DeepJudge: A Testing Framework for Copyright Protection of Deep Learning Models

DEEPJUDGE: A Testing Framework for Copyright Protection of Deep Learning Models

Jialuo Chen*, Youcheng Sun†, Jingyi Wang*, Peng Cheng*✉, Xingjun Ma‡
*Zhejiang University, †University of Manchester, ‡Fudan University
*{chenjialuo, wangjyee, lunarheart@zju.edu.cn}, †{youcheng.sun@manchester.ac.uk},‡{xingjunma@fudan.edu.cn}

Abstract—**Deep learning (DL) models have become one of the most valuable assets in modern society, and those most complex ones require millions of dollars for the model development. As a result, unauthorized duplication or reproduction of DL models can lead to copyright infringement and cause huge economic losses to model owners.**

In this work, we present DEEPJUDGE, a testing framework for DL copyright protection. DEEPJUDGE quantitatively tests the similarities between two DL models: a victim model and a suspect model. It leverages a diverse set of testing metrics and efficient test case generation algorithms to produce a chain of supporting evidence to help determine whether a suspect model is a copy of the victim model. Our experiments confirm the effectiveness of DEEPJUDGE under typical model copyright infringement scenarios. The tool has been made publicly available at https://github.com/Testing4AI/DeepJudge. A demo video can be found at https://www.youtube.com/watch?v=LhNeo615YOE.

I. INTRODUCTION

Deep learning models, e.g., deep neural networks (DNNs), have become the standard models for solving many complex real-world problems, such as image recognition [12], natural language processing [7], and autonomous driving [5]. However, training such models is by no means trivial, which requires not only large-scale datasets but also significant computational resources. It is thus of utmost importance to protect DNNs from unauthorized duplication or reproduction.

Recent studies have shown that stealing a DNN can be done very efficiently without leaving obvious traces [20], [19]. Arguably, unauthorized fine-tuning or pruning is the most straightforward way of model stealing, if the model parameters are publicly accessible (for research purposes only) or the adversary is an insider. Even when only the API is exposed, the adversary can still exploit advanced *model extraction* techniques [18], [3] to steal most functionalities of the hidden model. These attacks pose serious threats to the copyright of DL models, calling for effective protection methods.

DNN Watermarking The watermarking techniques have been the overwhelming approach to protect the copyright of DL models, which work in two main steps: *embedding* and *verification*. In the embedding step, the owner embeds a secret watermark into the model during the training process by exploiting the over-parameterization property of DNNs [1]. Depending on how much knowledge of the model is available in the verification step, existing watermarking methods can be broadly categorized into two classes: a) *white-box* methods for the case when model parameters are available; and b) *black-box* methods when only predictions of the model can be acquired. White-box watermarking embeds a pre-designed signature (e.g., a string of bits) into the parameter space of the model via certain regularization terms [8], [21]. The ownership could be claimed when the extracted signature from a suspect model is similar to that of the owner model. Black-box watermarking usually leverages backdoor attacks [11] to implant a watermark pattern into the owner model by training the model with a set of backdoor examples (also known as the trigger set) relabeled to a secret class [14], [23]. The ownership can then be claimed when the defender queries the suspect model for examples attached with the watermark trigger and receives the secret class as predictions.

However, watermarking is invasive, i.e., it needs to tamper with the model training procedure, which may affect the model's utility. Since the effectiveness of watermarking completely depends on how well the model memorizes the watermark, these techniques are also particularly vulnerable to the emerging model extraction attack that only extracts the key functionality of the model [6], [15], however, the embedded watermark is often task-irrelevant.

Very recently, a novel testing approach has been proposed for DL copyright protection [6]. In particular, it defines a family of complementary metrics to actively measure the similarities between a victim (owner) model and a suspect model from multiple angles. In this paper, we present in details DEEPJUDGE that implements the DL copyright testing theory in [6]. As illustrated in Fig. 1, DEEPJUDGE is composed of three major components. First, a set of multi-level testing metrics are used to fully characterize a DNN model from different angles. Second, efficient test case generation algorithms magnify the similarities (or differences) measured by the testing metrics between the two models. Finally, a 'yes'/'no' judgment, on whether the suspect model is a stolen copy, will be made for by the judging mechanism.

Compared to existing watermarking-based defenses, advantages of DEEPJUDGE include: 1) *non-invasive*, as it works directly on the model and does not tamper with the training process; 2) *efficient*, as it only needs a small set of seed test cases and a quick scan of the two models; 3) *flexible*, as it can easily incorporate new testing metrics or test case generation methods to obtain a more confident judgement, and can be applied in both the white-box and black-box settings; and 4) *robust* to various model stealing attacks.

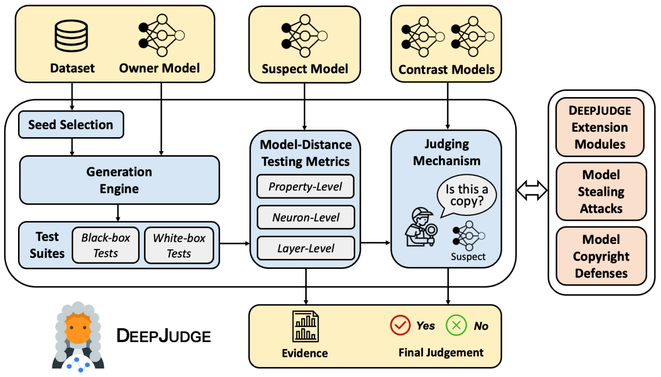

Fig. 1: The overview of DEEPJUDGE Testing Framework.

We have made DEEPJUDGE publicly available as an open-source toolkit and our experimental results confirm the effectiveness of DEEPJUDGE in providing strong evidence for identifying the stolen copies of a victim model.

II. THE DEEPJUDGE FRAMEWORK

The overall architecture of DEEPJUDGE tool is shown in Fig. 1. In principle, DEEPJUDGE takes the owner model (a.k.a. victim model), a set of data associated with the model and a suspect model as inputs, and it returns the measured values according to the testing metrics as evidence, as well as the final judgement. In order to produce more reliable judgement, DEEPJUDGE advocates evidence-based ownership verification via multi-level testing metrics that complement each other and smart test case generation methods to fully exercise the metrics.

Specifically, the tool has the following features.

a) Seed Selection: Seed selection prepares the seeds examples used to generate the test suites. By default, we apply the sampling strategy DeepGini [9] to select a set of high-certainty seeds from the provided dataset. The intuition is that high-certainty seeds are well-learned by the victim model, thus carrying more unique features of the victim model. More adaptive seed selection strategies are supported to fight against adaptive model attacks.

b) Multi-level Testing Metrics: Multi-level testing metrics comprehensively characterize the differences between models. The six testing metrics in DEEPJUDGE are summarized in Table I, with their suitable defense settings highlighted in the last column. In the white-box setting, DEEPJUDGE has full access to the internals of the suspect model, while in the black-box Setting, DEEPJUDGE can only query the suspect model to obtain the probability vectors or the predicted labels. We refer more details about the metrics to our previous work [6].

c) Test Case Generation Engine: Test Case Generation Engine generates the test suites to fully exercise the defined testing metrics. The engine respects the suspect model accessibility in the two defense settings. In the black-box setting, we integrate existing adversarial attacks (e.g., FGSM [10], PGD [17] and C&W [4]) to populate the test suite on the victim model. For the white-box setting, a more fine-grained fuzzing algorithm [6] is developed to explore the corner region [16] of each neuron's activation for a given hidden layer.

TABLE I: Proposed multi-level testing metrics.

Level	Metric	Defense Setting
Property-level	Robustness Distance (RobD)	Black-box
Neuron-level	Neuron Output Distance (NOD)	White-box
	Neuron Activation Distance (NAD)	White-box
Layer-level	Layer Outputs Distance (LOD)	White-box
	Layer Activation Distance (LAD)	White-box
	Jensen-Shanon Distance (JSD)	Black-box

d) Test Suites: The test suites are generated from DEEPJUDGE respectively for the two defense settings, where white-box tests are a set of extreme test cases with neuron locations, and black-box tests are a set of adversarial inputs with the corresponding ground-truth labels.

e) Judging Mechanism: With the metrics values, the judging mechanism identifies whether the suspect model is a copy of the owner model by two steps: thresholding and voting. The thresholding step determines a proper threshold for each testing metric based on the statistics of contrast models that are models trained independently from the owner model. The voting step examines the suspect model against each testing metric, and gives a positive vote if the measured distance is lower than the threshold. DEEPJUDGE identifies a positive copy if the suspect model gets a positive vote on more than half of the metrics.

f) Evidence and Judgement: DEEPJUDGE reports the judgement with supporting evidence. The judgement answers whether the suspect model is a copy of the victim model, and evidence are the measured multi-level metric values.

g) Extension Modules: As a highly flexible and extensible testing tool, more advanced testing metrics, generation algorithms, judging rules and functional extensions can be effortlessly incorporated into DEEPJUDGE to fight against diverse copyright threats in the arms race between DEEPJUDGE and the adversary. For instance, the neuron matching plug-in can help DEEPJUDGE defend against one new emergent model copyright threat, i.e., shuffling attacks [22].

h) Model Stealing Attacks: Typical model copyright threats, e.g., model fine-tuning, pruning, shuffling, and extraction attacks, have been implemented in the tool to help evaluate the effectiveness and robustness of copyright defenses in a standard attack setting.

i) Model Copyright defenses: DEEPJUDGE is a strong complement to existing model copyright defenses, and representative state-of-the-art works are implemented in the tool to help better understand the role of DEEPJUDGE, including both watermarking [23], [21] and fingerprinting [2].

A. Example Usage

The DEEPJUDGE framework supplies the users with a list of command line options. Here we explain its usage through several examples. For instance, the following command calls

DeepJudge to select the seeds for feeding the generation engine. The option --*model* and --*dataset* specifies the owner model and the provided dataset associated with the model. The option --*order* sets the selection rule and by *max*, the tool will select seed samples with high-certainties. The option --*num* is the number of seeds. The selected seeds then will be automatically stored in directory *seeds*.

```
python seed_selection.py --model victim.h5
       --dataset cifar10 --order max --num 1000
       --output seeds
```

Next, the command below calls DeepJudge to generate the white-box test suite for the victim model, which will be deposited into directory *tests*. The --*seeds* sets up the selected seed samples used for generation in the above step. The option --*layer* and --*neuron* set up the layer index and neuron index to test respectively. The option --*iters* 1000 configures the maximum iterations for the gradient optimization loop.

```
python whitebox_generation.py --model victim.h5
       --seeds seeds.npy --layer 3 --neuron 0
       --iters 1000 --output tests
```

Finally, DeepJudge calculates the similarity between the victim model (--*model*) and the suspect model (--*suspect*), by white-box testing metrics. The option --*tests* specifies the test suite used for model testing, and the results will be automatically stored in directory *results*.

```
python whitebox_evaluation.py --model victim.h5
       --suspect suspect.h5 --tests white_tests.npy
       --output results
```

III. Experiments

In this section, we demonstrate the effectiveness of DeepJudge against model fine-tuning and model pruning attacks, which are two main copyright threats extensively studied by existing defenses [1], [8], [21]. We run the experiments on the CIFAR-10 dataset [13]. The victim model is trained with ResNet-20 [12] structure, which is also provided as an example model in the repository.[1]

Positive suspect models are derived from the victim model via two types of stealing attacks, including three fine-tuning strategies, i.e., fine-tune the parameters of the last layer and all layers, and retrain all the layers (denoted as FT-LL, FT-AL and RT-AL), and two weight pruning strategies with 20% and 60% pruning rate (denoted as P-20% and P-60%). Data-augmentation techniques are also used to strengthen these attacks. These models are considered as stolen copies of the owner model, and DeepJudge should provide evidence for the ownership claim for them. Negative suspect models are trained independently using the same (Neg-1) or similar data (Neg-2). These models serve as the control group to make sure that DeepJudge does not claim wrong ownership, and are also used to calculate thresholds for the judging mechanism.

[1]Experiments in this paper focus on demonstrating the utility of DeepJudge tool. Advantages of DL copyright testing against other defenses including watermarking can be found in [6].

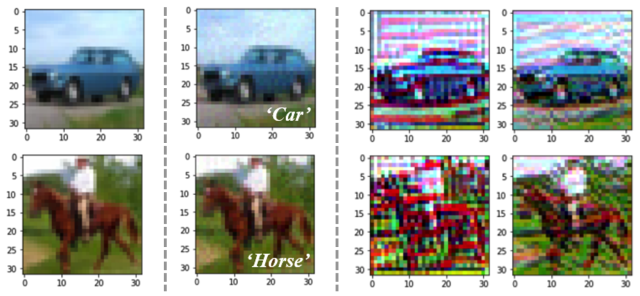

Fig. 2: Example test cases generated in the black-box and white-box settings. Note that the white-box test cases are not regular inputs and are unlabeled.

We apply the sampling strategy DeepGini [9] to select a set of high-certainty seeds from the victim dataset for the generation. For the black-box test suite, untargeted PGD-L^∞ attack is adopted, with 0.03 bound and 10 steps. For the white-box test suite, layer 6 with all the neurons are tested.

Our experiments are conducted on a work station with a 2.40GHz Intel Xeon E5 and NVIDIA GTX 1080 Ti. Overall, the generation process costs (seconds) about 4s for the black-box test suite (with matrix acceleration) and about 1,200s for the white-box test suite. This time cost is regarded as efficient since the test case generation is a one-time effort, and the costs of the metric evaluations are negligible (less than 1s).

Fig. 2 demonstrates several example test cases generated in the two settings. In DeepJudge, test inputs are used to measure the similarity between two models, and they are not necessarily of high picture quality or even human readable.

Fig. 3 shows the testing results for each metric respectively. We repeat the experiment multiple times and report the average value with standard deviation. For all the metrics, the smaller the value, the more similar the suspect model is to the victim model. Clearly, all positive suspect models are more similar to the victim model with significantly smaller metric values than negative suspect models. That is, the two sets of models are completely separable, leading to highly accurate detection of the positive copies. The gap between positive suspect models and negative ones in the white-box setting is larger than the black-box setting, which is not surprising as white-box testing can collect more fine-grained information from the suspect models. In both the black-box and white-box settings, all the positives (orange) are below the threshold while the negatives (blue) exceed the threshold, thus the voting step in DeepJudge overwhelmingly supports the correct final judgement. It is worth reminding that, compared to watermarking [1], [21], [8], DeepJudge does not need to tamper with the model training procedure.

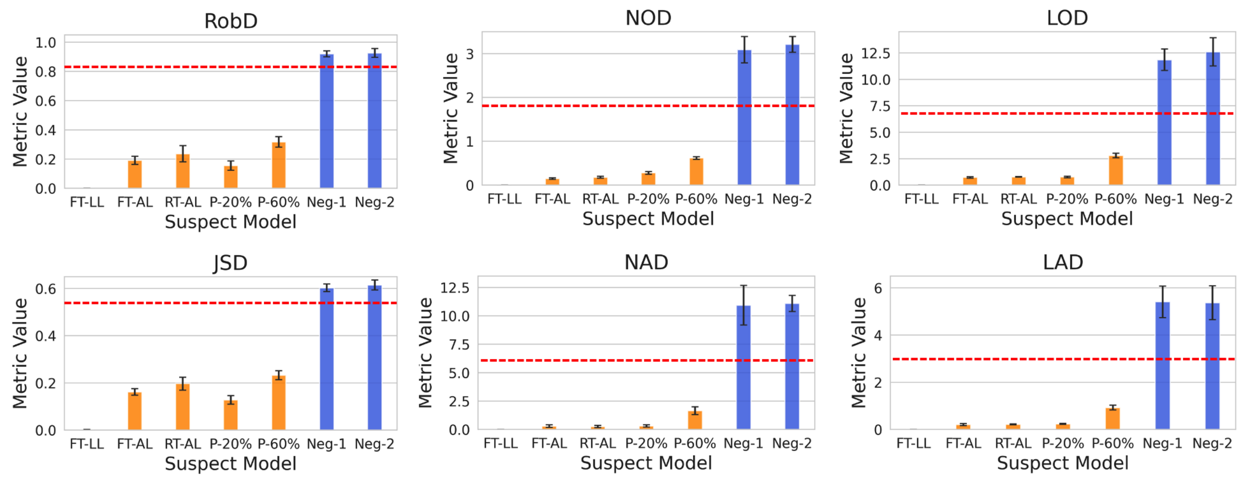

Fig. 3: Metric values (distances) of different suspect models to the victim model. The red dotted line is the calculated threshold for each testing metric. We use the orange color for the positive suspect models and the blue color for the negatives.

IV. Conclusion

DeepJudge is a novel testing framework for copyright protection of DL models. The core of DeepJudge is a family of multi-level testing metrics that characterize different aspects of similarities between the victim model and a suspect model. Efficient and flexible test case generation methods are also developed in DeepJudge to help boost the discriminating power of the testing metrics. As a generic testing framework, DeepJudge is applicable in both black-box and white-box settings, new testing metrics or generation methods can be effortlessly incorporated into DeepJudge to help defend against future threats. We foresee that there is an arms race between the model owner and the adversary, and we believe DeepJudge will play an indispensable role in the long run of model copyright protection.

Acknowledgement

This research was supported by the Key R&D Program of Zhejiang (2022C01018) and the NSFC Program (62102359, 61833015, U1911401).

References

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX Security*, 2018.

[2] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Asia CCS*, pages 14–25, 2021.

[3] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic extraction of neural network models. In *CRYPTO*. Springer, 2020.

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *S&P (Oakland)*, pages 39–57. IEEE, 2017.

[5] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deep-Driving: Learning affordance for direct perception in autonomous driving. In *ICCV*, pages 2722–2730, 2015.

[6] Jialuo Chen, Jingyi Wang, Tinglan Peng, Youcheng Sun, Peng Cheng, Shouling Ji, Xingjun Ma, Bo Li, and Dawn Song. Copy, right? A testing framework for copyright protection of deep learning models. In *IEEE S&P (Oakland)*, pages 824–841, 2022.

[7] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[8] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. DeepSigns: an end-to-end watermarking framework for ownership protection of deep neural networks. In *ASPLOS*, pages 485–497, 2019.

[9] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. DeepGini: prioritizing massive tests to enhance the robustness of deep neural networks. In *ISSTA*, pages 177–188, 2020.

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv:1412.6572*, 2014.

[11] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[13] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[14] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244, 2020.

[15] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. Sok: How robust is image classification deep neural network watermarking? In *S&P (Oakland)*, pages 787–804. IEEE, 2022.

[16] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *ASE*, 2018.

[17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[18] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *CVPR*, 2019.

[19] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Asia CCS*, pages 506–519, 2017.

[20] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *USENIX Security*, pages 601–618, 2016.

[21] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *ICMR*, 2017.

[22] Yifan Yan, Xudong Pan, Yining Wang, Mi Zhang, and Min Yang. "and then there were none": Cracking white-box dnn watermarks via invariant neuron transforms. *arXiv:2205.00199*, 2022.

[23] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Asia CCS*, pages 159–172, 2018.