Interesting Near-boundary Data: Model Ownership Inference for DNNs

Anonymous submission

Abstract

Deep neural networks (DNNs) require expensive training, which is why the protection of model intellectual property (IP) is becoming more critical. Recently, model stealing has emerged frequently, and many researchers have been inspired by digital media watermarking to design model watermarking and fingerprinting for verifying model ownership. However, attacks such as ambiguity statements have been used to break the current defense, which poses a challenge to model ownership verification. Therefore, this paper proposes an interesting near-boundary data as evidence for obtaining model ownership and innovatively proposes to infer model ownership instead of verifying model ownership. In this paper, we propose to construct the initial near-boundary data using an algorithm that adds slight noise to generate adversarial examples. We design a DCGAN-based data generator to privatize the near-boundary data. Our main observation is that the near-boundary data exhibit results close to the classification boundary in both the source model and its derived stolen model. At the end of this work, we design many experiments to verify the effectiveness of the proposed method. The experimental results demonstrate that model ownership can be inferred with high confidence. Noting that our method does not require the training data to be private, and it is extremely costly for model stealers to reuse our method.

1 Introduction

As a digital product, deep neural network (DNN) models not only condense the intelligence of designers but also consume large amounts of training data and computational resources(He et al. 2016). However, most models are often exposed to the public to provide services such as machine translation(Freitag et al. 2021) or image recognition(Krizhevsky, Sutskever, and Hinton 2012), which allows attackers to steal models through exposed interfaces(Chandrasekaran et al. 2020). In this case, the stealer derives the surrogate model only from the API access to the source model(Jia et al. 2021). Thus, an ownership problem is proposed: How does the owner prove that a suspect model stole their intellectual property. Specifically, our goal is to determine whether the potentially stolen model is derived from the owner's model or dataset.

Stealers may derive and steal the intellectual property of their victims in a variety of ways(Fang et al. 2019). One prominent way is model distillation(Hinton, Vinyals, and Dean 2015), in which stealers use access to the model's predictions (e.g., via an API) to copy the model at a lower cost than that due to model training. In addition, the stealers have full access to the victim model(Uchida et al. 2017; Darvish Rouhani, Chen, and Koushanfar 2019), but not to the dataset, and when victims wish to open source their work for academic purposes, the stealers may fine-tune or prune the victim model.

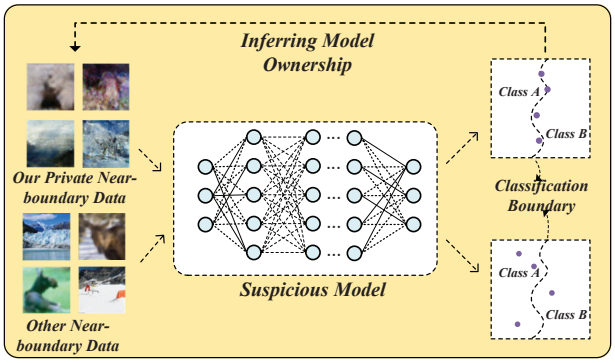


Figure 1: Inferring model ownership with near-boundary data.

In this paper, we propose near-boundary data, a special kind of samples distributed near the classification boundary. Model fingerprinting(Cao, Jia, and Gong 2021) uses adversarial examples to abstractly reflect model classification boundaries, and the same set of adversarial example inputs, with their induced changes in decision patterns, can be used to compare the similarity of model knowledge. However, this approach is fragile, and arbitrary operation on the model may destroy this property. Therefore, we do not directly compare changes in decision patterns, which are not confident, but rather compare the distance of the adversarial examples from the decision boundary. It is interesting to note that most adversarial examples are located near the decision boundary, i.e., they are close to the decision boundary. This property of adversarial examples is exploited by us to construct near-boundary data, and after testing we find that most model stealing methods cannot change this result, even if the classification result is affected, it still lies near the classification boundary. Motivated by this, it is traditional

to use near-boundary data as a fingerprint to verify ownership, even if it does not affect the accuracy of the model, but such a fingerprint is fragile and difficult to resist ambiguity attacks. We propose an ownership inference approach driven by near-boundary data. The idea is to construct private near-boundary data, and when verifying the ownership of a model, the model owner and the stealer provide their respective near-boundary data, and the closest to the classification boundary is inferred to gain ownership. An illustration of the idea is shown in Figure 1. The contributions of this paper are:

- We reveal the fragility of current ownership verification schemes and confirm the validity of data-driven inference of ownership.
- We propose to construct near-boundary data using adversarial examples to defend against model stealing.
- We design a DCGAN-based near-boundary data generator and construct private near-boundary data by it. We also propose a loss function to fine-tune the target classification boundary of the source model and increase the confidence of inferring ownership.
- We conduct extensive experiments on ResNet18, and the experimental results demonstrate the significant effect of near-boundary data in inferring model ownership.

2 Realated Work

Most of the current research is based on model watermarking approaches and all of them protect against steganographic threats by binding watermarking and model performance(Uchida et al. 2017). However, in most cases, adding noise to the model leads to degradation of the model performance(Maini, Yaghini, and Papernot 2021). Model watermarking(Jia et al. 2021; Liu, Dolan-Gavitt, and Garg 2018; Chen et al. 2021; Wang et al. 2019; Shafieinejad et al. 1906; Lao et al. 2022; Li et al. 2022). The purpose of watermarking is to verify model ownership rather than defend against model stealing, and embedding and extracting watermarks are important operations for model watermarking, such as embedding watermarks as parameters in neural networks(Uchida et al. 2017) and setting triggers as watermarks embedded in the model(Adi et al. 2018; Li et al. 2020). Moreover, to enhance the robustness of model modification, the idea of embedding passports(Fan, Ng, and Chan 2019; Zhang et al. 2020) was proposed, which operates by adding a particular passport layer to the model, and the threat of model stealing in the case of unavailable passports is reduced. Watermarking is widely used to verify ownership, which is susceptible to model performance, and an idea to extract knowledge from the model as a fingerprint(Cao, Jia, and Gong 2021; Lukas, Zhang, and Kerschbaum 2019; Pan et al. 2021; He, Zhang, and Lee 2019) and verify ownership is proposed. Researchers(Cao, Jia, and Gong 2021) want to be able to use decision boundaries as the fingerprint to verify ownership, and similarly, there is the use of properties of the activation function(Pan et al. 2021) to abstractly represent the knowledge of the second layer of the neural network as the model fingerprint. However, both watermarking and fingerprinting are vulnerable to ambiguity attacks(Fan, Ng, and Chan 2019; Li et al. 2019). Ambiguity attacks aim to cast doubt on ownership verification by forging alternative watermarks or fingerprints for DNN models. Intuitively, if an adversary can embed a second watermark or extract a second fingerprint on a model, there is a huge ambiguity in the IP ownership. It is important to note that ambiguity attacks are the concern of our proposed method.

Technical issues developed for model stealing involve model modification, including model fine-tuning, model pruning, and model compression. Fine-tuning(Uchida et al. 2017) is commonly used in transfer learning and involves re-tuning the model to change its parameters while maintaining performance. Model fine-tuning can derive many models by fine-tuning existing models. Pruning(Darvish Rouhani, Chen, and Koushanfar 2019) is a common method for deploying DNN by using parameter pruning to reduce the memory and computational overhead, while stealers may use pruning to remove watermarks or fingerprints. Model compression(Uchida et al. 2017) is common in distillation, which can significantly reduce memory requirements and computational overhead by distilling knowledge from a large model into a small model, and nowadays research(Hinton, Vinyals, and Dean 2015) can directly distill models using APIs without even needing the original training data, and thus is often used to derive models.

3 Theoretical Motivation

3.1 Limitations of Ownership Verification

Existing model IP protection methods focus on passive defense and only consider resistance to attack against model modifications. Model owners embed a watermark into a trained model or extract abstract model knowledge from it as a fingerprint (called **source model**), and when a model (called **suspicious model**) is suspected to have knowledge from the source model, the model owner can use the watermark or fingerprint to passively verify model ownership from the outside. Most works are based on the idea of designing different watermarks and fingerprints for verifying model ownership after the source model is stolen, but this is not highly robust. The drawbacks of model watermarking such as the impact on the performance and functionality of the source model, the additional cost caused by embedding the watermark are the critical points of the research work on watermarking. The purpose of model fingerprinting is to extract the intrinsic features that represent the model knowledge, compared to watermarking fingerprinting does not affect the source model, but fingerprinting is fragile because the model knowledge is susceptible to modifications, all fingerprinting methods try to find robust fingerprints that can withstand some modification attacks.

The goal of this paper focuses on another much-needed problem of watermarking and fingerprinting, ambiguity attacks, which are not concerned with removing watermarks and fingerprints to pass model ownership verification, but with forging additional watermarks and fingerprints to obfuscate ownership verification. Specifically, the stealer embeds a new watermark or extracts additional fingerprints on the source model to invalidate the original protection mea-

sures. Ambiguity attacks pose a serious threat to existing DNN IP protection methods, and there is research in the traditional digital watermarking field that suggests that robust watermark may not necessarily verify ownership unless the watermarking scheme is irreversible(Fan, Ng, and Chan 2019). In this paper, we argue that it is not sufficient to discuss stealing by verifying whether a suspicious model has a watermark or fingerprint specific to the source model, especially when ambiguity attacks occur, and therefore we propose to infer model ownership rather than verify it. This method is inspired by the ownership resolution proposed by Dataset inference(Maini, Yaghini, and Papernot 2021), which we will specifically discuss in the next section.

3.2 Model Ownership Inference

Dataset inference assumes that the knowledge of the source model comes from the training dataset. Whether the stolen model directly attacks the source model or is a byproduct of it, the knowledge of the stolen model is the knowledge contained in the source model. If the original training dataset is private, the model owner has a strong advantage over the adversary, and the source model performs much better in the original training data than in other datasets. Thus, ownership resolution can be obtained by combining statistical tests with estimating the distance from multiple data points to the decision boundary. This process is different from the traditional verification of model ownership, where an ownership resolution is inferred from the private dataset and the one with the largest resolution is considered to have the ownership. We can observe that dataset inference gives a notion of "most" and thus can effectively avoid ambiguity attacks.

Our work is inspired by dataset inference to verify model ownership, and we propose data-driven ownership inference instead of verifying ownership. We believe that ownership inference can solve the verification conflict problem while effectively proving the ownership attribution problem. Besides, data-driven ownership inference implies that it is only related to the input and output, and our method can work in both white-box and black-box environments.

Limitations of Dataset Inference. 1) The use of dataset inference assumes that the original training data is unavailable to the stealers and that the public dataset cannot be used to train the source model. However, in most realistic cases, only a tiny fraction of the work constructs private datasets for training models, and even this fraction has a narrow application point, which means that the risk of being stolen is low. Therefore, the dataset inference that relies on private datasets is used in a small range of practical applications and cannot be used in a significant way. 2) The core idea of the dataset inference is that the function of the source model outperforms other datasets on the training data, but there are cases where the function of the model may be similar while the structure and training data are different. Therefore the method may lead to misleading. (Li et al. 2022) verified this limitation and showed that the method produces results that are questionable.

We point out that the idea of inferring ownership needs to address the above issues, so we propose to construct private near-boundary data as a basis for inference and use the property that near-boundary data are close to the decision boundary to deal with misleading due to similarity of model functions. This is because even though the models are functionally similar, the decision boundaries are unlikely to be identical.

3.3 Threat Model

Based on existing works, our approach is deployed after the model is trained and ownership is inferred in a black-box environment. Our method does not focus on the process of model stealing, but aims at accurately inferring the ownership of the victim and identifying suspected model stealing. Most ownership verification techniques nowadays are black-box environments because model owners and attackers usually do not provide the full model. Our proposed method uses only the external API provided by the model to obtain decision results for near-boundary data to infer model ownership. In the usual assumption, there exists an official authority for arbitration, and when ownership is suspected for either model, victims and suspected adversaries can request and provide their respective private near-boundary data to the authority and infer ownership by our method. Note that our method can have effects in both white-box and black-box settings.

Problem Definition. We define a DNN classifier G as the source model, given an original training set D, and assume that the source model is a C-class DNN classifier, the output layer of the classifier is a softmax layer or some other decision layers, and the decision function $g_j(x)$ denotes the probability that a point x is classified into a j-th class, where $j = 1, 2, ..., c$. $Z_1, Z_2, ..., Z_c$ denotes the output of the full decision function of the source model classifier, whose results can be used as a basis for the classification boundary, thus

$$g_j(x) = \frac{exp(Z_j(x))}{\sum_{i=1}^{c} exp(Z_i(x))} \tag{1}$$

Note that the label y of the point x is inferred to be the class that has the maximum probability, e.g., $y = argmax_j g_j(x) = argmax_j Z_j(x)$.

Definition 1 (Classification Boundary). The classification boundary of a classifier is an abstract concept and we cannot describe it directly. Therefore we use the decision result of the classifier to reflect the classification boundary.

Generally speaking, finding data points that lie on the classification boundary is done by repeated random sampling of data points. Specifically, a data point is on a classification boundary if it satisfies Definition 1. However, simple repetitive sampling may take a lot of time to find even such data points. To solve such problems, we will discuss in the next section how to construct data points located on or near the classification boundary and make them private.

In previous work(Cao, Jia, and Gong 2021) the classification boundary is described as a data point that lies on the classification boundary if it has two equal maximum probabilities (or logit). Thus, we can abstractly represent a classification boundary using a set of data points that lie on the classification boundary. Note that classification boundaries exist

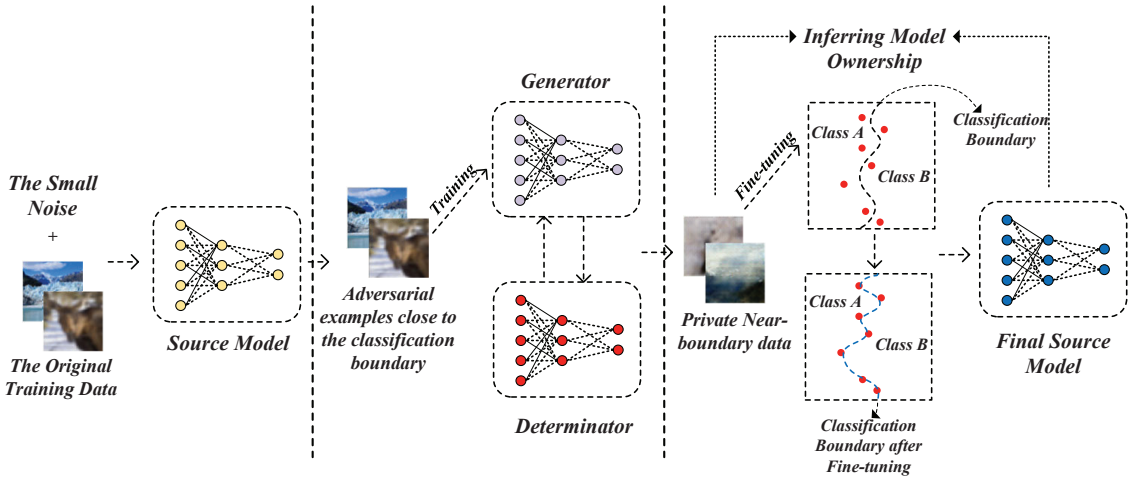


Figure 2: Construct private near-boundary data to infer model ownership.

objectively and are irrelevant to whether such data points can be found.

4 The Proposed Method

Based on the discussion in the previous section, we propose to construct near-boundary data and thus infer model ownership instead of verifying ownership. Specifically, as shown in Figure 2, our approach consists of three main phases, including (1) generating adversarial examples from the dataset as the initial near-boundary data, (2) designing a near-boundary data generator and privatizing and extending the near-boundary data, and (3) fine-tuning the target classification boundaries in the source model. The technical details are described in the following subsections.

4.1 Special Adversarial Examples

In this section, we will describe how to construct the first phase of the near-boundary data, which generates the basic data points close to the classification boundary. Before discussing the specific operation, we first summarize the concept of near-boundary data and give a definition.

Definition 2 (Near-boundary Data). Given a data point x, a threshold τ, if x satisfies $|g_i(x) - g_j(x)| \leq \tau$, where $i \neq j$ and $\min(g_i(x), g_j(x)) \geq \max_{k \neq i,j} g_k(x)$, then the data point x is called a near-boundary data.

In Section 5 and the Appendix, through extensive experiments, we demonstrate that near-boundary data are preserved in most of the model stealing techniques with their near-boundary features. Thus, near-boundary data can be used as evidence for ownership inference. Although the features of near-boundary data show significant effects in model IP protection, it is still difficult to obtain near-boundary samples because the natural near-boundary data is a relatively low percentage of the sample space. However, based on recent research(Cao, Jia, and Gong 2021) we know that adversarial examples are often used to determine the classification boundaries of classifiers. An adversarial example is essentially a data point that crosses the classification boundary of a classifier by adding a carefully designed noise. Specifically, adversarial examples have two classifications: the original classification and the target classification. The original classification is the original classification result of that example, while the target classification is the classification after noise is added. The adversarial examples' crossing of the classification boundary is reflected in the fact that the original classification remains in the vision but the classifier results in the target classification. In this paper, we argue that this feature can help to obtain more near-boundary data from adversarial examples. Therefore, we test several common methods for generating adversarial examples(Carlini and Wagner 2017; Goodfellow, Shlens, and Szegedy 2014; Kurakin, Goodfellow, and Bengio 2018) to construct near-boundary data. We focus on purposive methods because the desire to obtain data near arbitrary classification boundaries can be beneficial for our subsequent ideas.

Fast Gradient Sign Method (FGSM). FGSM(Goodfellow, Shlens, and Szegedy 2014) is the most classical way to construct adversarial examples by simply adding a noise bound ϵ to the initial examples, as in the following equation 2 where FGSM returns an adversarial example $x^{'}$:

$$x^{'} = clip(x - \epsilon \cdot sign(\bigtriangledown_x J(C, y^*; x))) \quad (2)$$

where $clip$ is the function to the function to project the adversarial example back to the feasible data domain, $\bigtriangledown_x$ is the gradient, J represents the loss function of classifier C, and y^* is the target label.

FGSM is very fast in generating adversarial examples, but its results depend on the choice of ϵ, so exploring different ϵ is the focus of using this method. In addition, we also test many advanced versions of FGSM such as IGSM(Kurakin, Goodfellow, and Bengio 2018) and RFGSM(Tramèr et al. 2017), which introduce iterative addition of noise and weak perturbations. IGSM iteratively makes examples cross classification boundaries until success, RFGSM increases the diversity of perturbations to generate adversarial examples in

a more refined way. In practical results we found that FGSM generates adversarial examples, although very fast, with a very low percentage of data located near the classification boundary, which is related to our specified target classification result. IGSM and RFGSM work better than FGSM, but are still not considered to satisfy our expectations. In extensive tests, we find that CW(Carlini and Wagner 2017) can generate a large number of samples near the classification boundary, and the specific test results are placed in the Appendix.

Carlini and Wagner's methods (CW). The CW method also adds noise to the adversarial examples, but it has three variants: CW-L_0, CW-L_2 and CW-L_∞. The different variants use different methods to measure the magnitude of the noise, among which CW-L_2 is the most effective in the experiments, so this paper uses it as the choice for generating the adversarial examples. Specifically, CW-L_2 iteratively searches for a slight noise for a given initial example to turn the example into an adversarial example. This idea allows the generated adversarial examples to be concentrated near the classification boundary, but accordingly CW-L_2 sacrifices efficiency.

In this phase, we select a portion of the data in the sample space of the source model as the initial examples to add slight noise, and target classification adversarial examples are generated in a purposive manner. Neither the training of the source model nor the original data are affected in any way at this phase, and the defender only needs to generate the adversarial examples in a purposive manner. However, the near-boundary data serve as important evidence for inferring ownership, and the direct generation of adversarial examples is also highly susceptible to replication by stealers. Therefore, we need to privatize the generated near-boundary data, which will be given in the following subsection.

4.2 Near-Boundary Data Privatization

Since the step of constructing near-boundary data by generating adversarial examples is very easy to reproduce, and most data used for model training nowadays are derived from public data. Therefore, we need to construct private near-boundary data from the public training data to prevent the model owner's near-boundary data from being easily imitated. In this paper, we hope that we can learn the features of the near-boundary adversarial examples by training a model and use it to generate new near-boundary data. This new data is not necessarily visually similar to the original data, but its original features as well as the added noise need to be learned, and the new examples generated based on the extracted features is also the near-boundary data for the source model. Therefore, in this paper we design a DCGAN-based(Radford, Metz, and Chintala 2015) near-boundary data generator and privatize the near-boundary data. Note that the generator takes the adversarial examples from CW-L_2 as input and outputs the privatized near-boundary data.

Specifically, the structure of DCGAN includes a discriminator D and a generator G, which is essentially a game process. The objective function of DCGAN is shown in the equation 3, which is a mutual adversarial process between the generator and the discriminator, where the generator generates as realistic an input sample as possible, and the discriminator discriminates whether the sample is a real sample or a fake sample as possible.

$$\min_G \max_D V(D,G) = \min_G \max_D E_{x\sim P_{data}(x)}[logD(x)] + E_{T\sim P_T(T)}[log(1-D(G(T)))] \quad (3)$$

Where T is the random noise used to generate the samples, the generator has no special requirements for the distribution of the noise T, but the common ones are Gaussian distribution and uniform distribution. The dimensionality of the noise T must be at least as high as the intrinsic dimensionality of the data stream shape to generate the near-boundary data we need. Note that the optimization process here is an alternating process.

Fine-tuning target classification boundary. Although the constructed near-boundary data are already all located near the target classification boundary, we still want the near-boundary data to be as close as possible to the target classification boundary. The closer the near-boundary data is to the target classification boundary, the greater the possibilities of inferring successful model ownership. In addition, although the generated near-boundary data are only owned by the model owner, the characteristics of the near-boundary may still be generalized for some models whose functions are easily generalized. Therefore, we propose to fine-tune the target classification boundary of the source model using the near-boundary data. Specifically, as shown in Eq. 4, $Loss_{FT}$ is the loss function for the target classification boundary, where n is the number of near-boundary data for this target classification boundary, $x_i^{'}$ is the generated near-boundary data, $g_t(\cdot)$ and $g_s(\cdot)$ denote the target classification probability and source classification probability (or logit), respectively. The essence of $Loss_{FT}$ is that we want the near-boundary data to be closer to the target classification boundary. The $Loss_{SM}$ is the loss function of the source model, and we design both of them to train the fine-tuned source model alternately, similar to the process of DCGAN, which is a game process.

$$Loss_{FT} = \frac{1}{n}\sum_{i=1}^{n}(g_t(x_i^{'}) - g_s(x_i^{'}))^2 \quad (4)$$

Fine-tuning the target classification boundary makes the association between the near-boundary data and the source model stronger. Note that since we only fine-tune the target classification boundary, and minimize the impact of fine-tuning on the source model by alternating fine-tuning. As shown in Table 1, the accuracy difference of the source model before and after fine-tuning is no more than $5pp$. Therefore, the effect of fine-tuning on the performance of the source model is minimal and can even be ignored, but it effectively improves the final ownership inference. More accuracy test results can be found in the Appendix.

4.3 Inference Ownership with Near-boundary data

In this paper, we construct private near-boundary data for model ownership inference by generating adversarial examples, privatizing near-boundary data, and fine-tuning target

Dataset	Acc. before Fine-tuning	Acc. after Fine-tuning
CIFAR-10	0.886	0.873
Heritage	0.879	0.856
Intel_images	0.794	0.786

Table 1: The influence of fine-tuning the target classification boundary on the source model.

classification boundaries in three phases. As the results discussed in Section 3.2, we believe that the past ideas of verifying model ownership have major limitations and most researches cannot defend against ambiguity attacks. Therefore, we propose the concept of inferring model ownership, which is a "most" idea. In a realistic scenario, we assume the existence of a third-party authority and agree on the target classification boundary. The victim submits the arbitration to the third-party institution and provides the near-boundary data, and the stealer also needs to provide the corresponding near-boundary data, and the third-party institution calculates the target classification boundary distance, respectively. In this paper, we consider that the owner of the near-boundary data holding the closest target classification boundary will gain ownership of the model. Note that since the near-boundary data is usually a set of data, it should be viewed based on the statistical results. In practice, we calculate the distance to the classification boundary for different sizes of near-boundary datasets on the source model, the stolen model and the uncorrelated model, then devise a hypothesis-test-based method to express the inferred confidence scores.

Hypothesis Test. We assume that the event C is the result of the computation of the private near-boundary data provided by the model owner on the suspected model, and the event C_S denotes the result of the computation of the near-boundary data provided by the stealer on the suspected model, or the result of the computation of the private near-boundary data provided by the model owner on the unrelated model. In this paper, we calculate the p-value of the assumption $H_0 : \mu > \mu_S$ ($H_1 : \mu \leq \mu_S$) and the effect size $\Delta\mu = \mu_S - \mu$, the larger the $\Delta\mu$, the higher the confidence of inference. If the p-value is lower than the predefined confidence score α, H_0 is rejected and the model being tested is said to be the stolen model. We repeat the statistical experiment 30 times to improve the confidence.

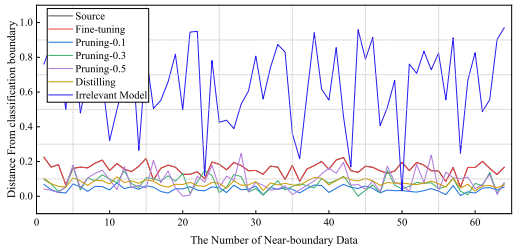
5 Experiments

5.1 Experiments Settings

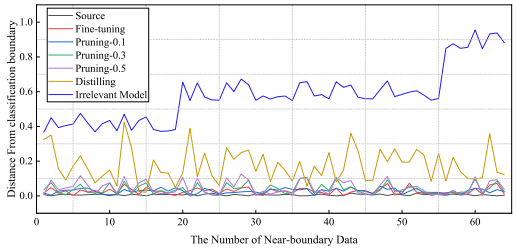
In this paper, we choose open source datasets including CIFAR-10(Krizhevsky, Hinton et al. 2009), Heritage[1] and Intel_image[2] as the basis for evaluation, and we choose ResNet18 as the source model for evaluation and VGG11 as an irrelevant model. The models used in this paper are trained on open source pre-trained models. Note that the all codes of this paper are open source[3].

Stolen Model Selection. We set up several common model stealing methods, including model fine-tuning, model pruning (with different pruning rates) and model distillation. We obtain the stolen model based on the source model, where the structure of the distillation model is from VGG11. Note that since we propose to fine-tune the target classification boundaries, we also set up multiple target classification boundaries and fine-tuned source models. More experimental settings, including model parameters, hyperparameters, etc. can be found in the Appendix.

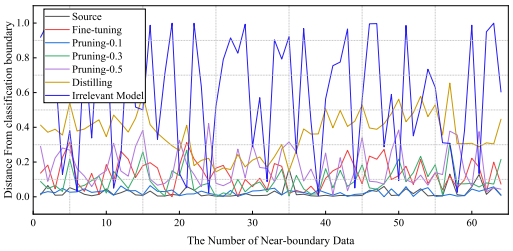
In this paper, we follow the evaluation of experimental results using the effect size $\Delta\mu$ and p-value. The results of specific experiments with 10 data sampled at a time are calculated for $\Delta\mu$ and p-value. Generally, the larger the $\Delta\mu$ the better, and the smaller the p-value the better, indicating that our method can satisfy the purpose of the model ownership inference.



(a) CIFAR-10



(b) Heritage



(c) Intel_image

Figure 3: The performance of near-boundary data on different datasets and different model stealing methods.

5.2 The Performance of Near-boundary Data

As shown in Fig. 3, the near-boundary data proposed in this paper exhibit properties close to the classification boundary in all stolen models. For example, our method exhibits no

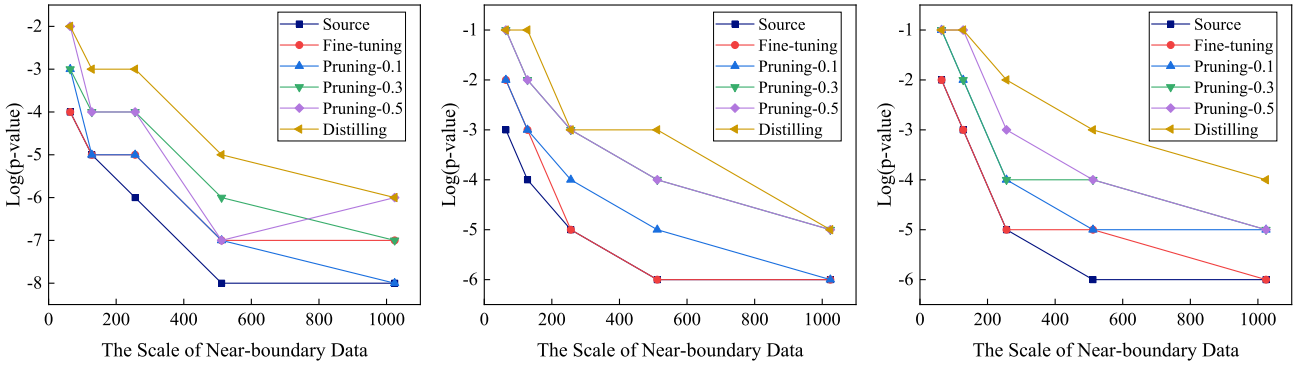
[1]https://datahub.io/dataset/architectural-heritage-elements-image-dataset

[2]https://www.kaggle.com/datasets/puneet6060/intel-image-classification

[3]Anonymous github

Dataset	Model Stealing	CB_1		CB_2		CB_3		CB_4		CB_5	
		$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value
CIFAR-10	Source	0.913	10^{-6}	0.954	10^{-6}	0.927	10^{-5}	0.967	10^{-5}	0.958	10^{-5}
	Fine-tuning	0.718	10^{-5}	**0.745**	10^{-6}	0.698	10^{-5}	0.692	10^{-4}	0.729	10^{-5}
	Pruning-0.1	**0.572**	10^{-5}	0.487	10^{-5}	0.458	10^{-5}	0.533	10^{-4}	0.512	10^{-4}
	Pruning-0.3	0.537	10^{-4}	0.497	10^{-4}	0.401	10^{-3}	0.428	10^{-4}	**0.587**	10^{-4}
	Pruning-0.5	0.545	10^{-4}	**0.614**	10^{-4}	0.506	10^{-3}	0.570	10^{-4}	0.484	10^{-3}
	Distilling	**0.372**	10^{-3}	0.297	10^{-3}	0.288	10^{-3}	0.308	10^{-3}	0.340	10^{-3}
Heritage	Source	0.876	10^{-5}	0.845	10^{-5}	0.859	10^{-4}	0.801	10^{-4}	0.837	10^{-5}
	Fine-tuning	0.815	10^{-5}	0.792	10^{-4}	0.824	10^{-4}	**0.833**	10^{-4}	0.784	10^{-4}
	Pruning-0.1	0.530	10^{-4}	**0.535**	10^{-3}	0.508	10^{-4}	0.486	10^{-3}	0.471	10^{-3}
	Pruning-0.3	**0.491**	10^{-3}	0.452	10^{-3}	0.469	10^{-4}	0.470	10^{-3}	0.427	10^{-4}
	Pruning-0.5	0.502	10^{-3}	**0.517**	10^{-3}	0.434	10^{-3}	0.451	10^{-3}	0.490	10^{-3}
	Distilling	0.329	10^{-3}	**0.365**	10^{-2}	0.238	10^{-3}	0.310	10^{-3}	0.274	10^{-3}
Intel_image	Source	0.859	10^{-5}	0.896	10^{-4}	0.872	10^{-4}	0.899	10^{-4}	0.914	10^{-4}
	Fine-tuning	0.717	10^{-5}	0.784	10^{-4}	0.752	10^{-4}	**0.791**	10^{-3}	0.709	10^{-4}
	Pruning-0.1	0.451	10^{-4}	0.522	10^{-4}	**0.539**	10^{-3}	0.472	10^{-3}	0.438	10^{-4}
	Pruning-0.3	0.407	10^{-4}	**0.415**	10^{-4}	0.346	10^{-3}	0.382	10^{-3}	0.395	10^{-3}
	Pruning-0.5	0.370	10^{-3}	0.395	10^{-3}	0.327	10^{-3}	0.360	10^{-3}	**0.458**	10^{-3}
	Distilling	0.336	10^{-2}	**0.395**	10^{-3}	0.360	10^{-2}	0.308	10^{-3}	0.287	10^{-2}

Table 2: The performance of inferring model ownership for different target classification boundaries.



(a) CIFAR-10 (b) Heritage (c) Intel_image

Figure 4: The scalability of Inferring Model Ownership on Near-Boundary Data.

relationship between the near-boundary data and the target classification boundary on the irrelevant models in the entire dataset. However, the near-boundary data perform well on various stolen models based on the source model, especially when the model modification method is more destructive such as model distillation, and still exhibits the property of being close to the target classification boundary. Note that the size of the near-boundary data is 64. In other words, our method is still effective on smaller-scale near-boundary data, and it is conceivable that our method will be more effective when the data size increases.

5.3 Defending against Model Stealing

In this section, we discuss the performance of our method in comparison with other near-boundary data on the stolen model. We simulate the near-boundary data that a stealer may provide, which consists of several components, including (1) near-boundary data selected from the original data, and (2) some adversarial examples generated by FGSM and CW. We perform hypothesis testing and calculate the $\Delta\mu$ and p-value on different datasets and stolen models for different target classification boundaries as in Section 4.3. As shown in Table 2, the p-value is significantly lower than 0.05 in all cases. In other words, our method inferring model ownership in different stealing methods is significantly effective in all cases. The experimental results show that using private near-boundary data is reliable for most model stealing methods, and we can claim a confidence score of at least 95% that the model is stolen. Model distillation is the largest challenge for ours and is equally a great challenge for other researches. In our experiments, it can be observed that our model can always label the distilled model as the stolen model, which proves the robustness of our method.

5.4 Evaluating Scaling

In this section, we test the scalability of near-boundary data of different sizes for the model ownership inference. Recall that our method requires sampling the data to perform hypothesis testing, and in general the larger the sample size the less adverse effects due to randomness in the testing process and the more confidence in ownership inference. As shown in Figure 4, p-value decreases as the size of the near-boundary data increases, which means that the more near-boundary data there are the more confidence in inference,

but this does not indicate that our method lacks robustness to small amounts of near-boundary data. We can observe from Figure 4 that even with a data size of 64, p-value is still less than 0.05, which proves that our method is equally effective for small data size.

6 Conclusion

In this paper, we discussed the limitations of the verification of model ownership in previous researches, and proposed replacing verification with model ownership inference. We argue that model stealing can be defended against from a data-driven perspective, i.e., if there is a measurable property of the data on the source model, then this property will also be inherited by the stolen model. Therefore, we constructed interesting near-boundary data for ownership inference and designed a method to generate adversarial examples using CW-L_2 iteratively adding slight noise, which is the initial near-boundary data. We trained a DCGAN-based near-boundary generator to privatize and extend the near-boundary data, and experimentally tested that the generator can significantly learn the features of the near-boundary data and generate new data. Finally, we designed a new loss function to fine-tune the classification boundary of the source model, and obtained the final version of the source model and the near-boundary data. We evaluated on the CIFAR-10, Heritage and Intel_image datasets and experimentally demonstrated that our method can infer model ownership with high confidence.

References

Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; and Keshet, J. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, 1615–1631.

Cao, X.; Jia, J.; and Gong, N. Z. 2021. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 14–25.

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, 39–57. Ieee.

Chandrasekaran, V.; Chaudhuri, K.; Giacomelli, I.; Jha, S.; and Yan, S. 2020. Exploring connections between active learning and model extraction. In *29th USENIX Security Symposium (USENIX Security 20)*, 1309–1326.

Chen, X.; Wang, W.; Bender, C.; Ding, Y.; Jia, R.; Li, B.; and Song, D. 2021. Refit: a unified watermark removal framework for deep learning systems with limited data. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 321–335.

Darvish Rouhani, B.; Chen, H.; and Koushanfar, F. 2019. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 485–497.

Fan, L.; Ng, K. W.; and Chan, C. S. 2019. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Advances in neural information processing systems*, 32.

Fang, G.; Song, J.; Shen, C.; Wang, X.; Chen, D.; and Song, M. 2019. Data-free adversarial distillation. *arXiv preprint arXiv:1912.11006*.

Freitag, M.; Foster, G.; Grangier, D.; Ratnakar, V.; Tan, Q.; and Macherey, W. 2021. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of the Association for Computational Linguistics*, 9: 1460–1474.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, Z.; Zhang, T.; and Lee, R. 2019. Sensitive-sample fingerprinting of deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4729–4737.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network (2015). *arXiv preprint arXiv:1503.02531*, 2.

Jia, H.; Choquette-Choo, C. A.; Chandrasekaran, V.; and Papernot, N. 2021. Entangled watermarks as a defense against model extraction. In *30th USENIX Security Symposium (USENIX Security 21)*, 1937–1954.

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

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2018. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, 99–112. Chapman and Hall/CRC.

Lao, Y.; Zhao, W.; Yang, P.; and Li, P. 2022. DeepAuth: A DNN Authentication Framework by Model-Unique and Fragile Signature Embedding. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI), Virtual Event*.

Li, H.; Wenger, E.; Shan, S.; Zhao, B. Y.; and Zheng, H. 2019. Piracy resistant watermarks for deep neural networks. *arXiv preprint arXiv:1910.01226*.

Li, Y.; Zhang, Z.; Bai, J.; Wu, B.; Jiang, Y.; and Xia, S.-T. 2020. Open-sourced dataset protection via backdoor watermarking. *arXiv preprint arXiv:2010.05821*.

Li, Y.; Zhu, L.; Jia, X.; Jiang, Y.; Xia, S.-T.; and Cao, X. 2022. Defending against model stealing via verifying embedded external features. AAAI.

Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, 273–294. Springer.

Lukas, N.; Zhang, Y.; and Kerschbaum, F. 2019. Deep neural network fingerprinting by conferrable adversarial examples. *arXiv preprint arXiv:1912.00888*.

Maini, P.; Yaghini, M.; and Papernot, N. 2021. Dataset inference: Ownership resolution in machine learning. *arXiv preprint arXiv:2104.10706*.

Pan, X.; Zhang, M.; Lu, Y.; and Yang, M. 2021. TAFA: A Task-Agnostic Fingerprinting Algorithm for Neural Networks. In *European Symposium on Research in Computer Security*, 542–562. Springer.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Shafieinejad, M.; et al. 1906. On the robustness of the backdoor-based watermarking in deep neural networks. CoRR (2019).

Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*.

Uchida, Y.; Nagai, Y.; Sakazawa, S.; and Satoh, S. 2017. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, 269–277.

Wang, B.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; and Zhao, B. Y. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, 707–723. IEEE.

Zhang, J.; Chen, D.; Liao, J.; Zhang, W.; Hua, G.; and Yu, N. 2020. Passport-aware normalization for deep model protection. *Advances in Neural Information Processing Systems*, 33: 22619–22628.