SwiftThief: Enhancing Query Efficiency of Model Stealing by Contrastive Learning

Jeonghyun Lee , **Sungmin Han** , **Sangkyun Lee***
School of Cybersecurity, Korea University
{nomar0107, sungmin_15, sangkyun}@korea.ac.kr

Abstract

Model-stealing attacks are emerging as a severe threat to AI-based services because an adversary can create models that duplicate the functionality of the black-box AI models inside the services with regular query-based access. To avoid detection or query costs, the model-stealing adversary must consider minimizing the number of queries to obtain an accurate clone model. To achieve this goal, we propose SwiftThief, a novel model-stealing framework that utilizes both queried and unqueried data to reduce query complexity. In particular, SwiftThief uses contrastive learning, a recent technique for representation learning. We formulate a new objective function for model stealing consisting of self-supervised (for abundant unqueried inputs from public datasets) and soft-supervised (for queried inputs) contrastive losses, jointly optimized with an output matching loss (for queried inputs). In addition, we suggest a new sampling strategy to prioritize rarely queried classes to improve attack performance. Our experiments proved that SwiftThief could significantly enhance the efficiency of model-stealing attacks compared to the existing methods, achieving similar attack performance using only half of the query budgets of the competing approaches. Also, SwiftThief showed high competence even when a defense was activated for the victims.

1 Introduction

An increasing number of AI models are commercialized and provided as cloud-based services [Ribeiro *et al.*, 2015], with the remarkable success of deep neural networks. AI models in such services are valuable intellectual properties for developers since training and model optimization typically require significant research and implementation costs. Such AI models may seem well-protected since they are black boxes; their details are inaccessible from the outside world. However, recent studies have revealed that an adversary can create a clone model that mimics the functionality of the black-box AI model (the victim) by collecting the victim model's responses to adversarial queries.

Such model stealing attacks have been improved in several directions since the early works [Tramèr *et al.*, 2016; Papernot *et al.*, 2016], to use public datasets instead of assuming that the adversary can access the victim's training data [Orekondy *et al.*, 2019], to use active sampling to reduce the number of required queries [Pal *et al.*, 2020] and to use adversarial examples [Yu *et al.*, 2020] or generative models [Gong *et al.*, 2021; Kariyappa *et al.*, 2021; Truong *et al.*, 2021] to synthesize attack queries. Despite the similarity to the knowledge distillation [Hinton *et al.*, 2014], model stealing is more challenging since the adversary cannot access the victim's training data in general. Furthermore, the number of queries must be minimized to avoid detection. Still, existing attack methods require large numbers of queries for successful model extraction.

This paper presents SwiftThief, a novel attack framework that significantly improves query efficiency by taking into account two overlooked subjects in model-stealing attacks:

Utilization of Unqueried Data. A model-stealing attacker often has many unqueried data points relative to the number of queried ones when the attack is in its early stages. Unfortunately, most existing attack methods do not effectively use unqueried inputs during an attack, except for a few recent works using semi-supervised learning [Jagielski *et al.*, 2020; Xiao *et al.*, 2022] or self-supervised learning [Zhao *et al.*, 2023]. To address the issue, SwiftThief uses both supervised and self-supervised contrastive learning to learn useful feature representations that are highly transferable [Islam *et al.*, 2021; Liu *et al.*, 2022a], hopefully to victims' tasks, from queried and unqueried attack vectors.

Class Imbalance in Attack Queries. Although class imbalance of training data is often a critical issue in learning problems, not enough attention has been paid to the subject in the context of model stealing. In particular, as shown in Figure 1, we found that even ActiveThief [Pal *et al.*, 2020], one of the recent attacks adopting active learning techniques, faces severe class imbalance in queried inputs, which deteriorates clone models' performance on rarely queried classes. To alleviate the problem, SwiftThief introduces prioritization of rarely queried classes in sampling, selecting inputs preferentially so that query outputs will have a high probability on rare

*Corresponding author

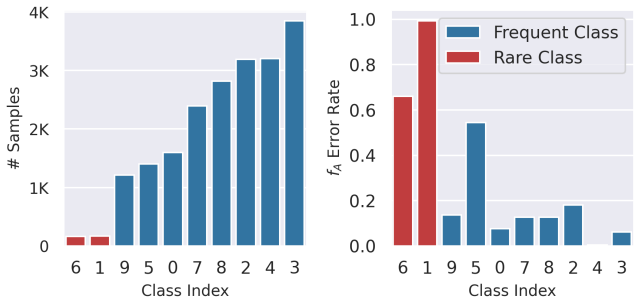

Figure 1: Class imbalance in ActiveThief (queries are from ImageNet, victim task is MNIST). Class-wise sample counts and error rates of clone models are shown.

classes based on the density estimation of classes in the clone model's feature space.

Our paper has the following contributions:

- We propose SwiftThief, a novel model-stealing approach utilizing both unqueried and queried data via self-supervised and newly designed soft-supervised contrastive loss, which can fully utilize the probability information from queried data.
- To address the issue of class imbalance in queried data, we introduce a weighted regularizer acting more strongly on rare classes and an enhanced query sampling strategy prioritizing candidates similar to queried rare-class samples in the clone model's representation space.
- SwiftThief achieved the maximum performance of the existing attacks, being $\times 2.27$ more query-efficient in our experiments. Also, ours outperformed other attacks even when the victim's outputs were perturbed by the defense mechanism or constrained to hard labels.

2 Related Works

2.1 Model Stealing Attacks

We summarize recent advances in model stealing attacks.

Sampling from Public Datasets. KnockoffNet [Orekondy *et al.*, 2019] showed that public datasets, that may not follow the victim's exact training distribution, can be used effectively for model-stealing queries. ActiveThief [Pal *et al.*, 2020] improved the idea by introducing active sampling mechanisms such as uncertainty-based sampling [Lewis and Gale, 1994]. Recently, MExMI [Xiao *et al.*, 2022] suggested using inputs with high similarity to the victim's training set, identified by the membership inference attack [Shokri *et al.*, 2017].

Data Generation. JBDA [Papernot *et al.*, 2016] introduced a data augmentation technique using a similar mechanism to FGSM [Goodfellow *et al.*, 2015]. Recently, CloudLeak [Yu *et al.*, 2020] suggested an enhanced adversarial sampling using the FeatureFool, an objective function representing the distance between the augmentation target and a guide input on the representation space. InverseNet [Gong *et al.*, 2021] applied the model inversion attack [Fredrikson *et al.*, 2015] to the victim and tried to generate attack queries following a similar distribution to the victim's training data. MAZE [Kariyappa *et al.*, 2021] and DFME [Truong *et al.*, 2021] utilized generative models [Goodfellow *et al.*, 2014] to generate inputs that maximize the disagreement between the victim's query responses and the predictions of the clone model. [Zhou *et al.*, 2020; Zhang *et al.*, 2022] improved the training loss for generative models to ensure the class diversity of synthesized inputs.

Utilization of Unqueried Data. The above methods have not considered using unqueried data for speeding up model-stealing attacks. Although ActiveThief considered unqueried data in its active sampling, it did not use them for actual stealing. [Jagielski *et al.*, 2020; Xiao *et al.*, 2022] showed that semi-supervised learning mechanisms like MixMatch [Berthelot *et al.*, 2019] could improve model stealing; however, our experience indicates that these methods may not be as effective as reported in the original experiments when an attacker has access to no sample from the victim's data distribution, which is the setting we consider in this work. [Zhao *et al.*, 2023] utilizes self-supervised contrastive learning [Chen *et al.*, 2020; He *et al.*, 2020] but only as a pre-training step for model stealing; our work further utilizes the knowledge from queried data for contrastive learning by introducing a new custom-designed soft-supervised contrastive loss for model stealing.

Defenses against Model Stealing. One of the mainstream approaches of model-stealing defense is to disrupt the training of clone models by introducing alterations to the outputs of the victim model. For instance, prediction poisoning [Orekondy *et al.*, 2020] and GRAD^2 [Mazeika *et al.*, 2022] introduced controlled perturbations to the victim model's probability outputs. DeepDefense [Lee *et al.*, 2022] suggested a different approach that exposed the probability outputs and the attribution maps [Selvaraju *et al.*, 2019; Lee and Han, 2022] from a misdirection model to deliver maximally distorted gradient information.

2.2 Contrastive Learning

Self-supervised contrastive learning has recently gained massive attention due to its capability to learn highly transferable features [Islam *et al.*, 2021; Liu *et al.*, 2022a]. SimCLR [Chen *et al.*, 2020] provided one of the first frameworks to use positive and negative pairs for representation learning. It requires a large batch size to keep plentiful negative samples and thereby avoid collapsed representation. MOCO [He *et al.*, 2020] provided a more efficient way to keep large negative samples by managing an embedding queue of negative samples across several recent batches. More recently, SimSiam [Chen and He, 2021] suggested the stop-gradient mechanism that detaches the positive samples from the computation graph of back-propagation to prevent collapsed representation.

Another direction of contrastive learning uses supervision [Khosla *et al.*, 2020] for composing positive and negative pairs. However, the existing framework allows only hard labels, whereas, in model-stealing, many victim models provide soft labels, e.g., the softmax probabilities. In this paper, we adapt the self-supervised and supervised contrastive learning frameworks for model stealing so that unqueried and queried data with soft labels can be utilized for maximal attack efficiency.

3 Threat Model

Model stealing assumes that a trained AI model (the victim) f is in service, providing an output for each query input $x \in \mathbb{R}^d$ from users. We assume that the victim model is a black box; that is, the details about the victim are hidden from attackers, including the model's architecture, parameters, learning algorithms, hyperparameters, the data distribution P_V, and training data. We suppose the victims provide softmax probabilities $f(x) \in \Delta^K$ regarding the input x, where Δ^K is a K-dimensional simplex.

The goal of a model stealing adversary is to train a clone model f_A, which mimics the functionality of the victim model f (we focus on achieving the victim's test accuracy on unseen data from P_V). To extract useful information from the victim, the adversary prepares a surrogate dataset $S := \{x : x \sim P_A\}$ sampled from the adversary's data distribution P_A (in general, $P_A \neq P_V$), and uses some $x \in S$ as queries to the victim to obtain the corresponding outputs $y = f(x) \in \Delta^K$. We denote the adversary's query budget as B and the set of (query, response) pairs of currently queried inputs as $Q := \{(x, y) : x \in S, y = f(x)\}$ such that $|Q| = q$. We denote the unqueried remainder as $U := S \setminus \{x \in S : (x, f(x)) \in Q\}$ where $|U| = u$ so that $|S| = q + u$. In classical model stealing, the attacker trains their clone model f_A with Q by minimizing the loss between the query outputs and the prediction results of the clone model, that is, $\sum_{(x,y)\in Q} L(f_A(x), y)$. Here, the function L can be the cross-entropy loss [Orekondy *et al.*, 2019] or the ℓ_p distance [Truong *et al.*, 2021] depending on attack methods.

4 SwiftThief

We propose a new model stealing attack called SwiftThief, which utilizes both the queried dataset Q and the unqueried remainder U to maximize cloning efficiency. To achieve the goal, we introduce an objective function consisting of two parts: $\mathcal{L}_\mathrm{c}$ for contrastive representation learning (which combines soft-supervised and self-supervised losses using Q and U, respectively) and $\mathcal{L}_\mathrm{m}$ for matching the outcomes of the clone and the victim models based on Q.

Our method uses a representation function $f_\mathrm{r}(\cdot; w_\mathrm{r}) : \mathbb{R}^d \to \mathbb{R}^a$ and a classification head $f_\mathrm{m}(\cdot; w_\mathrm{m}) : \mathbb{R}^a \to \Delta^K$, so that the clone model is defined as $f_A(\cdot; w_\mathrm{r}, w_\mathrm{m}) := f_\mathrm{m} \circ f_\mathrm{r}(\cdot)$. For contrastive learning, we use an auxiliary encoder $f_\mathrm{e}(\cdot; w_\mathrm{r}, w_\mathrm{h}) := f_\mathrm{h} \circ f_\mathrm{r}(\cdot)$ defined with a projection head $f_\mathrm{h}(\cdot; w_\mathrm{h}) : \mathbb{R}^a \to \mathbb{R}^{a'}$ and use a prediction head $f_\mathrm{p}(\cdot; w_\mathrm{p}) : \mathbb{R}^{a'} \to \mathbb{R}^{a'}$ to avoid collapsed representation as in SimSiam. Here, a and a' are the chosen dimensions of the embedding spaces. Figure 2 illustrates the composition of SwiftThief.

4.1 Learning Objective

Self-Supervised Contrastive Loss for Unqueried Data. To learn representations with the unqueried inputs in U, we use the self-supervised contrastive learning framework of SimSiam [Chen and He, 2021]. (SimSiam outperformed other alternatives such as SimCLR and MOCO in our experience.) We generate positive pairs from U as follows: given $2u$ transformations $\pi_1, \cdots, \pi_{2u}$ independently sampled from a transformation distribution Π, each input $x_i \in U$ is augmented as the positive pair $\tilde{x}_{2i-1} := \pi_{2i-1}(x_i)$ and $\tilde{x}_{2i} := \pi_{2i}(x_i)$, indexed by the set P with elements $(2i-1, 2i)$ for $i = 1, \ldots, u$. Using the encoded versions of the augmentations $z = f_\mathrm{e}(\tilde{x}) \in \mathbb{R}^{a'}$ and $z' = f_\mathrm{p}(f_\mathrm{e}(\tilde{x})) \in \mathbb{R}^{a'}$, we define the self-supervised contrastive loss for U as:

$$\mathcal{L}_\mathrm{c}^\mathrm{self}(w_\mathrm{r}, w_\mathrm{h}, w_\mathrm{p}) = -\mathbb{E}_{(\pi_1,\cdots,\pi_{2u})\sim\Pi}\left[\frac{1}{2|P|}\sum_{(i,j)\in P}(z_i^\top z'_j + z_j^\top z'_i)\right]. \tag{1}$$

Here, we treat z_i and z_j as constants, excluding them from the back-propagation during the optimization process as in the original paper of SimSiam [Chen and He, 2021].

Soft-Supervised Contrastive Loss for Queried Data. We propose a new soft-supervised contrastive loss to use the available soft labels during model stealing, generalizing the hard-label version in [Khosla *et al.*, 2020]. Given transformations $\hat{\pi}_1, \cdots, \hat{\pi}_{2q} \sim \hat{\Pi}$, we create two new augmented inputs $(\hat{x}_{2i-1}, \hat{y}_{2i-1}) := (\hat{\pi}_{2i-1}(x_i), y_i)$ and $(\hat{x}_{2i}, \hat{y}_{2i}) := (\hat{\pi}_{2i}(x_i), y_i)$ from each pair $(x_i, y_i) \in Q$ which consists of the queried input x_i and the corresponding victim's probability output $y_i \in \Delta^K$. Using the encoded versions of the augmentation $\hat{z} = f_\mathrm{e}(\hat{x})$ and $\hat{z}' = f_\mathrm{p}(f_\mathrm{e}(\hat{x}))$, we propose the new soft-supervised contrastive loss as follows:

$$\mathcal{L}_\mathrm{c}^\mathrm{soft}(w_\mathrm{r}, w_\mathrm{h}, w_\mathrm{p}) = -\mathbb{E}_{(\hat{\pi}_1,\cdots,\hat{\pi}_{2q})\sim\hat{\Pi}}\left[\sum_{i=1}^{2q}\sum_{j=1}^{2q}\eta_{ij}(\hat{z}_i^\top \hat{z}'_j + \hat{z}_j^\top \hat{z}'_i)\right]. \tag{2}$$

Here, $\eta_{ij} \in [0, 1]$ is a variable representing each pair's representation alignment. We designed η_{ij} to adjust the alignment intensity adaptively by focusing not only on pairs with identical class membership in the top-1 sense but also on those pairs whose entire softmax probability distributions show high similarity with great confidence:

$$\eta_{ij} := \mathbf{1}_{[i\neq j]}\left(1 + \frac{H(\hat{y}_i)}{\log K}\right)\left(1 + \frac{H(\hat{y}_j)}{\log K}\right)\cos\angle(\hat{y}_i, \hat{y}_j). \tag{3}$$

We denote by $\cos\angle(a, b)$ the cosine of the angle between vectors a and b, by $H(\cdot)$ the Shannon entropy $H(y) = -\sum_{\ell=1}^K (y)_\ell \log(y)_\ell$, where $(y)_\ell$ is the ℓ-th element of the vector y, and by $\mathbf{1}_{[c]}$ an indicator function taking the value of 1 when the condition c is true and 0 otherwise. The expressions in the parentheses of (3) represent reversed normalized entropy where the values become 0 for uniformly distributed victim's softmax outputs and 1 for one-hot outputs. It penalizes the intensity for inputs with high uncertainty in the victim's outputs, considering the risk associated with the potential misclassification of the victim. Note that given hard labels as one-hot vectors, (2) reduces to the existing loss for supervised contrastive learning [Khosla *et al.*, 2020].

Strong Regularization on Minority Classes. As noted in [Cao *et al.*, 2019; Cao *et al.*, 2021; Liu *et al.*, 2022b], minority classes can cause overfitting and need strong regularization. Motivated by this, we introduce a regularizer based on adversarial contrastive learning [Kim *et al.*, 2020]. For the two augmented inputs $\hat{x}_{2i-1} := \hat{\pi}_{2i-1}(x_i)$ and $\hat{x}_{2i} := \hat{\pi}_{2i}(x_i)$ and their respective encoded versions $\hat{z} = f_\mathrm{e}(\hat{x})$ and $\hat{z}' =$

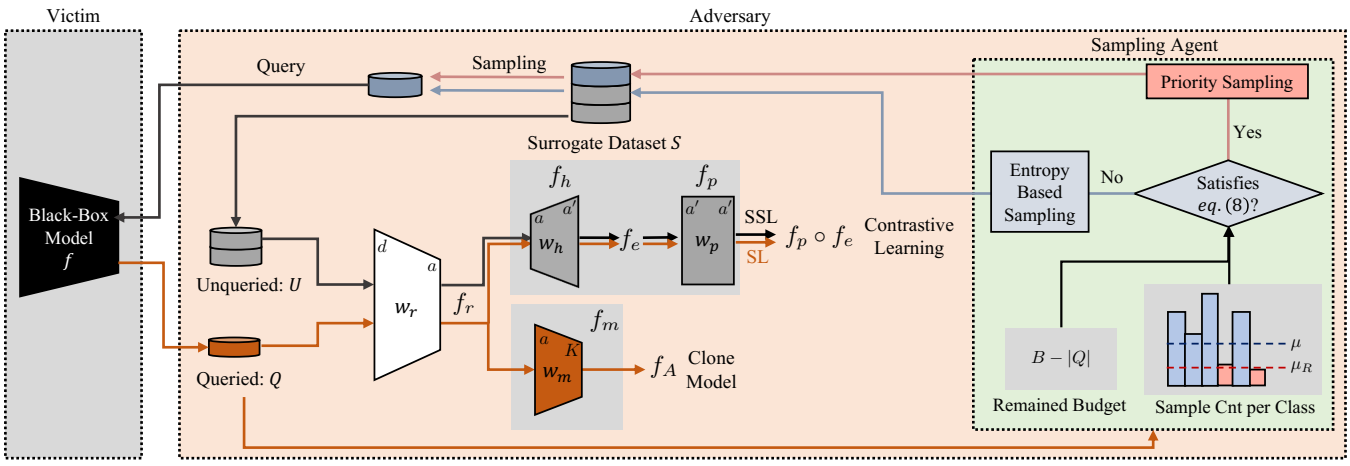

Figure 2: The schematic of SwiftThief.

$f_{\mathrm{p}}(f_{\mathrm{e}}(\hat{x}))$ we have used to construct (2), we consider an additive perturbation $\delta \in \mathbb{R}^d$ and denote $\hat{z}(\delta) = f_{\mathrm{e}}(\hat{x} + \delta)$ and $\hat{z}'(\delta) = f_{\mathrm{p}}(f_{\mathrm{e}}(\hat{x} + \delta))$ to define our regularizer as:

$$\mathcal{L}_{\mathrm{c}}^{\mathrm{reg}}(w_{\mathrm{r}}, w_{\mathrm{h}}, w_{\mathrm{p}}) = -\mathbb{E}_{(\hat{\pi}_1, \cdots, \hat{\pi}_{2q}) \sim \hat{\Pi}} \left[\sum_{i=1}^{q} c_{y_i} \max_{\|\delta\|_\infty \leq \epsilon} \left(\hat{z}_{2i-1}(\delta)^\top \hat{z}'_{2i} + \hat{z}'_{2i-1}(\delta)^\top \hat{z}_{2i} \right) \right], \tag{4}$$

where $\epsilon > 0$ is a hyperparameter that constrains the magnitude of the perturbation δ. We define the weight $c_{y_i} := (1-\beta)/(1-\beta^{N_{y_i}})$ following [Cui *et al.*, 2019], where N_{y_i} represents the number of samples in the class y_i and $\beta \in [0, 1)$ is a hyperparameter: the rarer y_i is, c_{y_i} takes larger values. The regularizer is intended to impose a strong penalty to the sharpness of the representation function around minor classes.

Combining the self-supervised (1), soft-supervised losses (2), and the regularizer (4), we construct the loss for our contrastive representation learning as follows ($\lambda_1 > 0$ and $\lambda_2 > 0$ are hyperparameters):

$$\mathcal{L}_{\mathrm{c}}(w_{\mathrm{r}}, w_{\mathrm{h}}, w_{\mathrm{p}}) := \mathcal{L}_{\mathrm{c}}^{\mathrm{self}}(w_{\mathrm{r}}, w_{\mathrm{h}}, w_{\mathrm{p}}) + \lambda_1 \mathcal{L}_{\mathrm{c}}^{\mathrm{soft}}(w_{\mathrm{r}}, w_{\mathrm{h}}, w_{\mathrm{p}}) + \lambda_2 \mathcal{L}_{\mathrm{c}}^{\mathrm{reg}}(w_{\mathrm{r}}, w_{\mathrm{h}}, w_{\mathrm{p}}). \tag{5}$$

Output Matching for Queried Data. For model stealing, we need to match the clone model's output $f_A(x_i; w_{\mathrm{r}}, w_{\mathrm{m}})$ to the victim's response y_i on the responded queries $(x_i, y_i) \in Q$. For this, we define the matching loss based on the cross-entropy function,

$$\mathcal{L}_{\mathrm{m}}(w_{\mathrm{r}}, w_{\mathrm{m}}) := -\frac{1}{q} \sum_{i=1}^{q} \sum_{\ell=1}^{K} (y_i)_\ell \log(f_A(x_i; w_{\mathrm{r}}, w_{\mathrm{m}})_\ell). \tag{6}$$

Finally, we define the total loss for SwiftThief using the contrastive (5) and the matching losses (6):

$$\mathcal{L}(w_{\mathrm{r}}, w_{\mathrm{h}}, w_{\mathrm{p}}, w_{\mathrm{m}}) := \mathcal{L}_{\mathrm{c}}(w_{\mathrm{r}}, w_{\mathrm{h}}, w_{\mathrm{p}}) + \lambda_3 \mathcal{L}_{\mathrm{m}}(w_{\mathrm{r}}, w_{\mathrm{m}}),$$

where $\lambda_3 > 0$ is a hyperparameter. In Algorithm 1, we outline the overall process of SwiftThief. In the algorithm (lines 15 and 16), we use an alternating optimization: at the k-th outer iteration, we minimize $\mathcal{L}_{\mathrm{c}}(w_{\mathrm{r}}, w_{\mathrm{h}}, w_{\mathrm{p}})$ to obtain $(w_{\mathrm{r}}^{k+\frac{1}{2}}, w_{\mathrm{h}}^{k+1}, w_{\mathrm{p}}^{k+1})$ and then minimize $\mathcal{L}_{\mathrm{m}}(w_{\mathrm{r}}, w_{\mathrm{m}})$, using $w_{\mathrm{r}}^{k+\frac{1}{2}}$ for warm-starting, to obtain $(w_{\mathrm{r}}^{k+1}, w_{\mathrm{m}}^{k+1})$. Our approach can be seen as a relaxed form of block coordinate descent [Tseng, 2001] to minimize $\mathcal{L}$, where we avoided, by warm-starting, the explicit duplication of the shared variable w_{r} between $\mathcal{L}_{\mathrm{c}}$ and $\mathcal{L}_{\mathrm{m}}$ required to form non-overlapping blocks of variables. This reduces the memory footprint and the dependency on the extra hyperparameter λ_3.

4.2 Prioritization of Rarely Queried Classes in Sampling

As briefly discussed in Sect. 1 and Figure 1, class imbalance in the queried samples can deteriorate an attacker's performance. However, class-balanced sampling of attack queries is challenging without making extra queries to the victim to obtain prediction results on unqueried samples.

To address the issue, we try to detect class imbalance in queried samples and prioritize rarely queried classes in future sampling. We conjecture that the samples from the same class (according to the victim) will be located nearby in the representation space of the attack model trained by contrastive learning. Based on this, we apply kernel density estimation to the representations of the most rarely sampled class in Q and select samples from U with the highest probability according to the estimated density function.

Specifically, we call a class rarely represented if it has a smaller sample size than the mean sample size of all classes in Q, where the class memberships of the samples in Q are determined by the maximum softmax probability from the victim's outputs. Let Q_{y_n} represent the inputs x_i in Q belonging to y_n, the most rarely represented class in Q. Also let $\kappa(\cdot, \sigma) = \exp\left(-\|\cdot\|_2^2/(2\sigma^2)\right)$ denote the Gaussian kernel with the bandwidth $\sigma > 0$. Using the so-far trained representation function $f_{\mathrm{r}}(\cdot; w_{\mathrm{r}})$, we assign the density score of each

query candidate $x_j \in U$ as follows:

$$s_j = \sum_{x_i \in Q_{y_n}} \kappa(f_{\mathrm{r}}(x_j; w_{\mathrm{r}}) - f_{\mathrm{r}}(x_i; w_{\mathrm{r}}), \sigma). \tag{7}$$

We choose the samples $x_j \in U$ with the largest density scores s_j that are most likely to be the class y_n if queried in the future, according to our conjecture. In fact, given the total query budget B and the number of outer iterations I of SwiftThief, we sample $\lceil B/I \rceil$ samples from U using this strategy. Also, to avoid the minority class from being too large after accessing the victim, we divide the above process into b steps: in each step, we sample $\lceil (B/I)/b \rceil$ from y_n and recompute y_n after the assignment of classes in the previous steps. (We used $b = 5$ and set B as a multiple of $I \cdot b$ in experiments.) However, samples chosen by the above prioritization could be far from the decision boundary, making newly chosen samples less informative for improving an attack model. Therefore, we allow switching our prioritization with entropy-based sampling [Lewis and Gale, 1994]. In particular, inspired by [Aggarwal *et al.*, 2020], we start our attack with entropy-based sampling and use our prioritization strategy when the following condition is satisfied:

$$B - |Q| \leq N_R \cdot (\mu - \mu_R). \tag{8}$$

Here, B is the total query budget for the attack, $|Q|$ is the number of queried samples so far, and N_R is the number of rarely queried classes. Also, μ and μ_R are the mean sample counts of the entire and rare classes in Q, respectively. The switching of sampling strategies is elaborated in the lines 4–10 of Algorithm 1.

Algorithm 1 The SwiftThief Algorithm

Require: Query budget B, max. iterations I, a surrogate dataset S, $\lambda_1 > 0$, $\lambda_2 > 0$, $\beta \in [0, 1)$ and $\sigma > 0$.

1: Randomly initialize $w_{\mathrm{r}}^0, w_{\mathrm{m}}^0, w_{\mathrm{h}}^0, w_{\mathrm{p}}^0$;
2: Let $k \leftarrow 0$, $Q \leftarrow \emptyset$ and $U \leftarrow S$;
3: **while** $k < I$ **do**
4: **if** $k = 0$ **then**
5: sampling-strategy $\leftarrow$ uniform random sampling;
6: **else if** $B - |Q| \leq N_R \cdot (\mu - \mu_R)$ **then**
7: sampling-strategy $\leftarrow$ sampling with rarely-queried class prioritization (using (7));
8: **else**
9: sampling-strategy $\leftarrow$ entropy-based sampling;
10: **end if**
11: Sample $\{x_1, \cdots, x_{B/I}\} \subset U$ using sampling-strategy;
12: Obtain $\{f(x_1), \cdots, f(x_{B/I})\}$ by querying f;
13: $Q \leftarrow Q \cup \{(x_1, f(x_1)), \cdots, (x_{B/I}, f(x_{B/I}))\}$;
14: $U \leftarrow U \setminus \{x_1, \cdots, x_{B/I}\}$;
15: Update $(w_{\mathrm{r}}^{k+\frac{1}{2}}, w_{\mathrm{h}}^{k+1}, w_{\mathrm{p}}^{k+1})$ by minimizing (5);
16: Update $(w_{\mathrm{r}}^{k+1}, w_{\mathrm{m}}^{k+1})$ by minimizing (6), warm-starting w_r from $w_{\mathrm{r}}^{k+\frac{1}{2}}$;
17: $k \leftarrow k + 1$;
18: **end while**
19: **return** $f_A(\cdot; w_{\mathrm{r}}^I, w_{\mathrm{m}}^I)$;

5 Experiments

5.1 Experimental Setup

Victim. We used ResNet-18 [He *et al.*, 2016] as the victim model, which was trained with five task-specific image datasets: MNIST [Lecun *et al.*, 1998], SVHN [Netzer *et al.*, 2011], GTSRB [Stallkamp *et al.*, 2011], CIFAR-10 [Krizhevsky, 2009], and EuroSAT [Helber *et al.*, 2018]. Test accuracy rates for each victim model were 98.46% for MNIST, 94.81% for SVHN, 96.68% for GTSRB, 94.46% for CIFAR-10, and 96.67% for EuroSAT.

Adversary. For comparison, we chose six state-of-the-art model-stealing attacks: KnockoffNet [Orekondy *et al.*, 2019], ActiveThief [Pal *et al.*, 2020], MixMatch [Berthelot *et al.*, 2019], InverseNet [Gong *et al.*, 2021], MExMI [Xiao *et al.*, 2022], and EPK (Extraction from Prior Knowledge) [Zhao *et al.*, 2023]. For the surrogate dataset S to choose attack queries from, we used the trainset of ILSVRC-2012 [Russakovsky *et al.*, 2015]. We note that the original MixMatch and MExMI assumed the adversary could access victims' training data. However, since our threat model assumes a more realistic setting where victims' data distributions are hidden, we performed these attacks without using victims' data. We used ResNet-18 as the clone model, which is identical to the victim model's architecture. We set the query budget B to $30,000$ unless otherwise stated. Finally, we used the clone model's accuracy on victims' test sets to measure model stealing performance.

Setup for SwiftThief. We used the part of the clone model from the input to the penultimate layer as the representation function f_{r} and the remainder to the output layer as the classification head f_{m}. For the contrastive representation learning in SwiftThief (denoted by ST), we set the dimensions a and a' as 512 and $2,048$ respectively as in the original SimSiam [Chen and He, 2021]. For solving the inner-maximization problem in (4), we adopted FGSM [Goodfellow *et al.*, 2015] with ϵ of 0.01. For the alternating optimization in ST, we set the number of outer iterations I to 10 and the number of epochs for each sub-problem to 40 while increasing the epochs for contrastive representation learning to 100 in the last outer iteration to ensure sufficient convergence. We set λ_1 and λ_2 in the loss of ST (5) to 1.0 and 0.01, resp. For the entire ILSVRC-2012 trainset S (with $1,281,167$ samples) and its unqueried portion U, we used randomly sampled $50,000$ inputs from U for the self-supervised contrastive learning of ST.

5.2 Results

RQ: How much improvement in attack can ST bring? To demonstrate the efficacy of our ST, we compared the test accuracy of the clone models produced by the attack methods mentioned above on nine different query budgets from 2K to 50K. Figure 3 shows the results: ST outperformed the competitors consistently on all query budgets and victim models tried. In particular, ST was considerably more effective at stealing with small query budgets. For example, at relatively small budgets between 2K and 8K, ST achieved $\times 2.66$, $\times 1.33$, $\times 1.67$, $\times 1.21$, and $\times 1.25$ attack performances compared to the second best attacks, on MNIST, SVHN, GTSRB, CIFAR-10, and

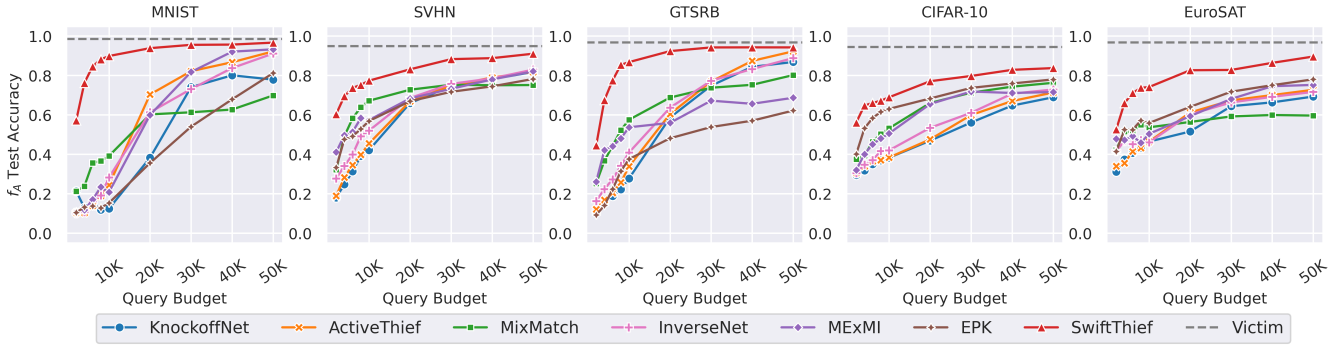

Figure 3: Model-stealing performance in test accuracy of clone models on five victim tasks.

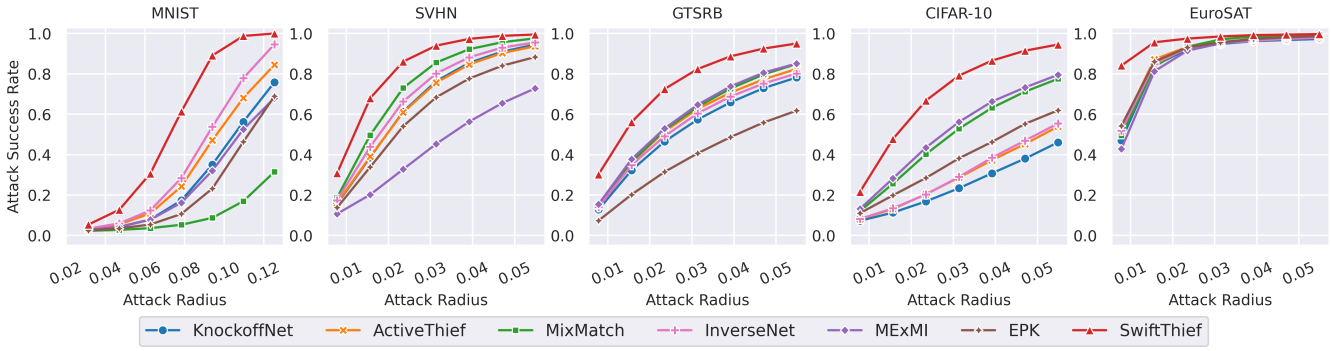

Figure 4: Attack success rates of black-box adversarial attacks through model stealing.

EuroSAT, respectively; for larger query budgets between 10K and 50K, it achieved $\times 1.37$, $\times 1.14$, $\times 1.23$, $\times 1.09$, and $\times 1.21$ attack performances compared to the second-best, respectively for the victims. Also, at the maximum query budget of 50K in our experiments, ST's clone models reached $\times 0.95$ of the victim's test accuracy on average, while the second-best attack methods achieved only $\times 0.88$ in contrast.

RQ: How effective is ST against defense? To demonstrate how well ST performs when a victim is equipped with a defense mechanism, we compared the performance of attacks against victims with and without one of the popular defense methods called Prediction Poisoning [Orekondy *et al.*, 2020] (denoted by PP). PP adds perturbations to the softmax outputs of the victim so that the attacker's training will use misleading gradient information. In Table 1, we can see that ST outperformed other attacks in terms of the test accuracy of the clone models even when PP was active. In particular, the decrease in attack performance due to PP was much smaller for ST than for other methods relying entirely on the queried dataset, namely KnockoffNet, ActiveThief, and InverseNet. On average, the attack performance of ST decreased to $\times 0.94$, while the performance of KnockoffNet, ActiveThief, and InverseNet decreased significantly to $\times 0.82$, $\times 0.86$ and $\times 0.84$, respectively. On the other hand, for MixMatch, MExMI, and EPK, the approaches using both queried and unqueried datasets, the performance drop was relatively small ($\times 0.92$, $\times 0.93$, and $\times 0.90$, resp.). Still, our method ST performed better than them in both defended and undefended cases.

RQ: How effective is ST in black-box adversarial attacks? To investigate the effectiveness of ST against black-box adversarial attacks, we applied adversarial attacks to the clone models generated by each model-stealing method. For the investigation, we adopted the PGD attack [Madry *et al.*, 2018] varying the perturbation radius ϵ in the ℓ_∞ norm from $8/255$ to $32/255$ for MNIST, and from $2/255$ to $14/255$ for other datasets. (MNIST generally requires stronger perturbations than other datasets for successful attacks [Ye *et al.*, 2019; Croce and Hein, 2020].) Figure 4 shows the attack success rates of the victim models on adversarial examples generated through the corresponding clone models. The result shows that black-box adversarial attacks through model stealing were the most successful with our attack method across all attack radii.

RQ: How effectively does ST deal with the class imbalance in queries? To investigate the RQ, we compared the sample ratios of the rarest classes to the most frequent classes in the queried datasets (denoted by the balance score; higher values mean better balancing) and the class-wise test accuracy of the clone models generated by ST and the best competitor (in terms of the test accuracy) for each victim task in Figure 5. We can see that ST significantly increases the balance score compared to the best competitor. Also, we can see the effect of making more class-balanced queries on attack performance. Regarding class-wise test accuracy rates, the best competitor fails to steal the model in particular classes (e.g., the class 7 of GTSRB) due to rare attack queries made for the class, while

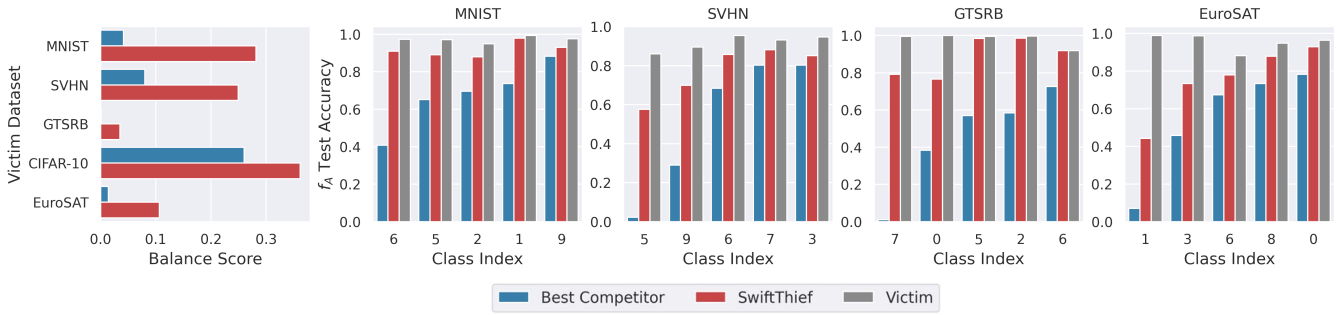

Figure 5: The first panel: the sample ratio of the rarest to the most frequent class (denoted by balance score). The second to the last panels: clone model's test accuracy for each class: class index (x-axis) were sorted in ascending order based on the class-wise test accuracy of the best competitor. The first five classes are shown due to space limits. CIFAR-10 is excluded since it showed the least class imbalance.

Attack	Victim Dataset MNIST		SVHN		GTSRB		CIFAR-10		EuroSAT	
	Undef	PP	Undef	PP	Undef	PP	Undef	PP	Undef	PP
KnockoffNet	74.07	53.34	74.56	63.26	74.80	65.35	56.12	41.00	64.39	60.07
ActiveThief	82.16	71.54	74.78	60.31	76.89	75.23	59.92	40.02	67.35	66.07
MixMatch	61.30	45.65	75.27	70.99	73.76	71.14	71.26	65.90	59.20	60.52
InverseNet	73.10	66.15	75.72	64.31	77.29	58.64	61.06	46.39	66.30	60.94
MExMI	81.75	71.53	73.21	70.75	67.20	63.05	71.83	66.77	68.07	64.00
EPK	54.01	44.38	71.68	69.41	53.82	49.96	73.68	68.21	71.74	63.06
SwiftThief	95.51	94.76	88.26	84.24	94.20	89.55	79.73	69.12	82.78	79.26

Table 1: The effect of defense on attack performance. Clone models' test accuracy rates (%) on victims' tasks are shown, created against undefended (Undef) and PP-defended (PP) victim models. The gray highlight indicates best-performing attack for each task.

Victim Dataset	Best Competitor Method	Best Competitor f_A Acc(%)	SwiftThief f_A Acc (%)
MNIST	InverseNet	59.33	80.87
SVHN	MixMatch	70.70	76.61
GTSRB	MExMI	67.11	77.61
CIFAR-10	MixMatch	66.56	67.70
EuroSAT	MExMI	63.33	70.37

Table 2: The attack performance in the hard label scenario. Clone models' test accuracy rates (%) on victims' tasks are shown, created by SwiftThief and the best competitor for each task.

Components of SwiftThief $\mathcal{L}_c^{soft}$	L_c^{self}	L_c^{reg}	Sampling	f_A Test Acc(%)
✗	✗	✗	✗	72.22 (×1.00)
✓	✗	✗	✗	79.62 (×1.10)
✓	✓	✗	✗	85.46 (×1.18)
✓	✓	✓	✗	86.76 (×1.20)
✓	✓	✓	✓	88.09 (×1.22)

Table 3: The contribution of the components of SwiftThief to model-stealing performance.

ST alleviates the problem significantly.

RQ: How does ST perform with different label types? Next, we investigated how effective ST is when only class membership information is provided as the query output. Table 2 displays the test accuracy of clone models created by ST and the best competitor for each victim task in the hard label scenario. We can see that our ST outperforms the best competitors in all victim tasks.

RQ: What are the contributions of the suggested components? Finally, we conducted an ablation study to see the contribution of using the soft-supervised contrastive loss $\mathcal{L}_c^{soft}$, the self-supervised contrastive loss $\mathcal{L}_c^{self}$, the weighted regularizer $\mathcal{L}_c^{reg}$, and the prioritization of rarely queried classes in sampling. Table 3 shows the test accuracy of ST's f_A averaged on five victim tasks. We can see that the entire components increased the clone models' test accuracy compared to previous cases, showing that all components of ST contribute to improved attack performance.

6 Conclusion

We proposed SwiftThief, a new query-efficient model-stealing attack framework that uses both queried and unqueried data utilizing soft- and self-supervised contrastive representation learning. We further enhanced our attack by mitigating the class imbalance problem using a new regularizer and an enhanced sampling strategy. Through experiments, we confirmed that SwiftThief outperforms current state-of-the-art attacks even with smaller queries and when a defense is applied. Our future works include extending SwiftThief to other data domains and sophisticating it to exploit new attack surfaces, such as the attribution maps for eXplainable AI (XAI).

Acknowledgments

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2024-2020-0-01749) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation).

References

[Aggarwal *et al.*, 2020] Umang Aggarwal, Adrian Popescu, and Céline Hudelot. Active learning for imbalanced datasets. In *WACV*, 2020.

[Berthelot *et al.*, 2019] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019.

[Cao *et al.*, 2019] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*, 2019.

[Cao *et al.*, 2021] Kaidi Cao, Yining Chen, Junwei Lu, Nikos Arechiga, Adrien Gaidon, and Tengyu Ma. Heteroskedastic and imbalanced deep learning with adaptive regularization. In *ICLR*, 2021.

[Chen and He, 2021] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021.

[Chen *et al.*, 2020] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[Croce and Hein, 2020] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

[Cui *et al.*, 2019] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, 2019.

[Fredrikson *et al.*, 2015] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM SIGSAC CCS*, 2015.

[Gong *et al.*, 2021] Xueluan Gong, Yanjiao Chen, Wenbin Yang, Guanghao Mei, and Qian Wang. Inversenet: Augmenting model extraction attacks with training data inversion. In *IJCAI*, 2021.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *NeurIPS*, 2014.

[Goodfellow *et al.*, 2015] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[He *et al.*, 2020] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[Helber *et al.*, 2018] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS*, 2018.

[Hinton *et al.*, 2014] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS workshop*, 2014.

[Islam *et al.*, 2021] Ashraful Islam, Chun-Fu Chen, Rameswar Panda, Leonid Karlinsky, Richard Radke, and Rogerio Feris. A broad study on the transferability of visual representations with contrastive learning. In *ICCV*, 2021.

[Jagielski *et al.*, 2020] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *USENIX Security*, 2020.

[Kariyappa *et al.*, 2021] Sanjay Kariyappa, Atul Prakash, and Moinuddin Qureshi. Maze: Data-free model stealing attack using zeroth-order gradient estimation. In *CVPR*, 2021.

[Khosla *et al.*, 2020] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *NeurIPS*, 2020.

[Kim *et al.*, 2020] Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. In *NeurIPS*, 2020.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[Lecun *et al.*, 1998] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Lee and Han, 2022] Sangkyun Lee and Sungmin Han. Libracam: An activation-based attribution based on the linear approximation of deep neural nets and threshold calibration. In *IJCAI*, 2022.

[Lee *et al.*, 2022] Jeonghyun Lee, Sungmin Han, and Sangkyun Lee. Model stealing defense against exploiting information leak through the interpretation of deep neural nets. In *IJCAI*, 2022.

[Lewis and Gale, 1994] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR*, 1994.

[Liu *et al.*, 2022a] Hong Liu, Jeff Z. HaoChen, Adrien Gaidon, and Tengyu Ma. Self-supervised learning is more robust to dataset imbalance. In *ICLR*, 2022.

[Liu *et al.*, 2022b] Hong Liu, Jeff Z. HaoChen, Adrien Gaidon, and Tengyu Ma. Self-supervised learning is more robust to dataset imbalance. In *ICLR*, 2022.

[Madry *et al.*, 2018] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[Mazeika *et al.*, 2022] Mantas Mazeika, Bo Li, and David Forsyth. How to steer your adversary: Targeted and efficient model stealing defenses with gradient redirection. In *ICML*, 2022.

[Netzer *et al.*, 2011] Yuval Netzer, Tao Wang, Adam Coates, A. Bissacco, Bo Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS workshop*, 2011.

[Orekondy *et al.*, 2019] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *CVPR*, 2019.

[Orekondy *et al.*, 2020] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against dnn model stealing attacks. In *ICLR*, 2020.

[Pal *et al.*, 2020] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish K. Shevade, and Vinod Ganapathy. Activethief: Model extraction using active learning and unannotated public data. In *AAAI*, 2020.

[Papernot *et al.*, 2016] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ACM ASIACCS*, 2016.

[Ribeiro *et al.*, 2015] Mauro Ribeiro, Katarina Grolinger, and Miriam A.M. Capretz. MLaaS: Machine Learning as a Service. In *ICMLA*, 2015.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[Selvaraju *et al.*, 2019] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *IJCV*, 2019.

[Shokri *et al.*, 2017] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE S&P*, 2017.

[Stallkamp *et al.*, 2011] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE IJCNN*, 2011.

[Tramèr *et al.*, 2016] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security*, 2016.

[Truong *et al.*, 2021] Jean-Baptiste Truong, Pratyush Maini, Robert J. Walls, and Nicolas Papernot. Data-free model extraction. In *CVPR*, 2021.

[Tseng, 2001] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475, 2001.

[Xiao *et al.*, 2022] Yaxin Xiao, Qingqing Ye, Haibo Hu, Huadi Zheng, Chengfang Fang, and Jie Shi. MExMI: Pool-based active model extraction crossover membership inference. In *NeurIPS*, 2022.

[Ye *et al.*, 2019] Shaokai Ye, Kaidi Xu, Sijia Liu, Jan-Henrik Lambrechts, Huan Zhang, Aojun Zhou, Kaisheng Ma, Yanzhi Wang, and Xue Lin. Adversarial robustness vs. model compression, or both? In *ICCV*, October 2019.

[Yu *et al.*, 2020] Honggang Yu, Kaichen Yang, Teng Zhang, Yun-Yun Tsai, Tsung-Yi Ho, and Yier Jin. Cloudleak: Large-scale deep learning models stealing through adversarial examples. In *NDSS*, 2020.

[Zhang *et al.*, 2022] Jie Zhang, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, Lei Zhang, and Chao Wu. Towards efficient data free black-box adversarial attack. In *CVPR*, pages 15115–15125, June 2022.

[Zhao *et al.*, 2023] Shiqian Zhao, Kangjie Chen, Meng Hao, Jian Zhang, Guowen Xu, Hongwei Li, and Tianwei Zhang. Extracting cloud-based model with prior knowledge. In *arXiv*, 2023.

[Zhou *et al.*, 2020] Mingyi Zhou, Jing Wu, Yipeng Liu, Shuaicheng Liu, and Ce Zhu. Dast: Data-free substitute training for adversarial attacks. In *CVPR*, 2020.