This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3261327

A Comprehensive Defense Framework against Model Extraction Attacks

Wenbo Jiang, *Student Member, IEEE,* Hongwei Li (Corresponding author), *Senior Member, IEEE,* Guowen Xu, *Member, IEEE,* Tianwei Zhang, *Member, IEEE,* and Rongxing Lu, *Fellow, IEEE*

Abstract—As a promising service, Machine Learning as a Service (MLaaS) provides personalized inference functions for clients through paid APIs. Nevertheless, it is vulnerable to model extraction attacks, in which an attacker can extract a functionally-equivalent model by repeatedly querying the APIs with crafted samples. While numerous works have been proposed to defend against model extraction attacks, existing efforts are accompanied by limitations and low comprehensiveness. In this paper, we propose AMAO, a comprehensive defense framework against model extraction attacks. Specifically, AMAO consists of four interlinked successive phases: adversarial training is first exploited to weaken the effectiveness of model extraction attacks. Then, malicious query detection is used to detect malicious queries and mark malicious users. After that, we develop a label-flipping poisoning attack to instruct the adaptive query responses to malicious users. Besides, the image pHash algorithm is employed to ensure the indistinguishability of the query responses. Finally, the perturbed results are served as a backdoor to verify the ownership of any suspicious model. Extensive experiments demonstrate that AMAO outperforms existing defenses in defending against model extraction attacks and is also robust against the adaptive adversary who is aware of the defense.

Index Terms—Machine-learning-as-a-service, Model extraction attacks, Deep learning.

1 Introduction

Deep learning has been investigated quite intensively in recent years and achieved very noticeable success in many application domains, e.g., speech recognition [1], image recognition [2], and autonomous vehicles [3]. To facilitate the deployment of machine learning services, many tech giants, such as Google [4], Amazon [5], Microsoft [6] and IBM [7], have launched their cloud-based Machine-Learning-as-a-Services (MLaaS). As shown in Fig 1, users can access machine learning prediction services through the published APIs and they are freed from the trouble of complicated local training. The well-trained deep learning models in MLaaS are considered valuable intellectual properties because of the substantial computing and storage resources for training a model.

Nevertheless, recent studies have demonstrated that well-trained deep learning models in MLaaS are vulnerable to model extraction attacks [8], [9], [10], [11], [12], [13], [14]. As depicted in Fig 2, a model extraction attack tries to reconstruct a substitute model with similar functionality as the original model (referred to as the victim model) via prediction APIs. Specifically, the attacker queries the victim model iteratively using carefully-crafted samples (or surrogate samples) to obtain the returned results (this is practical due to acceptable charges of commercial APIs). Then these query-output pairs are used as training data to reconstruct a substitute model. Generally, model extraction attacks have the following potential threats: (a) reconstructing the victim model [8], [10], [11]: the attacker may create a local copy of the victim model that replicates the performance of the victim model as closely as possible. This will undermine the intellectual property of MLaaS providers because the attacker does not have to pay any more for queries; (b) conducting an adversarial attack [15], [16], [17], [18]: through extracting the victim model, the attacker can obtain the parameters similar to the victim model and adversarial attacks can be facilitated significantly easily in this scenario; (c) launching privacy attacks such as the membership attack [19], model

Fig. 1: Machine-Learning-as-a-Service

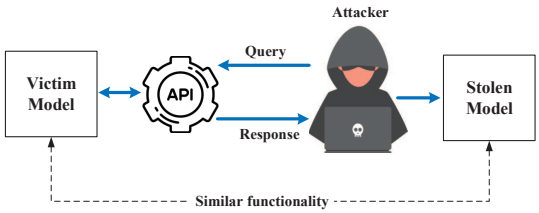


Fig. 2: Model extraction attack

- *W. Jiang, H. Li are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: wenbo_jiang@outlook.com, hongweili@uestc.edu.cn).*
- *G. Xu and T. Zhang are with School of Computer Science and Engineering, Nanyang Technological University, Singapore (e-mail: guowen.xu, tianwei.zhang@ntu.edu.sg).*
- *R. Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada E3B 5A3 (e-mail: RLU1@unb.ca).*

inversion attack [20] and property inference attack [21]: these privacy attacks are more effective with the availability of a white-box model (via model extraction) than a black-box one.

Recently, considerable works [10], [22], [23], [24], [25], [26], [27], [28] have been devoted to defending against model extraction attacks and they can be mainly divided into three categories: *behavior detection*, *prediction with perturbations* and *model watermarking*. *Behavior detection* [10], [22] mainly focuses on the query phase of the user. The model owner analyzes the behaviors (such as the distribution of queries) of all users to distinguish potential malicious users. *Prediction with perturbations* [23], [24], [25], [26] is exploited in the response phase. The model owner always returns the perturbed prediction results to users, which can significantly reduce the performance of the substitute model if it is trained with the perturbed results. *Model watermarking* [27], [28] is a post hoc defense against model extraction, which can verify the legitimacy of the stolen model.

However, existing efforts still have some limitations: in terms of *behavior detection*, an adversary may bypass the detection of [10] by involving multiple colluded malicious users or querying the victim model with surrogate samples (refer to Section 5.6.1 for more details); the overconfidence of classifiers may reduce the detection accuracy of the detection method proposed in [23] (refer to Section 4.3 for more details). We solve these limitations by employing the temperature-scaled maximum softmax probability to detect malicious queries. As for *prediction with perturbations*, prior work proposed to return malicious users with a random label [27], or add perturbations to all output predictions but ensure the predicted labels remain unchanged [25], or train a reverse model to output perturbed predictions for malicious users [23]. We propose an adaptive response strategy based on label-flipping poisoning attack, which can significantly improve the defensive effect. In terms of *model watermarking*, an adversary may know the existence of the model watermarking scheme (such as [27]) by querying the victim model with one sample and its transformed versions [12], [29], [30]. We address this problem by employing the image pHash (Perceptual Hash) algorithm to ensure the indistinguishability of the output results, which makes the defense more imperceptible to the adversary. Besides, none of the existing work considers the training phase of MLaaS for defense. We demonstrate in this work that employing adversarial training during the training phase can further decrease the attack effect of model extraction attacks.

To sum up, we develop AMAO, a comprehensive defense framework that has countermeasures against model extraction attacks for every phase of the model pipeline from training, prediction to release. AMAO consists of four successive phases: adversarial training, malicious query detection, adaptive query response and ownership verification. Specifically, adversarial training is utilized to weaken the effect of the attack, which forces an adversary to submit more malicious queries to achieve the desired attack goal. As a result, the adversary will be more easily detected by the subsequent detection phase. After that, an adaptive query response strategy is employed to instruct the victim model to reply to the malicious user with perturbed results. The perturbed results can not only mitigate the attack but also prepare for the subsequent ownership verification. Finally, the ownership can be verified through these perturbed sample-label pairs. Each phase of AMAO is designed with a competitive defense strategy, which enables the phases to promote each other and demonstrate the best overall defense capabilities. In short, our contributions can be elaborated in four aspects:

- We propose a comprehensive defense framework against model extraction attacks. In the defense framework, for the first time, we introduce the use of adversarial training to defend against model extraction attacks. Specifically, we analyze the reason why adversarial training can decrease the effect of model extraction attacks and demonstrate it through experiments.
- We point out the limitation of the detection methods proposed in [10] and [23], and employ the temperature-scaled maximum softmax probability as the metric to detect malicious queries. The detection method outperforms existing detection schemes and is also robust against the adaptive adversary who involves multiple colluded malicious users or employs surrogate samples as malicious queries.
- We propose an adaptive response strategy based on a label-flipping poisoning attack to perturb the results received by malicious users, thus reducing the effectiveness of model extraction attacks. Besides, the image pHash (Perceptual Hash Algorithm) is employed to ensure the indistinguishability of the output results.
- We conduct extensive experiments to evaluate the effectiveness of AMAO. The experimental results demonstrate the superiority of AMAO over state-of-the-art defenses against model extraction attacks including *JBDA-TR* [10], *Cloudleak* [12] and *KnockoffNet* [11]. Besides, experimental results show that AMAO is also robust against a variety of adaptive adversary scenarios.

The rest of this paper proceeds as follows: the preliminaries of this work are presented in Section 2. Section 3 describes our adversary model. Section 4 presents the details of AMAO. Experimental evaluation is shown in Section 5. Finally, Section 6 discusses the limitations of the proposed defense and Section 7 concludes the paper.

2 Preliminaries

2.1 Model extraction attacks

There have been a lot of model extraction attacks in stealing various aspects of a victim model, such as functionality [8], [10], [11], [12], [13], [18], [29], [31], hyperparameters [9] and architecture [32], [33]. In this work, we focus on functionality stealing, i.e., the goal of the attacker is to reconstruct a substitute model with similar functionality as the victim model. Specifically, the strategies of model functionality stealing attack can be divided into two categories according to the adversary's knowledge of the training data.

As for the scenario where the adversary knows a small portion of the training data, the adversary usually employs various methods (e.g., adversarial attack and data augmentation) to expand this small dataset and uses the expanded dataset as the query samples. For example, *Papernot et al.* [18] proposed *Jacobian-Based Dataset Augmentation (JBDA)*,

which constructed a synthetic dataset by iteratively augmenting an initial dataset of seed samples. Each synthetic sample is generated by a perturbed sample in the dataset using the jacobian of the substitute model's loss function. Then, the synthetic dataset is sent to the victim model for labeling and the substitute model is retrained with the labeled synthetic dataset. *Juuti et al.* [10] pointed out that the synthetic samples generated by *JBDA* tended to overlap, and did not contribute new information about the victim model. Thus, they proposed *JBDA-TR*, which mitigates this overlapping behavior by employing targeted randomly chosen iterative *FGSM* [34]. *Yu et al.* [12] developed *Cloudleak*, which employs a feature-based adversarial attack [35] to generate synthetic samples to gain more information of the victim model. Besides, *Cloudleak* also selects the substitute model from candidate Model Zoo [36] and uses transfer learning methods to retrain it with the obtained sample-label pairs.

As for the scenario where the adversary has no knowledge of the training data, the adversary usually selects (or generates) query samples from a surrogate dataset. For instance, *Orekondy et al.* [11] proposed *KnockoffNet*, which used surrogate data to query the victim model and trained the substitute model. *Pal et al.* [13] and *Gong et al.* [29] employed active learning to efficiently select query samples from a surrogate dataset. *Truong et al.* [31] generated query samples from the surrogate dataset using Generative Adversarial Networks (GAN). The effectiveness of this type of model stealing attacks depends heavily on the selection of the surrogate dataset.

In this work, we evaluate our defense against three state-of-the-art model extraction attacks *JBDA-TR* [10], *Cloudleak* [12] and *KnockoffNet* [11].

2.2 Defenses against model extraction attacks

The previous defenses against model extraction can be broadly divided into three categories:

Behavior detection aims to distinguish malicious queries or malicious users by analyzing the queries received by the victim model. For instance, *Juuti et al.* [10] proposed to detect malicious queries by evaluating the L_2 distance between successive queries. *Kariyappa et al.* [23] treated malicious queries as OOD (out-of-distribution) samples and employed OOD detection methods to identify them.

Prediction with perturbations reduces the effect of model extraction attacks by returning perturbed predictions. For example, *Lee et al.* [25] proposed a defense method that added noise to the output of the victim model, thereby degrading the effectiveness of model extraction attacks. *Orekondy et al.* [26] also proposed adding perturbations to the outputs, where the perturbations can be solved by a bilevel optimization problem. [23] trained a reverse model by minimizing a reverse cross-entropy loss to produce perturbed predictions and replied the perturbed results to malicious users.

Model watermarking is a technique to protect the intellectual property of a commercial model. Traditional DNN watermarking schemes [37], [38] train the model with pre-defined sample-label pairs. In this way, these sample-label pairs are embedded as watermarks into this model. The model owner can verify the legitimacy of a suspicious model by querying the suspicious model with these customized samples and checking whether it outputs the pre-defined labels. However, this method is inefficient in defending against model extraction attacks, because the adversary obtains the substitute model through training (under its query-output pairs) rather than directly copying model parameters. To combat that, several improvements have been proposed to verify the legitimacy of the extracted model. For instance, *Szyller et al.* [27] proposed a watermarking scheme against model extraction attacks. They changed the responses of a small subset of queries through a hash function and used this dataset as a watermark to verify the ownership of the model; *Jia et al.* [39] proposed a watermarking scheme that embedded watermarks into the victim model during the training phase. It is not only robust against model extraction attacks, but also effective in mitigating backdoor attacks [40].

2.3 Adversarial samples and adversarial training

An adversarial sample (or adversarial example) refers to a specially crafted input that is perturbed by hardly perceptible perturbations and induces a misclassification by a machine learning model. There are numerous methods to generate adversarial samples, we introduce two representative techniques as follows (which are used in our experiments).

Fast Gradient Sign Method (FGSM) [34] uses the gradients of the loss function to create an adversarial sample:

$$x_{adv} = x + \epsilon * \operatorname{sign}\left(\nabla_x J(x, y)\right) \tag{1}$$

where ∇ denotes the gradient, J is the loss function that measures the classification error in the machine learning algorithm, x_{adv} denotes the adversarial image, x is the clean sample and y is the corresponding label of x, ϵ is the step size of the perturbations and sign indicates the sign of the perturbations.

Project Gradient Descent (PGD) [41] can be regarded as an iterative version of *FGSM*. It initializes the adversarial sample by adding a random perturbation within the allowed norm ball to the original sample. Then, it updates the adversarial sample employing *FGSM* repeatedly. During each iteration, the adversarial sample will clip to the specified range. *PGD* can be summarised in the following formula:

$$x^{(t+1)} = \Pi_{x+\mathcal{S}}\left(x^{(t)} + \epsilon * \operatorname{sign}\left(\nabla_{x^{(t)}} J(x^{(t)}, y)\right)\right) \tag{2}$$

where $x^{(t)}$ denotes the adversarial sample of the t-th iteration. $\Pi_{x+\mathcal{S}}$ means that if the perturbations of the adversarial sample exceeds a certain range, it must be mapped back to the specified range $x + \mathcal{S}$.

Adversarial training [34] is one of the most effective defenses to mitigate adversarial attacks. It adds adversarial samples into the training dataset, thereby making the model more robust against adversarial samples. *Madry et al.* [41] defined the principle of adversarial training as the following Min-Max formula:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{r_{adv}\in\mathcal{S}} L\left(\theta, x + r_{adv}, y\right)\right] \tag{3}$$

the internal max is to find the worst perturbation to maximize the loss function, where $x + r_{adv}$ denotes the adversarial sample with a perturbation r_{adv}, y denotes the label of the sample, θ is the parameters of the model, L is the loss function of the adversarial sample and $\mathcal{S}$ is the range space of the perturbation; the external min is to find the most robust model parameters based on this attack method, where $\mathcal{D}$ is the distribution of the input samples.

3 Adversary Model

3.1 Adversary objective

The goal of model stealing attacks can be divided into three categories: architecture stealing [32], [33], hyperparameter stealing [9] and functionality stealing. This work focuses on functionality stealing, i.e., the attacker intends to achieve high performance of the substitute model. We take both test accuracy and fidelity as metrics to evaluate the performance of the substitute model (the two metrics are widely adopted in the previous model stealing attacks [8], [10], [11], [42]):

- **Test accuracy** refers to the accuracy of the extracted model f' on the test set D_{test} of the victim model. This goal may be of interest to a theft-motivated adversary who wants to create a local copy of the victim model. After replicating the functionality of the victim model, the adversary does not have to pay any more for public APIs.
- **Fidelity** is considered as label agreement in this work, i.e., the victim model and the extracted model output the same label for the same sample. This goal may be of interest to an attack-motivated adversary who wants to know the specifics of the victim model through model extraction attacks. After a model extraction attack, the adversary can more easily launch other attacks such as an adversarial attack or a membership attack.

3.2 Adversary knowledge

We assume the adversary is *data-limited* and has only *black-box* query access the victim model. *Data-limited* means the adversary only has a small number of natural samples. Black-box query access means the adversary can only access to the victim model on black-box interactions, i.e., samples $\{x_1, x_2, ..., x_n\}$ in, predictions $\{f(x_1), f(x_2), ..., f(x_n)\}$ out. According to the predictions of the victim model, model extraction attacks can be divided into two scenarios: the hard-label scenario where the adversary only obtains the predicted label of the query and the soft-label scenario where the adversary obtains the probability vector of the query. In this work, we evaluate our defense framework in both scenarios.

3.3 Adversary strategy

With a limited number of natural samples, the adversary can generate synthetic data or just use surrogate data to query the victim model and obtain the output results. After that, these query-output pairs are used as training data to train the substitute model. We consider three state-of-the-art attacks to evaluate our defense:

- ***JBDA-TR*** [10] is an improved version of the *jacobian-based dataset augmentation (JBDA)* proposed in [18]. The process of it is described in Algorithm 1. Concretely, for each sample $x \in D_{train}$, *JBDA-TR* generates a synthetic sample through targeted randomly chosen iterative *FGSM* (i.e., the targeted label of *FGSM* is changed during each iteration). The targeted randomly chosen iterative *FGSM* (*FGSM-TR*) can be formulated with the following equation:

$$x^{(t+1)} = x^{(t)} - \epsilon * \text{sign}\left(\nabla_{x^{(t)}} J(x^{(t)}, y_r)\right) \tag{4}$$

  where $x^{(t)}$ denotes the synthetic sample of the t-th iteration and y_r is a random target class (changes at each iteration). After that, these synthetic samples are labeled by the victim model and the sample-label pairs are used to augment D_{train} and retrain the substitute model.
- ***Cloudleak*** [12] also uses synthetic samples to query the victim model and obtains the prediction results. Different from *JBDA-TR*, *Cloudleak* employs a feature-based adversarial attack [35] to construct synthetic samples and fine-tunes a pre-trained substitute model (from candidate Model Zoo [36]) with the obtained sample-label pairs.
- ***KnockoffNet*** [11] uses surrogate data from another distribution or the same distribution to query the victim model and trains a substitute model with the predictions of the victim model.

Algorithm 1 The process of *JBDA-TR*

Input: unlabeled natural samples $D_{train} = \{x_1, x_2, ..., x_n\}$, the victim model f
Output: the substitute model f'

$\{f(x_1), f(x_2), ..., f(x_n)\} \leftarrow \text{LABEL}(\{x_1, x_2, ..., x_n\}, f)$
$D_{train} \leftarrow \{(x_1, f(x_1)), ..., (x_n, f(x_n))\}$
$f' \leftarrow \text{TRAIN}\ (f', D_{train})$
for $r = 1$ to r_{max} (round counter) **do**
    $\{x'_1, x'_2, ..., x'_n\} \leftarrow \text{FGSM-TR}\ (\{x_1, x_2, ..., x_n\})$
    $\{f'(x_1), ..., f'(x_n)\} \leftarrow \text{LABEL}(\{x'_1, ..., x'_n\}, f)$
    $D_{synthetic} \leftarrow \{(x'_1, f'(x'_1)), ..., (x'_n, f'(x'_n))\}$
    $D_{train} \leftarrow D_{train} \cup D_{synthetic}$
    $f' \leftarrow \text{TRAIN}\ (f', D_{train})$
return f'

4 Proposed AMAO

4.1 Overview

Fig 3 gives an overview on our proposed AMAO, each phase of AMAO is closely connected and one phase will be beneficial to the next phase. Below we describe these phases in detail.

4.2 Adversarial training for decreasing the effect of model extraction attacks

The intuition for employing adversarial training to defend against model extraction attacks is that the samples close to the decision boundary of the victim model are more informative in training the substitute model (especially in the hard-label scenario) and adversarial training makes it more difficult for an adversary to generate such samples.

Samples close to the decision boundary are more informative. For model stealing attacks that use synthetic samples, the effectiveness of the attack largely depends on

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3261327

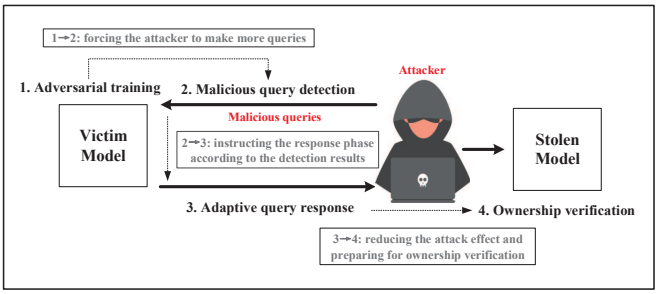


Fig. 3: The architecture of AMAO

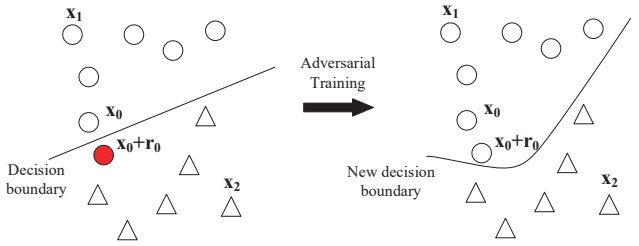


Fig. 4: Adversarial training

the quality of the synthetic samples. It is similar to the scenario of active learning, where the user actively selects (or generates) some informative samples that are worthy of being labeled and trains a model with these sample-label pairs. According to research on active learning [43], samples close to the decision boundary tend to be more informative and selecting these samples is a commonly used strategy in active learning (such as the *DeepFool Active Learning algorithm* [44]). Thus, many model extraction attacks such as *JDBA* [18], *JDBA-RT* [10], *Cloudleak* [12] and *Activethief* [13] proposed to employ adversarial samples to serve as synthetic samples, because the adversarial samples tend to be close to or across the decision boundary of the original category, which is more informative in extracting the victim model.

Adversarial training makes it more difficult for a black-box adversary to generate informative synthetic samples. Motivated by the above observation, we employ adversarial training to reduce the attack effect of model extraction attacks. Adversarial training makes it more difficult to generate adversarial samples that cross the decision boundary. Although the adversary is still able to generate adversarial samples through multiple iterations of querying and generating, similar to the scenario of black-box adversarial attacks, the adversary often needs to spend much more queries (i.e., more iterations of the adversarial attack) to generate an adversarial sample that crosses the decision boundary against an adversarially-trained model.

Fig 4 depicts a visualization of this observation. As depicted in the left part of Fig 4, x_0 and $x_0 + r_0$ are samples near the decision boundary and on the two sides of the boundary, where x_0 is the benign sample with the minimum classification confidence and $x_0 + r_0$ is the adversarial sample (generated from x_0) with the minimum classification confidence. Apparently, in comparison to those samples far from the decision boundary such as x_1 and x_2, x_0 and $x_0 + r_0$ can provide more useful information about the classifier, and the decision boundary of the victim model is more easily extracted through training with them. As depicted in the right part of Fig 4, the benign sample x_0 and the generated sample $x_0 + r_0$ may locate on the same side of the decision boundary after adversarial training, the information they reveal is greatly reduced (especially in the hard-label scenario mentioned in Section 3.2). Thus, the adversary is forced to submit more malicious queries on generating synthetic samples that cross the decision boundary. This provides favorable conditions for the next detection phase as more malicious queries make the adversary easier to be detected. More malicious queries also result in greater overhead for the adversary and make the attack less worthwhile. The experimental results in Section 5.2 demonstrate the effectiveness of adversarial training.

4.3 Malicious query detection

As described in Section 3.3, the adversary uses synthetic data or surrogate data to query the victim model and obtain the output results. The function of malicious query detection is to distinguish these samples from benign samples.

Generally, classifiers are always more confident in classifying benign queries and less confident in classifying malicious queries. Thus, the maximum softmax probability (MSP) can be used as a metric to distinguish malicious queries [23]. Specifically, the softmax probability (SP) and MSP of query x can be computed as Eq. (5) and Eq. (6):

$$SP(x,i) = \frac{\exp\left(z_i(x)\right)}{\sum_{k=1}^{K}\exp\left(z_k(x)\right)} \tag{5}$$

$$MSP(x) = \max[SP(x,i)] \quad i = 1,2,...,K \tag{6}$$

where $z_i(x)$ is the logit value (i.e., the output vector of the model) of the i-th class and K is the number of classes. Benign queries tend to have greater MSPs than malicious queries and they can be distinguished by a pre-defined threshold.

However, recent works [45], [46] have shown that classifiers tend to be overconfident in classifying adversarial samples (which are used as malicious queries by model extraction attacks [10], [12], [18]) and they also have large MSPs, making it difficult to distinguish them from benign queries. Therefore, we introduce the temperature scaling [47] technique to mitigate the problem of overconfidence of the classifier. Specifically, we employ the temperature-scaled maximum softmax probability (TMSP) as the metric to distinguish malicious queries. The temperature-scaled softmax probability (TSP) and TMSP of query x can be calculated as Eq. (7) and Eq. (8):

$$TSP(x,i,T) = \frac{\exp\left(z_i(x)/T\right)}{\sum_{k=1}^{K}\exp\left(z_k(x)/T\right)} \tag{7}$$

$$TMSP(x,T) = \max[TSP(x,i,T)] \quad i = 1,2,...,K \tag{8}$$

Temperature scaling is a soft-label smoothing method in knowledge distillation [47], which smooths or sharpens the predicted probability vector through a temperature coefficient T. Concretely, when $T \to \infty$, the prediction probability vector will be smoothed, which reduces the confidence in the prediction; when $T \to 0$, the prediction probability vector will become sharper and the confidence in the prediction

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3261327

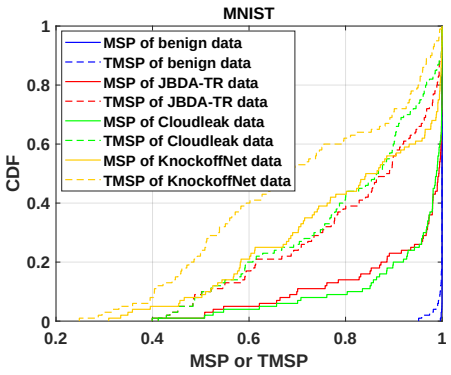


Fig. 5: CDF of MSP or TMSP for benign data, *JBDA-TR* data, and *KnockoffNet* data. High values of MSP or TMSP indicate benign data while low values of MSP or TMSP indicate malicious data.

will tend to 1, which increases the confidence of the classifier in the prediction.

In this work, the problem of overconfidence of the classifier can be mitigated by appropriately increasing the value of T. For example, we analyze the MSP and TMSP values of benign or malicious queries using a LeNet-5 network trained on the MNIST dataset. Fig 5 illustrates the CDF (Cumulative Distribution Function) of MSP and TMSP for benign queries and malicious queries. It indicates that some malicious queries (especially malicious queries generated by *JBDA-TR* and *Cloudleak*) can also produce high values of MSP, making it difficult for the detector to distinguish. However, this problem has been mitigated in the case of TMSP, CDFs of malicious queries are more concentrated in lower values of TMSP, making the malicious samples more easily distinguishable. It demonstrates that the temperature scaling technique makes the malicious queries more distinguishable, thus improving the performance of detector. More detailed experimental evaluations on the detection method will be provided in Section 5.3.

After calculating the TMSP of the query, a marker function $Detect(x)$ is used to flag benign queries and malicious queries:

$$Detect(x) = \begin{cases} 0, & TMSP(x,T) > \lambda \\ 1, & TMSP(x,T) \leq \lambda \end{cases} \tag{9}$$

$Detect(x)$ being 0 represents the query x is a benign query while $Detect(x)$ being 1 represents x is a malicious query. λ is the threshold to distinguish malicious queries from benign queries.

Finally, we define a suspect function S based on the detection result to measure the degree of suspicion of a user:

$$S = \frac{\sum_{i=1}^{N} Detect(x_i)}{N} \tag{10}$$

where N is the total number of queries submitted by the user[1].

1. Since S is a statistical probability value, N should be large enough (e.g., greater than 50 times) to ensure the reliability of S. The update of S does not need to be real-time, it can be updated intermittently.

4.4 Adaptive query response

After malicious query detection, in order to reduce the performance of the substitute model and prepare for the subsequent ownership verification, we propose to return perturbed results to malicious users with a pre-defined probability. The key problem of the adaptive query response is to determine which sample to perturb and how to perturb its predicted results. In our adaptive query response phase, we propose a label-flipping attack based on the optimal stopping theory to instruct the query response and employ the image pHash algorithm to ensure the indistinguishability of the response.

4.4.1 Instructing the query response through a label flipping attack

In this work, the defender can be formulated as an adversary of a label-flipping attack [48], [49], [50] whose goal is to choose the optimal poisoning samples to flip the labels, thereby decreasing the performance of the substitute model. Unlike label-flipping poisoning attacks where the adversary has full knowledge of the training dataset, the defender in this work can only decide whether the current query is worthy of a label-flipping because the adversary's queries and victim model responses are usually executed one by one. **Thus, we employ the optimal stopping theory [51] to instruct the label-flipping attack.**

Specifically, we first assume the pre-defined perturbed probability is $1/M$ and divide the queries received from a malicious user into groups of M. In this way, the defender replies with one perturbed result for one group of queries. After that, the defender calculates the gap between the maximum softmax probability and the minimum softmax probability of each query in the group as Eq. (11). Finally, the defender then selects the sample with the largest gap and flips its label to the category with the minimum confidence[2]. By doing so, the classification error of the substitute model is maximized when the substitute model is trained with these perturbed sample-result pairs.

$$\begin{aligned} Gap(x) &= \max[SP(x,i)] - \min[SP(x,i)] \\ & i = 1, 2, ..., K \end{aligned} \tag{11}$$

However, since the defender receives the adversary's queries individually, the defender only knows the confidence gap between the current query and previous queries. It can only decide whether to flip the result label of the current query, not the previous queries. The optimal stopping theory is used to solve this problem. Specifically, for a group of queries, the defender first observes the first $r-1$ queries and does not select them. In the following $M-r+1$ queries, if the gap of any query is larger than the largest gap of the first $r-1$ queries, the query is selected for label-flipping.

The best value of r can be obtained through the following calculations. We first define A as the event that the query i is the best candidate (i.e., the query with the largest gap)

2. In the soft-label scenario, our method exchanges the maximum probability value with the minimum probability value.

and B as the event that the query i is selected. For any r, the probability of the best candidate being selected is:

$$\begin{aligned} P(r) &= \sum_{i=1}^{M} P(A \cap B) \\ &= \sum_{i=1}^{M} P(A)P(B|A) \\ &= [\sum_{i=1}^{r-1} 0 + \sum_{i=r}^{M} P(C|A)] \cdot \frac{1}{M} \\ &= \sum_{i=r}^{M} \frac{r-1}{i-1} \times \frac{1}{M} = \frac{r-1}{M} \sum_{i=r}^{M} \frac{1}{i-1} \end{aligned}$$

where C represents that the best of the first $i-1$ queries is in the first $r-1$ rejected queries. For the final result, let M approach ∞, denote the limit of r/M as x and denote i/M as t, the result can be approximated as the following integral:

$$P(x) = x \int_{x}^{1} \frac{1}{t} dt = -x \ln x \tag{12}$$

Through computing the derivatives of the Eq. (12), we can obtain that the optimal x is equal to $1/e$. Thus, the optimal r equals to M/e.

Finally, the process of adaptive query response can be described as the following Algorithm 2.

Algorithm 2 The process of adaptive query response

Input: a set of queries $x_1, x_2, ..., x_M$
Output: the optimal query that needs to be label-flipped

$Gap^* = 0$
for $i = 1$ to $(M/e - 1)$ **do**
  if $Gap(x_i) > Gap^*$ **then**
    $Gap^* = Gap(x_i)$
for $j = M/e$ to M **do**
  if $Gap(x_j) > Gap^*$ **then**
    return x_j
  if $j == M$ **then**
    return x_M

4.4.2 Ensuring the indistinguishability of the response strategy through image pHash algorithm

As described in our threat model, the adversary only has a limited number of samples from the original dataset and it may generate synthetic samples from this initial seed dataset. Thus, the adversary's synthetic samples tend to have high similarities because they are generated by adversarial attacks or other methods (such as data augmentation [12], [29], [30]) from the same seed dataset. An adaptive adversary may be aware of the adaptive response strategy if it receives different predictions for the same or similar queries. Then, it can employ some methods to evade the defenses, such as discarding the perturbed results and querying the victim model with another benign account. Thus, the indistinguishability of the response strategy is indispensable for the defense, i.e., when returning a perturbed prediction for input x, the defender expects two similar inputs x and x' to have the same prediction. Previous work [27] used the hash value of the input to produce the random prediction and employed a mapping function to ensure the hash value is invariant to minor modifications. However, it is only applicable against small perturbations such as adversarial perturbations but can not guarantee indistinguishability under the case of data augmentation [12], [29], [30]. To address this problem, in this work, we employ the image pHash algorithm to ensure the indistinguishability of the response strategy.

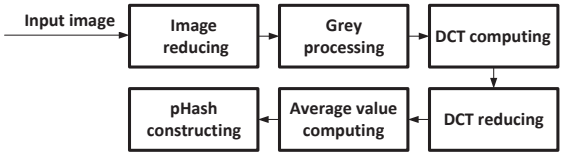


Fig. 6: The processes of the image pHash algorithm

The image pHash algorithm is a technique to accurately and quickly calculate the similarity between different images. The process of it is illustrated in Fig 6. The input image is first size-reduced and color-reduced to a grayscale one to simplify the subsequent DCT (Discrete Cosine Transform) [52] computation. Then, it calculates the DCT of the image and keeps the top-left 8*8 matrix, because this part represents the lowest frequency in the image. After that, it computes the average value of all matrixes. Finally, according to the 8*8 DCT matrix, a 64-bit hash value is constructed, with each bit of the hash set as 0 or 1 depending on whether each of the 64 DCT values is above or below the average value.

Concretely, if the victim model outputs a perturbed prediction, the defender will record the prediction and the corresponding query. After that, if any query from a malicious user is evaluated to be similar to a query already recorded by the image pHash algorithm, the defender directly returns the recorded perturbed prediction of the recorded query. This ensures the indistinguishability of the adaptive response strategy and makes it more imperceptible to the adversary. Furthermore, the recorded query-prediction pairs are also expected to be embedded as a backdoor[3] in the substitute model and will be used in the next ownership verification phase.

4.5 Ownership verification

As a preparation for the ownership verification, for each malicious user, the defender maintains a perturbed results set W_i $(i = 1, 2, ..., n)$ to store the perturbed query-prediction pairs $\{(x_1^i, y_1^i), (x_2^i, y_2^i), ..., (x_{m_i}^i, y_{m_i}^i)\}$, where n is the number of malicious users and m_i $(i = 1, 2, ..., n)$ represents the number of perturbed query-prediction pairs for the i-th malicious user.

Then, the ownership verification can be executed as Algorithm 3. For every perturbed results set W_i, the defender calculates the proportion of verified perturbed results to the total number of perturbed results in this set, where the perturbed result (x_j^i, y_j^i) successfully verifies when $f_s(x_j^i) = y_j^i$. If the proportion exceeds the pre-defined threshold t, the

3. According to previous works [27], [53], DNN models with a large number of parameters can remember a certain amount of training data with arbitrarily incorrect labels. This is the rationale for the existence of DNN backdoors and for our ownership verification scheme.

Algorithm 3 Identifying the ownership of the suspicious model

Input: threshold t, suspicious model f_s
Output: The result of whether f_s is a stolen model

Initialize $Flag = False$
for $i = 1$ to n **do** /* For all perturbed results sets */
  for $j = 1$ to m_i **do**
    $count = 0$
    if $f_s(x_j^i) = y_j^i$ **then**
      $count = count + 1$
  if $count/m_i > t$ **then**
    $Flag = True$
    return $Flag$
return $Flag$

suspicious model will be identified as an illegal one stolen from the victim model. Otherwise, the suspicious model will be identified as a benign one.

5 Experimental Evaluation

5.1 Setup

5.1.1 Datasets and victim models

We utilize LeNet-5, AlexNet, ResNet18 and ResNet34 as the victim models on MNIST [54], FashionMNIST (FMNIST) [55], CIFAR-10 [56] and ImageNette[4] [57], respectively. The details of the datasets are presented in TABLE 1.

TABLE 1: Details for the datasets

Dataset	Image size	Classes	Train data	Test data
MNIST	28×28×1	10	60,000	10,000
FMNIST	28×28×1	10	60,000	10,000
CIFAR-10	32×32×3	10	50,000	10,000
ImageNette	224×224×3	10	9,469	3,925

5.1.2 Attack configuration

To improve the efficiency of the attack and evaluate our defense under a stronger adversary, we use the same model architecture as the victim's model to train the substitute model[5]. Obviously, if our defense shows good defense effect under this strong assumption, it will perform better in the case that the adversary has no knowledge of the victim model.

As for *JBDA-TR* and *Cloudleak*, the initial seed dataset of the substitute model is made of 100, 100, 1,000 and 1,000 training samples from the MNIST, F-MNIST, CIFAR-10 and ImageNette, respectively. The query budget is about 10,000, 10,000, 100,000 and 100,000 for the four datasets. Specifically, for *JBDA-TR*, we perform 6 rounds of augmentation to steal the victim model, between each augmentation round, the substitute model is trained for 20 epochs; for *Cloudleak*, the pre-trained substitute model is fine-tuned for 20 epochs.

As for *KnockoffNet*, we use FashionMNIST, MNIST, CIFAR-100 and ImageNet[6] as the surrogate datasets of MNIST, FashionMNIST, CIFAR-10 and ImageNette datasets, respectively. The query budget is 60,000, 60,000, 50,000 and 13,000 for MNIST, F-MNIST, CIFAR-10 and ImageNette, respectively. The substitute models are trained for 50 epochs.

It is worth mentioning that we focus on evaluating the effectiveness of our defense against different model extraction attacks, rather than evaluating the effectiveness of different model extraction attacks. The settings of the query budget are different for the three attacks due to their different attack mechanisms. This setting will not affect the evaluations of the defenses.

5.1.3 Metrics

We use test accuracy and fidelity mentioned in Section 3.1 as the metrics for evaluation. Notably, we find that the experimental results of fidelity show the same tendency as the test accuracy. Therefore, the experimental results of fidelity are omitted for brevity.

5.1.4 Comparison with existing defenses

We compare our scheme with five state-of-the-art defense methods:

- ***PRADA* [10]**: *PRADA* is a detection method against malicious queries generated by model extraction attackers. Specifically, it calculates the minimum distance between a new queried sample x_i and any previous sample of the same class:

$$\min d_i = \min_{j<i, y_j=y_i} \|x_i - x_j\|_2 \tag{13}$$

  the intuition of this detection method is that the distribution of distances d_i between benign samples tends to follow Gaussian distribution. However, the synthetic samples are generated by a duplication phase from an initial seed dataset (such as *JBDA-TR* and *Cloudleak*). Thus, many synthetic samples are generated from the same seed sample, thereby causing the distribution of distances between successive malicious queries to deviate from the Gaussian distribution.
- ***OOD detection* [23]**: *OOD detection* treats malicious queries as OOD samples and employs OOD detection methods to distinguish them. This detection method is selected as the baseline to evaluate our detection method in Section 5.3.
- ***DAWN* [27]**: *DAWN* calculates the hash value of the input and returns the label according to this hash value as the predicted label. The perturbed label can not only decrease the performance of the substitute model but also serves as a backdoor to verify the ownership of any suspicious models.
- ***Deceptive Perturbation* [25]**: *Deceptive Perturbation* is an accuracy-preserving defense that adds perturbations to the probability vectors of the model but ensures that the predicted labels remain unchanged.

4. ImageNette is a representative subset of ImageNet.

5. It is possible for an adversary to steal the architecture and hyper-parameter of the model before stealing the functionality of the victim model (there has been some work in this area such as [9], [32], [33]).

6. The surrogate samples of ImageNet and the original training samples of ImageNette have no overlap.

TABLE 2: The test accuracy (%) of the victim models

Dataset	Standard training	Adversarial training
MNIST	99.21	99.13
F-MNIST	92.11	89.24
CIFAR-10	93.40	91.01
ImageNette	90.62	88.99

- ***Adaptive Misinformation*** **[23]**: *Adaptive Misinformation* trains a reverse model by minimizing a reverse cross-entropy loss to produce incorrect predictions. The reverse cross-entropy loss is defined below:

$$\mathcal{L}_{reverse} = \texttt{CE}((1 - f(x;\theta)), y) \tag{14}$$

where CE represents the cross-entropy loss. For the response to be perturbed, it feeds the query to the reverse model and returns the probability vector of the reverse model to the adversary.

5.1.5 Overview of the experimental evaluation

We first evaluate the effectiveness of adversarial training in decreasing the effect of model extraction attacks in Section 5.2. Then, we compare our detection method with *OOD detection* [23] in Section 5.3. After that, in Section 5.4, we evaluate the overall effectiveness and computational overhead of AMAO from end to end, where three prior works *DAWN* [27], *Deceptive Perturbation* [25] and *Adaptive Misinformation* [23] are selected as the baseline for comparison. Then, the effectiveness of the phase of ownership verification is evaluated in Section 5.5. Finally, we conduct extensive experiments in Section 5.6 to demonstrate that AMAO is more robust than prior works (such as *PRADA* [10] and *DAWN* [27]) against adaptive adversaries.

5.2 Evaluations on adversarial training

To evaluate the defense effect of adversarial training, we train two identical victim models: one is built with a traditional training process and the other is built with an adversarial training process. Test accuracy of these victim models is presented in TABLE 2.

After that, we perform *JBDA-TR*, *KnockoffNet* and *Cloudleak* on these models. As shown in TABLE 3, the accuracy of the substitute model constructed by stealing the adversarially trained model is always lower than the accuracy of the substitute model constructed by stealing the standard trained model. Besides, the results indicate that adversarial training is more effective in the hard-label scenario and less effective in the soft-label scenario. This is because the probability vector in the soft-label scenario is more informative than a hard label. The adversary can still obtain more information about the decision boundary of the adversarially trained victim model. Overall, the results demonstrate that adversarial training can decrease the attack effect of model extraction attacks and force the adversary to submit more malicious queries for the desired performance (the detailed reason for the effectiveness of adversarial training in decreasing the attack effect is described in Section 2.3.).

TABLE 3: The test accuracy (%) of the substitute model under the defense of adversarial training

Attack	Scenario	Dataset	Std. train	Adv. train
JBDA-TR	Hard label	MNIST	91.23	**87.08**
		F-MNIST	79.33	**75.44**
		CIFAR-10	42.80	**40.05**
		ImageNette	51.27	**47.26**
	Soft label	MNIST	95.58	**89.22**
		F-MNIST	81.44	**80.42**
		CIFAR-10	43.97	**40.36**
		ImageNette	55.76	**52.90**
KnockoffNet	Hard label	MNIST	89.57	**78.89**
		F-MNIST	40.38	**35.80**
		CIFAR-10	69.37	**66.89**
		ImageNette	55.90	**50.80**
	Soft label	MNIST	91.72	**87.53**
		F-MNIST	42.10	**41.16**
		CIFAR-10	73.02	**72.20**
		ImageNette	68.18	**65.01**
Cloudleak	Hard label	MNIST	83.72	**79.46**
		F-MNIST	76.07	**71.91**
		CIFAR-10	78.15	**76.69**
		ImageNette	86.64	**82.77**
	Soft label	MNIST	86.36	**84.77**
		F-MNIST	78.26	**74.83**
		CIFAR-10	80.04	**79.19**
		ImageNette	88.19	**85.50**

TABLE 4: The optimal hyperparameters of the detector

The attack method used to create the validation set	Dataset	λ	T
JBDA-TR	MNIST	0.998	1.53
	F-MNIST	0.998	1.51
	CIFAR-10	0.981	1.50
	ImageNette	0.907	1.24
KnockoffNet	MNIST	0.953	2.21
	F-MNIST	0.994	2.29
	CIFAR-10	0.993	1.21
	ImageNette	0.963	1.39
Cloudleak	MNIST	0.996	2.30
	F-MNIST	0.961	2.25
	CIFAR-10	0.919	1.92
	ImageNette	0.914	1.19

5.3 Evaluations on malicious query detection

Firstly, before evaluating the effectiveness of our detection method, we need to find the optimal hyperparameters (T and λ) first. Specifically, we define samples from the test dataset as benign and consider the synthetic (or surrogate) samples as malicious. The defender is assumed to has a small number (i.e., 100) of malicious samples from *JBDA-TR*, *KnockoffNet* and *Cloudleak*. It can construct a small validation dataset that includes malicious samples and benign samples to find the optimal hyperparameters of the detector (as provided in TABLE 4). Notably, as provided in TABLE 5, these hyperparameters have good generalizability that the optimal hyperparameters for one attack can also achieve good performance in detecting the other attacks.

After that, without loss of generality, we choose the hyperparameters for *JBDA-TR* in TABLE 4 and compare our detection method with the *OOD detection* [23]. In addition to accuracy, we introduce another comprehensive metric (i.e.,

TABLE 5: The generalizability of the hyperparameters

Dataset	Attack	The detection accuracy (%) with different hyperparameters		
		JBDA-TR	*KnockoffNet*	*Cloudleak*
MNIST	*JBDA-TR*	87.85	85.04	83.44
	KnockoffNet	93.75	94.17	86.89
	Cloudleak	84.99	82.18	85.13
F-MNIST	*JBDA-TR*	81.20	79.02	80.40
	KnockoffNet	69.89	70.11	69.75
	Cloudleak	74.37	74.76	74.87
CIFAR-10	*JBDA-TR*	78.99	77.15	75.99
	KnockoffNet	86.23	87.09	79.85
	Cloudleak	77.33	78.02	78.11
ImageNette	*JBDA-TR*	80.29	76.58	82.21
	KnockoffNet	81.13	79.30	81.39
	Cloudleak	83.04	76.79	83.31

TABLE 6: Performance comparison of our detection method and the *OOD detection* [23]

Dataset	Attack	Accuracy(%) [23]/Ours	F1-score [23]/Ours
MNIST	*JBDA-TR*	76.15/**87.85**	0.68/**0.87**
	KnockoffNet	86.20/**93.75**	0.82/**0.93**
	Cloudleak	72.23/**84.99**	0.61/**0.84**
F-MNIST	*JBDA-TR*	73.05/**81.20**	0.66/**0.81**
	KnockoffNet	65.91/**69.89**	0.53/**0.67**
	Cloudleak	66.33/**74.37**	0.54/**0.73**
CIFAR-10	*JBDA-TR*	73.41/**78.99**	0.67/**0.76**
	KnockoffNet	84.55/**86.23**	0.83/**0.87**
	Cloudleak	77.04/**77.33**	0.73/**0.76**
ImageNette	*JBDA-TR*	77.11/**80.29**	0.80/**0.82**
	KnockoffNet	72.58/**81.13**	0.76/**0.81**
	Cloudleak	77.16/**83.04**	0.80/**0.84**

F1-score) to evaluate the performance of the detector. F1-score is the commonly used evaluation metric to measure the overall performance of the detector. Specifically, it is the harmonic average of precision and True Positive Rate (TPR):

$$F1 = \frac{2 * Precision * TPR}{Precision + TPR} \tag{15}$$

Precision represents the proportion of correctly detected positive (malicious) queries to all queries that are detected to be positive. TPR (or Recall) indicates the proportion of correctly detected positive queries to all positive queries. They can be calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{16}$$

$$TPR = \frac{TP}{TP + FN} \tag{17}$$

where the definitions of true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN) are described in TABLE 7.

TABLE 7: Confusion matrix

Actually label \ Detection result	Positive (malicious)	Negative (normal)
Positive (malicious)	TP	FN
Negative (normal)	FP	TN

The results in TABLE 6 indicate that our detection method has higher accuracy and F1-score than *OOD detection* [23] in detecting all types of malicious queries. This is mainly because our method mitigates the problem of overconfidence in *OOD detection* by introducing the temperature scaling technique (more details can be found in Section 4.3).

Besides, we also compare the robustness of our detection method and *PRADA* [10] against adaptive adversaries in Section 5.6. The experimental results demonstrate that *PRADA* is ineffective in detecting surrogate samples (such as *KnockoffNet*) and colluded malicious users, but our detection method still achieves high detection accuracy in these scenarios.

5.4 Overall evaluations on AMAO from end to end

In this section, we evaluate the effectiveness and computational overhead of AMAO from end to end.

5.4.1 Effectiveness of AMAO

Fisrtly, we evaluate the effectiveness of AMAO from end to end through the metric of test accuracy of the substitute model. **Three state-of-the-art defenses *DAWN* [27], *Deceptive Perturbation* [25] and *Adaptive Misinformation* [23] are selected as the baseline for comparison.**

Specifically, we assume the defender returns perturbed results with a probability of 1/16 and evaluate the attack performance under these defenses[7]. TABLE 8 shows the test accuracy of the substitute model under these defenses, where the results of the baseline for the hard-label scenario show the results of *DAWN* [27]; the results of the baseline for the soft-label scenario show the best results of *Deceptive Perturbation* [25] and *Adaptive Misinformation* [23]. The results demonstrate that our AMAO is much more effective than these prior works in reducing the attack performance of the considered three model extraction attacks. This is because our label-flipping-attack-based adaptive query response scheme is more effective in disrupting the training process of the substitute models, which affects the performance of model stealing attacks. Besides, AMAO also introduces adversarial training to further decrease the attack performance, which is not considered in any of the existing model stealing defenses. In addition, the results indicate that the defensive effect of AMAO is better against the attack of *JBDA-TR* and *Cloudleak*. This is because the two attacks use natural data or synthetic data as malicious queries, replying to malicious users with perturbed results can more directly and effectively decrease the performance of the substitute model.

Besides, we also calculate the test accuracy of the victim model to evaluate whether the model utility is greatly influenced by these defenses. Specifically, AMAO and *Adaptive Misinformation* [23] first distinguish malicious users by detecting malicious queries, and only output perturbed results for malicious users. They have almost no effect on the normal utility of the victim model[8]; *Deceptive Perturbation* [25] perturbs the classification probability values of the

7. We have also tested with other probabilities such as 1/8 and 1/32. The experiments give the same conclusions.

8. Adversarial training in AMAO has a slight effect on the utility of the victim model.

TABLE 8: The test accuracy (%) of the substitute model under the defense of AMAO and the baseline defense [23], [25], [27]

Attack	Scenario	Dataset	No defense	Baseline defense	AMAO
JBDA-TR	Hard label	MNIST	91.23	87.30	**67.21**
		F-MNIST	79.33	74.21	**65.21**
		CIFAR-10	42.80	35.35	**32.62**
		ImageNette	51.27	47.88	**45.57**
	Soft label	MNIST	95.58	91.80	**81.30**
		F-MNIST	81.44	75.48	**61.59**
		CIFAR-10	43.97	40.97	**37.16**
		ImageNette	55.76	50.70	**48.53**
KnockoffNet	Hard label	MNIST	89.57	70.44	**68.03**
		F-MNIST	40.38	34.95	**32.46**
		CIFAR-10	69.37	63.48	**49.30**
		ImageNette	55.90	50.88	**44.20**
	Soft label	MNIST	91.72	80.56	**80.11**
		F-MNIST	42.10	37.81	**34.24**
		CIFAR-10	73.02	71.35	**70.04**
		ImageNette	68.18	61.45	**57.04**
Cloudleak	Hard label	MNIST	83.72	73.14	**67.28**
		F-MNIST	76.07	67.82	**63.64**
		CIFAR-10	78.15	67.59	**61.02**
		ImageNette	86.64	73.60	**68.04**
	Soft label	MNIST	86.36	75.93	**72.88**
		F-MNIST	78.26	71.33	**64.94**
		CIFAR-10	80.04	71.09	**67.50**
		ImageNette	88.19	78.10	**71.59**

TABLE 9: The utility (i.e., test accuracy) of the victim model under AMAO and the baseline defense [23], [25], [27]

Dataset	No defense	[23], [25]	AMAO	[27]
MNIST	99.21	99.21	99.13	93.01
F-MNIST	92.11	92.11	89.24	86.35
CIFAR-10	93.40	93.40	91.01	87.56
ImageNette	90.62	90.62	88.99	84.95

output results, but keeps the output labels unchanged, so it also has no effect on the top-1 accuracy of the victim model; *DAWN* [27] changes the output label with the predefined probability. The results in TABLE 9 indicates that the impact of AMAO on the model utility is quite small (less than 3%), which is acceptable for the defender.

5.4.2 Computational overhead of AMAO

We evaluate the computational overhead of AMAO and present the results in TABLE 10. All experiments are run on NVIDIA 24GB Tesla M40.

It is important to point out that adversarial training is performed during the training process of the model, which can be considered as an offline operation. Besides, ownership verification only executes when a suspected model needs to be verified for ownership. Thus, the computational overhead of AMAO for a single query only includes the phases of malicious query detection and adaptive query response, which are acceptable for the defender. Besides, the computational overhead will be further reduced if these operations are run on high performance cloud servers.

5.5 The effectiveness of the ownership verification

In this section, we evaluate the effectiveness of ownership verification, which is the post hoc defense of AMAO. Concretely, experiments are conducted to calculate the watermark accuracy of the perturbed (or watermarked) results[9] in the phase of adaptive query response, where the watermark accuracy denotes the verification probability of the perturbed results. Meanwhile, we also evaluate the watermark accuracy of a substitute model trained with non-watermarked results as a comparison.

TABLE 10: Computational overhead of AMAO

Dataset	Phase	Computational overhead
MNIST	Adversarial training	9.19 (min)
	Malicious query detection	<0.01 (ms)
	Adaptive query response	17.31 (ms)
	Ownership verification	607.68 (ms)
F-MNIST	Adversarial training	15.63 (min)
	Malicious query detection	<0.01 (ms)
	Adaptive query response	21.25 (ms)
	Ownership verification	648.96 (ms)
CIFAR-10	Adversarial training	90.74 (min)
	Malicious query detection	<0.01 (ms)
	Adaptive query response	37.61 (ms)
	Ownership verification	794.57 (ms)
ImageNette	Adversarial training	217.05(min)
	Malicious query detection	<0.01 (ms)
	Adaptive query response	67.14(ms)
	Ownership verification	978.30(ms)

As shown in TABLE 11, all the substitute models trained with perturbed results have high watermark accuracy, which demonstrates that the perturbed results can serve as a model watermark and verify the ownership after a model extraction attack. Besides, as presented in the most right column in TABLE 11, the watermark accuracy of all substitute models trained without perturbed results is close to 0, which indicates that the perturbed results are hardly verified in normal models. Therefore, the defender can easily verify the ownership of the suspicious model through our ownership verification algorithm (Algorithm 3).

5.6 The robustness of AMAO against adaptive adversaries

In this section, we first evaluate the robustness of AMAO against adaptive adversaries with prior knowledge of the malicious query detection phase, the adaptive query response phase and the ownership verification phase, respectively[10]. Then, we evaluate the robustness of AMAO under a state-of-the-art adaptive model extraction attack strategy *D-DAE* [58]. In this way, we demonstrate that AMAO shows a significant defensive advantage over prior works.

5.6.1 Aware of the malicious query detection phase

Firstly, we consider the scenario in which the adversary is aware of the malicious query detection phase. The adversary may collude with multiple malicious users or mix malicious queries with benign queries or use surrogate data as malicious queries (such as *KnockoffNet*) to evade detection. We

9. The probability of perturbing the results is set to 1/16, which follows the assumption made in Section 5.4.1.

10. In practice, adversarial training is a commonly used technique to defend against adversarial attacks. Prior knowledge of the adversarial training phase is not helpful for model stealing attacks aimed at model functionality stealing. Thus, we only consider the adaptive adversary who is aware of the other three defensive phases.

TABLE 11: The watermark accuracy (%) of the substitute models

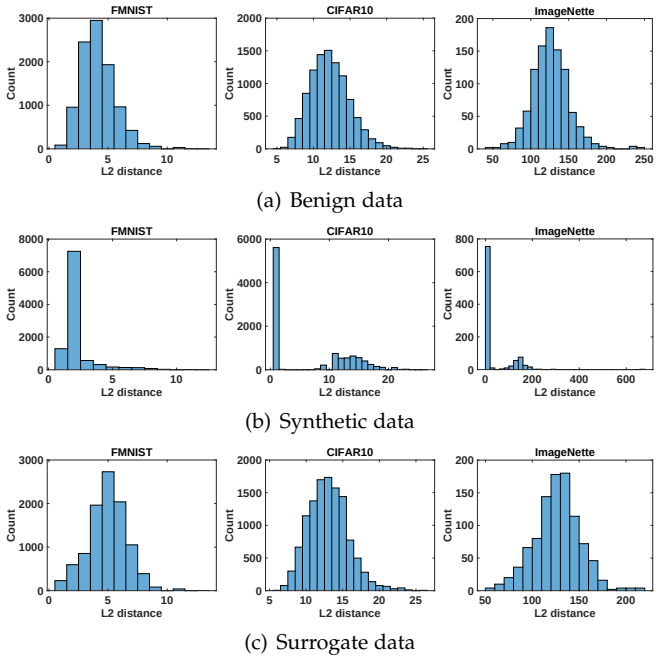
Attack	Scenario	Dataset	Watermarked results ✓	Watermarked results ×
JBDA-TR	Hard label	MNIST	89.71	0.31
		F-MNIST	96.75	0.01
		CIFAR-10	97.77	0.00
		ImageNette	90.88	0.37
	Soft label	MNIST	88.30	0.23
		F-MNIST	95.62	0.00
		CIFAR-10	96.13	0.00
		ImageNette	89.05	0.49
KnockoffNet	Hard label	MNIST	91.04	0.42
		F-MNIST	97.33	0.08
		CIFAR-10	100.00	0.27
		ImageNette	95.44	0.29
	Soft label	MNIST	90.26	0.37
		F-MNIST	97.65	0.04
		CIFAR-10	96.99	0.19
		ImageNette	87.68	0.36
Cloudleak	Hard label	MNIST	99.97	0.51
		F-MNIST	98.47	0.09
		CIFAR-10	97.17	0.24
		ImageNette	93.57	0.61
	Soft label	MNIST	96.82	0.40
		F-MNIST	94.23	0.02
		CIFAR-10	95.05	0.17
		ImageNette	91.10	0.35

demonstrate that an adaptive adversary employing these strategies can evade the detection of *PRADA* [10], but will be detected by our detection method.

Specifically, *PRADA* calculates the distribution of successive queries to distinguish whether a user is malicious or not. As illustrated in Fig 7(a), the distribution of distances between benign samples tends to follow Gaussian distribution. However, as illustrated in Fig 7(b), the distribution of distances between synthetic samples (generated by *JDBA-TR* or *Cloudleak*) apparently deviates from the Gaussian distribution. However, *PRADA* is ineffective in detecting surrogate samples such as *KnockoffNet*. Because the surrogate samples come from a natural data distribution and each sample is independent and identically distributed without correlation. As illustrated in Fig 7(c), the distributions of surrogate data *KnockoffNet* also follow Gaussian distribution and the defender can not distinguish them from the distributions of benign data.

Besides, an adaptive adversary may make its synthetic samples uncorrelated (i.e., as a natural data distribution) by involving multiple colluded malicious users, thereby evading the detection of *PRADA*. For example, we simulate the case where an adversary registers multiple user accounts and assigns the relevant (or similar) malicious queries to different users to make malicious queries sent by each colluded user unrelated (i.e., follows Gaussian distribution). The distribution of malicious queries from one of the colluded users is illustrated in Fig 8, which also follows Gaussian distribution.

On the contrary, our detection method performs independent detection for each query and is therefore more robust against these scenarios. Our experiments confirm that the detector of AMAO can still achieve the good performance presented in TABLE 6 against colluded malicious



(a) Benign data

(b) Synthetic data

(c) Surrogate data

Fig. 7: Distribution of L_2 distances for benign queries and malicious queries, where synthetic data includes samples generated by *JBDA-TR* (or *Cloudleak*) and surrogate data includes samples from *KnockoffNet* attack.

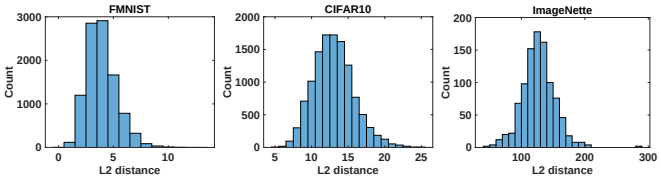


Fig. 8: Distribution of L_2 distances for *JBDA-TR* malicious queries from the colluded user.

users or surrogate samples such as *KnockoffNet*.

In addition, a malicious user may also mix malicious queries with benign queries in its queries to enhance the stealthiness of the attack. The suspect function S of the user can be reduced by employing this strategy. However, this will result in a significant increase in the adversary's query expense. For instance, we assume the detection accuracy of our detector is 80% and the threshold for S to determine whether a user is malicious is set to 0.1. The malicious user needs to mix more than 87.5% benign queries to ensure the stealthiness of the attack. Since the number of malicious queries is only 12.5% of the total number of queries, the adversary needs to spend about 10 times the number of queries to accomplish the same attack effect when employing this strategy. It is unrealistic for the adversary to have such a large number of benign samples and the model stealing attack will be unworthy with such a large query expense.

5.6.2 Aware of the adaptive query response phase

In practice, it is unrealistic for an adversary to manually distinguish which response result has been perturbed one by one. However, the adversary can distinguish whether the result is perturbed by analyzing the indistinguishability of

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3261327

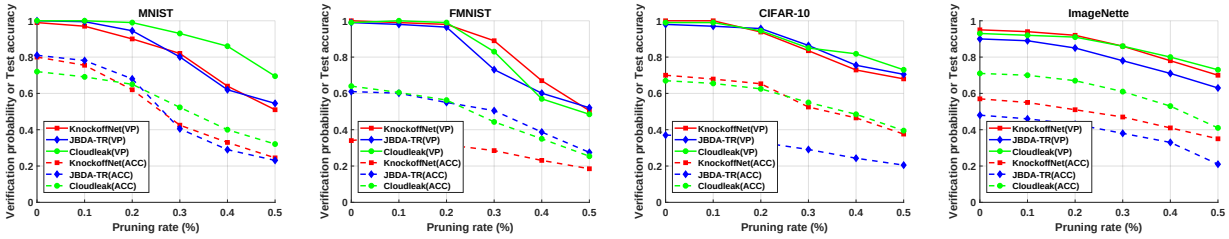


Fig. 9: Robustness of the ownership verification against model pruning

TABLE 12: Evaluations of the pHash algorithm

Model	Similarity threshold	Identification accuracy
MNIST	0.63	88.56
F-MNIST	0.66	83.83
CIFAR-10	0.64	92.03
ImageNette	0.63	95.20

the response result, i.e., whether two same (or similar) query samples yield different response results. If the adversary receives many different predictions for the same or similar query samples, it may be aware of the adaptive query response phase and discard the perturbed results. Actually, the synthetic samples of model extraction attacks (such as *JBDA-TR* and *Cloudleak*) tend to have high similarities because they are generated by adversarial attacks or other methods (such as data augmentation) from a seed dataset.

DAWN [27] used the hash value of the input to produce the random prediction and employed a mapping function to ensure this indistinguishability. However, it can not guarantee the indistinguishability in the case of data augmentation transformations [12], [29], [30]. In this work, we employ the image pHash algorithm to ensure the indistinguishability of our adaptive response strategy in the case of adversarial attacks and data augmentation.

Specifically, we first use the same method in Section 5.3 to find the optimal similarity threshold to distinguish the similar and dissimilar samples. After that, we use these similarity thresholds to determine whether an input sample is a synthetic sample generated from the recorded sample (the generation method includes *JBDA-TR*, *Cloudleak* and data augmentation [12], [29]). As presented in TABLE 12, the pHash algorithm can identify synthetic samples of the recorded sample with high accuracy. Besides, the computation overhead of the pHash algorithm is negligibly small (see TABLE 10). Thus, the defender can ensure the indistinguishability by returning the same perturbed result for the synthetic samples of the recorded sample and the adaptive adversary can not distinguish which result is perturbed.

5.6.3 Aware of the ownership verification phase

Finally, we consider the scenario where the adversary is aware of the ownership verification phase. The adversary may try to invalidate backdoors embedded in the stolen model through model compression methods, such as model pruning [59].

To evaluate the robustness of our ownership verification scheme against such an adaptive adversary, we adopt the model pruning algorithm used in [59], which trims the parameter with a small absolute value to zero. Then we calculate the verification probability (VP) of the perturbed results and test accuracy (ACC) of the substitute model with different degrees of model pruning. As shown in Fig 9, the verification probabilities are always higher than the test accuracies even when the model performance is degraded significantly due to the large pruning rate. It demonstrates that our ownership verification scheme is robust against such model pruning.

5.6.4 Robustness of AMAO against adaptive attack strategy D-DAE [58]

In this section, we evaluate the robustness of AMAO against a state-of-the-art adaptive model extraction attack strategy *D-DAE* [58], which aims to break the defenses of prediction with perturbations. Specifically, *D-DAE* includes two main modules, i.e., disruption detection and disruption recovery, where meta-learning is used in the disruption detection phase to detect disrupted results and well-designed generative models are used to restore the clean result from the disrupted result. We integrate *D-DAE* with the considered model extraction attacks and evaluate the performance of AMAO and the baseline defenses (*DAWN* [27], *Deceptive Perturbation* [25] and *Adaptive Misinformation* [23]) under this adaptive attack strategy.

The results of the test accuracy of the substitute models are presented in TABLE 13, where the results of the baseline for the hard-label scenario show the defense results of *DAWN* [27]; the results of the baseline for the soft-label scenario show the best defense results of *Deceptive Perturbation* [25] and *Adaptive Misinformation* [23]. The results indicate that the integration of *D-DAE* strategy can weaken the effectiveness of the defenses to some extent. However, AMAO can still achieve good defensive effect against the model extraction attacks integrated with *D-DAE* strategy. In comparison, AMAO can achieve better defensive effect than baseline defense regardless of whether the adversary uses *D-DAE* strategy or not.

In conclusion, the results in Section 5.6 demonstrate that AMAO is more robust than baseline defense against adaptive adversaries.

6 Limitations

It should be pointed out that we only consider *data-limited* adversary in this work, where the adversary only has a small number of private training samples. In terms of the adversary that has access to the complete private training dataset (unlabeled), it can use a semi-supervised learning

TABLE 13: The attack performance (i.e., test accuracy of the substitute model) of the model extraction attacks (without/with adaptive attack strategy *D-DAE*) under the defense of AMAO and the baseline defense.

Attack	Scenario	Dataset	Baseline defense	AMAO
JBDA-TR/ D-JBDA-TR	Hard label	MNIST	87.30/89.57	**67.21/69.32**
		F-MNIST	74.21/77.89	**65.21/68.33**
		CIFAR-10	35.35/38.15	**32.62/36.07**
		ImageNette	47.88/50.98	**45.57/48.19**
	Soft label	MNIST	91.80/93.48	**81.30/83.55**
		F-MNIST	75.48/78.20	**61.59/64.01**
		CIFAR-10	40.97/43.07	**37.16/38.99**
		ImageNette	50.70/53.06	**48.53/50.55**
KnockoffNet/ D-KnockoffNet	Hard label	MNIST	70.44/75.89	**68.03/73.24**
		F-MNIST	34.95/38.09	**32.46/35.23**
		CIFAR-10	63.48/67.24	**49.30/54.44**
		ImageNette	50.88/52.97	**44.20/47.03**
	Soft label	MNIST	80.56/85.66	**80.11/84.97**
		F-MNIST	37.81/41.20	**34.24/38.58**
		CIFAR-10	71.35/72.99	**70.04/71.67**
		ImageNette	61.45/65.07	**57.04/60.75**
Cloudleak/ D-Cloudleak	Hard label	MNIST	73.14/78.98	**67.28/75.06**
		F-MNIST	67.82/71.54	**63.64/67.30**
		CIFAR-10	67.59/72.87	**61.02/69.98**
		ImageNette	73.60/78.29	**68.04/73.80**
	Soft label	MNIST	75.93/78.50	**72.88/75.25**
		F-MNIST	71.33/74.41	**64.94/68.54**
		CIFAR-10	71.09/75.19	**67.50/71.58**
		ImageNette	78.10/82.88	**71.59/76.64**

[1] *D-JBDA-TR*, *D-KnockoffNet* and *D-Cloudleak* represent the considered model extraction attacks integrated with *D-DAE*.
[2] The attack performance of these model extraction attacks integrated with *D-DAE* under no defense is the same as the results presented in TABLE 8.

method (such as [42]) to query the victim model with unlabeled training samples and train a substitute model. Our proposed defense is less effective in this scenario.

To defend against such model extraction attacks that use original private training data (unlabeled) as malicious queries, defenders can use image matching methods [60] to identify private training data in malicious queries and detect such attacks. In addition, defenders can also use membership inference attack methods (such as *Dataset Inference* [61]) to determine if a suspicious model is trained based on the private training dataset and verify the model ownership. These methods can be also integrated into our AMAO and enable AMAO to defend against this type of attack. In the future, we intend to explore new defense methods to make the defense framework more comprehensive and able to defend against more types of model extraction attacks.

7 Conclusions

In this paper, we proposed AMAO, a comprehensive framework that has countermeasures against model extraction attacks for every stage in the model development pipeline from training, prediction to release. Specifically, it consists of four closely connected phases: adversarial training, malicious query detection, adaptive query response and ownership verification. Each phase of AMAO is designed with a competitive defense strategy, which outperforms previous work. Experiments conducted on four datasets (i.e., MNIST, F-MNIST, CIFAR-10 and ImageNette) clearly demonstrate the superiority of AMAO compare with state-of-the-art defenses in defending against model extraction attacks including *JBDA-TR*, *Cloudleak* and *KnockoffNet*. Besides, extensive experiments demonstrate AMAO is also robust against a variety of adaptive adversary scenarios.

Acknowledgment

This work is supported by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B0101360001, National Natural Science Foundation of China under Grants 62020106013, 61972454, and 61802051, Sichuan Science and Technology Program under Grants 2020JDTD0007 and 2020YFG0298, the Fundamental Research Funds for Chinese Central Universities under Grant ZYGX2020ZB027.

References

[1] Julian Salazar, Katrin Kirchhoff, and Zhiheng Huang. Self-attention networks for connectionist temporal classification in speech recognition. In *Proceedings of ICASSP*, pages 7115–7119, 2019.
[2] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of CVPR*, pages 8697–8710, 2018.
[3] M Mitchell Waldrop et al. No drivers required. *Nature*, 518(7537):20, 2015.
[4] Ekaba Bisong. An overview of google cloud platform services. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pages 7–10. 2019.
[5] Edo Liberty, Zohar Karnin, Bing Xiang, Laurence Rouesnel, Baris Coskun, Ramesh Nallapati, Julio Delgado, Amir Sadoughi, Yury Astashonok, Piali Das, et al. Elastic machine learning algorithms in amazon sagemaker. In *Proceedings of SIGMOD*, pages 731–737, 2020.
[6] Chenhui Hu and Vanja Paunic. Building forecasting solutions using open-source and azure machine learning. In *Proceedings of ACM SIGKDD*, pages 3497–3498, 2020.
[7] Rob High. The era of cognitive systems: An inside look at ibm watson and how it works. *IBM Corporation, Redbooks*, pages 1–16, 2012.
[8] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *Proceedings of USENIX Security Symposium*, pages 601–618, 2016.
[9] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *Proceedings of S&P*, pages 36–52, 2018.
[10] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *Proceedings of Euro S&P*, pages 512–527, 2019.
[11] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of CVPR*, pages 4954–4963, 2019.
[12] 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 *Proceedings of NDSS*, 2020.
[13] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish Shevade, and Vinod Ganapathy. Activethief: Model extraction using active learning and unannotated public data. In *Proceedings of AAAI*, volume 34, pages 865–872, 2020.
[14] Mingyi Zhou, Jing Wu, Yipeng Liu, Shuaicheng Liu, and Ce Zhu. Dast: Data-free substitute training for adversarial attacks. In *Proceedings of CVPR*, pages 234–243, 2020.
[15] Wenbo Jiang, Hongwei Li, Sen Liu, Yanzhi Ren, and Miao He. A flexible poisoning attack against machine learning. In *Proceedings of ICC*, pages 1–6, 2019.
[16] Wenbo Jiang, Hongwei Li, Sen Liu, Xizhao Luo, and Rongxing Lu. Poisoning and evasion attacks against deep learning algorithms in autonomous vehicles. *IEEE Transactions on Vehicular Technology*, 69(4):4439–4449, 2020.

[17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of CVPR*, pages 2574–2582, 2016.

[18] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of Asia CCS*, pages 506–519, 2017.

[19] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proceedings of S&P*, pages 3–18, 2017.

[20] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of CCS*, pages 1322–1333, 2015.

[21] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of CCS*, pages 619–633, 2018.

[22] Manish Kesarwani, Bhaskar Mukhoty, Vijay Arya, and Sameep Mehta. Model extraction warning in mlaas paradigm. In *Proceedings of ACSAC*, pages 371–380, 2018.

[23] Sanjay Kariyappa and Moinuddin K Qureshi. Defending against model stealing attacks with adaptive misinformation. In *Proceedings of CVPR*, pages 770–778, 2020.

[24] J. Grana. Perturbing inputs to prevent model stealing. In *Proceedings of IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, 2020.

[25] Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. Defending against neural network model stealing attacks using deceptive perturbations. In *Proceedings of S&P Workshops*, pages 43–49, 2019.

[26] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Utility-constrained defenses against model stealing attacks. *arXiv preprint arXiv:1906.10908*, 2019.

[27] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. Dawn: Dynamic adversarial watermarking of neural networks. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4417–4425, 2021.

[28] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. *arXiv preprint arXiv:1912.00888*, 2019.

[29] Xueluan Gong, Yanjiao Chen, Wenbin Yang, Guanghao Mei, and Qian Wang. Inversenet: Augmenting model extraction attacks with training data inversion. In *Proceedings of IJCAI*, pages 2439–2447, 2021.

[30] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. In *Proceedings of ICLR*, 2021.

[31] Jean-Baptiste Truong, Pratyush Maini, Robert J Walls, and Nicolas Papernot. Data-free model extraction. In *Proceedings of CVPR*, pages 4771–4780, 2021.

[32] Yicheng Zhang, Rozhin Yasaei, Hao Chen, Zhou Li, and Mohammad Abdullah Al Faruque. Stealing neural network structure through remote fpga side-channel analysis. *IEEE Transactions on Information Forensics and Security*, 16:4377–4388, 2021.

[33] Vasisht Duddu, Debasis Samanta, D Vijay Rao, and Valentina E Balas. Stealing neural networks via timing side channels. *arXiv preprint arXiv:1812.11720*, 2018.

[34] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[35] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J Fleet. Adversarial manipulation of deep representations. In *Proceedings of ICLR*, 2016.

[36] Model zoo: Discover open source deep learning code and pretrained models.

[37] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of AsiaCCS*, pages 159–172, 2018.

[38] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *Proceedings of USENIX Security Symposium*, pages 1615–1631, 2018.

[39] Hengrui Jia, Christopher A Choquette-Choo, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. *arXiv preprint arXiv:2002.12200*, 2020.

[40] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proceedings of S&P*, pages 707–723, 2019.

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

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

[43] Dan Wang and Yi Shang. A new active labeling method for deep learning. In *Proceedings of IJCNN*, pages 112–119, 2014.

[44] Melanie Ducoffe and Frederic Precioso. Adversarial active learning for deep networks: a margin based approach. *arXiv preprint arXiv:1802.09841*, 2018.

[45] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Proceedings of ICLR*, 2017.

[46] Chang Liu, Bo Li, Yevgeniy Vorobeychik, and Alina Oprea. Robust linear regression against training data poisoning. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 91–102. ACM, 2017.

[47] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[48] Hongpo Zhang, Ning Cheng, Yang Zhang, and Zhanbo Li. Label flipping attacks against naive bayes on spam filtering systems. *Applied Intelligence*, pages 1–12, 2021.

[49] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *Proceedings of european conference on artificial intelligence*, pages 870–875. 2012.

[50] Rahim Taheri, Reza Javidan, Mohammad Shojafar, Zahra Pooranian, Ali Miri, and Mauro Conti. On defending against label flipping attacks on malware detection systems. *Neural Computing and Applications*, 32(18):14781–14800, 2020.

[51] Albert N Shiryaev. *Optimal stopping rules*, volume 8. Springer Science & Business Media, 2007.

[52] Nasir Ahmed, T Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[53] C Zhang, S Bengio, M Hardt, B Recht, and O Vinyals. Understanding deep learning requires rethinking generalization int. In *Proceedings of ICLR*, 2017.

[54] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[55] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[56] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

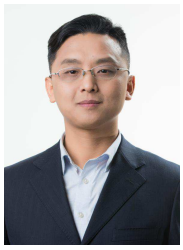
[58] Yanjiao Chen, Rui Guan, Xueluan Gong, Jianshuo Dong, and Meng Xue. D-dae: Defense-penetrating model extraction attacks. In *Proceedings of S&P*, pages 432–449, 2022.

[59] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of ICMR*, pages 269–277, 2017.

[60] Clark F Olson. Maximum-likelihood image matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):853–857, 2002.

[61] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset inference: Ownership resolution in machine learning. In *Proceedings of ICLR*, 2020.

Wenbo Jiang received his B.S. degree in information security from University of Electronic Science and Technology of China (UESTC) in 2017. Currently, he is a PhD student at the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), China. His research interests include adversarial machine learning and model extraction attacks.



Hongwei Li (M'12-SM'18) is currently the Head and a Professor at Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China. He received the Ph.D. degree from University of Electronic Science and Technology of China in June 2008. He worked as a Postdoctoral Fellow at the University of Waterloo from October 2011 to October 2012. He is the Senior Member of IEEE, the Distinguished Lecturer of IEEE Vehicular Technology Society.

Guowen Xu is currently a Research Fellow with Nanyang Technological University, Singapore. He received the PhD degree from the University of Electronic Science and Technology of China (UESTC) in 2020. As the first author, he has published more than 15 papers in reputable venues, including ACM ACSAC, ACM ASIACCS, IEEE TDSC and IEEE TIFS. He is the recipient of the Best Paper Award of the 26th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2020).

Tianwei Zhang is an assistant professor in School of Computer Science and Engineering, at Nanyang Technological University. His research focuses on computer system security. He is particularly interested in security threats and defenses in machine learning systems, autonomous systems, computer architecture and distributed systems. He received his Bachelor's degree at Peking University in 2011, and the Ph.D degree in at Princeton University in 2017.

Rongxing Lu (S'09-M'11-SM'15-F'21) is currently an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. He was awarded the most prestigious "Governor General's Gold Medal", when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. He is presently an IEEE Fellow.