Inversion-Guided Defense: Detecting Model Stealing Attacks by Output Inverting

Shuai Zhou, *Graduate Student Member, IEEE*, Tianqing Zhu, *Member, IEEE*, Dayong Ye, Wanlei Zhou, *Senior Member, IEEE*, and Wei Zhao, *Fellow, IEEE*

***Abstract*— Model stealing attacks involve creating copies of machine learning models that have similar functionalities to the original model without proper authorization. Such attacks raise significant concerns about the intellectual property of the machine learning models. Nonetheless, current defense mechanisms against such attacks tend to exhibit certain drawbacks, notably in terms of utility, and robustness. For example, watermarking-based defenses require victim models to be retrained for embedding watermarks, which can potentially impact the main task performance. Moreover, other defenses, especially fingerprinting-based methods, often rely on specific samples like adversarial examples to verify ownership of the target model. These approaches might prove less robust against adaptive attacks, such as model stealing with adversarial training. It remains unclear whether normal examples, as opposed to adversarial ones, can effectively reflect the characteristics of stolen models. To tackle these challenges, we propose a novel method that leverages a neural network as a decoder to inverse the suspicious model's outputs. Inspired by model inversion attacks, we argue that this decoding process will unveil hidden patterns inherent in the original outputs of the suspicious model. Drawing from these decoding outcomes, we calculate specific metrics to determine the legitimacy of the suspicious models. We validate the efficacy of our defense technique against diverse model stealing attacks, specifically within the domain of classification tasks based on deep neural networks.**

***Index Terms*— Model stealing attacks, model IP protection, fingerprinting, model inversion, adversarial examples.**

I. Introduction

MACHINE learning models have become widely used tools in daily life, providing significant financial value, like ChatGPT [1] and Google's cloud vision API [2]. Training these models is a time-consuming and resource-intensive process, requiring vast amounts of valuable training data and substantial computational power [3]. However, when well-trained models are uploaded to the cloud for various services, there is a risk of malicious users launching attacks to steal the model [3], [4]. These attacks involve unauthorized copying of a model, harming the interests of the original model owner. Therefore, it is important for the model owner to be able to claim ownership of the models and detect the stolen models.

In recent years, several methods that defend against model stealing attacks have been proposed [5], [6], [7], [8], [9], [10]. These defenses can be broadly categorized into two types: proactive defense and reactive defense. The former, known as proactive defense, focuses on preventing model stealing attacks before they occur [6], [7]. It involves techniques such as poisoning attacks [6] to contaminate the data used by attackers to train surrogate models. On the other hand, reactive defense focuses on responses after an attack has taken place. It involves measures to detect model stealing attacks and verify the ownership [8], [9], [10]. In this paper, we focus on the reactive defense category. Furthermore, existing reactive defenses include watermarking-based methods implemented during the training procedure [8] and fingerprinting-based methods that are applied during the inference stage of trained models [9], [10], [11]. These fingerprinting-based defenses insert adversarial perturbations into the validation sets to check the change of the predicted labels in the testing phase.

Defenses against model stealing attacks can also be implemented under white-box and black-box setting [3]. In the white-box setting, defenders have complete access to the suspicious model's information and internal outputs. In contrast, the black-box defenders can only query the model and access the outputs of the final layer. Chen et al. [5] demonstrated that it is easier for white-box defenders to identify stolen models due to the richer information provided by latent outputs.

When revisiting previous defense strategies, it becomes evident that watermarking-based approaches are intrusive and negatively impact the primary task's performance. On the other hand, non-intrusive defenses against model stealing attacks typically require the creation of fingerprints, such as adversarial examples, for the victim DNN models. However, an intriguing question arises: can a stolen model be identified solely by analyzing the models' output without relying on fingerprints? We argued that the output of a stolen model exhibits distinct patterns when compared to that of a legitimate model, which can serve as a basis for detecting stolen models. Consequently, this paper aims to introduce a novel category

Manuscript received 10 October 2023; revised 3 February 2024; accepted 29 February 2024. Date of publication 14 March 2024; date of current version 2 May 2024. This work was supported by the Australian Research Council Linkage Project under Grant LP220200808. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Grigorios Loukides. *(Corresponding author: Tianqing Zhu.)*

Shuai Zhou, Tianqing Zhu, and Wanlei Zhou are with the Faculty of Data Science, City University of Macau, Macau, China (e-mail: zhoushuaincepu@126.com; tqzhu@cityu.edu.mo; wlzhou@cityu.edu.mo).

Dayong Ye is with the Centre of Cyber Security and Privacy and the School of Computer Science, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: dayong.ye@uts.edu.au).

Wei Zhao is with the Faculty of Computer Science and Control Engineering, Shenzhen University of Advanced Technology, Shenzhen 518055, China (e-mail: weizhao86@outlook.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TIFS.2024.3376190, provided by the authors.

Digital Object Identifier 10.1109/TIFS.2024.3376190

of defense methods that are distinct from watermarking or fingerprinting techniques. Our new defense is non-intrusive, as it does not require modifications during the training phase, distinguishing it from watermarking-based defenses. Additionally, our defense does not depend on specific examples, such as adversarial examples, setting it apart from fingerprinting-based methods. More specifically, our defense is founded on the analysis of the hidden pattern within the model's output (i.e., the confidence vectors) since we claim that the hidden pattern possesses the capability to characterize the stolen model. We initiate the design of this novel defense strategy by analyzing the challenges encountered in previous defense approaches:

A. Watermarking: Decreased Performance

One significant challenge with existing watermarking-based approaches is that they often require modifying the victim model training process by inserting a backdoor. This backdoor is activated when a predetermined trigger is detected in the input, causing the model to produce unusual outputs [8]. Modifying the training procedure inevitably affects the performance of the main tasks, leading to a decrease in the victim model's classification accuracy. Additionally, for some large-scale models, the retraining process to insert backdoors can be time-consuming.

B. Fingerprinting: Requiring Special Examples

To ensure zero loss in the performance of the victim model, fingerprinting-based methods are proposed as complementary defenses to watermarking defenses, which verify the ownership of victim models based on certain distinctive characteristics [9], [10], [11]. Adversarial examples [9] and [11] and misclassified examples [10] are often selected as these special characteristics (i.e., fingerprints) to protect intellectual property.

In these approaches, defenders typically only consider the predicted labels of special examples and overlook the valuable information contained in the confidence scores produced by the victim model. We believe that the distribution of confidence scores can provide useful information for determining stolen models. Importantly, methods relying on special examples are vulnerable to adaptive techniques like adversarial training. If the attacker is aware of these defenses, they can easily evade them by adaptively modifying attacks [9], [12].

C. A Paradigm Shift From Existing Defenses

Due to the limitations of previous defenses, this paper proposes a new type of defense to address the problem of model stealing in a black-box setting. This approach is non-intrusive and relies solely on normal examples. Motivated by the observation that fingerprinting-based methods primarily focus on predicted hard labels without considering confidence scores in output vectors, we argue that the unique patterns hidden within these confidence vectors can serve as distinctive features to effectively characterize stolen models, offering new insights for our new defense strategies.

In detail, existing fingerprinting-based defenses may not have fully utilized the information provided by model outputs during the detection of stolen models due to their exclusive focus on hard labels. More valuable information within the confidence scores is ignored. Consequently, we argue that incorporating complete confidence vectors can achieve similar effects using normal examples, eliminating the need for special examples. By comparing the two confidence vectors produced by a stolen model and a legitimate model, certain distinctive patterns can be revealed. Even if these two confidence vectors yield the same predicted label, their hidden patterns in the confidence scores may differ, thereby assisting in the differentiation between the stolen model and the legitimate model.

Nevertheless, an important question naturally arises:

"What indicators can effectively depict the special hidden patterns in the confidence vectors?"

To answer this question, we introduced a method for extracting information embedded within the confidence scores of output vectors (also known as soft labels). To be specific, we drew inspiration from model inversion attacks [13] and employed a neural network to reverse-engineer the output vectors. The resulting reconstructions can serve as a proxy for the extracted features, characterizing the victim model. Building upon these reconstructions, we introduced three new indicators for normal examples, denoted as hidden pattern indicators. These indicators form the core of our approach, known as Inversion-guided Defense (IGD). In essence, while fingerprinting-based defenses focus on variations in predicted labels, IGD places emphasis on extracting features from the victim or suspicious models. The richer information within the extracted features enables our IGD to use normal examples as test cases, eliminating the need to generate adversarial or misclassified examples.

Additionally, it incorporates a decision tree model that automatically assigns significance to various hidden pattern indicators and calculates appropriate thresholds for each, resulting in a more precise and robust approach to threshold determination. In summary, our proposed method contributes to advancing defensive strategies in this domain. Our method proves particularly valuable in a practical commercial scenario where the defender is not the model owner and lacks full access to the victim model. This often occurs in institutes such as hospitals, where sensitive data is used to train models but expertise in information security is lacking. Consequently, these institutes have to seek external assistance to deploy defensive strategies against model stealing attacks and safeguard the intellectual property of their valuable models. In such cases, the defender must implement the defensive strategies with limited access to the protected model. This highlights the practical nature of our approach, as it allows the defender to effectively protect the model even without full knowledge of its internal workings.

However, it should be noted that our defense approach is specifically tailored to protect classification models that utilize deep neural networks, such as convolutional neural networks. For other types of models, such as language models and image generation models, significant modifications would be necessary in order to uncover hidden patterns that can be used to effectively detect model stealing behaviors.

D. Contributions

Overall, our contributions can be summarized as follows:

- We present a novel class of defenses against model stealing attacks by leveraging hidden pattern analysis within the confidence vectors. Our approach stands apart from previous methods like watermarking and fingerprinting defenses. Unlike these approaches, our defense mechanism overcomes performance degradation and eliminates the need for special examples.
- To the best of our knowledge, this work is the first to analyze the stealing behaviors of model stealing attacks on normal examples, as opposed to previous methods relying on special examples like adversarial examples. We aim to uncover any special characteristics of stolen models and investigate whether it is possible to analyze these special characteristics based on normal examples.
- We exploit a novel method (i.e., model inversion attack) to uncover the hidden pattern embedded in the confidence scores of output vectors. Previous defenses often overlook these confidence scores, but our approach leverages a decoder to reveal the hidden patterns in these confidence scores by introducing three hidden pattern indicators.
- We introduce an automatic identifying method based on multiple indicators that can reflect the special characteristics. By recognizing that different indicators may have varying degrees of importance in detecting model stealing attacks, our approach enables the development of more precise defense methods.

II. Related Work

A. Model Stealing Attacks

Attacks targeting intellectual property in machine learning field can be categorized into two main types: concerning exact model properties and approximating model behaviors [3]. The former typically seeks to extract valuable properties of a given machine learning model, including the model's training hyper-parameters [14], its architectural design [15], and the specific learned parameters [16]. The motivation behind this approach is rooted in the fact that configuring the model structure and critical hyperparameters, such as the learning rate, for Deep Neural Networks (DNNs) demands a high level of expertise and specialized knowledge. Furthermore, the learning of these parameters generally involves substantial computational resources and time investments. However, extracting exact model properties can be challenging. Therefore, attackers often opt to create copies with similar functionality for financial gain [4], [17]. This paper primarily focuses on defenses against attacks that aim to closely approximate the behaviors of classification models. Several commonly used terms in the field of model stealing attacks include:

- **The victim model**: The target model that attackers intend to approximate.
- **The stolen model**: A duplicated version of the victim model, designed to achieve performance similar to that of the victim model.
- **The legitimate model**: A model trained from scratch, which, despite delivering similar performance, is unrelated to the target model.
- **The suspicious model**: The target model in the verification procedure of ownership, which can be a stolen model or a legitimate model.

B. Proactive Defenses Against Model Stealing Attacks

To protect the intellectual property of the victim model, certain defensive measures are often deployed proactively before potential attacks [6], [7], [18]. Oreknody et al. [6] propose a method involving data poisoning attacks to manipulate the model's outputs, regardless of the presence of the attacker. This approach leads to a poisoning objective in the training of stolen models, aiming to deceive the attacker and make the stolen model less effective. APMSA [7] also inserts perturbations to the model's predictions to hide the rich information and hinder the effective training of stolen models. In the craft of the perturbations in APMSA, the hard labels are kept unchanged, but the perturbations push the model's predictions closer to the decision boundary. In summary, these proactive methods modify the model's predictions regardless of the presence of an attacker, leading to a partial loss of information within the predictions.

C. Reactive Defenses Against Model Stealing Attacks

In this paper, we assume that model stealing attacks have already been successfully executed. In this context, two categories of defenses are commonly used [9], including watermarking-based methods (integrated during the training process) and fingerprinting-based methods (implemented post-training).

1) Watermarking-Based Defenses: One of the most straightforward defenses involves embedding a secret watermark, such as backdoors, into the victim model during the training process, which can be used in the verification of the ownership. This backdoor can be activated by a trigger, misleading both the victim model and the stolen model to predict a predetermined label. In detail, the defender can use two strategies to embed the watermark. Firstly, the secret watermark can be inserted into the parameters of the victim models [19]. Alternatively, a backdoor can be integrated into the victim model through training on a triggered dataset [8]. In this case, the trained model will provide a wrong prediction for inputs containing a predefined trigger.

However, both these two strategies involve the training procedure, and the parameters will be modified compared to the naturally trained victim models. As a result, the performance of the main task (e.g., the classification accuracy) might be sacrificed [8]. Such accuracy degradation might be unacceptable, particularly in critical domains [9]. Moreover, retraining well-established, large-scale models could be impractical. As a result, post-training defenses are receiving more attention.

2) Fingerprinting-Based Defenses: Fingerprinting defenses focus on identifying examples that can uniquely reflect the victim model's distinct characteristics. These defenses often leverage the transferability of adversarial examples, generated

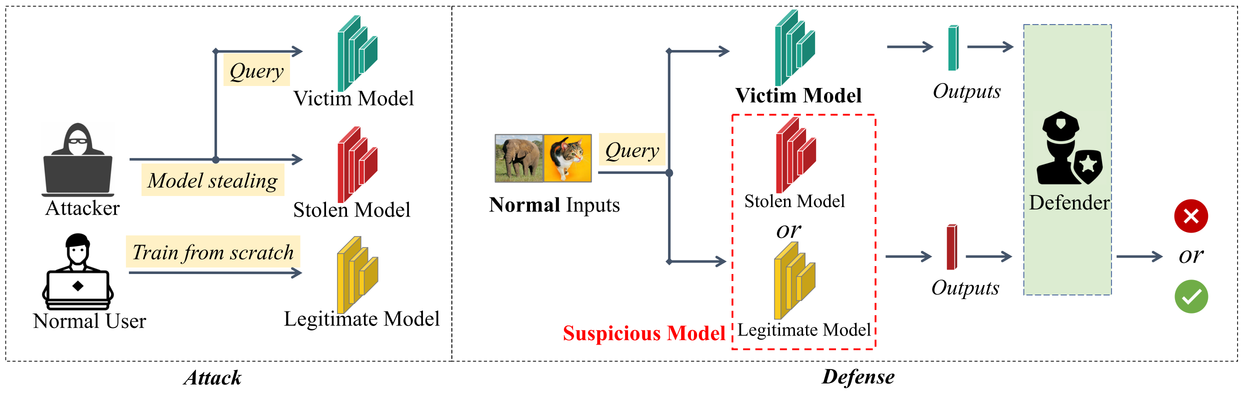

Fig. 1. Procedure of defending against model stealing attacks: The attacker attempts to replicate a stolen model, while the normal user trains a legitimate model from scratch. The defender's objective is to judge whether the model is a stolen model or not.

against the victim model and capable of misleading stolen models as well. For instance, IPGuard [9] extracts fingerprinting points that characterize the victim model's classification decision boundary, uniquely representing a DNN model. Similarly, FUAP [11] employs universal adversarial examples (UAEs) as fingerprinting points, asserting that UAEs belong to a low-dimensional subspace, which are more consistent with that of the stolen versions. DeepJudge [5] presents a testing framework with various testing indicators to determine ownership of a suspicious model in both white-box and black-box scenarios. When the defender has solely black-box access to the suspicious model, there are two testing indicators (i.e., RobD and JSD) involved, both of which are computed based on adversarial examples. Hence, DeepJudge in the black-box setting serves as a specialized case of fingerprinting defenses relying on adversarial examples.

Nevertheless, the effectiveness (including transferability) of adversarial examples is crucial in fingerprinting defense. However, reaching a balance between the effectiveness of adversarial examples and their efficiency remains an ongoing challenge [9]. Some efficient methods for generating adversarial examples tend to achieve sub-optimal success rates in attacks [20], while certain effective attack strategies typically require substantial time investments [12], [21].

In addition, fingerprinting defenses built on adversarial examples exhibit vulnerability due to their susceptibility to adversarial defenses (e.g., adversarial training). To tackle this issue, Guan et al. [10] propose sample correlation (SAC), employing misclassified examples from normal instances (rather than adversarial examples) to detect model stealing attacks. SAC increases the robustness of defenses against some adaptive attackers. However, to guarantee detection accuracy, these misclassified examples are defined as examples that are wrongly classified by the victim model and its derivatives using knowledge distillation techniques simultaneously. In other words, SAC needs to train some additional models with knowledge distillation strategies from the victim model, which might influence the efficiency of this method. Furthermore, Zheng et al. [22] assume that the defender has access to the internal parameters of the target model and project the front-layer weights onto a random space defined by the model owner to identify the stolen models. However, this approach is not suitable for scenarios involving a third-party defender who lacks access to the model's weights.

D. Model Inversion Attacks

Model inversion attacks aim to recover private data that was used to train the target model [23], [24]. Most model inversion attacks employ Generative Adversarial Networks (GANs) to synthesize previously unknown data that can optimize the probability of generating a predefined response [23]. In contrast, Yang et al. [13] proposed to use a separate network as the decoder to invert the target model's outputs. This decoder is trained to reconstruct the original input from predictions. Throughout the learning process, the attacker aims to identify an optimal inversion model $\mathcal{I}$, which minimizes the defined objective function $\mathcal{L}(\mathcal{I})$. This objective function is designed to measure the difference between the reconstructed data $I(F(x))$ and the original data x. One common metric used to quantify this reconstruction error is the mean squared error (MSE) within the image domain. In this scenario, the reconstruction of private data can be likened to an upsampling process. Consequently, the attacker must make full use of the information contained in the predictions.

E. Discussion

Our method differs from existing defenses against model stealing attacks in the following ways:

- Unlike **proactive defense approaches**, our method assumes that attacks have already occurred and focuses on passive defenses. It does not require perturbing the model's outputs to prevent model stealing, thus avoiding any loss of information within the predictions.
- Unlike **watermarking-based methods**, our method does not involve the training phase of the model. Therefore, it is a non-intrusive defense approach that avoids the need for retraining, making it more flexible.
- Unlike **fingerprinting-based methods**, our approach does not rely on any specific samples. Instead, it analyzes hidden patterns in normal outputs, eliminating the time-consuming process of searching for specific samples and making it more efficient.

III. Problem Formulation

Deep learning models have been applied in various industries. In this paper, we focus on the models used in image classification tasks. Deep learning models (referred to as F) that provide online services typically take the client's data x as input and generate a prediction $c = F(x)$ as a response. Formally, the prediction c should be a vector with a length of k, which can also be called a confidence vector. Furthermore, c_i represents a confidence score corresponding to the i-th class, and the sum of all c_i should equal 1.

A. Threat Model

In our setting, the attacker is assumed to have white-box access to the target victim model. In other words, the attacker not only has the capability to query the victim model and obtain the prediction vectors but also has knowledge of the parameters of the victim model. With model stealing attack methods, e.g., fine-tuning and model extraction, the attacker can copy a model with similar functionality as the victim model. As illustrated in Fig. 1, given a target victim model F_v, the attacker will copy a stolen model F_s by querying the victim model. Instead, a normal user typically trains a model from scratch, which is denoted as the legitimate model F_l.

Considering the protection of intellectual property, the defender's objective is to determine whether a suspicious model has been stolen from the victim model by the attacker. The workflow diagram is depicted in Fig. 1. The defender aims to distinguish between stolen models and legitimate models by querying them. It is important to note that the defender can only query the victim model and suspicious models (e.g., stolen or legitimate models) in a black-box manner. The defender is unable to access the weights of the victim model, which is common in the real world. For instance, hospitals often have sensitive patient data and may prefer to train their own models. However, due to limited expertise in information security, they may not be able to effectively implement countermeasures against model stealing attacks. In such cases, third-party companies can serve as defenders, assisting in the protection of the models while having limited access to them.

Additionally, the defender is limited to querying normal examples (as opposed to adversarial examples). Therefore, the defender must explore the valuable information present in the model's output and base its determination on these output.

B. Problem Definition

Formally, we consider a detection game, wherein the defender must guess whether a suspicious model is stolen or legitimate. This game presents a generalized framework of the defenses based on hidden pattern indicators [5], [10].

Game 1 (Stealing Detection Game): This game proceeds between a defender and a challenger. The defender can query the victim model F_v in a black-box manner, and has access to the training data distribution X.

1) The challenger trains two models (F_0, F_1), where F_0 is a stolen version of F_v and F_1 is a legitimate model trained from scratch.
2) The challenger randomly selects a bit $b \in \{0, 1\}$, and provides access services to model F_b for the defender.
3) The defender randomly selects a set of normal data points T from the data distribution X.
4) The defender queries the victim model and the suspicious model F_b simultaneously with the test cases $x \in T$.
5) The challenger return the responses $F_b(x)$ to the defender.
6) Based on the output $F_v(x)$ and $F_b(x)$, the defender compute some hidden pattern indicators $(M_1, M_2, \cdots)$ to make the final guess $b' = 0$ or 1.
7) The defender wins the game if $b' = b$.

In this game, the suspicious model F_b is considered stolen when $b = 0$ or legitimate when $b = 1$. The hidden pattern indicators $(M_1, M_2, \cdots)$ are computed based on normal examples. Existing indicators such as RobD and JSD [5] have been developed to assess the distinction between confidence vectors of stolen and legitimate models, especially in relation to adversarial examples. These indicators, previously introduced by Chen et al. [5], can be utilized in defending against model stealing attacks, particularly when the defender has limited black-box access to the suspicious model. In this paper, we also present some new hidden pattern indicators designed to work with normal examples for detecting model stealing, with detailed explanations provided in Section IV-B2.

Success Metric: For a suspicious model F_b, we consider a true-positive or true-negative outcome as a correct prediction. This means that the defender wins the game when $b' = b$, regardless of whether b is 0 or 1. When provided with a set of suspicious models by the challenger, we calculate both the true-positive rate and the false-positive rate over multiple iterations of this game. These values are then used to compute a commonly used metric, the Area Under the ROC Curve (AUC), for assessing the defender's performance. A higher AUC value indicates a stronger defender.

IV. Uncovering: Depicting the Hidden Patterns

A. Identifying Inadequacies of Existing Testing Indicators

Previous studies, like fingerprinting-based defenses, have examined suspicious models' outputs to detect model stealing. They argue that distinctions exist between the outputs of stolen models and legitimate models. As a result, researchers have proposed testing metrics that leverage model outputs to assess these distinctions. For instance, in the black-box scenario, Chen et al. [5] introduce RobD and JSD based on adversarial examples. Specifically, RobD quantifies the robustness distance, i.e., the differences in attack accuracy of adversarial examples across stolen and legitimate models. Furthermore, JSD (Jensen-Shannon distance) measures the similarity between confidence vectors of the suspicious model and the victim model. In addition, Guan et al. [10] introduce SAC, which assesses sample correlations via computing the correlations between the outputs of models. However, these testing indicators are designed for adversarial and misclassified examples. We further studied the performance of these testing indicators when applied to normal examples.

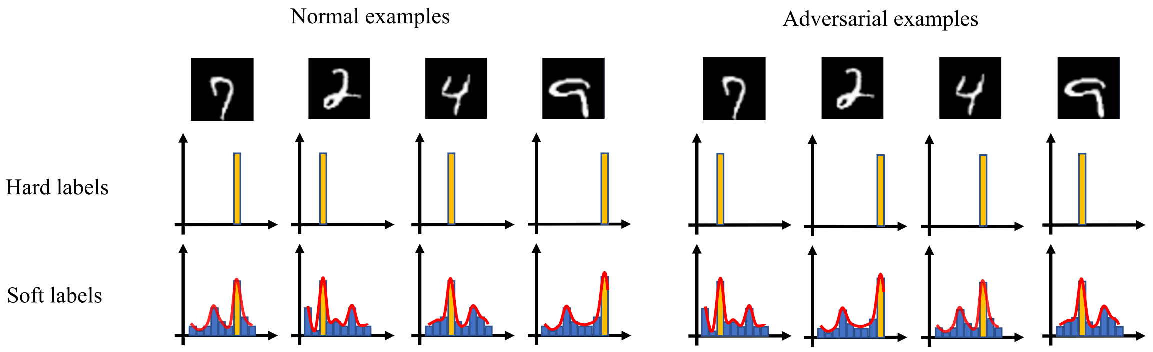

Fig. 2. The intuitive comparison of hard labels and soft labels of the input data.

1) Robustness Distance (RobD): Given a model F and a test set $T = \{x_1, x_2, \cdots, x_n\}$ with labels of $Y_t = \{y_1, y_2, \cdots, y_n\}$, some adversarial examples $T' = \{x'_1, x'_2, \cdots, x'_n\}$ can be generated and they can mislead the model F to give a wrong predicted label. That is, $F(x'_i) \neq y_i$. Based on these adversarial examples T', the robustness property of F can be defined with the classification accuracy:

$$Rob(F, T') = \frac{1}{|T'|} \sum_{x'_i \in T'} \mathbb{1}\{F(x'_i) = y_i\}.$$

For a given victim model F_v and a suspicious model F_{sus}, RobD can be defined as the absolute difference between their robustness properties on the set of adversarial examples T':

$$RobD(F_v, F_{sus}, T') = |Rob(F_v, T') - Rob(F_{sus}, T')|.$$

Undoubtedly, the effectiveness of RobD would experience a significant decrease when replacing the test cases T' with normal examples. This is attributed to the fact that most of the normal examples are accurately classified by both the victim and suspicious models, resulting in only subtle differences in their classification accuracy. Consequently, the direct application of RobD to test cases comprising normal examples is unfeasible. Furthermore, it is important to highlight that RobD exclusively employs hard labels ($l = \arg\max_i\{c_i\}$) in the predictions c of the test cases, namely, the adversarial examples. Regrettably, this approach overlooks the valuable soft labels containing additional confidence scores. To visually emphasize the distinction between hard and soft labels, we provide some examples in Fig. 2. Soft labels provide more comprehensive information, including the confidence score distribution, compared to hard labels. Therefore, using testing indicators that utilize soft labels can improve the performance of model-stealing detection when analyzing model outputs.

2) Jensen-Shannon Distance (JSD): JSD, a derivative of the Kullback-Leibler (KL) divergence, serves as a tool to measure the similarity between two probability distributions. A smaller JSD signifies a larger similarity [5]. As mentioned above, the deep learning model outputs a confidence vector, essentially a soft label, which can be considered a distribution of the confidence scores. Therefore, JSD can be applied to quantify the difference between the victim model's output and the suspicious model's output, and its formal expression is presented below:

$$JSD(F_v, F_{sus}, T') = \frac{1}{2|T'|} \sum_{x'_i \in T'} KL(F_v(x'_i), F_{avg}) + KL(F_{sus}(x'_i), F_{avg})$$

where $F_{avg} = (F_v(x'_i) + F_{sus}(x'_i))/2$. Chen et al. [5] have established the validation that the JSD values attributed to stolen models when evaluated on adversarial examples are noticeably lower than those associated with legitimate models. This discrepancy arises from the inherent similarity between the stolen model and the victim model. In simpler terms, the JSD value of a stolen model, denoted as $JSD(F_v, F_s, T')$, is expected to be small, while the JSD value of a legitimate model, indicated as $JSD(F_v, F_l, T')$, should be larger.

It appears that JSD exploits the information embedded within the soft labels. However, we argue that the effectiveness of the JSD indicator could potentially stem solely from the divergence in the predicted hard labels produced by the stolen models or the legitimate models. To elaborate, the hard label l corresponds to a maximal confidence score in the confidence vector. In mathematical terms, $c_l = max\{c_1, c_2, \cdots, c_k\}$. Notably, c_l generally is much larger than other confidence scores. For a given adversarial example x', we use c^v to represent the confidence vector made by the victim model F_v and denote that of the suspicious model as c^{sus}. In addition, we introduce $c^{avg} = (c^v + c^{sus})/2$, representing the average of these two confidence vectors.

$$KL(F_v(x'_i), F_{avg}) = KL(c^v, c^{avg}) = \sum_{i=1,\ldots,k} c^v_i \cdot \ln \frac{c^v_i}{c^{avg}_i} \tag{1}$$

Since the test case set T' consists of adversarial examples rather than normal examples, it is plausible that both the victim model and the stolen model F_s might yield identical predicted labels l. Nevertheless, because x' is generated against the victim model, it might not work against the legitimate F_l. Consequently, x' could potentially function as a normal example against the context of F_l. As shown in Fig. 2, the predicted label of adversarial examples (for F_v) is different from that of normal examples (for legitimate model F_l). Mathematically, $arg\max_j\{c^v_j\} = arg\max_j\{c^s_j\}$, while $arg\max_j\{c^v_j\} = arg\max_j\{c^l_j\}$. As shown in Equation 1,

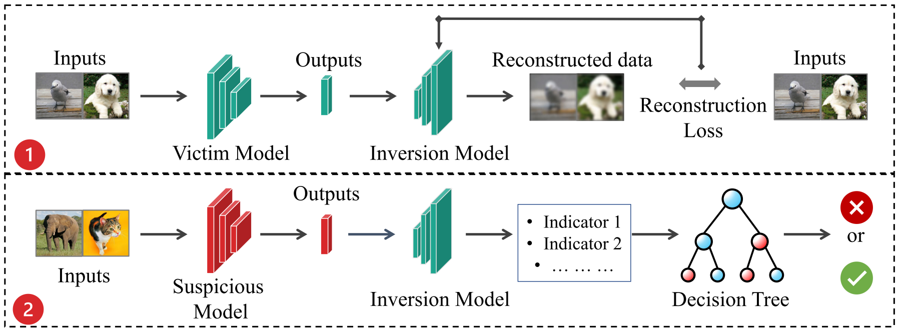

Fig. 3. Our defense method overview consists of two primary sub-procedures. Firstly, we train an inversion model tailored to the victim model's characteristics, enabling it to reconstruct the training data from the model outputs, specifically the confidence vectors. The primary training objective focuses on minimizing the reconstruction loss. Secondly, we select inputs at random and feed them into the suspicious model. Subsequently, we employ the trained inversion model to generate reconstructed versions of these randomly selected inputs. Based on these reconstructed data, we compute various hidden pattern indicators to uncover hidden patterns within the outputs of the suspicious model. These hidden pattern indicators are then utilized by a decision tree model. Consequently, the decision tree model determines whether the suspicious model should be classified as stolen or legitimate.

since l is the label of x' predicted by F_v, c_l^v is much larger than other confidence scores in c^v and c_l^{avg} due to the special characteristics of confidence vectors. As a result, we can say that $KL(F_v(x_i'), F_{avg})$ is always dominated by $c_l^v \cdot \ln \frac{c_l^v}{c_l^{avg}}$ when compared to other terms in Equation 1. Consequently, only the maximal confidence score within the confidence vector substantially contributes to the information of computing KL. This scenario is similar to the setting relying on the hard label rather than the soft label. Therefore, we argue that the efficacy of JSD might also rely on the hard label.

However, when JSD is computed on normal examples, $F_v(x_i')$ and F_{avg} would share a predicted label. With c_l^v and c_l^{sus} being large, there is no single term dominating the KL value. Each confidence score will contribute to the KL value, thereby allowing effective exploration of the potential of soft labels. Despite this, it remains crucial to conduct a more in-depth investigation into whether JSD on normal examples is effective in countering model-stealing attacks.

B. How to Depict the Hidden Pattern?

In this section, we introduce a method aimed at analyzing the hidden patterns within the model's output. The hidden patterns are imperceptible to humans and challenging to describe through conventional mathematical terms. To decipher this hidden pattern, we turn to neural networks for assistance.

Drawing inspiration from model inversion attacks [13], [24], we can employ an inversion model capable of generating reconstructed training data of the target model based on the outputs. The process of data reconstruction can be likened to a type of upsampling. Throughout this process, the inversion model must extract a substantial amount of information from the confidence vectors to produce high-quality reconstructions. As a result, the essential information is thoughtfully rearranged from the perspective of a neural network (i.e., the inversion model) in the generation of reconstructed data, and certain imperceptible yet crucial patterns may subtly emerge within the reconstructions. Based on the reconstructed results by inverting the confidence vectors, we can employ various indicators to depict the model's unique characteristics.

In general, we utilize data reconstruction techniques to delve deep into the hidden pattern within the model's outputs, allowing us to detect the stolen models effectively. This innovative defense approach is referred to as an inversion-guided defense (IGD).

The overview of our defense method, IGD, is given in Fig. 3. Our IGD eliminates the need for crafting fingerprints and does not require specific samples to detect model-stealing behavior. It consists of two sub-procedures: training of the inversion model, and indicators-based determination. Firstly, the defender trains an inversion model for the victim model, which acts as a unique analyzer to uncover hidden patterns in model's outputs. Secondly, the defender randomly selects inputs to feed the suspicious model, and the inversion model computes multiple hidden pattern indicators based on the suspicious model's outputs to uncover the patterns hidden in the confidence vectors. To simultaneously leverage these hidden pattern indicators to determine whether the suspicious model is stolen or legitimate, the defender also trains a decision tree model to make the final determination.

1) Training of the Inversion Model: Existing methods of exploring the disparity between the outputs of stolen models and those of legitimate models often rely on special examples to maximize the distinction. When using normal examples for verification, these methods may prove ineffective. Yet, we argue that even in the case of normal examples, the outputs of the stolen model can still be distinguished from those of the legitimate model. Identifying the precise key difference between outputs is undoubtedly challenging.

Model inversion was initially proposed for interpretability of deep neural networks by reconstructing data [13]. It helps to understand the inner workings of a trained classifier by uncovering the crucial features it has learned.

Model inversion was originally proposed as a means of interpreting deep neural networks through data reconstruction in the field of explainable AI. Given a trained classifier, model inversion can uncover the crucial features learned by this classifier, enabling a deeper understanding of its inner workings. As the stolen model is essentially a replica of the victim model, they share similar feature extractors and prioritize similar features. During the data reconstruction process, stolen models will exhibit greater similarity to the victim model compared to legitimate models. Building upon this observation, we train an inversion model as an exclusive decoder for the victim model, enabling the reconstruction of input data and decoding of the embedded information in the model's outputs. The training procedure of the inversion model is introduced as follows.

As shown in the upper part of Fig. 3, the classifier F_v (i.e., the victim model) maps the inputs $x \in \mathcal{X}$ into outputs $Y = F_v(x) \in \mathcal{Y}$. The number of the classes is $k = |Y|$ and the predicted label is obtained as $y = \arg max_i Y_i$. The inversion model $\mathcal{I}$ is trained to minimize the reconstruction loss, such as the mean squared error (MSE) between the reconstructed data and inputs. An effective inversion model is expected to reconstruct data that closely resembles the original inputs, with an accurate representation of crucial features. The training objective can be formulated as follows:

$$\mathcal{L} = \mathbb{E}_{x \in X} \text{MSE}(x, \hat{x}) \quad \text{where} \quad \hat{x} = \mathcal{I}(F_v(x)). \tag{2}$$

A lower MSE between x and $\hat{x}$ indicates a higher quality of reconstructions. Furthermore, if the attacker has the access to the parameters of the victim model F_v, an advanced loss function incorporating a regularized term can be utilized to enhance the efficiency of the inversion model:

$$\mathcal{L} = \mathbb{E}_{x \in X} \left[\text{MSE}(x, \hat{x}) + \lambda \cdot \text{CE}(F_v(\hat{x}), y) \right], \tag{3}$$

where CE represents the cross entropy function. In addition, $\text{CE}(F_v(\hat{x}), y)$ refers to the ratio of reconstructed data that are correctly classified by machine learning models, commonly referred to as the attack accuracy of model inversion attacks. Greater attack accuracy equates to better quality of the reconstructed data. Intuitively, the model's outputs can be seen as a specialized encoding of the inputs, and the inversion model is trained as a decoder specifically for the victim model. Thus, it excels at reconstructing high-quality data based on the victim model's outputs but falls short when attempting the same with the legitimate model's outputs. This observation can be leveraged by the inversion model to differentiate between legitimate and stolen models, considering the stolen model's nature as a replicated version with similar functionality to the victim model.

2) New Hidden Pattern Indicators: Once the inversion model is well-trained, the second step in our defense is the evaluation phase where we compute some indicators that effectively reflect the hidden patterns within the confidence vectors. The defender randomly selects a set of normal data as test cases for evaluation. Subsequently, the defender queries the suspicious model to obtain its outputs. As illustrated in the lower part of Fig. 3, the defender's objective is to determine whether the suspicious model is a stolen version or not, based on its outputs.

To reveal the hidden patterns in the suspicious model's outputs, the defender employs the inversion model to decode the outputs, generating reconstructed versions of the original test cases. Using these reconstructed data, we propose three indicators that characterize the similarities between the suspicious model and the victim model.

a) Reconstruction Error (RE): The first hidden pattern indicator proposed in this paper is the RE, which quantifies the difference between the reconstructed data and the original input. A small reconstruction error indicates a higher quality of reconstruction, suggesting that the inversion model can accurately decode the embedded information in the suspicious model's outputs, and the suspicious model is similar to the victim model. In other words, a smaller reconstruction error implies a higher likelihood of the suspicious model being stolen. Formally, for a suspicious model F_{sus} and a set of test cases T, the hidden pattern indicator of RE can be computed as follows:

$$M_{RE} = \mathbb{E}_{x \in T} \text{MSE}(x, \mathcal{I}(F_{sus}(x)).$$

b) Attack Accuracy (AA): The second hidden pattern indicator is the AA, which represents the ratio of the reconstructed data that the victim model F_v classifies to the same class as the original data. If a reconstructed data point is classified to the target class, it indicates that the inversion model has successfully captured essential features about the classification tasks during the reconstruction process. In such cases, we can infer that the suspicious model's outputs accurately encode the input similarly to the victim model, further suggesting a higher probability of the suspicious model being a stolen version. Formally, for a suspicious model F_{sus} and a set of test cases T, the hidden pattern indicator of AA can be computed as follows:

$$M_{AA} = \frac{\sum_{x \in T} \mathbb{1}\{F_v(\mathcal{I}(F_{sus}(x))) = F_v(x)\}}{|T|}$$

where $|T|$ represents the number of test cases in the set T.

c) Attack Accuracy Cross-Entropy (AACE): The third hidden pattern indicator is the AACE. The test cases are divided into d subgroups, including $T_1, T_2, \cdots, T_n$, and fed into the victim model to compute an attack accuracy for each subgroup, yielding $AAs_1 = [acc_1, acc_2, \cdots, acc_d]$, where

$$acc_1 = \frac{\sum_{x \in T_1} \mathbb{1}\{F_v(\mathcal{I}(F_v(x))) = F_v(x)\}}{|T_1|}.$$

Simultaneously, these test case subgroups are also fed into the suspicious model to compute the classification accuracy, resulting in $AAs_2 = [acc'_1, acc'_2, \cdots, acc'_d]$, where

$$acc'_1 = \frac{\sum_{x \in T_1} \mathbb{1}\{F_v(\mathcal{I}(F_{sus}(x))) = F_v(x)\}}{|T_1|}.$$

If the suspicious model is similar to the victim model, the cross-entropy between AAs_1 and AAs_2 should be sufficiently low. Therefore, the third hidden pattern indicator is defined as:

$$M_{AACE} = \text{CrossEntropy}(AAs_1, AAs_2).$$

d) Summary: The first two hidden pattern indicators, RE and AA, assess the quality of reconstructed data using the inversion model $\mathcal{I}$. The inversion model is specifically trained for the victim model, which allows it to achieve good reconstructions on the victim model's outputs. Consequently, its effectiveness on the suspicious model's outputs suggests a similarity between the victim and suspicious model. Additionally, the third hidden pattern indicator, AACE, measures the consistency between the victim model and the suspicious model. If the attack accuracy patterns on different subsets of test cases are similar, it indicates a high level of consistency between the victim model and the suspicious model.

These three newly proposed hidden pattern indicators are integrated with the two existing testing indicators (i.e., JSD and RobD) in our method. It is worth noting that JSD and RobD were originally introduced in the context of adversarial examples. Instead, we compute these indicators using normal examples, which distinguishes our method from the original work that introduced JSD and RobD indicators.

3) Indicators Fusion: Our three new hidden pattern indicators are incorporated with the two previously used testing indicators, JSD and RobD, to verify the ownership of models. Notably, when multiple testing indicators are utilized in the verification procedure, it is crucial to adopt an appropriate decision-making strategy. Chen et al. [5] employed a simple voting strategy to reach the final decision, which may lead to inaccuracies due to the varying significance of these testing indicators. Therefore, we propose to use a decision tree model to automatically determine the decision, taking into account the relative importance of each testing indicator.

a) Training of the decision tree model: To train an effective decision tree model capable of accurately determining stolen models, we need to construct a training dataset with two classes of data points: positive data (i.e., stolen models) and negative data (i.e., legitimate models). Each instance in the dataset is associated with five features (i.e., five hidden pattern indicators including JSD, RobD, RE, AA, and AACE) and a corresponding label (i.e., 0 or 1). Specifically, we can obtain positive data by computing the hidden pattern indicators for a stolen model, and negative data by calculating the hidden pattern indicators for legitimate models. Based on this generated dataset, the defender can train a decision tree model, where a tree-like structure is constructed, and each internal node represents a hidden pattern indicator. This decision tree model can be used to make predictions for new data based on these five hidden pattern indicators.

b) Automatic determination: In general, with the trained inversion model and decision model, the defender can easily determine whether the suspicious model is stolen from the victim model or not. Specifically, the defender randomly selects some normal examples as test cases and feeds them into the suspicious model. Subsequently, two hidden pattern indicators, JSD and RobD, are computed based on the outputs of these normal test cases. Additionally, the inversion model is utilized to decode the outputs, allowing the computation of the other three hidden pattern indicators proposed in this paper, namely, RE, AA, and AACE. The values of these five hidden pattern indicators are then used as input to the decision tree model, enabling the defender to make the final determination.

C. Discussion

In summary, our method effectively addresses two significant challenges associated with prior defenses against model stealing attacks. Firstly, our IGD operates as a post-training defense to avoid any involvement in the victim model's training procedure. Consequently, the model's performance remains unaffected. Secondly, IGD exploits the information embedded within the confidence scores in the output. Instead of relying solely on the hard label to assess the similarity between the victim model and the suspicious model, IGD studies the similarity between the full confidence vectors of these models. As a result, we can use normal examples as test cases, eliminating the need to generate adversarial or misclassified examples. Lastly, IGD incorporates a decision tree model that autonomously assigns importance to different hidden patterns and calculates suitable thresholds for each indicator, providing a more precise method for threshold setting.

Furthermore, regarding the research question, we can answer it. Stolen models can exhibit divergent patterns compared to legitimate models. This hidden pattern in the confidence vectors can be captured through the inversion of their outputs and quantified using specific indicators (e.g., RE, AA, and AACE). The rationale behind this is that the inversion model trained for the victim model can achieve superior inversion quality when working with the outputs of stolen models. This is because stolen models share similar extracted features with the victim model. Consequently, stolen models can be identified clearly based on the hidden patterns in the confidence vectors, which can be captured and demonstrated by lower RE, higher AA, and lower AACE values.

D. Comparison With Autoencoders

Our method uses an inversion model to uncover hidden patterns. The combination of this inversion model and the victim model is similar to the structures of encoder-decoder, such as autoencoders [25]. Specifically, regarding their similarity, our approach can be viewed as a specialized instance of encoder-decoder structures. The victim model acts as the encoder, transforming the input into confidence vectors (i.e., bottleneck latent features), while the inversion model reconstructs the input from these confidence vectors. However, their differences can be demonstrated as follows: **(1)** In autoencoders, the encoder and decoder are trained concurrently, whereas in our approach, the victim model remains fixed, and only the inversion model undergoes training. **(2)** Our setting imposes an additional requirement of classification accuracy for the victim model compared to the encoders in autoencoders. **(3)** Given our assumption that the victim model is accessed in a black-box manner, the inversion model is trained without knowledge of the internal parameters of the victim model, differing from the training of the decoder in autoencoders.

Despite these disparities, we claim that encoder-decoder structures possess significant potential for boosting defenses against model stealing attacks, particularly implemented in the training stage of the victim model. For instance, if the defender can actively participate in the training of the victim model, the victim model can be regarded as the encoder, and a corresponding decoder can be designed as the inversion model, trained alternately with the victim model. Joint training facilitates the integration of various techniques, such as skip connections, thereby enhancing the performance of the inversion in unveiling hidden patterns within the confidence vectors. Furthermore, in this context, watermarking-based defenses can be seamlessly integrated into the method. The inversion model is specifically designed to detect watermarks in unauthorized copies of the victim model, contributing to the decrease of false positive rates in watermarking-based defenses. In conclusion, leveraging encoder-decoder structures for countering model stealing would be an intriguing direction for future research.

V. Experiments

We evaluated our proposed method, IGD, against the commonly used model stealing attacks (as described in Appendix) on different datasets and model architectures. In our context, the defender uses normal examples to distinguish the stolen models and the legitimate models.

A. Experimental Setup

1) Baselines: To explore the performance of existing defenses against the attacks aforementioned, we also selected two state-of-the-art fingerprinting defenses against model stealing attacks as the baselines in this paper. The first one is DeepJudge [5], a testing framework for the copyright protection of models. We focused on evaluating the efficiency of DeepJudge in the black-box setting. The second baseline method is a fingerprinting-based defense called SAC-w [10], which relies on misclassified examples and the mean correlation between the model's outputs to distinguish stolen models from legitimate models. In other words, SAC-w needs to generate misclassified examples before the verification procedure.

2) Evaluation Metrics: To assess the defensive effectiveness of different methods, we use four metrics as indicators of their capability to distinguish between stolen and legitimate models. The first evaluation metric is the Area Under the ROC Curve (AUC-ROC, abbreviated as AUC), which is a performance metric used to assess the quality of a binary classifier. AUC measures the ability of a model to distinguish between the positive and negative classes based on its predicted probabilities. For instance, if the AUC of a method reaches 1, it means that the method can perfectly distinguish between the two classes. Moreover, there are three widely used metrics related to classification performance, including accuracy, precision, and recall. A larger value for these metrics indicates a better ability to correctly classify the models.

3) Inversion Model Architectures: In our experiments, we adopt an inversion model with a simple architecture, comprising four transposed convolutional neural network (CNN) blocks. Each of the first three transposed CNN blocks includes a transposed convolutional layer, followed by a batch normalization layer, and a $Tanh$ activation function. The last block contains a transposed convolutional layer followed by a Sigmoid function.

B. Experimental Results

1) Defending Against Finetuning Attacks: We compared the defensive effectiveness of our methods with the two baseline methods on MNIST and CIFAR10. For DeepJudge, we evaluated two settings: one with only the JSD indicator (DeepJudge1) and another with two testing indicators, including JSD and RobD (DeepJudge2). Note that evaluating the AUC value of DeepJudge2 using a simple voting strategy for decision-making is challenging. Therefore, to evaluate the AUC of DeepJudge2, we also used a shallow decision tree with two features (JSD and RobD) to make the final decision. The quantified results are shown in Table I. In terms of the AUC, it can be observed that the AUCs of DeepJudge1 and DeepJudge2 are comparable, which aligns with our theoretical analysis in Section IV. The reason is that the distinguishing capability of RobD overlaps with that of JSD. However, for the victim model trained on CIFAR10 using ResNet, the AUC becomes lower when RobD is incorporated with JSD. This could be attributed to overfitting during the training of the decision model with both testing indicators (JSD and RobD), resulting in decreased defensive performance on the test dataset. This serves as evidence of the ineffectiveness of solely using JSD and RobD.

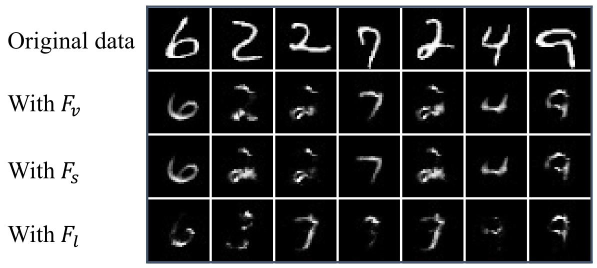

Fig. 4. The reconstructed data obtained from the outputs of various models. The term 'Original data' pertains to the input data that are used as input for different models. 'With F_v' signifies the reconstructed data derived from the victim model's outputs. On the other hand, 'With F_s' and 'With F_l' represent the reconstructed data sourced from the outputs of the stolen model and the legitimate model respectively.

During the decision-making process of our IGD, five hidden pattern indicators come into play. Among these, three new hidden pattern indicators introduced in this paper (namely, RE, AA, and AACE) are computed based on the reconstructed data. These reconstructions are produced from the model's outputs, and thus, they can be regarded as proxies for the extracted features by the corresponding models. The reconstructed data from different models are illustrated in Fig. 4. A noticeable observation is that the reconstructions stemming from the victim model's outputs exhibit the highest visual quality when compared to those from the stolen or legitimate models. This phenomenon arises from the fact that the inversion model is tailored to the victim model, enabling it to accurately reverse the extracted features of the victim model.

In addition, the reconstructed results based on the stolen model's outputs closely resemble those generated from the victim model's outputs. This similarity can be attributed to the shared extracted features between the stolen model and the victim model. However, recovering the original data from the legitimate model's outputs is more challenging. This difficulty arises due to the possibility of the legitimate model utilizing distinct features for classification when compared to the victim model. Consequently, when an inversion model trained against the victim model is applied, the reconstructions from the legitimate model's outputs exhibit inferior quality. This underlines that the underlying patterns in the outputs of the legitimate model differ from those in the victim model's outputs.

TABLE I
DEFENSIVE EFFECTIVENESS OF DIFFERENT METHODS AGAINST THE FINETUNING ATTACKS

Archi.	LeNet				VGG				ResNet			
Dataset	MNIST				MNIST				CIFAR10			
Finetune	**Accuracy**	**Precision**	**Recall**	**AUC**	**Accuracy**	**Precision**	**Recall**	**AUC**	**Accuracy**	**Precision**	**Recall**	**AUC**
DeepJudge1	0.667	0.639	0.958	0.656	0.576	0.667	0.133	0.559	0.881	0.952	0.833	0.927
DeepJudge2	0.667	0.639	**0.958**	0.722	0.576	0.667	0.133	0.559	0.833	0.84	0.875	0.919
Ours	**0.881**	**0.913**	0.875	**0.941**	**0.807**	**1.0**	**0.538**	**0.962**	**0.929**	**0.889**	**1.0**	**0.969**

Table I also provides a comprehensive view of our method's consistent superiority in terms of AUC across all settings when compared to DeepJudge. Our IGD incorporates another three hidden pattern indicators (i.e., RE, AA, and AACE) than the two indicators (i.e., RobD and JSD) used in DeepJudge. These three new hidden pattern indicators are designed for normal examples and can make full use of the rich information within the confidence scores, while RobD and JSD pay more attention to the hard labels. More valuable information extracted from the confidence scores can characterize the victim model more closely, and allow our IGD to achieve higher AUC. Moreover, we conducted evaluations based on three crucial classification metrics: accuracy, precision, and recall. In the indicator-based determination of classification tasks, these classification metrics depend on the selection of hidden pattern indicator thresholds. An excessively large threshold can yield high precision but low recall. Hence, determining an appropriate threshold is of great importance. However, DeepJudge relies on manual threshold determination [5]. Our experimental results clearly demonstrate that our approach outperforms the DeepJudge method. This improvement can be attributed to our method, which incorporates a broader range of hidden pattern indicators capable of identifying stealing behaviors, as well as its automated decision-making process based on computed hidden pattern indicators. The inclusion of more hidden pattern indicators grants IGD exceptional classification capabilities, resulting in higher values for the three classification metrics. Simultaneously, the automated decision-making process achieves a superior balance among these three metrics.

2) Defending Against Model Extraction Attacks: For model extraction attacks, watermarking-based defenses are typically not robust. Therefore, we evaluated the defensive effectiveness of our methods and the baseline method against model extraction attacks on MNIST and CIFAR10 datasets. The quantified results are presented in Table II.

When RobD is incorporated with JSD to determine the stealing behaviors against LeNet victim models, overfitting occurs, leading to a decrease in the AUC compared to that of DeepJudge1 (with only JSD). However, our method demonstrates better defense compared to both DeepJudge1 and DeepJudge2 in all situations. For both the VGG victim model trained on MNIST and the ResNet model trained on CIFAR10, our method effectively distinguishes between stolen models and legitimate models.

The experimental results illustrate that the detection of model extraction behaviors is generally easier compared to finetuning attacks. This is because model extraction attackers only have black-box access to the victim model, and therefore, the implementation of model extraction attacks is more difficult. During our experiments, we observed that most stolen models created through model extraction exhibited poor utility, with an accuracy of less than 70%. These models with high utility might be typically very similar to the victim model and are consequently easier for the defender to detect.

Our method performs better for complex models, such as VGG and ResNet, compared to simple models like LeNet. This observation can be attributed to the fact that complex models typically have a larger number of parameters, making the feature extractor of the victim model less susceptible to modifications. As a result, the stolen version of the feature extractor may share similar extracted features with the original feature extractor of the victim model. Since our hidden pattern indicators (i.e., Re, AA, and AACE) rely on the extracted features obtained during the model inversion process, this similarity in extracted features enables our method to effectively determine the similarity between the stolen models and the victim models. Consequently, our method achieves better AUC scores for complex models compared to simple models.

3) Exploiting Misclassified Examples: We also evaluated our methods in a setting where misclassified examples can be used for verification and compared our method with SAC on CIFAR-10. Similar to SAC-w [10], misclassified examples are those that are incorrectly classified by both the victim model and two additional substitute models trained using knowledge distillation techniques. The results against finetuning attacks are presented in Table III, and the results against model extraction attacks are illustrated in Table IV.

When defending against finetuning attacks, the defensive effectiveness of DeepJudge and our method improves when using misclassified examples. However, our method's effectiveness is still lower than that of SAC-w. It is important to note that our method is specifically designed to explore the information contained in the normal examples' outputs that can be used to distinguish stolen models from legitimate ones. In the presence of misclassified examples, the slight decrease in our defense's effectiveness when compared to other baseline methods is acceptable.

Furthermore, in the context of defending against model extraction attacks, we evaluated the effectiveness of our method and the baseline methods with misclassified examples. The results indicate that our method achieves the best defensive performance, even outperforming SAC-w. SAC-w relies on the mean correlation between the victim model's outputs

TABLE II
DEFENSIVE EFFECTIVENESS OF DIFFERENT METHODS AGAINST THE MODEL EXTRACTION ATTACKS

Archi.	LeNet				VGG				ResNet			
Dataset	MNIST				MNIST				CIFAR10			
Extraction	**Accuracy**	**Precision**	**Recall**	**AUC**	**Accuracy**	**Precision**	**Recall**	**AUC**	**Accuracy**	**Precision**	**Recall**	**AUC**
DeepJudge1	0.697	0.778	0.467	0.815	0.963	1.0	0.889	0.944	1.0	1.0	1.0	1.0
DeepJudge2	0.727	0.800	0.533	0.800	0.963	1.0	0.889	0.944	1.0	1.0	1.0	1.0
Ours	**0.970**	**0.9375**	**1.0**	**0.9722**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**

TABLE III
DEFENSIVE EFFECTIVENESS OF OUR METHOD AND SAC AGAINST THE FINETUNING ATTACKS WHEN THE MISCLASSIFIED EXAMPLES ARE USED

Finetune	**Accuracy**	**Precision**	**Recall**	**AUC**
DeepJudge1	0.833	0.869	0.833	0.882
DeepJudge2	0.810	0.750	1.0	0.943
Ours	0.905	0.858	**1.0**	0.944
SAC-w	**0.985**	**0.989**	0.989	**0.992**

TABLE IV
DEFENSIVE EFFECTIVENESS OF OUR METHOD AND SAC AGAINST THE MODEL EXTRACTION ATTACKS WHEN THE MISCLASSIFIED EXAMPLES ARE USED

Extraction	**Accuracy**	**Precision**	**Recall**	**AUC**
DeepJudge1	0.692	0	0	0.729
DeepJudge2	1.0	1.0	1.0	1.0
Ours	**1.0**	**1.0**	**1.0**	**1.0**
SAC-w	0.973	0.967	0.935	0.958

and the suspicious model's outputs, which measures the degree of similarity between the victim model's parameters and the suspicious model's parameters. However, our method relies on the extracted features of the victim model. As model extraction attacks can lead to significant changes in the model's parameters, SAC-w's effectiveness might decrease, whereas our method remains unaffected.

4) Adaptation to White-Box Attacks: Though our method assumes that the defender has only black-box access to the victim model, it can also be adapted in white-box scenarios. In this section, we consider three types of white-box settings, as illustrated in Table V. In the first white-box setting (known as white-box1), the defender plays the role of the judge and has access to the parameters of both the victim model and the suspicious model [5]. However, this scenario is uncommon in the real world, so we focus on the other two types of white-box settings. In the second white-box setting (denoted as white-box2), as the model trainer, the defender can manipulate the training procedure of the victim model. Therefore, the defender can jointly train the victim model and its corresponding inversion model, similar to encoder-decoder structures. This approach ensures that the features exploited by the inversion model in uncovering the hidden pattern better align with the features extracted by the victim model. In the third white-box setting (known as white-box3), the defender may be the model owner, who has the parameters of the victim model but no knowledge of the suspicious model. In this scenario, the defender can use the victim model to measure the quality of the reconstructions produced by the inversion model, guiding and enhancing the training of the inversion model. In general, the white-box defender can acquire more information to train a better inversion model compared to the defender in black-box scenarios. We compared the experimental results on CIFAR100 in black-box and white-box scenarios in Table VI.

TABLE V
THREE TYPES OF WHITE-BOX SETTINGS. 'TRAINING' INDICATES WHETHER THE DEFENDER CAN MANIPULATE THE TRAINING PROCEDURE OF THE VICTIM MODEL, WHILE 'PARAMETERS' INDICATES WHETHER THE DEFENDER HAS ACCESS TO THE PARAMETERS OF THE VICTIM MODEL OR THE SUSPICIOUS MODEL

Scenario	Role	Training	Parameters	
			Victim	Suspicious
White-box1	Judge	×	✓	✓
White-box2	Model trainer	✓	✓	×
White-box3	Model owner	×	✓	×
Black-box	A third party	×	×	×

TABLE VI
DEFENSIVE EFFECTIVENESS OF OUR METHOD ON CIFAR100. 'OURS-BB' INDICATES THE PERFORMANCE OF OUR METHOD IN THE BLACK-BOX SETTING, WHILE 'OURS-WB2' AND 'OURS-WB3' REPRESENT THE RESULTS IN THE WHITE-BOX2 AND WHITE-BOX3 SETTINGS, RESPECTIVELY

Finetune	**Accuracy**	**Precision**	**Recall**	**AUC**
DeepJudge1	0.738	0.741	0.833	0.837
DeepJudge2	0.881	0.880	0.917	0.870
Ours-BB	0.929	0.920	0.958	0.924
Ours-WB2	**0.976**	0.960	**1.0**	0.972
Ours-WB3	**0.976**	**0.999**	0.958	**0.979**

From the experimental results, our methods consistently outperform both DeepJudge1 and DeepJudge2 by a significant margin [5]. We observe the detection performance in the white-box2 and white-box3 settings is similar, both of which are better than that in the black-box setting. This improvement is attributed to the white-box defenders' ability to obtain a superior inversion model, which can accurately uncover hidden patterns. Furthermore, the recall in the white-box2 setting is optimal, with a value of 1.0, indicating the successful detection of all stolen models. This success can be attributed to the joint training of the victim model and inversion model, enabling the inversion model to precisely identify features learned by the victim model. Consequently, stolen models sharing similar feature extractors with the victim model can be accurately identified by the inversion model, resulting in a high recall.

5) *Robustness to Adaptive Attacks:* In this section, we examine the potential capabilities of adaptive attackers who possess knowledge of the exact defensive strategies and can implement countermeasures to enhance their attacks. Traditional defenses often depend on specialized examples, such as adversarial and misclassified instances. As a result, adaptive attackers can strengthen their attacks by retraining stolen models using these special samples. This retraining allows the models to evade certain defenses that rely on these specialized examples, as mentioned in [5]. In contrast, our defense approach, IGD, utilizes information derived from normal examples, effectively blocking adaptive attacks that rely on retraining with specialized samples.

However, even when specialized examples are unavailable, adaptive attackers familiar with our defensive strategy can adjust the stealing process to avoid detection by IGD. One straightforward modification involves altering the loss function. The adaptive attacker can introduce an additional loss component that evaluates the quality of reconstructed data, such as the MSE between the original input and the reconstructed input, into the existing loss function (i.e., the cross-entropy loss between the victim model's outputs and the stolen model's outputs). This additional loss value relates to the stealthiness of adaptive model stealing attacks. During the training of the stolen models, the introduction of the MSE between the original and reconstructed inputs would increase, causing the stolen model to appear more unrelated to the victim model from the perspective of our IGD. The newly incorporated loss function can be formulated as follows:

$$\mathcal{L} = \mathbb{E}_{x \in X} \mathrm{CE}(F_v(x), F_s(x)) - \eta \cdot \mathrm{MSE}(x, \mathcal{I}(F_v(x))). \quad (4)$$

We assess the efficacy of IGD against adaptive attacks that employ fine-tuning techniques. The parameter η serves as an indicator of the balance between the primary tasks and stealthiness. A higher η signifies that the attacker prioritizes stealthiness over the primary tasks. The experimental results on MNIST are shown in Table VII. Notably, the classification accuracy is comparable as the coefficient η varies. In other words, the advanced loss function, as modified by adaptive attackers, causes little impact on the main tasks of the stolen models. Furthermore, the impact of this advanced loss function on the effectiveness of our IGD is also not readily evident. The experimental results reveal that, with the exception of the setting where η equals 0.01, nearly all AUCs exhibit similar values. Surprisingly, IGD achieves its highest AUC when the attacker places the greatest emphasis on stealthiness (i.e., $\eta = 0.1$). Therefore, we argue that our IGD is also effective against adaptive attacks.

Moreover, it is worth noting that modifying the loss function in this way will lead to an increase in the time cost of stealing the models because this type of adaptive attack requires backpropagation operations on the inversion model, substantially augmenting the number of parameters involved in training the stolen models. We also validated this assumption on MNIST, and the quantitative outcomes were presented in Table VII. These findings confirm that the average training time for attacks with $\eta = 0$ is merely 79 seconds, significantly less than the time required for attacks with $\eta > 0$. This emphasizes the effectiveness of our IGD even in the face of adaptive attacks.

TABLE VII
DEFENSIVE EFFECTIVENESS OF OUR METHOD AGAINST ADAPTIVE ATTACKS BASED ON FINETUNING TECHNIQUES AND THE TIME COST OF ATTACKS IN VARIOUS SETTINGS. 'TIME' REPRESENTS THE TRAINING DURATION OF THE STOLEN MODELS, PRESENTED IN SECONDS

η	**0**	**0.001**	**0.01**	**0.1**
Accuracy	**0.9777**	0.9771	0.9767	0.9757
AUC	0.9323	0.9282	0.8704	**0.9375**
time (s)	**79**	179	179	180

TABLE VIII
DEFENSIVE EFFECTIVENESS OF OUR METHOD ON CIFAR100 WITH DIFFERENT INVERSION MODELS

Finetune	**Accuracy**	**Precision**	**Recall**	**AUC**
Inversion_4	0.929	0.920	0.958	0.924
Inversion_5	**0.976**	0.960	**1.0**	0.969
Inversion_6	0.929	**1.0**	0.875	0.972
Inversion_7	**0.976**	**1.0**	0.958	**0.979**

6) *Factors Evaluation:* In this section, we assess several factors that could influence the detection performance of our defense, IGD. Analyzing these factors can assist defenders in selecting appropriate parameters to optimize the defense and achieve the highest level of protection.

a) *Architecture of the inversion model:* Our IGD utilizes an inversion model to uncover hidden patterns. We evaluated the impact of the architecture of the inversion model on defensive performance. In the aforementioned experiments, the inversion model contains 4 transposed CNN blocks, denoted as Inversion_4. Here, we introduced three additional inversion models, Inversion_5, Inversion_6, and Inversion_7, which include five, six, and seven additional transposed CNN blocks, respectively. More transposed CNN blocks indicate more parameters within the inversion model, which may lead to longer training time.

The experimental results on CIFAR100 are presented in Table VIII. We utilize different inversion models to uncover hidden patterns and detect stealing behaviors for a victim classifier with the ResNet architecture. From the results, it can be observed that the defensive performance improves as the number of transposed CNN blocks increases. Inversion models with more than five transposed blocks exhibit significantly higher AUCs compared to Inversion_4. This is because the increased number of parameters in the inversion model potentially enables a more accurate representation of hidden patterns. Nevertheless, the defensive performance of these inversion models with more than five transposed blocks remains relatively comparable, indicating a gradual diminishing of the incremental improvement of defensive performance gained by increasing the number of parameters.

b) *Number of test cases:* Our IGD distinguishes between stolen models and legitimate models using hidden pattern indicators computed on test cases. Consequently, the number of

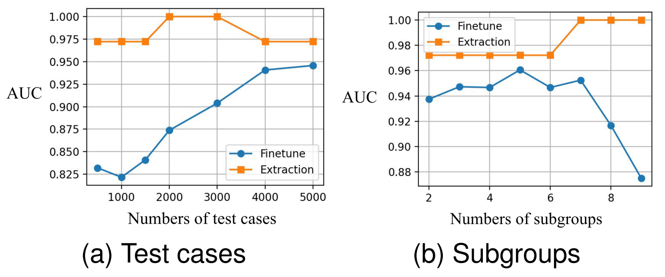

Fig. 5. Evaluating the factors that affect the performance of IGD.

TABLE IX

Time Cost of Our Defense on CIFAR10 and CIFAR100, Presented in Seconds. 'RobD&JSD' Represents the Computation Time of RobD and JSD Indicators, While 'Our Indicators' Indicates the Other Three Indicators Proposed by Our Method

Dataset	Classifier	Inversion	RobD&JSD	Our indicators
CIFAR10	552.1	745.3	7.02	4.01
CIFAR100	547.8	663.7	1.72	0.86

test cases can influence the performance of this discrimination. We conducted an evaluation to assess the relationship between the number of test cases and the detection performance of our IGD on a LeNet victim model trained on MNIST. The experimental results, presented in Fig. 5a, indicate that for model extraction attacks, even with a limited number of test cases (e.g., 500 samples), the defensive performance remains satisfactory. As the number of test cases increases, there is not a slightly varying trend in defensive performance. However, for finetuning attacks, there is a different situation. Specifically, the defensive performance of IGD against finetuning attacks is closely related to the number of test cases. When the defender has a small number of test cases (e.g., fewer than 2000 examples), the AUCs for successfully detecting stolen models remain below 0.875. As the number of available test cases increases, there is a noticeable upward trend in defensive performance, and it stabilizes when the number reaches 4000. Hence, when determining the number of test cases, defenders should carefully adjust this parameter to achieve optimal detection performance against finetuning attacks.

c) Number of subgroups in AACE: As one of the newly proposed hidden pattern indicators in our method, AACE contributes to the detection of model stealing behavior by assessing the consistency between the attack accuracy (AA) of the suspicious model and that of the victim model on specific test cases. In the computation of AACE, the test cases are divided into d subgroups. For a given set of test cases with a size of K, each subgroup contains K/d samples. Consequently, as d increases, more entries are involved in the computation of cross entropy between two groups of AAs, and the consistency property of AACE can more accurately reflect the similarity between the suspicious model and the victim model. However, a larger d also implies a smaller number of samples in each subgroup, which might introduce increased randomness and potentially reduce detection accuracy.

Fig. 5b illustrates the experimental results using a LeNet victim model trained on MNIST. For finetuning-based attacks, the AUC value consistently improves as the number of subgroups (denoted as d) increases. This is because AACE, with a larger d, more accurately reflects the consistency between the suspicious model and the victim model. However, after a certain point (e.g., $d = 5$), the AUC value starts to decline. This is due to the small number of samples in each subgroup, which becomes insufficient to represent the full set of test cases effectively.

In contrast, when it comes to model extraction attacks, the number of subgroups has a negligible impact on the AUC values. Detecting model extraction behaviors is generally easier compared to finetuning attacks because stolen models created using model extraction attacks, especially those with high utility, tend to be very similar to the victim model. Therefore, it is easier for the defender to detect model extraction attacks than finetuning attacks. Even without the AACE, the other four hidden pattern indicators remain effective in defending against model extraction attacks. This is why the AUC values are not significantly influenced by the number of subgroups.

Therefore, when determining the parameter for the number of subgroups, the defender should carefully choose an optimal d to achieve a balance between the number of subgroups and the representativeness of test cases within each subgroup.

7) Computational Cost: Our defense method introduces an inversion model, which does incur additional computational costs. These costs include training the inversion model and carrying out indicator computations. However, for a given victim model, our method only requires training the inversion model once, even when dealing with numerous suspicious models. This is because the trained inversion model can be applied to multiple different suspicious models, eliminating the need for training a separate inversion model for each one. Consequently, once the training of this inversion model is finished, the model's ownership verification in our method solely involves computing indicators that reflect hidden patterns. In this section, we conducted an evaluation of the time costs associated with our defenses on two datasets, namely CIFAR10 and CIFAR100.

The results are summarized in Table IX. The training process of a CIFAR10 classifier with the ResNet architecture took approximately 552.1 seconds. In comparison, training the inversion model (i.e., Inversion_4) took around 745.3 seconds, which is roughly 1.5 times longer than the training time of the classifier. However, it is important to note that the computation of indicators based on the trained inversion model is significantly time-saving. For example, using a total of 1638 test cases to compute the RobD and JSD indicators required only 4.01 seconds. Similarly, the computation of the other three indicators proposed in this paper (RE, AA, and AACE) could be completed in just 7.02 seconds.

Regarding CIFAR100, the training time of the inversion model is shorter compared to CIFAR10 due to the smaller size of the training dataset. Additionally, since we only used 198 test cases to compute indicators on CIFAR100, the verification procedure is faster compared to CIFAR10. In conclusion, our method trains the inversion model only once and the computation of indicators is efficient. This enables our method to effectively determine whether a suspicious model is stolen from the victim model or not.

VI. Conclusion

In this paper, we aim to explore the unique characteristics of stolen models, like the hidden patterns contained in the model's outputs. What sets our method apart is that we replace the commonly used adversarial examples in fingerprinting-based defenses with normal examples. We argue that even in cases involving normal examples within the test set, there are differences between the outputs of stolen models and those of unrelated models, which can help distinguish them.

To capture these differences, we apply an inversion model to decode a model's outputs and reconstruct the original input. The reconstructions are used to depict the hidden patterns in the output vectors. Our research expands the possibilities of test cases used to detect model stealing attacks. We also introduce a decision tree model to automatically assign suitable weights to different hidden pattern indicators. The decision tree model allows us to incorporate any new hidden pattern indicators proposed in the future into our method, which makes our method more flexible.

However, our proposed defense approach is specifically designed for classification models utilizing deep neural network architectures. It may not be directly applicable to other types of models, including language models [26], [27] and image-to-image models [28], [29]. For instance, in the case of image-to-image models, we can try to extract some special patterns that can reflect the characteristics of the generative model. However, this would require a redesign of the method used to recover these distinct patterns. Exploring this area will undoubtedly lead to exciting possibilities and will be a primary focus of our future research efforts.

References

[1] OpenAI. (2023). *ChatGPT [Software]*. Accessed: Mar. 7, 2024. Available: https://chat.openai.com

[2] Google. (2015). *Google Cloud API*. Accessed: Jul. 7, 2022. [Online]. Available: https://cloud.google.com/vision/

[3] D. Oliynyk, R. Mayer, and A. Rauber, "I know what you trained last summer: A survey on stealing machine learning models and defences," *ACM Comput. Surveys*, vol. 55, no. 14s, pp. 1–41, Dec. 2023.

[4] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff Nets: Stealing functionality of black-box models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 4954–4963.

[5] J. Chen et al., "Copy, right? A testing framework for copyright protection of deep learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 824–841.

[6] T. Orekondy, B. Schiele, and M. Fritz, "Prediction poisoning: Towards defenses against DNN model stealing attacks," 2019, *arXiv:1906.10908*.

[7] J. Zhang, S. Peng, Y. Gao, Z. Zhang, and Q. Hong, "APMSA: Adversarial perturbation against model stealing attacks," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1667–1679, Feb. 2023, doi: 10.1109/TIFS.2023.3246766.

[8] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, "Entangled watermarks as a defense against model extraction," in *Proc. USENIX Secur. Symp.*, 2021, pp. 1937–1954.

[9] X. Cao, J. Jia, and N. Z. Gong, "IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, May 2021, pp. 14–25.

[10] J. Guan, J. Liang, and R. He, "Are you stealing my model? Sample correlation for fingerprinting deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 36571–36584.

[11] Z. Peng, S. Li, G. Chen, C. Zhang, H. Zhu, and M. Xue, "Fingerprinting deep neural networks globally via universal adversarial perturbations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 13420–13429.

[12] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–10.

[13] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, "Neural network inversion in adversarial setting via background knowledge alignment," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.* New York, NY, USA: ACM, Nov. 2019, pp. 225–240, doi: 10.1145/3319535.3354261.

[14] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 36–52.

[15] Y. Zhu, Y. Cheng, H. Zhou, and Y. Lu, "Hermes Attack: Steal DNN models with lossless inference accuracy," in *Proc. USENIX Secur. Symp.*, Aug. 2021, pp. 1973–1988.

[16] R. N. Reith, T. Schneider, and O. Tkachenko, "Efficiently stealing your machine learning models," in *Proc. 18th ACM Workshop Privacy Electron. Soc.*, Nov. 2019, pp. 198–210.

[17] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 506–519.

[18] H. Zhang, G. Hua, X. Wang, H. Jiang, and W. Yang, "Categorical inference poisoning: Verifiable defense against black-box DNN model stealing without constraining surrogate data and query times," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1473–1486, Feb. 2023.

[19] L. Fan, K. W. Ng, and C. S. Chan, "Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–10.

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015, *arXiv:1412.6572*.

[21] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.

[22] Y. Zheng, S. Wang, and C.-H. Chang, "A DNN fingerprint for non-repudiable model ownership identification and piracy detection," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 2977–2989, Aug. 2022.

[23] S. Chen, M. Kahla, R. Jia, and G.-J. Qi, "Knowledge-enriched distributional model inversion attacks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 16158–16167.

[24] S. Zhou, T. Zhu, D. Ye, X. Yu, and W. Zhou, "Boosting model inversion attacks with adversarial examples," *IEEE Trans. Dependable Secur. Comput.*, pp. 1–18, 2023.

[25] P. Li, Y. Pei, and J. Li, "A comprehensive survey on design and application of autoencoder in deep learning," *Appl. Soft Comput.*, vol. 138, May 2023, Art. no. 110176.

[26] A. Naseh, K. Krishna, M. Iyyer, and A. Houmansadr, "Stealing the decoding algorithms of language models," 2023, *arXiv:2303.04729*.

[27] Z. Li et al., "On extracting specialized code abilities from large language models: A feasibility study," 2023, *arXiv:2303.03012*.

[28] T. Qiao et al., "A novel model watermarking for protecting generative adversarial network," *Comput. Secur.*, vol. 127, Apr. 2023, Art. no. 103102.

[29] H. Chen, T. Zhu, C. Liu, S. Yu, and W. Zhou, "High-frequency matters: Attack and defense for image-processing model watermarking," *IEEE Trans. Services Comput.*, pp. 1–15, 2024.

Shuai Zhou (Graduate Student Member, IEEE) received the B.Sc. degree from North China Electric Power University, China, in 2016, the M.Eng. degree from Chinese Academy of Sciences, China, in 2019, and the Ph.D. degree from the University of Technology Sydney, Australia, in 2024. He is currently an Assistant Professor with the Faculty of Data Science, City University of Macau. His current research interests include privacy preservation and adversarial attacks.

Tianqing Zhu (Member, IEEE) received the B.Eng. and M.Eng. degrees from Wuhan University, China, in 2000 and 2004, respectively, and the Ph.D. degree in computer science from Deakin University, Australia, in 2014. She is currently a Professor with the Faculty of Data Science, City University of Macau. Her research interests include privacy-preserving, data mining, and network security.

Dayong Ye received the M.Sc. and Ph.D. degrees from the University of Wollongong, Australia, in 2009 and 2013, respectively. He is currently a Research Fellow of cyber-security with the University of Technology Sydney, Australia. His research interests include differential privacy, privacy-preserving, and multi-agent systems.

Wanlei Zhou (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from The Australian National University, Canberra, Australia, in 1991, and the D.Sc. degree from Deakin University, Australia, in 2002. He is currently the Vice Rector (Academic Affairs) and the Dean of the Faculty of Data Science, City University of Macau, Macau, SAR, China. He has published more than 400 papers in refereed international journals and refereed international conference proceedings, including many articles in IEEE TRANSACTIONS and journals. His main research interests include security, privacy, and distributed computing.

Wei Zhao (Fellow, IEEE) is an internationally renowned scholar, who has served important leadership roles in academia, including the eighth Rector (i.e., President) of the University of Macau; the Dean of Science at the Rensselaer Polytechnic Institute; the Director for the Division of Computer and Network Systems, U.S. National Science Foundation; the Chair of Academic Council at Shenzhen University of Advanced Technology and CAS Shenzhen Institute of Advanced Technology; and the Senior Associate Vice President for Research at Texas A&M University. He led the effort to define the research agenda and to create the very first funding program for cyber-physical systems, when he served as the NSF CNS Division Director in 2006. In 2011, he was named by Chinese Ministry of Science and Technology as the Chief Scientist of the National 973 Internet of Things Project. As an IEEE Fellow, he has made significant contributions to cyber-physical systems, distributed computing, and cyberspace security. He was awarded the Lifelong Achievement Award by Chinese Association of Science and Technology. He was honored with the Overseas Achievement Award by Chinese Computer Federation.