Fingerprinting deep neural networks - a DeepFool approach

Wang, Si; Chang, Chip Hong

2021

Wang, S. & Chang, C. H. (2021). Fingerprinting deep neural networks - a DeepFool approach. 2021 IEEE International Symposium on Circuits and Systems (ISCAS).
<https://dx.doi.org/10.1109/ISCAS51556.2021.9401119>

<https://hdl.handle.net/10356/147023>

<https://doi.org/10.1109/ISCAS51556.2021.9401119>

© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at:
<https://doi.org/10.1109/ISCAS51556.2021.9401119>.

Downloaded on 26 Apr 2023 22:40:27 SGT

Fingerprinting Deep Neural Networks - A DeepFool Approach

Si Wang and Chip-Hong Chang

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

Email: si003@e.ntu.edu.sg, echchang@ntu.edu.sg

Abstract—A well-trained deep learning classifier is an expensive intellectual property of the model owner. However, recently proposed model extraction attacks and reverse engineering techniques make model theft possible and similar quality deep learning solution reproducible at a low cost. To protect the interest and revenue of the model owner, watermarking on Deep Neural Network (DNN) has been proposed. However, the extra components and computations due to the embedded watermark tend to interfere with the model training process and result in inevitable degradation in classification accuracy. In this paper, we utilize the geometry characteristics inherited in the DeepFool algorithm to extract data points near the classification boundary of the target model for ownership verification. As the fingerprint is extracted after the training process has been completed, the original achievable classification accuracy will not be compromised. This countermeasure is founded on the hypothesis that different models possess different classification boundaries determined solely by the hyperparameters of the DNN and the training it has undergone. Therefore, given a set of fingerprint data points, a pirated model or its post-processed version will produce similar prediction but another originally designed and trained DNN for the same task will produce very different prediction even if they have similar or better classification accuracy. The effectiveness of the proposed Intellectual Property (IP) protection method is validated on the CIFAR-10, CIFAR-100 and ImageNet datasets. The results show a detection rate of 100% and a false positive rate of 0% for each dataset. More importantly, the fingerprint extraction and its run time are both dataset independent. It is on average $\sim 130\times$ faster than two state-of-the-art fingerprinting methods.

I. INTRODUCTION

It is not an overstatement to call 2020 a decade of deep learning, with the sighting of so many previously intractable problems in computer vision and natural language processing it solved elegantly. The successes in its applications to real-world problems have been made possible by training the deep neural network (DNN) model with enormous amount of labeled data and computing power. Both require huge capital investment in manpower, time and physical resources, which may not be affordable or accessible by small corporations. The trained model is a pricey proprietary asset of a business and is normally not disclosed publicly. Even then, due to the strong incentive, model extraction [1]–[4] and reverse engineering [5] have been used by business opponents to rebuild similar AI product or solution to the same problem from a trained model to gain unfair competitiveness. Unfortunately, it is difficult to prove the ownership of a DNN model when the model is implemented onto different hardware platforms.

Existing countermeasures to detect this deep learning IP theft attack can be broadly dichotomized into watermarking and fingerprinting. The former can be achieved via either white-box or black-box approaches. A white-box strategy often embeds a secret message into the model internals during training. It extracts the watermark from a suspect DNN model and compares if it is similar to the target model [6]. However, this approach requires physical access to the suspect model for verification, which is not always feasible before getting into the judicial procedure. In contrast, the black-box strategy trains the target model on a small set of modified inputs acquired through either content perturbation or wrong labeling, apart

from the original dataset [7]–[9]. This set of data is then served as the trigger to produce a unique prediction for the target model. During verification, this trigger set is submitted to query the suspect model and the prediction results are compared against those obtained from the target model. The accessibility issue is resolved by sacrificing classification accuracy in most techniques of watermark embedding through the model training process. Unlike watermarking, no accuracy loss is incurred in fingerprinting methods, as signature embedding is avoided in the training process of the target model. It aims to generate robust inputs that have similar prediction results as the target model for a pirated or post-processed version of the target model, but are otherwise different for an original model [10], [11]. This set of robust inputs and their labels predicted by the target model form the fingerprint of the target model.

Previous fingerprinting methods either omit the effect of different fingerprints extracted using the same mechanism on the detection efficiency or incur a comparatively large latency overhead in fingerprint extraction. In this paper, we propose a fast IP theft detection technique that makes use of DeepFool [12] algorithm to extract fingerprint of a target model. DeepFool intrinsically generates data points lying near the decision boundary. As a classifier can be characterized by its decision boundary, the generated data points as well as their labels predicted by the target model will be leveraged for fingerprinting. The matching rate, which is the proportion of fingerprint data points that have the same labels predicted by both the suspect and target classifiers, is used to optimize the fingerprint performance. The influence of different sets of fingerprint data points on the matching rate is investigated. Selected data points are encoded and verified by Bayes' Theorem [13] to provide a confidence level to determine if a suspect classifier possesses a valid fingerprint of a target model. An iterative process is utilized to obtain a prediction with either extremely high confidence that indicates a high likelihood of the suspect classifier being not original (i.e., pirated) or extremely low confidence that indicates a high possibility of the suspect classifier being original. The proposed DeepFool based fingerprinting method is evaluated on CIFAR-10, CIFAR-100 and ImageNet datasets. Four prevalent post-processing techniques that an attacker may apply to the pirated target model to evade detection are also considered.

The rest of this paper is organized as follows. Section II briefly introduces the DeepFool algorithm and the post-processing techniques. Section III elaborates the proposed fingerprinting approach. Section IV presents and discusses the simulation results. Section V concludes the paper.

II. PRELIMINARIES

A. DeepFool

DeepFool [12] is an adversarial attack that adds imperceptibly small perturbations to the benign inputs to delude the target classifier into wrong predictions. It makes use of geometry property to explore the minimum perturbations required to cross the decision boundary of the target classifier for a given input. This attack algorithm simplifies the DNN model to an

entirely linear structure where the classes are separated by hyperplanes. The problem is then formulated into computing the distance between the input point and the closest hyperplane. To generalize the solution to the nonlinear classifier, DeepFool approximates the boundary of the initially correct class label of the point of interest iteratively by borrowing the idea from Newton's iterative algorithm.

B. Post-processing of Deep Neural Networks

It is common that an attacker post-processes the stolen model for transfer learning, performance improvement, or model compression before deployment. Four post-processing techniques are introduced as follows.

Feature Extraction (FE): The weights of the final layer of the target model are updated using the training data. All other layers are fixed.

Fine-tuning (FT): All the parameters of the target model are updated using the training data.

Weight Pruning (WP): Less significant weights of the target model are pruned for producing lightweight model. The pruned model is usually retrained to enhance the classification accuracy.

Filter Pruning (FP): Less significant filters of the target model are pruned for model compression. The pruned model is usually retrained to restore the classification accuracy.

III. PROPOSED FINGERPRINTING APPROACH

A. Threat Model

Two parties are modeled in our threat model: model owner and attacker. The model owner has trained and deployed a DNN classifier as an AI service to the users. The target model internals are kept confidential. The attacker is assumed to be capable of stealing or pirating the parameters as well as the architecture of the target DNN classifier and deploying it as its own service. This is made possible via exploitation of side channel leakage such as cache misses [3], [14], memory access pattern [4], [15], power trace [16] and performance counters [17] during the model execution. The attacker may post-process the pirated model for performance improvement or adaptation to resource-constrained platforms before the deployment. The model owner could derive a fingerprint from his own classifier and use it to validate against a suspect classifier. To be realistic and to avoid dispute, we assume that the model owner can only perform the verification on a suspect in a black-box manner without being able to access to or intervene with its internal signals or computations. If the suspect classifier can produce a similar fingerprint as a target model, it is a positive suspect of pirating. The model owner can take further actions such as collecting additional evidences before filing a lawsuit. Otherwise, it is a negative suspect.

B. Proposed IP Protection Approach

Our proposed IP protection approach is based on the assumption that a deep learning classifier can be characterized by its unique classification boundary. In other words, different classifiers will have different classification boundaries. Therefore, fingerprinting the classification boundary of the target classifier could give convincing evidence for model ownership verification. Generally, we extract a set of data points near the classification boundary of the target classifier and their labels predicted by the target classifier as the fingerprint. If the predicted labels after querying a suspect classifier with the extracted fingerprint data points largely match those in the fingerprint, the suspect classifier is likely to be the pirated or post-processed version of the target classifier. The definition of

Fig. 1: A given clean input and its corresponding adversarial input generated by DeepFool.

the classification boundary as well as the details of our fingerprinting extraction and verification processes are elaborated as follows:

Classification Boundary: Given an k -class classifier, its classification boundary is defined as follows [10]:

$$CB = \{x | \exists i, j, i \neq j \text{ and } g_i(x) = g_j(x) \geq \max_{t \neq i, j} g_t(x)\} \quad (1)$$

where CB is a set of input data points building up the classification boundary of the classifier, $g_i(x)$, $g_j(x)$ and $g_t(x)$ are the probabilities that the input data x is predicted to be in class i , j and t , respectively. In other words, a data point is claimed to be on the decision boundary of a given classifier if at least two classes have the same largest probability for this input data such that the classifier cannot decide the label of this input data.

Fingerprint Extraction: Given a target classifier C_t and a set of n randomly selected input data points $X = \{x_1, x_2, \dots, x_n\}$, the proposed method extracts a fingerprint by $f_{C_t} = \text{Extract}(C_t, X)$, where f_{C_t} and Extract are the fingerprint and the DeepFool algorithm, respectively. Since the set of n data points $\hat{X} = \{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n\}$ are generated from DeepFool by adding only minimum perturbations to the corresponding given inputs X , they are selected from those located nearest to the decision boundary. Fig. 1 shows an example for a given input and its corresponding adversarial input generated by DeepFool. It can be observed that there is only very minor deviation between the largest probability and the second largest probability predicted for the adversarial input. The extracted fingerprint for an input X is given by $f_{C_t} = \{(\hat{x}_1, \hat{y}_1), (\hat{x}_2, \hat{y}_2), \dots, (\hat{x}_n, \hat{y}_n)\}$, where $\hat{x}_i$ and $\hat{y}_i$ are the i -th fingerprint data point and its label predicted by the target classifier C_t .

Fingerprint Verification: Given a suspect DNN classifier C_s , the DNN model owner executes $\text{Verify}(f_{C_t}, C_s)$ to output a probability P as the confidence level of C_s to be a pirated or post-processed version of C_t . If $P \geq \tau_{pos}$, C_s is predicted to be similar to C_t with high confidence, and C_s is claimed to be a positive suspect classifier. If $P \leq \tau_{neg}$, C_s is predicted to be a different classifier unrelated to C_t with high confidence, and C_s is said to be a negative suspect classifier. If $\tau_{neg} < P < \tau_{pos}$, there is not enough confidence to claim or deny C_s as a pirated copy of C_t . In this situation, another set of randomly selected inputs are fed into the function $\text{Extract}()$ to generate another fingerprint from C_t . The verification function $\text{Verify}()$ is executed with the probability P calculated from both the newly produced fingerprint and the previous fingerprint. The process continues until P falls into either $[\tau_{pos}, 1]$ or $[0, \tau_{neg}]$.

In $Verify()$, $f_{C_s} = \{(\widehat{x}_1, y_1'), (\widehat{x}_2, y_2'), \dots, (\widehat{x}_n, y_n')\}$ are generated, where y_i' is the label predicted by C_s for the i -th fingerprint data point $\widehat{x}_i$. Then, the matching rate $\frac{m}{n}$ is calculated, where m is the number of fingerprint data points that have the same labels predicted by C_s and C_t . Finally, P is calculated by applying the Bayes' Theorem [13]:

$$P = p(pos|\mathbf{r}) = \frac{p(pos)p(\mathbf{r}|pos)}{p(\mathbf{r})} \quad (2)$$

where pos and $\mathbf{r}$ are a positive class and an interval vector $(r_1, r_2, \dots, r_N)$, respectively. r_i represents the range or interval of the matching rate for the i -th fingerprint. In order to calculate P , a frequency or prior probability table needs to be established based on the predetermined matching rate intervals. Therefore, M negative suspect classifiers and M positive suspect classifiers are generated. K fingerprints are produced and applied to each suspect classifier to investigate the convergence of the matching rate. Then, (2) can be further simplified as follows:

$$P = \frac{\sum_{j=1}^M \prod_{i=1}^N p_j(r_i|pos)}{\sum_{j=1}^M \prod_{i=1}^N p_j(r_i|pos) + \sum_{j=1}^M \prod_{i=1}^N p_j(r_i|neg)} \quad (3)$$

where $p_j(r_i|pos)$ represents the likelihood for the matching rate to fall in the interval r_i if it is the j -th positive suspect classifier, and $p_j(r_i|neg)$ represents the likelihood for the matching rate to fall in the interval r_i if it is the j -th negative suspect classifier.

IV. IMPLEMENTATIONS AND RESULTS

A. Experimental Setup

1) *Datasets and Target Classifiers*: Three popular image classification datasets are considered in the experiment: CIFAR-10, CIFAR-100 and ImageNet-10. CIFAR-10 contains a total of 60000 color images with a resolution of 32×32 in 10 classes. It has 50000 images for training and 10000 images for testing. A pretrained ResNet20 [18] model is used as the target classifier for CIFAR-10 [19]. Similar to CIFAR-10, CIFAR-100 has the same number of color images with the resolution of 32×32 for training and testing, except that it is a 100-class classification problem. A pretrained ResNet20 [18] model is used as the target classifier for CIFAR-100 [19]. Unlike CIFAR-10 and CIFAR-100, ImageNet is a dataset of color images with high resolution, provided by the ImageNet Large Scale Visual Recognition Challenge 2012. It consists of 1.2 million training images and 50000 validation images, all of which are labeled with 1000 categories. However, due to the limited memory, only the former 10 classes are taken for our evaluation. Therefore, only 12000 training images and 500 validation images are utilized in our experiment. We distinctively name this dataset as ImageNet-10. A fine-tuned ResNet18 [18] model is used as the target classifier for ImageNet-10.

2) *Suspect Classifiers*: There are two types of suspect classifiers: positive and negative, as defined in Section III-A. Four commonly used post-processing techniques that can be applied to a pirated model are considered: feature extraction (FE), fine-tuning (FT), weight pruning (WP) and filter pruning (FP). For weight pruning, p fraction of weights that have the smallest absolute values are pruned, followed by a retraining to restore the classification accuracy. To limit the compromise of classification accuracy, p is initialized to 0.1 and incremented at a step size of 0.1 until the accuracy loss is larger than or equal to 3%. For filter pruning, c fraction of filters that have the smallest sum of the absolute value of each entry in each layer are pruned, followed by a retraining to enhance the accuracy. c is initialized to $\frac{1}{16}$ and incremented with a step size of $\frac{1}{16}$ until the accuracy loss is larger than or equal to 3%. DNN models that

TABLE I: Testing accuracies of the target and suspect classifiers for the three datasets. “-” means not applicable.

Classifier type	Classifiers	CIFAR-10	CIFAR-100	ImageNet-10
Target	Resnet20	91.65%	66.61%	-
	Resnet18	-	-	86.60%
Positive	FE	91.89%	65.62%	87.40%
	FT	91.92%	66.38%	86.00%
	WP	$p = 0.1$	91.78%	66.86%
		$p = 0.2$	91.78%	66.63%
		$p = 0.3$	91.79%	66.46%
		$p = 0.4$	91.56%	66.22%
		$p = 0.5$	91.41%	64.66%
		$p = 0.6$	91.10%	64.66%
	FP	$c = 1 \setminus 16$	91.07%	65.19%
		$c = 2 \setminus 16$	90.45%	64.48%
		$c = 3 \setminus 16$	90.04%	-
	ResNet32	92.81%	68.74%	-
	ResNet34	-	-	88.20%
	ResNet44	93.24%	69.49%	-
	ResNet50	-	-	90.20%
Negative	ResNet56	93.69%	70.79%	-
	VGG16	93.91%	32.77%	90.60%
	DenseNet121	94.14%	62.90%	90.20%
	GoogleNet	92.73%	59.19%	-
	MobileNet-v2	94.17%	65.50%	-
	Inception-v3	-	63.78%	-
	AlexNet	-	-	82.80%
	SqueezeNet	-	-	84.40%

TABLE II: Datasets, target classifiers and numbers of positive and negative suspect classifiers.

Dataset	Target classifier	# Positive	# Negative
CIFAR-10	ResNet20	11	7
CIFAR-100	ResNet20	10	8
ImageNet-10	ResNet18	7	6

have architectures different from the target classifier for each dataset are trained as the negative suspect classifiers. Details about the target and suspect classifiers are shown in Table I. The number of positive and negative suspect classifiers used for each dataset is summarized in Table II.

B. Results

The number of data points for each fingerprint is set to be 100 ($n = 100$). To investigate the convergence of the matching rate, 100 fingerprints are generated for each target classifier ($K = 100$). Table III shows the matching rate range as well as the average matching rate for each suspect classifier for the 100 fingerprints. It can be observed that positive suspect classifiers have larger matching rates than the negative suspect classifiers for a given fingerprint most of the time. Furthermore, the matching rate of a positive suspect weight-pruned classifier decreases with increasing p . Similarly, the matching rate of a positive suspect filter-pruned classifier decreases with increasing c . This is attributable to the possibly larger variation in the decision boundary when more weights or filters are pruned. It is noted that the matching rates of the positive suspect classifiers all fall in the range of [0.15, 0.66] and the average matching rates over the 100 fingerprints of the positive suspect classifiers all fall in [0.2366, 0.5491], regardless of post-processing technique, dataset and fingerprint. The matching rates of the negative suspect classifiers all fall in [0.02, 0.29] and the average matching rates over the 100 fingerprints of the negative suspect classifiers all fall in [0.0663, 0.1595]. There is a gap of approximately 0.08 between the average matching rates of positive suspect classifiers and those of the negative suspect classifiers, but there is an overlap interval [0.15, 0.29] for the matching rates.

To increase the separability between positive and negative suspect classifiers with high confidence, the verification is iteratively processed as proposed in Section III-B. Based on the overlap interval, the matching rates are divided into three

TABLE III: Matching rate range and average matching rate for various suspect classifiers across 100 fingerprints.

Type	Model	CIFAR-10	CIFAR-100	ImageNet-10
Positive	FE	[(0.40, 0.66], 0.5303)	[(0.27, 0.52], 0.4023)	[(0.24, 0.46], 0.3457)
	FT	[(0.28, 0.49], 0.3668)	[(0.23, 0.46], 0.3390)	[(0.21, 0.40], 0.2877)
	$p = 0.1$	[(0.28, 0.52], 0.4164)	[(0.34, 0.57], 0.4415)	[(0.44, 0.66], 0.5491)
	$p = 0.2$	[(0.30, 0.50], 0.4041)	[(0.32, 0.56], 0.4255)	[(0.40, 0.65], 0.5387)
	$p = 0.3$	[(0.29, 0.53], 0.3959)	[(0.27, 0.52], 0.3989)	[(0.32, 0.59], 0.4732)
	$p = 0.4$	[(0.24, 0.46], 0.3528)	[(0.26, 0.49], 0.3687)	[(0.21, 0.49], 0.3876)
	$p = 0.5$	[(0.24, 0.44], 0.3283)	[(0.23, 0.44], 0.3201)	[(0.17, 0.40], 0.2718)
	$p = 0.6$	[(0.20, 0.43], 0.2958)	[(0.18, 0.39], 0.2827)	-
	$c = 1 \setminus 16$	[(0.18, 0.43], 0.3109)	[(0.20, 0.44], 0.3024)	-
	$c = 2 \setminus 16$	[(0.17, 0.43], 0.2823)	[(0.16, 0.35], 0.2571)	-
Negative	$c = 3 \setminus 16$	[(0.15, 0.35], 0.2366)	-	-
	ResNet32	[(0.08, 0.29], 0.1595)	[(0.07, 0.24], 0.1506)	-
	ResNet34	-	-	[(0.02 , 0.16], 0.0883)
	ResNet44	[(0.07, 0.28], 0.1594)	[(0.06, 0.23], 0.1467)	-
	ResNet50	-	-	[(0.03, 0.15], 0.0781)
	ResNet56	[(0.08, 0.29], 0.1557)	[(0.08, 0.23], 0.1507)	-
	VGG16	[(0.04, 0.23], 0.1394)	[(0.03, 0.16], 0.0809)	[(0.02 , 0.13], 0.0663)
	DenseNet121	[(0.05, 0.23], 0.1307)	[(0.04, 0.19], 0.1205)	[(0.02 , 0.14], 0.0683)
	GoogleNet	[(0.08, 0.26], 0.1550)	[(0.05, 0.20], 0.1219)	-
	MobileNet-v2	[(0.07, 0.28], 0.1523)	[(0.05, 0.23], 0.1435)	-
	Inception-v3	-	[(0.06, 0.20], 0.1259)	-
	AlexNet	-	-	[(0.03, 0.16], 0.0851)
	SqueezeNet	-	-	[(0.03, 0.18], 0.0894)

TABLE IV: Matching rate vectors $\mathbf{m}$ and probability vectors $\mathbf{P}$ of various suspect classifiers for CIFAR-10, CIFAR-100 and ImageNet-10 datasets using the proposed method.

Type	Model	CIFAR-10	CIFAR-100	ImageNet-10
Positive	FE	0.53, 1	0.40, 1	0.34, 1
	FT	0.38, 1	0.37, 1	0.31, 1
	$p = 0.1$	0.40, 1	0.34, 1	0.52, 1
	$p = 0.2$	0.43, 1	0.38, 1	0.51, 1
	$p = 0.3$	0.36, 1	0.40, 1	0.41, 1
	$p = 0.4$	0.27, 0.94	0.34, 1	0.32, 1
	$p = 0.5$	0.32, 1	0.38, 1	0.23, 1
	$p = 0.6$	0.32, 1	0.32, 1	-
	$c = 1 \setminus 16$	0.3, 1	0.3, 1	-
	$c = 2 \setminus 16$	0.32, 1	0.3, 1	-
Negative	$c = 3 \setminus 16$	0.29, 0.94	-	-
	ResNet32	0.13, 0	0.16, 0	-
	ResNet34	-	-	0.08, 0
	ResNet44	(0.2, 0.19, 0.1), (0.48, 0.17, 0)	0.16, 0	-
	ResNet50	-	-	0.06, 0
	ResNet56	0.14, 0	0.14, 0	-
	VGG16	0.14, 0	0.15, 0	0.05, 0
	DenseNet121	0.17, 0.07	0.14, 0	0.04, 0
	GoogleNet	0.13, 0	0.14, 0	-
	MobileNet-v2	0.18, 0.07	0.18, 0	-
	Inception-v3	-	0.12, 0	-
	AlexNet	-	-	0.06, 0
	SqueezeNet	-	-	0.05, 0

bins: [0, 0.15), [0.15, 0.3) and [0.3, 1]. Since matching rate for a given positive classifier never falls into the bin [0, 0.15) and matching rate for a given negative suspect classifier never falls into the bin [0.3, 1], only the middle interval [0.15, 0.3) needs to be further divided to reduce the number of iterations during verification, this interval is evenly divided into three bins: [0.15, 0.2), [0.2, 0.25) and [0.25, 0.3). The positive and negative thresholds for the confidence level are set to be 0.9 and 0.1 ($\tau_{pos} = 0.9$, $\tau_{neg} = 0.1$), respectively. Based on these parameters, a prior probability table with respect to a dataset can be built for the probability calculation required by *Verify()* for an unseen fingerprint. Table IV presents the matching rate vectors $\mathbf{m} = (\frac{m_1}{n}, \frac{m_2}{n}, \dots, \frac{m_N}{n})$ and probability vectors $\mathbf{P} = (P_1, P_2, \dots, P_N)$ of the various suspect classifiers for CIFAR-10, CIFAR-100 and ImageNet-10 datasets using the proposed method, where $\frac{m_i}{n}$, P_i and N are the matching rate for the i -th newly generated fingerprint, the probability output of *Verify()* in the i -th round and the total number of iterations required for the verification, respectively. It can be seen that all the suspect classifiers can be correctly recognized with a high confidence in one round, except only the ResNet44 model for CIFAR-10, which requires three rounds to validate.

C. Comparison With Two Recent Fingerprinting Methods

Table V summarizes the comparison between this work and the state-of-the-art fingerprinting methods. The convergence metric refers to the demonstration of the upper and lower boundaries of the matching rate for a suspect classifier with various different fingerprints extracted from a target classifier.

TABLE V: Comparison with two recent works.

Metrics	This work	Cao's [10]	Lukas's [11]
Method	DeepFool	Adam optimizer	C-BIM
Convergence	Yes	No	No
#Test Inputs (n)	100	100	-
True Positive Rate	100%	100%	100%
False Positive Rate	0%	0%	0%
Matching Rate Gap	0.05	0.62	0.05
Run Time (s)	CIFAR-10	20	37.8
	CIFAR-100	17	249.9
	ImageNet	24	7634.3

True Positive Rate (TPR) and False Positive Rate (FPR) are calculated by:

$$TPR = TP / (TP + FN) \quad (4)$$

$$FPR = FP / (FP + TN) \quad (5)$$

where TP, FN, FP and TN are acronyms for true positives, false negatives, false positives and true negatives. Matching rate gap is the difference between the minimum matching rate in positive suspect classifiers and the maximum matching rate in negative suspect classifiers. Run time is the time elapsed for the fingerprint extraction, which is executed on a PC equipped with an Intel(R) Core(TM) i7-7700HQ 2.80GHz CPU, 16GB system memory, and a GeForce GTX 1060Ti.

It can be seen that although Cao's work has a much larger matching rate gap than the proposed work, the proposed work is much faster than Cao's work. While Cao's work has a rapid growth in run time from CIFAR-10 to ImageNet, our work has a relatively stable execution time independent of dataset. One unique advantage of our method is the extraction of fingerprint data points is image agnostic. This has greatly enhanced the non-repudiability of ownership proof and piracy detection as the model can be validated by using fingerprint data points extracted from an arbitrary set of images. The other two works did not demonstrate this possibility.

V. CONCLUSIONS

A DeepFool based fingerprinting method is proposed to detect the DNN IP theft attack. The fingerprint data points are generated by DeepFool from an arbitrary set of input data points and their labels predicted by the target classifier. Difference in matching rates between a pirated classifier and an originally designed classifier is observed for a given fingerprint. The Bayes' Theorem is applied to provide a confidence level for a verification result to enhance its non-repudiability. An iterative verification is adopted to increase the confidence of a correct detection of a pirated classifier or its post processed version. Simulation results show 100% detection rate and 0% false positive rate with significantly faster fingerprint extraction from the target model for verification than two recently reported fingerprinting methods. Our future work is to enlarge the matching rate gap between the positive and negative suspect classifiers to increase the robustness of the fingerprint. The splitting of matching rate interval will also be improved accordingly to further enhance the verification efficiency.

ACKNOWLEDGEMENT

This research is supported by the National Research Foundation, Singapore, under its National Cybersecurity Research & Development Programme/Cyber-Hardware Forensic & Assurance Evaluation R&D Programme (Award: CHFA-GC1-AW01).

REFERENCES

- [1] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, “Stealing machine learning models via prediction apis,” in *25th USENIX Security Symposium*, Austin, TX, Aug. 2016, pp. 601–618.
- [2] B. Wang and N. Z. Gong, “Stealing hyperparameters in machine learning,” in *2018 IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, May 2018, pp. 36–52.
- [3] M. Yan, C. W. Fletcher, and J. Torrellas, “Cache telepathy: Leveraging shared resource attacks to learn dnn architectures,” in *USENIX Security Symposium*, Aug. 2020.
- [4] X. Hu *et al.*, “Neural network model extraction attacks in edge devices by hearing architectural hints,” *CoRR*, vol. abs/1903.03916, 2019. [Online]. Available: <http://arxiv.org/abs/1903.03916>
- [5] S. J. Oh, B. Schiele, and M. Fritz, “Towards reverse-engineering black-box neural networks,” in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 2019, pp. 121–144.
- [6] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, “Embedding watermarks into deep neural networks,” in *Proc. 2017 ACM Int. Conf. Multimedia Retrieval*, Bucharest, Romania, Jun. 2017, pp. 269–277. [Online]. Available: <https://doi.org/10.1145/3078971.3078974>
- [7] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, “Turning your weakness into a strength: Watermarking deep neural networks by back-dooring,” in *Proc. 27th USENIX Security Symposium (USENIX Security 18)*, Baltimore, MD, Aug. 2018, pp. 1615–1631. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity18/presentation/adi>
- [8] J. Zhang *et al.*, “Protecting intellectual property of deep neural networks with watermarking,” in *Proc. 2018 Asia Conf. Computer Communications Security*, Incheon, Republic of Korea, Jun. 2018, pp. 159–172. [Online]. Available: <https://doi.org/10.1145/3196494.3196550>
- [9] B. Darvish Rouhani, H. Chen, and F. Koushanfar, “Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks,” in *Proc. 24th Int. Conf. Architectural Support Programming Languages Operating Systems*, Providence, RI, USA, Apr. 2019, pp. 485–497. [Online]. Available: <https://doi.org/10.1145/3297858.3304051>
- [10] X. Cao, J. Jia, and N. Z. Gong, “IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary,” 2020. [Online]. Available: <http://arxiv.org/pdf/1910.12903.pdf>
- [11] N. Lukas, Y. Zhang, and F. Kerschbaum, “Deep neural network fingerprinting by conferrable adversarial examples,” 2020. [Online]. Available: <http://arxiv.org/pdf/1912.00888.pdf>
- [12] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “Deepfool: a simple and accurate method to fool deep neural networks,” in *Proc. IEEE Conf. Computer Vision Pattern Recogn.*, Las Vegas, Nevada, USA, Jun. 2016, pp. 2574–2582.
- [13] J. Joyce, “Bayes’ theorem,” 2003.
- [14] S. Hong *et al.*, “Security analysis of deep neural networks operating in the presence of cache side-channel attacks,” *CoRR*, vol. abs/1810.03487, 2018. [Online]. Available: <http://arxiv.org/abs/1810.03487>
- [15] H. Naghibijouybari, A. Neupane, Z. Qian, and N. Abu-Ghazaleh, “Rendered insecure: GPU side channel attacks are practical,” in *Proc. 2018 ACM SIGSAC Conf. Comput. Communications Security*, Toronto, Canada, Jan. 2018, pp. 2139–2153. [Online]. Available: <https://doi.org/10.1145/3243734.3243831>
- [16] L. Batina, S. Bhasin, D. Jap, and S. Picek, “CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel,” in *Proc. 28th USENIX Security Symposium (USENIX Security 19)*, Santa Clara, CA, Aug. 2019, pp. 515–532. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity19/presentation/batina>
- [17] M. Alam and D. Mukhopadhyay, “How secure are deep learning algorithms from side-channel based reverse engineering?” in *Proc. 56th Annual Design Automation Conference 2019*, Las Vegas, NV, USA, Jun. 2019. [Online]. Available: <https://doi.org/10.1145/3316781.3322465>
- [18] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. 2016 IEEE Conf. Computer Vision Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
- [19] Y. Idelbayev, “Proper ResNet implementation for CIFAR10/CIFAR100 in PyTorch,” https://github.com/akamaster/pytorch_resnet_cifar10, accessed: 2020-09-20.