Queries, Representation & Detection: The Next 100 Model Fingerprinting Schemes

Augustin Godinot[1, 2, 3, 5], **Erwan Le Merrer**[2], **Camilla Penzo**[5], **François Taïani**[1, 2, 3], **Gilles Trédan**[4]

[1]Université de Rennes, France,
[2]Inria, Rennes, France,
[3]IRISA/CNRS, Rennes, France,
[4]LAAS/CNRS, Toulouse, France,
[5]PEReN, Paris, France
augustin.godinot@inria.fr

Abstract

The deployment of machine learning models in operational contexts represents a significant investment for any organisation. Consequently, the risk of these models being misappropriated by competitors needs to be addressed. In recent years, numerous proposals have been put forth to detect instances of model stealing. However, these proposals operate under implicit and disparate data and model access assumptions; as a consequence, it remains unclear how they can be effectively compared to one another. Our evaluation shows that a simple baseline that we introduce performs on par with existing state-of-the-art fingerprints, which, on the other hand, are much more complex. To uncover the reasons behind this intriguing result, this paper introduces a systematic approach to both the creation of model fingerprinting schemes and their evaluation benchmarks. By dividing model fingerprinting into three core components – Query, Representation and Detection (QuRD) – we are able to identify ~ 100 previously unexplored QuRD combinations and gain insights into their performance. Finally, we introduce a set of metrics to compare and guide the creation of more representative model stealing detection benchmarks. Our approach reveals the need for more challenging benchmarks and a sound comparison with baselines. To foster the creation of new fingerprinting schemes and benchmarks, we open-source our fingerprinting toolbox.

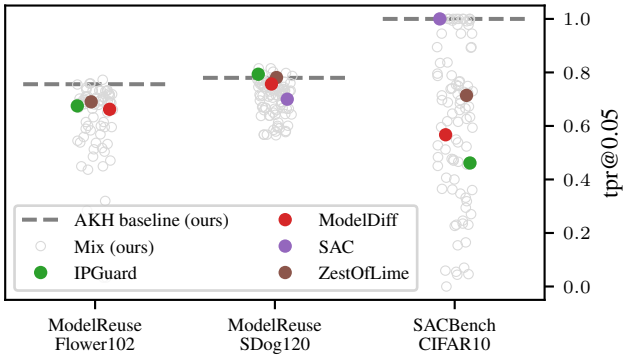

Figure 1: The TPR@5% of most of the fingerprinting schemes proposed in the literature is at best as good as the simple baseline we introduce. Each colored dot represents the performance of an existing fingerprinting scheme evaluated on a given benchmark. The gray dots are fingerprinting schemes we created using our Query, Representation and Detection (QuRD) decomposition.

Introduction

Companies devote considerable resources (i.e. manpower, funds and energy) to developing efficient and accurate machine learning (ML) models. Many of these models are then deployed in production on online platforms to solve a wide array of business-critical tasks (e.g. recommendations or predictions of all kinds). However, it is well understood that extraction attacks, or simply infrastructure leaks, can allow competitors to access the model architecture (Oh et al. 2018), weights (Carlini et al. 2024), and hyperparameters (Wang and Gong 2018). From financial risks, when the attacker can provide the same functionality at a fraction of the cost, to integrity risks, when the attacker could use the stolen model as a step to craft adversarial examples, *Model stealing attacks* pose great risks for the model developer. Although efforts have been devoted to defend models against extraction attacks (Tang et al. 2024; Orekondy, Schiele, and Fritz 2019; Lee et al. 2019), extraction defences have not yet been proven secure. Therefore, in addition to *preventing* model stealing, companies need tools to *detect* it. One such tool is *model fingerprinting*. Similarly to how fingerprints can analyse the provenance of an image by identifying artefacts due to the compression scheme, the specific sensor technology, or even the up-scaling method (Ojha, Li, and Lee 2023), model fingerprints analyse the outputs of a ML model h to extract artefacts that are characteristic of h itself. In order to build a fingerprint for a given model h, the model owner first carefully selects a set of inputs S. The model owner then extracts a unique representation Z_h from the output of their model h when given S as input. The representation Z_h will serve as a fingerprint. The fingerprint Z_h can later be compared with the fingerprint $Z_{h'}$ extracted from the model h', which is suspected to be stolen. The fingerprint scheme depends on the input modality (e.g. text, image, or tabular data), on the model's task (e.g. classification, score, or recommendations), and hence on the domain of the output of h. In this work, as in most of the model finger-

printing literature, we focus on image classification models. Note that, contrary to model watermarking methods, fingerprinting does not provide any theoretical guarantees on the false alarm rate (e.g. false positives). Thus, a strong empirical evaluation of model fingerprinting schemes is paramount to ensure their soundness in practice.

Problem This paper presents a surprising artefact of fingerprinting evaluation. Fingerprinting evaluation consists in generating *positive* and *negative* model pairs (h, h'), where positive model pairs consist in a victim model h and a model h' stolen from h (e.g. through model extraction), while for negative model pairs, h and h' are totally unrelated (e.g. trained on a different dataset). A collection of such positive and negative pairs is called *benchmark*. Figure 1 displays the True Positive Rate (TPR@5%) of existing fingerprints on two existing benchmarks, ModelReuse (Li et al. 2021) and SACBench (Guan, Liang, and He 2022). Figure 1 demonstrates that *the simple baseline that we introduce (gray dashed lines) performs on par with existing state-of-the-art fingerprinting schemes (coloured dots), which are much more complex.*

In the following, we seek to understand the reasons behind this result by exploring the two key aspects of Figure 1: How do existing fingerprints and benchmarks compare. Our contributions will be the following.

1. We introduce a simple yet powerful baseline and provide theoretical guarantees on its performance. Albeit on a simple model copy detection task, this constitutes the first theoretical analysis of the guarantees of a model fingerprinting scheme.
2. We survey and compare existing fingerprinting schemes for classification tasks. Our novel queries-representation-calibration decomposition (hereafter we coin QuRD) enables us to systematise and thus uncover new and unexplored fingerprinting schemes. The novelty of QuRD lies in its mix of geometrical (distance between fingerprints leads to distance between models) and statistical insights (the fingerprint is then used to perform a statistical property test).
3. We compare existing benchmarks and investigate their differences in both the way the pair of test models (h, h') are generated and the distinguishability of the victim h and suspected h' models. Our work constitutes the first systematic comparison of classifier fingerprinting benchmarks, and reveals insights into how to build more informative and challenging benchmarks. All the code required to re-run our experiments, implement new benchmarks and evaluate new fingerprints is available online at https://github.com/grodino/QuRD.

Background and Setting

Stealing ML models The possibilities for an adversary to steal a given model are endless. They could break into the infrastructure of their victim (Ben-Sasson and Tzadik 2024), perform black-box model extraction attacks (Jagielski et al. 2020; Truong et al. 2021) or just use the output of the victim's model to train their own. In this work, we consider adversaries seeking to steal the functionality of the victim's model.

Detecting IP violation via model fingerprinting The dominant approach to model fingerprinting is based on comparing the outputs of models on adversarial queries, as in AFA (Zhao et al. 2020), TAFA (Pan et al. 2021), IPGuard (Cao, Jia, and Gong 2021), ModelDiff (Li et al. 2021), FUAP (Peng et al. 2022), FCAE (Lukas, Zhang, and Kerschbaum 2020), DeepFoolF (Wang and Chang 2021), and DeepJudge (Chen et al. 2022). Other approaches leverage the sensitivity of ML models at random points sampled from the train set (e.g. SSF (He, Zhang, and Lee 2019), ModelGif (Song et al. 2023)), some explanations generated from the victim model h ZestOfLIME (Jia et al. 2022) or even train classifiers to distinguish stolen from benign model MetaV (Pan et al. 2022). Some other works explore the use of natural images (images in the training/validation set) to craft their query set S, as in FBI (Maho, Furon, and Le Merrer 2023) or SAC (Guan, Liang, and He 2022). All of these works try to detect model stealing, however comparison among them and the assumptions they make are rarely taken into consideration. In this work, we introduce a framework to compare and evaluate these fingerprints.

Problem setting Consider an input space $\mathcal{X}$, a space of labels $\mathcal{Y} = \{1, \dots, C\}$ with C classes, a data distribution $\mathcal{D}$ on $\mathcal{X}$ and a ground truth concept $c \in \{1, \dots, C\}^{\mathcal{X}}$. A first party called the *victim* trains a model h on a classification task $\mathcal{C}$, then deploys this model in production. A second party called the *adversary* wishes to recreate a model h' that is close to identical to h ($h' \approx h$) to deploy it at a low cost.

The task of checking whether a *suspected model* h' is a copy of the *victim model* h is modeled as a property test (Goldreich 2017). A tester $\mathcal{T}$ is a (randomized) algorithm that takes two models h and h' as input and returns 1 with high probability if h' is stolen from h, 0 else.

$$\begin{array}{ll} \text{if } h = h',\ \mathbb{P}\left(\mathcal{T}(h, h') = 1\right) > \frac{2}{3} & \textit{Copied model !} \\ \text{if } h \neq h',\ \mathbb{P}\left(\mathcal{T}(h, h') = 0\right) > \frac{2}{3} & \textit{Just an other model} \end{array}$$

The fingerprint (a.k.a. the property test) should be *effective*, *robust* and *unique*. We also require the fingerprint to be *efficient* in terms of queries and samples.

1. *Effectiveness*: if $h' = h$, then the suspected model is flagged by the victim with high probability.
2. *Robustness*: if h' is a slightly modified version of h (via fine-tuning, pruning, model extraction ...), then the suspected model should still be flagged.
3. *Uniqueness*: Original models $h' \neq h$ are not flagged.
4. *Efficiency*: the test uses few queries to the suspected model h' and few samples x from the data distribution.

Accessibility of data and models The type of fingerprinting scheme that can be used by the victim depends on the access the victim has to the suspected model h'. We will assume that the victim can freely query the suspected model h'. Yet, the output of the suspected model will range from label-only query access, to top-K labels query access, probits or logits query access and even to gradients query access. Following the fingerprinting literature, it is assumed

that the victim has full access to its training data and model h.

Filling the Gaps with the AKH Baseline

The first contribution of this paper is the proposal and analysis of a simple yet powerful baseline, which, as we observed in Figure 1 performs at least as well as State-Of-the-Art fingerprinting schemes.

It is assumed that the victim has access to samples from the input distribution, for example the test set they used to validate their model. The baseline refers to Tolstoy's Anna Karenina principle that states "All happy families are alike; each unhappy family is unhappy in its own way". Thus, instead of using random samples for the input space $\mathcal{X}$, we look for points that are mis-classified by h and compare the victim and suspected models on those points. Our baseline, coined the *Anna Karenina Heuristic* (AKH), proceeds as follows. First, the victim chooses a negative input: a point $x \sim \mathcal{D}$ such that h wrongly classifies x: $h(x) \neq c(x)$. We write $\overline{\mathcal{D}_h}$ the resulting negative inputs distribution. Then, the victim queries the suspected model h' on x. Finally, if $h'(x) = h(x)$ the suspected model h' is flagged as stolen, otherwise h' is deemed benign.

Proposition 1. *Consider $h, h' \in \mathcal{Y}^{\mathcal{X}}$ two models and $\alpha = \mathbb{P}(h(x) = c(x))$ (resp. $\alpha' = \mathbb{P}(h'(x) = c(x))$) their accuracy. Let $\delta = d_H(h, h')$ be the relative Hamming distance between h and h' and $\delta_C = \mathbb{P}(h(x) \neq h'(x) \,|\, h(x) \neq c(x))$. The property test $\mathcal{T}_b$ defined by AKH enjoys the following guarantees:*

$$\text{If } h = h',\ \mathbb{P}_{\mathcal{D}}(\mathcal{T}_b(h, h') = 1) = 1 \tag{1}$$

$$\text{If } h \neq h',\ \mathbb{P}_{\mathcal{D}}(\mathcal{T}_b(h, h') = 0) = \delta_C \geq \frac{\delta - (1 - \alpha')}{1 - \alpha} \tag{2}$$

The proof of Proposition 1 and the detailed algorithm can be found in the technical appendix. Proposition 1 establishes that AKH is a one-sided error test. Thus, in the favorable scenario where h' is copied (i.e. not tampered with), $\mathcal{T}_b$ will always detect it. To simplify the analysis, we defined AKH using only one query to the suspected model. To further decrease the False Negative Rate, one should run the baseline multiple times. A majority vote among the values returned by $\mathcal{T}_b$ decreases the False Negative Rate exponentially (Goldreich 2017). If instead of selecting negative examples (points $x \in \mathcal{X}$ that are wrongly classified by h), the victim was to use random samples according to $\mathcal{D}$, the test would still have a one-sided error but the True Negative Rate $\mathbb{P}_{\mathcal{D}}(\mathcal{T}(h, h') = 0)$ would be equal to the hamming distance δ between h and h'. This gives us an idea on when AKH can outperform schemes based on random sampling: either when the error rate $1 - \alpha$ of the victim model h is low or when the error rate $1 - \alpha'$ of the suspected classifier h' is low compared to $1 - \alpha$.

In practice, the TPR@5% of AKH is displayed in Figure 1 in gray dashed lines. On ModelReuse (SDog120 dataset) and on SACBench, AKH performs on par with the best existing fingerprints. On ModelReuse (Flower102 dataset), AKH even performs better than the best existing fingerprints. In the two following sections we explore the reasons behind this observation by looking at the two players of Figure 1: the fingerprints and the benchmarks used to compare them.

Query, Representation & Detection: the QuRD Framework

The literature on model fingerprinting does not provide a unified definition of model stealing detection. Most works focus on particular transformations of the stolen model, which they seek to detect. Only a few works (Cao, Jia, and Gong 2021; Maho, Furon, and Le Merrer 2023; Peng et al. 2022) are based on a mathematical formulation of the problem. Some fingerprinting schemes (e.g. ZestOfLIME or ModelGif) are described from a geometrical point of view: the goal is to create a distance between models to distinguish stolen models from unrelated models. On the other hand, some works are described from a statistical point of view: the goal is to test whether $h' = h$ or not. Thus, comparing and categorizing existing fingerprints is not trivial. As a second contribution to this paper, we propose an original decomposition of the existing (and future) fingerprinting schemes into three core components:

1. **Query Sampling**, which generates the query set $S \subset \mathcal{X}$ on which to query h and h', e.g. selecting a subset of the victim model training set h.
2. **Representation**, which computes a compact representation $Z_h{=}g(Y_h)$ and $Z_{h'}{=}g(Y_{h'})$ of the answers $Y_h{=}\{h(x) : x \in S\}$ and $Y_{h'}{=}\{h'(x) : x \in S\}$ that are returned by the two models h and h' on the sample S. A basic strategy is to use the raw answers as a representation, that is, $Z_h{=}Y_h, Z_{h'}{=}Y_{h'}$.
3. **Detection**, which uses the two fingerprints Z_h and $Z_{h'}$, and possibly a set of calibration fingerprints $\{Z_i\}_i$, to decide whether h' is a stolen version of h or not.

Query Sampling (Q)

Existing approaches use four main techniques to build the query set when generating fingerprints, *Uniform sampling*, *Adversarial sampling*, *Negative sampling*, and *Subsampling* (see Table 1). Query Sampling (Q) methods are based on the transformation of a seed query set S_{seed}, which is either the training set or the test set used by the victim when generating h (both assumed to follow the same data distribution $\mathcal{D}$), or images composed of random pixel values.

Uniform sampling The easiest way to generate S is to sample uniformly from the data distribution or from a seed set $S_{\text{seed}} \subset \mathcal{X}$.

$$S \sim \mathcal{D} \text{ or } S \sim \mathcal{U}(S_{\text{seed}}) \tag{3}$$

Adversarial sampling Adversarial sampling exploits the intuition that models tend to be characterized by their decision-boundary (Le Merrer, Pérez, and Trédan 2020; Cao, Jia, and Gong 2021; Li et al. 2021). Compared to uniform sampling, adversarial sampling leads to a better detection rate for a lower query budget s. Starting from a set of seed inputs $S_{\text{seed}} \subset \mathcal{X}$, adversarial sampling computes a set of

		Uniform	Adversarial	Negative	Subsampling	Joint detector training
Seed set S_{seed}	input space	$\emptyset$	IPGuard	$\emptyset$	$\emptyset$	MetaV
	test set	$\emptyset$	DeepJudge, FCAE	FBI	$\emptyset$	$\emptyset$
	train set	ModelGif	*ModelDiff*, FUAP, IPGuard, AFA, ModelGif, DeepJudge, SSF[1]	**SAC**	**ZestOfLIME**, **SAC**	FUAP

Table 1: Type of seed set S_{seed} (rows), Query Sampling (Q) (columns), model access (emphasis) and Representation (R) (decorations) used. Adversarial sampling dominates the fingerprinting literature. Fingerprinting scheme appearing in multiple cells either require or can accomodate both Sampling/seed types. The text decoration stands for the access required to the remote suspected model h': no decoration = label access, underline = probits access, dashed underline = label or probit access, wavy underline = gradients access. The text emphasis indicate the type of Representation: no emphasis = raw model outputs, *italicized* = pairwise representation, **bold** = listwise representation. [1]SSF actually uses *sensitive samples* instead of adversarial samples.

samples S_{adv}, targeted or not, using the following optimization procedure.

$$S_{\text{adv}} = \left\{ \underset{u, \|x-u\|<\epsilon}{\arg\max}\ d\big(h(x), h(u)\big), x \in S_{\text{seed}} \right\} \quad (4)$$

Common methods used for solving Equation (4) include Projected Gradient Descent (Madry et al. 2018) or DeepFool (Moosavi-Dezfooli, Fawzi, and Frossard 2016). Finally, the final query set is the concatenation of the seed and adversarial samples $S = (S_{\text{seed}}, S_{\text{adv}})$.

Negative sampling As for adversarial sampling, negative sampling (Guan, Liang, and He 2022) enjoys better detection rates for a given query budget. However, it does not need to compute gradients of h, it just needs query access to h, which can dramatically speed up the generation of the query set S. The core intuition follows that if h' makes the same mistakes as h, there is a high probability that the adversary stole h.

$$S \subset S_{\text{seed}} \text{ subject to } \forall x \in S, h(x) \neq c(x) \quad (5)$$

Subsampling Subsampling exploits domain knowledge to create new samples $V(x) = \{x_j\}_j$ in the vicinity of a seed point x. Compared to negative and adversarial sampling, subsampling allows to create a large query-set with few samples from the data distribution.

$$S = (S_{\text{seed}}, \{V(x)\}_{x \in S_{\text{seed}}}). \quad (6)$$

Jia et al. uses the super-pixel sampling technique of LIME (Ribeiro, Singh, and Guestrin 2016) to generate images around each image in a seed set S_{seed}.

Representation (R)

Once the model h and h' have been queried on a sample of data points, the resulting outputs Y_h and $Y_{h'}$ must be recorded using some representation. We have identified three strategies in the literature: *Raw Labels/Logits*, *Pairwise correlation*, and *Listwise correlation*.

Raw labels/logits The simplest representation of the set of answers collected from the two models would be the set of answers themselves (labels or logits). However, depending on the way h' was constructed (or not) from h, different representations are more suitable.

$$Z_h = Y_h \in (\mathbb{R}^C)^s \text{ (logits) or } \{1, \dots, C\}^s \text{ (labels)} \quad (7)$$

Pairwise correlation When the audit set S consists of pairs of samples (x, u) that have a specific meaning (e.g. u is an adversarial version of x as in ModelDiff), it is interesting to use these pairwise comparisons as the representation of the model.

$$Z_h = (d(h(x), h(u)))_{(x,u) \in S} \in \mathbb{R}^{\frac{s}{2}} \quad (8)$$

Listwise correlation Generalizing the idea of pairwise correlation, if the audit samples are not specifically paired but comparison is still meaningful, the victim can compute the similarity between all pairs of answers and use the resulting similarity matrix as representation. This is what is used by SAC.

$$Z_h = (d(h(x), h(u)))_{x \in S, u \in S} \in \mathbb{R}^{s \times s} \quad (9)$$

Detection (D)

Finally, once the victim has generated the fingerprints of their model and that of the suspected model (Z_h and $Z_{h'}$), the last step is to compare Z_h and $Z_{h'}$ to decide whether to flag h' or not.

There exists two approaches to Detection (D): directly compute a distance (e.g. hamming as in AFA or mutual information as in FBI) between the generated fingerprints or learn a classifier that takes the two fingerprints and outputs a theft probability score as in MetaV. In both cases, the victim needs access to its own pool of fingerprints from unrelated models $\mathcal{G} = \{G_1, \dots, G_{|\mathcal{G}|}\}$, to calibrate the detection threshold.

The next 100 fingerprints

In this subsection, we highlight the benefits of our novel QuRD decomposition for creating new and improved fingerprinting schemes and compare the existing fingerprints on a previously underexplored axis: the query budget.

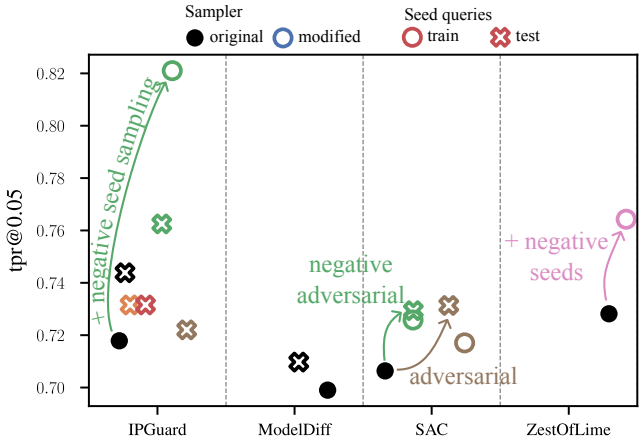

Figure 2: TPR@5% gains on ModelReuse obtained by modifying the sampler of existing fingerprints. The sampler can be modified in two ways: drawing seed queries from the train vs test set (materialized as circles vs crosses) or using a different queries sampler (materialized as a different color). Selecting negative seed inputs for adversarial generation instead of the original seeds can lead to improvements on the order of 10 points (+14%).

Fingerprint evaluation The *Effectiveness*, *Robustness* and *Uniqueness* of fingerprints are evaluated by computing the True Positive Rate at 5% of False Positives (TPR@5%). The final Detection (D) step consists in thresholding a distance or the output of a classifier based on the fingerprints Z_h and $Z_{h'}$. The True Positive Rate is the proportion of positive pairs (h, h') that are flagged as positive by the fingerprint. The False Positive Rate (FPR), which is the proportion of negative pairs (h, h') that are flagged as positive by the fingerprint. The TPR@5% captures the cost to the victim of missing a stolen model while recognizing to the cost of wrongly flagging a model as stolen. All TPR@5% values are averaged over 5 runs with independent random seeds.

Creating new fingerprints using the QuRD framework Following our QuRD framework, Table 1 categorizes exiting fingerprints (listed previously in Background and Setting). Table 1 shows that a large part of the literature focused on fingerprints based on adversarial sampling. Several QuRD combinations have not been explored yet by the literature. Moreover, the schemes always focus on using only one type of Query Sampling (Q) but very rarely explore chaining or mixing, e.g.using negative samples as the seeds for generating adversarial examples. Thus, to explore the space of QuRD combinations, we reimplemented the Query Sampler, Representation, and Detection of four existing fingerprints: ModelDiff, SAC, IPGuard and ZestOfLIME. We mixed them to create ∼100 new fingerprints. In Figure 1, gray-edged dots represent such QuRD combinations. Of course, not all new combinations are worth considering, as many QuRD combinations exhibit lower TPR@5% than existing fingerprints. Thus, in Figure 2 we show the potential improvements that can be reached by modifying the Query Sampler (Q) and/or the seed set S_{seed} of existing schemes on

		ModelReuse Flower102	ModelReuse SDog120	SACBench CIFAR10
model leak	same	✓	✓	✓
	quantize	✓	✓	×
	finetune	×	×	✓
	transfer	×	×	✓
	prune	✓	✓	✓
Model extraction	probits	✓	✓	✓
	label	✓	✓	✓
	adversarial (labels)	×	×	✓

Table 2: Stealing and obfuscation methods implemented by different benchmarks.

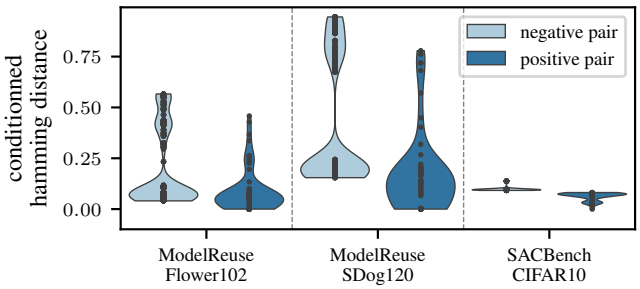

Figure 3: Distribution of the conditioned Hamming distance $d_C(h, h')$ between the models of each positive/negative (h, h') pair.

ModelReuse. Figure 2 shows that it is possible to increase the TPR@5% of IPGuard by 10 points (+14%) simply by choosing negative seed samples as the starting points for the generation of adversarial examples.

Comparing apples to apples: a focus on the query budget Although not displayed in Table 1, the query budget required by the existing fingerprints can vary greatly. For example, ZestOfLIME requires from 1000 to 128 000 queries while FBI only requires ∼ 100 queries to reach the advertised performance. In Figure 4 we show the TPR@5% of existing fingerprints along our AKH baseline and selected QuRD variations. Keeping a small query budget is of paramount importance, mainly to remain stealthy against potential defenses (Oliynyk, Mayer, and Rauber 2023), but also to avoid disrupting the remote service with (tens to hundreds of) thousands of queries. Once more, we observe that fingerprints based on negative sampling equal or outperform fingerprints based on adversarial sampling. From 0 to 100 queries for SACBench and 0 to 50 for ModelReuse, most fingerprints exhibit notable improvements at each query budget increment. After 100 (or 50) queries, most fingerprints show a plateau. Thus, it appears that there exists an optimal query budget, dependent on the benchmark but not on the fingerprinting scheme. Finally, schemes based on negative sampling appear to suffer a lower variance than adversarial-based fingerprints, especially on SACBench.

Although the performance of most fingerprints plateau

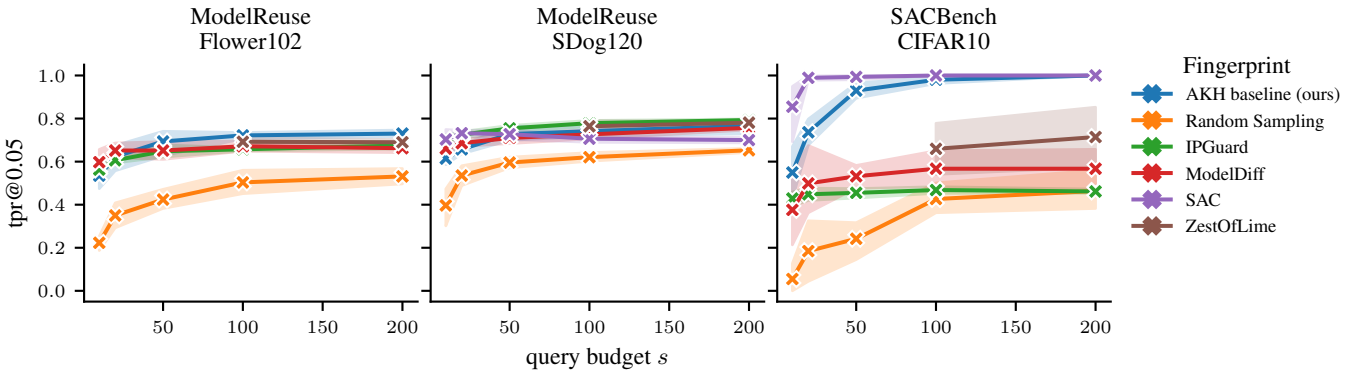

Figure 4: The effect of the query budget s on the *Efficiency* and *Robustness* of existing fingerprints, as measured by TPR@5%.

after 50-100 queries, the performance of some fingerprints (e.g. ModelDiff and SAC) suffers when the query budget increases from 100 to 400 queries. This phenomenon is observable only for schemes whose representations are based on a pairwise or a listwise comparison. We believe that when the number of query points is increased, the self-correlation increases regardless of the fact that a pair is positive or negative. Thus, the gap between the positive pair distance and the negative pair distance decreases with budget, which in turn decreases the performance of the fingerprint.

Fingerprinting Benchmarks

Because there are no strong guarantees regarding *Effectiveness* and *Robustness* of fingerprinting schemes, proper empirical evaluation is critical to assessing their performance. The main difficulty of evaluation lies in the definition (and implementation) of realistic *positive* ($h'{=}h$) and *negative* ($h'{\neq}h$) model pairs. To do this, we need to separate how the adversary steals the model (how to achieve $h'{=}h$) and how the adversary tries to conceal their theft by modifying the stolen model to avoid detection by the victim).

Stealing a model **1) Model leak**: the adversary directly steals the architecture and weights of the model h and uses them to solve the same task. This can happen via an internal leak (Franzen 2024) or an attack on the company infrastructure (Ben-Sasson and Tzadik 2024). **2) (Adversarial) model extraction** The adversary only has query access to the source model and trains their model based on the probits or the labels of the source model. The model extraction can either be probits or labels-based (Jagielski et al. 2020; Truong et al. 2021). In addition, depending on the threat model, the architecture trained by the attacker is not always the same as the victim model h and the adversary might not have access to samples from the input domain (Truong et al. 2021).

Stolen model obfuscation Once an attacker has stolen the model h, they will try obfuscating their model to hide their theft. To avoid detection by model fingerprinting, the adversary may act on a combination of three aspects of the model inference process. **1) Model/weights tampering** As first approach, the adversary can directly modify the model itself to remove potential watermarks embedded in the weights of the model: weights pruning (Liu, Dolan-Gavitt, and Garg 2018; Li et al. 2017), model quantization and finetuning or transferring the model to a small private dataset (Li et al. 2021). **2) Input modifications** The second concealment trick is to apply transformations to the inputs fed to the model to limit the effect of adversarial inputs (Maho, Furon, and Le Merrer 2023): JPEG compression, equalization, or posterization. **3) Output noise**: Finally, to avoid giving away too much information, the adversary can try to slightly alter the outputs of the model, e.g. returning only the Top-K labels, averaging the outputs over a neighbourhood of the input (Cohen, Rosenfeld, and Kolter 2019) or implementing model-stealing defences (Tang et al. 2024; Orekondy, Schiele, and Fritz 2019).

The majority of benchmarked tasks are solved

The performance shown previously in Figures 1, 3 and 4 were all aggregated at a benchmark level. In this section, we separate the performance of the fingerprints with respect to the model-stealing and obfuscation methods. We will seek to answer the question *What type of stealing and obfuscation methods can be considered as resolved issues and, hence, on which ones should practitioners focus?* Positive pairs are grouped by task, i.e., how the copied model h' was created from h, along with their corresponding negative pairs. Each task corresponds to the combination of a stealing and an obfuscation method. This decomposition is especially interesting since, as we will observe, a large portion of the tasks are solved by all the fingerprints, while the rest, and more complicated tasks, allows to discriminate the different fingerprints much more clearly.

As for benchmark-aggregated performance discussed in the QuRD Section, Table 3 shows that AKH is on par or surpasses all the previously introduced schemes. More interestingly, Table 3 reveals that a large part of the tasks considered by ModelReuse and SACBench (namely the same, quantization, finetuning, and transfer tasks) are completely solved by existing fingerprints, as well as by AKH. The remaining unsolved tasks consist of model stealing by model extraction, using no obfuscation attempts. Surprisingly, adversarial label extraction is easily detected by fingerprints

Fingerprint	Model leak					Probit extraction	Label extraction	
	same	quantize	finetune	transfer	prune	vanilla	vanilla	adversarial
IPGuard	**1.0 ± 0**	**1.0 ± 0**	**1.0 ± 0**	**1.0 ± 0**	**0.94 ± .01**	0.64 ± .02	0.12 ± 0	0.02 ± .01
ModelDiff	**1.0 ± 0**	**1.0 ± 0**	**1.0 ± 0**	**1.0 ± 0**	**0.94 ± .01**	0.59 ± .05	0.14 ± .02	0.16 ± .07
Random Sampling	**1.0 ± 0**	0.93 ± .03	**1.0 ± 0**	0.48 ± .2	0.71 ± .02	0.46 ± .01	0.07 ± .02	0.06 ± .05
SAC	**1.0 ± 0**	**1.0 ± 0**	**1.0 ± 0**	**1.0 ± 0**	0.92 ± 0	**0.81 ± 0**	**0.59 ± .02**	**1.0 ± 0**
ZestOfLime	**1.0 ± 0**	**1.0 ± 0**	**1.0 ± 0**	0.78 ± .17	0.86 ± 0	0.74 ± .02	0.38 ± .05	0.29 ± .11
AKH (ours)	**1.0 ± 0**	**1.0 ± 0**	**1.0 ± 0**	**1.0 ± 0**	0.91 ± .01	0.78 ± .01	0.46 ± .01	0.92 ± .03

Table 3: TPR@0.05 of the existing fingerprints with a budget of 100 queries. For each task, the best performance are highlighted.

based on negative sampling but not by adversarial, random, or subsampling-based fingerprints. Model extraction detection is, thus, a hard subtask of model stealing detection.

The results of Table 3 highlight an issue with the current benchmarks: trying to detect if a suspected model h' is the same as the victim's h up to small model perturbations (pruning, quantization, etc.) is fundamentally different from detecting model extraction. These two objectives differ not only in difficulty to be detected, but also in the efforts the adversary has to consent to reach the same accuracy.

Why does SACBench look so easy?

As we observed in Figure 1, the performance of fingerprints varies greatly from one benchmark to another. In this section, we try to uncover the reasons for this variability. A fingerprinting benchmark is essentially a procedure to generate positive and negative model pairs (h, h') by varying the model stealing and obfuscation methods. In the following, we investigate the properties of positive and negative pairs for each benchmark, in order to better understand the reasons why the various benchmarks seem to be unable to discriminate proposed fingerprint schemes and are beaten by the simple baseline presented in the previous section. ModelReuse and SACBench employ the same set of model stealing and obfuscation methods with two exceptions: ModelReuse uses model quantization as an obfuscation strategy, while SACBench performs adversarial model extraction. This explains the inferior performance of fingerprints based on adversarial sampling (ModelDiff and IPGuard) on SACBench.

However, the slight choice difference of the stealing and obfuscation methods included in ModelReuse compared to SACBench does not explain the exceptional performance of AKH and SAC compared to the other fingerprints. To that end, in Figure 3 we show the value of the conditioned Hamming distance δ_C (see Proposition 1) for all model pairs (h, h'). We note that the variability of the distance between h and h' is much higher for ModelReuse than for SACBench. This indicates that SACBench's process for creating the positive and negative pairs may not introduce enough diversity in the generated models, which could lead to overestimating the performance of its fingerprints. However, as observed in Figure 1, except SAC, all fingerprints have a comparable TPR@5% on SACBench and ModelReuse. To explain the difference in performance of AKH and SAC, we need to consider the separation between the distribution of $\delta_C(h, h')$ for the positive and negative model pairs (h, h'). Figure 3 shows a better separation between $\delta(h, h')$ for positive and negative pairs in SACBench. On the other hand, both datasets of ModelReuse show a large overlap in the distributions of distances of positive and negative pairs. Thus, since SAC is based on negative sampling, it appears that the generated positive and negative pairs of SACBench are especially well suited to the SAC fingerprint they introduce.

Related Works

Model-theft proactive defenses An alternative to fingerprinting is for the victim to choose a proactive solution consisting in *watermarking* their model (see, e.g., (Boenisch 2021; Regazzoni et al. 2021) for an overview), or by defending it using defenses implemented at training or inference time (Oliynyk, Mayer, and Rauber 2023; Tang et al. 2024).

Connections with tampering detection A problem closely related to model fingerprinting is *tampering* detection. The goal is to detect if a model served by a platform is the intended model originally sent by the owner, or if the model has been tampered with (Le Merrer and Trédan 2019; He, Zhang, and Lee 2019), by backdoor attacks (Gu, Dolan-Gavitt, and Garg 2019) for instance.

Connections with interpretable model distance *Interpretable* model distances help debug models. Instead of giving a single distance value, it also gives an explanation such as domains on where the models differ the most (Rida et al. 2023) or a simple approximation of the difference of the two models (Nair et al. 2021).

Conclusion

Our systematic analysis of the existing model fingerprinting schemes and benchmarks revealed a concerning evaluation artifact: the benchmarks studied are either not discriminative or solved by our simple AKH baseline. Firstly, most tasks are solved with almost any fingerprint. Secondly, the created victim/stolen model pairs are too easy to distinguish from victim/benign model pairs. Moreover, our QuRD framework reveals that schemes based on adversarial sampling are brittle compared to schemes using natural images.

While some of the tasks of model stealing detection can now be considered solved, several open challenges remain. One key issue is ensuring the robustness of fingerprinting techniques against adaptive adversaries who may actively attempt to evade detection. Furthermore, the development of effective fingerprints for other modalities than images would require further exploration.

Acknowledgments

The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-24-CE23-7787 (project PACMAM). This project was provided with computing AI and storage resources by GENCI at IDRIS thanks to the grant AD011015350 on the supercomputer Jean Zay's V100 partition. This research work was partially supported by the Hi! PARIS Center. A.G. would like to thank Dimitrios Los for the fruitful discussions on the theoretical analysis.

References

Ben-Sasson, H.; and Tzadik, S. 2024. Isolation or Hallucination? Hacking AI Infrastructure Providers for Fun and Weights.

Boenisch, F. 2021. A Systematic Review on Model Watermarking for Neural Networks. *Frontiers in Big Data*, 4: 729663.

Cao, X.; Jia, J.; and Gong, N. Z. 2021. IPGuard: Protecting Intellectual Property of Deep Neural Networks via Fingerprinting the Classification Boundary. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, ASIA CCS '21, 14–25. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-8287-8.

Carlini, N.; Paleka, D.; Dvijotham, K. D.; Steinke, T.; Hayase, J.; Cooper, A. F.; Lee, K.; Jagielski, M.; Nasr, M.; Conmy, A.; Wallace, E.; Rolnick, D.; and Tramèr, F. 2024. Stealing Part of a Production Language Model. arXiv:2403.06634.

Chen, J.; Wang, J.; Peng, T.; Sun, Y.; Cheng, P.; Ji, S.; Ma, X.; Li, B.; and Song, D. 2022. Copy, Right? A Testing Framework for Copyright Protection of Deep Learning Models. In *2022 IEEE Symposium on Security and Privacy (SP)*, 824–841.

Cohen, J.; Rosenfeld, E.; and Kolter, Z. 2019. Certified Adversarial Robustness via Randomized Smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, 1310–1320. PMLR.

Franzen, C. 2024. Mistral CEO Confirms 'Leak' of New Open Source AI Model Nearing GPT-4 Performance.

Goldreich, O. 2017. *Introduction to Property Testing*. Cambridge University Press, 1 edition. ISBN 978-1-107-19405-2 978-1-108-13525-2.

Gu, T.; Dolan-Gavitt, B.; and Garg, S. 2019. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. arXiv:1708.06733.

Guan, J.; Liang, J.; and He, R. 2022. Are You Stealing My Model? Sample Correlation for Fingerprinting Deep Neural Networks. In *Advances in Neural Information Processing Systems*, volume 35, 36571–36584.

He, Z.; Zhang, T.; and Lee, R. 2019. Sensitive-Sample Fingerprinting of Deep Neural Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4724–4732.

Jagielski, M.; Carlini, N.; Berthelot, D.; Kurakin, A.; and Papernot, N. 2020. High Accuracy and High Fidelity Extraction of Neural Networks. In *29th USENIX Security Symposium (USENIX Security 20)*, 1345–1362. ISBN 978-1-939133-17-5.

Jia, H.; Chen, H.; Guan, J.; Shamsabadi, A. S.; and Papernot, N. 2022. A Zest of LIME: Towards Architecture-Independent Model Distances. In *International Conference on Learning Representations*.

Le Merrer, E.; Pérez, P.; and Trédan, G. 2020. Adversarial Frontier Stitching for Remote Neural Network Watermarking. *Neural Computing and Applications*, 32(13): 9233–9244.

Le Merrer, E.; and Trédan, G. 2019. TamperNN: Efficient Tampering Detection of Deployed Neural Nets. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 424–434.

Lee, T.; Edwards, B.; Molloy, I.; and Su, D. 2019. Defending Against Neural Network Model Stealing Attacks Using Deceptive Perturbations. In *2019 IEEE Security and Privacy Workshops (SPW)*, 43–49.

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. P. 2017. Pruning Filters for Efficient ConvNets. In *International Conference on Learning Representations*.

Li, Y.; Zhang, Z.; Liu, B.; Yang, Z.; and Liu, Y. 2021. ModelDiff: Testing-Based DNN Similarity Comparison for Model Reuse Detection. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2021, 139–151. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-8459-9.

Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2018. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. In Bailey, M.; Holz, T.; Stamatogiannakis, M.; and Ioannidis, S., eds., *Research in Attacks, Intrusions, and Defenses*, 273–294. Cham: Springer International Publishing. ISBN 978-3-030-00470-5.

Lukas, N.; Zhang, Y.; and Kerschbaum, F. 2020. Deep Neural Network Fingerprinting by Conferrable Adversarial Examples. In *International Conference on Learning Representations*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

Maho, T.; Furon, T.; and Le Merrer, E. 2023. Fingerprinting Classifiers With Benign Inputs. *IEEE Transactions on Information Forensics and Security*, 18: 5459–5472.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2574–2582.

Nair, R.; Mattetti, M.; Daly, E.; Wei, D.; Alkan, O.; and Zhang, Y. 2021. What Changed? Interpretable Model Comparison. In *Twenty-Ninth International Joint Conference on Artificial Intelligence*, volume 3, 2855–2861.

Oh, S. J.; Augustin, M.; Fritz, M.; and Schiele, B. 2018. Towards Reverse-Engineering Black-Box Neural Networks. In *International Conference on Learning Representations*.

Ojha, U.; Li, Y.; and Lee, Y. J. 2023. Towards Universal Fake Image Detectors That Generalize Across Generative Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 24480–24489.

Oliynyk, D.; Mayer, R.; and Rauber, A. 2023. I Know What You Trained Last Summer: A Survey on Stealing Machine Learning Models and Defences. *ACM Computing Surveys*, 55(14s): 324:1–324:41.

Orekondy, T.; Schiele, B.; and Fritz, M. 2019. Prediction Poisoning: Towards Defenses Against DNN Model Stealing Attacks. In *International Conference on Learning Representations*.

Pan, X.; Yan, Y.; Zhang, M.; and Yang, M. 2022. MetaV: A Meta-Verifier Approach to Task-Agnostic Model Fingerprinting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, 1327–1336. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-9385-0.

Pan, X.; Zhang, M.; Lu, Y.; and Yang, M. 2021. TAFA: A Task-Agnostic Fingerprinting Algorithm for Neural Networks. In Bertino, E.; Shulman, H.; and Waidner, M., eds., *Computer Security – ESORICS 2021*, 542–562. Cham: Springer International Publishing. ISBN 978-3-030-88418-5.

Peng, Z.; Li, S.; Chen, G.; Zhang, C.; Zhu, H.; and Xue, M. 2022. Fingerprinting Deep Neural Networks Globally via Universal Adversarial Perturbations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13430–13439.

Regazzoni, F.; Palmieri, P.; Smailbegovic, F.; Cammarota, R.; and Polian, I. 2021. Protecting Artificial Intelligence IPs: A Survey of Watermarking and Fingerprinting for Machine Learning. *CAAI Transactions on Intelligence Technology*, 6(2): 180–191.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 1135–1144. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-4232-2.

Rida, A.; Lesot, M.-J.; Renard, X.; and Marsala, C. 2023. Dynamic Interpretability for Model Comparison via Decision Rules. arXiv:2309.17095.

Song, J.; Xu, Z.; Wu, S.; Chen, G.; and Song, M. 2023. ModelGiF: Gradient Fields for Model Functional Distance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6125–6135.

Tang, M.; Dai, A.; DiValentin, L.; Ding, A.; Hass, A.; Gong, N. Z.; and Chen, Y. 2024. MODELGUARD: Information-Theoretic Defense Against Model Extraction Attacks. In *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association.

Truong, J.-B.; Maini, P.; Walls, R. J.; and Papernot, N. 2021. Data-Free Model Extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4771–4780.

Wang, B.; and Gong, N. Z. 2018. Stealing Hyperparameters in Machine Learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, 36–52. IEEE Computer Society. ISBN 978-1-5386-4353-2.

Wang, S.; and Chang, C.-H. 2021. Fingerprinting Deep Neural Networks - a DeepFool Approach. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5.

Zhao, J.; Hu, Q.; Liu, G.; Ma, X.; Chen, F.; and Hassan, M. M. 2020. AFA: Adversarial Fingerprinting Authentication for Deep Neural Networks. *Computer Communications*, 150: 488–497.