DeepCore: Simple Fingerprint Construction for Differentiating Homologous and Piracy Models

Haifeng Sun, Lan Zhang, and Xiang-Yang Li (✉)

University of Science and Technology of China
Hefei, China
sun1998@mail.ustc.edu.cn
zhanglan@ustc.edu.cn
xiangyangli@ustc.edu.cn

Abstract. As intellectual property rights, the copyright protection of deep models is becoming increasingly important. Existing work has made many attempts at model watermarking and fingerprinting, but they have ignored homologous models trained with similar structures or training datasets. We highlight challenges in efficiently querying black-box piracy models to protect model copyrights without misidentifying homologous models. To address these challenges, we propose a novel method called DeepCore, which discovers that the classification confidence of the model is positively correlated with the distance of the predicted sample from the model decision boundary and piracy models behave more similarly at high-confidence classified sample points. Then DeepCore constructs core points far away from the decision boundary by optimizing the predicted confidence of a few sample points and leverages behavioral discrepancies between piracy and homologous models to identify piracy models. Finally, we design different model identification methods, including two similarity-based methods and a clustering-based method, to identify piracy models using the models' predictions of core points. Extensive experiments show the effectiveness of DeepCore in identifying various piracy models, achieving lower missed and false identification rates, and outperforming state-of-the-art methods.

Keywords: Model Copyright Protection · Piracy · Homologous Model.

1 Introduction

In recent years, deep learning has witnessed rapid development and found extensive applications in various fields, such as computer vision [33], speech recognition [13], and natural language processing [37]. Many companies choose not to open-source deep learning models to protect their commercial interests. This is due to the significant resources required for training advanced neural network models, including massive datasets, substantial computing power, and the expertise of the designers. For example, training models like GPT-3 demand 45TB of data and incur training costs exceeding 12 million US dollars. However, the issue of model copyright faces numerous security threats. Adversaries could obtain white-box models through unconventional means and make modifications [22,27] like fine-tuning, pruning, and adversarial training. Additionally, adversaries can steal models through model extraction attacks [16,31,39,3]. Consequently, there has been an upsurge in research to address these threats [19]. Existing studies can be broadly categorized into two types. The first type employs model watermarking methods [17,36,5,40,1,8], which require modifying the original model and often lead to model performance degradation. Additionally, most watermarking methods struggle to withstand model extraction attacks. The second type adopts model fingerprinting methods [23,2,21,10,32], with current research primarily focusing on decision boundaries. These methods characterize the similarity of decision boundaries using adversarial examples. However, adversaries can manipulate decision boundaries through adversarial training, rendering the fingerprint ineffective. Besides, as shown in Fig. 1, there are homologous models trained by

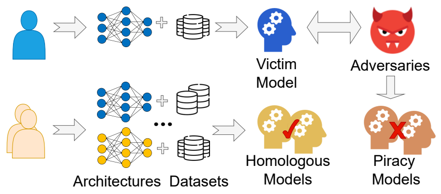


Fig. 1. Homologous models have similar model architectures or training data of the victim model and train independently. Piracy models depend on the victim model through illegal theft of the white-box model or model extraction attacks.

other legitimate users using similar model architectures or datasets. The existence of homologous models could increase the difficulty of identifying piracy models. However, existing work ignored the study of distinguishing homologous models. Consequently, there is an urgent need to safeguard deep neural network models against illegal copying and protect homologous models.

Our research confronts three main challenges. Firstly, we strive to ensure no piracy models are overlooked and avoid misidentifying homologous models. This cannot be easy because similar model architectures and datasets could train similar models. Secondly, the model owner only has black-box access to piracy models and gets the models' predictions of query samples. Thirdly, the model fingerprint not only needs to be effective but also efficient. Using as few query samples as possible can reduce costs and, at the same time, avoid being detected by adversaries. To address these challenges, we propose DeepCore that constructs high-confidence samples named core points to obtain the model fingerprint. Through experimental analysis, we first have three insights: 1) the higher the predicted score of a sample, the farther it is from the model decision boundary, 2) piracy models output scores closer than the homologous models on core points, and 3) the farther the sample is from the model decision boundary, the greater the output score difference between homologous models and piracy models. Core points have more similar predicted scores between piracy models and the victim model due to the similarity of their decision boundaries. So we can identify Homologous and Piracy Models by constructing such high-confidence samples. To solve the black-box limitation, we provide three identification methods of piracy models, including L_1_dist-based, Cos_dist-based, and clustering-based methods to measure the model outputs' correlation. For query efficiency, DeepCore constructs at most one core point for each classification category of the victim model. The main contributions of our work are summarized as follows:

- We propose a simple but novel method called DeepCore to construct a model fingerprint, which can effectively and efficiently identify piracy models without misidentifying homologous models.
- We are the first to discover the behavioral discrepancies on high-confidence classified samples between piracy and homologous models. We have derived three insights through experimental analysis, and we utilize these insights to construct such high-confident samples. Finally, we design different model identification methods using these high-confidence samples.
- Extensive experimental results demonstrate the effectiveness of DeepCore in identifying various piracy models across different architectures and datasets. Specifically, DeepCore can achieve a missed identification rate (MIR) and a false identification rate (FIR) of 0 for piracy models, surpassing the performance of state-of-the-art methods.

2 Related Work

The production of piracy models poses a serious threat to the legitimate rights and interests of model owners. These models can be broadly categorized into the following types: (1) Fine-tuning [34]. (2) Pruning [22,27,11]. (3) Adversarial training [25]. (4) Model extraction attack [16,31]. Many methods have emerged recently to protect the copyrights of model owners, which can be roughly divided into two categories: model watermarking methods and model fingerprinting methods. (1) Model watermarking methods [17,36,5,40,1,8,14,7] need to modify the parameters of the victim model and embed the watermarks carefully designed by the defender in the model, and finally verify whether it is a piracy model by detecting the watermarks. However, most watermarking methods are not effective against model extraction attacks. VEF [20] can survive during the model extraction, but it needs white-box access to the adversary's model. In addition, the watermarking methods can also cause the loss of accuracy [6,8]. (2) Model fingerprint methods [23,2,21,32,10,38,26] do not need to modify the victim model. Instead, these methods determine if a suspected model is a piracy model by analyzing its output behavior using a carefully constructed fingerprint set. Many existing methods [23,2,21,32] utilize adversarial examples to measure the similarity of decision boundaries between the suspected model and the victim model, determining whether the suspected model is a piracy model. However, these methods rely heavily on adversarial examples and are not robust against adversarial defense mechanisms [25]. While MetaFinger [38] fingerprints the inner decision space of the model by meta-training instead of using decision boundaries. However, it does not consider homologous models and piracy models obtained by model extraction attacks. SAC [10] proposes a novel model stealing identification method based on sample correlation, but does not address the false identification of homologous models. Inspired by the ideas of MetaFinger and SAC, we propose DeepCore, a novel approach that utilizes robust samples from the inner decision space strongly associated with the victim model as the fingerprint. Our method can effectively distinguish between different types of piracy models and homologous models.

3 Framework

3.1 Overview

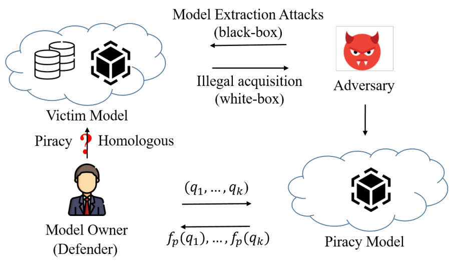


Fig. 2. A threat model of copyright protection of deep models.

Our framework formalizes the threat of model piracy through three models—victim models (f_v), homologous models (f_h), and piracy models ($f_p/f_{\hat{p}}$), and proposes DeepCore to distinguish homologous models from piracy models.

Definition 1 (Victim Model). We denote the victim model training data as $X_v \subset \{(x,y)|x \in [0,1]^M, x \sim \mu, y \in Y = \{1,\ldots,N\}\}$, where y represents the true label and μ represents the data distribution. We denote the victim model trained by X_v as $f_v : [0,1]^M \to R^N$, aiming to optimize: $P_{(x,y)\sim X_v}(\arg\max_i f_v(x)_i = y)$.

Definition 2 (Homologous Model). Homologous models denoted as $f_h : [0,1]^M \to R^N$ are legally trained by others. The training data is denoted as $X_h \subset \{(x,y)|x \in [0,1]^M, x \sim \mu, y \in Y\}$. The overlap ratio between X_h and X_v can be defined by $overlap(X_h, X_v) = \frac{|X_h \cap X_v|}{|X_v|}$. The model architecture can be different from the victim model. Aiming to optimize: $P_{(x,y)\sim X_h}(\arg\max_i f_h(x)_i = y)$.

Definition 3 (Piracy Model). Piracy models obtained by the adversary can be denoted as f_p or $f_{\hat{p}} : [0,1]^M \to R^N$. We define $X_p \subset \{(x,y)|x \in [0,1]^M, x \sim \mu, y \in Y\}$ as the adversary's attack dataset. For piracy models obtained by illegal acquisition of the victim model, the piracy model is defined by $f_p = \Phi_{X_p}(f_v)$, where Φ represents a post-processing operation, such as fine-tuning, pruning, and adversarial training. For model extraction attacks, the adversary aims to optimize the piracy model $f_{\hat{p}}$ as follows: $P_{(x,y)\sim X_p}(\arg\max_i f_{\hat{p}}(x)_i = \arg\max_i f_v(x)_i)$.

3.2 Threat Model

Fig. 2 gives a threat model. The threat model consists of two main entities: an adversary and a defender, who is also the model owner. The adversary can obtain the defender's model in two ways: model extraction attacks and illegally acquiring the white-box model. Then the adversary could use post-processing techniques to modify the model, such as fine-tuning, pruning, and adversarial training, aiming to monetize the model by just providing the model API. Thus, the defender only has black-box access to the adversary's model, denoted as f_p. This means that the defender can only obtain the output of the adversary's model, denoted as $f_p(q_i)$ when providing a query sample q_i. The defender's goal is to identify whether it is a piracy model by analyzing the outputs of the adversary's model.

3.3 DeepCore Design

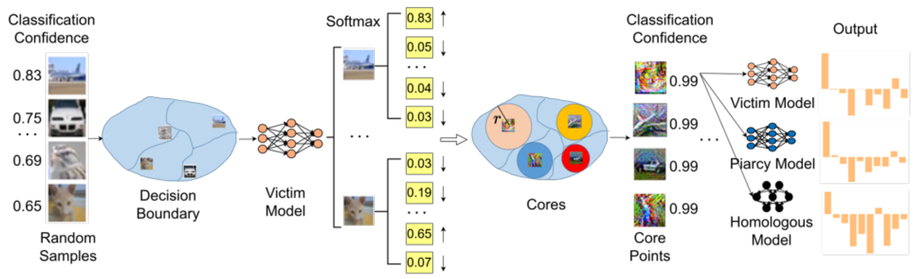


Fig. 3. Given random samples as initial core points, these samples may be near the model decision boundary. For a sample, DeepCore constrains its softmax score so that its classification confidence in a certain category continues to increase and in other categories continues to decrease. Finally, core points far from the decision boundary can be obtained, and the output of the piracy model for these core points is more similar to the victim model than the homologous model.

Fig. 3 gives the DeepCore design. The core idea of DeepCore is to construct samples strongly related to the victim model's classification confidence. DeepCore is based on three insights, which are analyzed in section 4.2. For each category of the model outputs, DeepCore can build high-confidence samples. Due to the behavioral discrepancies on high-confidence classified samples between piracy and ho- mologous models, piracy models have higher classification confidence for such samples, and the output scores are closer to the victim model than homologous models. Here, the score means the model's last-layer logit of the corresponding label, and the confidence means post-softmax probability. DeepCore aims to optimize the i-th core point denoted as ϕ_i of the corresponding label i by the following loss:

$$\text{loss} = -\log \sigma(f_v(\phi_i))_i, \tag{1}$$

where $\sigma : \mathbb{R}^N \to (0,1)^N$ is a standard softmax function defined by the formula $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$, for $i = 1, \ldots, N$, $z = (z_1, \ldots, z_N) \in \mathbb{R}^N$. The goal is to make the victim model's prediction confidence of label i as high as possible.

Algorithm 1: DeepCore

```
Input: The victim model $f_v$, hyper-parameter $\theta$, $\gamma$
Output: Cores $\{B_{r_i}(\phi_i)\}$
 1: for Every label $i \in$ Y do
 2:     Initialize $\phi_i$, $r$, $\Delta$
 3:     while $\Delta > \gamma$ do
 4:         $L = -\log \sigma(f_v(\phi_i))_i$
 5:         update $\phi_i = \phi_i - \theta \nabla L$
 6:         Initialize $j \leftarrow 0$, $\hat{\phi}_j \leftarrow \phi_i$
 7:         while $a : \arg\max_k f_v(\hat{\phi}_j)_k = b : \arg\max_k f_v(\phi_i)_k$ do
 8:             for $l \neq b$ do
 9:                 $\omega_l = \nabla f_v(\hat{\phi}_j)_l - \nabla f_v(\hat{\phi}_j)_b$
10:             end for
11:             $\hat{l} = \arg\min_{l \neq b} \frac{|f_v(\hat{\phi}_j)_l - f_v(\hat{\phi}_j)_b|}{||\omega_l||_2}$
12:             $\delta_j = \frac{|f_v(\hat{\phi}_j)_{\hat{l}} - f_v(\hat{\phi}_j)_b|}{||\omega_{\hat{l}}||_2}$
13:             $\hat{\phi}_{j+1} = \hat{\phi}_j + \delta_j$
14:             $j \leftarrow j + 1$
15:         end while
16:         $r_i = || \sum_j \delta_j ||_2$
17:         $\Delta = |r_i - r|$
18:         $r \leftarrow r_i$
19:     end while
20: end for
21: return $\{B_{r_i}(\phi_i)\} \leftarrow \{\phi_i, r_i | i \in Y\}$
```

Algorithm 1 gives the details on generating the cores. DeepCore defines $B_{r_i}(\phi_i)$ as the victim model core, where r_i is the core radius. To initialize the core point ϕ_i for each label $i \in Y$, DeepCore randomly selects a sample from the distribution μ. Next, DeepCore reduces the loss (1) by gradient descent to update the core point. Lines 6-16 of the algorithm are used to calculate the shortest distance from the current core point to the model decision boundary, which is defined as the core radius. To calculate the radius of the core, DeepFool [28] is leveraged, which can effectively compute the minimum perturbation that causes misclassification in deep neural networks. Through this method, for the core point $\hat{\phi}_j$ we are optimizing, we continuously iterate by adding noise $delta_j$ until the model's classification results change ($l \neq b$). Throughout this process, all noise vectors are accumulated, and their norm $r_i = || \sum_j \delta_j ||_2$ is calculated to determine the current core radius. Then Δ is used to calculate the difference in core radius before and after updating the core points. The algorithm terminates when each core radius converges (i.e., $\Delta < \gamma$, where $\gamma \to 0$). Finally, the model fingerprint denoted as F is defined by $F = \{\phi_i | i = 1, \ldots, N\}$. We can improve query efficiency by further reducing the number of core points with large core radii as the fingerprint.

3.4 DeepCore Identification

We provide three methods to identify homologous models and piracy models. DeepCore constructs core points by optimizing the confidence of samples in specific categories, making the core points predicted by the piracy model have scores closer to the victim model in the corresponding categories. Therefore, the first method uses the L_1 distance between a victim and a suspected model to identify. Given a suspected model $f_s : [0,1]^M \to R^N$, we define the L_1 model distance on the model fingerprint F as follows: $L_1\_dist(f_v, f_s, F) = \sum_{i=1}^{N} |f_v(\phi_i)_i - f_s(\phi_i)_i|$. Note that the core points constructed by DeepCore can make the score of a specific category much higher than that of other categories. Each core point in the fingerprint set is very different; the second method utilizes this difference in categories to identify piracy models. To capture the correlations between the suspected model's outputs on the model fingerprint F, we use cosine similarity, denoted as $Cos(f_s, F)$ [30,10] to express as follows: $Cos(f_s, F)_{ij} = \frac{f_s(\phi_i)^T f_s(\phi_j)}{||f_s(\phi_i)|| ||f_s(\phi_j)||}$. We define the model distance of cosine similarity as follows: $Cos\_dist(f_v, f_s, F) = \frac{||Cos(f_v,F) - Cos(f_s,F)||_1}{N^2}$. By empirically stetting discriminant thresholds d_1, d_2, the suspected model is a piracy model if $L_1\_dist(f_v, f_s, F) < d_1$ or $Cos\_dist(f_v, f_s, F) < d_2$. In the experiments, to determine the thresholds d_1 and d_2, we statistically analyzed the distribution of L1 distances and cosine similarities of various existing models to select suitable thresholds, aiming to maximize the differentiation between homologous and piracy models. However, the first two threshold methods will no longer apply to identify the piracy model types. We use a clustering idea to solve this challenge. For example, we want to successfully identify homologous models, post-processing piracy models, and piracy models by model extraction attacks. Assume post-processing piracy models, denoted as f_p, piracy models obtained by model extraction attacks, denoted as $f_{\hat{p}}$, and homologous models,denoted as f_h, respectively satisfy the distribution π_p, $\pi_{\hat{p}}$ and π_h. we can get three cluster centers denoted as $\{\hat{c_1}, \hat{c_2}, \hat{c_3}\}$ from set $\{o_h, o_p, o_{\hat{p}} | f_p \sim \pi_p, f_{\hat{p}} \sim \pi_{\hat{p}}, f_h \sim \pi_h\}$, where $o_j = (f_j(\phi_1), \ldots, f_j(\phi_N))$, $j \in \{h, p, \hat{p}\}$. We use the fingerprint set as the input to query the suspected model to get an output denoted by $o_s = (f_s(\phi_1), \ldots, f_s(\phi_N))$ and compare the output with the cluster centers to determine which class the suspected model belongs to.

4 Experiments

4.1 Setup

Dataset We use CIFAR-10 and CIFAR-100 [18] as base datasets. CIFAR-10 comprises 60,000 images of size 32×32 pixels, evenly distributed into ten categories. CIFAR-100 has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class.

Victim and Homologous Model Training Data We divide the datasets into three parts, in which the victim model training data, the homologous model training data, and the adversary's attack data ratio is 2:2:1. We set the overlap ratio between the homologous model training set and the victim model training set from 0 to 1 and the interval is 0.1.

Piracy Model Attack Methods We conduct experimental evaluations on the following piracy models: **Fine-tuning [34]**: There are two commonly used fine-tuning methods. One is to use the adversary's dataset to fine-tune only the last layer before the model output, and the other is to fine-tune all the model layers. **Pruning**: The pruning strategy adopted in the experiment is Fine Pruning [22], a combination of pruning and fine-tuning, which shows that it successfully weakens or even eliminates the model backdoor. **Adversarial Training [25]**: To evade the traditional fingerprint method of adversarial samples, the adversary introduces adversarial training to evade identification. **Model Extraction Attacks**: We leverage the label-based model extraction attacks [16,31] and probability-based model extraction attacks [16,35,9].

Homologous Model and Piracy Model Setting All the piracy and homologous models are trained on ResNet [12], DenseNet [15], and we use the ResNet model as the victim model architecture. Table 1 shows the type and number of

Table 1. The type and number of models.

Type	HM_SA	HM_DA	PM_P	PM_FL	PM_FA
Number	10	11	10	10	10
Type	PM_Adv	EM_SA_L	EM_DA_L	EM_SA_Pr	EM_DA_Pr
Number	10	10	10	10	10

test models. HM_SA represents a homologous model trained using the same model architecture as the victim model, HM_DA represents a homologous model trained using a different model architecture, PM_P represents a piracy model obtained by pruning, PM_FL represents a piracy model obtained by Fine-tuning the last layer of the victim model, PM_FA represents a piracy model obtained by Fine-tuning all the layers of the victim model, PM_Adv represents a piracy model obtained by adversarial training, EM_SA_L represents a piracy model obtained by label-based model extraction attacks with the same model architecture as the victim model, EM_DA_L represents a piracy model obtained by label-based model extraction attacks with a different model architecture, EM_SA_Pr represents a piracy model obtained by probability-based model extraction attacks with the same model architecture as the victim model, EM_DA_Pr represents a piracy model obtained by probability-based model extraction attacks with a different model architecture.

Model IP Protection Baselines To validate our method's performance, we compare it with three state-of-the-art works: SAC-w, SAC-m [10]: SAC-w selects wrongly classified samples as model inputs and calculates the mean correlation among their model outputs. SAC-m selects cut-mix augmented samples as model inputs without training the surrogate models or generating adversarial examples. MetaFinger [38]: MetaFinger proposes a robust fingerprint method about the inner decision space of the model by meta-training. FUAP [32]: FUAP proposes a novel and practical mechanism to construct fingerprints by Universal Adversarial Perturbations (UAPs).

Evaluation Metrics To verify the effectiveness of different fingerprint methods, we have two metrics: Missed Identification Rate (MIR): The missed identification rate refers to the ratio of the number of undetected piracy models to the number of all piracy models. False Identification Rate (FIR): The false identification rate refers to the ratio of the number of homologous models detected as piracy models to the number of all homologous models.

Table 2. The core radii and scores of different core points on CIFAR-10.

Radius | score	r_0| s_0	r_1| s_1	r_2| s_2	r_3| s_3	r_4| s_4	r_5| s_5	r_6| s_6	r_7| s_7	r_8| s_8	r_9| s_9
Core$_1$	9.83|23.15	16.94|43.05	14.11|35.27	9.89|31.70	10.67|33.05	15.55|38.14	12.19|32.84	25.31|46.34	14.17|34.79	15.50|40.32
Core$_2$	13.61|29.06	23.51|56.58	20.24|42.35	11.97|37.52	13.64|38.74	18.02|46.31	14.82|38.05	28.88|54.63	18.77|44.79	18.56|46.79
Core$_3$	15.41|32.29	26.97|67.39	21.98|45.92	13.46|41.07	15.88|41.79	22.59|52.24	13.87|41.16	31.10|59.88	25.70|52.20	20.32|50.43

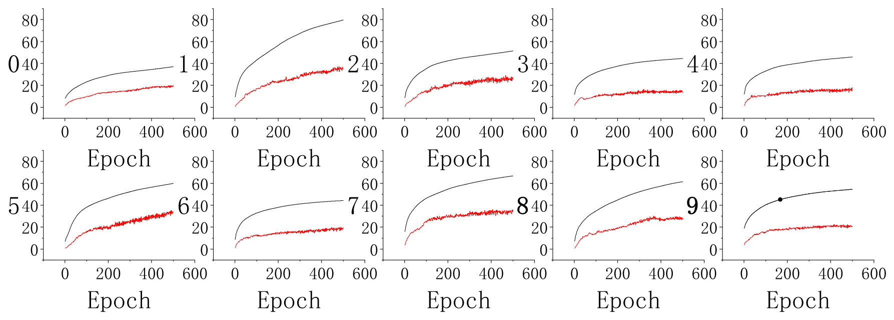


Fig. 4. The black line represents the core point's score of the victim model, and the red line represents the distance from the core point to the model decision boundary. 0-9 represents the label of the current sample.

4.2 Insight Analysis

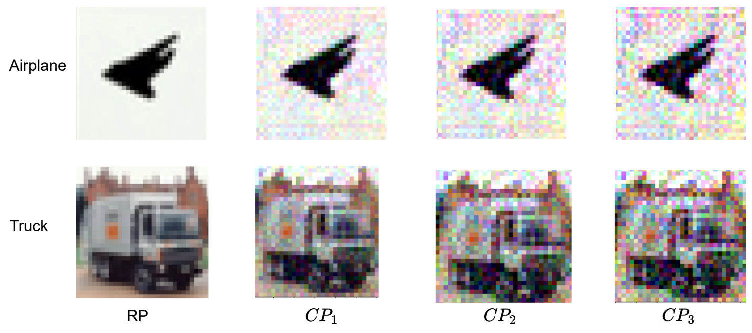


Fig. 5. The visualization of the core points on CIFAR-10. The first row is an airplane. The second row is a truck. RP represents the sample randomly selected as an initial core point, and CP_i represents a core point trained for $i \times 100$ epochs.

Insight 1: The higher the core point's predicted score, the farther the core point is from the model decision boundary

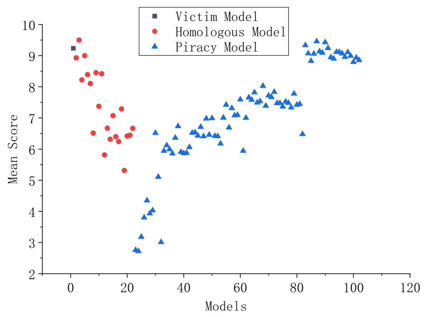
Table 2 shows the core radii and scores of different core points on CIFAR-10. Core$_i$ represents the cores with core points trained for $i \times 100$ epochs. s_i represents the core point's score corresponding to label i, denoted as $f_v(\phi_i)_i$. As the number of training epochs increases, the core radius of each category grows together with the score. Fig. 4 shows that ten core points' scores of the victim model are positively correlated with the core radii within 500 training epochs. These indicate that the higher the confidence of the core point, the further away the core point is from the model decision boundary. Fig. 5 gives the visualization of the core points on CIFAR-10. The more training epochs, the more pixels are changed and the more obvious they are, which leads to poor visual effects.

Insight 2: The piracy models output scores closer to the victim model than the homologous models on core points

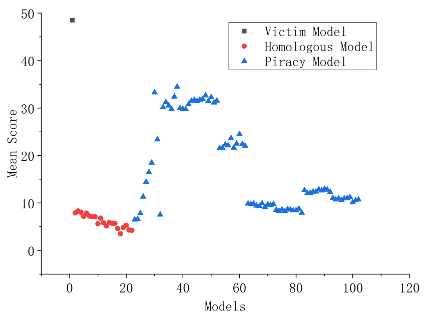
Table 3. The average score difference between different models and the victim model.

Label	0	1	2	3	4	5	6	7	8	9
RP_h	-1.53 ±2.44	0.93±1.72	-0.69±2.85	5.08±1.94	4.39±2.40	0.13±1.41	0.55±2.80	5.95±2.56	-1.68±2.26	5.79± 2.42
RP_p	0.29±1.95	1.02±1.54	1.30±2.07	3.66±1.94	3.13±1.88	1.93±1.36	1.00±1.98	5.47±2.41	-1.01±1.78	4.72±2.59
CP$_1$_h	15.43±2.14	32.96±2.08	27.51±1.87	27.44±2.34	25.62±2.51	32.84±2.18	21.46±2.54	35.55±2.71	26.49±1.92	28.97±2.53
CP$_1$_p	12.86±3.78	24.36±8.76	23.29±5.68	20.02±3.87	20.18±4.95	25.76±6.77	19.08±5.08	28.36±7.02	19.19±7.15	21.29±6.16
CP$_2$_h	22.83±2.23	47.34±2.71	34.64±1.80	35.20±2.73	32.43±2.61	42.71±2.47	26.54±2.31	44.89±3.39	39.36±2.10	36.76±2.70
CP$_2$_p	17.72±5.33	33.06±12.95	28.87±7.46	24.84±5.30	24.59±6.47	32.00±8.79	22.94±6.26	34.32±9.00	26.77±10.69	26.17±7.92
CP$_3$_h	26.96±2.49	58.87±3.36	38.67±1.92	39.18±2.99	36.98±2.22	49.71±2.96	30.79±2.29	51.66±3.59	49.11±2.50	40.92±2.82
CP$_3$_p	20.29±6.30	40.59±16.50	31.71±8.52	27.87±6.32	27.40±7.64	36.96±10.61	25.58±7.06	38.05±10.35	32.74±13.23	29.01±9.06

Fig. 6 shows different models' output scores for different samples. There are 102 models in the figure. The first model is the victim model. Then, according to the order in Table 1 order, the 2nd to the 22nd represent the homologous models,



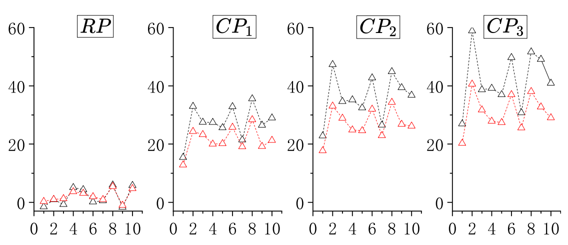
(a) A comparison of different models' scores for random samples.



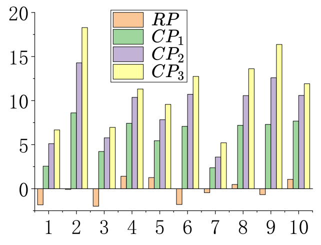
(b) A comparison of different models' scores for core points.

Fig. 6. A comparison of different models' scores for the random samples and core points ϕ_i on CIFAR-10. The x-axis presents different models. The y-axis represents the average prediction score of the model on different samples.

and numbers 23 to 102 represent various piracy models. The piracy models are more sensitive to the core points of the victim model, and most score differences between piracy models and the victim model are more minor. However, for the sample randomly selected, it is difficult to identify the homologous and piracy models from the score difference.



(a) The black triangle symbol represents the label score difference between the victim model and homologous models and the red triangle symbol represents the score difference between the victim model and piracy models.

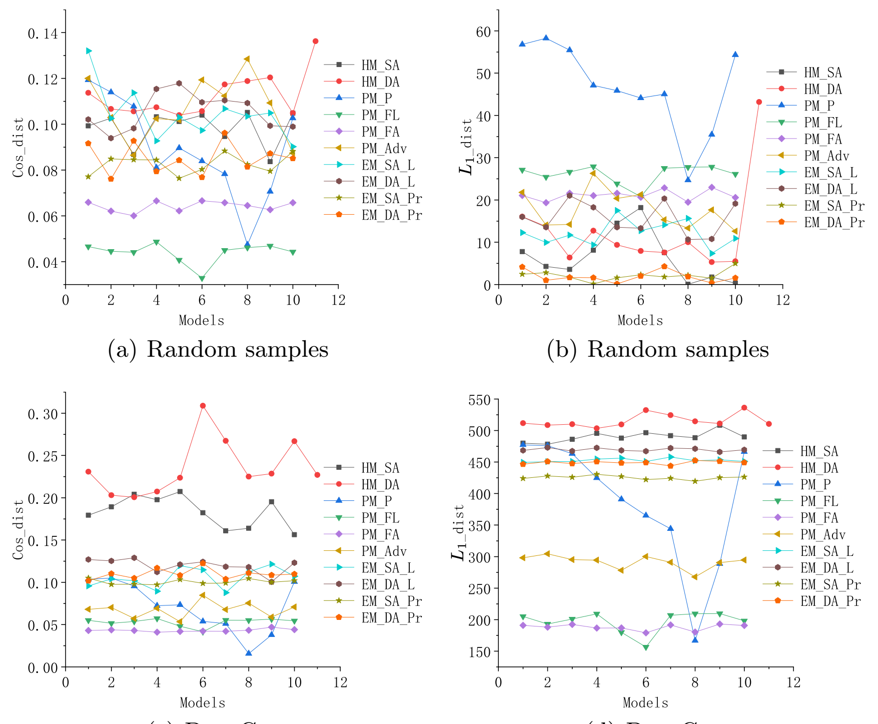


(b) The average score difference between homologous models and piracy models.

Fig. 7. The x-axis in the figure represents ten core points. The y-axis represents the label score difference.

Insight 3: The larger the core point's radius, the greater the core point's score difference between homologous models and piracy models In Table 3, we calculate the average score difference and the variance between different models and the victim model. A label score difference between a suspected model and the victim model is denoted as $f_v(\phi_i)_i - f_s(\phi_i)_i$ for the core point ϕ_i.

RP_h represents the initial core point's score difference between a homologous model and the victim model. RP_p represents such a difference between a piracy model and the victim model. CP$_i$ represents a core point trained for i × 100 epochs. CP$_i$_h represents CP$_i$'s score difference between a homologous model and the victim model. CP$_i$_p represents CP$_i$'s score difference between a piracy model and the victim model. 0 to 9, respectively, represent the 10 RP or CP$_i$ of the corresponding label i. Combined with Fig. 7, we can draw the following conclusions: The larger the core radius, the greater the score difference between homologous models and piracy models. Fig. 4 shows that the larger the epoch, the larger the core radius. Therefore, training core points for more epochs makes it easier to identify piracy models.



(a) Random samples (b) Random samples

(c) DeepCore (d) DeepCore

Fig. 8. The effectiveness of the random samples' optimization of DeepCore.

4.3 DeepCore Performance

The effectiveness of DeepCore Fig. 8 shows the effectiveness of DeepCore compared to initial random samples. If we use random samples as fingerprints, it is difficult to use the threshold to separate the homologous models from the piracy models.

Comparative experiments Table 4 gives the performance comparison between DeepCore and baselines. It can be seen that the DeepCore has the best MIR and FIR. Moreover, DeepCore can perform well under three different identification methods.

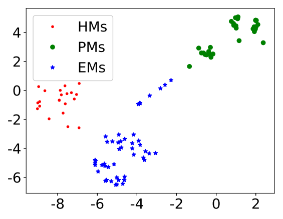
Table 4. Comparative experiments on CIFAR-10 and CIFAR-100.

Method	CIFAR-10		CIFAR-100	
	$MIR\downarrow$	$FIR\downarrow$	$MIR\downarrow$	$FIR\downarrow$
SAC-w [10]	0.05	0.20	0.23	0.43
SAC-m [10]	0.21	0.10	0.61	0.52
FUAP [32]	0.04	0.05	0.06	0.05
MetaFinger [38]	0.07	0.65	0.14	0.94
DeepCore (L_1_dist)	0.03	0.00	0.09	0.05
DeepCore (Cos_dist)	**0.00**	**0.00**	**0.03**	**0.04**
DeepCore *Clustering*	**0.00**	**0.00**	0.00	0.09

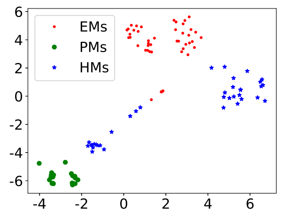
Table 5. The experiment results of clustering methods.

Clustering Methods	KMeans	SC	AC
MIR (PMs)	0.20	0.48	0.23
FIR (PMs)	0.00	0.00	0.00
MIR (EMs)	0.00	0.00	0.00
FIR (EMs)	0.17	0.11	0.20
MIR (HMs)	0.00	0.00	0.00
FIR (HMs)	0.00	0.40	0.00

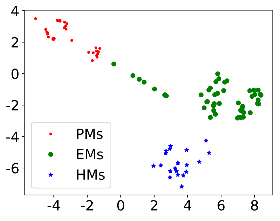
The impact of different clustering methods In the DeepCore identification stage, we can use clustering methods to identify piracy models at a more fine-grained level. In our experiments, we use Kmeans [24], SpectralClustering (SC) [29], and AgglomerativeClustering (AC) [4].



(a) KMeans



(b) SpectralClustering



(c) AgglomerativeClustering

Fig. 9. The visualization of different clustering methods.

Table 5 shows the MIR and FIR of PMs, EMs, and HMs for different clustering methods. PMs refer to models that have undergone post-processing techniques. EMs represent piracy models acquired through model extraction attacks. HMs represent homologous models. Kmeans and AC perform better than SC, which shows that different clustering methods also affect identification accuracy. Fig. 9 intuitively shows the SpectralClustering results of HMs are very scattered and can mistakenly identify piracy models. Although Kmeans and AgglomerativeClustering cannot wholly distinguish different piracy models, they can at least distinguish homologous models well. Therefore, choosing a suitable classification algorithm is also crucial and worthwhile to continue exploring. The clustering-based method can identify piracy models in a more fine-grained manner, but the disadvantage is that the number of types of piracy models needs to be known in advance.

5 Conclusion and Future Work

This work proposes DeepCore, a simple fingerprint construction framework designed to identify piracy and homologous models by exploring the impact of sample points within the model decision space on output behavioral discrepancies. However, the approach has limitations: it requires empirically setting thresholds or cluster numbers, the constructed samples have poor visual effects (which could be mitigated by limiting pixel variation), and questions remain about the minimum number of core points needed for effective fingerprints. Additionally, extending this work to increasingly transformer-based models presents a significant future challenge. Despite these limitations, DeepCore demonstrates improved false and missed identification rates over baselines and contributes to the broader exploration of model decision spaces.

Acknowledgments

The research is partially supported by National Key R&D Program of China 2018YFB0803400, National Key R&D Program of China under Grant No. 2021ZD0110400, China National Natural Science Foundation with No. 62132018, Key Research Program of Frontier Sciences, CAS. No. QYZDY-SSW-JSC002, The University Synergy Innovation Program of Anhui Province with No. GXXT-2019-024.

References

1. Adi, Y., Baum, C., Cisse, M., Pinkas, B., Keshet, J.: Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In: 27th {USENIX} Security Symposium ({USENIX} Security 18). pp. 1615–1631 (2018)
2. Cao, X., Jia, J., Gong, N.Z.: Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In: Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security. pp. 14–25 (2021)
3. Chandrasekaran, V., Chaudhuri, K., Giacomelli, I., Jha, S., Yan, S.: Exploring connections between active learning and model extraction. In: Proceedings of the 29th USENIX Conference on Security Symposium. pp. 1309–1326 (2020)

4. Dasgupta, S., Long, P.M.: Performance guarantees for hierarchical clustering. Journal of Computer and System Sciences **70**(4), 555–569 (2005)
5. Fan, L., Ng, K.W., Chan, C.S.: Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. Advances in neural information processing systems **32** (2019)
6. Fan, L., Ng, K.W., Chan, C.S., Yang, Q.: Deepip: Deep neural network intellectual property protection with passports. IEEE Transactions on Pattern Analysis & Machine Intelligence pp. 1–1 (2021)
7. Fei, J., Xia, Z., Tondi, B., Barni, M.: Wide flat minimum watermarking for robust ownership verification of gans. IEEE Transactions on Information Forensics and Security **19**, 8322–8337 (2024). https://doi.org/10.1109/TIFS.2024.3443650
8. Ge, Y., Wang, Q., Zheng, B., Zhuang, X., Li, Q., Shen, C., Wang, C.: Anti-distillation backdoor attacks: Backdoors can really survive in knowledge distillation. In: Proceedings of the 29th ACM International Conference on Multimedia. pp. 826–834 (2021)
9. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: A survey. International Journal of Computer Vision **129**, 1789–1819 (2021)
10. Guan, J., Liang, J., He, R.: Are you stealing my model? sample correlation for fingerprinting deep neural networks. Advances in Neural Information Processing Systems **35**, 36571–36584 (2022)
11. Guan, J., Tu, Z., He, R., Tao, D.: Few-shot backdoor defense using shapley estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13358–13367 (2022)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
13. Hoy, M.B.: Alexa, siri, cortana, and more: an introduction to voice assistants. Medical reference services quarterly **37**(1), 81–88 (2018)
14. Hua, G., Teoh, A.B.J., Xiang, Y., Jiang, H.: Unambiguous and high-fidelity backdoor watermarking for deep neural networks. IEEE Transactions on Neural Networks and Learning Systems **35**(8), 11204–11217 (2024). https://doi.org/10.1109/TNNLS.2023.3250210
15. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
16. Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A., Papernot, N.: High accuracy and high fidelity extraction of neural networks. In: Proceedings of the 29th USENIX Conference on Security Symposium. pp. 1345–1362 (2020)
17. Jia, H., Choquette-Choo, C.A., Chandrasekaran, V., Papernot, N.: Entangled watermarks as a defense against model extraction. In: USENIX Security Symposium. pp. 1937–1954 (2021)
18. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Handbook of Systemic Autoimmune Diseases **1**(4) (2009)
19. Lederer, I., Mayer, R., Rauber, A.: Identifying appropriate intellectual property protection mechanisms for machine learning models: A systematization of watermarking, fingerprinting, model access, and attacks. IEEE Transactions on Neural Networks and Learning Systems **35**(10), 13082–13100 (2024). https://doi.org/10.1109/TNNLS.2023.3270135
20. Li, Y., Zhu, L., Jia, X., Jiang, Y., Xia, S.T., Cao, X.: Defending against model stealing via verifying embedded external features. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 1464–1472 (2022)

21. Li, Y., Zhang, Z., Liu, B., Yang, Z., Liu, Y.: Modeldiff: testing-based dnn similarity comparison for model reuse detection. In: Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis. pp. 139–151 (2021)
22. Liu, K., Dolan-Gavitt, B., Garg, S.: Fine-pruning: Defending against backdooring attacks on deep neural networks. In: Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21. pp. 273–294. Springer (2018)
23. Lukas, N., Zhang, Y., Kerschbaum, F.: Deep neural network fingerprinting by conferrable adversarial examples. arXiv preprint arXiv:1912.00888 (2019)
24. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. vol. 1, pp. 281–297. Oakland, CA, USA (1967)
25. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
26. Maho, T., Furon, T., Merrer, E.L.: Fingerprinting classifiers with benign inputs. IEEE Transactions on Information Forensics and Security **18**, 5459–5472 (2023). `https://doi.org/10.1109/TIFS.2023.3301268`
27. Molchanov, P., Mallya, A., Tyree, S., Frosio, I., Kautz, J.: Importance estimation for neural network pruning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11264–11272 (2019)
28. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2574–2582 (2016)
29. Ng, A., Jordan, M., Weiss, Y.: On spectral clustering: Analysis and an algorithm. Advances in neural information processing systems **14** (2001)
30. Nguyen, H.V., Bai, L.: Cosine similarity metric learning for face verification. In: Asian conference on computer vision. pp. 709–720. Springer (2010)
31. Orekondy, T., Schiele, B., Fritz, M.: Knockoff nets: Stealing functionality of black-box models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4954–4963 (2019)
32. Peng, Z., Li, S., Chen, G., Zhang, C., Zhu, H., Xue, M.: Fingerprinting deep neural networks globally via universal adversarial perturbations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13430–13439 (2022)
33. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1701–1708 (2014)
34. Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B., Liang, J.: Convolutional neural networks for medical image analysis: Full training or fine tuning? IEEE transactions on medical imaging **35**(5), 1299–1312 (2016)
35. Truong, J.B., Maini, P., Walls, R.J., Papernot, N.: Data-free model extraction. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4771–4780 (2021)
36. Uchida, Y., Nagai, Y., Sakazawa, S., Satoh, S.: Embedding watermarks into deep neural networks. In: Proceedings of the 2017 ACM on international conference on multimedia retrieval. pp. 269–277 (2017)
37. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

38. Yang, K., Wang, R., Wang, L.: Metafinger: Fingerprinting the deep neural networks with meta-training. In: Thirty-First International Joint Conference on Artificial Intelligence. pp. 759–765 (2022)
39. Zanella-Beguelin, S., Tople, S., Paverd, A., Köpf, B.: Grey-box extraction of natural language models. In: International Conference on Machine Learning. pp. 12278–12286. PMLR (2021)
40. Zhang, J., Chen, D., Liao, J., Zhang, W., Hua, G., Yu, N.: Passport-aware normalization for deep model protection. Advances in Neural Information Processing Systems **33**, 22619–22628 (2020)