ACCEPTED FROM OPEN CALL

Model Extraction Attacks and Defenses on Cloud-Based Machine Learning Models

Xueluan Gong, Yanjiao Chen, Qian Wang, Wang Yang, and Xinchang Jiang

The authors conduct an investigation of existing approaches to model extraction attacks and defenses on cloud-based models. They classify attack schemes into two categories based on whether the attacker aims to steal the property or the functionality of the model. They also categorize defending schemes into two groups based on whether the scheme relies on output disturbance or query observation.

Abstract

Machine learning models have achieved state-of-the-art performance in various fields, from image classification to speech recognition. However, such models are trained with a large amount of sensitive training data, and are typically computationally expensive to build. As a result, many cloud providers (e.g., Google) have launched machine-learning-as-a-service, which helps clients benefit from the sophisticated cloud-based machine learning models via accessing public APIs. Such a business paradigm significantly expedites and simplifies the development circles. Unfortunately, the commercial value of such cloud-based machine learning models motivates attackers to conduct *model extraction attacks* for free use or as a springboard to conduct other attacks (e.g., craft adversarial examples in black-box settings). In this article, we conduct a thorough investigation of existing approaches to model extraction attacks and defenses on cloud-based models. We classify the state-of-the-art attack schemes into two categories based on whether the attacker aims to steal the property (i.e., parameters, hyperparameters, and architecture) or the functionality of the model. We also categorize defending schemes into two groups based on whether the scheme relies on output disturbance or query observation. We not only present a detailed survey of each method, but also demonstrate the comparison of both attack and defense approaches via experiments. We highlight several future directions in both model extraction attacks and its defenses, which shed light on possible avenues for further studies.

Introduction

Machine learning models have been widely used in many areas including speech recognition, natural language processing (NLP), image recognition, and so forth. These models are trained with enormous data and massive parameters to attain a high prediction power, which is computationally prohibitive for users with limited time and resources. To cater to such business opportunities, machine-learning-as-a-service (MLaaS) has been launched by cloud service providers (e.g., Amazon, Google), which host sophisticated machine learning models remotely on the cloud and charge clients for accessing models via prediction application programming interfaces (APIs) on a pay-per-query basis, as shown in Fig. 1. Since developing these models consumes time, money, and human effort, cloud service providers keep the details of such cloud-based models (e.g., datasets, model architecture, model hyperparameters) confidential. For ordinary clients, these APIs are black-box, which take queries as inputs and return the prediction results without revealing internal operations. For various commercial and other motivations (e.g., implementing black-box adversarial examples [1]), extracting such cloud-based models is attractive.

However, conducting model extraction attacks is not trivial, especially for stealing complicated deep neural networks (DNNs) with massive parameters and hyperparameters. With the rapid development of neural network studies, the possible architectures of DNNs expand dramatically and become increasingly complex, making model extraction attacks more challenging.

Nowadays, model extraction attacks have been extensively studied from various aspects, including parameter stealing [2], hyperparameter stealing [3], architecture extraction [4], decision boundary inference [1, 5], and functionality stealing [6, 7]. State-of-the-art attacks have been proved to be effective against modern commercial MLaaS services, including BigML [2], Amazon [1–3], Microsoft [3], Google [1], and MetaMind [1]. At the same time, in order to reduce the losses caused by model extraction attacks, many defense strategies have been proposed to detect or prevent such attacks.

In this article, to the best of our knowledge, we are the first to present a comprehensive review of model extraction attacks and defenses for cloud-based MLaaS services and put forward a classification of existing state-of-the-art approaches. We not only summarize the current progress on model extraction attacks and defenses, but also compare their advantages and disadvantages from various aspects. We conduct experiments to provide detailed evaluations of representative attack and defense approaches. We highlight several future research directions in advancing the effectiveness of existing attack and defense schemes.

Preliminaries

Machine Learning as a Service

Many cloud providers, including IBM, Amazon, and Microsoft, have materialized the business model of MLaaS, which helps resource-constrained users enjoy the benefit of

Digital Object Identifier:
10.1109/MCOM.001.2000196

Xueluan Gong, Yanjiao Chen, Qian Wang, Wang Yang, and Xinchang Jiang are with Wuhan University.

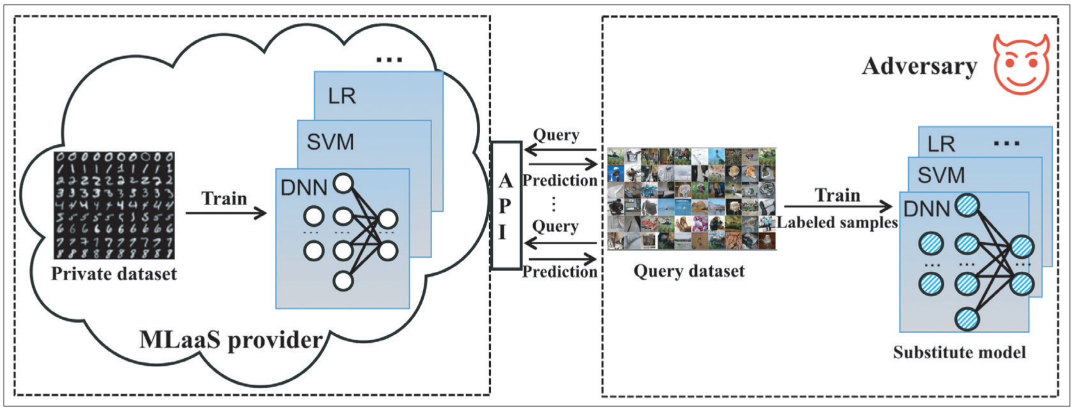

Figure 1. An overview of model extraction attack against an MLaaS provider. The left part denotes how the cloud-based models are developed in the MLaaS provider, and the right part demonstrates the flow of the model extraction attack.

The training of the ML model can be formulated as an optimization problem to minimize the loss function that measures the difference between the ground-truth label and the predicted label over all training data samples. To avoid overfitting, a regularisation function is usually added to the loss function.

machine learning. Ready-made, generic machine learning tools (e.g., predictive analytics, APIs, data visualization, and natural language processing) are provided by MLaaS for use and adaptation by small and medium-size companies. Users pay the cloud providers on a pay-per-query basis; for example, typical image classification service costs around $1~$10 per 1000 queries, depending on the specialty and complexity of the machine learning model.

MLaaS services are diversified across different providers. We give a comprehensive comparison of the major commercial MLaaS providers in Table 1. *White-box* MLaaS enables clients to download and implement machine learning models locally, while *black-box* MLaaS only allows clients to access cloud-based models via a prediction query interface. Except for PredictionIO, all listed MLaaS services provide both the predicted label and the confidence score, which can boost the model stealing process. Some MLaas platforms allow clients to upload their own models and charge other clients for using their models (monetization).

There are various kinds of machine learning models adopted by MLaaS, including logistic regression, support vector machine, and DNN. An ML model can be described as a function *f* that takes a *d*-dimensional vector as the input and outputs the corresponding predicted label. The training of the ML model can be formulated as an optimization problem to minimize the loss function that measures the difference between the ground-truth label and the predicted label over all training data samples. To avoid overfitting, a regularization function is usually added to the loss function.

Provider	Availability	Confidence score	Monetization
Google	Black-box	Available	YES
Amazon	Black-box	Available	NO
BigML	White-box	Available	YES
Microsoft	Black-box	Available	NO
PredictionIO	White-box	Not available	NO

Table 1. Comparison of major commercial MLaaS providers.

Problem Description

We consider an adversary who aims to conduct model extraction attack on a cloud-based model which is protected by the cloud service provider. In this part, we outline both the attack and defense goals. Each attack or defense methodology mentioned later is designed according to the relevant goals.

Attack Objective: The attacker's goal is to learn the private attribute information (e.g., structure, parameters) of the cloud-based model or establish a substitute model with similar functionalities to the cloud-based model. If the attacker has access to the training dataset of the cloud-based models, a substitute model can easily be trained with this training dataset. However, in the real-world MLaaS scenario, the adversary knows nothing about the training data or the test data used to train and evaluate the model. Furthermore, the adversary has no access to the internals of the cloud-based models, including model parameters, hyperparameters, and model architecture. The attacker can only access the API provided by the MLaaS provider, with inputs in and prediction results out.

Defense Objective: The defender's goal is to prevent the attacker from stealing private information or replicating the cloud-based model's functionality. Specifically, given a certain budget of the attacker, the defender aims to reduce the accuracy of the stolen model established by the attacker. The defender also needs to ensure high classification accuracy to the benign clients to retain the utility of the cloud-based model for the prediction tasks. The defender tries to increase the cost of stealing the model to reach a certain accuracy target, thus the attacker is discouraged from performing the attack.

State-of-the-Art Model Extraction Attacks

In this section, we present a thorough investigation of existing approaches to model extraction attacks on cloud-based models and give a comprehensive comparison of them.

Attack Approaches

According to the purpose of the attacker, we classify state-of-the-art attack schemes into two categories: model attribute extraction and model

The defender's goal is to prevent the attacker from stealing private information or replicating the cloud-based model's functionality. Specifically, given a certain budget of the attacker, the defender aims to reduce the accuracy of the stolen model established by the attacker.

Method	Attack type	Extract DNNs	Need confidence information	Works with a limited query budget
StealML [2]	Model attribute extraction	NO	YES	YES
StealHyperparameters [3]	Model attribute extraction	YES	YES	YES
StealNN [4]	Model attribute extraction	YES	NO	NO
PRADA [5]	Model attribute extraction	YES	NO	YES
KnockoffNets [7]	Model functionality extraction	YES	NO	YES
StealClassifier [6]	Model functionality extraction	NO	NO	YES

Table 2. Comparison of state-of-the-art model extraction attacks.

functionality extraction.

Model Attribute Extraction: Most existing extraction attacks focus on attribute stealing, including parameters [2], hyperparameters [3], architecture [4], and decision boundary [5], which are tightly related to the intellectual property rights of the cloud-based models.

Tramer *et al*. proposed StealML [2], which requires knowledge of both prediction labels and confidence scores, and works with various popular cloud-based models such as logistic regression, support vector machine (SVM), decision tree, and shallow neural networks. In StealML, the attacker uses a set of crafted or publicly available data samples to query the cloud-based model, and then uses the input-output pairs to construct a series of equations of parameters. By solving these equations, the adversary can infer the internal parameters of the targeted model. This scheme is conceptually simple, but the attacker needs to know the family of the cloud-based model. Moreover, StealML only applies to simple models with a small amount of parameters and is ineffective for DNN models.

Apart from parameters, hyperparameters of the cloud-based ML models are also of significant commercial value. Wang *et al*. proposed Steal-Hyperparameter [3] based on the observation that model parameters are decided to minimize the objective function, and the corresponding *gradient* of the objective function is close to 0. Hence, the adversary can calculate the gradient of the objective function at the value of the model parameters and make the objective function equal 0 to obtain a system of linear equations about hyperparameters. The *linear least square method* [8] is utilized to solve the overdetermined problem in linear systems (the number of equations is larger than the number of unknown variables).

To infer the architecture of the cloud-based DNN model, Duddu *et al*. proposed StealNN [4] using the techniques of timing side channels. It is based on the assumption that the total execution time of the neural network depends on the depth of the network. The adversary needs to know the hardware on which the victim model is running, and use the same hardware to build timing profiles of different neural networks. Specifically, the attacker uses iterative membership inference [9] to reconstruct the training dataset, which is used to query the cloud-based model to calculate the total execution time. Then a regressor is trained to quantify the relationship between execution times and the neural network depth. Finally, the attacker can infer the depth (the number of layers) of the DNN using the pre-trained regressor and the total execution time.

Recently, Juuti *et al*. presented an iterative model extraction strategy named PRADA that includes many duplication rounds [5]. Each round consists of three phases: querying the cloud-based model, training the substitute model on the obtained labels, and crafting new synthetic samples based on Jacobian-Based Data Augmentation (JBDA) using the updated substitute model. Different from other extraction methods, PRADA is evaluated by an extra metric, the *transferability* of the adversarial samples, which concerns extracting the decision boundary of the targeted black-box model.

Model Functionality Extraction: Apart from attribute extraction, another attack goal is to construct a substitute model that has similar performance to the targeted cloud-based model for the same tasks (i.e., functionality attacks).

Shi *et al*. proposed StealClassifier [6] to extract the functionality of both Naive Bayes and SVM classifiers for text classification. It is shown that it is possible to extract the functionality of simple models using more complicated models (using DNN to extract the functionality of SVM), but the reverse is impractical (using SVM or Naive Bayes models to extract the functionality of DNNs).

To steal the functionality of more sophisticated architectures (i.e., DNN models), KnockoffNets [7] is proposed. The overall process includes three phases: substitute model architecture selection, query dataset construction, and substitute model training. Since the architecture of the cloud-based model is unknown to the attacker, selecting an appropriate architecture is nontrivial. Fortunately, it is proved that a duly complicated architecture is able to reproduce the functionality of common DNN models well [1, 7], and the substitute model and the victim model may not even be in the same family; for example, VGG-16 may be used to extract ResNet-10. For query dataset construction, the adversary usually chooses a huge benchmark dataset (e.g., ILSVRC, OpenImages) as the query dataset. To query more efficiently, *reinforcement learning* strategies are leveraged to select the best query samples. Once the attacker gets the prediction results (e.g., label and confidence scores) from the cloud-based model, he/she can use the selected architecture to train these query-output pairs to obtain a knockoff model.

Comprehensive Comparison of Attacks

For model extraction attacks, we give a comprehensive comparison in Table 2 from various aspects.

- *Extract DNNs.* The ability to steal the cloud-based DNN models is significant for the attacker because DNN models are increasingly used in MLaaS for their excellent performance. We can see that except for StealML [2] and StealClassifier [6], all other approaches are capable of extracting DNNs. To recap, StealML [2] only works on SVMs, LR, decision trees, and shallow neural networks, while StealClassifier [6] works only on SVM and Naive Bayes.
- *Confidence score information.* After querying the cloud-based model, the attacker will receive the returned prediction results (e.g., label, confidence scores). The confidence scores provide extra information to the attacker but are not available in certain MLaaS APIs (e.g., PredictionIO). Approaches that only require prediction labels are more practical but are more challenging to implement. With the advancement in model extraction studies, we can see that more strategies have considered such a constrained situation [4–7].
- *Query budget.* To attain a performance target of the stolen model, there should be enough queries to collect information about the cloud-based model. The cost of querying the MLaaS APIs cannot be neglected since every query is billed on a pay-per-query basis. If the cost of model stealing is higher than that of model training (including training data collection), there is no motivation for the attacker to steal models. Most state-of-the-art strategies can meet a realistic query budget. In comparison, StealNN [4] needs more queries than others.

State-of-the-Art Defense Approaches

Defense Approaches

In this section, we introduce two categories of state-of-the-art countermeasures against model extraction attacks: defending by output perturbation (i.e., injecting special perturbations to model predictions) and detecting by observing the queries (i.e., monitoring the user-server streams and generating a warning if malicious behaviors are detected). Furthermore, we give a comprehensive comparison of these defense approaches.

Defense by Output Perturbation: Since the attackers need certain information (e.g., prediction labels and confidence scores) to steal the victim model, the cloud service provider can defend their models by reducing or obfuscating the information provided by APIs.

Lee *et al.* proposed Deceptive Perturbations [10], which perturb the softmax activation function that produces the confidence scores such that the parameters are hidden from the attacker. Specifically, original responses are altered by adding perturbations using Reverse Sigmoid [10] with a sum-to-one normalizer. However, the top-1 labels are not tampered with; thus, some extraction attacks that work only with prediction labels are still effective [5, 7].

To address this problem, Orekondy *et al.* proposed Prediction Poisoning [11], a utility-constrained defense framework that satisfies both privacy and utility objectives via perturbing the predictions. The perturbations maximize the angular deviation between the gradient of the poisoned posteriors and that of the original model. A tunable parameter is used to balance the security and the prediction accuracy. However, this method requires a lot of gradient calculations, which results in high computational costs and increases inference delays by several orders of magnitude.

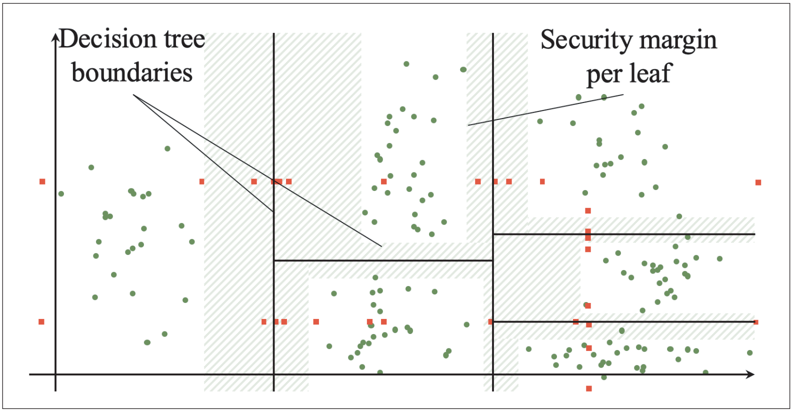

Figure 2. Defense strategy using the closeness-to-the-boundary concept.

BDPL uses *differential privacy* to perturb the output [12]. A boundary differential privacy layer is added to obfuscate the responses of queries around the decision boundary. Specifically, BDPL identifies whether a query is sensitive by checking the corner points with a given radius to the query on all dimensions. If there is a flipping point within the ball centered at the query with the given radius, the current query is identified as sensitive. Then the defender will return obfuscated responses using the boundary randomized response algorithm controlled by the privacy budget.

The perturbation strategy is an active defense strategy, which does not differentiate potential attackers from benign clients. Therefore, the utility of the cloud-based models may be affected due to perturbations.

Detection by Observing Queries: Apart from reducing the information returned to the attacker, the cloud service provider may detect potential attacks by observing queries from the clients. Such a passive defense strategy will not impair the utility of cloud-based models.

Kesarwani *et al.* proposed Extraction Warning [13] to discover potential extraction attacks and issue extraction status warnings. The model owner needs to set an extraction threshold, and an alarm will be issued whenever the extraction threshold is hit. To measure the knowledge gained by the adversary, two methods are designed. The first approach measures the total information gain by training and updating a local proxy model for each client to estimate the information gain concerning a given validation set, which is computationally expensive. The second method is to maintain *compact query summaries* [13] for each client and computes the feature space covered by the client's queries.

Forgotten Siblings [14] applies *closeness-to-the-boundary* (a concept from digital watermarking) to reveal extraction attacks against decision trees. Since adversaries usually use the queries around

Method	Defense type	Defense effectiveness	Apply for DNN	Defend colluding attackers
Deceptive Perturbations [10]	Output perturbation	*Accuracy drop*[1] more than 20%	YES	NO
Prediction Poisoning [11]	Output perturbation	*Accuracy drop* more than 20%	YES	NO
BDPL [12]	Output perturbation	*Extraction rate*[2] drop up to 12%	YES	NO
PRADA [5]	Query observation	Nearly 0.0% *FPR*[3]	YES	NO
Extraction Warning [13]	Query observation	Most *extraction rates* exceed threshold[4]	NO	YES
Forgotten Siblings [14]	Query observation	nearly 0.11 p[5]	NO	NO

[1] *Accuracy drop* measures the accuracy decrease of the substitute model before and after applying the defense strategy.

[2] *Extraction rate* measures the proportion of matching predictions: given the same input, the substitute model and the cloud-based model have the same output.

[3] *FPR* evaluates the ratio of false alarms to all query sequences.

[4] *Extraction rate* measures the knowledge learned by clients. A warning is issued if the extraction rate exceeds a predestined threshold.

[5] The metric p denotes the fraction of successfully extracted leaves, which is used to determine the knowledge gain by the attacker. (The lower the value, the more effective the defense. $p = 1$ denotes 100% attack success rate.)

Table 3. Comparison of state-of-the-art model extraction defenses.

the decision boundary, the defender can identify malicious queries by tracking the closeness of the user's queries to the class boundary. Forgotten Siblings defines the security margins alongside the decision tree boundary using the statistical distribution of the training data, as shown in Fig. 2. Every submitted query is determined to be within the margin or not. Subsequently, the average ratio between the queries inside and outside the margin for all leaves is calculated. If the query ratio exceeds the threshold, the current query will be identified as a malicious query (red square).

However, the above two schemes only work on the cloud-based models with linearly separated prediction classes (e.g., decision trees) but do not work for complicated models like DNNs. In addition, false alarms may be raised as a benign client may also explore vast areas of the input spaces.

PRADA [5] was proposed to detect attacks against any cloud-based ML models. It is based on the assumption that the distances between queries used by the attacker are usually artificially controlled, and thus will deviate from a normal (Gaussian) distribution. Given a detection threshold, a domain-specific distance metric is used to compute distances between queries. PRADA keeps track of the minimum distance between a new input sample and all previous samples in the same class to model the distribution of queries. The *Shapiro-Wilk test statistic* is used to quantify if the distribution of the current queries fits a normal distribution. An attack is considered to occur if the Shapiro-Wilk test statistic is below the threshold. However, PRADA may not work if the adversary adds dummy queries to fit a normal distribution.

Comprehensive Comparison of Defenses

A comparison of the defense approaches is given in Table 3.

- *Defense effectiveness.* The defense effectiveness describes whether a defending method is useful when defending against model extraction attacks. There are no unified metrics to measure the defense effectiveness of all approaches for several reasons. The source codes of most defending approaches are not open. Different defending schemes focus on different metrics, such as *accuracy drop* in Deceptive Perturbations [10] and Prediction Poisoning [11], *extraction rate drop* in BDPL [12], *FPR* in PRADA [5], *fraction of successfully extracted leaves p* in Forgotten Siblings [14]. Moreover, different detection defense strategies target different models; for example, DNN in PRADA [5], Deceptive Perturbations [10], Prediction Poisoning [11], and BDPL [12], and decision trees in Extraction Warning [13] and Forgotten Siblings [14].
- *Apply for DNN.* Since most of the current cloud-based models are DNNs, developing strategies to resist DNN model extraction attacks is increasingly essential for cloud service providers. Except for Extraction Warning [13] and Forgotten Siblings [14], which are designed for decision trees, we can see that most defense approaches can apply to complex DNN models.
- *Colluding attacker defense.* Defending against collusion attacks is critical for cloud service providers. An attacker may distribute his/her carefully crafted queries to colluding partners so that the statistics of queries of each malicious client is below the alarming threshold. Therefore, the attacker can evade the defending strategy. In Table 3, we can see that only Extraction Warning [13] among all the defense works has considered this issue.

Performance Evaluation

To evaluate the performance of different model extraction attacks, we conduct experiments on StealML [2], PRADA [5], and KnockoffNets [7] in terms of attack effectiveness. We use two evaluation metrics: *absolute accuracy* and *relative accuracy*. Absolute accuracy is the prediction accuracy of the extracted substitute model. Relative accuracy is the relative prediction accuracy of the extracted model compared to the prediction accuracy of the cloud-based model. Following [1, 5], we have established a black-box model on the MNIST dataset (28×28) with LeNet structure (two convolution layers with pooling and two fully connected layers), and its accuracy is 99.47 percent. The learning rates of StealML [2], PRADA [5], and KnockoffNets [7] are all set as 0.01. We set the batch size of StealML [2] as 20, PRADA [5] as 50, and KnockoffNets [7] as 8. In the experiments, the total number of queries to the cloud APIs is 10,000. StealML [2] and PRADA [5] use the crafted query samples, and KnockoffNets [7] uses the EMNIST Letters as the query dataset. All experiments are carried out on a machine with an Intel Core CPU with 6-core operating at 3.70 GHz and equipped with 64 GB RAM, running a Linux 4.15.0 operating system.

The evaluation results are shown in Fig. 3. We can see that both KnockoffNets [7] and PRADA [5] achieve a higher relative accuracy than StealML [2]. The possible reason is that StealML [2] only works on shallow neural networks (i.e., with only one hidden layer), while LeNet-5 structure is used in the experiments. Furthermore, since KnockoffNets [7] uses EMNIST as the query samples, which consists of handwritten character digits, it performs slightly better than PRADA [5].

As mentioned above, we cannot compare all the defenses' work in a fair way due to the lack of source codes and unified metrics. Therefore, we make analyses in a qualitative way according to their original contexts and conduct part of the experiments. We present the comparison results in the third column of Table 3. We conduct the experiments on PRADA [5] and Forgotten Siblings [14] with their open source codes. Following the same settings of [5], we protect the MNIST LeNet-5 model from StealML [2] attacks and set the threshold as 0.90. The experiment results have demonstrated the effectiveness of PRADA that produced nearly 0.0 percent (FPR false positive rate). Similarly, following [14], we protect a pre-trained model (trained with the Wine Quality dataset) from StealML [2] attacks, and set the threshold as 0.3. The experiments have shown that the fraction of successfully extracted leaves *p* is only 0.11, which confirms the effectiveness of the defense strategy.

Future Research Directions

Potential Research Directions on Attacks

From the perspective of attacks, there are three aspects worth further exploring.

First, most of the existing model extraction attacks focus on attribute extraction, and existing functionality extraction strategies are not ideal. KnockoffNets [7] is effective to steal simple DNN architectures, but also requires access to large amounts of related public datasets that serve as the *transfer dataset* to construct substitute models that are highly similar to the original cloud-based models. Therefore, how to design a functionality extraction algorithm that depends less on public datasets but can still achieve high accuracy of the substitute model needs to be studied in the future.

Second, the attacks are significantly less effective with less information; for example, the accuracy of the substitute model will drop rapidly as there are fewer returned top-*k* labels (a smaller *k*) in KnockoffNets attacks [7]. This problem is especially severe for complex DNNs. How to maintain/increase the capability of model extraction attacks when cloud APIs only return the label with the highest confidence score is a problem worth studying.

Finally, existing model extraction approaches mainly work on stealing DNN models with relatively simple structures [3, 5, 6]. When stealing real-world APIs with millions of parameters, such approaches may not achieve satisfying results. How to design an effective attacking strategy in terms of stealing sophisticated cloud-based DNNs is a possible future research direction.

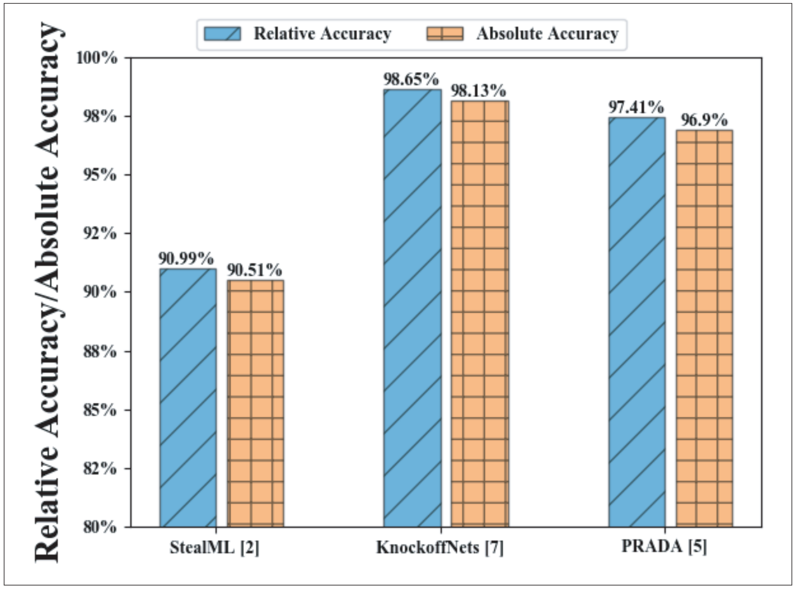

Figure 3. Performance comparison of state-of-the-art model extraction attacks.

Potential Research Directions on Defenses

From the perspective of defenses, we highlight four aspects.

First, the adversary may evade the detection using some special techniques. For instance, the adversary can make *dummy queries*, which are not used to build the substitute model but can mimic the query distribution of benign clients [5], or use samples with lower information gains and limited coverage of feature space [13]. How to effectively detect model extraction attacks facing these more intelligent attacks needs further research.

Second, state-of-the-art defending approaches based on perturbing prediction results usually sacrifice the utility to achieve the security target [11]. As the perturbation increases, the model prediction accuracy deteriorates, and the security level increases. However, even a slight decrease in performance may make commercial cloud-based models lose competitive advantage. How to effectively protect cloud-based models while maintaining its utility should also be considered.

Third, when considering attacks under federated learning or the blockchain scenario, we think the cloud provider can employ the differential privacy technique and encrypt the trained model aiming to prevent the attackers from extracting the model or inferring original training data via reverse engineering.

Last but not least, recent studies have shown that the strategies of side-channel attacks can be applied to model extraction attacks (e.g., timing side-channel [4] and electromagnetic side-channel [15]). Both can achieve a high attack success rate. However, there are no defending strategies designed for these kinds of model extraction attacks. It should be an important direction for

Recent studies have shown that the strategies of side-channel attacks can be applied to model extraction attacks, e.g., timing side-channel and electromagnetic side-channel. Both can achieve a high attack success rate. However, there is no defending strategies designed for such kind of model extraction attacks.

future investigation.

Conclusion

Model extraction attacks on cloud-based MLaaS services pose serious security issues. In this article, we provide a comprehensive summary of existing methods of model extraction attacks and possible defenses. We review the latest research findings, compare advantages and disadvantages of different approaches, and conduct experimental evaluations of representative attack and defense methods. Finally, we highlight potential directions that are worth exploring to further propel the research in this field.

References

[1] N. Papernot *et al.*, "Practical Black-Box Attacks Against Machine Learning," *ACM on Asia Conf. Computer and Commun. Security*, 2017, pp. 506–19.

[2] F. Tramér *et al.*, "Stealing Machine Learning Models Via Prediction Apis," *25th USENIX Security Symp.*, 2016, pp. 601–18.

[3] B. Wang and N. Z. Gong, "Stealing Hyperparameters in Machine Learning," *IEEE Symp. Security and Privacy*, 2018, pp. 36–52.

[4] V. Duddu *et al.*, "Stealing Neural Networks Via Timing Side Channels," arXiv preprint arXiv:1812.11720, 2018.

[5] M. Juuti *et al.*, "Prada: Protecting Against DNN Model Stealing Attacks," *IEEE Euro. Symp. Security and Privacy*, 2019, pp. 512–27.

[6] Y. Shi, Y. Sagduyu, and A. Grushin, "How to Steal a Machine Learning Classifier With Deep Learning," *IEEE Int'l. Symp. Technologies for Homeland Security*, 2017, pp. 1–5.

[7] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff Nets: Stealing Functionality of Black-Box Models," *IEEE Conf. Computer Vision and Pattern Recognition*, 2019, pp. 4954–63.

[8] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, Wiley, 2012, vol. 821.

[9] A. Salem *et al.*, "Ml-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models," *26th Annual Network and Distributed System Security Symp.*, The Internet Society, 2019.

[10] T. Lee *et al.*, "Defending Against Neural Network Model Stealing Attacks Using Deceptive Perturbations," *IEEE Security and Privacy Wksps.*, 2019, pp. 43–49.

[11] T. Orekondy, B. Schiele, and M. Fritz, "Prediction Poisoning: Utilityconstrained Defenses Against Model Stealing Attacks," arXiv preprint arXiv:1906.10908, 2019.

[12] H. Zheng *et al.*, "BDPL: A Boundary Differentially Private Layer Against Machine Learning Model Extraction Attacks," *Euro. Symp. Research in Computer Security*, Springer, 2019, pp. 66–83.

[13] M. Kesarwani *et al.*, "Model Extraction Warning in Mlaas Paradigm," *34th Annual Computer Security Applications Conf.*, ACM, 2018, pp. 371–80.

[14] E. Quiring, D. Arp, and K. Rieck, "Forgotten Siblings: Unifying Attacks on Machine Learning and Digital Watermarking," *IEEE Euro. Symp. Security and Privacy*, 2018, pp. 488–502.

[15] L. Batina *et al.*, "CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel," *28th USENIX Security Symp*, 2019, pp. 515–32.

Biographies

Xueluan Gong (xueluangong@whu.edu.cn) received her B.S. degree in computer science and electronic engineering from Hunan University in 2018. She is currently pursuing a Ph.D. degree in computer science with Wuhan University, China. Her research interests include network security, AI security, and data mining.

Yanjiao Chen (chenyanjiao@whu.edu.cn) received her B.E. degree in electronic engineering from Tsinghua University in 2010 and her Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology in 2015. She is currently a professor at Wuhan University, China. Her research interests include network economics, network security, and quality of experience of multimedia delivery/distribution.

Qian Wang [SM] (qianwang@whu.edu.cn) is a professor with the School of Cyber Science and Engineering, Wuhan University. He received his Ph.D. degree from Illinois Institute of Technology. His research interests include AI security, data storage, search and computation outsourcing security and privacy, wireless system security, big data security and privacy, applied cryptography, and so on. He received the National Science Fund for Excellent Young Scholars of China award in 2018. He is also an expert under the National "1000 Young Talents Program" of China. He is a recipient of the 2018 IEEE TCSC Award for Excellence in Scalable Computing for Early Career Researcher and also the 2016 IEEE Asia-Pacific Outstanding Young Researcher Award. He serves as an Associate Editor for *IEEE Transactions on Dependable and Secure Computing* and *IEEE Transactions on Information Forensics and Security*. He is a member of ACM.

Wang Yang (yang_nsme@whu.edu.cn) is currently pursuing a Bachelor's degree in computer science at Wuhan University, China. His research interests include information security and AI security.

Xinchang Jiang (xinchangjiang@whu.edu.cn) is currently pursuing a Bachelor's degree in computer science at Wuhan University. Her research interests include information security and AI security.