Hybrid Fingerprinting for Effective Detection of Cloned Neural Networks

Can Aknesil[1] Elena Dubrova[1]
Niklas Lindskog[2] Jakob Sternby[2] Håkan Englund[2]

[1]KTH Royal Institute of Technology, Stockholm, Sweden
{aknesil, dubrova}@kth.se

[2]Ericsson AB, Lund, Sweden
{niklas.lindskog, jakob.sternby, hakan.englund}@ericsson.com

Abstract

As artificial intelligence plays an increasingly important role in decision-making within critical infrastructure, ensuring the authenticity and integrity of neural networks is crucial. This paper addresses the problem of detecting cloned neural networks. We present a method for identifying clones that employs a combination of metrics from both the information and physical domains: output predictions, probability score vectors, and power traces measured from the device running the neural network during inference. We compare the effectiveness of each metric individually, as well as in combination. Our results show that the effectiveness of both the information and the physical domain metrics is excellent for a clone that is a near replica of the target neural network. Furthermore, both the physical domain metric individually and the hybrid approach outperformed the information domain metrics at detecting clones whose weights were extracted with low accuracy. The presented method offers a practical solution for verifying neural network authenticity and integrity. It is particularly useful in scenarios where neural networks are at risk of model extraction attacks, such as in cloud-based machine learning services.

Keywords—fingerprinting, neural networks, intellectual property, model extraction, power side channels.

1 Introduction

With the growing deployment of artificial intelligence in various applications [1], protecting the authenticity and integrity of neural networks (NN) becomes increasingly important. A significant threat is the *cloning* of NNs, where an adversary obtains a replica of a legitimate NN. This process, known as *model extraction*, comes in multiple flavors. Depending on the extraction method used by the adversary, the clone can be related to the original model in different ways. For instance, the clone can be an exact copy of the original model, a near replica that has similar, but not equal, weights, or it can be trained using the original model's predictions, known as *retraining* [2].

Identifying cloned NNs is critical for safeguarding against their unauthorized use and ensuring that the deployed models' decisions are reliable. A promising approach to tackle this problem is the *fingerprinting* of NNs. By creating a unique fingerprint based on specific parameters of a NN, we can determine whether a clone has been derived from another model. It is also important to be able to distinguish between a clone M_1' of a neural network M_1 and a legitimate neural network M_2, trained independently on a different dataset from the one used for training M_1.

In this paper, an extended version of [3], we propose a novel NN fingerprinting method that combines output predictions, probability score vectors, and power traces. We show that integrating both information and physical domains into a composite fingerprint provides a highly effective way of identifying cloned models.

Our main contributions are summarized as follows:

- We introduce a hybrid method for fingerprinting NNs using a combination of output predictions, probability score vectors, and power traces.
- We demonstrate that the effectiveness of the hybrid method is excellent, with a balanced accuracy ranging from 99.92% to 100% for clones extracted with high accuracy.
- We also show that both the physical domain metric individually, and the hybrid approach are more effective than the information domain metrics at detecting a clone extracted with low accuracy.
- For fingerprinting, three different sets of inputs, namely (1) independent inputs taken from the testing dataset of the original model, (2) inputs taken from the training dataset of the original model, and (3) randomly generated inputs that do not have corresponding labels, have been used and compared.
- It is demonstrated that using randomly generated inputs instead of samples from the testing dataset of the original model further improves effectiveness at detecting models extracted with low accuracy.

The source code and data of our experiments are available at `https://github.com/canaknesil/hybrid-fingerprinting-ext`.

The rest of the paper is organized as follows. Section 2 provides the necessary background. Section 3 reviews the related work. Section 4 describes the adversary model. Section 5 details the presented method for fingerprinting NNs. Section 6 describes the experimental setup. Section 7 presents the experimental results. Section 8 discusses future directions. Section 9 concludes the paper.

2 Background

This section provides background for neural networks and model extraction attacks on neural networks.

2.1 Neural Networks

A *neural network* is a computational model inspired by the human brain, consisting of interconnected nodes (neurons) organized in layers. It can be used for tasks such as classification, regression, and pattern recognition. NNs have become the cornerstone of modern artificial intelligence, powering applications ranging from image recognition to natural language processing [1].

2.2 Model Extraction Attacks on Neural Networks

NNs are susceptible to various types of extraction attacks, including:

Direct Access In these attacks, an adversary obtains unauthorized access to the original NN and ends up having an exact copy [4].

Numerical Analysis In this category, the adversary tries to extract weights of the original model through numerical analysis [2, 5]. The resulting clone either has the same weights as the original model, or weights that are similar to the original model.

Side-Channel Attacks These attacks exploit physical phenomena, such as power consumption or electromagnetic emissions, to gain insights into the internal operation of a NN [6]. A clone created through side-channel analysis typically has similar weights to the original model.

Retraining In these attacks, an adversary aims to replicate the functionality of a target model by querying it and using the responses to train a surrogate model [2]. A surrogate model typically has very different weights from the original model, even though the adversary uses the same architecture as the original model, due to the random initialization of the weights at the beginning of training.

2.3 Fingerprinting of Neural Networks

Fingerprinting of NNs involves creating unique identifiers that can be used to verify the identity of a NN. This technique can help detect cloned models and protect intellectual property.

The concept of fingerprinting NNs has been explored in various studies, including [7], which provides an overview of methods to create unique fingerprints for NNs. It is important to ensure that a fingerprint is difficult to remove, as there are various fingerprint removal techniques [8].

3 Previous Work

The identification of cloned NNs and the protection of model integrity have been areas of active research. Two approaches for solving the problem are *model fingerprinting* and *watermarking*. Both approaches are still under development, however, as stated in [9], fingerprinting is currently the most promising technique for ownership verification. In this

section, we focus on previous work in the area of NN fingerprinting, as well as fingerprint removal attacks.

3.1 Neural Network Fingerprinting Methods

The majority of existing NN fingerprinting methods introduce a distance measure to quantify the similarity between a given pair of NNs. One of the methods, called *dataset inference*, measures a given suspect model's confidence regarding samples drawn from the original model's training set [10]. They observe that a clone has a distinguishably better confidence on the training dataset of the original model than an independently trained model. As presented in the subsequent work [11], dataset inference is shown to have high false positive and false negative rates.

Another work proposes a method that extracts from an original model a set of *conferrable* examples that transfer exclusively to the clone, and are not present in an independently trained model [12]. A distance measure, introduced in [13], is based on the similarity of score vectors for inputs specifically selected to reflect decision boundaries. In [14], score vectors are compared for inputs for which the predicted classes differ. Multiple distance measures are proposed in [15] based on not only the model output, that is, class prediction and score vector, but also on intermediate layer outputs. In [16] a given set of models is compared according to the gradient of input points with respect to the model output.

To the best of our knowledge, the only previous work that incorporates power side channels into NN fingerprinting is [17]. The concept of model fingerprinting assumed in [17] is rather specific compared to the bulk of the existing work in this area, including the presented work. They assume that a clone is not identical to the original model, thus, its fingerprint should be different from the original model's. In the contrary, we make the more general assumption that the clone may or may not be identical to the original model. In both cases, the clone's fingerprint should relate the clone to the original model. The focus of [17] is Binary Neural Networks (BNN) running on an FPGA. They introduce an input selection method to be used during fingerprint evaluation, leveraging the unpredictable behavior of the model for samples that are distinct from the training samples. Their method achieves 100% accuracy at differentiating two models that are trained using the same training dataset. Due to the specific nature of their fingerprinting method, their results are not directly comparable to ours.

In addition to the works mentioned above, in [18], it has been proposed to extract a fingerprint from a NN by saving checkpoints during training, called the *training chain*.

3.2 Fingerprint Removal Attacks

Fingerprint removal aims to reduce the dependence of a clone on the original model's training dataset. A fingerprint removal approach is to modify a model through fine-tuning, or model compression [8].

Retraining with different architectures and adversarial training can also be used for fingerprint removal. In [12], a removal attack called the *ground-truth attack* is proposed. The attack is based on the adversary having a fraction of the ground-truth labels, and using them during retraining to make the clone less dependent on the original model.

Another removal approach is to reduce the effectiveness of fingerprint extraction by altering the response of a cloned NN, for the inputs that may be used to extract a fingerprint [19].

4 Adversary and Arbiter Models

This section defines our assumptions on the adversary and the arbiter in the context of NN fingerprinting.

The primary objectives of the adversary are

1. to clone the target NN by replicating its functionality,
2. to evade detection of the cloned NN as a replica of the target NN,
3. to utilize the cloned NN for purposes such as unauthorized use or distribution.

We assume that the adversary has knowledge and resources to implement one of the model extraction attacks explained in section 2.2. This also requires access to the target NN, its prediction API, or the physical device running the inference, to implement an extraction attack. The adversary also has knowledge of the target NN's architecture and replicates the same architecture for the clone.

The primary objective of the arbiter is to correctly identify a suspect NN with the highest possible accuracy.

We assume that the arbiter has access to a given pair of NNs (a target and a suspect) through a prediction API or a similar interface that allows querying the NNs with input data. We consider various scenarios where the arbiter has capability to collect the following sets of information from the target and the suspect for a range of inputs:

1. Output predictions
2. Probability score vectors (imply output predictions)
3. Power traces
4. Power traces along with output predictions
5. Power traces along with probability score vectors

In scenarios where power traces are required, the arbiter has physical access to the device running the inference or can measure power consumption indirectly through remote side-channel monitoring [20]. When a testing dataset is used for fingerprinting, the arbiter has access to or is capable of generating an independent testing dataset. When the training dataset of the original model is used, the arbiter has access to a portion of the training dataset. When random inputs are used, the arbiter does not need any dataset as the inputs can easily be generated. Finally, the arbiter has access to computational resources for analyzing and comparing collected metrics, namely output predictions, probability score vectors, and power traces.

5 Hybrid Fingerprinting Method

This section describes the new method for identifying cloned NNs which combines metrics from both the information and the physical domains to create a NN fingerprint.

5.1 Overview

The proposed fingerprinting method consists of three primary metrics: output predictions, probability score vectors, and power traces. Each metric provides distinct information about the model, and their combination enhances the accuracy of distinguishing between cloned and legitimate NNs. First, we define similarity measures for each of these metrics.

5.2 Information Domain Metrics

Output Predictions Output predictions refer to the class labels or predictions generated by the model for a given set of inputs. By analyzing these predictions, we can infer certain characteristics of the model's decision boundaries and classification patterns.

Let an original and a suspect model be queried using the same set of inputs. We define the similarity measure S_p as the ratio of the matching predictions to all queries:

$$S_p = \frac{1}{N}|\{i \in 1..N | p_{o,i} = p_{s,i}\}|, \tag{1}$$

where $p_{o,i}$ and $p_{s,i}$ are the predictions made by the original and the suspect model, respectively, for the input i, and N is the number of queries made.

Probability Score Vectors A probability score vector represents the model's confidence in each class for a given input. Probability score vectors are essential for understanding how the model differentiates between classes, and can reveal subtleties in the model's behavior that are not captured by output predictions alone.

We define the similarity measure S_s based on the average distance of the two sets of score vectors:

$$S_s = 1 - \frac{1}{2N}\sum_{i=1}^{N} D(v_{o,i}, v_{s,i}), \tag{2}$$

where $v_{o,i}$ and $v_{s,i}$ are the score vectors returned by the original and the suspect model, respectively, for the input i, N is the number of queries made, and D is the distance between two score vectors, defined as follows:

$$D(u, v) = \sum_{c=1}^{C} |u_c - v_c|, \tag{3}$$

where C is the number of classes and u_c is the confidence value for class c in the score vector u.

We assume that the confidence values of the score vector add up to 1, thus the image of D is in the range $[0, 2]$. Consequently, S_s takes values in range $[0, 1]$, 1 representing the highest similarity.

5.3 Physical Domain Metric

Power Traces Power traces are measurements of the power consumption of the device running the NN during inference. Power traces provide information about the model's computational workload and can be used to differentiate between models based on their physical execution characteristics.

We define the similarity measure S_t as the average intersection between the probability distributions of the two trace sets with respect to each trace point index:

$$S_t = \frac{1}{P} \sum_{j=1}^{P} O(f_{T_o^{(j)}}, f_{T_s^{(j)}}), \tag{4}$$

where T_o and T_s are $N \times P$ matrices that represent N power traces of the original and the suspect model, respectively, where each trace has P data points. $T^{(j)}$ denotes the j^{th} column of matrix T, which represents the j^{th} point of all the traces in T. f_R denotes the probability density function (PDF) of a random variable R. We estimate $f_{T_o^{(j)}}$ and $f_{T_s^{(j)}}$ according to N values that $T_o^{(j)}$ and $T_s^{(j)}$ each take. Finally, O denotes the intersection (or overlap) between two PDFs defined as follows:

$$O(f, g) = \int \min(f(x), g(x))dx \tag{5}$$

The overlap O is a symmetric and bounded measure of similarity of two PDFs. The overlap O, as well as S_t, take values in range $[0, 1]$, 1 representing the highest similarity.

5.4 Fingerprint Extraction and Comparison

Single Metric Let f_1 be the PDF of the values a similarity function S can take when comparing a possible clone to the original model that the clone is based on. Let f_2 be the PDF of values that S can take when comparing two independently trained models. f_1 and f_2 may partially overlap. Fingerprint verification starts by calculating the similarity $x = S(M_o, M_s)$ of an original model M_o to a suspect model M_s. Depending on the distribution on which x falls, a decision is made. If $f_1(x) > f_2(x)$, the suspect is identified as a clone. If $f_1(x) < f_2(x)$, the suspect is identified as an independent model.

Joint Metric In the case of the hybrid approach, each of f_1 and f_2 is a joint PDF of two similarity measures S_1 and S_2. During fingerprint verification, two similarities $x = S_1(M_o, M_s)$ and $y = S_2(M_o, M_s)$ are calculated. If $f_1(x, y) > f_2(x, y)$, the suspect is identified as a clone and vice versa.

5.5 Fingerprint Effectiveness

The effectiveness of a fingerprint is its ability to correctly identify a cloned NN as a clone, as well as a legitimate, independently trained NN as benign. We leverage two statistical measures, *sensitivity* and *specificity* [21] to quantify the effectiveness of each metric explained in section 5.2 and 5.3, individually and in combination.

Sensitivity (SEN) is the probability of a randomly selected clone to be correctly identified:

$$\text{SEN} = \int_{f_1(x)>f_2(x)} f_1(x)dx \tag{6}$$

Specificity (SPC) is the probability of a randomly selected independent model to be correctly identified:

$$\text{SPC} = \int_{f_2(x)>f_1(x)} f_2(x)dx \tag{7}$$

In the case of the hybrid approach:

$$\text{SEN} = \iint_{f_1(x,y)>f_2(x,y)} f_1(x,y)dxdy \tag{8}$$

$$\text{SPC} = \iint_{f_2(x,y)>f_1(x,y)} f_2(x,y)dxdy \tag{9}$$

Another statistical measure, *balanced accuracy*, can be used to combine sensitivity and specificity into a single measure. It is defined as the arithmetic average of the two.

6 Experimental Setup

This section presents the experimental setup used to evaluate the effectiveness of the presented fingerprinting method.

Models We test the method on a Multilayer Perceptron (MLP) architecture shown in Table 1.

Table 1: MLP architecture used in our experiments.

Layer	**Output size**	**Activation function**	**Nr. of parameters**
Flatten	49	–	0
Dense	8	ReLU	400
Dense	10	Softmax	90

The EMNIST digits dataset [22] is used for training and testing. The dataset includes 240,000 training and 40,000 testing examples. Each example contains a 28×28 grayscale image. We reduced the image resolution to 7×7 to adjust to the underlying device on which we perform the inference.

The dataset was split into 12 independent, smaller datasets. We trained two models per dataset, each with a different randomly initialized set of weights, obtaining 24 models in total. The average testing accuracy of the models is 86.89%.

Next, 72 additional models were created by adding three different levels of Gaussian noise to the weights of each one of the initial 24 models: 24 models with SNR 1000, 24 models with SNR 100, and 24 models with SNR 10. The noisy models represent models that are extracted via numerical analysis and side-channel attacks, explained in

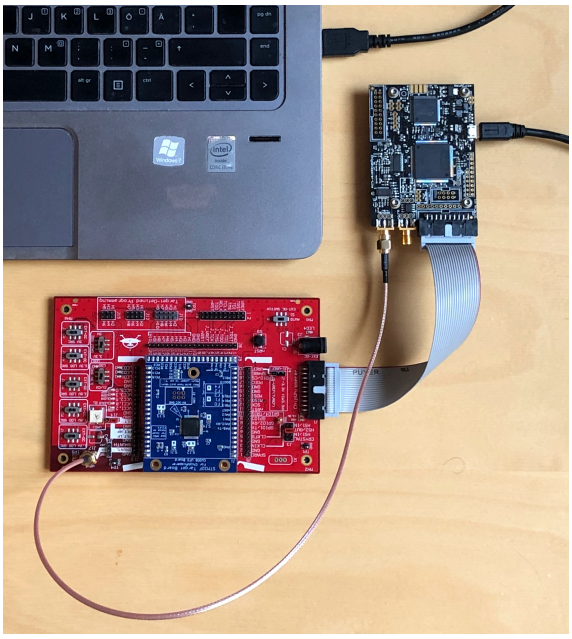

Figure 1: The target (lower left) and capture (upper right) boards used in our experiments.

section 2.2. The average testing accuracy of the noisy models are 86.9%, 86.86%, and 85.10%, respective to the above noise levels. Notice that adding a noise up to SNR 10 reduced the testing accuracy only by 1.79%. Adding a noise with SNR 1000 even slightly improved the testing accuracy.

Finally, we retrained 12 of the initial 24 models using the datasets used to train the remaining 12 models. We performed each retraining twice with different randomly initialized sets of weights, obtaining 24 clones in total. The average testing accuracy of the retrained models is 85.97%.

Data Collection We queried each model with the following sets of inputs:

1. Inputs taken from the testing dataset of the original model. The testing inputs are similar to the training inputs but have been seen by neither the original model nor any clone.

2. Inputs taken from the training dataset of the original model.

3. Randomly generated inputs, where each pixel is taken from a uniform distribution over the pixel range.

Every set includes 2,000 inputs.

We captured a power trace of 24,400 points for every input, one point per clock cycle. Along with the traces, the output predictions and score vectors are saved.

We also captured additional traces from each of the 24 initial models using each of the input sets listed above to represent model extraction via direct access.

Testbed TensorFlow 2.16.1 [23] is used for creation of the NNs. CW308T-STM32F4 [24] is used as the target device on which model inference is executed. ChipWhisperer-Lite [25] capture board is used to capture power traces (see Fig. 1). The capture board has a clock synchronized with the target device. TensorFlow Lite for microcontrollers [26] is used to execute TensorFlow models on the target device. The exact commit IDs for this framework and its dependencies are listed in the public source code repository.

7 Experimental Results

First, the results of using testing inputs for fingerprinting are presented. Subsequently, the results of using training inputs and random inputs are presented in comparison to using test inputs.

7.1 Fingerprinting Using Test Inputs

The original NN's similarity to different types of clones, calculated using prediction similarity S_p, score vector similarity S_s, and power trace similarity S_t, is listed in Table 2-a. An important observation is that, for all similarity measures, the average similarity between the original model and any type of clone is higher than the average similarity between the original and the independent model. This means that all three metrics individually can be helpful to identify all clone types that we consider.

The sensitivity and specificity of five fingerprinting methods that leverage three individual and two hybrid metrics are presented in Table 3-a.

Effectiveness of the Information Domain Metrics Our results show that each information domain metric, namely the prediction similarity S_p and score vector similarity S_s, alone is very successful, with a balanced accuracy of 100%, at identifying a clone that is either an exact copy of the original NN, or a replica with a small amount of noise in the weights (SNR$\geq$100). However, when the noise level reaches up to SNR of 10, S_p and S_s no longer perform as good, with balanced accuracies 67% and 70.01%, respectively. Finally, models that are extracted via retraining can be identified via S_p and S_s with balanced accuracies 90.25% and 92.66%, respectively.

Effectiveness of the Physical Domain Metric For a clone that is an exact copy of the original NN and a clone that has noisy weights with SNR$\geq$100, the physical domain metric, namely the power trace similarity S_t, performs almost as good as the information domain metrics, with a balanced accuracy ranging from 99.97% to 100%. For retrained clones, S_t has a balanced accuracy of 70.49% compared to 90.25% and 92.66% for the information domain metrics. For the highly noisy clones with SNR of 10, S_t is better than S_p and S_s, with a balanced accuracy of 91.3% compared to 67% and 70.01%, respectively.

Effectiveness of Hybrid Metrics For a highly noisy clone with SNR of 10, the hybrid metric (S_p, S_t) improves the balanced accuracy by 25.95% over S_p, and by 1.65% over S_t. The hybrid metric (S_s, S_t) improves the balanced accuracy by 23.15% over S_s, and by 1.86% over S_t. For the other types of clones, the hybrid approach does not give a significant improvement.

Table 2: Original NN's similarity (average ± standard deviation) to different types of clones and to independently trained models, calculated using S_p, S_s, and S_t.

(a) Testing inputs used for fingerprinting

	$\mathbf{S_p}$	$\mathbf{S_s}$	$\mathbf{S_t}$
$\mathbf{M_o}$ **(copy)**	1.0000 ± 0.0000	1.0000 ± 0.0000	0.9647 ± 0.0016
$\mathbf{M_{SNR=1000}}$	0.9994 ± 0.0005	0.9994 ± 0.0001	0.9625 ± 0.0021
$\mathbf{M_{SNR=100}}$	0.9940 ± 0.0020	0.9934 ± 0.0012	0.9613 ± 0.0021
$\mathbf{M_{SNR=10}}$	0.9384 ± 0.0173	0.9273 ± 0.0192	0.9565 ± 0.0050
$\mathbf{M_{retrained}}$	0.9627 ± 0.0095	0.9539 ± 0.0103	0.9491 ± 0.0039
$\mathbf{M_{independent}}$	0.9337 ± 0.0093	0.9199 ± 0.0103	0.9443 ± 0.0031

(b) Training inputs used for fingerprinting

	$\mathbf{S_p}$	$\mathbf{S_s}$	$\mathbf{S_t}$
$\mathbf{M_o}$ **(copy)**	1.0000 ± 0.0000	1.0000 ± 0.0000	0.9651 ± 0.0016
$\mathbf{M_{SNR=1000}}$	0.9995 ± 0.0005	0.9994 ± 0.0001	0.9613 ± 0.0030
$\mathbf{M_{SNR=100}}$	0.9948 ± 0.0020	0.9934 ± 0.0013	0.9597 ± 0.0029
$\mathbf{M_{SNR=10}}$	0.9359 ± 0.0194	0.9267 ± 0.0198	0.9557 ± 0.0044
$\mathbf{M_{retrained}}$	0.9633 ± 0.0082	0.9534 ± 0.0107	0.9486 ± 0.0035
$\mathbf{M_{independent}}$	0.9390 ± 0.0086	0.9202 ± 0.0101	0.9459 ± 0.0022

(a) Randomly generated inputs used for fingerprinting

	$\mathbf{S_p}$	$\mathbf{S_s}$	$\mathbf{S_t}$
$\mathbf{M_o}$ **(copy)**	1.0000 ± 0.0000	1.0000 ± 0.0000	0.9623 ± 0.0028
$\mathbf{M_{SNR=1000}}$	0.9985 ± 0.0012	0.9987 ± 0.0006	0.9617 ± 0.0021
$\mathbf{M_{SNR=100}}$	0.9860 ± 0.0070	0.9875 ± 0.0051	0.9599 ± 0.0021
$\mathbf{M_{SNR=10}}$	0.8653 ± 0.0635	0.8786 ± 0.0518	0.9529 ± 0.0055
$\mathbf{M_{retrained}}$	0.6594 ± 0.1123	0.6935 ± 0.0982	0.9039 ± 0.0194
$\mathbf{M_{independent}}$	0.6163 ± 0.1076	0.6415 ± 0.1011	0.9168 ± 0.0141

Table 3: Sensitivity (SEN) and specificity (SPC) of fingerprinting methods using individual and hybrid similarity metrics for different types of clones.

(a) Testing inputs used for fingerprinting

	$\mathbf{S_p}$		$\mathbf{S_s}$		$\mathbf{S_t}$		$(\mathbf{S_p}, \mathbf{S_t})$		$(\mathbf{S_s}, \mathbf{S_t})$	
	SEN	**SPC**	**SEN**	**SPC**	**SEN**	**SPC**	**SEN**	**SPC**	**SEN**	**SPC**
$\mathbf{M_o}$ **(copy)**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
$\mathbf{M_{SNR=1000}}$	1.0000	1.0000	1.0000	1.0000	0.9998	0.9997	1.0000	1.0000	0.9984	1.0000
$\mathbf{M_{SNR=100}}$	1.0000	1.0000	1.0000	1.0000	0.9997	0.9996	1.0000	1.0000	1.0000	1.0000
$\mathbf{M_{SNR=10}}$	0.5741	0.7659	0.5822	0.8180	**0.8501**	**0.9759**	**0.8831**	**0.9746**	**0.8874**	**0.9757**
$\mathbf{M_{retrained}}$	0.8789	0.9261	0.9363	0.9169	0.7082	0.7016	0.8925	0.9359	0.9330	0.9194

(b) Training inputs used for fingerprinting

	$\mathbf{S_p}$		$\mathbf{S_s}$		$\mathbf{S_t}$		$(\mathbf{S_p}, \mathbf{S_t})$		$(\mathbf{S_s}, \mathbf{S_t})$	
	SEN	**SPC**	**SEN**	**SPC**	**SEN**	**SPC**	**SEN**	**SPC**	**SEN**	**SPC**
$\mathbf{M_o}$ **(copy)**	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
$\mathbf{M_{SNR=1000}}$	1.0000	1.0000	1.0000	1.0000	0.9965	0.9971	0.9999	1.0000	0.9982	1.0000
$\mathbf{M_{SNR=100}}$	1.0000	1.0000	1.0000	1.0000	0.9714	0.9903	0.9998	1.0000	0.9998	1.0000
$\mathbf{M_{SNR=10}}$	0.4638	0.7991	0.5651	0.8301	0.8451	0.9570	0.8950	0.9647	0.8888	0.9664
$\mathbf{M_{retrained}}$	0.8583	0.9162	0.9173	0.9156	0.6107	0.7325	0.8901	0.9299	0.9130	0.9356

(c) Randomly generated inputs used for fingerprinting

	$\mathbf{S_p}$		$\mathbf{S_s}$		$\mathbf{S_t}$		$(\mathbf{S_p}, \mathbf{S_t})$		$(\mathbf{S_s}, \mathbf{S_t})$	
	SEN	**SPC**	**SEN**	**SPC**	**SEN**	**SPC**	**SEN**	**SPC**	**SEN**	**SPC**
$\mathbf{M_o}$ **(copy)**	1.0000	1.0000	1.0000	1.0000	0.9992	0.9953	1.0000	1.0000	1.0000	1.0000
$\mathbf{M_{SNR=1000}}$	1.0000	0.9993	1.0000	0.9993	0.9995	0.9956	1.0000	0.9978	0.9966	0.9980
$\mathbf{M_{SNR=100}}$	1.0000	0.9992	1.0000	0.9992	0.9992	0.9935	1.0000	0.9977	1.0000	0.9977
$\mathbf{M_{SNR=10}}$	**0.8762**	**0.9140**	**0.8989**	**0.9253**	**0.9432**	0.9492	**0.9749**	0.9548	**0.9792**	0.9607
$\mathbf{M_{retrained}}$	0.4540	0.6569	0.4624	0.6904	0.5039	**0.7754**	0.5481	0.7798	0.5465	0.7846

7.2 Fingerprinting Using Training Inputs

Equivalent results for using training inputs are shown in Tables 2-b and 3-b.

Using training inputs, compared to testing inputs, does not provide a significant improvement, and, in most cases, especially in the case prediction similarity S_p is used to detect models $M_{\text{SNR}=10}$ extracted with low accuracy, it reduces the effectiveness.

7.3 Fingerprinting Using Random Inputs

Equivalent results for using random inputs are shown in Tables 2-c and 3-c.

Using random inputs with information domain metrics S_p and S_s, compared to using testing inputs, significantly improves both the sensitivity and specificity at detecting clones $M_{\text{SNR}=10}$ extracted with low accuracy, by 30.21% and 14.81% for S_p, and by 31.67% and 10.73% for S_s. Using random inputs with the physical domain metric and the two hybrid metrics, on the other hand, only improves sensitivity at detecting $M_{\text{SNR}=10}$ by 9.31%, 9.18%, and 9.18%, respectively. Furthermore, using random inputs with the physical domain metric improves specificity at detecting retrained clones by 7.38% but worsens sensitivity by 10.68%.

8 Discussion

The experimental results confirm that combining information domain metrics with physical domain metrics provides a robust and effective solution for detecting cloned NNs.

Three types of inputs have been evaluated for fingerprinting. Using testing inputs that neither the original model nor the suspect model had seen gave the most consistent results overall. Using the training inputs of the original model did not improve effectiveness, accordingly, we do not recommend its usage in the presented fingerprinting method. Using randomly generated inputs is a viable alternative as it improves effectiveness at detecting clones extracted with low accuracy but performs worse for the retrained clones. We recommend its usage when it is improbable that a given suspect model is cloned via retraining, and when false positive detection is less harmful. Random inputs can also be used when a testing dataset is not available.

8.1 Possible Improvements

The presented method shows a promising potential in identifying cloned NNs. However, several directions for improvements remain, including:

Scalability While the presented method shows high effectiveness, its scalability in large-scale deployments should be further explored. Accordingly, future work should consider optimizing the data collection and fingerprint extraction processes to handle larger models more efficiently.

Generalization across architectures Our experiments involved MLPs. Extending the presented approach to other types of NN architectures, such as transformers or graph neural networks, could be valuable. Assessing how well the fingerprinting method generalizes across various architectures would help ensure its applicability in diverse contexts.

Resistance to fingerprint removal Adversaries may employ fingerprinting removal techniques, such as those presented in [8,12,19], to evade fingerprint detection. Investigating how different types of attacks could potentially circumvent the presented fingerprinting approach and developing countermeasures are important.

9 Conclusion

In this paper, we presented a novel method for identifying cloned NNs. The key contribution is a hybrid fingerprinting technique that combines metrics from both the information and physical domains. Our experiments show that, while each metric individually provides useful information, the combination of the metrics improves the accuracy of model identification, especially in the case of a clone that is extracted with low accuracy.

The method is particularly useful in scenarios where NNs are at risk of model extraction attacks, such as in cloud-based machine learning services. It offers a practical solution for verifying model ownership and distinguishing between original and cloned models, contributing to the ongoing efforts in securing NNs against model cloning.

Acknowledgement

This work was supported in part by the Sweden's Innovation Agency Vinnova (Grant No. 2023-00221) and the Swedish Civil Contingencies Agency (Grant No. 2020-11632).

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 601–618.

[3] C. Aknesil, E. Dubrova, N. Lindskog, J. Sternby, and H. Englund, "Hybrid fingerprinting for effective detection of cloned neural networks," 2025, accepted to the 2025 IEEE 55th International Symposium on Multiple-Valued Logic (ISMVL).

[4] K. Li, "AI model security: Reverse engineering machine learning models," Talk in HITB2018DXB, 2018. [Online]. Available: https://conference.hitb.org/hitbsecconf2018dxb/materials/D1T1%20-%20AI%20Model%20Security%20-%20Reverse%20Engineering%20Machine%20Learning%20Models%20-%20Kang%20Li.pdf

[5] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 1345–1362.

[6] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," *2018 27th USENIX Security Symposium (USENIX Security 18)*, pp. 515–532, 2018.

[7] J. Guo and C. Wang, "DeepFingerprint: Undermining ownership of deep neural networks through fingerprint extractions," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018, pp. 264–279.

[8] H. Yao, Z. Li, K. Huang, J. Lou, Z. Qin, and K. Ren, "RemovalNet: DNN fingerprint removal attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 4, pp. 2645–2658, 2024.

[9] A. Waheed, V. Duddu, and N. Asokan, "GrOVe: Ownership verification of graph neural networks using embeddings," 2023. [Online]. Available: https://arxiv.org/abs/2304.08566

[10] P. Maini, M. Yaghini, and N. Papernot, "Dataset inference: Ownership resolution in machine learning," 2021. [Online]. Available: https://arxiv.org/abs/2104.10706

[11] S. Szyller, R. Zhang, J. Liu, and N. Asokan, "On the robustness of dataset inference," 2023. [Online]. Available: https://arxiv.org/abs/2210.13631

[12] N. Lukas, Y. Zhang, and F. Kerschbaum, "Deep neural network fingerprinting by conferrable adversarial examples," *CoRR*, vol. abs/1912.00888, 2019.

[13] Y. Li, Z. Zhang, B. Liu, Z. Yang, and Y. Liu, "ModelDiff: testing-based DNN similarity comparison for model reuse detection," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 139–151.

[14] J. Guan, J. Liang, and R. He, "Are you stealing my model? Sample correlation for fingerprinting deep neural networks," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 36 571–36 584.

[15] J. Chen, J. Wang, T. Peng, Y. Sun, P. Cheng, S. Ji, X. Ma, B. Li, and D. Song, "Copy, right? A testing framework for copyright protection of deep learning models," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 824–841.

[16] J. Song, Z. Xu, S. Wu, G. Chen, and M. Song, "ModelGiF: Gradient fields for model functional distance," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 6125–6135.

[17] V. Meyers, M. Hefenbrock, D. Gnad, and M. Tahoori, "Remote identification of neural network FPGA accelerators by power fingerprints," in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*, 2023, pp. 259–264.

[18] Y. Liu, K. Li, Z. Liu, B. Wen, K. Xu, W. Wang, W. Zhao, and Q. Li, "Provenance of training without training data: Towards privacy-preserving DNN model ownership verification," in *Proceedings of the ACM Web Conference 2023*, ser. WWW '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1980–1990.

[19] M. Wang, H. Qiu, T. Zhang, M. Qiu, and B. Thuraisingham, "Mitigating query-based neural network fingerprinting via data augmentation," *ACM Trans. Sen. Netw.*, May 2023, just accepted.

[20] S. Moini, S. Tian, D. Holcomb, J. Szefer, and R. Tessier, "Power side-channel attacks on BNN accelerators in remote FPGAs," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 2, pp. 357–370, 2021.

[21] D. G. Altman and J. M. Bland, "Statistics notes: Diagnostic tests 1: sensitivity and specificity," *BMJ*, vol. 308, no. 6943, p. 1552, 1994. [Online]. Available: https://www.bmj.com/content/308/6943/1552

[22] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of MNIST to handwritten letters," 2017. [Online]. Available: https://arxiv.org/abs/1702.05373

[23] Accessed: Sep. 30, 2024. [Online]. Available: https://www.tensorflow.org/

[24] Accessed: Sep. 30, 2024. [Online]. Available: https://rtfm.newae.com/Targets/UFO%20Targets/CW308T-STM32F/

[25] Accessed: Sep. 30, 2024. [Online]. Available: https://rtfm.newae.com/Capture/ChipWhisperer-Lite/

[26] Accessed: Sep. 30, 2024. [Online]. Available: https://github.com/tensorflow/tflite-micro