# DivTheft: An Ensemble Model Stealing Attack by Divide-and-Conquer

Zhuo Ma, Xinjing Liu, Yang Liu, Ximeng Liu, *Member, IEEE*, Zhan Qin, and Kui Ren, *Fellow, IEEE*

**Abstract**—Recently, model stealing attacks are widely studied but most of them are focused on stealing a single non-discrete model, e.g., neural networks. For ensemble models, these attacks are either non-executable or suffer from intolerant performance degradation due to the complex model structure (multiple sub-models) and the discreteness possessed by the sub-model (e.g., decision trees). To overcome the bottleneck, this paper proposes a divide-and-conquer strategy called DivTheft to formulate the model stealing attack to common ensemble models by combining active learning (AL). Specifically, based on the boosting learning concept, we divide a hard ensemble model stealing task into multiple simpler ones about single sub-model stealing. Then, we adopt AL to conquer the data-free sub-model stealing task. During the process, the current AL algorithm easily causes the stolen model to be biased because of ignoring the past useful memories. Thus, DivTheft involves a newly designed uncertainty sampling scheme to filter reusable samples from the previously used ones. Experiments show that compared with the prior work, DivTheft can save almost 50% queries while ensuring a competitive agreement rate to the victim model.

**Index Terms**—Ensemble learning, model stealing/extraction attack, MLaaS, black-box attack

## 1 Introduction

Recent advances in cloud computing expedite the boom of the Machine Learning-as-a-Service (MLaaS) platform. Many AI companies leverage their rich data resources to train machine learning models and provide pay-per-query model prediction services for users, such as Google and BigML [1]. Recent reports even predict an economic boost of 14 trillion dollars in the MLaaS market [2]. The huge potential values of machine learning models lead to the proposal of a novel attack, called *model stealing* (or model extraction) [3], [4]. A malicious user can utilize the model stealing attack to steal a model that performs equivalently or near-equivalently to the remote model through a limited number of queries, thereby breaking the model privacy and causing economic losses to the MLaaS operators.

- *Zhuo Ma, Xinjing Liu, and Yang Liu are with the School of Cyber Engineering, Xidian University, Xi'an, Shaanxi 710071, China. E-mail: mazhuo@mail.xidian.edu.cn, liuxinjing_j@163.com, bcds2018@foxmail.com.*
- *Ximeng Liu is with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, Fujian 350025, China, and also with the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, Guangdong Province 518066, China. E-mail: snbnix@gmail.com.*
- *Zhan Qin and Kui Ren are with the Institute of Cyberspace Research, Zhejiang University, Hangzhou, Zhejiang 310027, China. E-mail: {qinzhan, kuiren}@zju.edu.cn.*

*Manuscript received 9 January 2022; revised 17 October 2022; accepted 15 December 2022. Date of publication 5 January 2023; date of current version 13 November 2023.*
*This work was supported in part by the National Key Research and Development Program of China under Grants 2022YFB3103500 and 2021YFB3101100, in part by the National Natural Science Foundation of China under Grants U21A20464 and 61872283, in part by the Natural Science Basic Research Program of Shaanxi under Grant 2021JC-22, and in part by CNKLSTISS, the China 111 Project No. B16037.*
*(Corresponding author: Yang Liu.)*
*Digital Object Identifier no. 10.1109/TDSC.2023.3234355*

Although there are various promising studies on model stealing attacks [3], [4], [5], [6], challenges still arise while expanding them to a wider range of scenarios.

*Ensemble Model Stealing.* The first challenge is to effectively expand the attack to common ensemble models with a lower query budget. Due to the high capability, ensemble models, especially Random Forest (RF) [7], are prevalent in many real-world applications. However, considering model stealing attacks, most of them are focused on simple *single model stealing* [4], [8], [9], like the logistic regression model. As being expanded to the ensemble model, whose output is formed by a group of sub-models, the existing attacks always suffer from a high query complexity. Moreover, common ensemble models can be obtained in varying ways [10], such as bagging, boosting and stacking. Up to now, only a few works [4] discuss how to steal the bagging based RF, but none explores a feasible way to steal the models trained in other common ensemble ways.

*Limited Observable Information.* The second challenge is to achieve model stealing while only the label outputs of the target model are given, i.e., achieving black-box model stealing. Some of the prior works assume the attacker to be capable of observing informative clues from the prediction APIs provided by MLaaS platforms, such as gradients [11], or the decision paths of trees [3]. However, as being commercialized or proprietary, many real-world systems prefer to provide more conservative prediction APIs that only output class labels, e.g., PlateRecognizer [12], to protect model privacy. Although black-box model stealing is harder to achieve, it is more practical in applications.

To overcome the challenges, we propose a *divide-and-conquer* attack called DivTheft to achieve ensemble model stealing. The key observation that motivates our methodology is that *divide-and-conquer can degrade ensemble model stealing to single model stealing*. In more detail, prior works [4], [5], [6] have explored promising methods to steal a single model effectively. Supposing that we can divide the hard problem of ensemble model stealing into multiple simple ones about single model stealing, ensemble model stealing can be simply conquered with the prior attack method.

Specifically, for "dividing", a direct way is to independently steal multiple sub-models, each expected to perform equivalently to the target model. However, referring to XGBoost [13], such a naive way fails to "copy" the internal connection between the sub-models of the target model. Thus, DivTheft introduces the boosting learning concept into the "dividing" process. Instead of stealing each sub-model independently, DivTheft estimates each newly stolen sub-model by fitting the "residual error" between the ensemble output of past extracted models and the output of the target model, e.g., voting ratio error for RF. In this way, DivTheft can make sub-model stealing only have to care about narrowing the residual error and ensure the expected convergence of the stolen model.

To conquer the task of free-data sub-model stealing, DivTheft basically follows the importance weighted active learning algorithm [14]. For active learning, the key step is to select informative samples to accelerate model training. However, in our evaluation, we observe that the common active learning technique used in model stealing [4], [15] only considers selecting samples from a fresh unlabeled data set and ignoring the previously used labeled samples. It works well in single model stealing, while such a design can cause the stolen sub-model to easily fall into local optimality and increase the required query budget. Therefore, the active learning algorithm leveraged in DivTheft involves two sample selection parts, error set selection and auxiliary set selection. The error set is responsible for selecting fresh unlabeled samples with high importance weights. Considering that DivTheft is based on QS active learning, the sample pool is quite different from the private training set in distribution. Deviation of the distribution makes some of the newly selected samples noisy and they can be 'misleading' for the stolen model training thus causing local optimum. The auxiliary set is designed to reduce such noise by filtering reusable samples from the labeled ones, so that the new sub-model can update in the same direction as the previous ones. In our work, the most efficient strategy, uncertainty sampling strategy is applied for auxiliary sample selection.

Our Contributions:

1) We propose a simple yet effective divide-and-conquer attack DivTheft for black-box ensemble model stealing with samples following an underlying distribution. DivTheft is by far the most effective model stealing attack against the algorithm family of common ensemble models.
2) We improve the current active learning technique leveraged in model stealing attacks through a newly designed auxiliary set selection algorithm, which introduces an uncertainty sampling mechanism to lower the noise of the attack sample pool and smooth the convergence of the stolen model.
3) We conduct extensive experiments on models trained with eight datasets in five ensemble ways. The results show that DivTheft can always make the stolen model achieve around 90% agreement rate to the target model while consuming only thousands of queries.

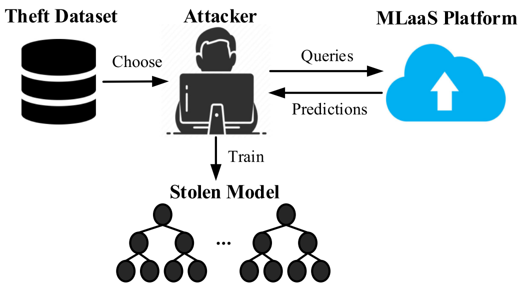


Fig. 1. Diagram of the model stealing attack. The victim model is deployed on an MLaaS Platform and the attacker trains the stolen model according to query and prediction pairs provided by the victim model.

The remainder of this paper proceeds as follows. In Section 2, we discuss related work. In Section 3, we define the model stealing attack and the capabilities of the participants. In Section 4, we describe the proposed DivTheft for ensemble model stealing attack. In Section 5, we evaluate the performance of DivTheft on varied datasets with baselines and validate the efficiency of each component. In Section 6, we conclude this work.

## 2 Related Work

In this section, we briefly introduce the elementary technical background of the model stealing attack.

### 2.1 Model Stealing Attack

The model stealing attack was first presented by [3] and became threatening with the popularity of MLaaS platforms. In model stealing attack, an attacker gains knowledge of a machine learning model (which is also called the target model), i.e., model function. In other words, the aim of an attacker is to reconstruct a model which performs similar to the target model with a limited budget of queries, as shown in Fig. 1. The difficulty of the model stealing attack depends on extracting useful information contained in the confidence vectors provided by the target model. Commonly, there are two kinds of confidence vectors: 1) actual confidence output [16]; 2) top-1 output. Assume that the confidence vector is represented as $v = (v_0, v_1, ..., v_{k-1})$. An actual confidence vector, $v_i \in [0, 1], i \in \{1, ..., k\}$ is computed by a $softmax$ function [17]. In a tree model, it is determined by sample splitting proportion during the training phase. In a top-1 output, only the index of the maximum probability in $v$ is provided by the target model.

At first, the model stealing attack was only against some simple machine learning models like Linear Regression (LR) and Decision Tree (DT) by taking advantage of abundant information provided by MLaaS servers [3]. Then, the attack is expanded to other more complicated models. Yuan [18] showed that their attack could perform well on Deep neural networks with heuristically generated synthetic data. Takemura [9] applied model extraction on Recurrent Neural Networks (RNNs) by features of RNNs. Yu [19] launches model stealing attack by adversarial examples to ensure the queries lying near the decision boundary of the target model. However, the attacker requires the prediction confidence vector to measure the distance between the queries and the decision boundary. In many cases, this requirement cannot be satisfied. Moreover, Chandrasekaran [4] demonstrated the similarity between active learning and black-box model stealing, and claimed that active learning should become a promising method to achieve model stealing without access to any public data (or training data of the target model), i.e., data-free model stealing attacks. Pal [15] used active learning strategy to select the samples with the most distant confidence vector between the stolen model and the target model.

### 2.2 Active Learning

In supervised model training, massive labeled samples should be used to train a model. To get rid of the efforts to label samples, the active learning technique is developed. Formally, active learning refers to interactions between two parties, an active learner $\mathcal{L}$ and an oracle $\mathcal{O}$. For a chosen sample $x \in X$ from the learner $\mathcal{L}$, the oracle $\mathcal{O}$ responds with a label $y \in Y$. Therefore, $\mathcal{L}$ can learn a predictor $f$ by analyzing information contained in (x, y) pairs. In order to reduce the cost of labeling, the active learner needs to apply a strategy to pick the most informative unlabeled $x$, called active learning. Generally, the common active learning approaches can be divided into two types: 1) query synthesis (QS learning); 2) PAC active learning [20].

*PAC Active Learning.* In PAC active learning, data-points are sampled from an actual underlying distribution similar to initial labeled samples. Two strategies are commonly applied for sampling, *stream-based* and *pool-based* sampling. The former samples one data-point at a time, then $\mathcal{L}$ decides whether to query. It is suitable in the scenario when obtaining data-points is cheap while labeling is expensive. While the latter selects data-points in a static pool by greedy algorithm. Different from stream-based sampling, it has to go through data-points sequentially.

*QS Active Learning.* In QS active learning, a learner can query for any sample (including arbitrary samples) in the input space, no matter it is in the same distribution of the training data used to train the target model.

Although arbitrary samples are difficult to label artificially, they can be labeled easily by automated oracles [21]. In this work, QS active learning is adopted.

For example, given a set of unlabeled data, a learner can follow stream-based sampling or pool-based sampling with a query strategy (QS or PAC active learning) which evaluates the contribution of the unlabeled sample, including uncertainty-based approach [22], [23], expected error reduction [24], diversity-based approach [25], or expected model change [26], [27] to augment data. Moreover, the use of importance weights is also a standard sample selection strategy for active learning. Beygelzimer [28] proposed importance weighted active learning scheme to address the high label complexity and non-adaptive defect of previous PAC learning algorithms.

## 3 Problem Formulation

This paper aims at designing a new attack on ensemble model stealing in the black-box scenario, like [4]. Here, we detailedly discuss the attack scenario, including a victim MLaaS platform and a malicious user as the attacker, and formulate the attack process.

*MLaaS Server.* Formally, given a hypothesis ensemble model class $\mathcal{F} : \{x \rightarrow y\}$, $\mathcal{F}$ represents model types, e.g., voting based RF. Let an MLaaS server $\mathcal{S}$ be with the knowledge of a specific $F \in \mathcal{F}$, denoted as $\mathcal{S}(F)$, where $F$ specifies the ensemble model in DivTheft. $\mathcal{S}(F)$ owns a private training set $\mathcal{D}_{train}$. These two datasets consist of a set of samples in the task domain $\mathcal{O}$. The target model $F$ is well-trained on $\mathcal{D}_{train}$ by $\mathcal{S}$. Provided with a valid input vector $x$, $F$ outputs a top-1 confidence vector as a response. The MLaaS server provides all users with a description of the valid input $x$, including the feature number $k \in \mathbb{N}$ of each input and the input range of each feature.

*Attacker.* Denote the attacker as $\mathcal{A}$. Like benign users, $\mathcal{A}$ is given a limited description of $S(F)$, including $\mathcal{F}$ and the input dimension $k$. The other model info, e.g., the data distribution of $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$, the model architecture, model parameters and hyper-parameters is hidden from $\mathcal{A}$. By fixing an error function $err(F, F^*)$, $\mathcal{A}$ outputs a model $F^*$ with at most $Q$ queries. The goal of $\mathcal{A}$ is to build $F^*$ functionally equivalent to $S(F)$ with $Q$ as small as possible. Formally, the model stealing attack $\mathcal{A}^{\varepsilon}_{\mathcal{F}}(\mathcal{S}(F), Q)$ proceeds as follows:

1) According to the description of $S(F)$, $\mathcal{A}$ generates *de novo* $\mathcal{X}$ following Gaussian distribution. Note that $\mathcal{X}$ generated by $\mathcal{A}$ is independent of the distribution of $\mathcal{D}_{train}$. Then $\mathcal{A}$ uses these samples to form an attack sample pool $\mathcal{P}$.
2) $\mathcal{A}$ uses stream-based sampling to select a sample $x_i \in \mathcal{X}$ from $\mathcal{P}$ by PAC active learning algorithm.
3) The remote server $\mathcal{S}$ gets $x_i$ from $\mathcal{A}$, and provide the response $F(x_i) = (v_0, v_1, ..., v_{k-1})$.
4) $\mathcal{A}$ updates $F^*$ and the query budget is added by 1.
5) The error function is defined as $err(F, F^*) = \frac{1}{N_{\mathcal{D}_{test}}} \sum_{(x,y) \in \mathcal{D}_{test}} d(F(x), F^*(x))$, i.e., the average error over the testing set $\mathcal{D}_{test}$, where $N_{\mathcal{D}_{test}}$ represents the number of samples in $\mathcal{D}_{test}$. We say that if $err(F, F^*)$ is less than the desired error bound $\varepsilon$, the attack is considered to be successful.

Informally, $\mathcal{A}^{\varepsilon}_{\mathcal{F}}(\mathcal{S}(F), Q)$ copies an ensemble model that approximates to $F$. $err(F, F^*)$ is tested at the end of each iteration on $\mathcal{D}_{test}$.

## 4 Ensemble Model Stealing

In this section, we take the voting based RF as an example to outline the attack workflow and then expand the attack to a broader class of ensemble models.

### 4.1 Intuition

To illustrate the idea of DivTheft, we first recall the prior way [4] to achieve black-box RF stealing. Given a target model $F$, $\mathcal{A}$ initiates a new model $F^*$ with $n$ sub-models and updates it iteratively. During this process, $\mathcal{A}$ reduces $err(F, F^*)$, which is to minimizes the probability $Pr(\cdot)$ of $F^*$ to output different results as $F$.

$$\text{minimize} \quad \Pr_{x \in \mathcal{X}}[F^*(x) \neq F(x)], \tag{1}$$

where $\mathcal{X}$ is the set of samples randomly generated by $\mathcal{A}$, filtered by active learning and labeled by querying. PAC active learning is applied for sample selection during the

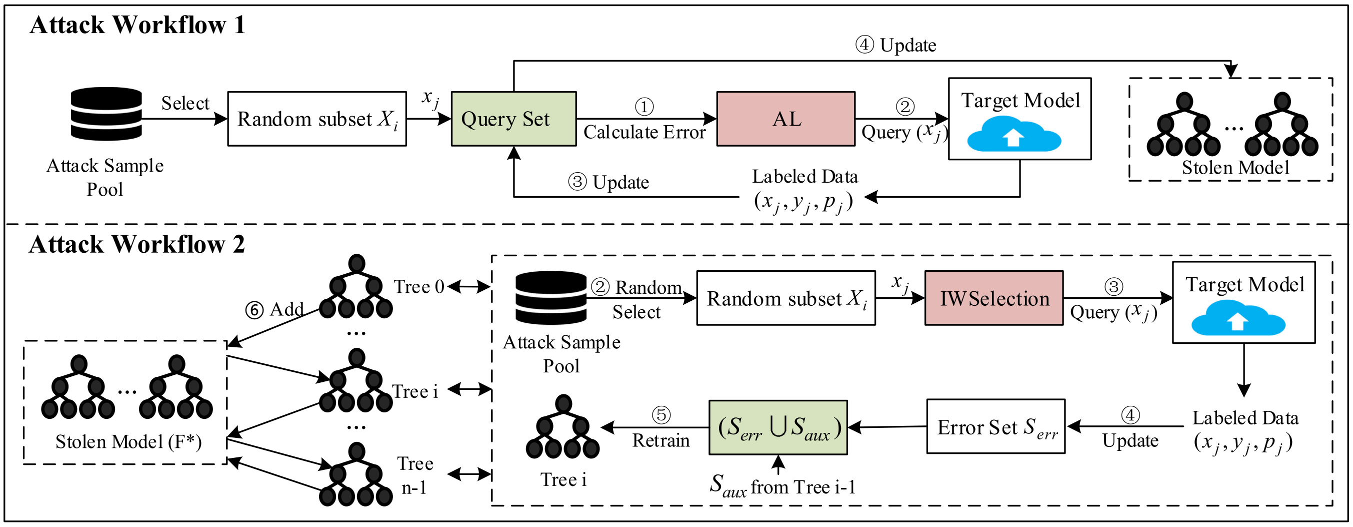


Fig. 2. The overview of prior work and DivTheft. *Attack Workflow 1* shows prior process of AL-based model stealing attack [4], [15]. *Attack Workflow 2* shows the process of DivTheft in detail. It shows the process of sub-model construction at iteration $i$. 1) the $i_{th}$ sub-model is trained on error set and co set from iteration $(i-1)_{th}$; 2) `IWSelection` calculates error and selects a sample; 3) The sample is queried and the labeled data is returned; 4) The returned prediction is added into the error set; 5) The $i_{th}$ sub-model is updated on newly added sample; 6) After checking $X_i$, the $i_{th}$ sub-model is added into the stolen model; 7) error set and auxiliary set are updated.

updating phase. Theoretically, if enough samples in $\mathcal{X}$ are queried, Eq. (1) is always resolvable [3], [4].

**Algorithm 1.** Dividing for Ensemble Model Stealing

```
Input: The target model $F$; the attacker $\mathcal{A}$; the initial attack sample pool $\mathcal{P}$; iteration number $n$;
Output: The stolen forest $F^* = \{f_0, f_1, ..., f_{n-1}\}$.
 1: To steal $F$, the attacker $\mathcal{A}$ does the following steps.
 2: Randomly split the attack pool $\mathcal{P}$ into $n$ subsets $\{X_0, ..., X_n\}$,
    where $X_i \subseteq \mathcal{P}$ and $X_i \cap X_j \neq \varnothing, 1 \leq i, j \leq n$.
 3: Initialize an error set $S_{err} = \varnothing$, an auxiliary set $S_{aux} = \varnothing$ and
    an empty tree set $F^* = \varnothing$.
 4: Train the first tree $f_0 \leftarrow train(X_0)$ by querying the target model
    to label $X_0$.
 5: for $i = 0 : n-1$ do
 6:     Conduct attack sample selection $(F^*, S^i_{err}, S^i_{aux}) \leftarrow$
        IWSelection$(f_i, F^*, X_i, X^*_i, S_{aux}, F)$.
 7:     Update $S_{err} \leftarrow S_{err} \cup S^i_{err}$ and $S_{aux} \leftarrow S^i_{aux}$.
 8:     Randomly select subsets $S^*_{err} \subseteq S_{err}$ and $S^*_{aux} \subseteq S_{aux}$, and
        then, update $X^*_{i+1} \leftarrow S^*_{err} \cup S^*_{aux}$.
 9:     Train $f_{i+1} \leftarrow train(X^*_{i+1})$ based on Eq. (2).
10: end for
```

Considering the characteristics of PAC active learning, the above method encounters high computing complexity of $\mathcal{A}$. PAC active learning requires $\mathcal{A}$ to exert modification on $F^*$ at each interation, so that the prediction of sample $x_i$ can be flipped by $F^*$ from -1 to +1, which means combinations of modification on any $m$ sub-models should be considered. In addition, flipping labels can be much more complex towards $t$ $(t > 2)$ classification tasks, because the prediction need to be flipped from 0 to {+1, +2, ..., +(t-1)}. In addition, at each iteration, samples in the query set are newly selected to reduce $err(F, F^*)$, which causes an increase in the query complexity. From the above, such a scheme increases computing and query complexity, which has lower practicality in real-world applications.

Intuitively, a solution to deal with the issue is to divide the hard task about whole RF stealing into simple ones about single tree stealing, and conquer them one by one, i.e., *dividing and conquering*, as shown in Fig. 2.

## 4.2 Dividing

For dividing, a direct method is to separate the stealing problem into independent problems about single tree stealing. Assume the target model is a voting based RF $F$ with $n$ sub-models. Given a specific sample $x_0$, at least $\lfloor \frac{n}{2} + 1 \rfloor$ trees vote 1 for $x_0$. For an attacker, the best strategy is to allocate the goal of every stolen tree to be classifying $x_0$ into 1. Clearly, such a setting ignores the internal details of $F$ about the number of trees that classify $x_0$ into 1, which is against the model stealing goal to make the stolen model perform "equivalently" to the target model.

Due to the observation, DivTheft makes the following improvements on the dividing process. First, DivTheft applies the boosting concept to realize the model stealing attack, in which new stolen sub-models are sequentially trained with the goal of reducing the "residual error" caused by the previously stolen ones. In this way, if the past stolen sub-models behavior diverges from the target model, the latter stolen submodels can still narrow the bias, and thus, the internal details of the target model can be kept as much as possible. Second, as suggested by [4], DivTheft adopts importance weighted active learning (IWAL) [14], [29] to achieve the conquering task about single model stealing (details discussed in the next section). According to the traditional IWAL, the samples are all selected freshly to reduce the residual error $err(F, F^*)$ between the stolen model and the target model at each iteration. While for ensemble model stealing, ignorance of the past samples causes a significant increase in the query budget. What's worse, in our evaluation, such a design of IWAL makes it easy for the stolen model to fall into local optimality, thus lowering the performance of the stolen model. To overcome this issue, DivTheft adds the consideration of previously used samples (i.e., the auxiliary set) during the dividing process (line 6-8, Algorithm 1). Above all, we can rewrite Eq. (2) for RF stealing as follows.

$$\begin{array}{ll} \text{minimize} & F(x) - \frac{1}{i}\sum_{j=0}^{i} 1_{f_j^*(x)=\varphi(F(x))}, \\ \text{such that} & \mathcal{X} \leftarrow S_{err} \cup S_{aux}, \quad x \in \mathcal{X}. \end{array} \tag{2}$$

where $\varphi(\cdot)$ transforms the soft label outputs of $F$ to the hard label space, $S_{err}$ and $S_{aux}$ are the error set and auxiliary set selected by `IWSelection` (line 7, Algorithm 1), respectively.

## 4.3 Conquering

DivTheft leverages IWAL, the key step of which is importance weighted sample selection, to "conquer" the data-free sub-model stealing task and lower the required query budget. As illustrated in Algorithm 2, two parts comprise our weighted sample selection algorithm `IWSelection`, namely error set selection and auxiliary set selection.

*Error Set Selection.* Given a batch of data $X_i$ for the $i_{th}$ sub-model stealing, $\mathcal{A}$ first filters fresh informative samples from $X_i$ to form the error set by referring to the original IWAL scheme [29].

Specifically, denote the $i_{th}$ stolen sub-model to be $f_i$. With $S_{err}^i$, $\mathcal{A}$ first computes its weight $e$ on $f_i$.

$$e = err(f_i, S_{err}^i) = \frac{1}{|S_{err}^i|}\sum_{(x_j,y_j,p_j)\in S_{err}^i} \frac{1}{p_j} 1_{f_i(x_j)\neq y_j},$$

where $err(\cdot)$ is the importance weighted empirical error (IWER) function [14], $S_{err}^i$ is the error set selected at the $i_{th}$ iteration, $|S_{err}^i|$ is the number of samples in $S_{err}^i$, and $p_j$ is the probability of sample $x_j$ to be queried, which can be computed by Eq. (3). Then, while predicting $x_j \in X_i$ with $f_i$, $\mathcal{A}$ locates the leaf node that $x_j$ lies on. Assume the label space of the learning task to be $\mathcal{O}$. $\mathcal{A}$ forcibly modifies the output of the located leaf node to be any $o \neq f_i(x_j)$ and $o \in \mathcal{O}$, and thus, can obtain a candidate set of modified sub-models $\mathcal{F}^*$ satisfying $f_i^*(x_j) \neq f_i(x_j)$, $f_i^* \in \mathcal{F}^*$. Subsequently, $\mathcal{A}$ can compute another IWER for these modified models.

$$e^* \leftarrow \arg\min\{err(f_j^*, S_{err}^i) : f_j^* \in \mathcal{F}^* \wedge f_j^*(x_j) \neq f_i(x_j)\}.$$

Next, a coin is flipped with probability $p_j$ to determine whether $x_j$ is added into $S_{err}^i$.

$$p_j = \begin{cases} 1 & \text{if } (e^* - e) \leq \sqrt{\frac{c_0 log(j)}{j-1}} + \frac{c_0 log(j)}{j-1}, \\ w_j, & \text{otherwise} \end{cases} \tag{3}$$

where $w_j$ is the positive solution to the following equation.

$$e^* - e = \left(\frac{c_1}{\sqrt{w_j} - c_1 + 1}\right)\cdot\sqrt{C_j} + \left(\frac{c_2}{\sqrt{w_j} - c_2 + 1}\right)\cdot C_j.$$

$c_0$, $c_1$ and $c_2$ are all hyperparameters that are usually set to be 1 [14]. Finally, if $x_j$ is selected, $f_i$ is updated with $x_j$ for the next sample selection.

*Auxiliary Set Selection.* To maintain the reusable memory of past data, the model stealing of DivTheft adds an auxiliary set whose selection is based on sampling with replacement mechanism (SWRM) [30]. However, different from normal SWRM, the sampling of DivTheft introduces an uncertainty sampling mechanism to improve the quality of selected data and expedite the attack process. To evaluate the degree of uncertainty, DivTheft adopts an intuitive idea based on entropy maximization. Loosely speaking, the samples that attain lower similarity (correctly classified by fewer sub-models) to the target model have a higher level of entropy, and should be given a higher probability to be reconsidered in the next iteration. Formally, the weighted SWRM probability $q$ is computed by Eq. (4).

$$q = 1 - \frac{1}{N}\sum_{f\in F^*} 1_{f^*(x_k)=y_k}, \tag{4}$$

where $N$ is the number of sub-models in $F^*$. Again, a coin is flipped to determine whether the sample $x_k$ is added in $S_{aux}^i$ and used for the next sub-model stealing. In our evaluation, we will demonstrate that such a rule can effectively improve the performance of ensemble model stealing (see Fig. 4).

## 4.4 Extension Discussion

In this section, we discuss how to expand DivTheft to other common types of ensemble models. Besides bagging-based ensemble mentioned above, there are two more common methods for model ensemble, boosting and stacking. In these two ensemble methods, the sub-models are trained sequentially. For boosting-based ensemble, the $i_{th}$ sub-model is developed based on the former $i-1$ sub-models. Each sub-model aims to reduce bias caused by previous ones, such as boosting [13] and Adaboost [31]. Therefore, based on the characteristic, Eq. (2) should be rewritten as follows.

$$\begin{array}{ll} \text{minimize} & \mathcal{L}(f_i(x) + F_{i-1}^*(x), F(x)), \\ \text{such that} & x \in S_{err} \cup S_{aux}, \end{array} \tag{5}$$

where $\mathcal{L}$ is a loss function to evaluate the difference between the predictions of $F^*$ and $F$. It should be noted that, sub-models can be trained with different weights in adaptive boosting-based ensemble like Adaboost [31]. In this case, the attacker has no prior knowledge of the sub-model weights, but Eq. (5) can still be solved by giving the same weight to all sub-models. For stacking-based ensemble, the attacker similarly gives sub-models the same weight and trains $F^*$ iteratively.

Moreover, since leaf node labels cannot be simply flipped (line 5, Algorithm 2) in the boosting and stacking modes, we utilize another active learning strategy, query-by-committee (QBC) [32], in `IWSelection` to form $S_{err}$. With QBC, the attack samples $S_{err}$ and $S_{aux}$ are simply partitioned into multiple subsets through sampling with replacement, each of which is leveraged to train a candidate model (line 5, Algorithm 2). Other steps are the same as before.

Further, referring to our algorithms, the feasibility of DivTheft does not rely on the sub-model type. Thus, all the above methods can be applied to any other type of sub-model, e.g., SVM, without any modification.

## 5 Experiments

In this section, we conduct extensive experiments to evaluate the performance of DivTheft. First, the experimental settings are listed in Section 5.1. Then, in Section 5.2, we evaluate the effectiveness of DivTheft by comparing with prior work, and consider the attack over varied ensemble models to show the robustness of DivTheft. At last, ablation studies are conducted to to validate the improvements of the sub-algorithms proposed in DivTheft (5.3.1, 5.3.2 and 5.3.3), and the influence of query size, target model accuracy and SWRM algorithm are evaluated in 5.3.4, 5.3.5, 5.3.7, respectively. The convergence curves of DivTheft and prior work are studied in 5.3.6.

## 5.1 Experimental Settings

### 5.1.1 Datasets

To comprehensively evaluate the effectiveness of DivTheft over different learning tasks, we collect a bunch of datasets from the scikit-learn library, the UCI machine learning repository[1] and the Kaggle website[2]. Eight kaggle datasets are involved in the experiments, varying from textual (IMDB, Fake News, Spam), image (MNIST) to multivariate (Iris, Adult, Steak, GSSHappiness) datasets [33]. The datasets (IMDB, Fake News, Spam, Steak and GSSHappiness) that do not contain a testing set natively are shuffled and divided into the training set and testing set with the proportion of 0.8 and 0.2. The description of the eight datasets is as follows.

- *IMDB.* The IMDB dataset is a binary sentiment classification (positive or negative) dataset that includes 50K movie reviews, half of which are positive and the others are negative.
- *Fake News.* The Fake News dataset contains 44898 pieces of news, 23481 for reliable news and 21417 for unreliable news.
- *Spam.* The Spam dataset is a set of 5574 SMS messages tagged to be ham (legitimate) or spam, given the presence of certain words in its content.
- *MNIST.* The MNIST dataset contains 60K training and 10K testing samples, which are 0-9 handwritten digital pictures. Each picture is $28 \times 28$ in size, with pixel values varying from 0 to 255.
- *Iris.* The Iris dataset contains 150 data samples in 3 classes, each with 50 samples and 4 attributes.
- *Adult.* The Adult dataset is about predicting whether the annual income exceeds 50K$, and contains 48842 records (32561 for training and 16281 for testing) with 8 discrete features and 6 numerical continuous features. Moreover, 23.93% samples of the dataset are labeled to be positive and the remaining are labeled to be negative.
- *Steak.* The Steak dataset is a binary classification dataset that contains 549 samples with 13 discrete features.
- *GSSHappiness.* The GSSHappiness dataset contains 51020 samples in 3 classes with 6 discrete features and 3 numerical features.

### 5.1.2 Baselines

In this work, we consider the black-box model stealing of ensemble models. Hardware-based attack like side-channel monitoring [34] is out of scope. Studies on model stealing have covered varied model types [3], [6], [15], [19], [35], [36], [37], [38], while they cannot be directly applied to achieve ensemble model stealing in the black-box setting. [3] takes advantage of the additional information in the MLaaS prediction API, such as the decision path of decision tree, which breaks the black-box setting. ActiveThief [15] applies active learning to reduce the query budget. However, some previous works, the attacker can access the training set or decision path [3] or the hardware [34], which breaks our black-box setting.

**Algorithm 2.** Conquering with Active Learning

```
Input: The sub-model $f_i$; the stolen ensemble model $F^*$; a batch
    of unlabeled samples $X_i$, and the labeled samples $X_i^*$;
    the auxiliary set $S_{aux}$; the target model $F$.
Output: The updated stolen model $F^*$, the $i_{th}$ error set $S_{err}^i$ and
    the $i_{th}$ auxiliary set $S_{aux}^i$.
1: The attacker $\mathcal{A}$ initializes a new pair of error and auxiliary
   sets $S_{err}^i = \varnothing$ and $S_{aux}^i = \varnothing$.
2: $\mathcal{A}$ sets $f_0 = f_i$ and does the following steps.
3: for each $x_j \in X_i$ do
4:    Compute $e \leftarrow err(f_i, S_{err}^i)$.
5:    Modify the leaf node of $f_i$ to enumerate candidate models
      $f_i^*(x_j) \neq f_i(x_j)$ and compute $e^* \leftarrow \arg\min\{err(f_i^*, S_{err}^i) :
      f_i^*(x_j) \neq f_i(x_j)\}$.
6:    if $(e^* - e) \leq \sqrt{\frac{c_0 log(j)}{j-1}} + \frac{c_0 log(j)}{j-1}$ then
7:       Update $S_{err}^i \leftarrow S_{err}^i \cup (x_j, y_j, p_j)$ by querying the target
         model $y_j \leftarrow F(x_j)$.
8:    else
9:       Flip a coin with probability $p_j$. If it is headed, do Step 7,
         otherwise, do nothing.
10:   end If
11:   Retrain $f_i \leftarrow train(X_i^* \cup S_{err}^i)$ based on Eq. (2).
12: end For
13: Update $F^* \leftarrow F^* \cup f_i$.
14: For each $(x_k, y_k)$ in $S_{aux}^i \cup S_{err}^i$ do
15:    Compute the selection probability for correctly classified
       sample $q = \frac{1}{N}(N - \sum_{f \in F^*} 1_{f(x_k)=y_k})$ where $N$ is number
       of sub-models in $F^*$.
16:    Flip a coin with probability $q$. If the coin is headed, add
       $S_{aux}^i \leftarrow S_{aux}^i \cup x_k$; otherwise, do nothing.
17: end For
```

In some previous works, the attacker Tramèr et al. [3] considers the attacker to be able to access the decision path of each decision tree, which we think to be an unfair setting compared with ExpAL and our work. Besides, most of the prior works are about neural networks stealing, and cannot be applied on discrete models, e.g., RF. To our best knowledge, the only existing work that involves ensemble model stealing via top-1 output predictions is the active learning-based scheme proposed in [4] (abbreviated as ExpAL). Therefore, we mainly choose ExpAL as the baseline in our experiments. Moreover, we implement ExpAL for RF stealing as suggested by [4], i.e., querying with randomly generated samples and treating the whole forest as one model as conducting the label-flipping operation.

### 5.1.3 Data Generation

During the attack process, the attacker needs to randomly generate an attack sample pool $\mathcal{P}$ for querying. In our experiments, we synthesize the attack samples by using a

1. https://archive.ics.uci.edu/ml/index.php
2. https://www.kaggle.com/datasets

TABLE 1
Target Model Settings

| Dataset | Bagging (RF) | | XGBoost | | | AdaBoost | | Stacking 1 | Stacking 2 |
|---|---|---|---|---|---|---|---|---|---|
| | estimators | criterion | booster | max_depth | estimators | algorithm | estimators | base models | base models |
| **Adult** | 300 | Entropy | Gbtree | 10 | 100 | SAMME | 100 | DT, SVM, KNN | RF, AdaBoost, KNN |
| **Iris** | 100 | Entropy | Gbtree | 6 | 100 | SAMME | 100 | DT, SVM, KNN | RF, AdaBoost, KNN |
| **GSS** | 700 | Gini | Gbtree | 10 | 300 | SAMME | 300 | DT, SVM, KNN | RF, AdaBoost, KNN |
| **Steak** | 200 | Entropy | Gbtree | 10 | 100 | SAMME | 100 | DT, SVM, KNN | RF, AdaBoost, KNN |
| **MNIST** | 100 | Gini | Gbtree | 7 | 150 | SAMME | 150 | DT, SVM, KNN | RF, AdaBoost, KNN |
| **Fake** | 200 | Gini | Gbtree | 6 | 70 | SAMME | 70 | DT, SVM, KNN | RF, AdaBoost, KNN |
| **IMDB** | 800 | Gini | Gbtree | 20 | 300 | SAMME | 300 | DT, SVM, KNN | RF, AdaBoost, KNN |
| **Spam** | 100 | Gini | Gbtree | 6 | 100 | SAMME | 100 | DT, SVM, KNN | RF, AdaBoost, KNN |

pseudo-random number generator $\mathcal{G}$ that follows the Gaussian distribution.

- *Image datasets*. For the image datasets like MNIST ($28 \times 28$ in size and pixel values in a range of 0-255), we directly use $\mathcal{G}$ to generate $28 \times 28$ random values from 0-255 to serve the attack sample.
- *Multivariate datasets*. For multivariate datasets, we assume the meta info (including input dimension and the range of features) about each feature to be known to the attacker, and thus, the attacker can simply generate random attack samples according to the info.
- *Textual datasets*. Random text can be achieved in a similar way to the multivariate data by using the toolkit NLTK[3]. Take IMDB as an example, word bag of Spam is applied to generate random sentences by NLTK. Other details can be found in [4].

### 5.1.4 Ensemble Types

We experiment with the three most commonly used model ensemble methods, which are bagging, boosting, and stacking. Also, the experiments about each type of ensemble method involve several typical model types. For bagging, we experiment with the classic RF model. For boosting, the recently proposed XGBoost and AdaBoost models are considered. For stacking, we choose two types of models. The first one (abbreviated as Stacking1) chooses DT, SVM and K-Means as base estimators and LR as the final model. The second one (abbreviated as Stacking2) chooses RF, AdaBoost and K-means as base estimators and LR as the final model. The detailed setting about the ensemble ways is given in Table 1.

### 5.1.5 Deployment

The experiments are all simulated on a workstation equipped with Tesla V100 GPU and 16GB VRAM. In the simulation experiments, the attacker can only get access to the outputs of the target model after providing a valid query like the real-world applications. Especially, to fairly evaluate the performance of the stolen model, we assume the attacker to be able to access the testing set for evaluation but cannot use the testing set for launching attacks. The virtual server and the attacker are set up locally. The indicators (agreement, accuracy and queries, which will be introduced later) are tested and can evaluate the scheme the same as real-world applications. The target models are owned by the server, which are constructed locally according to Table 1. And the attacker is authenticated to get the final result of classification according to its input.

### 5.1.6 Indicators

Similar to [15], [39], we mainly record three indicators for evaluating the performance of model extraction attacks, all of which are defined below:

- *Accuracy*. This common indicator shows the percentage of samples in testing set $\mathcal{D}_{test}$ that are correctly predicted by the stolen model, which is represented as:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

  TP is true positive, FP is false positive, TN is true negative and FN is false negative.
- *Queries*. Another significant indicator, the number of queries, denotes how many times the attacker queries for the victim model during launching attacks. When the $err(F, F^*) < \varepsilon$, indicator queries stops increasing. The less query number means that the attacker can save more costs and time for launching an attack and make it harder to be identified by a defender.
- *Agreement*. Besides *accuracy* (Acc) and *queries*, another indicator is the *agreement rate* (Agree) between the the stolen model $F^*$ and the target model $F$. This indicator indicates the similarity between the outputs of the stolen model $F^*$ and the victim model $F$, which is computed by:

$$agreement = \frac{1}{N_t} \sum_{x_i \in X_{test}} 1_{F(x_i)=F^*(x_i)}. \tag{7}$$

  $N_t$ is the number of samples in the testing set $\mathcal{D}_{test}$.

### 5.1.7 Others

The three hyperparameters $c_0$, $c_1$ and $c_2$ that determine the coin flipping probability in Algorithm 2 are all set to be 1 as suggested by [14]. The default ratio between the numbers of error set $S_{err}$ and auxiliary set $S_{aux}$ is set to be 1.0. All experimental results are averaged over five trials. Moreover, for fairness, the initial attack samples for both DivTheft and ExpAL

3. https://www.nltk.org/

TABLE 2
Comparison with ExpAL on RF

| Dataset | Target | Acc | | Agree | | Queries | |
|---|---|---|---|---|---|---|---|
| | | ExpAL | Ours | ExpAL | Ours | ExpAL | Ours |
| **Adult** | 86.2 | 75.3 | **80.7** | 79.1 | **85.5** | 890 | **810** |
| **Iris** | 99.0 | **98.9** | **98.9** | **99.9** | **99.9** | 360 | **270** |
| **GSS** | 59.2 | 59.5 | **60.0** | 74.1 | **75.3** | 1940 | **450** |
| **Steak** | 57.2 | 61.8 | **62.7** | 77.3 | **80.0** | 950 | **900** |
| **MNIST** | 96.7 | 94.8 | **95.3** | 96.6 | **96.7** | 23000 | **21800** |
| **Fake** | 99.3 | 96.5 | **97.5** | 97.8 | **99.2** | 550 | **450** |
| **IMDB** | 86.4 | 60.8 | **79.8** | 70.0 | **82.9** | **6800** | 18500 |
| **Spam** | 97.4 | 94.7 | **95.9** | 97.2 | **98.0** | 2410 | **1700** |

are randomly generated from the same range with the same seed. Our prototype attack codes are publicly available[4].

## 5.2 Attack Effectiveness

### 5.2.1 Comparison With State-of-the-Art

The attack effectiveness is mainly measured by three indicators, namely *agreement*, *accuracy* and *queries*, which are defined before. To validate the effectiveness of DivTheft, we first compare the performance of DivTheft with ExpAL while attacking the same kind of model. Notably, according to [4], ExpAL is designed for stealing the RF model and non-executable for other ensemble models. Thus, the metrics in Table 2 are all measured with RF models. Moreover, in Table 2, the first column lists the datasets involved in the experiments. Column 2 is the accuracy that the target model can achieve over the testing set. Columns 3-5 show the three indicators of our work and ExpAL, respectively. The bold values indicate a better performance, e.g., higher accuracy, agreement and a lower number of queries.

Generally, the model stolen by DivTheft attains close to, or even a little higher, accuracy than the target model, and obtains a significant advantage over ExpAL on all three indicators. Moreover, considering the experiments about multivariate datasets, it can be observed that the increased complexity of the learning task (increased feature numbers) further magnifies the advantage. In more detail, for IMDB (Row 7), our accuracy and agreement are 19.0% and 12.9% higher than ExpAL with fewer queries. For Adult (Row 1), our agreement is only 6.4% higher than baseline with a slight increase in accuracy of 5.4%. This is because with more features, the model stealing problem becomes harder to be resolved (more parameters required to be stolen), and thus, the gap between DivTheft and ExpAL is magnified due to the increased attack ability of DivTheft. Interestingly, we notice that the accuracy of the stolen model is a little higher than that of the target model sometimes. Especially, the phenomenon can be easier to occur when there are not enough samples in the original training set and the learning task is not complicated. This may be due to the fact that the actively selected data can sometimes provide more informative samples than the ones in the training set.

### 5.2.2 Extension to Other Ensemble Types

Then, we conduct extended experiments on the scenarios of stealing the other two common types of ensemble models, boosting and stacking. Note that as above-mentioned, we choose a different loss function in Algorithm 1 to adapt to the scenarios of boosting and stacking model stealing. Table 3 shows the results about boosting and stacking based ensemble model stealing. The results hint that DivTheft is robust to ensemble methods used to train the target model. No matter which kind of ensemble way is chosen, the stolen model can always achieve a negligible accuracy loss (mostly less than 4%) on accuracy and a high agreement rate compared with the target model. Meanwhile, in most conditions, only thousands of queries are required. Furthermore, Fig. 3 shows the detailed curves of the stolen with increased queries. Clearly, the stolen models always converge towards the correct direction (to the target model) steadily. The fact validates that DivTheft is an effective and robust ensemble model stealing method.

## 5.3 Further Understanding

Besides the divide-and-conquer strategy, DivTheft claims to achieve three points of improvements on model stealing attacks, which are the boosting based dividing strategy, the introduction of active learning, i.e., `IWSelection`, and the proposal of auxiliary sample selection (line 13-17, Algorithm 2). To further understand these improvements, we perform an ablation study by recording the performance change of DivTheft before and after removing the above improvements. For brevity, the following experiments are only conducted on RF model by default.

### 5.3.1 Effect of Boosting

To validate the effectiveness of the boosting based dividing strategy (named with Boost in Table 4), we realize Boost along with the Non-Boost strategy mentioned before for comparison. From the results, we can observe that on average, the application of boosting improves the accuracy and the agreement rate of the stolen model by 4.3% and 2.7%, while saving 21.13% queries. The fact verifies that the introduction of Boost can significantly improve the attack effect of model stealing. Moreover, as claimed before, Boost can retain more internal details of the target model than Non-Boost, which makes the model stolen with Boost to be able to perform more "equivalently" to the target model. Intuitively, the equivalence can be reflected in the distance between the prediction scores of the stolen model and the target model. Therefore, to evaluate the equivalence, we plot the prediction points of the target model and the models stolen by Boost and Non-Boost as shown in Fig. 5. The prediction point coordinates in the figure are computed by leveraging the principal component analysis (PCA) algorithm to transform all prediction scores into two-dimension values. From the scatter diagrams in Fig. 5, we can notice that compared with Non-Boosting, more points of Boost are closer to the ones of the target model, which further proves the better performance of Boost than Non-Boost.

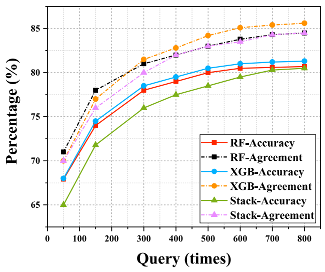
### 5.3.2 Effect of `IWSelection`

Then, this part of the experiment aims to find out how `IWSelection` contributes to the performance of the stolen model. Thus, we implement DivTheft for stealing RF models with and without the `IWSelection` All points are

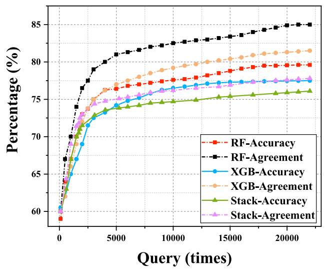
4. https://github.com/Yailor/Ensemble-Model-Stealing.

TABLE 3
The Performance of DivTheft on Boosting and Stacking Models

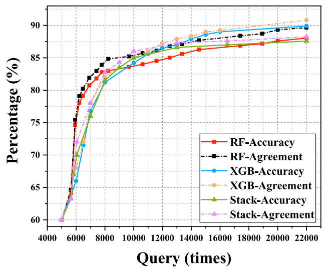
| Dataset | XGBoost | | | | AdaBoost | | | | Stacking 1 | | | | Stacking 2 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Target | Ours | | | Target | Ours | | | Target | Ours | | | Target | Ours | | |
| | | Acc | Agree | Queries | | Acc | Agree | Queries | | Acc | Agree | Queries | | Acc | Agree | Queries |
| **Adult** | 86.7 | 83.3 | 85.6 | 3200 | 85.3 | 82.2 | 83 | 2240 | 87 | 82.7 | 97.4 | 550 | 79.4 | 77.7 | 96.5 | 580 |
| **Iris** | 96.7 | 93.3 | 100 | 348 | 99.6 | 99.6 | 99.9 | 300 | 96.7 | 93.3 | 96.7 | 420 | 96.7 | 93.3 | 96.7 | 270 |
| **GSS** | 61.1 | 61.1 | 78.4 | 4100 | 56.3 | 60.3 | 70.7 | 3220 | 61.5 | 60.0 | 76.9 | 1040 | 61.2 | 60.1 | 76 | 1270 |
| **Steak** | 52.7 | 50 | 79.5 | 7300 | 51.4 | 50.5 | 78.6 | 8100 | 52.7 | 50.0 | 61 | 2550 | 50.1 | 50.1 | 65 | 600 |
| **MNIST** | 96.9 | 93.1 | 89.9 | 17210 | 97.6 | 92.9 | 89.4 | 24400 | 92.5 | 87.8 | 89.1 | 21000 | 92.8 | 88.6 | 89.5 | 21300 |
| **Fake** | 99.8 | 99.4 | 99.6 | 350 | 99.8 | 99.4 | 99.4 | 270 | 99.8 | 99.5 | 99.4 | 560 | 99.8 | 99.4 | 99.5 | 570 |
| **IMDB** | 87.3 | 83.5 | 84.8 | 21000 | 85.4 | 80.2 | 75.3 | 22300 | 89.2 | 84.4 | 86.4 | 22300 | 89.2 | 83.3 | 86.2 | 21500 |
| **Spam** | 97.5 | 93.5 | 94.1 | 1600 | 98.1 | 93.8 | 92.5 | 2340 | 98.1 | 93 | 95.9 | 2330 | 98.4 | 93.8 | 94.1 | 2400 |



(a) The performance on different stolen ensemble models on Adult.



(b) The performance on different stolen ensemble models on IMDB.



(c) The performance on different stolen ensemble models on MNIST.

Fig. 3. The detailed convergence curves of models stolen by DivTheft on Adult, IMDB and MNIST datasets.

determined by using PCA to transform the prediction scores into two-dimension points. For brevity, we only use the first three classes of MNIST (i.e., Class 0, 1 and 2) in the scatter diagram. The points of a better method tend to be closer to the points of the target model. step (line 7, Algorithm 1), as shown in Fig. 4. In the table, the attack without the `IWSelection` step is abbreviated as Non-IWSelection for brevity. Non-IWSelection specifies that all samples in $X_i$ are randomly generated and without the filtering of active learning. It can be observed that the attack with `IWSelection` gains an obvious advantage over Non-IWSelection. Take MNIST as an example. `IWSelection` successfully achieves around 37% higher accuracy and agreement than Non-IWSelection while saving 3000 queries.

### 5.3.3 Effect of Auxiliary Set

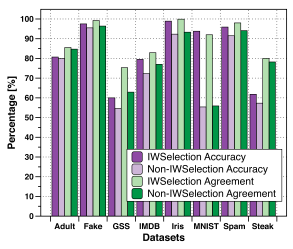
Finally, to validate the effect of the auxiliary set, we implement DivTheft with and without the auxiliary set (Aux and Non-Aux in Fig. 4). Again, the RF model is taken as the experimental subject, and the experimental results are given in Table 6. Not surprisingly, the addition of the auxiliary set helps DivTheft to obtain an improvement on the attack performance. For example, compared with Non-Aux, Aux improves the accuracy of the stolen model by 4.2% and the agreement by 4.5% while maintaining a similar level of queries as processing the textual dataset IMDB. Further, we can notice that from the perspective of the three indicators, the improvement of Aux is not as obvious as `IWSelection`. Furthermore, the auxiliary set counts a lot for the stolen model to smooth the converge of the stolen model during the attack process. This is because the auxiliary set retains memories previously stolen from the target model, and avoids the stolen model being trapped into the local optimality of freshly added samples. To better verify the statement, Fig. 6 plots the detailed accuracy and agreement changes of the stolen model along with the increased query number. As shown in the figures, all curves about Non-Aux dramatically fluctuate during the attack, and also, show a lower level of accuracy and agreement. In contrast, the curves of Aux grow steadily and positively with the increased query number.

TABLE 4
Effect of Boosting

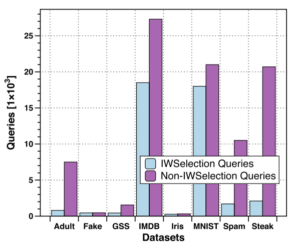
| Dataset | Acc | | Agree | | Queries | |
|---|---|---|---|---|---|---|
| | Boost | Non-Boost | Boost | Non-Boost | Boost | Non-Boost |
| **Adult** | 81.7 | 79.2 | 86.2 | 82.6 | 900 | 1300 |
| **Iris** | 98.9 | 96.7 | 99.9 | 99.9 | 250 | 310 |
| **GSS** | 60.0 | 57.1 | 75.3 | 70.6 | 437 | 510 |
| **Steak** | 61.8 | 50.0 | 80.0 | 73.7 | 2100 | 3000 |
| **MNIST** | 93.8 | 84.8 | 92.0 | 86.4 | 18000 | 10200 |
| **Fake** | 99.0 | 97.5 | 99.2 | 98.5 | 270 | 470 |
| **IMDB** | 79.5 | 62.0 | 82.9 | 63.7 | 18500 | 19000 |
| **Spam** | 95.9 | 91.3 | 98.0 | 93.4 | 1700 | 1830 |

### 5.3.4 Impact of Query Size

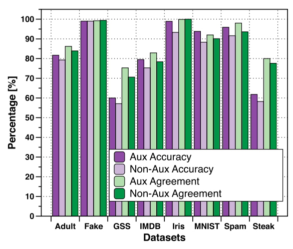
The attacker has to form a query set at each iteration, and the selected samples are used to train a new sub-model. A more appropriate size of $X_i$ can derive a better trade-off between the attack effect and the query costs, so that, additional experiments are conducted to discuss how the size of $X_i$ affects the performance of DivTheft. We realize model stealing attacks on three kinds of target models:



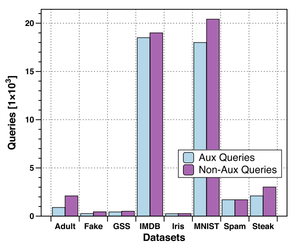
(a) The accuracy and agreement of the stolen model with and without IWSelection.



(b) The queries required to launch attacks with and without IWSelection.



(c) The accuracy and agreement of the stolen model with and without the auxiliary set.



(d) The queries required to launch attacks with and without the auxiliary set.

Fig. 4. The improvement analysis of our active learning algorithm IWSelection and the addition of the auxiliary set.

RF, XGBoost and Stacking 1. And the performances of the stolen model on Spam and IMDB datasets are shown in Fig. 7. $K = 1$ represents the initial size of $X_i$ (200 for Spam and 500 for IMDB). And we multiply the size of $X_i$ from 1 to 7. Each testing point aborts when the number of queries is equivalent to that of $K = 1$. It is notable that, as the size of $X_i$ increases, the accuracy and agreement tend to decrease with the same query times. The accuracy and agreement are reduced by 7% and 8% for IMDB, respectively. `IWSelection` chooses more samples to query when the size of the augmented attack subset is larger, which makes it more "costly" to train a sub-model. However, the divide-and-conquer strategy requires more sub-models to lower the complexity of the training target. So expanding the size of the augmented subset may improve each sub-model, but meanwhile, increase the query cost. In our former experiments, the size of $X_i$ is set to be 0.3 times as that of $S_{err}$, which is the proper value we suggest.

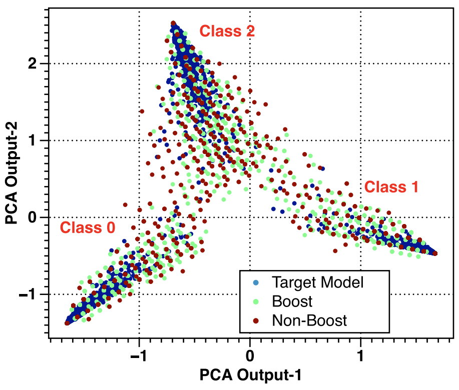


Fig. 5. The prediction points of the target model (blue points), the models stolen by Boost (green points) and the models stolen by Non-Boost (brown points).

### 5.3.5 Impact of Target Model Accuracy

We further discuss whether the performance of the target model influences the function of DivTheft. A low-accuracy RF model (76.8%) is trained on MNIST as the target model, and the high-accuracy model is the same as previous studies. We realize model stealing attacks on low-accuracy and high-accuracy target models (represented as L-Tar and H-Tar, respectively). And Fig. 8a plots the detailed accuracy and agreement changes of the stolen model along with the increased query number on MNIST. For simplicity, the stolen model with L-Tar is denoted as L-Stolen, the one with H-Tar is denoted as H-Stolen. Not surprisingly, the accuracy of L-Stolen is much lower than that of H-Stolen, but both of them reach a high agreement, which shows that low accuracy model won't lower the agreement rate because the label is given by the target model, and thus, $e*$ measures the uncertainty of the current stolen model to the target model's outputs. It can well validate the effectiveness of the divide-and-conquer process proposed in our work, by which an attacker can reconstruct a stolen model with a similar function as the target one.

### 5.3.6 Comparison With ExpAL

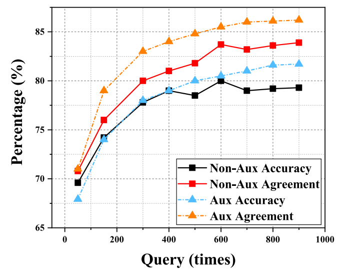
We further plot the convergence curve of the accuracy and agreement change with the query number of DivTheft and ExpAL, as shown in Figs 8b and 8c. For maximum query limit, we determine the queries as the stolen model is converged. With the same query limit (2.2K for MNIST and 800 for Adult), DivTheft can reach a higher accuracy (1.5% for MNIST and 5.4% for Adult) and agreement (1.7% for

TABLE 5
Effect of IWSelection

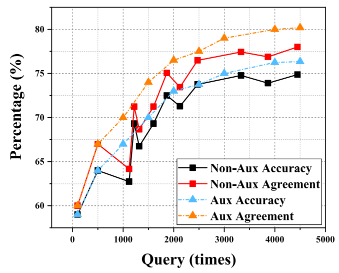
| Dataset | IWSelection | | | Non-IWSelection | | |
|---|---|---|---|---|---|---|
| | Acc | Agree | Queries | Acc | Agree | Queries |
| **Adult** | 80.7 | 85.5 | 800 | 79.9 | 84.7 | 7500 |
| **Iris** | 98.9 | 99.9 | 270 | 92.3 | 93.3 | 320 |
| **GSS** | 60.0 | 75.3 | 437 | 54.6 | 62.9 | 1550 |
| **Steak** | 61.8 | 80.0 | 2100 | 57.3 | 78.2 | 20700 |
| **MNIST** | 93.8 | 92.0 | 18000 | 55.39 | 55.9 | 21000 |
| **Fake** | 97.5 | 99.2 | 450 | 95.5 | 96.4 | 470 |
| **IMDB** | 79.5 | 82.9 | 18500 | 72.3 | 77.0 | 27300 |
| **Spam** | 95.9 | 98.0 | 1700 | 91.5 | 94.1 | 10500 |

TABLE 6
Effect of Auxiliary Set

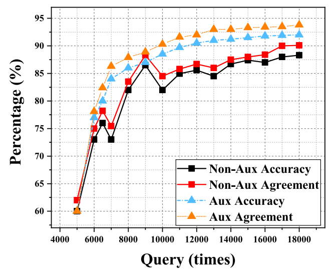
| Dataset | Acc | | Agree | | Queries | |
|---|---|---|---|---|---|---|
| | Aux | Non-Aux | Aux | Non-Aux | Aux | Non-Aux |
| **Adult** | 81.7 | 79.3 | 86.2 | 83.9 | 900 | 2100 |
| **Iris** | 98.9 | 93.3 | 99.9 | 100 | 250 | 270 |
| **GSS** | 60.0 | 57.1 | 75.3 | 70.6 | 437 | 510 |
| **Steak** | 61.8 | 58.2 | 80.0 | 77.6 | 2100 | 3020 |
| **MNIST** | 93.8 | 88.3 | 92.0 | 90.1 | 18000 | 20420 |
| **Fake** | 99.0 | 99.0 | 99.2 | 99.4 | 270 | 450 |
| **IMDB** | 79.5 | 75.3 | 82.9 | 78.4 | 18500 | 19000 |
| **Spam** | 95.9 | 91.6 | 98.0 | 93.6 | 1700 | 1700 |



(a) Accuracy and agreement changes with the increased query number on Adult.

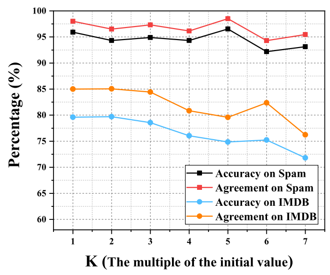


(b) Accuracy and agreement changes with the increased query number on IMDB.

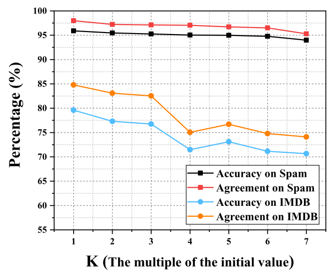


(c) Accuracy and agreement changes with the increased query number on MNIST.

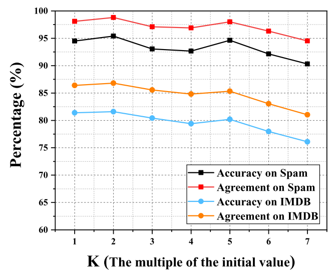
Fig. 6. The detailed comparison of DivTheft with and without the auxiliary set on datasets Adult, IMDB, and MNIST.



(a) The accuracy and agreement with different K on RF.

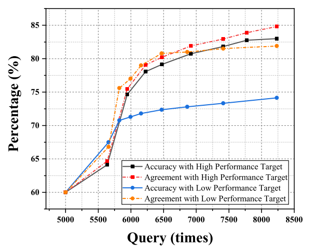


(b) The accuracy and agreement with different K on XGB.

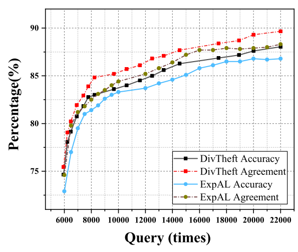


(c) The accuracy and agreement with different K on Stacking 1.

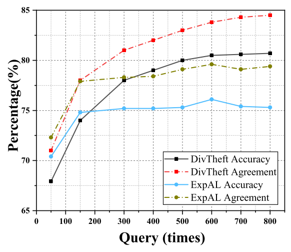
Fig. 7. Influence of the size of attack subset. K represents the multiple of the initial value. K=1 represents the initial value (200 for Spam and 500 for IMDB). K=7 means the size is expanded to $7 \times 200 = 1400$ for Spam.



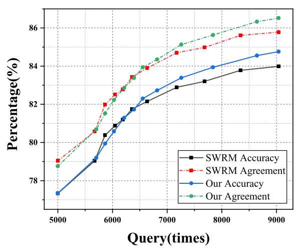
(a) Impact of target model accuracy on MNIST.



(b) Comparison with ExpAL on MNIST.



(c) Comparison with ExpAL on Adult.



(d) Impact of SWRM on MNIST.

Fig. 8. The results of: (a) attack on a low-accuracy target on MNIST; (b) convergence curve of DivTheft and ExpAL on MNIST; (c) convergence curve of DivTheft and ExpAL on Adult; (d) influence of SWRM.

MNIST and 5.1%) over ExpAL. By observing the convergence curve, the accuracy and the agreement curves of DivTheft are above the corresponding curves of ExpAL during the whole attack process, which shows that DivTheft outperforms ExpAL.

### 5.3.7 *Influence of SWRM*

Sampling with replacement mechanism (SWRM) is introduced and improved when forming an auxiliary set. We discuss whether the sample selection process combined with SWRM affects the performance of DivTheft. The variations of accuracy and agreement with query number with an improved SWRM (represented as Our-SWRM) and with normal SWRM (represented as SWRM) are shown in Fig. 8d. The result shows that with improved SWRM algorithm, the accuracy and agreement of the stolen model are slightly higher than the result with normal SWRM. Because normal SWRM cannot distinguish the group of more informative samples from the auxiliary set. It validates the effectiveness of our sample selection method, which reserves more representative samples of the queried ones.

## 6 Conclusion

In this paper, we introduced a method that expanded the existing model stealing attack to ensemble models via label-only prediction APIs. Specifically, we improved the current model stealing attack method from two perspectives. First, the divide-and-conquer strategy was introduced to lower the complexity of ensemble model stealing and improve the attack effectiveness. Second, considering that active learning was a vital technique for model stealing and the currently used method could often make the stolen model fall into local optimality, we proposed an auxiliary set selection algorithm to mitigate the problem. Experiments showed that our method could support model stealing over three kinds of common ensemble ways, bagging, boosting and stacking, and outperformed state-of-the-art model stealing methods.

## REFERENCES

[1] A. Qayyum et al., "Securing machine learning in the cloud: A systematic review of cloud machine learning security," *Front. Big Data*, vol. 3, 2020, Art. no. 587139.

[2] J. Maguire, "Top performing artificial intelligence companies of 2021," 2021. Accessed: Apr. 09, 2021. [Online]. Available: https://www.datamation.com/artificial-intelligence/ai-companies/

[3] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. 25th {USENIX} Secur. Symp.*, 2016, pp. 601–618.

[4] V. Chandrasekaran, K. Chaudhuri, I. Giacomelli, S. Jha, and S. Yan, "Exploring connections between active learning and model extraction," in *Proc. 29th {USENIX} Secur. Symp.*, 2020, pp. 1309–1326.

[5] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang, "Stealing links from graph neural networks," in *Proc. 30th {USENIX} Secur. Symp.*, 2021, pp. 2669–2686.

[6] K. Chen, S. Guo, T. Zhang, X. Xie, and Y. Liu, "Stealing deep reinforcement learning models for fun and profit," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2021, pp. 307–319.

[7] M. Belgiu and L. Drăguţ, "Random forest in remote sensing: A review of applications and future directions," *ISPRS J. Photogrammetry Remote Sens.*, vol. 114, pp. 24–31, 2016.

[8] J. R. Correia-Silva, R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos, "Copycat CNN: Stealing knowledge by persuading confession with random non-labeled data," in *Proc. Int. Joint Conf. Neural Netw.*, 2018, pp. 1–8.

[9] T. Takemura, N. Yanai, and T. Fujiwara, "Model extraction attacks on recurrent neural networks," *J. Inf. Process.*, vol. 28, pp. 1010–1024, 2020.

[10] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Front. Comput. Sci.*, vol. 14, no. 2, pp. 241–258, 2020.

[11] S. Milli, L. Schmidt, A. D. Dragan, and M. Hardt, "Model reconstruction from model explanations," in *Proc. Conf. Fairness, Accountability, Transparency*, 2019, pp. 1–9.

[12] P. Recognizer, "Plate recognizer automatic license plate recognition software," 2021. [Online]. Available: https://platerecognizer.com/

[13] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794.

[14] A. Beygelzimer, D. J. Hsu, J. Langford, and T. Zhang, "Agnostic active learning without constraints," *Adv. Neural Inf. Process. Syst.*, vol. 23, pp. 199–207, 2010.

[15] S. Pal, Y. Gupta, A. Shukla, A. Kanade, S. Shevade, and V. Ganapathy, "ActiveThief: Model extraction using active learning and unannotated public data," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 865–872.

[16] I. Mosafi, E. O. David, and N. S. Netanyahu, "Stealing knowledge from protected deep neural networks using composite unlabeled data," in *Proc. Int. Joint Conf. Neural Netw.*, 2019, pp. 1–8.

[17] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proc. Conf. Learn. Representations*, 2017.

[18] X. Yuan, L. Ding, L. Zhang, X. Li, and D. O. Wu, "ES attack: Model stealing against deep neural networks without data hurdles," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 9, pp. 1258–1270, 2020.

[19] H. Yu, K. Yang, T. Zhang, Y.-Y. Tsai, T.-Y. Ho, and Y. Jin, "CloudLeak: Large-scale deep learning models stealing through adversarial examples," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020.

[20] C. Zhang, "Efficient active learning of sparse halfspaces," in *Proc. 31st Conf. Learn. Theory*, 2018, pp. 1856–1880. [Online]. Available: https://proceedings.mlr.press/v75/zhang18b.html

[21] The automation of science, *Science*, vol. 324, no. 5923, pp. 85–89, 2009. [Online]. Available: http://www.jstor.org/stable/20493640

[22] W. H. Beluch, T. Genewein, A. Nurnberger, and J. M. Kohler, "The power of ensembles for active learning in image classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9368–9377.

[23] C. Schröder, A. Niekler, and M. Potthast, "Uncertainty-based query strategies for active learning with transformers," in *Proc. Findings Assoc. Comput. Linguistics*, 2021, pp. 2194–2203.

[24] W. Fu, M. Wang, S. Hao, and X. Wu, "Scalable active learning by approximated error reduction," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1396–1405.

[25] Y. Kim, "Deep active learning for sequence labeling based on diversity and uncertainty in gradient," in *Proc. Workshop Life-Long Learn. Spoken Lang. Syst.*, 2020, pp. 1–8.

[26] A. Prochaska, J. Pillas, and B. Bäker, "Active output selection strategies for multiple learning regression models," in *Proc. 10th Int. Conf. Pattern Recognit. Appl. Methods*, 2021, pp. 150–157.

[27] C.-A. Brust, C. Käding, and J. Denzler, "Active and incremental learning with weak supervision," *KI - Künstliche Intelligenz*, vol. 34, no. 2, pp. 165–180, Jan. 2020.

[28] A. Beygelzimer, S. Dasgupta, and J. Langford, "Importance weighted active learning," in *Proc. Annu. Int. Conf. Mach. Learn.*, 2009, pp. 49–56.

[29] T.-K. Huang, A. Agarwal, D. Hsu, J. Langford, and R. E. Schapire, "Efficient and parsimonious agnostic active learning," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 2755–2763.

[30] D. Basu, "On sampling with and without replacement," *Sankhyā: The Indian J. Statist.*, vol. 20, no. 3/4, pp. 287–294, 1958.

[31] A. Taherkhani, G. Cosma, and T. M. McGinnity, "AdaBoost-CNN: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning," *Neurocomputing*, vol. 404, pp. 351–366, 2020.

[32] B. Mozafari, P. Sarkar, M. Franklin, M. Jordan, and S. Madden, "Scaling up crowd-sourcing to very large datasets: A case for active learning," *Proc. VLDB Endowment*, vol. 8, no. 2, pp. 125–136, 2014.

[33] V. Khadse, P. Mahalle, and G. Shinde, "Statistical study of machine learning algorithms using parametric and non-parametric tests: A comparative analysis and recommendations," *Int. J. Ambient Comput. Intell.*, vol. 11, pp. 80–105, 2020.

[34] V. Duddu, D. Samanta, D. V. Rao, and V. E. Balas, "Stealing neural networks via timing side channels," 2018, *arXiv:1812.11720*.

[35] S. Kariyappa, A. Prakash, and M. Qureshi, "Maze: Data-free model stealing attack using zeroth-order gradient estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13814–13823.

[36] Y. Zhu, Y. Cheng, H. Zhou, and Y. Lu, "Hermes attack: Steal DNN models with lossless inference accuracy," *USENIX Secur. Sympos.*, 2020.

[37] L. Lyu, X. He, F. Wu, and L. Sun, "Killing two birds with one stone: Stealing model and inferring attribute from BERT-based APIs," 2021, *arXiv:2105.10909*.

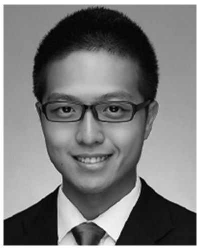
[38] T. Takemura, N. Yanai, and T. Fujiwara, "Model extraction attacks against recurrent neural networks," 2020, *arXiv:2002.00123*.

[39] S. Pal, Y. Gupta, A. Shukla, A. Kanade, S. Shevade, and V. Ganapathy, "A framework for the extraction of deep neural networks by leveraging public data," 2019, *arXiv:1905.09165*.

**Zhuo Ma** received the PhD degree in computer architecture from Xidian University, Xi'an, China, in 2010. Now, he is a professor with the school of Cyber Engineering, Xidian University. He awards "Huashan Scholars" distinguished professor. His research interests include cryptography, AI security, and IoT security.

**Xinjing Liu** received the BS degree in Internet of things from the Xi'an University of Technology in 2020. She is currently working toward the master's degree with the School of Cyber Engineering, Xidian University. Her research interests cover machine learning in cyber security and formal analysis of authentication protocols.

**Yang Liu** received the BS degree in computer science and technology from Xidian University in 2017. He is currently working toward the master's degree with the School of Cyber Engineering, Xidian University. His research interests cover formal analysis of authentication protocols and deep learning neural network in cyber security.



**Ximeng Liu** (Member, IEEE) received the BSc degree in electronic engineering from Xidian University, Xi'an, China, in 2010 and the PhD degree in Cryptography from Xidian University, in 2015. Now he is a full professor with the College of Mathematics and Computer Science, Fuzhou University. Also, he is a research fellow with the School of Information System, Singapore Management University, Singapore. He has published more than 100 papers on the topics of cloud security and big data security-including papers in *IEEE Transactions on Computers*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Service Computing*, *IEEE Internet of Things Journal*, and so on. He awards "Minjiang Scholars" distinguished professor, "Qishan Scholars" in Fuzhou University, and ACM SIGSAC China Rising Star Award (2018). His research interests include cloud security, applied cryptography and big data security. He is a member of ACM, CCF.

**Zhan Qin** received the PhD degree from the Computer Science and Engineering department, State University of New York at Buffalo in 2017. He is currently a ZJU100 young professor, with both the College of Computer Science and Technology and the Institute of Cyberspace Research (ICSR) at Zhejiang University, China. He was an assistant professor with the Department of Electrical and Computer Engineering in the University of Texas at San Antonio. His current research interests include data security and privacy, secure computation outsourcing, artificial intelligence security, and cyber-physical security in the context of the Internet of Things. His works explore and develop novel security sensitive algorithms and protocols for computation and communication on the general context of Cloud and Internet devices.

**Kui Ren** (Fellow, IEEE) is currently a distinguished professor with the School of Computer Science and Technology at Zhejiang University, where he also directs the Institute of Cyber Space Research. His research interests include Cloud and data security, AI and IoT security, and privacy-enhancing technologies. He is a distinguished member of ACM.