Data-Free Hard-Label Robustness Stealing Attack

Xiaojian Yuan[1], Kejiang Chen*[1], Wen Huang[1], Jie Zhang[2], Weiming Zhang[1], Nenghai Yu[1],

[1]University of Science and Technology of China, [2]Nanyang Technological University,
xjyuan@mail.ustc.edu.cn, chenkj@ustc.edu.cn, hw2000@mail.ustc.edu.cn,
jie_zhang@ntu.edu.sg, zhangwm@ustc.edu.cn, ynh@ustc.edu.cn

Abstract

The popularity of Machine Learning as a Service (MLaaS) has led to increased concerns about Model Stealing Attacks (MSA), which aim to craft a clone model by querying MLaaS. Currently, most research on MSA assumes that MLaaS can provide soft labels and that the attacker has a proxy dataset with a similar distribution. However, this fails to encapsulate the more practical scenario where only hard labels are returned by MLaaS and the data distribution remains elusive. Furthermore, most existing work focuses solely on stealing the model accuracy, neglecting the model robustness, while robustness is essential in security-sensitive scenarios, e.g., face-scan payment. Notably, improving model robustness often necessitates the use of expensive techniques such as adversarial training, thereby further making stealing robustness a more lucrative prospect. In response to these identified gaps, we introduce a novel Data-Free Hard-Label Robustness Stealing (DFHL-RS) attack in this paper, which enables the stealing of both model accuracy and robustness by simply querying hard labels of the target model without the help of any natural data. Comprehensive experiments demonstrate the effectiveness of our method. The clone model achieves a clean accuracy of 77.86% and a robust accuracy of 39.51% against AutoAttack, which are only 4.71% and 8.40% lower than the target model on the CIFAR-10 dataset, significantly exceeding the baselines. Our code is available at: *https://github.com/LetheSec/DFHL-RS-Attack*.

1 Introduction

Machine learning as a service (MLaaS) has gained significant popularity due to its ease of deployment and cost-effectiveness, which provides users with pre-trained models and APIs. Unfortunately, MLaaS is susceptible to privacy attacks, with Model Stealing Attacks (MSA) being particularly harmful (Tramèr et al. 2016; Orekondy, Schiele, and Fritz 2019; Jagielski et al. 2020; Yuan et al. 2022; Wang et al. 2022), where an attacker can train a clone model by querying its public API, without accessing its parameters or training data. This attack not only poses a threat to intellectual property but also compromises the privacy of individuals whose data was used to train the original model. Moreover, the clone model can serve as a surrogate model for other black-box attacks, e.g., adversarial examples (AE) (Zhang et al. 2022b), membership inference (Shokri et al. 2017), and model inversion (Yuan et al. 2023).

Furthermore, numerous security-sensitive scenarios require deployed models are not only accurate but also robust to various attacks, such as adversarial attacks. To address this issue, MLaaS providers can employ adversarial training (AT) techniques (Madry et al. 2017) to improve the robustness of their models (Goodman and Xin 2020; Shafique et al. 2020). Despite the target model's robustness, most existing MSA are limited to Accuracy Stealing, i.e., reconstructing a model with similar accuracy to the target model, and fail at Robustness Stealing, i.e., acquiring the adversarial robustness of the target model while maintaining accuracy. Since the improvement of model robustness requires much more computational resources and extra data (Schmidt et al. 2018; Gowal et al. 2021), robustness stealing will bring greater losses to MLaaS providers. Moreover, if an attacker seeks to train a clone model for transfer-based adversarial attacks against a robust target model, then it becomes crucial to employ robustness stealing to achieve effective attack performance (Dong et al. 2020; Gao et al. 2020).

In addition, most previous MSA require MLaaS to provide prediction logits, i.e., soft labels. However, this requirement is overly stringent in typical scenarios where MLaaS can only return top-1 prediction, i.e., hard label, for each query. Since models that require robustness are more likely trained on sensitive or private datasets, it is difficult for attackers to obtain public data with similar distributions, let alone access to the original data. Hence, the investigation of MSA targeting robustness in a data-free hard-label setting is highly valuable and remains unexplored.

To tackle the above issues, we propose Data-Free Hard-Label Robustness Stealing (DFHL-RS) attack, which can effectively steal both the accuracy and robustness of the target model. We first demonstrate that direct use of AT during MSA is suboptimal and point out the limitations of using Uncertain Example (UE) (Li et al. 2023a) for robustness stealing. Then the concept of High-Entropy Example (HEE) is introduced, which can characterize a more complete classification boundary shape of the target model. By imitating the target model's prediction of HEE, the clone model gradually approaches its classification boundaries, so as to achieve prediction consistency for various samples. More-

*Corresponding author.

over, to eliminate the reliance on natural data and the logits of the target model, we design a data-free robustness stealing framework in a hard-label setting. Specifically, we first train a generator to synthesize substitute data for approximating the distribution of the target data. Since only hard labels are available, we cannot use the target model for gradient back-propagation (Chen et al. 2019; Zhang et al. 2022a) or gradient estimation (Truong et al. 2021; Kariyappa, Prakash, and Qureshi 2021). Thus, we use the clone model as a surrogate to guide the direction of the synthesized images. To prevent the generator from overfitting to the clone model, we adopt label smoothing and data augmentation techniques. Then we sample multiple batches from the memory bank storing synthesized images and use the proposed algorithm to construct HEE. Finally, we employ HEE to query the target model and obtain pseudo-labels for training the clone model.

Our contributions can be summarized as follows:

- For the first time, we explore a novel attack namely Data-Free Hard-Label Robustness Stealing (DFHL-RS) to achieve both accuracy and robustness stealing by leveraging only hard labels without any natural data.
- We propose the concept of High-Entropy Examples (HEE), which can better characterize the complete shape of the classification boundary.
- Extensive experiments demonstrate the effectiveness and stability of our proposed attack framework under various configurations.

2 Related Work

Data-Free Knowledge Distillation. Knowledge distillation (Hinton, Vinyals, and Dean 2015) aims to transfer the knowledge of a large teacher model to a smaller student model. In some cases, it is not feasible to access the training data due to storage costs or privacy concerns. Therefore, some proposed distillation techniques utilizing proxy datasets with similar distributions (Lopes, Fenu, and Starner 2017; Addepalli et al. 2020). ZSKD (Nayak et al. 2019) first proposed Data-Free Knowledge Distillation (DFKD), which uses the teacher's predictions to optimize synthetic data. DAFL (Chen et al. 2019) introduced the generator for synthesizing query samples, and proposed several generative losses to promote the diversity. Adversarial DFKD (Micaelli and Storkey 2019) utilized adversarial learning to explore the data space more efficiently. Some follow-up work attempted to mitigate the catastrophic overfitting (Binici et al. 2022b,a), mode collapse (Fang et al. 2021) in DFKD, and to speed up the training process (Fang et al. 2022). However, all of these methods necessitate white-box access to the teacher. ZSDB3KD (Wang 2021) proposed DFKD in black-box scenarios, but it has high computational costs and requires a large number of queries (4000 million).

Data-Free Model Stealing. The main difference between Data-Free Model Stealing (DFMS) and DFKD is that it only has black-box access to the teacher model, i.e., the target model. Some early work required the use of a proxy dataset for attacks (Orekondy, Schiele, and Fritz 2019; Barbalau et al. 2020; Wang et al. 2022). Based on Adversarial DFKD, recent works MAZE (Kariyappa, Prakash, and Qureshi 2021) and DFME (Truong et al. 2021) utilized gradient estimation techniques to achieve DFMS, which require the target model to return soft labels. Therefore, DFMS-HL (Sanyal, Addepalli, and Babu 2022) extended the problem to the hard-label setting. However, it still needs to use a proxy dataset or a synthetic dataset of random shapes generated on colored backgrounds, which breaks the truly data-free setting. To address this issue, DS (Beetham et al. 2023) proposed to train two student models simultaneously, which allows the generator to use one of the students as a proxy for the target model. However, these methods only achieved accuracy stealing, but cannot obtain the model's robustness.

Adversarial Robustness Distillation. Large models tend to be more robust than small models due to their greater capacity. Therefore, Goldblum et al. (Goldblum et al. 2020) first proposed Adversarial Robustness Distillation (ARD). By distilling the robustness of the teacher, the student obtained higher robustness than AT from scratch. RSLAD (Zi et al. 2021) found that using pseudo-labels provided by a robust teacher can further improve the robustness. IAD (Zhu et al. 2022) found that the guidance from the teacher model is progressively unreliable and proposed a multi-stage strategy to address this issue. However, these methods require access to the training set and the parameters of the teacher. BEST (Li et al. 2023a) proposed to steal the robustness of the target model in the black-box setting, but it necessitated a proxy dataset. DFARD (Wang et al. 2023) proposed ARD in a data-free setting, yet still required white-box access.

3 How To Steal Robustness?

In this section, for a fair comparison, we first assume that the attacker can obtain a proxy dataset for attacks as (Li et al. 2023a). Given a target model M_T for a classification task that is built via AT and exhibits certain adversarial robustness, our goal is to train a clone model M_C that performs similarly to M_T on both clean and adversarial examples. The attacker has no prior knowledge of M_T, including the architecture, parameters, and training strategies, and is only granted black-box access to M_T. We consider the typical MLaaS scenario, where M_T only returns top-1 predicted labels, i.e., hard-label setting. Most MSA methods usually use proxy data as query samples to query M_T, with the purpose of merely stealing accuracy. Recently, there have been new attempts of robustness stealing (Li et al. 2023a). Here, we provide a comprehensive summary and analysis.

3.1 Adversarial Training (AT)

A straightforward way to steal robustness is to conduct AT during the attack. Specifically, the attacker first queries M_T with query samples x_q and obtains top-1 predictions y_q, and then conducts standard AT on M_C with (x_q, y_q) as follows:

$$\min_{\theta_{M_C}} \mathbb{E}_{(x_q,y_q)\in\mathcal{D}_p}(\arg\max_{\delta} \mathcal{L}_{\text{CE}}(\theta_{M_C}, x_q+\delta, y_q)), \quad (1)$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss function, θ_{M_C} represents parameters of the clone model, and $x_q + \delta$ represents the adversarial examples generated using PGD (Madry et al. 2017). However, this method will also inherit the shortcomings of AT, which will seriously reduce the clean accuracy.

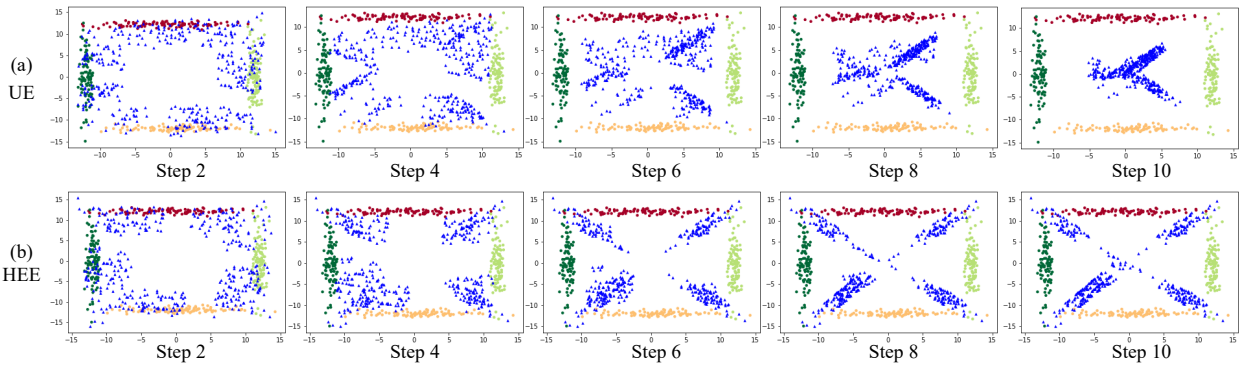

Figure 1: Illustration of the superiority of HEE over UE in characterizing the classification boundaries. We make a two-dimensional dataset with four classes to train a two-layer MLP, represented by four colors. Blue points in (a) and (b) represent UE and HEE constructed at different steps according to Eq. (2) and Eq. (3), respectively.

Cloen Model	Method	Clean Acc	Robust Acc	Avg.
ResNet18	AT	34.22	**22.58**	28.40
	UE	43.98	15.78	29.88
	AE	34.76	20.32	27.54
	HEE	**49.38**	19.60	**34.49**
MobileNet	AT	34.42	**22.64**	28.53
	UE	48.84	14.86	31.85
	AE	35.46	21.00	28.23
	HEE	**50.12**	18.98	**34.55**

Table 1: Quantitative comparison of attacks using different query samples. M_T is ResNet18 trained on CIFAR-10 training set, M_C is ResNet18 or MobileNetV2, the proxy dataset is a random half of CIFAR-100 test set. Clean Acc and Robust Acc represent the accuracy (%) of clean samples and adversarial samples generated by PGD, respectively.

	Clean Acc	Robust Acc	Avg.
HEE	**50.12**	**18.98**	**34.55**
HEE **w/o** $\mathcal{H}(\cdot)$	48.26	18.26	33.26
HEE **w/** l_∞-norm	48.86	15.90	32.38

Table 2: Ablation study about HEE. "w/o $\mathcal{H}(\cdot)$" represents the same objective as UE. "w/ l_∞-norm" indicates that l_∞-norm constraints are used in the process of constructing HEE like UE. M_C is MobileNetV2.

3.2 Uncertain Examples (UE)

In addition, Li et al.introduced Uncertain Examples (UE) to achieve robustness stealing. The main idea is to find samples that the model predicts with the highest uncertainty across all classes and use them to query M_T. The construction process of UE is similar to that of AE, i.e., iteratively adds small noise to the query sample. Specifically, first assign the same target $Y = [1/K, \ldots, 1/K]$ (K is the number of classes) to each query sample, then use the Kullback–Leibler (KL) divergence to compute and minimize the distance between the prediction of M_T and Y. Given a starting point x^0_{UE} which is a random neighbor of the original query sample, an iterative update is performed with:

$$x^{t+1}_{\text{UE}} = \Pi_{\mathcal{B}_\epsilon[x_{\text{UE}}]}(x^t_{\text{UE}} - \alpha \cdot \text{sign}(\nabla_{x^t_{\text{UE}}} \mathcal{L}_{\text{KL}}(M_C(x^t_{\text{UE}}) \| Y))), \quad (2)$$

where $\mathcal{L}_{\text{KL}}(\cdot\|\cdot)$ is the KL divergence, $\nabla_{x^{(t)}_{\text{UE}}}$ denotes the gradient of the loss function w.r.t. the uncertain example x^t_{UE} in step t, α denotes step size, and $\Pi_{\mathcal{B}_\epsilon[x_{\text{UE}}]}(\cdot)$ projects its input onto the ϵ-bounded neighborhood of the original one.

However, although this method alleviates the negative impact on clean accuracy, it reduces the robustness of M_C, as shown in Table 1. We analyze the limitations of UE:

- First, using the manually defined Y as the target in Eq. (2) is too hard for optimization, which will make UE more inclined to be located at the junction of the classification boundaries of all classes. The iterative process in Fig. 1(a) confirms this conjecture.
- Additionally, UE, like AE, uses the l_∞-norm to constrain the magnitude of added noise with the aim of improving the visual quality of samples. However, this will sacrifice attack performance, as shown in Table 2. Actually, the constraint is not necessary in MSA, because the attacker typically uses abnormal samples or even unnatural synthetic samples in the data-free setting

3.3 High-Entropy Examples (HEE)

We hope to construct samples that can characterize a more complete shape of the classification boundary, not just the junction as UE. Intuitively, samples located near the classification boundaries usually have larger prediction entropy, and the model will give similar confidence in multiple classes. Therefore, we propose the concept of High-Entropy Examples (HEE), which can be constructed by directly maximizing the prediction entropy as follows:

$$x^{t+1}_{\text{HEE}} = x^t_{\text{HEE}} + \alpha \cdot \text{sign}(\nabla_{x^t_{\text{HEE}}} \mathcal{H}(M_C(x^t_{\text{HEE}}))), \quad (3)$$

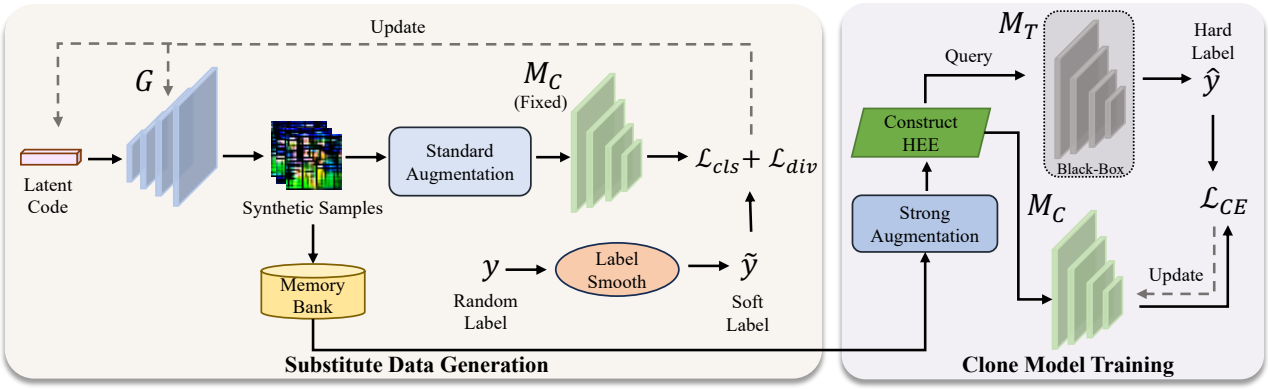

Figure 2: The pipeline of our DFHL-RS attack, which consists of two alternately executed stages in each epoch. In the first stage, we optimize latent code and generators to generate substitute data and store it in the memory bank. In the second stage, we sample multiple batches from the memory bank and construct HEE to query the target model for hard labels, then use them to update the parameters of the clone model.

where $\mathcal{H}(\cdot)$ calculates the entropy of the prediction, $\nabla_{x_{\text{HEE}}^{(t)}}$ denotes the gradient of the entropy loss function w.r.t. the high-entropy example x_{HEE}^t in step t and α is the step size. Note that we no longer use the l_∞-norm to constrain the modification of the sample during iterations.

Compared to manually assigning the same confidence value to all classes in UE, Eq. (3) provides an adaptive optimization objective, allowing samples to explore among several similar classes rather than all classes. It can be seen from Fig. 1(b) that HEE can gradually distribute near the classification boundaries over steps. Therefore, keeping the predictions of M_C on these samples consistent with M_T enables it to learn the shape of the classification boundary.

3.4 Adversarial Examples (AE)

Previous work (He, Li, and Song 2018; Li et al. 2023a) noticed that AE is also close to the classification boundaries, which may achieve the similar effect of UE and HEE. For robustness stealing, we can first obtain pseudo-labels of proxy data by querying M_T and use the standard PGD to construct AE. Then we use AE to query M_T again for corresponding labels to train M_C.

3.5 Comparison of Different Query Samples

For quantitative evaluation, we follow the same setting as (Li et al. 2023a). As shown in Table 1, although AT and AE can make M_T obtain higher robustness, they will greatly reduce the clean accuracy. While UE can improve the clean accuracy, the robustness will be significantly reduced. Our proposed HEE achieves the best balance between clean accuracy and robustness. In Table 2, we also conduct an ablation study on different components of HEE.

To illustrate the superiority of HEE over UE in characterizing classification boundaries, we make a two-dimensional dataset with four classes to train a MLP (see appendix for details), as shown in Fig. 1. As the iteration steps increase during the construction process, UE will gradually concentrate at the junction of classification boundaries, while HEE will be uniformly distributed around the classification boundaries, thus characterizing its more complete shape. This is in line with our previous analysis. Therefore, querying with HEE allows M_C to better approximate the classification boundaries of M_T.

4 Data-Free Hard-Label Robustness Stealing

In this section, we further explore the challenging data-free scenario, where the attacker CAN NOT obtain any similar natural samples as proxy data. The framework is illustrated in Fig. 2 and the training algorithm is in the appendix.

4.1 Overview

Unlike previous work (Sanyal, Addepalli, and Babu 2022; Beetham et al. 2023), which uses M_T as a discriminator and plays a min-max game, we decouple the training process of the generator and the training process of M_C into two stages: 1) Substitute Data Generation and 2) Clone Model Training. In the first stage, we train a generator to synthesize substitute data to approximate the distribution of the target data and store them in a memory bank. Due to the hard-label setting, we use M_C to guide the training of the generator, rather than M_T as in previous work (Kariyappa, Prakash, and Qureshi 2021; Truong et al. 2021). In the second stage, we randomly sample multiple batches of substitute data from the memory bank and use Eq. (3) to construct HEE, then use HEE to query M_T for hard labels to optimize the parameters of M_C. These two stages are executed alternately in each epoch.

4.2 Substitute Data Generation

We adopt the "batch-by-batch" manner to synthesize substitute data, that is, only one batch of images will be synthesized by a new generator in each epoch. Specifically, at the beginning of an arbitrary epoch i, we resample a batch of latent code $z_i \sim \mathcal{N}(0, 1)$ and reinitialize the generator G_i with parameters from the last epoch. Then we randomly sample a batch of corresponding labels $\tilde{y}_i$ from the uniform distribution. Intuitively, we hope that the images synthesized by the

generator can be classified into the specified class by M_T. This means that the distribution of the synthetic data is similar to the target data. For this, we can iteratively optimize the latent code z_i as well as the parameters θ_{G_i} as follows:

$$\mathcal{L}_{\text{cls}} = \underset{z_i, \theta_{G_i}}{\arg\min}\, \mathcal{L}_{\text{CE}}(M_T(G_i(z_i; \theta_{G_i}), \tilde{y}_i)), \tag{4}$$

However, the backpropagation of Eq. (4) will violate the principles of a black-box setting. The gradient estimation techniques (Truong et al. 2021; Kariyappa, Prakash, and Qureshi 2021) also cannot be applied due to the unavailability of soft labels. To address this issue, we use the latest M_C as a surrogate for M_T, providing gradients for optimization. The optimization problem can be formulated as follows:

$$\mathcal{L}_{\text{cls}} = \underset{z_i, \theta_{G_i}}{\arg\min}\, \mathcal{L}_{\text{CE}}(M_{C_{i-1}}(G_i(z_i; \theta_{G_i}), \tilde{y}_i)). \tag{5}$$

To prevent overfitting of synthetic data to the clone model, we use a small number of iterations (see appendix for details). We perform standard data augmentation on synthetic samples to ensure that they are not deceptive (e.g., adversarial example). In addition, we also adopt label smoothing (Szegedy et al. 2016) to soften the corresponding labels to further alleviate overfitting.

Moreover, merely ensuring the generator produces images from the desired distribution is inadequate, as it may still encounter issues such as mode collapse and insufficient diversity (Chen et al. 2019; Sanyal, Addepalli, and Babu 2022). In a hard-label setting, a lack of diversity among classes can significantly hamper the learning process of M_C, particularly for the less represented classes. Therefore, to promote the generation of diverse images across all classes in each batch, we adopt a class-diversity loss as follows:

$$\mathcal{L}_{\text{div}} = \sum_{j=0}^{K} \alpha_j \log \alpha_j, \alpha_j = \frac{1}{N} \sum_{i=1}^{N} \text{softmax}(M_C\,(x_i))_j, \tag{6}$$

where K denotes the number of classes, α_j denotes the expected confidence value for every class j over a batch with N samples and the $\mathcal{L}_{\text{div}}$ calculates the negative entropy. When the loss is minimized, the entropy of the number of synthetic samples per class gets maximized such that each class has a similar amount of samples.

By combining the aforementioned two loss functions, we obtain the final objective:

$$\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{cls}} + \lambda \cdot \mathcal{L}_{\text{div}}, \tag{7}$$

where λ is a hyperparameter for balancing two different terms. Note that we does not require training a global generator to model the entire distribution of the target data. Instead, in each epoch, only a temporary generator is trained to handle a specific data distribution for one batch. Besides, we keep the parameters of M_C fixed in this stage. After optimization by Eq. (7), we store this batch of synthetic data into the memory bank and discard the generator.

4.3 Clone Model Training

In the second stage, we train M_C with synthetic substitute data. However, if we optimize M_C using only one batch of samples synthesized in the first stage of the current epoch, it will suffer from catastrophic forgetting (Binici et al. 2022b,a; Do et al. 2022). The reason is that the substitute data is synthesized under the guidance of M_C, as shown in Eq. (5). With each epoch, the gap between M_C and M_T decreases, causing a shift in the distribution of synthetic data over time. Hence, if M_C does not periodically relearn previously synthesized samples, the knowledge acquired during the early training phase may be lost. This can result in performance degradation or even failure to converge over time.

To address this issue, we use a memory bank to store all previously synthesized samples in the first stage. In each epoch, we optimize M_C for N_C steps. In each step, we randomly select a batch of synthetic samples from the memory bank. Then we use Eq. (3) to construct HEE x_{HEE}, but before that, we need to perform strong augmentation (see appendix for details) on x to promote the diversity . This is beneficial to characterize more complete classification boundaries. Finally, we feed x_{HEE} into M_T to query the hard label $\hat{y}$ and use cross-entropy loss to optimize M_C as follows:

$$\mathcal{L}_C = \mathcal{L}_{\text{CE}}(M_C(x_{\text{HEE}}), \hat{y}). \tag{8}$$

By minimizing $\mathcal{L}_C$, M_C can obtain similar classification boundaries as M_T by imitating its predictions for x_{HEE}, thus simultaneously stealing the accuracy and robustness.

5 Experiments

In this section, we first provide the detailed experimental settings. To demonstrate the effectiveness of our method, we evaluate it from several perspectives.

5.1 Experimental Settings

Datasets. We consider two benchmark datasets commonly used in AT research (Madry et al. 2017; Zhang et al. 2019; Li et al. 2023b), CIFAR-10 and CIFAR-100 (Krizhevsky, Hinton et al. 2009), as target datasets. Prior work requires a proxy dataset with samples from the same distribution (Tramèr et al. 2016; Jagielski et al. 2020; Orekondy, Schiele, and Fritz 2019) or the same task domain (Sanyal, Addepalli, and Babu 2022; Li et al. 2023a) as the target dataset. However, we avoid using any natural data.

Models. We evaluate our attack on different models with various architectures. M_T is selected from ResNet18 (He et al. 2016) and WideResNet-34-10 (Zagoruyko and Komodakis 2016). M_C may be different from M_T in architecture, so we use two additional models, i.e., ResNet34 and MobileNetV2 (Sandler et al. 2018). We employ two commonly used AT strategies, namely PGD-AT (Madry et al. 2017) and TRADES (Zhang et al. 2019), and a state-of-the-art method, STAT-AWP (Li et al. 2023b), to improve the robustness of M_T. We follow the same generator architecture as (Fang et al. 2021, 2022).

Baselines. This is the first work to achieve data-free hard-label robustness stealing attack. The closest work to ours is *BEST* (Li et al. 2023a), but it requires a proxy dataset to attack. Therefore, we also make some modifications to the second stage of our framework as baselines: (1) *Data-Free*

Target Data	Data-Free	Method	Clean Acc	FGSM	PGD-20	PGD-100	CW-100	AA
CIFAR10	/	Target Model	82.57	56.99	51.31	50.92	49.68	47.91
	×	BEST (Li et al. 2023a)	67.31	32.68	27.48	27.27	28.33	28.33
	✓	Data-Free AT	36.15	15.86	11.87	11.73	12.03	11.43
		Data-Free AE	67.78	32.20	28.20	28.07	28.50	27.89
		Data-Free UE	74.24	39.78	35.00	34.80	35.28	34.41
		DFHL-RS(Ours)	**77.86**	**44.94**	**40.07**	**39.87**	**40.64**	**39.51**
CIFAR100	/	Target Model	56.76	31.96	28.96	28.83	26.84	26.84
	×	BEST (Li et al. 2023a)	28.33	18.78	18.78	15.08	16.28	14.59
	✓	Data-Free AT	20.22	9.79	8.63	8.62	8.73	8.74
		Data-Free AE	37.76	15.41	13.15	13.00	13.90	12.77
		Data-Free UE	39.25	16.88	14.18	14.06	14.94	14.94
		DFHL-RS(Ours)	**51.94**	**23.68**	**20.02**	**19.88**	**20.91**	**19.30**

Table 3: Attack performance comparison between different methods. M_T and M_C are ResNet18. The AT strategy is PGD-AT. We report the clean accuracy and robust accuracy (%) of M_C.

Target Model	AT Strategy	Clean Acc	PGD-100	AA
ResNet18	PGD-AT	**77.86**	**39.87**	**39.51**
	TRADES	72.15	37.24	37.09
	STAT-AWP	72.24	37.79	37.73
WideResNet	PGD-AT	**77.75**	36.65	36.55
	TRADES	71.04	34.20	33.94
	STAT-AWP	73.81	**38.16**	**38.13**

Table 4: Attack performance under different architectures of M_T and various AT strategies. M_C is ResNet18.

AT: use synthetic samples to query M_T for corresponding labels and then perform AT on M_C. (2) *Data-Free UE*: construct uncertain examples with synthetic samples to query M_T. (3) *Data-Free AE*: construct adversarial examples with synthetic samples to query M_T.

Metrics. In this task, M_C should not only have good usability, but also need to have certain robustness. Therefore, in addition to considering clean accuracy, i.e., the accuracy over clean samples, we also measure robust accuracy against various adversarial attacks, including FGSM (Goodfellow, Shlens, and Szegedy 2014), PGD-20, PGD-100 (Madry et al. 2017), CW-100 (Carlini and Wagner 2017) and AutoAttack (AA) (Croce and Hein 2020). The attack settings are $\epsilon = 8/255$ and $\eta = 2/255$. The number of attack step is 20 for PGD-20, and 100 for PGD-100 and CW-100.

Implementation Details. For substitute data generation, we use Adam optimizer with $\beta = (0.5, 0.999)$ and set the hyperparameter $\lambda = 3$ in Eq. (7). For CIFAR-10, we set learning rates $\eta_G = 0.002, \eta_z = 0.01$, number of iterations $N_G = 10$ and the label smoothing factor is set to 0.2. For CIFAR-100, we set learning rates $\eta_G = 0.005, \eta_z = 0.015$, number of iterations $N_G = 15$ and the label smoothing factor is set to 0.02. For training M_C, we use SGD optimizer with an initial learning rate of 0.1, a momentum of 0.9 and a weight decay of $1e{-}4$. For constructing HEE, the step size α in Eq. (3) is set to 0.03 and the number of iterations is set to 10. We set the iterations of the clone model $N_C = 500$. The batch sizes for CIFAR-10 and CIFAR100 are set to $B = 256$ and $B = 512$, respectively. We apply a cosine decay learning rate schedule and the training epoch is $E = 300$.

Clone Model	Clean Acc	PGD-100	AA
MobileNet	73.50	33.32	33.19
ResNet34	**78.49**	40.25	39.97
WideResNet	77.38	**40.66**	**40.33**

Table 5: Attack performance using different architectures of M_C. M_T is ResNet18 using PGD-AT strategy.

5.2 Experimental Results

Performance on Robustness Stealing. We first compare the attack effects of different methods by evaluating the clone model, as shown in Table 3. Our DFHL-RS significantly outperforms the baselines in both accuracy and robustness. When the target data is CIFAR-10, our method achieves 77.86% clean accuracy, which is only 4.71% lower than the target model. Meanwhile, it also achieves 39.51% robust accuracy against AA, which is only 8.40% lower than the target model. When the target data is CIFAR100, our method achieves 51.94% clean accuracy and 19.30% robust accuracy against AA, which are only 4.82% and 7.53% lower than the target model, respectively. It should be emphasized that our method does not require any natural data, but still outperforms BEST which requires a proxy dataset.

Different Model Architectures and AT Strategies. In real scenarios, MLaaS providers may use different architectures and strategies to improve the robustness of the target model, so we study the impact of different target model architectures using various AT strategies on the attack. See appendix for the performance of all target models. As shown in Table 4, when the target model is ResNet18, PGD-AT is more vulnerable to robustness stealing attacks. Although the target model using PGD-AT has lower robustness than using STAT-AWP, the clone model obtained by the attack has

Epsilon(ϵ)	Attack	*White-box*	*AT Proxy*	Data-Free Model Stealing Method			
				MAZE	DFME	DFMS-HL	**DFHL-RS(Ours)**
$4/255$	FGSM	*12.49*	*9.84*	4.36	3.90	6.09	**10.29**
	PGD-20	*13.54*	*10.27*	4.31	3.71	6.11	**10.54**
$8/255$	FGSM	*25.58*	*21.44*	10.51	9.25	13.95	**22.77**
	PGD-20	*31.25*	*24.33*	10.13	9.06	14.52	**24.59**
$12/255$	FGSM	*36.99*	*31.94*	17.42	15.57	22.07	**34.30**
	PGD-20	*49.02*	*40.44*	71.73	15.80	23.93	**39.25**

Table 6: Attack Success Rate (ASR) (%) of transfer-based adversarial attack using the clone model obtained by different methods as a surrogate model. AT Proxy represents the adversarial training model on the target data. The target model is trained on CIFAR-10 dataset using PGD-AT with $\epsilon = 8/255$.

Modification		Query Budget	Clean Acc	AA
Default		38.4M	77.86	39.51
B	$256 \rightarrow 128$	19.2M	**75.79**	**36.42**
E	$300 \rightarrow 150$		73.81	34.73
N_C	$500 \rightarrow 250$		74.07	35.19

Table 7: Attack performance after modifying hyperparameters to halve query budget.

	Clean Acc	AA
DFHL-RS	**77.86**	**39.51**
w/o Div Loss	76.87	38.87
w/o Label Smoothing	77.10	38.42
w/o Standard Augmentation	74.99	36.80
w/o Strong Augmentation	77.13	38.66

Table 8: Ablation study of different components.

the best performance. When the target model is WideResNet with a different architecture, STAT-AWP can make the clone model obtain the highest robustness, but PGD-AT still has the best clean accuracy. Consider the black-box setting, the attacker may use different clone model architectures for the attack. As shown in Table 5, close attack performance is achieved when the clone model is ResNet34 and WideResNet, while the attack performance drops slightly when the clone model is MobileNetv2, probably due to the larger difference in architecture. In conclusion, our method achieves stable attack performance across different configurations.

Transfer-Based Adversarial Attacks. We study the effectiveness of black-box transfer-based adversarial attacks on the target model using the clone model as a surrogate. As shown in Table 6, although black-box attacks on robust models are extremely challenging (Dong et al. 2019, 2020), our method significantly improves the attack success rate (ASR) at various ϵ compared to the baselines. In most cases, our method even slightly exceeds the AT Proxy, which directly uses the target data to train a surrogate model, and is slightly worse than the white-box attack. As ϵ decreases, the gap between our method and the white-box attack gets smaller.

Attack Cost of DFHL-RS. We also consider the attack cost of our DFHL-RS. In terms of query budget, our method eliminates the need for any querying during the data generation stage. It only requires querying the target model with each HEE sample for the corresponding label. Therefore, the total query budget depends on the training epoch, batch size and the clone model iterations, i.e., $E \times B \times N_C$. Our default query budget for CIFAR-10 is 38M, whereas it is 20M for DFME and DS, and 30M for MAZE. This is reasonable because acquiring robustness typically comes at a higher cost. By modifying these three hyperparameters, we can control the query budget as shown in Table 7. When the query budget is halved, the clean accuracy and robust accuracy are only reduced by approximately 2% and 3%, respectively. More results can be found in the appendix.

In terms of time overhead, since only a few iterations per epoch are sufficient for the first stage, it takes very little time. The time overhead of the second stage is mainly concentrated on the inner loop for constructing HEE, and its computational complexity is similar to that of standard AT. The time overhead mainly depends on the number of training samples per epoch, i.e., $B \times N_C$, which is comparable to many AT techniques.

Ablation Study. We further investigate the effects of the various components in our DFHL-RS framework as shown in Table 8. When the diversity loss in Eq. (7) is discarded, the clean accuracy and robust accuracy both decrease. In addition, removing the label smoothing and standard augmentation in the first stage will also affect the attack performance. The strong data augmentation in the second stage improves the attack effect by promoting the diversity of HEE.

6 Conclusion

In this paper, we first explore a novel and challenging task called Data-Free Hard-Label Robustness Stealing (DFHL-RS) attack, which can steal both clean accuracy and adversarial robustness by simply querying a target model for hard labels without the participation of any natural data. Experiments show that our method achieves excellent and stable attack performance under various configurations. Our work aims to advance research to improve security and privacy by raising awareness of the vulnerabilities of machine learning models through attacks.

Acknowledgements

This work was supported in part by the Natural Science Foundation of China under Grant U20B2047, 62102386, U2336206, 62072421, 62002334 and 62121002, and by Xiaomi Young Scholars Program.

References

Addepalli, S.; Nayak, G. K.; Chakraborty, A.; and Radhakrishnan, V. B. 2020. Degan: Data-enriching gan for retrieving representative samples from a trained classifier. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04): 3130–3137.

Barbalau, A.; Cosma, A.; Ionescu, R. T.; and Popescu, M. 2020. Black-Box Ripper: Copying black-box models using generative evolutionary algorithms. *Advances in Neural Information Processing Systems*, 33: 20120–20129.

Beetham, J.; Kardan, N.; Mian, A. S.; and Shah, M. 2023. Dual Student Networks for Data-Free Model Stealing. In *International Conference on Learning Representations*.

Binici, K.; Aggarwal, S.; Pham, N. T.; Leman, K.; and Mitra, T. 2022a. Robust and resource-efficient data-free knowledge distillation by generative pseudo replay. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6): 6089–6096.

Binici, K.; Pham, N. T.; Mitra, T.; and Leman, K. 2022b. Preventing catastrophic forgetting and distribution mismatch in knowledge distillation via synthetic data. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 663–671.

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57. Ieee.

Chen, H.; Wang, Y.; Xu, C.; Yang, Z.; Liu, C.; Shi, B.; Xu, C.; Xu, C.; and Tian, Q. 2019. Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3514–3522.

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, 2206–2216. PMLR.

Do, K.; Le, T. H.; Nguyen, D.; Nguyen, D.; Harikumar, H.; Tran, T.; Rana, S.; and Venkatesh, S. 2022. Momentum Adversarial Distillation: Handling Large Distribution Shifts in Data-Free Knowledge Distillation. *Advances in Neural Information Processing Systems*, 35: 10055–10067.

Dong, Y.; Fu, Q.-A.; Yang, X.; Pang, T.; Su, H.; Xiao, Z.; and Zhu, J. 2020. Benchmarking Adversarial Robustness on Image Classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Dong, Y.; Pang, T.; Su, H.; and Zhu, J. 2019. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4312–4321.

Fang, G.; Mo, K.; Wang, X.; Song, J.; Bei, S.; Zhang, H.; and Song, M. 2022. Up to 100x faster data-free knowledge distillation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6): 6597–6604.

Fang, G.; Song, J.; Wang, X.; Shen, C.; Wang, X.; and Song, M. 2021. Contrastive model inversion for data-free knowledge distillation. *arXiv preprint arXiv:2105.08584*.

Gao, L.; Zhang, Q.; Song, J.; Liu, X.; and Shen, H. T. 2020. Patch-wise attack for fooling deep neural network. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, 307–322. Springer.

Goldblum, M.; Fowl, L.; Feizi, S.; and Goldstein, T. 2020. Adversarially robust distillation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04): 3996–4003.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Goodman, D.; and Xin, H. 2020. Attacking and defending machine learning applications of public cloud. *arXiv preprint arXiv:2008.02076*.

Gowal, S.; Rebuffi, S.-A.; Wiles, O.; Stimberg, F.; Calian, D. A.; and Mann, T. A. 2021. Improving robustness using generated data. *Advances in Neural Information Processing Systems*, 34: 4218–4233.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 770–778.

He, W.; Li, B.; and Song, D. 2018. Decision boundary analysis of adversarial examples. In *International Conference on Learning Representations*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jagielski, M.; Carlini, N.; Berthelot, D.; Kurakin, A.; and Papernot, N. 2020. High accuracy and high fidelity extraction of neural networks. In *USENIX Security Symposium*, 1345–1362.

Kariyappa, S.; Prakash, A.; and Qureshi, M. K. 2021. Maze: Data-free model stealing attack using zeroth-order gradient estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13814–13823.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. *Tech Report*.

Li, G.; Xu, G.; Guo, S.; Qiu, H.; Li, J.; and Zhang, T. 2023a. Extracting Robust Models with Uncertain Examples. In *The Eleventh International Conference on Learning Representations*.

Li, Q.; Guo, Y.; Zuo, W.; and Chen, H. 2023b. Squeeze Training for Adversarial Robustness. In *International Conference on Learning Representations*.

Lopes, R. G.; Fenu, S.; and Starner, T. 2017. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Micaelli, P.; and Storkey, A. J. 2019. Zero-shot knowledge transfer via adversarial belief matching. *Advances in Neural Information Processing Systems*, 32.

Nayak, G. K.; Mopuri, K. R.; Shaj, V.; Radhakrishnan, V. B.; and Chakraborty, A. 2019. Zero-shot knowledge distillation in deep networks. In *International Conference on Machine Learning*, 4743–4751. PMLR.

Orekondy, T.; Schiele, B.; and Fritz, M. 2019. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4954–4963.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4510–4520.

Sanyal, S.; Addepalli, S.; and Babu, R. V. 2022. Towards data-free model stealing in a hard label setting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15284–15293.

Schmidt, L.; Santurkar, S.; Tsipras, D.; Talwar, K.; and Madry, A. 2018. Adversarially robust generalization requires more data. *Advances in Neural Information Processing Systems*, 31.

Shafique, M.; Naseer, M.; Theocharides, T.; Kyrkou, C.; Mutlu, O.; Orosa, L.; and Choi, J. 2020. Robust machine learning systems: Challenges, current trends, perspectives, and the road ahead. *IEEE Design & Test*, 37(2): 30–57.

Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, 3–18. IEEE.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2818–2826.

Tramèr, F.; Zhang, F.; Juels, A.; Reiter, M. K.; and Ristenpart, T. 2016. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security Symposium*, volume 16, 601–618.

Truong, J.-B.; Maini, P.; Walls, R. J.; and Papernot, N. 2021. Data-free model extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4771–4780.

Wang, Y.; Chen, Z.; Yang, D.; Guo, P.; Jiang, K.; Zhang, W.; and Qi, L. 2023. Model Robustness Meets Data Privacy: Adversarial Robustness Distillation without Original Data. *arXiv preprint arXiv:2303.11611*.

Wang, Y.; Li, J.; Liu, H.; Wang, Y.; Wu, Y.; Huang, F.; and Ji, R. 2022. Black-box dissector: Towards erasing-based hard-label model stealing attack. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*, 192–208. Springer.

Wang, Z. 2021. Zero-shot knowledge distillation from a decision-based black-box model. In *International Conference on Machine Learning*, 10675–10685. PMLR.

Yuan, X.; Chen, K.; Zhang, J.; Zhang, W.; Yu, N.; and Zhang, Y. 2023. Pseudo Label-Guided Model Inversion Attack via Conditional Generative Adversarial Network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(3): 3349–3357.

Yuan, X.; Ding, L.; Zhang, L.; Li, X.; and Wu, D. O. 2022. Es attack: Model stealing against deep neural networks without data hurdles. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(5): 1258–1270.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E.; El Ghaoui, L.; and Jordan, M. 2019. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, 7472–7482. PMLR.

Zhang, J.; Chen, C.; Dong, J.; Jia, R.; and Lyu, L. 2022a. QEKD: query-efficient and data-free knowledge distillation from black-box models. *arXiv preprint arXiv:2205.11158*.

Zhang, J.; Li, B.; Xu, J.; Wu, S.; Ding, S.; Zhang, L.; and Wu, C. 2022b. Towards Efficient Data Free Black-Box Adversarial Attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15115–15125.

Zhu, J.; Yao, J.; Han, B.; Zhang, J.; Liu, T.; Niu, G.; Zhou, J.; Xu, J.; and Yang, H. 2022. Reliable adversarial distillation with unreliable teachers. In *International Conference on Learning Representations*.

Zi, B.; Zhao, S.; Ma, X.; and Jiang, Y.-G. 2021. Revisiting adversarial robustness distillation: Robust soft labels make student better. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 16443–16452.

A Appendix

A.1 Details of Figure 1

To illustrate the superiority of HEE over UE in characterizing classification boundaries, we make a two-dimensional toy dataset comprising of four classes. Each class is generated by sampling data points from a two-dimensional Gaussian distribution. The means of the Gaussian distributions for the four classes are: (0, 12), (0, -12), (12, 0), (-12, 0). The corresponding standard deviations are: (5, 0.5), (5, 0.5), (0.5, 5), (0.5, 5). In addition, we use a MLP which consists of a linear layer with output size of 10 followed by a ReLU, and a final linear layer with output size of 4 for classification. We provide pytorch-style pseudocodes in Pseudocode 1 and Pseudocode 2. We train the MLP using the SGD optimizer with a learning rate of 0.02 for 100 epochs. For constructing UE and HEE, we discard the l_∞-norm constraint and use a step size of 1, the number of iterations is set from 1 to 10.

```
# n is the number of samples per class
def toy_dataset(n=100):
    x = torch.zeros(n * 4, 2)
    y = torch.zeros(n * 4, dtype=torch.long)
    means = [[0, 12],
             [0, -12],
             [12, 0],
             [-12, 0]]
    stds = [[5, 0.5],
            [5, 0.5],
            [0.5, 5],
            [0.5, 5]]

    for i, (mean, std) in
        enumerate(zip(means, stds)):
        x[i * n: (i + 1) * n] =
            torch.randn(n, 2) *
            torch.tensor(std) +
            torch.tensor(mean)
        y[i * n: (i + 1) * n] = i

    return x, y
```

Pseudocode 1: Make a toy dataset.

```
class MLP(torch.nn.Module):
    def __init__(self, n_features=2,
        n_hidden=10, n_output=4):
        super(MLP, self).__init__()
        self.hidden =
            torch.nn.Linear(n_features,
            n_hidden)
        self.predict =
            torch.nn.Linear(n_hidden,
            n_output)

    def forward(self, x):
        x = F.relu(self.hidden(x))
        x = self.predict(x)
        return x
```

Pseudocode 2: MLP Architecture.

A.2 Algorithm

The training process of our DFHL-RS framework is demonstrated in Algorithm. 1.

Algorithm 1: Training Process of DFHL-RS

Input: Target model M_T, generator $G(\cdot;\theta_G)$, epochs E, generator iterations N_G in each epoch, clone model iterations N_C in each epoch, learning rate of generator η_G, learning rate of latent code η_z,learning rate of clone model η_C, memory bank M_B

Output: Clone model $M_C(\cdot;\theta_C)$

for $e = 1, \ldots, E$ **do**
  // Substitute Data Generation
  Initialize the generator G
  Sample a batch of latent code $z \sim \mathcal{N}(0, 1)$ and corresponding labels $\tilde{y}$
  for $i = 1, \ldots, N_G$ **do**
    Generate synthetic samples by $G(z;\theta_G)$
    Perform standard augmentation and label smoothing
    Compute $\mathcal{L}_{\text{gen}}$ by Eq. (7)
    Update $z \leftarrow \eta_z \nabla_z \mathcal{L}_{\text{gen}}$
    Update $\theta_G \leftarrow \eta_G \nabla_{\theta_G} \mathcal{L}_{\text{gen}}$
  end
  Save this batch of synthetic samples to M_B
  // Clone Model Training
  for $i = 1, \ldots, N_C$ **do**
    Randomly sample a batch from M_B and perform strong augmentation
    Construct HEE x_{hee} by Eq. (3)
    Query M_T with HEE for hard labels $\hat{y}$
    Compute $\mathcal{L}_C$ by Eq. (8)
    Update $\theta_C \leftarrow \eta_C \nabla_{\theta_C} \mathcal{L}_C$
  end
end

N_G	3	5	10	15	20
Clean Acc	76.98	76.10	**77.86**	75.39	76.81
PGD-20	38.01	38.11	**40.07**	36.86	38.02
AA	37.58	37.72	**39.51**	36.58	37.56

Table 9: Clean accuracy and robust accuracy (%) of the clone model using different N_G in the first stage.

A.3 The Impact of Different N_G

We modify the number of iterations N_G in the first stage, the results are shown in Table 9. We can observe that too large or too small N_G cannot obtain the optimal attack performance. We speculate that a small N_G leads to poor synthetic data due to insufficient optimization. However, a large N_G leads to overfitting, making the synthetic data unsuitable for future training. Therefore, based on empirical experiments, we set $N_G = 10$ for our attack.

Target Model	AT Strategy	Clean Acc	FGSM	PGD-20	PGD-100	CW-100	AA
ResNet18	PGD-AT	82.57	56.99	51.31	50.92	49.68	47.91
	TRADES	80.85	55.92	51.23	51.18	48.09	48.09
	STAT-AWP	83.03	60.21	56.37	56.22	52.61	52.19
WideResNet	PGD-AT	86.48	61.16	54.90	54.38	54.12	52.14
	TRADES	84.38	60.50	54.65	54.40	53.21	52.21
	STAT-AWP	86.41	64.83	60.18	60.04	56.63	56.63

Table 10: Clean accuracy and robust accuracy (%) of target models with different architectures and AT Strategies.

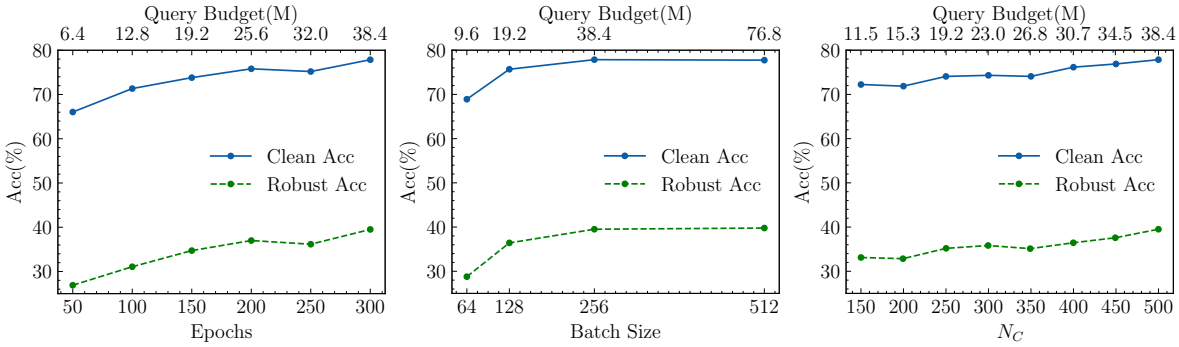

Figure 3: Ablation study of query budgets corresponding to different configurations. We report the clean accuracy and robust accuracy against AA.

Approach	Accuracy Extraction			Robustness Extraction	
	White-Box	Black-Box		White-Box	Black-Box
		Soft Label	Hard Label		
Data	Standard KD	KnockoffNets (Orekondy, Schiele, and Fritz 2019) Black-box dissector (Wang et al. 2022)	-	ARD (Goldblum et al. 2020) RSLAD (Zi et al. 2021)	BEST (Li et al. 2023a)
Data-Free	ZSKD (Nayak et al. 2019) DAFL (Chen et al. 2019)	MAZE (Kariyappa, Prakash, and Qureshi 2021) DFME (Truong et al. 2021)	DFMS-HL (Sanyal, Addepalli, and Babu 2022) DS (Beetham et al. 2023)	DFARD (Wang et al. 2023)	**DFHL-RS (Ours)**

Table 11: Taxonomy of prior works.

A.4 Performance of Target Models

In our experiments, we train several target models with different architectures and various adversarial training strategies on CIFAR-10 dataset. As shown in Table 10, we report the clean accuracy and robust accuracy of each target model.

A.5 More Ablation Results of Query Budget

As mentioned in the main manuscript, the query budget of our method depends on the training epoch E, batch size B and clone model iterations N_C, i.e., $E \times B \times N_C$. In the default configuration, the three hyperparameters are $E = 300, B = 256, N_C = 500$. By modifying them, we can control the corresponding query budget, as shown in Fig. 3. The reduction of epoch has a greater impact on the attack performance, while the attack performance is relatively stable when N_C is reduced. When the batch size is halved, there is only a slight decrease in attack performance, while doubling the batch size brings more query budget, but almost no improvement. In summary, our attack maintains stability and effectiveness, even with a limited query budget.

A.6 Taxonomy of Prior Work

We summarize existing work with different goals and varied levels of access to the target model as shown Table 11. Our DFHL-RS is the first to achieve robust extraction in the data-free and black-box (i.e., hard-label) setting.

A.7 Details of Data Augmentation

In the first stage, we use the standard data augmentation, i.e., random cropping and random horizontal flipping, on the synthetic samples. This is commonly used for standard training CIFAR-10 classification models. In the second stage, we apply a strong data augmentation, including central cropping, adding Gaussian noise, random horizontal or vertical flipping, and random rotation, on the samples before constructing HEE.