated From the Surrogate Encoder.** In Cont-Steal's loss functions, besides $D^{-}_{encoder}$, we also consider the distance of negative pairs generated




(a) SimCLR (b) MoCo (c) BYOL (d) SimSiam

Figure 10. Heatmap of the agreement scores of model stealing attacks. We show the performance of 16 combinations of different information that the target model outputs and the adversary's knowledge of target training data. Target models are trained on CIFAR10.

from the surrogate encoder itself, i.e., $D^-_{self}$. To evaluate the necessity of $D^-_{self}$, we take the target encoder trained by BYOL on CIFAR10 and the downstream task on STL10 as an example and study the attack performance with and without $D^-_{self}$. The results are summarized in Table 5. We find that adding $D^-_{self}$ greatly improves the attack performance in both accuracy and agreement. For instance, when the surrogate dataset is STL10, the surrogate model stolen by Cont-Steal with $D^-_{self}$ achieves $0.817$ agreement while only $0.314$ if without $D^-_{self}$. The reason behind this is that the negative pairs generated from the surrogate encoder can serve as extra "anchors" to better locate the position of the embedding, which leads to higher agreement. Such observation demonstrates that it is important to introduce $D^-_{self}$ in Cont-Steal as well.

Table 5. The agreement and accuracy of different contrastive losses. We use BYOL trained on STL10 as the target model.

| Dataset | Method | BYOL | |
|---|---|---|---|
| | | Agreement | Accuracy |
| CIFAR10 | w/o $D^-_{encoder}$ | 0.242 | 0.242 |
| | w $D^-_{encoder}$ | 0.844 | 0.843 |
| F-MNIST | w/o $D^-_{encoder}$ | 0.215 | 0.217 |
| | w $D^-_{encoder}$ | 0.647 | 0.641 |
| STL10 | w/o $D^-_{encoder}$ | 0.314 | 0.320 |
| | w $D^-_{encoder}$ | 0.817 | 0.811 |
| SVHN | w/o $D^-_{encoder}$ | 0.176 | 0.175 |
| | w $D^-_{encoder}$ | 0.655 | 0.650 |

## A.3. Further Attacks Based on Cont-Steal

As we have mentioned in the introduction part, model stealing can be used as a stepping stone for further attacks. In this section, we select adversary sample attacks as a case study to show the importance of model stealing for further attacks on the target model. Normally, the adversary can not obtain the gradient from the target model. But to conduct adversary sample attacks, the adversary needs to obtain the gradient in most attack scenarios. Therefore, the adversary can construct a surrogate model to generate the adversary sample and transfer it to the target model to perform the attack. We consider three widely used mechanisms to generate adversarial examples, including Fast Gradient Sign Attack (FGSM) [16], Basic Iterative Methods (BIM) [29], and Projected Gradient Descent (PGD) [32]. Our target model is SimCLR pre-trained on CIFAR10 and the last layer classifier trained on STL10. We also use STL10 as the surrogate dataset to conduct Cont-Steal and generate adversary samples. Experiments show that the surrogate model can generate adversary samples that are valid for the target model (Table 6). To show the necessity of the surrogate model as a springboard for the attack, we also conduct the baseline attack, which uses another model as the springboard to attack the target model. We choose the normal ResNet18 model trained on SVHN as our baseline model and then apply the adversary example to attack the target model. We observe that compared to the adversarial examples generated from the baseline model, those adversarial examples generated from the surrogate model constructed by Cont-Steal can better transfer to the target model. For instance, with PGD, the adversarial examples obtained from the surrogate model can lead to a lower classification accuracy (0.203) on the target model than those generated from the baseline model (0.246). This implies that the model stealing attack can be a valid stepping stone for more effective further attacks.

Table 6. The different methods to create adversary sample to attack on surrogate model and target model. [Lower is better]

| Method | Surrogate model (acc) | Target model (acc) | Baseline (acc) |
|---|---|---|---|
| FGSM [16] | 0.097 | 0.131 | 0.194 |
| BIM [29] | 0.054 | 0.192 | 0.235 |
| PGD [32] | 0.092 | 0.203 | 0.246 |

## A.4. More Results on Conventional Attacks

Figure 11, Figure 12, and Figure 13 show the results of the conventional attacks on target models whose encoders are pre-trained on CIFAR10 and downstream classifiers are trained on STL10, F-MNIST, and SVHN, respectively. Figure 14, Figure 15, and Figure 16 show the results

of the conventional attacks on classifiers whose encoders are pre-trained on ImageNet100 and downstream classifiers are trained on STL10, F-MNIST, and SVHN, respectively.

## A.5. More Results on Cont-Steal

Figure 17, Figure 18, and Figure 19 show the results of the Cont-Steal on target models whose encoders are pre-trained on CIFAR10 and downstream classifiers are trained on STL10, F-MNIST, and SVHN, respectively. Figure 20, Figure 21, Figure 22 show the results of the Cont-Steal on target models whose encoders are pre-trained on ImageNet100 and downstream classifiers are trained on CIFAR10, STL10, and SVHN, respectively.

## A.6. Attacks Performance on Other Visual Models

Apart from four contrastive models we tried in the paper, we also conduct our Cont-Steal on other large, state-of-the-art models such as ViT and CLIP. We show that Cont-Steal can perform very well on ViT, MAE, and the image encoder of CLIP in Table 7. The results demonstrate the scalability of Cont-Steal.

## A.7. Compare With Other Existing Works

Note that we are the first work to systematically propose model stealing attacks against image encoders. There are also some parallel and follow-up works on this domain proposed after our work. Here, we compare our works with other existing methods. The main difference between our work and recent works is our designed contrastive steal loss and the usage of data augmentation. Compared to StolenEncoder [31], our loss focuses on the comparison of positive and negative samples, while StolenEncoder focuses on the combination of augmentation and non-augmentation loss. The main difference between our work and the methods listed in [14] is that 1) we leverage data augmentation as part of the methods. 2) we design the loss function ourselves to consider more negative examples compared to the INFONCE loss. We show in Table 9 that our method works better. Note that KL divergence is also a loss function used by knowledge distillation. As knowledge distillation is a similar task to model stealing, we also report the results of KL divergence.

Table 7. The performance of Cont-Steal and conventional attacks against state-of-the-art models. Note that all of our target encoders are pre-trained encoders available online and downstream classifiers are trained on CIFAR10.

| Surrogate Dataset | Metric | Attacks | ViT | MAE | CLIP |
|---|---|---|---|---|---|
| Original performance | Accuracy | NaN | 0.896 | 0.900 | 0.903 |
| CIFAR10 | Agreement | Conventional | 0.745 | 0.555 | 0.815 |
| | Agreement | Cont-Steal | **0.967** | **0.712** | **0.889** |
| STL10 | Agreement | Conventional | 0.553 | 0.451 | 0.550 |
| | Agreement | Cont-Steal | **0.942** | **0.624** | **0.905** |
| SVHN | Agreement | Conventional | 0.587 | 0.419 | 0.578 |
| | Agreement | Cont-Steal | **0.944** | **0.548** | **0.893** |
| F-MNIST | Agreement | Conventional | 0.602 | 0.395 | 0.465 |
| | Agreement | Cont-Steal | **0.696** | **0.501** | **0.598** |

Table 8. Dataset inference performance on Cont-Steal.

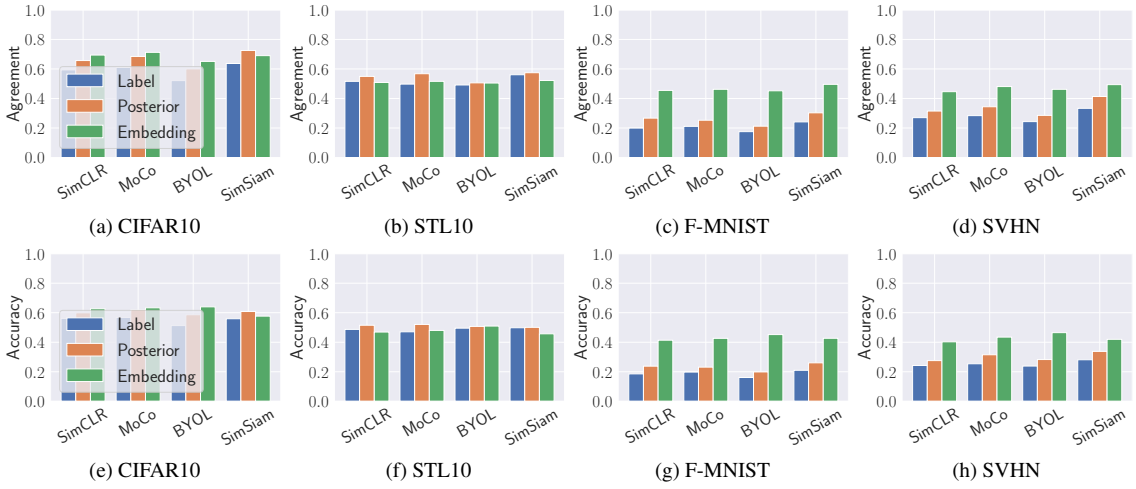
| Model | Dataset | $S(\cdot, E_T)$ | $C(\cdot, E_T)$ |
|---|---|---|---|
| Target Encoder | CIFAR10 | 1.000 | 1.000 |
| Surrogate Encoder | SVHN | 0.412 | 0.393 |
| Surrogate Encoder (fine-tuning) | SVHN | 0.17 | 0.00 |
| Independent Encoder | SVHN | 0.11 | 0.00 |

Table 9. The comparison of Cont-Steal and other existing works. Both the target encoder and downstream classifier are trained on CIFAR10. Note that our results are different from the original paper of [14] because we test the surrogate encoder on the original task.

| | CIFAR10 | | STL10 | |
|---|---|---|---|---|
| | Agreement | Accuracy | Agreement | Accuracy |
| Baseline | 0.785 | 0.790 | 0.499 | 0.500 |
| StolenEncoder | 0.811 | 0.808 | 0.766 | 0.767 |
| KL Divergence | 0.213 | 0.203 | 0.178 | 0.162 |
| INFONCE | 0.826 | 0.828 | 0.806 | 0.797 |
| Cont-Steal | **0.845** | **0.854** | **0.829** | **0.828** |

## A.8. More Defenses.

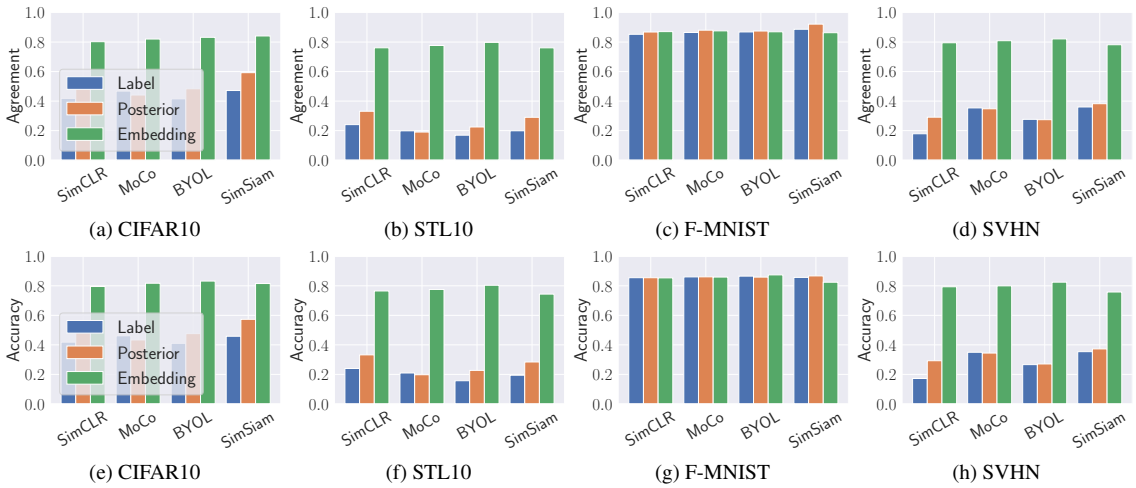
We implement the dataset inference defense in [15] (see Table 8). $S(\cdot, E_T)$/$C(\cdot, E_T)$ represents the mutual information/cosine similarity between the given model and the target model (the higher, the more similar). Note that the surrogate encoder will be fine-tuned for downstream tasks. We find the fine-tuning process [18, 55] will disable the defense. Normally, the open-source encoders are trained on very large public datasets instead of limited private datasets, which makes the defense less practical.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

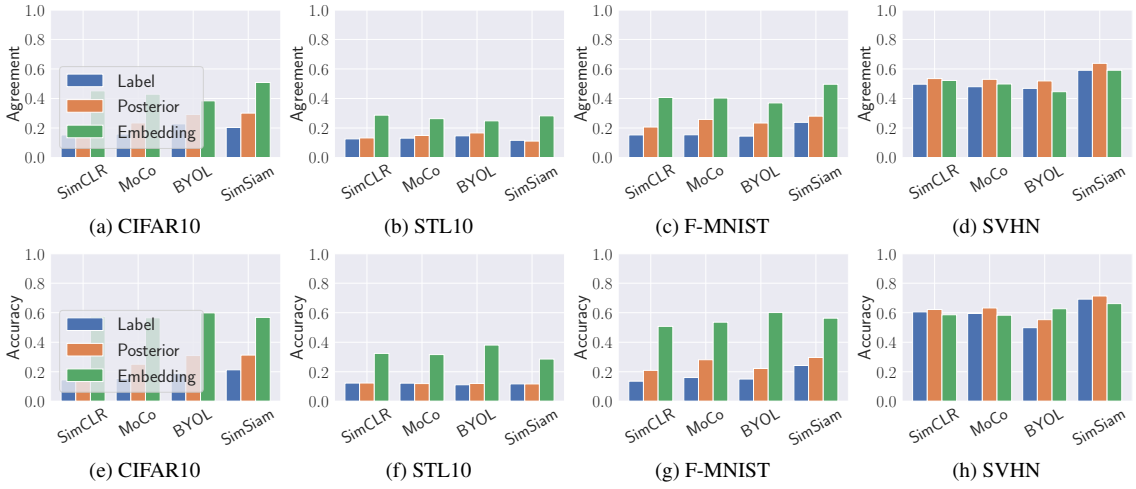
Figure 11. The performance of model stealing attack against target encodes and downstream classifiers trained on CIFAR10 and STL10. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

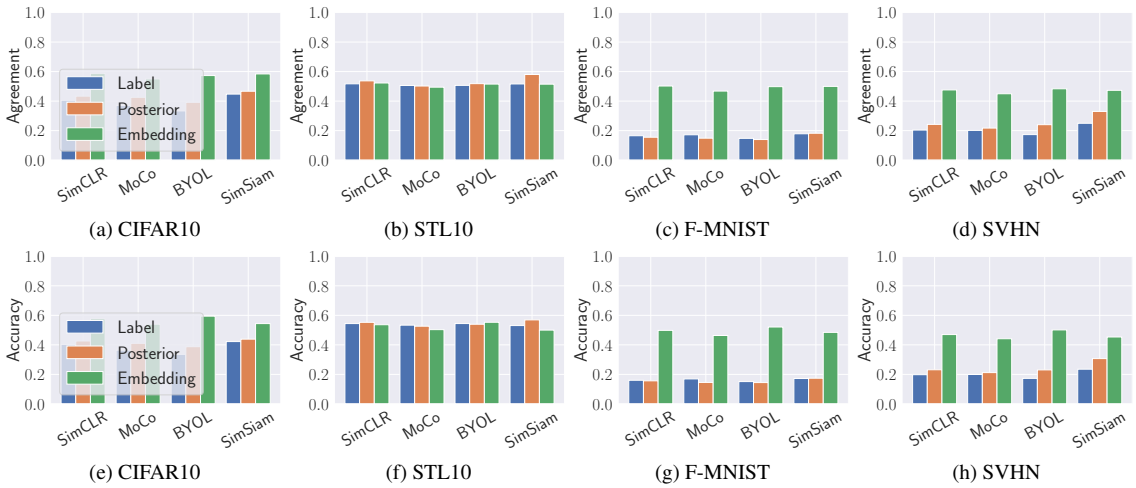
Figure 12. The performance of model stealing attack against target encodes and downstream classifiers trained on CIFAR10 and Fashon-MNIST. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

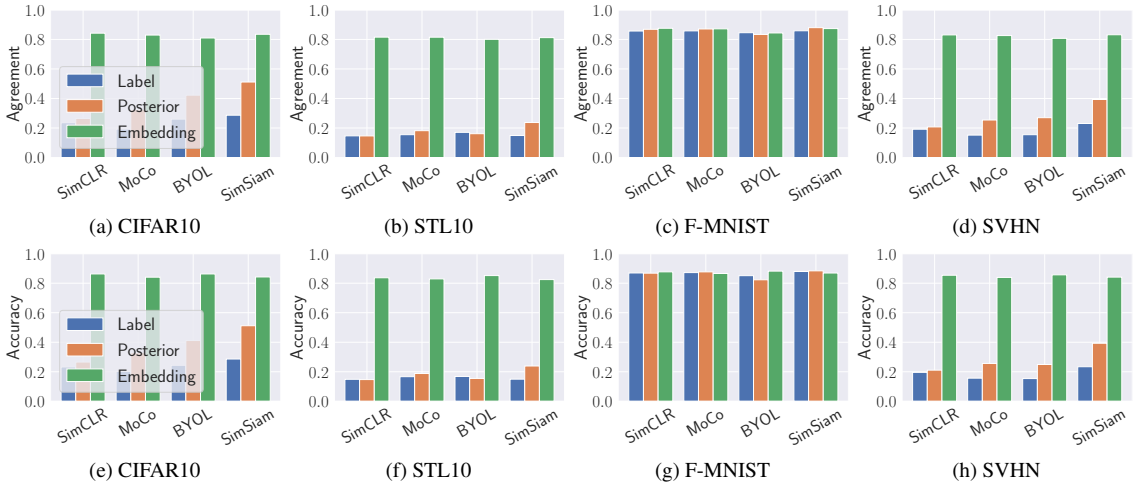
Figure 13. The performance of model stealing attack against target encodes and downstream classifiers trained on CIFAR10 and SVHN. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

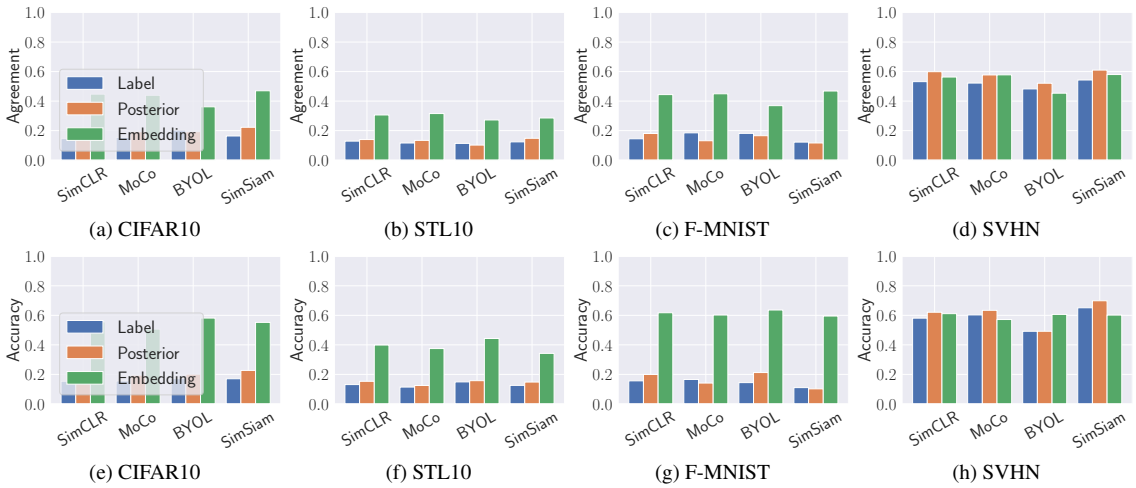
Figure 14. The performance of model stealing attack against target encodes and downstream classifiers trained on ImageNet and STL10. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

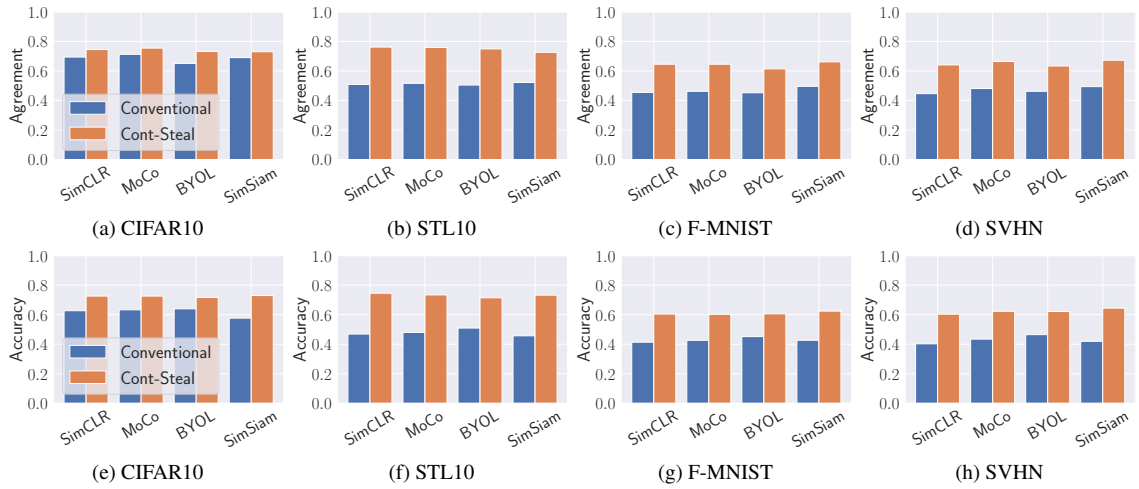
Figure 15. The performance of model stealing attack against target encodes and downstream classifiers trained on ImageNet and Fashion-MNIST. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

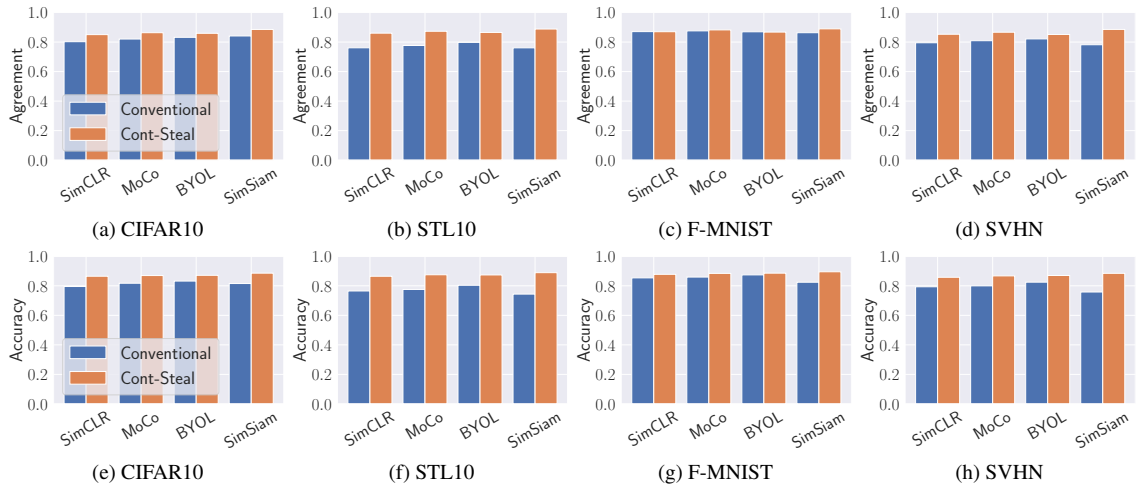
Figure 16. The performance of model stealing attack against target encodes and downstream classifiers trained on ImageNet and SVHN. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

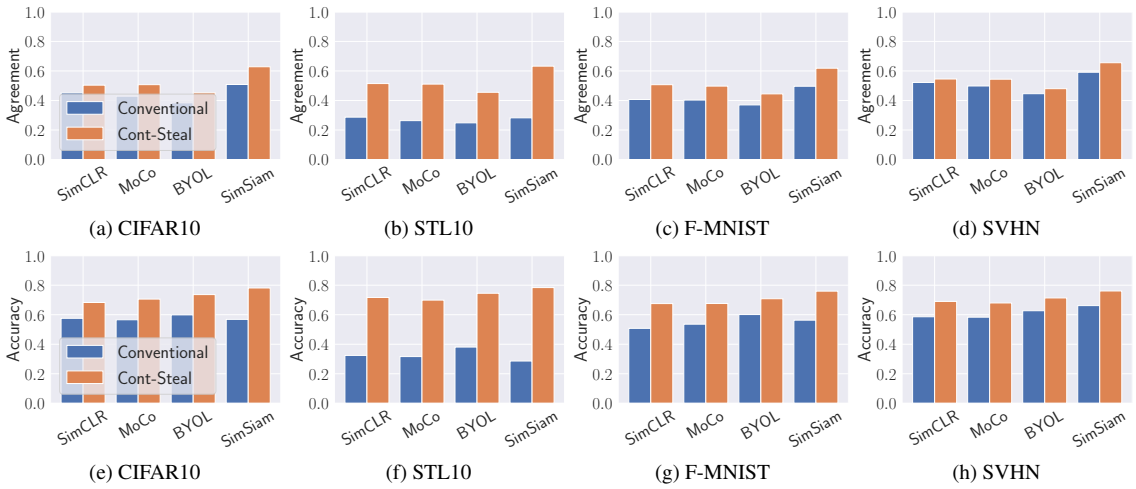
Figure 17. The performance of Cont-Steal and conventional attack against target encoders trained on CIFAR10. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses STL10 as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

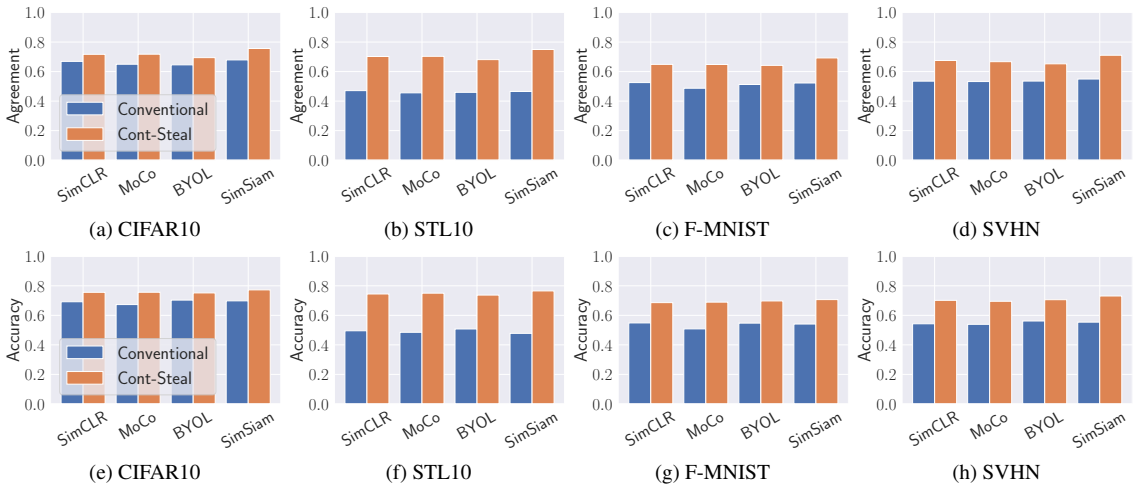
Figure 18. The performance of Cont-Steal and conventional attack against target encoders trained on CIFAR10. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses F-MNIST as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

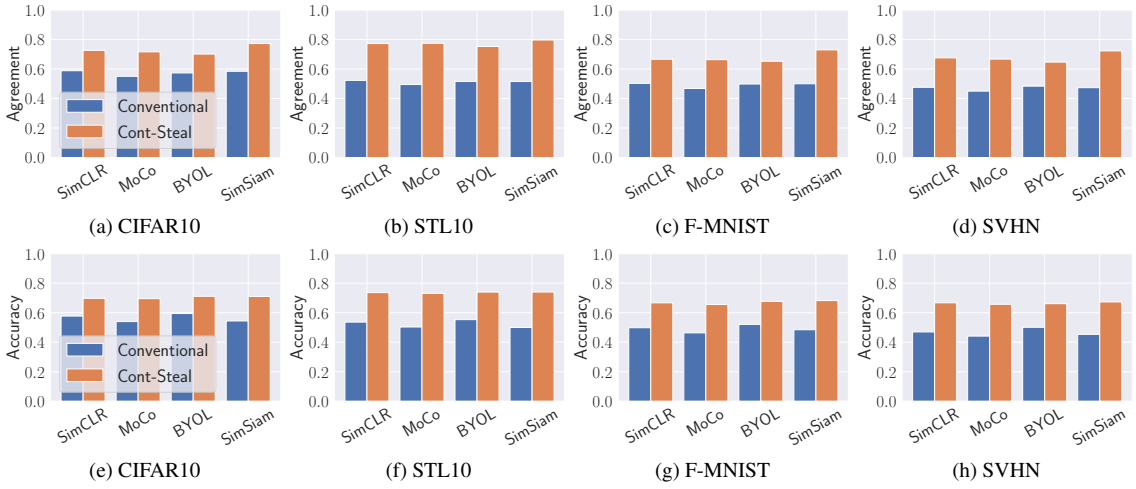
Figure 19. The performance of Cont-Steal and conventional attack against target encoders trained on CIFAR10. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses SVHN as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

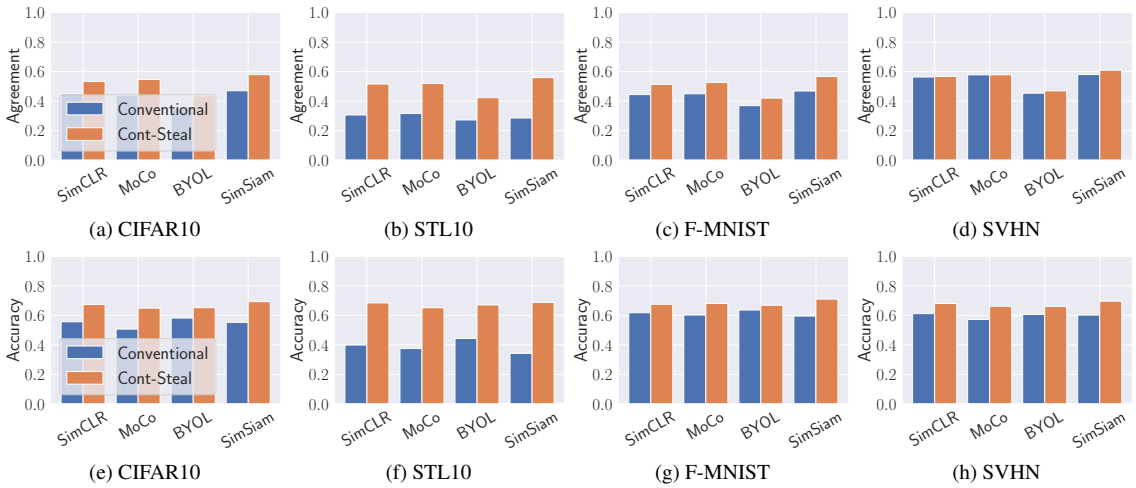
Figure 20. The performance of Cont-Steal and conventional attack against target encoders trained on ImageNet100. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses CIFAR10 as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

Figure 21. The performance of Cont-Steal and conventional attack against target encoders trained on ImageNet100. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses F-MNIST as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.




(a) CIFAR10 (b) STL10 (c) F-MNIST (d) SVHN

(e) CIFAR10 (f) STL10 (g) F-MNIST (h) SVHN

Figure 22. The performance of Cont-Steal and conventional attack against target encoders trained on ImagNet100. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses SVHN as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack.