Refine, Discriminate and Align: Stealing Encoders via Sample-Wise Prototypes and Multi-Relational Extraction

Shuchi Wu[1] Chuan Ma[2✉] Kang Wei[3✉] Xiaogang Xu[4] Ming Ding[5]
Yuwen Qian[1] Di Xiao[2] Tao Xiang[2]
[1] NJUST [2] CQU [3] PolyU [4] CUHK [5] Data61, CSIRO

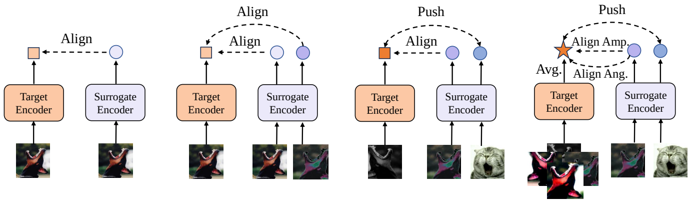
Abstract. This paper introduces **RDA**, a pioneering approach designed to address two primary deficiencies prevalent in previous endeavors aiming at stealing pre-trained encoders: (1) suboptimal performances attributed to biased optimization objectives, and (2) elevated query costs stemming from the end-to-end paradigm that necessitates querying the target encoder every epoch. Specifically, we initially **R**efine the representations of the target encoder for each training sample, thereby establishing a less biased optimization objective before the steal-training phase. This is accomplished via a sample-wise prototype, which consolidates the target encoder's representations for a given sample's various perspectives. Demanding exponentially fewer queries compared to the end-to-end approach, prototypes can be instantiated to guide subsequent query-free training. For more potent efficacy, we develop a multi-relational extraction loss that trains the surrogate encoder to **D**iscriminate mismatched embedding-prototype pairs while **A**ligning those matched ones in terms of both amplitude and angle. In this way, the trained surrogate encoder achieves state-of-the-art results across the board in various downstream datasets with limited queries. Moreover, RDA is shown to be robust to multiple widely-used defenses. Our code is available at `https://github.com/ShuchiWu/RDA`.

Keywords: Model Stealing · Self-Supervised Learning · Prototype

1 Introduction

Self-supervised learning (SSL) [4, 5, 9, 11, 12] is endowed with the capability to harness unlabeled data for pre-training a versatile encoder that applicable to a range of downstream tasks, or even showcasing groundbreaking zero-shot performances, e.g., CLIP [32]. However, SSL typically requires a large volume of data and computation resources to achieve a convincing performance, i.e., high-performance encoders are expensive to train [35]. To safeguard the confidentiality and economic worth of these encoders, entities like OpenAI offer their encoders

✉ Corresponding Author: `chuan.ma@cqu.edu.cn, adam-kang.wei@polyu.edu.hk`



(a) Conventional [34] (b) StolenEncoder [26] (c) Cont-Steal [34] (d) RDA (Ours)

Fig. 1: Illustrations of four stealing methods against SSL. The dotted arrows and text beside interpret how each method optimizes the surrogate encoder. Surrogate encoder branches in (b)-(d) involve data augmentations for training. Both (c) and (d) augment each sample before querying the target encoder but adopting different schemes.

as a premium service, exclusively unveiling the service API to the public. Users have the privilege of soliciting embeddings for their data, facilitating the training of diverse models tailored for specific downstream tasks. Regrettably, the substantial value of these encoders and their exposure to publicly accessible APIs render them susceptible to model stealing attacks [10, 26, 34].

Given a surrogate dataset, model stealing attacks aim to mimic the outputs of a target model to train either a high-accuracy copy of comparable performance or a high-fidelity copy that can serve as a stepping stone to perform further attacks like adversarial examples [31], membership inference attacks [33, 36], etc. The goal of this paper is to develop an approach that can train a surrogate encoder with competitive performance on downstream tasks by only accessing the target encoder's output embeddings. Meanwhile, it should require as little cost as possible, which is primarily derived from querying the target encoder.

We begin with systematically studying existing techniques for stealing pre-trained encoders. Specifically, the conventional method [34] and StolenEncoder [26] are similar, and both only need to query the target encoder once with each sample before the training. The output embedding can be viewed as a "ground truth" or an "optimization objective" to the sample, and the surrogate encoder is optimized to output a similar embedding when the same sample is fed, as illustrated in Figure 1 (a). The sole distinction lies in the optimization of StolenEncoder, which incorporates data augmentations (refer to Figure 1 (b)), as it supposes the target encoder will produce similar embeddings for an image and its augmentations. Nevertheless, we contend that this presumed similarity is of a modest degree, given the disparity between the surrogate and pre-training data. The target encoder's representations for such data's various augmentations may diverge and even be biased to other samples, as visually demonstrated in Figure 2 using a toy experiment. In this sense, only using the embedding of an image's single perspective (regardless of the original or augmented version) as the optimization objective is inadvisable.

On the other hand, Cont-Steal [34] augments each sample into two perspectives in each epoch: one is used to query the target encoder while the other is fed to the surrogate encoder. Two embeddings of the same sample from the tar-

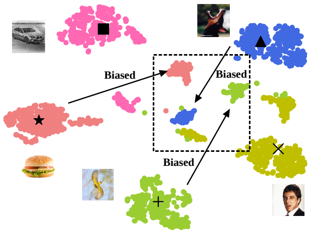


Fig. 2: t-SNE of embeddings belonging to five different images generated by an encoder pre-trained on CIFAR10, with each image augmented into 500 patches and fed into the encoder. **Each black marker represents the mean of the 500 embeddings of a certain image, i.e., its prototype.** Among the embeddings of an image's various augmentations, some can be **diverged** or even **biased**. In contrast, each image's prototype is **more distinguishable**, i.e., **less biased**.

get and surrogate encoders will be aligned, while those of different samples will be pushed apart, as illustrated in Figure 1 (c). In spite of such an end-to-end training scheme performing better, it suffers from high query costs since each sample is augmented and used to query in each epoch.

To tackle these issues, we propose ***RDA***. Specifically, we first **R**efine the target encoder's representations for each sample in the surrogate dataset by averagely aggregating its several (e.g., 10) different augmentations' embeddings, i.e., their mean value, to establish a sample-level prototype for it. Then, the sample-wise prototype is used to guide the surrogate encoder optimization, which can mitigate the impact of biased embeddings, as shown in Figure 2 (see black markers). We have an experiment presented in Figure 10 of our supplementary material to further quantitatively reveal the benefit of prototypes, i.e., significantly more similar with each augmentation patch's embedding, showcasing it is less biased. With a prototype, there is a static optimization objective for each sample across the entire training, and thus, it is done in a query-free manner. This enables RDA to have far less query cost than the end-to-end approach (less than 10%). To further enhance the attack efficacy with limited queries, we develop a *multi-relational extraction loss* that trains the surrogate encoder to **D**iscriminate mismatched embedding-prototype pairs while **A**ligning those matched ones in terms of both amplitude and angle. The framework and pipeline of RDA are depicted in Figures 1 (d) and 4, respectively.

Extensive experiments show that RDA acquires a favorable trade-off between the query cost (a.k.a, the money cost) and attack efficacy. As depicted in Figure 3, when the target encoder is pre-trained on CIFAR10, RDA averagely outperforms the state-of-the-art (SOTA), i.e., Cont-Steal [34], by 1.22% on the stealing efficacy over seven different downstream datasets, with a competitive smaller query cost, i.e., only 1% of Cont-Steal. Further, this performance gap can be widened to 5.20% with 10% query cost of Cont-Steal. In particular, RDA is demonstrated to be robust against multiple prevalent defenses [18, 30, 44].

In conclusion, our contributions are three folds:

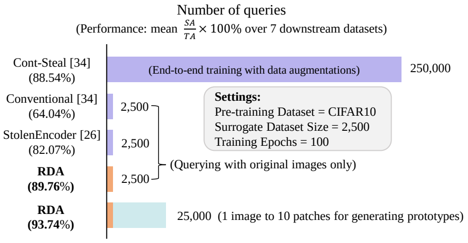


Fig. 3: Performance comparisons between four stealing methods against SSL. The presented results are the mean values achieved by each method over seven different downstream classification tasks, with their corresponding query costs. **Our proposed RDA can achieve SOTA results with the least query cost**.

- We comprehensively investigate two critical inadequacies of existing stealing methods against SSL, i.e., suboptimal efficacy and high query costs, and analyze the causes.
- We develop a novel approach to train surrogate encoders, namely RDA, with sample-wise prototype guidance and a multi-relational extraction loss.
- Extensive experiments are conducted to verify the effectiveness and robustness of RDA.

2 Related Work

Prototype Learning. A prototype refers to the mean of embeddings belonging to all images of an identical class, which serves as a proxy of the class [40]. With each class having a prototype, the model is trained to match its output with the corresponding prototype when inputting a sample of a certain class. Prototype learning has proven beneficial for various learning scenarios, e.g., federated learning [8, 16, 41] and few-shot learning [28, 38, 42]. In this paper, the optimization objective (i.e., the mean of embeddings belonging to an identical sample's multiple augmentations) we generate for each training sample is conceptually similar to prototypes, and thus we name it a sample-wise prototype.
Model Stealing Attacks against SSL. Dziedzic *et al.* [10] pointed out that the higher dimension of embeddings leaks more information than labels, making SSL more easily stolen. The primary objective of a stealing attack is to train a surrogate encoder to achieve high accuracy on downstream tasks or recreate a high-fidelity copy that can be used to mount further attacks such as adversarial examples [3] and membership inference attacks [25].
Defenses against Model Stealing Attacks. Existing defenses against model stealing attacks can be categorized based on when they are applied [10]. Perturbation-based defenses, e.g., adding noise [30], top-k [30] and truncating outputs [44], are applied before a stealing attack happens, aiming to limit the information leakage. On the other hand, watermarking defenses embed watermarks or unique

identifiers into the target model and use them to detect whether it is stolen after a stealing attack happens [17].

3 Methodology

3.1 Threat Model

Given a target encoder E_T, the attacker aims to train a surrogate encoder E_S at the lowest possible cost, which can perform competitively on downstream tasks with E_T. To achieve this goal, the attacker queries E_T for embeddings of an unlabeled surrogated dataset $D_S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ consisting of N images, and guide E_S to mimic the output of E_T for each sample in D_S. Specifically, we consider a black-box setting, where the attacker has direct access only to the outputs of E_T while remaining unaware of its architecture and training configurations, including pre-training datasets, loss functions, data augmentations schemes, etc.

3.2 Sample-Wise Prototypes

Recalling the discussion in Section 1, we expect to refine the target encoder's representations to find a less biased optimization objective for each sample in D_S. Enlightened by prototype learning [40] and Extreme-Multi-Patch-SSL (EMP-SSL) [43], which establish a prototype/benchmark for each class/sample, we propose a *sample-wise prototype* generation method for this purpose. In detail, the process of prototype generation can be divided into three steps as follows: ❶ Cropping and augmenting each image $\boldsymbol{x}_i \in D_S$ into n augmentation patches denoted as $\{\boldsymbol{x}'_{i,t,c}\}_{c=1}^{n} = \{\boldsymbol{x}'_{i,t,1}, \ldots, \boldsymbol{x}'_{i,t,n}\}$; ❷ Querying E_T with each augmentation patch in $\{\boldsymbol{x}'_{i,t,c}\}_{c=1}^{n}$, resulting in a set of embeddings denoted as $\{E_T(\boldsymbol{x}'_{i,t,c})\}_{c=1}^{n} = \{E_T(\boldsymbol{x}'_{i,t,1}), \ldots, E_T(\boldsymbol{x}'_{i,t,n})\}$; ❸ Calculating the mean of $\{E_T(\boldsymbol{x}'_{i,t,c})\}_{c=1}^{n}$ as the prototype $p_{\boldsymbol{x}_i}$ for $\boldsymbol{x}_i$ as follows:

$$p_{\boldsymbol{x}_i} = \tfrac{1}{n} \textstyle\sum_{c=1}^{n} E_T(\boldsymbol{x}'_{i,t,c}), \quad \boldsymbol{x}_i \in D_s. \tag{1}$$

Each generated prototype will be stored in a memory bank. These sample-wise prototypes can provide a stable optimization objective for each sample throughout the training process. Consequently, the attacker can accomplish the training in a query-free manner. Notably, setting n to a value that is far less than the training epochs, e.g., 10 vs. 100, is sufficient to yield a favorable performance. The prototype generation process is illustrated in step ① of Figure 4. Supplementary A.2 explains how we build the memory bank in practice.

3.3 Multi-Relational Extraction Loss

As step ② of Figure 4 shows, during training E_S, we also perform cropping and augmentation on each image $\boldsymbol{x}_i \in D_S$ to create m patches, forming an augmentation patch set denoted as $\{\boldsymbol{x}'_{i,s,q}\}_{q=1}^{m} = \{\boldsymbol{x}'_{i,s,1}, \ldots, \boldsymbol{x}'_{i,s,m}\}$. The m does not necessarily need to be set equal to n. Next, we feed each augmentation patch from $\{\boldsymbol{x}'_{i,s,q}\}_{q=1}^{m}$ to E_S, obtaining a set of embeddings denoted as

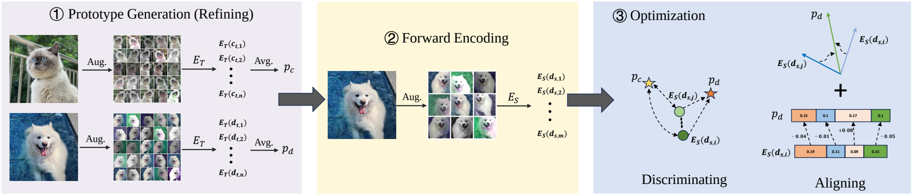


Fig. 4: Pipeline of RDA. **Prototype generation**: augment one sample into n patches and use them to query the target encoder (E_T). The mean of the n patches's embeddings is defined as a prototype for this sample. **Forward encoding**: crop one image into m patches and feed them to the surrogate encoder (E_S) for their embeddings. **Optimization:** align embeddings from the surrogate encoder to their matched prototypes in both angle and amplitude while pushing away those belonging to different samples.

$\{E_S(\boldsymbol{x}'_{i,s,q})\}_{q=1}^m = \{E_S(\boldsymbol{x}'_{i,s,1}), \dots, E_S(\boldsymbol{x}'_{i,s,m})\}$. To optimize E_S, we develop a *multi-relational extraction loss* composed of two parts, i.e., the *discriminating loss* and the *aligning loss*.

Discriminating Loss. The discriminating loss is responsible for training E_S to distinguish between different samples, as depicted in step ③ of Figure 4. For this goal, it pushes each embedding in $\{E_S(\boldsymbol{x}'_{i,s,q})\}_{q=1}^m$ away from mismatched prototypes $p_{\boldsymbol{x}_j}, i \neq j$ (denoted as negative pairs). In this regard, contrastive learning [5,6,12,19] offers a solution, as it can pull embeddings of positive pairs close while pushing embeddings of negative pairs apart. Furthermore, since more negative pairs can help improve the performance of contrastive learning [5], we refer to the loss designed by Sha *et al.* [34] to propose our discriminating loss $\mathcal{L}_D$. In particular, $\mathcal{L}_D$ considers both mismatched prototype-embedding pairs from different samples as well as embeddings from E_S for different samples as negative pairs. Formally, $\mathcal{L}_D$ can be expressed as follows:

$$\mathcal{L}_{pos}(\boldsymbol{x}_i) = \tfrac{1}{m} \textstyle\sum_{q=1}^m \exp\left(sim(E_S(\boldsymbol{x}'_{i,s,q}), p_{\boldsymbol{x}_i})/\tau\right), \tag{2}$$

$$\begin{aligned} \mathcal{L}_{neg}(\boldsymbol{x}_i) = \frac{1}{m} \sum_{q=1}^{m} \sum_{k=1}^{N} \mathbb{1}_{[i \neq k]} (&\exp\left(sim(E_S(\boldsymbol{x}'_{i,s,q}), p_{\boldsymbol{x}_k})/\tau\right) \\ &+ \exp\left(sim(E_S(\boldsymbol{x}'_{i,s,q}), E_S(\boldsymbol{x}'_{k,s,q}))/\tau\right)), \end{aligned} \tag{3}$$

$$\mathcal{L}_D = -\tfrac{1}{N} \textstyle\sum_{i=1}^N \log \tfrac{\mathcal{L}_{pos}(\boldsymbol{x}_i)}{\mathcal{L}_{neg}(\boldsymbol{x}_i)}, \tag{4}$$

where $sim(u, v)$ represents the cosine similarity between u and v, and τ is a temperature parameter.

Aligning Loss. While $\mathcal{L}_D$ can separate different samples effectively, it does not fully leverage the potential of prototypes. To further enhance the attack efficacy, we propose to align embeddings from E_S more thoroughly with their matched prototypes from the high-performance E_T, i.e., in terms of both amplitude and angle, as step ③ of Figure 4 shows. To measure the amplitude and angle deviations between two embeddings, we employ the mean square error (MSE) and cosine similarity to quantify them respectively, which can be formulated as follows:

$$\mathcal{L}'_{amp}(\boldsymbol{x}_i) = \tfrac{1}{m} \textstyle\sum_{q=1}^m \|E_S(\boldsymbol{x}'_{i,s,q}) - p_{\boldsymbol{x}_i}\|^2, \tag{5}$$

$$\mathcal{L}'_{ang}(\boldsymbol{x}_i) = \tfrac{1}{m} \textstyle\sum_{q=1}^{m} sim(E_S(\boldsymbol{x}'_{i,s,q}), p_{\boldsymbol{x}_i}). \tag{6}$$

Notice that we will normalize the value of $\mathcal{L}'_{ang}$ to $[0, 1]$. However, there is an issue concerning the equal penalty given by $\mathcal{L}'_{amp}$ and $\mathcal{L}'_{ang}$ to every deviation increase. For instance, when the MSE increases from 0.8 to 0.9 or from 0.3 to 0.4, both penalties given by $\mathcal{L}'_{amp}$ are 0.1, which is imprudent. On the contrary, we should have a more rigorous penalty for the increase in MSE from 0.3 to 0.4, as a smaller MSE is more desirable. Likewise, the same penalizing regime should be applied on $1/\mathcal{L}'_{ang}$ for the same reason. To this end, we adopt a logarithmic function to redefine the ultimate formulations of $\mathcal{L}_{amp}$ and $\mathcal{L}_{ang}$ as follows:

$$\mathcal{L}_{amp}(\boldsymbol{x}_i) = \log \mathcal{L}'_{amp}(\boldsymbol{x}_i), \tag{7}$$

$$\mathcal{L}_{ang}(\boldsymbol{x}_i) = \log(1/\mathcal{L}'_{ang}(\boldsymbol{x}_i)) = -\log \mathcal{L}'_{ang}(\boldsymbol{x}_i). \tag{8}$$

We posit the amplitude and angle deviations hold equal importance, and formulate the aligning loss as follows:

$$\mathcal{L}_A = \frac{1}{N} \sum_{i=1}^{N} (\mathcal{L}_{amp}(\boldsymbol{x}_i) + \mathcal{L}_{ang}(\boldsymbol{x}_i)) = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\mathcal{L}'_{ang}(\boldsymbol{x}_i)}{\mathcal{L}'_{amp}(\boldsymbol{x}_i)}. \tag{9}$$

Our experiments in Supplementary A.4 demonstrate the combination of $\mathcal{L}_{amp}$ and $\mathcal{L}_{ang}$ offers superior results than using each solely.

Lastly, the loss function for stealing pre-trained encoders is as follows:

$$\mathcal{L} = \lambda_1 \cdot \mathcal{L}_D + \lambda_2 \cdot \mathcal{L}_A, \tag{10}$$

where λ_1 and λ_2 are preset coefficients to adjust the weight of each part in $\mathcal{L}$. To show the superiority of our design, especially on the loss function, we have conducted ablation studies in Section 4.4 and explored several alternative designs in Supplementary A.3.

Detailed steps of RDA are summarized in Algorithm 1 of Supplementary A.2.

4 Experiments

4.1 Experimental Setup

Target Encoder Settings. We use SimCLR [5] to pre-train two ResNet18 [13] encoders on CIFAR10 [21] and STL10 [7], respectively, as two medium-scale target encoders. Furthermore, we consider two real-world large-scale ResNet50 encoders as the targets, i.e., the ImageNet encoder pre-trained by Google [5], and the CLIP encoder pre-trained by OpenAI [32]. Besides the ResNet family, RDA also has been demonstrated effective upon other backbones, i.e., VGG19_bn [37], DenseNet121 [15], and MobileNetV2 [14], in Supplementary A.4.

Attack Settings. Regarding the surrogate dataset, it is derived from Tiny ImageNet [23]. Specifically, we randomly sample 2,500 images and resize them to

Table 1: Results of RDA against two medium-scale encoders.

Pre-training Dataset	Downstream Dataset	TA	SA	$\frac{\text{SA}}{\text{TA}} \times 100\%$
CIFAR10	MNIST	97.27	96.62	99.33
	F-MNIST	88.58	89.32	100.84
	GTSRB	61.76	62.75	101.60
	SVHN	73.78	73.74	99.95
STL10	MNIST	96.84	96.31	99.45
	F-MNIST	90.08	87.91	97.59
	GTSRB	64.45	57.70	89.53
	SVHN	61.85	70.67	114.26

Table 2: Results of RDA against the ImageNet Encoder and CLIP.

Target Encoder	Downstream Dataset	TA	SA	$\frac{\text{SA}}{\text{TA}} \times 100\%$
ImageNet Encoder	MNIST	97.38	95.89	98.47
	F-MNIST	91.67	90.83	99.08
	GTSRB	63.14	59.62	94.43
	SVHN	73.20	69.21	94.55
CLIP	MNIST	97.90	93.96	95.98
	F-MNIST	89.92	87.43	97.23
	GTSRB	68.61	55.84	81.39
	SVHN	69.53	57.58	82.81

Table 3: Computation resources required by pre-training the two targeted real-world encoders from scratch and RDA to steal them.

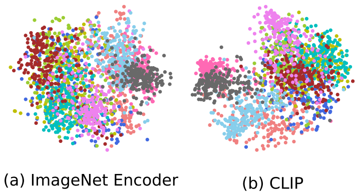
Target	Pre-training		RDA	
Encoder	Hardware	Time (hrs)	Hardware	Time (hrs)
ImageNet Encoder	TPU v3	192	RTX A5000	10.8
CLIP	V100 GPU	255,744		16.2

32×32 for stealing encoders pre-trained on CIFAR10 and STL10. When stealing the ImageNet encoder and CLIP, the number is 40,000 and 60,000, respectively, with each image resized to 224×224. For the surrogate encoder architecture, we adopt a ResNet18 across our experiments. During the attack, we set n in Eq. 1 as 10, m in Eq. 2, 3, 5, and 6 as 5, τ in Eq. 2 and 3 as 0.07, and both λ_1 and λ_2 in Eq. 10 as 1 unless stated otherwise. For training, the batch size is set as 100, and we employ an Adam optimizer [20] with a learning rate of 0.001.

Evaluation Settings. We train each surrogate encoder for 100 epochs and test its KNN accuracy [45] on CIFAR10 after each epoch. The best-trained surrogate encoders, as well as target encoders, will be used to train downstream classifiers for the linear probing evaluation. Each downstream classifier will be trained for 100 epochs with an Adam optimizer and a learning rate of 0.0001. Specifically, we totally consider seven downstream datasets, namely MNIST [24], CIFAR10 [21], STL10 [7], GTSRB [39], CIFAR100 [22], SVHN [29], and F-MNIST [46]. Additional small-scale experiments on more complex datasets (e.g., Food 101 [2]) are included in our supplementary material. As done in [26], we use Target Accuracy (TA) to evaluate target encoders, Steal Accuracy (SA) to evaluate surrogate encoders, and $\frac{\text{SA}}{\text{TA}} \times 100\%$ to evaluate the efficacy of the stealing attack.

4.2 Effectiveness of RDA

Stealing Medium-Scale Encoders. Table 1 shows the results achieved by RDA upon two medium-scale target encoders pre-trained on CIFAR10 and STL10, respectively. As the results show, using a surrogate dataset of 5% the size of CIFAR10 (2,500 / 50,000) and 2.4% the size of STL10 (2,500 / 105,000), RDA can steal nearly 100% of the functionality of both target encoders, except for the GTSRB scenario when stealing the encoder pre-trained on STL10. Nonetheless, even in this exceptional case, the obtained $\frac{\text{SA}}{\text{TA}} \times 100\%$ is approximately 90%,



(a) ImageNet Encoder (b) CLIP

Fig. 5: t-SNE of embeddings of 2,000 images sampled from CIFAR10 generated by the ImageNet encoder and CLIP.

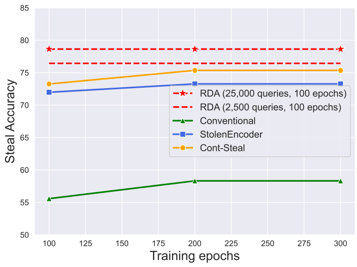


Fig. 6: The training epochs of RDA are fixed to **100** while those of baselines are prolonged to **300**.

which exemplifies the remarkable effectiveness of RDA. Furthermore, the surrogate encoder trained through RDA even outperforms the target encoder on multiple datasets. Our interpretation of this is that the target encoder fits the pre-training dataset more strongly compared to the RDA-trained surrogate encoder, thus exhibiting inferior performance on some out-of-distribution datasets.

Stealing Real-World Large-Scale Encoders. We further evaluate RDA on two real-world pre-trained encoders, i.e., the ImageNet encoder and CLIP. We aim to demonstrate the effectiveness of RDA upon such large-scale encoders trained with a massive amount of data. Specifically, the ImageNet encoder is pre-trained on the ImageNet dataset, which has 1.3 million images. The CLIP encoder is pre-trained on a web-scale dataset that consists of 400 million image-text pairs. During the attack, we set the patch number for training as 1 (i.e., m in Eq. 2, 3, 5, and 6) to save time. Other parameters follow the default setting except the λ_1 and λ_2 when stealing CLIP, in which we set λ_1 and λ_2 to 1 and 20, respectively. To interpret the reason for this adjustment, we randomly sample 200 images from each class in the testing set of CIFAR10 and visualize the output embeddings from the ImageNet encoder and CLIP using t-SNE [27]. Figure 5 shows that embeddings from the ImageNet encoder are more uniformly distributed compared to those from CLIP. This is because CLIP is a multimodal encoder, which is pre-trained by conducting contrastive learning between image-text pairs. Therefore, the text also shares part of the embedding space and is mutually exclusive with mismatched images. Although $\mathcal{L}_D$ can push embeddings of different samples apart, a significant weight of it will make all embeddings more uniformly distributed, i.e., not align with CLIP. Therefore, we set a smaller weight to $\mathcal{L}_D$ when stealing CLIP. This adjustment is practical since the attacker has access to the outputs of the target encoder, and thus can observe its embedding space to adjust its attack settings. Table 2 shows that using a surrogate dataset of only 3.07% the size of ImageNet (40K / 1.3M) and 0.015% the size of the training data of CLIP (60K / 400M), RDA can achieve comparable performances with the two encoders across various downstream tasks. The results also reveal that even under the architecture of the surrogate and target encoders being distinct (ResNet18 vs. ResNet50), RDA still exhibits high effec-

Table 4: Comparisons between RDA and baselines to steal the encoder pre-trained on CIFAR10 under the same surrogate dataset size. We report the $\frac{\mathbf{SA}}{\mathbf{TA}} \times 100\%$ achieved by each method and its query cost. The Optimal and suboptimal results are highlighted.

Method	CIFAR10	CIFAR100	MNIST	GTSRB	SVHN	STL10	F-MNIST	Queries
Conventional [34]	63.36	45.40	96.16	20.34	68.26	60.25	94.52	2,500
StolenEncoder [26]	79.03	68.78	98.76	63.70	92.01	76.29	95.91	2,500
Cont-Steal [34]	80.43	73.94	98.03	95.55	96.39	80.08	95.39	250,000
RDA	83.84	77.49	98.97	94.27	96.79	82.25	94.69	2,500
	85.85	83.08	99.33	101.60	99.95	85.56	100.84	25,000

Table 5: Comparisons between RDA and StolenEncoder under the same query cost.

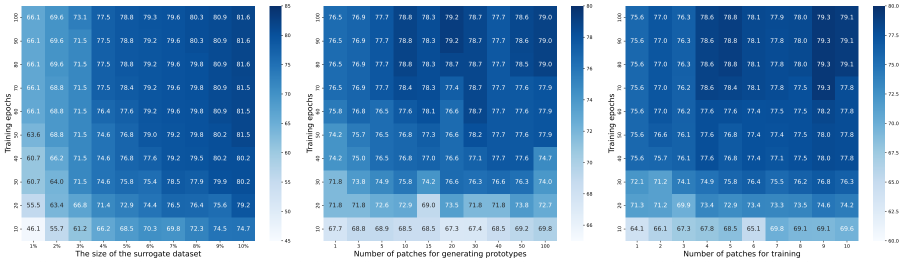
Queries	Method	setting	SA
2500	StolenEnoder	2,500 images	71.98
	RDA	500 images×5 patches	65.38
		625 images×4 patches	67.07
		1,250 images×2 patches	71.44
		2,500 images× 1 patch	76.36
25,000	StolenEnoder	25,000 images	76.23
	RDA	2,500 images×10 patches	78.50

tiveness. Moreover, Table 3 shows that RDA requires much fewer computation resources to steal the two encoders than pre-training them from scratch, owing to the small surrogate dataset and lightweight network architecture.

4.3 Comparision with Existing Methods

We consider the *conventional method* (CVPR 2023, proposed as the baseline by [34]), *StolenEncoder* (CCS 2022) [26], and *Cont-Steal* (CVPR 2023) [34] as our baselines. A more detailed explanation of them is in Supplementary A.1. Specifically, we compare RDA with them to steal the encoder pre-trained on CIFAR10 under three different scenarios. For a fair comparison, the surrogate dataset remains identical across all baselines (except for some results presented in Table 5 due to different sizes of surrogate datasets).

Under the Same Surrogate Dataset Size. In this subsection, we fix the surrogate dataset size to 2,500 and make it identical across all methods. As Table 4 shows, RDA outperforms baselines by a significant margin across all seven downstream datasets with a moderate query cost. We also evaluate RDA under a setting that has the same query cost as the conventional method and StolenEncoder (i.e., 2,500 queries), for which we query with each sample once in its origin version. The results show that RDA still surpasses baselines over five out of seven datasets with the least query cost. In addition, the comparison between the results achieved by RDA with 25,000 and 2,500 queries reveals that more patches for generating prototypes will enhance the attack. The embedding space of the surrogate encoder trained by each method is visualized in Figure 11 of Supplementary A.4, which shows that our RDA can train surrogate encoders to discriminate different samples better. Besides, We have summarized the mean



(a) Ablation studies on the size of the surrogate dataset

(b) Ablation studies on the patch number for generating prototypes

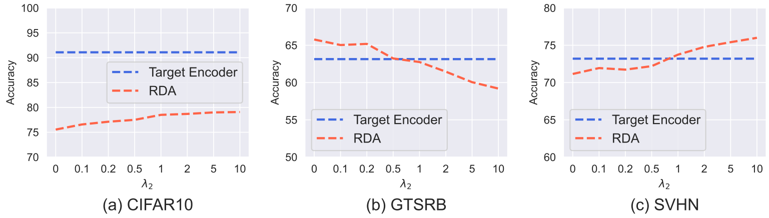
(c) Ablation studies on the training patch number

Fig. 7: Heat map of the achieved SAs (better zoom in). This figure shows the performance of 100 combinations of different training epochs and (a) the surrogate dataset size, (b) the patch number for generating prototypes, and (c) the patch number for training.

value achieved by each method over the seven downstream datasets in Figure 3 to show the superiority of RDA.

Under the Same Query Cost. To demonstrate the superiority of RDA against baselines under an identical query cost, we consider two fixed values, i.e., 2,500 and 25,000. We take StolenEncoder as the rival since it is the best baseline with the least query cost. We note that Cont-Steal is beyond our consideration because the data volume for it would be too small under such a setting, which is only 25 images for 2,500 queries and 250 images for 25,000 queries. N images × n patches in the table indicates that we use N images as the surrogate dataset and crop each image into n patches to generate prototypes. Therefore, the query cost of RDA is $N \times n$. As shown in Table 5, under the same query cost of 2,500, RDA can achieve comparable results with StolenEncoder with half the size of its surrogate dataset. With the same 2,500 images, RDA outperforms StolenEncoder by a significant margin, showcasing its superiority. Moreover, under a query cost of 25,000, RDA outperforms StolenEncoder by 2.14% with only 10% the size of its surrogate dataset. Besides, comparisons between different configurations of RDA under queries of 2,500 indicate that the training data volume has a dominant impact on the performance.

Under the Same Time Cost. Since each sample is also cropped and augmented into multiple patches for training (i.e., needing to forward encode each sample multiple times), RDA takes the longest time for training. Table 10 in Supplementary A.4 contains the detailed time cost of each method. To make a fair comparison, we prolong the training for another 200 epochs for baseline methods, i.e., 300 epochs in total, which is sufficient to lead their performances to saturate. The trained surrogate encoders are then evaluated on CIFAR10 as the downstream dataset. Figure 6 reveals that even though baseline methods take longer time to train, RDA still outperforms them with the least query cost, and the performance gap can be further widened by a moderate query cost.



(a) CIFAR10 (b) GTSRB (c) SVHN

Fig. 8: Ablation studies on the weight of each part in $\mathcal{L}$, with $\lambda_1 = 1$ and λ_2 varying.

Table 6: Ablation studies on loss functions. The presented results are SAs.

Loss Type	CIFAR10	STL10	CIFAR100	MNIST	F-MNIST	GTSRB	SVHN	AVG
MSE	65.66	53.83	31.00	94.63	86.12	32.79	62.97	61.00
Cosine Similarity	80.23	66.56	43.81	97.00	88.76	55.53	75.94	72.54
KL Divergence	77.35	65.30	40.98	96.23	87.82	57.04	69.38	70.58
InfoNCE	73.99	61.68	39.24	95.76	87.86	58.82	68.88	69.43
$\mathcal{L}_D$	75.54	63.2	43.16	96.52	89.27	65.77	71.15	72.23
$\mathcal{L}_A$	79.70	65.59	43.79	96.89	89.24	55.96	76.70	72.55
$\mathcal{L}_A + \mathcal{L}_D$	79.39	66.76	44.27	96.74	89.32	62.75	73.74	73.28

4.4 Ablation Studies

In this subsection, we conduct ablation studies to investigate the impact of the surrogate dataset size, the patch number for generating prototypes (i.e., n in Eq. 1), the patch number for training (i.e., m in Eq. 2, 3, 5, and 6), and loss functions. All the experiments are conducted to steal the encoder pre-trained on CIFAR10. For evaluations, the trained surrogate encoders are assessed on CIFAR10 in the first three ablation studies. While examining the influence of loss functions, we consider seven distinct downstream datasets, as shown in Table 4, to derive a more comprehensive conclusion. Other configurations remain at their default settings, and we present the resulting SAs.

Surrogate Dataset Size. Recall our default setting where we randomly sample 2,500 images from Tiny ImageNet to construct the surrogate dataset, which is 5% the size of CIFAR-10. We vary the ratio from 1% to 10% to investigate its impact. Figure 7 (a) shows that a larger surrogate dataset will accelerate the training and improve the attack performance.

Patch Number for Generating Prototypes. We vary the patch number for generating prototypes from 1 to 100 to investigate its impact. Figure 7 (b) shows that there is a tendency for more patches for generating prototypes will improve the attack performance and accelerate the convergence. However, when the patch number reaches 10, more patches do not further improve the attack performance due to the limited surrogate dataset size.

Patch Number for Training. We vary the patch number for training from 1 to 10 to investigate its impact. Figure 7 (c) reveals a tendency for more patches for training will accelerate the training and achieve better results. However, more patches also indicate longer training time and more computation resources.

Loss Functions. To demonstrate the superiority of the design of our loss function and the necessity of each part, we conduct ablation studies as shown in Table 6. Table 6 shows that $\mathcal{L}_D$ largely surpass other loss functions on GTSRB while $\mathcal{L}_A$ largely surpass other loss functions on SVHN. We hypothesize that $\mathcal{L}_A$ attains optimal outcomes on SVHN due to the pronounced resemblance between SVHN and the surrogate dataset. It appears that $\mathcal{L}_A$ inclines towards inducing the surrogate encoder to overfit the surrogate data, thereby elucidating its suboptimal performance on GTSRB, a dataset characterized by less similarity with the surrogate dataset. On the contrary, $\mathcal{L}_D$ makes the surrogate encoder fit less to the surrogate dataset and has the best result on GTSRB. To further investigate the effect of each part in our loss, we have conducted an ablation experiment about the weight of each part in our loss. As Figure 8 shows, the surrogate encoder will perform better on SVHN while worse on GTSRB when the weight of $\mathcal{L}_A$ increases. We refer to Supplementary A.4 for a more detailed analysis. Combining $\mathcal{L}_D$ and $\mathcal{L}_A$, our loss is the most robust one, which achieves the best average result over the seven tested datasets. For example, although cosine similarity and $\mathcal{L}_A$ performs well on SVHN (about 76%), their performance on GTSRB is terrible (about 56%). Our loss design can improve their performance on GTSRB by about 7% in the cost of less than 3% decrease on SVHN.

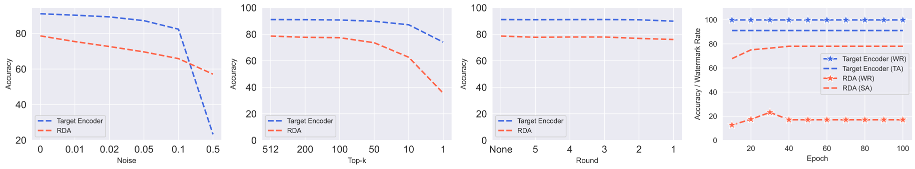
4.5 Robustness to Defenses

In this section, we evaluate the robustness of RDA against three perturbation-based defenses that aim to limit information leakage and one watermarking defense that aims to detect whether a suspected encoder is stolen. The pre-training and downstream datasets both are set to CIFAR10.

Perturbation-Based Defense. We evaluate three common practices of this type of defense, i.e., adding noise [30], top-k [30] and rounding [44].

- **Adding noise:** Adding noise means that the defender will introduce noise to the original outputs of the target encoder. Following [10], we set the mean of the noise to 0 and vary the standard deviations to control the noise level.
- **Top-k:** Top-k means that the defender will only output the first k largest number of each embedding from the target encoder and set the rest as 0. We vary the value of k to simulate different perturbation levels.
- **Rounding:** Rounding truncates each value in the embedding to a specific precision, which we vary to simulate different perturbation levels.

The results of the three perturbation-based defense methods are summarized in (a), (b), and (c) of Figure 9, respectively. We can observe in Figure 9 (a) that while adding noise can mitigate the model stealing attack, it decreases the utility of the target encoder more significantly. Moreover, Figure 9 (b) shows that though top-k can decrease the efficacy of RDA as the defender lowers the



(a) Adding noise (b) Top-k (c) Rounding (d) Watermarking

Fig. 9: The performance of different defense methods on CIFAR10.

value of k, it also severely deteriorates the target encoder's performance. On the other hand, Figure 9 (c) reveals that rounding has a minimal effect on the performance of both the target encoder and attack. Our experimental results demonstrate the robustness of RDA against perturbation-based defenses.

Watermarking Defense. As shown by Adi *et al.* [1], backdoors can be used as watermarks to claim the ownership of a model. In this sense, we follow [34] and leverage BadEncoder [18] to embed a backdoor into the target encoder as the watermark. Ideally, if the watermarked encoder is stolen, the surrogate encoder trained via stealing should also be triggered by the defender-specified trigger to exhibit certain behaviors, and thus the ownership can be claimed. Fortunately, Figure 9 (d) shows that RDA can steal the functionality of the target encoder while unlearning the watermark. More specifically, the watermark rate (WR) of the target encoder is 99.87% but only 17.04% for the surrogate encoder when the training converges. This observation indicates that the watermark embedded in the target encoder cannot be preserved by RDA.

5 Conclusion

In this paper, we have proposed a novel model stealing method against SSL named RDA, which stands for *refine*, *discriminate*, and *align*. Compared with previous methods [26, 34], RDA establishes a less biased optimization objective for each training sample and strives to extract more abundant functionality of the target encoder. This empowers RDA to exhibit tremendous effectiveness, efficiency, and robustness across diverse datasets and settings.

Ethical Concerns and Possible Defenses. We underscore that the misuse of model stealing techniques can jeopardize the privacy and economic rights of embedding service providers. We hope that our work will inspire the development of more sophisticated defense mechanisms to thwart model stealing attacks. A possible defense involves rejection schemes for suspicious queries that appear highly similar, which we have more detailed discussions about in the Supplementary.

Limitation and Future Work. The query cost of RDA is not yet optimal and can be further reduced by techniques like clustering to establish an identical prototype for multiple similar images. Moreover, more unique (e.g., more efficient) designs of RDA for transformer-based huger models remain largely unexplored.

Acknowledgements. This work is supported by the National Key R&D Program of China under Grant 2022YFB3103500, the National Natural Science Foundation of China under Grant U20A20176 and 62072062, the National Natural Science Foundation of Chongqing under Grant cstc2022ycjh-bgzxm0031, and the Chongqing Research Program of Basic Research and Frontier Technology (Chongqing Talent) under Grant cstc2024ycjh-bgzxm0048.

References

1. Adi, Y., Baum, C., Cisse, M., Pinkas, B., Keshet, J.: Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 1615–1631 (2018)
2. Bossard, L., Guillaumin, M., Van Gool, L.: Food-101–mining discriminative components with random forests. In: Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part VI 13. pp. 446–461. Springer (2014)
3. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp). pp. 39–57. Ieee (2017)
4. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9650–9660 (2021)
5. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020)
6. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020)
7. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 215–223. JMLR Workshop and Conference Proceedings (2011)
8. Dai, Y., Chen, Z., Li, J., Heinecke, S., Sun, L., Xu, R.: Tackling data heterogeneity in federated learning with class prototypes. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 7314–7322 (2023)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
10. Dziedzic, A., Dhawan, N., Kaleem, M.A., Guan, J., Papernot, N.: On the difficulty of defending self-supervised learning against model extraction. In: International Conference on Machine Learning. pp. 5757–5776. PMLR (2022)
11. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al.: Bootstrap your own latent-a new approach to self-supervised learning. Advances in neural information processing systems **33**, 21271–21284 (2020)
12. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9729–9738 (2020)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

14. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
15. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
16. Huang, W., Ye, M., Shi, Z., Li, H., Du, B.: Rethinking federated learning with domain shift: A prototype view. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 16312–16322. IEEE (2023)
17. Jia, H., Choquette-Choo, C.A., Chandrasekaran, V., Papernot, N.: Entangled watermarks as a defense against model extraction. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 1937–1954 (2021)
18. Jia, J., Liu, Y., Gong, N.Z.: Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 2043–2059. IEEE (2022)
19. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. Advances in neural information processing systems **33**, 18661–18673 (2020)
20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
21. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
22. Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 (canadian institute for advanced research). URL http://www. cs. toronto. edu/kriz/cifar. html **5**(4), 1 (2010)
23. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. CS 231N **7**(7), 3 (2015)
24. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010), `http://yann.lecun.com/exdb/mnist/`
25. Liu, H., Jia, J., Qu, W., Gong, N.Z.: Encodermi: Membership inference against pre-trained encoders in contrastive learning. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 2081–2095 (2021)
26. Liu, Y., Jia, J., Liu, H., Gong, N.Z.: Stolenencoder: stealing pre-trained encoders in self-supervised learning. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 2115–2128 (2022)
27. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)
28. Mettes, P., Van der Pol, E., Snoek, C.: Hyperspherical prototype networks. Advances in neural information processing systems **32** (2019)
29. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
30. Orekondy, T., Schiele, B., Fritz, M.: Knockoff nets: Stealing functionality of black-box models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4954–4963 (2019)
31. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security. pp. 506–519 (2017)
32. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)

33. Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., Backes, M.: Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. arXiv preprint arXiv:1806.01246 (2018)
34. Sha, Z., He, X., Yu, N., Backes, M., Zhang, Y.: Can't steal? cont-steal! contrastive stealing attacks against image encoders. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16373–16383 (2023)
35. Sharir, O., Peleg, B., Shoham, Y.: The cost of training nlp models: A concise overview. arXiv preprint arXiv:2004.08900 (2020)
36. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE symposium on security and privacy (SP). pp. 3–18. IEEE (2017)
37. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
38. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. Advances in neural information processing systems **30** (2017)
39. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. Neural networks **32**, 323–332 (2012)
40. Tan, Y., Long, G., Liu, L., Zhou, T., Lu, Q., Jiang, J., Zhang, C.: Fedproto: Federated prototype learning across heterogeneous clients. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 8432–8440 (2022)
41. Tan, Y., Long, G., Ma, J., Liu, L., Zhou, T., Jiang, J.: Federated learning from pretrained models: A contrastive learning approach. Advances in Neural Information Processing Systems **35**, 19332–19344 (2022)
42. Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J.B., Isola, P.: Rethinking few-shot image classification: a good embedding is all you need? In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16. pp. 266–282. Springer (2020)
43. Tong, S., Chen, Y., Ma, Y., Lecun, Y.: Emp-ssl: Towards self-supervised learning in one training epoch. arXiv preprint arXiv:2304.03977 (2023)
44. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction {APIs}. In: 25th USENIX security symposium (USENIX Security 16). pp. 601–618 (2016)
45. Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via nonparametric instance discrimination. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3733–3742 (2018)
46. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)