Knowledge Representation of Training Data With Adversarial Examples Supporting Decision Boundary

Zehao Tian, Zixiong Wang, *Student Member, IEEE*, Ahmed M. Abdelmoniem, *Member, IEEE*, Gaoyang Liu, *Member, IEEE*, and Chen Wang, *Senior Member, IEEE*

***Abstract*— Deep learning (DL) has achieved tremendous success in recent years in many fields. The success of DL typically relies on a considerable amount of training data and the expensive model optimization process. Therefore, a trained DL model and its corresponding training data have become valuable assets whose intellectual property (IP) needs to be protected. Once a DL model or its training dataset is released, there is currently no mechanism for the entity that owns one part to establish a clear relationship with the other. In this paper, we aim to reveal the integrated relationship between a given DL model and the corresponding training dataset, by framing the problem of knowledge representation of a dataset with respect to DL models trained on it: *how to effectively represent the knowledge transferred from a training dataset to a DL model?* Our basic idea is that the knowledge transferred from a training dataset to a DL model can be uniquely represented by the model's decision boundary. Therefore, we design a novel generation method that utilizes geometric consistency to find the samples supporting the decision boundary, which can serve as the proxy for the knowledge representation. We evaluate our method in three different cases: IP audit of training data, IP audit of DL models, and adversarial knowledge distillation. The experimental results show that our method can improve the performance of existing works in all cases, which confirm that our method can effectively represent the knowledge transferred from a training dataset to a DL model.**

***Index Terms*— Knowledge representation, intellectual property, adversarial example, boundary supporting sample.**

I. Introduction

IN RECENT years, deep learning (DL) has achieved success in many fields, ranging from computer vision [1], [2] to natural language processing [3], [4]. Constructing a DL model typically involves two main aspects contributing to costs and efforts: data processing (including data collection, data annotation, and data transformation) and model training (including model structure selection, training hyper-parameter selection, and model iterative optimization). Therefore, the trained DL model and the corresponding training dataset become a crucial form of intellectual property (IP) of the owners that require protection, and many efforts have been made recently to verify the ownership of the data [5], [6] and the model [7], [8], [9].

However, existing IP verification studies [5], [7], [8], [9] normally isolate DL models and training datasets, and verify the IP of models or datasets separately, while ignoring the interaction between a DL model and its training data during the training process. Specifically, Dataset Inference [5] utilizes the common decision boundary of a given model and its copies to detect the usage of a target model's training dataset, while IPGuard [7] and Peng et al. [9] create a fingerprint based on data points near the classification boundary to profile a model's decision boundary. In addition to leveraging decision boundary information, Zhang et al. [8] propose to embed a special task-agnostic watermark into the target model's prediction behavior, which enables verification of a model's IP by extracting the hidden watermark with a surrogate model.

Nevertheless, the DL learning algorithm usually finds patterns in the training data that map the input data attributes to the target outputs and gets a DL model that captures these patterns (i.e., the knowledge that the training dataset conveys). Therefore, the training process of DL models can be regarded as the transferring process of knowledge from the training dataset to the corresponding model. Consequently, once a trained DL model or its training data is released, there is no mechanism for the entity that owns one part to establish a clear relationship with the other. This would be of immense utility when the model owner needs to claim the ownership of this model as well as its post-processed versions generated by model pruning [10], model distillation [11], or model fine-tuning [12], or when a data owner suspects that his data has been stolen [6] and expects to detect whether a suspect model embezzles the stolen data.

In this paper, we aim to reveal the integrated and hidden relationship between a given DL model and its training dataset. We focus on one fundamental problem known as knowledge representation of a dataset w.r.t. DL models trained on it: *how to effectively represent the knowledge transferred from a training dataset to a DL model through the training process?*

Manuscript received 23 March 2023; revised 16 June 2023; accepted 4 July 2023. Date of publication 7 July 2023; date of current version 17 July 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62272183, Grant 62171189, Grant 62002104, and Grant 62071192; in part by the Key Research and Development Program of Hubei Province under Grant 2021BAA026; and in part by the Special Fund for Wuhan Yellow Crane Talents (Excellent Young Scholar). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Edgar Weippl. *(Corresponding author: Gaoyang Liu.)*

Zehao Tian, Zixiong Wang, and Chen Wang are with the Hubei Key Laboratory of Smart Internet Technology, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: zhtian@hust.edu.cn; zixwang@hust.edu.cn; chenwang@hust.edu.cn).

Ahmed M. Abdelmoniem is with the School of Electronic Engineering and Computer Science, Queen Mary University of London, E1 4NS London, U.K. (e-mail: ahmed.sayed@qmul.ac.uk).

Gaoyang Liu is with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: gaoyangliu2020@gmail.com).

Digital Object Identifier 10.1109/TIFS.2023.3293418

To address this problem, we present a model-agnostic knowledge representation method, which is independent of the model structure, training algorithms, or data types. Our representation method not only avoids isolating the model from its training data but also effectively captures the intricate relationship forged between the model and its training data throughout the training process.

The basic idea is that the knowledge transferred from a training dataset to a DL model can be uniquely represented by the trained model's decision boundary, which essentially divides the data space into different class regions. Therefore, we first find the data samples that lie on the decision boundary of the given DL model (dubbed *boundary supporting samples*), and then directly treat these samples as the proxy of the desired knowledge representation. Although the idea sounds simple, we still confront two major challenges. The first challenge is how to effectively represent the decision boundary of a DL model. Since DL models usually learn the decision boundary with multiple layers and have a large number of parameters, adequately representing the decision boundary in a useful way is difficult. Inspired by recent works that leverage adversarial attacks [13], [14], [15] to perform relevant tasks, such as membership inference attacks [16], [17], [18] and adversarial robustness enhancement [19], [20], we propose a method to represent the decision boundary in the sample space using adversarial attacks. Specifically, we leverage adversarial attacks to find the boundary supporting samples to represent the model's decision boundary. To get the beneficial samples for knowledge representation, we adopt a geometric perspective and design an iterative generation method to search for the samples located on the given model's decision boundary. Our method uses the direction consistency between the normal vector of the decision boundary and the vector from the training sample to its corresponding boundary samples. Then we can regard the relationship between the training sample and its boundary sample generated by our method as the proxy for the corresponding knowledge representation.

Due to the high cost of finding boundary samples, the second challenge is that we can hardly construct the knowledge representation with all or the majority of training samples. Fortunately, various works have demonstrated that DL models are usually biased toward overly smooth decision boundary (i.e., low-frequency functionality) [21], [22], [23], [24]. Recent studies [25], [26] have demonstrated the success of treating the model's decision boundary as a linear function in the fields of adversarial examples and knowledge distillation. To increase the effectiveness of our method, we regard the decision boundary of DL models as a linear line, which dramatically simplifies the complexity of characterizing the decision boundary. In such a simplified case, we could use only one data sample and its projection onto the decision boundary to alleviate the construction overhead of our representation method. Consequently, for each class in the training data, we find the boundary samples of its centroid samples w.r.t. every decision boundary of the DL model. Then we combine the boundary samples of all classes to represent the knowledge learned by this model (c.f. Fig. 1).

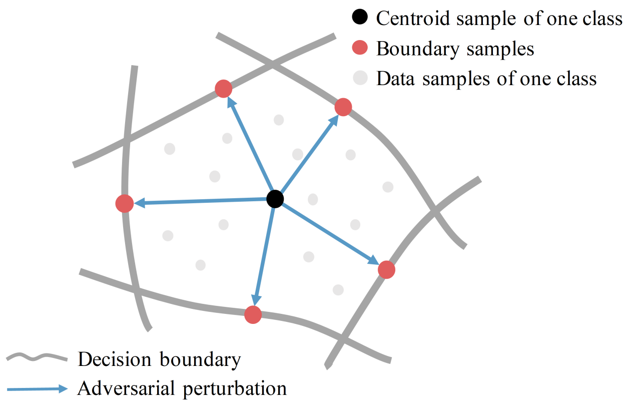


Fig. 1. Illustration of knowledge representation. Given a DL model and its training data, we demonstrate that the centroid and the boundary samples of all classes can represent the knowledge learned by the given model from the given dataset. For each class, our method first gets the centroid sample of the training data, and obtains the corresponding boundary samples which are located on the decision boundary of the given model. Then our method aggregates these samples together to form the complete knowledge representation.

To further validate the performance, we extend our knowledge representation method to three application scenarios: IP audit of training data, IP audit of DL models, and adversarial knowledge distillation. We compare our method with the state-of-the-art methods in each scenario [5], [7], [26]. We consider several post-processing techniques, including model pruning, retraining from scratch, and fine-tuning, which may breach the validity of our knowledge representation. The experiments demonstrate that our method effectively represents the knowledge of a training dataset w.r.t. a DL model, and can even retain the knowledge concerning the post-processed models.

We summarize our major contributions as follows:

- We formulate the knowledge representation problem for DL models in the field of IP protection and contribute a novel model-agnostic method for effectively representing the knowledge present in a dataset, which can be learned by DL models. We view the DL model, training data, and training process as interconnected components, and construct the knowledge representation by holistically considering all these components.
- To effectively represent the knowledge transferred from the training data to the DL model, we leverage the training samples along with their corresponding samples supporting the model's decision boundary. Accordingly, we introduce a novel method, specifically designed for IP verification purposes, to effectively identify these supporting samples from a geometric perspective.
- We evaluate our knowledge representation in three different cases: IP audit of training data, IP audit of DL models, and knowledge distillation. The experimental results validate that our representation method adeptly profiles the relationship between a DL model and its training data throughout the training process. Consequently, our approach can capture the knowledge transfer from the training data to the learned DL model. The code has been released for reproducibility purposes.[1]

[1]https://www.dropbox.com/s/upqzelbbtud8c6n/KR-Code.zip?dl=0

The remainder of this paper is organized as follows. Section II describes some preliminary knowledge on adversarial examples and boundary supporting samples. Section III formulates the problem of knowledge representation, followed by our design in Section IV. Sections V, VI and VII respectively evaluate the performance of our method on IP audit of training data and DL models, as well as adversarial knowledge distillation. Finally, Section VIII concludes this paper.

II. Preliminary

A. Adversarial Examples

Although DL models have achieved tremendous success in many applications, they have been demonstrated to be easily fooled by specially crafted inputs, a.k.a. adversarial examples [27]. An adversarial example is designed to look "normal" to humans but can cause misclassification to a given DL model. Generally, a single benign sample can traverse the decision boundary of a given model through adding a specially crafted adversarial noise, resulting in the creation of an adversarial example:

$$\min d(\mathbf{x}', \mathbf{x}) \quad \text{such that } \mathcal{M}(\mathbf{x}') \neq \mathcal{M}(\mathbf{x}) \tag{1}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are the benign sample and its corresponding adversarial example, respectively. The distance metric d measures the magnitude of adversarial noise added to $\mathbf{x}$, and $\mathcal{M}$ is the given DL model that $\mathbf{x}'$ aims to fool.

Many methods have been developed to construct adversarial examples in the past decade. Fast Gradient Sign Method (FGSM) [13] is the first adversarial attack, which crafts the adversarial noise in the same direction as the gradient of the training loss w.r.t. the benign input. Carlini and Wagner (C&W) attack [14] iteratively searches for the adversarial noise that turns the benign input into an adversarial example. Deepfool [25] generates the adversarial example directly along the direction of the orthogonal projection of the benign input to the decision boundary. HopSkipJump Attack [15] starts from a largely perturbed adversarial example, and then updates this example along the estimated gradient of a point at the model's decision boundary.

In this paper, we leverage the adversarial example generation methods to find the boundary supporting samples. However, none of the existing methods leveraging adversarial examples are particularly aimed at finding the decision boundary, but try to move a benign sample beyond a nearby decision boundary. To get samples that are beneficial to the knowledge representation, we specially design a novel adversarial example generation method, which can generate boundary supporting samples to characterize the geometric relationship between the training data and decision boundary of the given DL model.

B. Boundary Supporting Samples

The notion of boundary supporting samples [26] comes from the field of knowledge distillation, which aims to transfer the knowledge from a large DL model to a smaller one. In knowledge distillation, boundary supporting samples are defined as samples that lie near the decision boundary of DL models. These samples could contain the classification information of the decision boundary, and therefore provide a more accurate transfer of the inherent information. Correspondingly, Heo et al. [26] apply a modified adversarial attack to generate boundary supporting samples and demonstrate that they could represent the knowledge contained in a DL model.

Recently, a part of works also leverage the boundary decision samples to help a small model to mimic a larger model's prediction behavior. DeepDIG [28] proposes a bi-direction generation method for discovering the boundary supporting samples, and then takes advantage of these samples as a bridge between the decision boundary in the input space and the embedding space, which provides a multifaceted understanding of DNNs. Later, Qimera [29] improves the performance of data-free model compression by using synthetic boundary supporting samples generated from the superposed latent embeddings.

Moreover, boundary supporting samples have been successfully deployed in many fields, such as domain adaption [30], model quantization [29], and knowledge distillation [28]. However, the samples that support the decision boundary of a model have limited capacity to accurately represent the knowledge of a training dataset. Therefore, solely relying on these samples for IP protection of DL models and training datasets cannot achieve desired performance. In this paper, in order to efficiently and accurately audit IP of DL models and the corresponding training datasets, we leverage the boundary supporting samples of a given DL model to explicitly represent the knowledge learned by this model from its training dataset.

III. Problem Formation

The objective of this work is to propose a method for effectively representing the knowledge learned by a DL model from a training dataset. For practical purposes, we consider a third entity which has access to the prediction interface of the trained model and the corresponding training data. Therefore, the considered entity cannot control the training process or modify the model structure. Formally, the process of knowledge representation can be expressed as follows:

$$\text{Represent}(\mathcal{M}, \text{D}) \rightarrow \mathcal{KR} \tag{2}$$

where $\mathcal{M}$ is the given model's prediction interface, D is the training dataset, and $\mathcal{KR}$ is the extracted knowledge representation.

We aim to design a knowledge representation method that preserves the following properties.

A. Fidelity

The method is desired to well represent the knowledge contained in a dataset that can be learned by DL models. With this property, the knowledge representation has the ability to play the role as a medium for transferring knowledge.

B. Robustness

If a DL model is a post-processed version of the target model (i.e. knowledge distillation, model extraction, or model pruning), the representation of the post-processed model is expected to be very close to that of the target model.

C. Uniqueness

The knowledge representation should be unique to the target DL model and its training data. If a DL model is not the target model or its post-processed version, the knowledge representation for this model should be distinguishable.

D. Independence

The representation method should be independent of the model training algorithms and thus not require the modification of the training process and model structures of DL models.

IV. Knowledge Representation

A. Overview

In the design of the knowledge representation, we consider that the knowledge transferred from a training dataset to a DL model can be uniquely represented by the decision boundary of the training model. To represent the decision boundary, we design an adversarial example generation method to find the boundary samples of the training data located on the given model's decision boundary. Then the knowledge of the training data is represented by the relationship between training data and their corresponding boundary representation samples. Overall, our knowledge representation mainly includes the following two parts:

1) Decision Boundary Representation: We specifically design a boundary sample generation method to represent the given model's decision boundary. In this approach, we find an adversarial version of a benign sample $\mathbf{x}$ and project it onto the decision boundary of a given DL model $\mathcal{M}$, thus obtaining a boundary supporting sample denoted as $\bar{\mathbf{x}}$. To obtain $\bar{\mathbf{x}}$, our method leverages the normal vector on the decision boundary of $\mathcal{M}$ in the vicinity of $\mathbf{x}$ to get the perturbation required to misguide the prediction of $\mathcal{M}$. Then $\bar{\mathbf{x}}$ serves as a representation of the decision boundary for the given model.

2) Knowledge Representation Construction: With the boundary representation, we use the minimum perturbation from $\mathbf{x}$ to $\bar{\mathbf{x}}$ as the knowledge transferred between $\mathbf{x}$ and $\mathcal{M}$. Considering a whole training dataset and a multi-class DL model, for each class in the training data, we find the boundary samples of the class centroid samples w.r.t. every decision boundary of the DL model. Then we combine the boundary samples of all classes to represent the knowledge contained in the training dataset learned by this model.

B. Decision Boundary Representation

In this section, we introduce our boundary sample generation method in detail.

1) Boundary Representation Sample Generation: In order to represent the decision boundary of a DL model, we first design a novel adversarial attack to get the adversarial example $\tilde{\mathbf{x}}$ for the training sample $\mathbf{x}$ w.r.t $\mathcal{M}$. Then we project $\tilde{\mathbf{x}}$ to the decision boundary of the given model. However, a trained DL model and its training data are the product of massive cost and expertise efforts. Usually, they are kept secret in practice to protect the owner's benefit. Furthermore, the owners often make their DL models available to the public to obtain financial interest, typically in the form of black-box APIs. Therefore, to ensure that the method is agnostic to both the model and the data, we consider a targeted adversarial attack against a black-box model, which aims to generate $\tilde{\mathbf{x}}$ to change the predicted label of $\mathbf{x}$ to a certain label y_k:

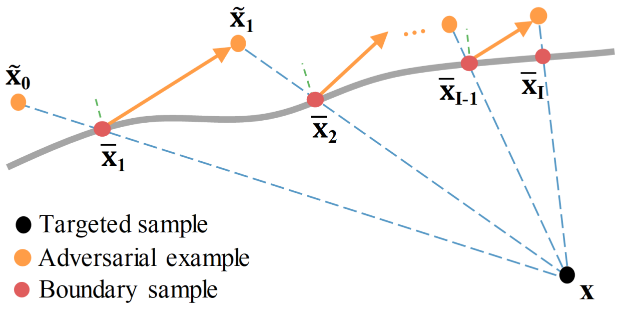


Fig. 2. Illustration of our boundary representation sample generation. (a) Initializing an adversarial point $\tilde{\mathbf{x}}_0$ at the first iteration. (b) Binary search to find the boundary point $\bar{\mathbf{x}}_i$ with $\tilde{\mathbf{x}}_{i-1}$ and $\mathbf{x}$. (c) Estimate normal vector at $\bar{\mathbf{x}}_i$ and compute $\tilde{\mathbf{x}}_{i-1} - \mathbf{x}$, and then get the loss of the objective function by Equation (4). (d) Optimize the objective function and update $\bar{\mathbf{x}}_i \rightarrow \tilde{\mathbf{x}}_i$. (e) The adversarial point $\tilde{\mathbf{x}}_i$ is used as the initial point for the next iteration.

$$\min_{v} \ \mathcal{D}(\mathbf{x}, \tilde{\mathbf{x}}) \ \text{ such that } \ \mathcal{M}(\tilde{\mathbf{x}}) = y_k \tag{3}$$

where $\mathcal{M}$ is the target model, y_k is the targeted label which could be any label but $\mathcal{M}(\mathbf{x})$, and $\mathcal{D}$ is a distance function for $\mathbf{x}$ and $\tilde{\mathbf{x}}$, such as cosine distance and $L2$ distance.

In order to generate $\bar{\mathbf{x}}$, we are motivated by the geometry property that the adversarial perturbation $v (v = \tilde{\mathbf{x}} - \mathbf{x})$ should align with the normal vector direction of the decision boundary of the model $\mathcal{M}$ at the boundary sample $\bar{\mathbf{x}}$. Therefore, we use cosine similarity to measure the direction deviation and carefully design the objective function as follows:

$$\begin{aligned}
\arg\min_{\tilde{\mathbf{x}}} \ \mathcal{L}(\tilde{\mathbf{x}}) &= -\frac{< \tilde{\mathbf{x}} - \mathbf{x}, \mathbf{N}|_{\bar{\mathbf{x}},\mathcal{M}} >}{\left\|\tilde{\mathbf{x}} - \mathbf{x}\right\|_2 \ \left\|\mathbf{N}|_{\bar{\mathbf{x}},\mathcal{M}}\right\|_2} \\
s.t. \ \mathcal{M}(\tilde{\mathbf{x}})|_{y_k} &= \mathcal{M}(\bar{\mathbf{x}})|_{y_{gt}} \\
\bar{\mathbf{x}} &= \lambda \mathbf{x} + (1 - \lambda)\tilde{\mathbf{x}} \\
\lambda &\in [0, 1]
\end{aligned} \tag{4}$$

where $<, >$ represents the inner product of vectors, $\|\cdot\|_2$ represents the length of the vector, $\mathbf{N}|_{\bar{\mathbf{x}},\mathcal{M}}$ is the normal vector of $\mathcal{M}$'s decision boundary at the boundary sample $\bar{\mathbf{x}}$, and $\tilde{\mathbf{x}} - \mathbf{x}$ is the adversarial perturbation. y_t and y_{gt} are the adversarial targeted label and the ground truth label of $\mathbf{x}$, respectively.

To optimize the above objective function, we combine the binary search and gradient descent mechanisms. For clarity, we name our boundary sample generation method as **Min**imizing **A**ngular **D**eviation (MinAD). The optimization process of MinAD is described in Algorithm 1. It is worth noting that the initial adversarial example $\tilde{\mathbf{x}}_0$ could be any random sample that satisfies the condition $\mathcal{M}(\tilde{\mathbf{x}}_0) = y_k$.

The process of MinAD is illustrated in Fig. 2, which mainly contains three steps:

2) Binary Search: Given the target sample $\mathbf{x}$ and its adversarial version $\tilde{\mathbf{x}}$ meeting the condition of $\mathcal{M}(\tilde{\mathbf{x}}) = y_t$,

Algorithm 1 MinAD Algorithm

Input: Target sample $\mathbf{x}$, initial adversarial example $\tilde{\mathbf{x}}_0$, target model $\mathcal{M}$.
Parameter: Max iteration number I, binary search threshold θ, sampling size B, sampling radii δ, initial learning rate η.
Output: Boundary sample $\bar{\mathbf{x}}$.

```
1: for $i = 1$ to $I$ do
2:     $\bar{\mathbf{x}}_i \leftarrow$ BinarySearch($\mathbf{x},\ \tilde{\mathbf{x}}_{i-1},\ \mathcal{M},\ \theta$)
3:     $\mathbf{N}|_{\bar{\mathbf{x}}_i,\mathcal{M}} \leftarrow$ NormalVectorEstimation($\bar{\mathbf{x}}_i, \mathcal{M}, B,\ \delta$)
4:     $\mathcal{L}(\tilde{\mathbf{x}}_{i-1}) \leftarrow (\tilde{\mathbf{x}}_{i-1} - \mathbf{x},\ \mathbf{N}|_{\mathbf{x}_i,\mathcal{M}})$
5:     $\tilde{\mathbf{x}}_i \leftarrow$ExampleUpdate($x_t,\ \mathcal{L}(\tilde{\mathbf{x}}_{i-1}),\ \eta$)
6: end for
7: $\bar{\mathbf{x}}_I \leftarrow$ BinarySearch($\mathbf{x},\ \tilde{\mathbf{x}}_I,\ \mathcal{M},\ \theta$)
8: return $\bar{\mathbf{x}}_I$
```

we find the boundary sample $\bar{\mathbf{x}}$ by using the binary search algorithm. As shown in Fig. 2, the boundary sample is the one located at the decision boundary of the target model along the line between $\mathbf{x}$ and $\tilde{\mathbf{x}}$.

3) Normal Vector Estimation: Given the sample $\bar{\mathbf{x}}$ located at the decision boundary of $\mathcal{M}$, we approximate the normal vector $\mathbf{N}|_{\bar{\mathbf{x}},\mathcal{M}}$ via the Monte Carlo estimation method:

$$\mathbf{N}|_{\bar{\mathbf{x}},\mathcal{M}} := \frac{1}{B}\sum_{b=1}^{B}\phi(\bar{\mathbf{x}} + \delta u_b)u_b, \tag{5}$$

$$\phi(\bar{\mathbf{x}} + \delta u_b) = \begin{cases} +1, & \mathcal{M}(\bar{\mathbf{x}} + \delta u_b) \neq y_k \\ -1, & \mathcal{M}(\bar{\mathbf{x}} + \delta u_b) = y_k. \end{cases} \tag{6}$$

where $\phi(\bar{\mathbf{x}} + \delta u_b)$ represents whether the sample $\bar{\mathbf{x}} + \delta u_b$ is adversarial or not, $\{u_b\}_{b=1}^{B}$ are i.i.d. drawn from the uniform distribution over the d-dimensional sphere, δ is a small positive constant and is proportional to d^{-1} [15], and B is the number of disturbed groups.

4) Example Update: We implement our data updates based on the objective function in Equation (4). We obtain the gradient $\nabla\mathcal{L}(\tilde{\mathbf{x}}_i)$ directly and use the gradient descent algorithm for the adversarial example updates as follows:

$$\tilde{\mathbf{x}}_i = \bar{\mathbf{x}}_{i-1} - \eta\nabla\mathcal{L}(\tilde{\mathbf{x}}_{i-1}) \tag{7}$$

where η represents the learning rate of the updates, and we achieve high optimization efficiency by adjusting the learning rate dynamically. In particular, we reduce the learning rate by half after each update, and set an appropriate threshold to control the step size per update. After that, we use binary search again to bring it back to the decision boundary.

Finally, we treat the generated sample $\bar{\mathbf{x}}_I$as the representation of the decision boundary of the target model, w.r.t. the given target sample $\mathbf{x}$.

C. Knowledge Representation Construction

After generating the boundary sample $\bar{\mathbf{x}}_I$, our subsequent objective is to effectively represent the knowledge that has been transferred between the data $\mathbf{x}$ and the model $\mathcal{M}$. We first compute the perturbation $\mathbf{r} = \bar{\mathbf{x}}_I - \mathbf{x}$. Then we regard the perturbation $\mathbf{r}$ as the knowledge transferred between the training sample $\mathbf{x}$ and the DL model $\mathcal{M}$.

Now we extend the data-model relationship measurement method to the case of the whole dataset and multi-class model. For each class in the training dataset, we select the centroid sample and find its boundary samples at the decision boundary of every class of the DL model. Thus, the knowledge learned by this model can be represented by all boundary samples. Therefore, we need to select the centroid sample first and then construct the knowledge representation.

1) Centroid Sample Selection: Based on the existing work [31], we conclude that the selection of a centroid sample greatly affects the final performance, and so we aim to choose a simple but effective selection method. Consider a training dataset $S = S_0 \cup S_1 \cup \ldots \cup S_{K-1}$, where K is the number of classes. We choose the centroid sample of each class, defined as the sample with the lowest cumulative distance to other points within the same class. Specifically, we find the centroid sample $\mathbf{s}_k$ for class k $(0 \leq k < K)$ as follows:

$$\min_{\mathbf{s}_k^t \in S_k} \sum_{j=0}^{N-1} \left\| \mathbf{s}_k^t - \mathbf{s}_k^j \right\|_2 \tag{8}$$

where N represents the number of data in class k $(0 \leq t, j < N)$, $\|\cdot\|_2$ represents the length of the vector. $\mathbf{s}_k^t$ is taken as the centroid sample $\mathbf{s}_k$ of class k. We get the centroid samples of all K classes.

2) Knowledge Representation Matrix: We next construct the knowledge representation matrix to further represent the knowledge of training data. For class k, we get the perturbations of $\mathbf{s}_k$ to the model $\mathcal{M}$'s boundaries, and construct the knowledge representation matrix as $KRM_k = (\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{C-1})$, where C represents the number of target labels for boundary sample generation, $\mathbf{r}_c$ represents the perturbation of $\mathbf{s}_k$ to the boundary with class c. We combine the knowledge representation matrix of each class and get the knowledge representation matrix KRM of the whole training dataset, which is denoted as:

$$\begin{aligned} KRM &= (KRM_0, KRM_1, \ldots, KRM_{K-1})^T \\ &= \begin{pmatrix} \mathbf{r}_0^0 & \mathbf{r}_0^1 & \mathbf{r}_0^2 & \cdots & \mathbf{r}_0^{C-1} \\ \mathbf{r}_1^0 & \mathbf{r}_1^1 & \mathbf{r}_1^2 & \cdots & \mathbf{r}_1^{C-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{r}_{K-1}^0 & \mathbf{r}_{K-1}^1 & \mathbf{r}_{K-1}^2 & \cdots & \mathbf{r}_{K-1}^{C-1} \end{pmatrix} \end{aligned} \tag{9}$$

Finally, the knowledge representation matrix will represent the knowledge representation of training data with centroid samples. In general, C is equal to K. Nevertheless, due to the complexity of the operation for some large datasets such as CIFAR-100 and TinyImageNet, we appropriately reduce C to improve the effectiveness of our knowledge representation. It is worth noting that the knowledge can also be expressed in terms of the norm of $\mathbf{r}$, i.e., $\|\mathbf{r}\|_2$ in specific application scenarios, such as IP audit of the training dataset. We would evaluate the performance of our knowledge representation in the form of $\|\mathbf{r}\|_2$ in the following sections.

V. Case 1: IP Audit of Training Data

A. Threat Model

The IP audit of training data aims to verify whether a DL model embezzles a training dataset whose IP permission the model trainer does not have. The state-of-the-art work Dataset inference (DI) [5] makes use of the basic idea that knowledge contained in the stolen model's training set is common to all stolen copies. Therefore, if our knowledge representation could achieve a resemble or even improved performance compared with DI, we could verify that our representation method can effectively represent the knowledge of a training dataset.

1) Attack Model: We consider a victim $\mathcal{V}$ with its private dataset $S_{\mathcal{V}}$. The goal of an adversary $\mathcal{A}$ is to extract the knowledge or by-products of $S_{\mathcal{V}}$, and use it to train its model $f_{\mathcal{A}}$. The adversary $\mathcal{A}$ can breach IP of the victim's training data in many ways. For instance, $\mathcal{A}$ is able to gain complete access to the victim's private dataset $S_{\mathcal{V}}$ or the queries of the victim's model $f_{\mathcal{V}}$, and trains its own model $f_{\mathcal{A}}$ with the information of the victim's dataset and model. Besides, $\mathcal{A}$ can steal the model $f_{\mathcal{V}}$ by model extraction [32], which is the most common way of violating the IP of the training data.

2) Target Model: The victim $\mathcal{V}$ trains a model $f_{\mathcal{V}}$ on its private dataset $S_{\mathcal{V}}$. The victim $\mathcal{V}$ aims to judge whether the suspected model $f_{\mathcal{A}}$ is derived from its private dataset $S_{\mathcal{V}}$.

B. Comparison Methods

1) Dataset Inference (DI): [5]. For DI, the victim $\mathcal{V}$ extracts the distances between a part of samples and decision boundary of the target model, and labels these distances with in or out labels, which means whether the corresponding sample belongs to $\mathcal{V}$'s private dataset $S_{\mathcal{V}}$ or not. These labeled distances are used to train a regression model $g_{\mathcal{V}}$ to predict whether a sample contains the private information from $S_{\mathcal{V}}$. Finally, we sample an equal number of samples from S and $S_{\mathcal{V}}$ to query suspected model $f_{\mathcal{A}}$, and calculate the confidence score vector with $g_{\mathcal{V}}$, which would be used to perform the IP audit of DI. DI proposed two different methods for distance measuring: the white-box adversarial attack named MinGD and Blind Walk method for the black-box setting.

2) DI + MinAD: We replace MinGD and Blind Walk with our MinAD in DI. The distance between the generated adversarial sample and the target sample is taken as the distance from the target sample to the decision boundary. The remaining settings of this method remain consistent with those used in DI.

3) DI + KRM: We replace the randomly selected samples in DI with the class centroid samples of training data, thereby making use the KRM contained adversarial examples to perform the IP audit. The remaining steps are the same for DI. We generate KRM by MinGD and Black Walk, respectively.

4) DI + MinAD + KRM: We simultaneously replace the distance measurement method and the adversarial examples corresponding to selected samples with our MinAD and KRM. The remaining steps are the same for DI.

C. Experimental Setup

1) Datasets and Models: We use CIFAR-10 [33] and CIFAR-100 [33] datasets to evaluate the performance for comparisons.

CIFAR-10 Dataset. CIFAR-10 dataset contains 50,000 training color images and 10,000 testing color images from 10 classes. Each class has 5,000 training images and 1,000 testing images. The size of each image is 32×32.

CIFAR-100 Dataset. CIFAR-100 dataset has the same format as CIFAR-10, but it has 100 classes with 600 images each. In particular, each class has 500 training images and 100 testing images.

For both datasets, we useachieve WideResNet [34] with a depth of 28 and a widening factor of 10 (WRN-28-10) as the victim model, and this model is trained with a dropout rate of 0.3. For the stealing attack model, we use WRN-16-1 on CIFAR-10 and WRN-16-10 on CIFAR-100, respectively.

2) Stealing Attacks: The adversary $\mathcal{A}$ would like to steal the private knowledge $S_{\mathcal{V}}$ of victim $\mathcal{V}$ and train its own model $f_{\mathcal{A}}$. We consider seven model stealing attacks and involve different levels of access to the private knowledge of the victim as follows.

(**1**) $\mathcal{A}$ is able to gain complete access to the dataset $S_{\mathcal{V}}$. (**a**) Model distillation [35]. $\mathcal{A}$ wants to enhance the performance of smaller model $f_{\mathcal{A}}$ by the logits of larger model $f_{\mathcal{V}}$. (**b**) Pruning. $\mathcal{A}$ attempts to train an alternative architecture on $S_{\mathcal{V}}$ and improve the robustness of $f_{\mathcal{A}}$.

(**2**) $\mathcal{A}$ has direct access to the $f_{\mathcal{V}}$. (**a**) Fine-tuning. $\mathcal{A}$ uses the confidence vectors of the $f_{\mathcal{V}}$ on unlabeled public data to optimize its decision boundary. (**b**) Zero-shot learning [36]. $\mathcal{A}$ performs the data-free knowledge transformation process to train its model.

(**3**) $\mathcal{A}$ can steal the $f_{\mathcal{V}}$ by model extraction [32]. (**a**) Model extraction using labels. $\mathcal{A}$ trains $f_{\mathcal{A}}$ on a pseudo-dataset provided by $\mathcal{V}$. (**b**) $\mathcal{A}$ trains its $f_{\mathcal{A}}$ to minimize the KL divergence with the outputs of $f_{\mathcal{V}}$ on a public (or non-task specific) dataset.

3) Parameter Settings: For the model training, we fine-tune the student model for 10 epochs and train it for 20 epochs on the whole training set. Moreover, we use a subset of 500,000 unlabeled TinyImages closest to CIFAR-10 [37] for model extraction and fine-tuning. Additionally, we use cross-entropy loss and SGD optimizer to train the models, with the learning rate decaying by a factor of 0.2 at the end of the $0.3\times$, $0.6\times$, and $0.8\times$ the total number of epochs.

For MinAD, we set 20 iterations and 1,000 queries per iteration. We use SGD optimization with a learning rate decaying by half from 16 to 0.2 in each round. In the case of CIFAR-100, we only consider the 10 most confident target classes for the centroid sample of each class.

For DI, we take 200 samples from $S_{\mathcal{V}}$ and 200 from S to train $g_{\mathcal{V}}$, and 10 samples from $S_{\mathcal{V}}$ for hypothesis testing.

4) Metrics: We use the same metrics as DI [5]. We calculate the confidence score vectors c and $c_{\mathcal{V}}$ from the public and private datasets with $g_{\mathcal{V}}$. Same as DI, we form a two-sample T-test on the distribution of c and $c_{\mathcal{V}}$, and then derive the p-value from the one-sided hypothesis $H_0 : \mu < \mu_{\mathcal{V}}$. If a vector contains samples from $S_{\mathcal{V}}$, it would get a lower confidence score and decrease the p-value of the hypothesis test. To capture the average confidence of the hypothesis test in claiming that a model was stolen, we calculate the effect size $\Delta\mu = \mu - \mu_{\mathcal{V}}$ and *p-value*, where $\mu = \bar{c}$ and $\mu_{\mathcal{V}} = \bar{c}_{V}$ are

TABLE I
THE PERFORMANCE OF IP AUDIT OF TRAINING DATA ON CIFAR-10

Attacks	DI (MinGD)		DI (Blind Walk)		DI +MinAD		DI (MinGD) +KRM		DI (Blind Walk) +KRM		DI +MinAD+KRM	
	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value
Source	1.57	10^{-4}	1.99	10^{-25}	1.90	10^{-35}	1.98	10^{-5}	1.99	10^{-19}	1.99	10^{-18}
Distillation	0.92	10^{-2}	1.30	10^{-3}	1.27	10^{-4}	1.90	10^{-6}	1.89	10^{-5}	1.99	10^{-22}
Pruning	0.90	10^{-2}	1.02	10^{-2}	1.05	10^{-3}	1.76	10^{-4}	1.76	10^{-4}	1.97	10^{-7}
Zero-shot	0.63	10^{-2}	0.84	10^{-2}	0.90	10^{-7}	1.86	10^{-5}	1.83	10^{-5}	1.99	10^{-17}
Fine-tuning	1.03	10^{-3}	1.38	10^{-3}	1.19	10^{-4}	1.88	10^{-5}	1.95	10^{-7}	1.93	10^{-6}
Label-query	0.87	10^{-2}	0.75	10^{-2}	1.00	10^{-2}	1.94	10^{-6}	1.95	10^{-7}	1.97	10^{-6}
Logit-query	0.97	10^{-2}	1.04	10^{-3}	1.02	10^{-3}	1.91	10^{-6}	1.79	10^{-6}	1.92	10^{-6}

TABLE II
THE PERFORMANCE OF IP AUDIT OF TRAINING DATA ON CIFAR-100

Attacks	DI (MinGD)		DI (Blind Walk)		DI +MinAD		DI (MinGD) +KRM		DI (Blind Walk) +KRM		DI +MinAD+KRM	
	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value
Source	1.98	10^{-7}	1.99	10^{-23}	1.93	10^{-25}	1.78	10^{-4}	1.99	10^{-28}	2.00	10^{-20}
Distillation	0.92	10^{-3}	1.06	10^{-3}	1.15	10^{-4}	1.87	10^{-5}	1.97	10^{-6}	1.96	10^{-6}
Pruning	1.08	10^{-3}	1.05	10^{-2}	0.93	10^{-2}	1.80	10^{-4}	1.96	10^{-6}	1.99	10^{-7}
Zero-shot	1.00	10^{-3}	0.80	10^{-2}	0.90	10^{-2}	1.86	10^{-5}	1.99	10^{-10}	1.99	10^{-10}
Fine-tuning	0.41	10^{-1}	0.97	10^{-3}	0.47	10^{-1}	1.85	10^{-4}	1.92	10^{-7}	1.91	10^{-6}
Label-query	1.18	10^{-4}	0.81	10^{-2}	0.84	10^{-2}	1.89	10^{-5}	1.82	10^{-6}	1.91	10^{-6}
Logit-query	1.20	10^{-3}	1.03	10^{-3}	1.53	10^{-5}	1.85	10^{-5}	1.87	10^{-7}	1.91	10^{-6}

mean confidence score vectors of public and private datasets respectively. If the p-value is less than a predefined threshold, $H0$ is rejected and the model is marked as stolen. For more details about the calculation of p-value and $\Delta\mu$, please refer to DI [5].

D. Experimental Results

1) Performance of IP Audit of Training Data: Table I and Table II show the results of IP audit of the data. We consider seven model stealing attacks in our experiments. Besides, we introduce an attack as a comparison baseline, dubbed as Source Attack, which directly steals the victim model. The larger $\Delta\mu$ and smaller p-value mean the better performance of IP audit of training data. The results show that, compared with MinGD and Blind Walk proposed by the original DI, DI with MinAD achieves better performance regardless of the use of KRM. For instance, DI with MinAD achieves a $\Delta\mu$ of 1.00 when the attacker can only access the target model's prediction interface for CIFAR-10. In such a case, MinGD and Blind Walk can only achieve a $\Delta\mu$ less than 0.90. Moreover, combining DI with both MinAD and KRM could achieve an improved $\Delta\mu$ of 1.97, which outperforms the original DI. Besides, when we combine MinGD or Blind Walk with KRM, the performance of DI also improves by more than 0.5 for $\Delta\mu$, while the corresponding p-values are all less than 10^{-4}.

Our proposed MinAD provides a more accurate measurement of the relationship between the training data and the corresponding DL model, thereby enhancing the performance of DI. Additionally, our KRM enhances DI performance with fewer samples. Overall, our representation encompasses the most crucial and representative knowledge of the data, effectively capturing the relationship between the given model and its training data.

From the experimental results shown in Tables I and II, we can find an abnormal phenomenon confronting the Source Attack. DI with MinAD has the smallest p-value, while DI (Blind Walk) and DI + MinAD + KRM have similar performance, which seemingly indicates that our MinAD and KRM cannot improve the performance of DI. The main reason for this phenomenon is that in the case of the model stolen by the Source attack, the training dataset of the stolen model is an exact replica of the victim model's training dataset. Consequently, the center samples of the KRM will be identical to those of the victim model. As MinAD exhibits greater accuracy in establishing the relationship between samples and decision boundaries, the fingerprint generated by DI+MinAD+KRM becomes challenging to differentiate between the victim model and the model stolen by the Source attack, thus leading to a similar performance as DI (Blind Walk). On the other hand, due to the precision of MinAD, DI could construct more precise fingerprints with multiple random training samples. Consequently, DI with MinAD exhibits a substantial improvement in performance, resulting in the smallest p-value when facing the Source attack.

2) Impact of Number of KRMs: The number of KRMs is directly related to the number of adversarial examples generated, thus impacting the final performance. To get more than one KRM, we divide each class equally and get centroid samples of each part as the sub-centroid sample. The KRMs

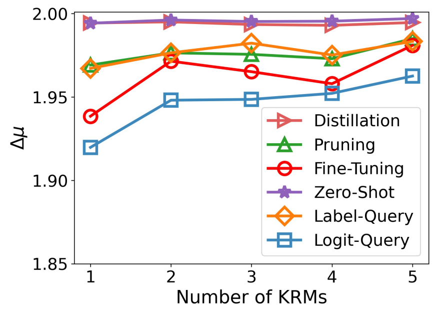


Fig. 3. The performance of IP audit of training data on CIFAR-10 with different number of KRMs.

corresponding to sub-centroid samples will be used in the experiments with those of the centroid sample.

The proportion of important knowledge contained in the KRM will decrease if the number of KRMs becomes excessive. Therefore, we employ a limited number of KRMs, ranging from 1 to 5, in order to explore their impact. We combine DI with MinAD and KRM for CIFAR-10 dataset. The results are shown in Fig. 3. For DI, we get a larger effect size $\Delta\mu$ with an increasing number of KRMs. Notably, $\Delta\mu$ can approach 2.0 for most of the suspected models when only five KRMs are involved. We speculate that training the regression model $g_{\mathcal{V}}$ with more samples improves its prediction performance.

VI. Case 2: IP Audit of DL Models

A. Threat Model

The objective of IP audit of DL models is to verify whether a given DL model is a stolen or the post-processed version of the target model. The state-of-the-art work IPGuard [7] makes use of the key observation that a DNN model can be uniquely represented by its decision boundary. The decision boundary is learned from the training data and thus essentially represents knowledge that the model learns from its training data through the training process. Therefore, if our knowledge representation could achieve a similar or even improved performance compared with IPGuard, we could demonstrate that our method could achieve the goal of representing knowledge.

1) Attack Model: Consider a victim $\mathcal{V}$ with its model $f_{\mathcal{V}}$. The goal of the adversary $\mathcal{A}$ is to steal the prediction function of model $f_{\mathcal{V}}$ to obtain its own model $f_{\mathcal{A}}$. $\mathcal{A}$ can get access to the model $f_{\mathcal{V}}$'s training data, training settings, and model parameters directly. With these information, $\mathcal{A}$ could derive a stolen model that has a similar function as the target model.

2) Target Model: The victim $\mathcal{V}$ trains its own model $f_{\mathcal{V}}$ on its training dataset. $\mathcal{V}$ would like to determine whether the suspected model $f_{\mathcal{A}}$ has stolen the prediction function of $f_{\mathcal{V}}$.

B. Comparison Methods

1) IPGuard: [7]. The basic idea of IPGuard is that a model can be uniquely fingerprinted by its decision boundary. Therefore, IPGuard leverages the samples near the decision boundary as the fingerprints to perform the IP audit. If the suspected model predicts the same labels for most data points near the decision boundary as those of the target model, the suspected model is then regarded as the stolen version of the victim model $f_{\mathcal{V}}$. IPGuard generates a set of samples S near the decision boundary of $f_{\mathcal{V}}$ through adversarial attacks. The victim $\mathcal{V}$ queries the prediction interface of the suspected model $f_{\mathcal{A}}$ and gets the prediction labels of these samples. If the predicted labels of $f_{\mathcal{A}}$ are similar to $\mathcal{V}$'s own predictions, $f_{\mathcal{A}}$ is considered as the counterfeit version of $f_{\mathcal{V}}$. IPGuard designs a white-box adversarial attack to generate fingerprints.

2) IPGuard + MinAD: We directly replace the adversarial example generation method in [7] with our MinAD, and the sample set S is composed by adversarial examples generated with different initialization. The rest of the settings are the same as IPGuard.

3) IPGuard + KRM: We replace the composition of the fingerprint set S with our KRM. In particular, for class k, the sample set S_k is the composition of the centroid sample $\mathbf{s}_k$ and KRM_k. The rest is the same as IPGuard. We use the adversarial attack of IPGuard to generate each adversarial example in KRM.

4) IPGuard + MinAD + KRM: We replace the adversarial example generation method and the fingerprint samples with our MinAD and KRM simultaneously. In particular, for class k, the sample set S_k is the combination of the centroid sample $\mathbf{s}_k$ and KRM_k. The rest is the same as IPGuard.

C. Experimental Setup

1) Datasets and Models: We use CIFAR-10 and CIFAR-100 in this section. The details of these datasets can be found in Section V. ResNet20 with an accuracy of 0.91 is used as the victim model for CIFAR-10. For CIFAR-100 dataset, we use WRN-22-4 with an accuracy of 0.76. All DL models are trained with 25 epochs.

2) Stealing Attacks: The adversary $\mathcal{A}$ aims to steal the complete model $f_{\mathcal{V}}$ of victim $\mathcal{V}$. As for the suspected model $f_{\mathcal{A}}$, we consider three post-processing, including fine-tuning, retraining, and pruning. These post-processing models are called *positive suspected models*. Besides, we set two types of models which are different from $f_{\mathcal{V}}$, where attackers train their models without stealing the model $f_{\mathcal{V}}$, and we call them the *negative suspected models*.

(**1**) $f_{\mathcal{A}}$ is the positive suspected model. (**a**) Fine-tuning. $\mathcal{A}$ fine-tunes any layers of $f_{\mathcal{V}}$ using the training data. Here we consider the case where only the last layer (i.e., fully connected layer) is fine-tuned, and others are fixed. (**b**) Retraining. $\mathcal{A}$ initializes the weight of any layers of $f_{\mathcal{V}}$ and trains the model using the training data. Here we consider the case where only the model weightsW of the last layer is fine-tuned, and the rest weights are fixed. (**c**) Pruning. $\mathcal{A}$ can achieve model compression and network lightweight by weight pruning [38]. Here we consider $\mathcal{A}$ prunes 10% of model parameters that have the smallest absolute value of $f_{\mathcal{V}}$ and retrains the model.

(**2**) $f_{\mathcal{A}}$ is the negative suspected model. (**a**) Same-architecture models (SA). We define the models that have the same architecture but different parameter initialization, optimizer, or loss functions with the victim model as the negative models. (**b**) Different-architecture model (DA). We adopt

TABLE III
THE COMPARISON RESULTS OF IP AUDIT OF DL MODELS ON CIFAR-10 AND CIFAR-100

Suspected Models		CIFAR-10				CIFAR-100			
		IPGuard	IPGuard +MinAD	IPGuard +KRM	IPGuard +MinAD +KRM	IPGuard	IPGuard +MinAD	IPGuard +KRM	IPGuard +MinAD +KRM
Positive Suspected Model	Fine-tune	0.68	0.92	0.83	0.93	0.69	0.72	0.81	0.89
	Retrain	0.72	0.85	0.86	0.93	0.74	0.79	0.79	0.88
	Prune	0.61	0.61	0.69	0.97	0.71	0.67	0.80	0.92
Negative Suspected Model	SA	0.44	0.06	0.37	0.13	0.64	0.07	0.43	0.08
	DA-LENET	0.18	0.11	0.20	0.05	0.30	0.09	0.17	0.08
	DA-VGG	0.38	0.09	0.27	0.03	0.34	0.10	0.28	0.07

LeNet-5 (DA-LENET) and VGG16 (DA-VGG) architecture for CIFAR-10 and CIFAR-100 to construct the negative suspected model.

3) Parameter Settings: For model training, we load the pre-trained model using the torchvision package. We train VGG16 for 5 epochs and other suspected models for 25 epochs on the entire training set to ensure model convergence. We use cross-entropy loss as our loss function and the SGD optimizer. The learning rate is decayed by a factor of 0.2 at the end of $0.3\times$, $0.6\times$, and $0.8\times$ the total number of epochs.

For MinAD, we set 20 iterations and 1,000 queries per iteration. We use SGD optimization with a learning rate decaying from 16 to 0.2, reducing by 0.5 each round. In the case of CIFAR-100, we generate adversarial examples only for the centroid samples corresponding to the decision boundaries of the 10 most confident target classes.

4) Metrics: We use the same metrics as IPGuard [7]. We use the *matching rate* as our metric, which represents the proportion of samples for which the prediction labels of the suspected model match those of the victim's model. If the matching rate exceeds a predefined threshold, we conclude that the suspected model is a post-processing of the original model.

D. Experimental Results

1) Performance of IP Audit of DL Models: In this case, we consider six types of suspected models constructed by different post-processing. For the original IPGuard, we randomly sample 100 (resp. 1000) samples from CIFAR-10 dataset (resp. CIFAR-100) and then generate the corresponding adversarial examples as the model IP fingerprints. We use only one KRM as the fingerprints for CIFAR-10. Particularly for CIFAR-100, to ensure the fairness of the experiments, we only generate 10 adversarial examples for each class centroid sample. Therefore, the total number of samples contained in the KRM equals 1000. We use the matching rate to measure the performance of the model IP audit, which is denoted as the fraction of fingerprint samples whose labels are predicted by the suspect model and align with those predicted by the target model. A positive suspected model (i.e. the post-processed version of the target model) should have a large matching rate, whereas the negative model is anticipated to have a low rate.

Table III shows the performance of IP audit for DL models. Results show that our MinAD can improve the performance of IPGuard, indicating that the adversarial samples generated by MinAD can provide a more accurate representation of the decision boundary than those generated by IPGuard. We also find that IPGuard cannot identify suspected models with the same architecture as the target models. When combining IPGuard with MinAD, we can achieve improved matching rates of 0.06 for CIFAR-10 and 0.07 for CIFAR-100 respectively. Additionally, the performance of IPGuard can be further improved by 10% on average when combined with KRM. When we combine IPGuard with both MinAD and KRM, the improved IPGuard achieves the best performance. The matching rate of positive and negative suspected models can reach 1.0 and 0.0, respectively, for both CIFAR-10 and CIFAR-100 datasets. These experimental results demonstrate that our knowledge representation can effectively represent what a DL model learned from the training data.

Specifically, when considering the CIFAR-100 dataset, IPGuard+MinAD exhibits inferior performance compared to IPGuard for several types of suspected models. This can be primarily attributed to the heightened effectiveness of our MinAD approach in establishing a more precise relationship between a training sample and the decision boundary of the DL model. MinAD outperforms IPGuard in this regard by generating fingerprints that are more sensitive to changes in the decision boundary, ultimately resulting in a decreased matching rate of the suspected models. On the other hand, when utilizing KRM, we strategically obtain boundary-supporting samples exclusively from the nearest boundaries of the training samples. This approach enhances the model's resilience and ability to tolerate changes in the decision boundary. The experimental results clearly demonstrate that IPGuard + KRM outperforms IPGuard, supporting the aforementioned point.

2) Impact of Number of KRMs: To evaluate the impact of the number of KRMs on the performance of our knowledge representation, we change the number of KRMs from 1 to 5. We combine DI with MinAD and KRM for CIFAR-10 dataset. Fig. 4 shows the experimental results, which indicate that the number of KRMs does not affect the matching rate for IPGuard. This is because the generated adversarial examples are only used for testing. In addition, another reason is that the total number of examples is limited.

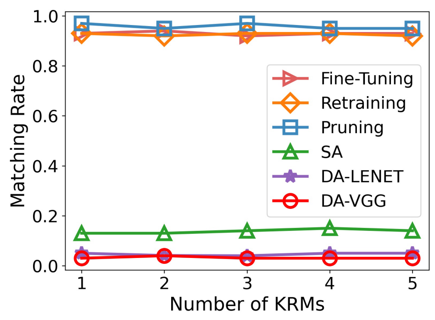


Fig. 4. The performance of IP audit of DL models on CIFAR-10 with different number of KRMs.

VII. Case 3: Adversarial Knowledge Distillation

Adversarial knowledge distillation aims to transfer the knowledge contained in a large model (i.e., the teacher model) to a small model (i.e., the student model) using samples close to the decision boundary of the teacher model. It is well known that the performance of a model highly depends on how well the model learns the true decision boundary from the training data distribution. Therefore, knowledge distillation with the teacher model's decision boundary information could improve the distillation performance of the student model. Based on this idea, Adversarial knowledge distillation (AdvKD) [26] utilizes an adversarial attack to find samples near the decision boundary of the teacher model and trains a student model on these samples. AdvKD achieves state-of-the-art performance but with fewer training samples. This indicates that the samples close to the decision boundary of a model convey the knowledge of this model. Therefore, if our knowledge representation could achieve a resemble or even improved performance compared with AdvKD, we could verify that our representation is effective in conveying the knowledge of a model and its corresponding training data.

A. Comparison Methods

1) Adversarial Knowledge Distillation (AdvKD): [26]. Considering a teacher model $f_{\mathcal{T}}$ and a student model $f_{\mathcal{S}}$, AdvKD generates a set of samples near the boundary of $f_{\mathcal{T}}$ and computes the boundary supporting loss with these samples, which is introduced to transfer the information about the decision boundary from the teacher model to the student model. This loss is then used in the training of $f_{\mathcal{S}}$ along with the classification loss and knowledge distillation loss [35]. AdvKD [26] leverages Deepfool [25] to generate the adversarial samples that are close to the decision boundary of $f_{\mathcal{T}}$.

2) AdvKD + MinAD: We replace Deepfool with our MinAD in AdvKD and get the samples near the boundary using MinAD.

3) AdvKD + KRM: We consider all adversarial examples corresponding to KRM as the samples near the boundary for AdvKD. Then we use the adversarial examples from our KRM to train $f_{\mathcal{S}}$. Specifically, for class k, the sample set is a combination of the centroid sample $\mathbf{s}k$ and $KRMk$. The remaining steps remain unchanged. We use the adversarial attack described in [26] to generate KRM.

TABLE IV

The Top-1 Test Accuracy (%) of Student Models for Knowledge Distillation on CIFAR-10

Student Models	AdvKD	AdvKD +MinAD	AdvKD +KRM	AdvKD +MinAD +KRM
ResNet8	81.88	84.49	80.71	80.85
ResNet14	86.16	87.18	85.04	85.33
ResNet20	87.20	87.24	86.38	86.13

4) AdvKD + MinAD + KRM: We replace the adversarial example generation method and the training data of $f_{\mathcal{S}}$ in AdvKD with MinAD and KRM simultaneously. The process of knowledge distillation remains the same as that of AdvKD.

B. Experimental Setup

1) Datasets and Models: We use CIFAR-10 and Tiny-ImageNet to evaluate the performance of our knowledge representation in the case of knowledge distillation.

CIFAR-10 Dataset. The dataset information of CIFAR-10 can be found in Section V.

TinyImageNet Dataset. TinyImageNet dataset contains 100,000 training color images and 10,000 testing color images from 200 classes. Each class has 500 training images and 50 testing images. The size of each image is 64×64.

For CIFAR-10 dataset, we use ResNet-26 with an top-1 testing accuracy of 0.92 as the teacher model, and use ResNet-8, ResNet-14, and ResNet-20 as the student models successively. For TinyImageNet dataset, a pre-trained ResNet-50 with a top-1 testing accuracy of 0.57 and a top-5 accuracy of 0.81 is selected as the teacher model. The teacher models are pre-trained for more than 20 epochs on this dataset. A ResNet18 is used as the student model.

2) Parameter Settings: For the generation of adversarial samples supporting the decision boundary of $f_{\mathcal{T}}$, we randomly select the sample as the initial sample whose ground truth label has the highest probability among both the teacher and student models. We set the decision boundary of $f_{\mathcal{T}}$ as the adversarial label closest to the initial sample's predicted label. We train the student model for at least 20 iterations and choose 64 initial samples per batch for each iteration.

For MinAD, we set the number of iterations to 20 and the number of queries per iteration to 1,000. We use the SGD optimizer with a learning rate decaying by half from 16 to 0.2 in each round for CIFAR-10 and from 1280 to 16 for TinyImageNet. In the case of TinyImageNet, we only generate the adversarial examples concerning the decision boundaries of the 10 most confident target classes.

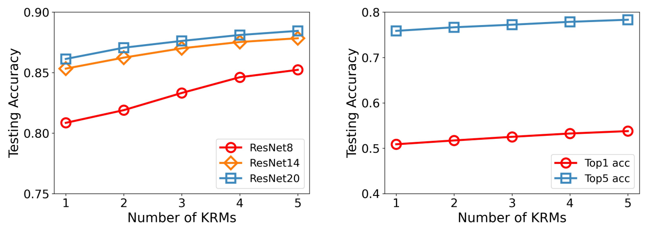
3) Metrics: Following [26], we use the *test accuracy* of the student model to measure the performance of the comparisons.

C. Experimental Results

1) Performance of Knowledge Distillation: Table IV and Table V show the results of knowledge distillation. We evaluate the performance of the student models and record the corresponding top-1 accuracy for CIFAR-10 and the top-1

TABLE V
THE TEST ACCURACY (%) OF STUDENT MODELS FOR KNOWLEDGE DISTILLATION ON TINYIMAGENET

Test Accuracy	AdvKD	AdvKD +MinAD	AdvKD +KRM	AdvKD +MinAD +KRM
Top-1	44.00	53.34	49.10	50.88
Top-5	71.25	77.25	74.07	75.87



(a) CIFAR-10 Dataset (b) TinyImageNet Dataset

Fig. 5. The performance of knowledge distillation on CIFAR-10 and TinyImageNet with different number of KRMs.

and top-5 test accuracies for TinyImageNet. The experimental results show that our MinAD can improve the performance of AdvKD, regardless of whether KRM is used or not. For instance, the test accuracy of the student model of ResNet8 can achieve 84.49% when AdvKD is combined with MinAD for CIFAR-10 dataset. This is significantly better than the original AdvKD which can only obtain a student model with a test accuracy of 81.88%. For TinyImageNet dataset, the student model can get a top-1 test accuracy 53.34% and a top-5 test accuracy of 77.25% when we combine AdvKD with MinAD. In contrast, the original AdvKD only gets the top-1 and top-5 accuracies of 44.00% and 71.25%, respectively.

Based on the experimental results, it is evident that KRM, despite containing a limited number of samples near the teacher's decision boundary, can achieve comparable or even superior performance compared to the original AdvKD method. However, it falls short of the performance achieved by AdvKD when integrated with MinAD. Among all the methods evaluated, AdvKD integrated with MinAD emerges as the top-performing approach. The main reason lies in the difference in how data is utilized for knowledge distillation. AdvKD and AdvKD with MinAD leverage a substantial amount of data during the distillation process. In contrast, AdvKD with KRM relies on a single KRM to distill knowledge from the teacher model to the student model. It is noticed that the dataset sizes associated with this single KRM are significantly smaller, consisting of only 100 samples for CIFAR-10 and 2,000 samples for TinyImagenet, while AdvKD employs a larger dataset of 10,000 samples. Consequently, the exclusive use of a single KRM with AdvKD leads to a decrease in the test accuracy of the distilled models compared to AdvKD with MinAD.

2) Impact of Number of KRMs: Like other cases, we increase the number of KRMs from 1 to 5. We combine AdvKD with both MinAD and KRM for CIFAR-10 dataset and TinyImageNet dataset. Fig. 5 shows that the test performance of the student models increases with the number of KRMs growing. Specifically, it is observed that when utilizing 5 KRMs (i.e. 500 samples for CIFAR-10 and 10,000 samples for TinyImageNet) as training data, the performance of AdvKD surpasses that of the case with only one KRM. For instance, in the case of the CIFAR-10 dataset and a ResNet20 student model, the accuracy of the student model improves to 88.20% when employing 5 KRMs in conjunction with both AdvKR and MinAD. In contrast, the accuracy with only one KRM is recorded as 87.24%.

VIII. CONCLUSION

In this paper, we propose a knowledge representation method that captures the knowledge contained in a dataset learned by DL models. We introduce a novel generation method to identify samples located on the decision boundary of a given DL model. Subsequently, we extract the boundary supporting samples from the class centroid samples of the training data and utilize them as proxies for knowledge representation. Our knowledge representation framework can be seamlessly applied to IP audit of training data, IP audit of DL models, and knowledge distillation. Experimental results from three application scenarios demonstrate that our knowledge representation improves the performance of state-of-the-art approaches in these domains. We consider our work to be a significant step towards characterizing the knowledge embedded in a dataset learned by DL models and shedding light on the intrinsic properties of the DL training process.

In future research, we intend to advance our method by incorporating more sophisticated representations of the model's decision boundary, surpassing the current approximation of a straight line. This expansion will enable us to explore the nuanced knowledge embedded within the decision boundary structure. With the fine-grained representation, we could develop efficient representations capable of handling intricate decision boundary forms, thereby leading to performance gain across diverse application scenarios.

REFERENCES

[1] M. Hassaballah and A. I. Awad, *Deep Learning in Computer Vision: Principles and Applications*. Boca Raton, FL, USA: CRC Press, 2020.
[2] S. Chen et al., "TransZero: Attribute-guided transformer for zero-shot learning," in *Proc. AAAI*, 2022, pp. 330–338.
[3] Y. Chaudhary, H. Schütze, and P. Gupta, "Explainable and discourse topic-aware neural language understanding," in *Proc. ICML*, 2020, pp. 1479–1488.
[4] H. B. Zia, I. Castro, A. Zubiaga, and G. Tyson, "Improving zero-shot cross-lingual hate speech detection with pseudo-label fine-tuning of transformer language models," in *Proc. AAAI*, 2022, pp. 1435–1439.
[5] P. Maini, M. Yaghini, and N. Papernot, "Dataset inference: Ownership resolution in machine learning," in *Proc. ICLR*, 2021, pp. 1–22.
[6] G. Liu, T. Xu, X. Ma, and C. Wang, "Your model trains on my data? Protecting intellectual property of training data via membership fingerprint authentication," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 1024–1037, 2022.
[7] 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.

[8] J. Zhang et al., "Deep model intellectual property protection via deep watermarking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4005–4020, Aug. 2022.

[9] 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.

[10] Y. Wang et al., "Pruning from scratch," in *Proc. AAAI*, 2020, pp. 12273–12280.

[11] D. Y. Park et al., "Learning student-friendly teacher networks for knowledge distillation," in *Proc. NeurIPS*, vol. 34, 2021, pp. 13292–13303.

[12] Y. Wang, D. Ramanan, and M. Hebert, "Growing a brain: Fine-tuning by increasing model capacity," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3029–3038.

[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

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

[15] J. Chen, M. I. Jordan, and M. J. Wainwright, "HopSkipJumpAttack: A query-efficient decision-based attack," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 1277–1294.

[16] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot, "Label-only membership inference attacks," in *Proc. ICML*, 2021, pp. 1964–1974.

[17] Z. Li and Y. Zhang, "Membership leakage in label-only exposures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 880–895.

[18] S. Rezaei and X. Liu, "On the difficulty of membership inference attacks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7888–7896.

[19] H. Xu, X. Liu, Y. Li, A. Jain, and J. Tang, "To be robust or to be fair: Towards fairness in adversarial training," in *Proc. ICML*, 2021, pp. 11492–11501.

[20] J. Cui, S. Liu, L. Wang, and J. Jia, "Learnable boundary guided adversarial training," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 15701–15710.

[21] N. Rahaman et al., "On the spectral bias of neural networks," in *Proc. ICML*, 2019, pp. 5301–5310.

[22] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on tabular data," 2022, *arXiv:2207.08815*.

[23] B. Neyshabur, R. Tomioka, and N. Srebro, "In search of the real inductive bias: On the role of implicit regularization in deep learning," in *Proc. ICLR Workshop*, 2015, pp. 1–9.

[24] G. Valle-Perez, C. Q. Camargo, and A. A. Louis, "Deep learning generalizes because the parameter-function map is biased towards simple functions," in *Proc. ICLR*, 2018, pp. 1–35.

[25] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2574–2582.

[26] B. Heo, M. Lee, S. Yun, and J. Y. Choi, "Knowledge distillation with adversarial samples supporting decision boundary," in *Proc. AAAI*, 2019, pp. 3771–3778.

[27] A. Serban, E. Poll, and J. Visser, "Adversarial examples on object recognition: A comprehensive survey," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–38, May 2021.

[28] H. Karimi, T. Derr, and J. Tang, "Characterizing the decision boundary of deep neural networks," 2019, *arXiv:1912.11460*.

[29] K. Choi, D. Hong, N. Park, Y. Kim, and J. Lee, "Qimera: Data-free quantization with synthetic boundary supporting samples," in *Proc. NeurIPS*, vol. 34, 2021, pp. 14835–14847.

[30] K. Saito, D. Kim, S. Sclaroff, and K. Saenko, "Universal domain adaptation through self supervision," in *Proc. NIPS*, vol. 33, 2020, pp. 16282–16292.

[31] B. Sorscher, R. Geirhos, S. Shekhar, S. Ganguli, and A. S. Morcos, "Beyond neural scaling laws: Beating power law scaling via data pruning," 2022, *arXiv:2206.14486*.

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

[33] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, 2009.

[34] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.

[35] G. Hinton et al., "Distilling the knowledge in a neural network," in *Proc. NeurIPS Deep Learn. Represent. Learn. Workshop*, 2015, pp. 1–9.

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

[37] Y. Carmon, A. Raghunathan, L. Schmidt, J. C. Duchi, and P. S. Liang, "Unlabeled data improves adversarial robustness," in *Proc. NeurIPS*, 2019, pp. 1–12.

[38] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. NeurIPS*, 2015, pp. 1–9.