Intersecting-Boundary-Sensitive Fingerprinting for Tampering Detection of DNN Models

Xiaofan Bai [*1 2 3 4 5] **Chaoxiang He** [*1 2 3 4 5] **Xiaojing Ma** [1 2 3 4 5] **Bin Benjamin Zhu** [6] **Hai Jin** [7 2 3 8]

Abstract

Cloud-based AI services offer numerous benefits but also introduce vulnerabilities, allowing for tampering with deployed DNN models, ranging from injecting malicious behaviors to reducing computing resources. Fingerprint samples are generated to query models to detect such tampering. In this paper, we present *Intersecting-Boundary-Sensitive Fingerprinting (IBSF)*, a novel method for black-box integrity verification of DNN models using only top-1 labels. Recognizing that tampering with a model alters its decision boundary, IBSF crafts fingerprint samples from normal samples by maximizing the partial Shannon entropy of a selected subset of categories to position the fingerprint samples near decision boundaries where the categories in the subset intersect. These fingerprint samples are almost indistinguishable from their source samples. We theoretically establish and confirm experimentally that these fingerprint samples' expected sensitivity to tampering increases with the cardinality of the subset. Extensive evaluation demonstrates that IBSF surpasses existing state-of-the-art fingerprinting methods, particularly with larger subset cardinality, establishing its state-of-the-art performance in black-box tampering detection using only top-1 labels. The IBSF code is available at: https://github.com/CGCL-codes/IBSF.

[*]Equal contribution [1] School of Cyber Science and Engineering, Huazhong University of Science and Technology [2] National Engineering Research Center for Big Data Technology and System [3] Services Computing Technology and System Lab [4] Hubei Engineering Research Center on Big Data Security [5] Hubei Key Laboratory of Distributed System Security [6]Microsoft [7] School of Computer Science and Technology, Huazhong University of Science and Technology [8] Cluster and Grid Computing Lab. Correspondence to: Xiaojing Ma <lindahust@hust.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

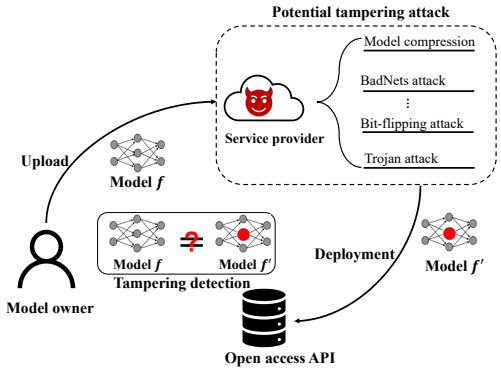


Figure 1. Potential adversarial tampering attacks on deployed models and the use of tampering detection to thwart them.

1. Introduction

Deep neural networks (DNNs) have made significant strides across various domains, such as in computer vision (He et al., 2016; Redmon et al., 2016; Long et al., 2015). To broaden the utilization of DNN models, cloud providers offer AI platforms like Microsoft Azure ML (Microsoft Azure, 2023), Google AutoML (Google Cloud, 2023), and Amazon SageMaker (Amazon Web Services, 2023). These platforms enable users to deploy their DNN models in the cloud, enhancing accessibility and scalability.

However, deploying DNN models in the cloud introduces several security risks, as depicted in Fig. 1. Notably, adversaries may exploit deployed DNN models through Trojan (Liu et al., 2018) or backdoor attacks (Gu et al., 2019), introducing malicious behavior. Furthermore, Unethical model providers might maliciously alter a competitor's model to undermine its performance and gain a competitive advantage. Additionally, dishonest cloud service providers could surreptitiously replace a deployed model with a simpler, compressed version to cut operational costs. These actions jeopardize the integrity and reliability of cloud-deployed DNN models.

To address these threats, various fingerprinting techniques have been proposed for tampering detection in deployed DNN models (He et al., 2019; Wang et al., 2023). These techniques verify deployed models by querying them with fingerprint samples, akin to normal queries. Given that deployed models often return only the top-1 category labels in

real-world applications, fingerprint samples must effectively differentiate the authentic model (referred to as the *target model* of fingerprints) from tampered models by yielding different top-1 label values. Thus, fingerprint samples must be highly sensitive to any alterations in the target model, ensuring that modifications lead to distinct label outputs for the same fingerprint sample. During the verification process, one or several fingerprint samples may be employed to query the model. The higher the sensitivity of these samples, the fewer are needed for effective verification, thus enhancing the efficiency of fingerprinting.

Several fingerprinting methods have been proposed. *Sensitive-sample fingerprinting (SSF)* (He et al., 2019) and its variants (Docena et al., 2021; Kuttichira et al., 2022) use a white-box approach to craft fingerprint samples for the target model by approximating and maximizing their sensitivity to tampering. In contrast, PublicCheck (Wang et al., 2023) employs a generative model to produce fingerprint samples positioned near the decision boundary of two intersected categories of the target model using a black-box approach. Among these methods, PublicCheck achieves *state-of-the-art (SOTA)* performance for integrity verification of DNN models. The fingerprint samples generated by these methods appear natural and are nearly visually indistinguishable from normal samples, thwarting adversaries' attempts to identify them for integrity verification (He et al., 2019). Additionally, to enable public integrity verification, (Wang et al., 2023) introduces a trusted third party to distribute fingerprint samples, allowing any interested party to verify if the deployed model has been tampered with.

In this paper, we propose a novel fingerprinting method, *Intersecting-Boundary-Sensitive Fingerprinting (IBSF)*. Based on the understanding that tampering with a model inevitably alters its decision boundary, IBSF strategically positions fingerprint samples near decision boundaries where K categories intersect (referred to as K-intersecting boundaries) of a target model for tampering detection. We formally define the sensitivity of samples to model tampering as the divergence of prediction vectors before and after tampering, measured with cross-entropy. Assuming that model tampering generally induces local boundary shifts that follow a probabilistically isotropic distribution, we theoretically establish that the tampering sensitivity of fingerprint samples located on the decision boundary increases with K, i.e., the number of intersected categories at its located boundary. Leveraging this analysis, IBSF employs partial Shannon entropy loss to effectively and efficiently locate K-intersecting boundaries and generate highly sensitive fingerprint samples near them, starting from normal samples. Generated fingerprint samples are almost indistinguishable from their source samples.

Similar to most existing fingerprinting methods (He et al., 2019; Docena et al., 2021), IBSF generates fingerprint samples with white-box access to the target model. This differs from PublicCheck (Wang et al., 2023), which employs black-box access to the target model in generating fingerprint samples. We argue that the practical advantage of black-box generation of fingerprint samples is negligible due to the very purpose of generating these fingerprint samples. Fingerprint samples can be generated by the model owner or a delegated party with access to the target model and sent to a trusted third party to provide fingerprint samples for public integrity verification. In this scenario, there is no need for any party to generate fingerprint samples in a black-box manner. Moreover, black-box generation of fingerprint samples introduces an additional risk: the model used in generating fingerprint samples cannot be verified and thus may already be tampered with, leaving a loophole for adversaries to evade fingerprint detection. On the other hand, IBSF conducts integrity verification of a deployed model in a black-box manner, similar to SSF (He et al., 2019) and PublicCheck (Wang et al., 2023). Only top-1 category labels are used in our integrity verification.

We conduct extensive experiments on three widely used datasets, CIFAR10, GTSRB, and ImageNet, to evaluate IBSF and compare it with two leading fingerprinting methods, SSF (He et al., 2019) and PublicCheck (Wang et al., 2023). Our evaluation encompasses various tampering scenarios, including backdoor and Trojan attacks, bit-flipping attacks, poisoning degradation attacks, targeted attacks, and model compression. The results consistently showed that IBSF outperformed existing methods across different types of tampering and datasets, highlighting its effectiveness in detecting DNN model tampering.

Our major contributions can be summarized as follows:

- We conduct theoretical analyses of fingerprint samples' sensitivity to model tampering and its relationship to K-intersecting boundary. We establish that sensitivity increases with K when the local decision boundary shifts due to model tampering following an isotropic probability distribution.
- Based on our theoretical analyses, we propose *intersecting boundary sensitive fingerprinting (IBSF)*, a novel fingerprinting method that maximizes the sensitivity of generated fingerprint samples by positioning them near boundaries where more categories intersect.
- We propose using partial Shannon entropy loss to effectively and efficiently locate K-intersecting boundaries for a selection of K categories.
- We conduct extensive empirical evaluations to demonstrate IBSF's superior performance over existing SOTA fingerprinting methods, including SSF and PublicCheck, and its robustness against adaptive attacks when a large number of fingerprint samples are leaked.

2. Related Work

2.1. Black-box Integrity Verification

Black-box integrity verification of DNN models has been explored using watermarking and fingerprinting techniques. Watermarking (Zhu et al., 2021) involves embedding a unique signature or pattern into the model during training. This embedded watermark can subsequently be queried with specific inputs to compare the model's output with the watermark for integrity verification of the model. However, watermarking necessitates modifications to the model during training, which could potentially impact its performance or efficiency.

In contrast, fingerprinting techniques, such as *Sensitive-Sample Fingerprinting(SSF)* (He et al., 2019), do not require modification to the model. They generate a set of sensitive samples that elicit distinct responses from models with slight parameter variations from the target model. These generated samples serve as fingerprints for integrity verification, with their responses being used to assess the integrity of the target model.

While fingerprinting techniques have also been employed for *intellectual property (IP)* protection (Cao et al., 2021; Wang & Chang, 2021; Lukas et al., 2021; Wang et al., 2021; Chen et al., 2021; Li et al., 2021; Zhao et al., 2020; Le Merrer et al., 2020; Yang et al., 2022; Pan et al., 2022), our focus here is on fingerprinting for integrity verification.

2.2. Fingerprinting Techniques for Integrity Verification

Sensitive-Sample Fingerprinting (SSF) (He et al., 2019) pioneered fingerprinting for detecting tampering in DNN models. It defines the sensitivity of a model to tampering and maximizes it to generate a collection of sensitive and visually natural samples that elicit significant output changes in response to parameter variations. SSF selects a subset of these samples to achieve maximum coverage of activated neurons, which then serve as fingerprint samples for integrity verification of the model.

Following the seminal work of SSF, *symbolic constraint solvers* are used in (Docena et al., 2021), and *Bayesian optimization (BO)* is used in (Kuttichira et al., 2022) to solve SSF's optimization problem in generating sensitive samples. PublicCheck (Wang et al., 2023) employs generative models to generate fingerprint samples with black-box access to the target model. It searches for the decision boundary that separates two different categories and generates fingerprint samples located in close proximity to the found decision boundary.

Both *AID* (Aramoon et al., 2021) and *Decision-based Fragile Watermarking (DBFW)* (Yin et al., 2023) generate fingerprint samples with uniform probability vectors by minimizing the logit and probability variance between different labels, respectively, utilizing *mean square error (MSE)* loss functions. The fingerprint samples generated by these methods exhibit similarities to those produced by our IBSF at decision boundaries where all categories intersect. However, unlike these methods, our IBSF is robustly supported by theoretical analysis of the relationship between fingerprint sensitivity and decision boundaries. Additionally, our IBSF employs a more efficient loss function, partial Shannon informative loss, compared to the MSE loss used in AID and DBFW.

All the above fingerprinting techniques utilize top-1 labels returned by the tested model to verify integrity. Our proposed method, IBSF, also follows this integrity verification paradigm.

3. Decision Boundary and Sample Sensitivity

In this paper, we focus on tampering detection for DNN models of classification tasks. This section begins by defining the decision boundary of a DNN model and the sensitivity of tampering detection. We then analyze the relationship between the decision boundary, where K categories intersect, and detection sensitivity.

3.1. Decision Boundary of DNN Models

A DNN can be represented as a function f parameterized by parameters W, mapping an input sample x to an output vector $\mathbf{y}$, where $\mathbf{y} = f(W, x)$. In classification tasks, an output vector typically represents the probabilities associated with various categories.

Definition 3.1 (Decision Boundary of a DNN Model). *The decision boundary is the hypersurface where the model changes its classification decision from one class to another, i.e., where the model assigns equal probabilities to two or more classes. For multi-class classification problems, the decision boundary between any two classes i and j can be represented as:*

$$f_i(W, x) = f_j(W, x) \tag{1}$$

where f_i and f_j are the model's predicted probabilities of classes i and j, respectively.

In classification tasks with C categories, when sample x is positioned near the decision boundaries where $k (k \leqslant C)$ categories intersect, the predicted probabilities of these k categories will be very close while significantly higher than those of the remaining $C - k$ categories.

3.2. Sample Sensitivity in Tampering Detection

Tampering with a DNN model involves modifying the model's parameters W. We denote model tampering as

ΔW. This alteration inevitably changes the decision boundary of the model and affects the predictions of some samples, forming the basis for black-box tampering detection using fingerprint samples.

Definition 3.2 (Sample Sensitivity to Model Tampering). *For a given model tampering ΔW, the sensitivity of a sample x to ΔW is the prediction divergence of x when ΔW is applied:*

$$S(x) = D(f(W,x) \| f(W+\Delta W, x)) = D(y \| y+\Delta y) \quad (2)$$

where $D(\cdot\|\cdot)$ denotes a divergence measure function of two probability vectors, y represents the prediction vector before tampering ΔW, and Δy represents the change in the prediction vector induced by ΔW.

Sample sensitivity measures the extent to which a sample's prediction deviates from the original when subjected to tampering. Effective fingerprint samples for tampering detection must be sensitive to various types of model tampering, including those unknown at the time of their creation, rather than being specific to only one type.

Definition 3.3 (Expected Sample Sensitivity over Model Tampering Distribution). *For a given model tampering distribution $\Delta W \sim \Phi$, the sensitivity of a sample x over Φ is the expected prediction divergence of x when $\Delta W \sim \Phi$ is applied:*

$$\begin{aligned} \mathbb{S}(x) &= \mathbb{E}_{\Delta W \sim \Phi}(D(f(W,x) \| f(W+\Delta W, x))) \\ &= \mathbb{E}_{\Delta y}(D(y \| y+\Delta y)) \end{aligned} \quad (3)$$

where $\mathbb{E}$ represents the expectation function.

We utilize cross-entropy as the divergence measure function, as it is commonly used in classification tasks. The cross-entropy between two probability vectors P and Q is defined as:

$$H(P,Q) = -\sum_i Q_i \log P_i \quad (4)$$

where i is a category index.

Combining Eq. 3 and Eq. 4, we have:

$$\begin{aligned} \mathbb{S}(x) &= \mathbb{E}_{\Delta y}(-\sum_i (y_i + \Delta y_i) \log y_i)) \\ &= (-\sum_i (y_i + \mathbb{E}(\Delta y_i)) \log y_i) \end{aligned} \quad (5)$$

This equation illustrates that, for a given sample x and a model tampering distribution Φ to detect, sample sensitivity to the tampering distribution can be optimized by sampling ΔW from Φ and calculating the empirically estimated expectation of prediction change $\mathbb{E}(\Delta y)$. While this approach is general, it is challenging to execute and find the optimal solution. In the following subsection, we will focus on the local decision boundary around which a fingerprint sample is located, and assume a local model tampering distribution around this location, instead of optimizing Eq. 5 with a global model tampering distribution. This approach will establish the relationship between the decision boundary where K categories intersect (referred to as K-intersecting boundary) and sample sensitivity.

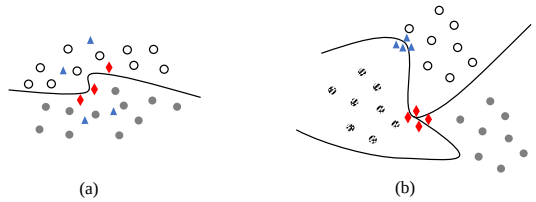


Figure 2. Illustration of IBSF. (a): The blue triangles represent normal samples, while the red diamonds represent fingerprint samples generated by IBSF. Due to the proximity of the fingerprint samples to the decision boundary, they are sensitive to shifts of the decision boundary. (b): The blue triangles represent fingerprint samples located at the intersecting decision boundary of two categories, while the red diamonds represent fingerprint samples located at the intersecting decision boundary of three categories.

3.3. Relationship between K-Intersecting Boundary and Sample Sensitivity

When a classification model is tampered with, its parameters change, inevitably altering the decision boundary. For black-box integrity verification of DNN models with top-1 category labels, fingerprint samples should be located around the decision boundary to be sensitive to changes in its local decision boundary, thus detecting whether the model has been tampered with or not. As shown in Fig. 2(a), any attempt to tamper with a DNN model results in a shift of the model's decision boundary, and this shift may alter the model's predicted category labels for fingerprint samples located around the local decision boundary, thereby triggering detection.

For a fingerprint sample, we can focus on the local decision boundary around which the fingerprint sample is located. Model tampering can alter the local decision boundary in different directions. We assume that model tampering alters the local boundary in any direction with equal probability, i.e., the local model tampering distribution is an isotropic probability. With this assumption, the expected prediction change would be zero, i.e., $\mathbb{E}(\Delta y_i) = 0$ for each category i.

Proposition 3.4. *Given that model tampering induces an expected prediction change of zero, i.e., $\mathbb{E}(\Delta y_i) = 0$, the expected sensitivity of samples located on $(k+1)$-intersecting boundaries is greater than that of those located on k-intersecting boundaries.*

Proof. Combining $\mathbb{E}(\Delta y_i) = 0$ and Eq. 5, we obtain:

$$\mathbb{S}(x) = \sum_{i=1}^{C} -y_i \cdot log\, y_i \quad (6)$$

This is Shannon entropy. Suppose $x^{(1)}$ and $x^{(2)}$ are fingerprint samples located on $(k+1)$- and k-intersecting boundaries, respectively, with prediction vectors of $y^{(1)} = \left(\frac{1}{k+1}, \frac{1}{k+1}, ..., \frac{1}{k+1}, 0, ..., 0\right)$ and $y^{(2)} = \left(\frac{1}{k}, \frac{1}{k}, ..., \frac{1}{k}, 0, ..., 0\right)$. Substituting these prediction vectors into Eq. 6, we obtain their expected sample sensitivities: $\mathbb{S}(x^{(1)}) = \log(k+1)$ and $\mathbb{S}(x^{(2)}) = \log k$. Since $\mathbb{S}(x^{(1)}) > \mathbb{S}(x^{(2)})$, we conclude that the expected sensitivity of the samples located near $(k+1)$-intersecting boundaries is higher than that of the samples located near k-intersecting boundaries. □

Proposition 3.4 establishes that samples located on decision boundaries where more categories intersect exhibit higher expected detection sensitivity to model tampering.

Proposition 3.5. *The maxima of Eq. 6 can be achieved when fingerprint samples are located on C-intersecting boundaries, where C is the number of categories, and $y_i = \frac{1}{C}$ for every category index i.*

Proof. We employ the *Lagrange multipliers method* to solve the constrained optimization problem of $\mathbb{S}(x) = \sum_{i=1}^{C} -y_i \cdot \log y_i$ with $\sum_{i=1}^{C} y_i = 1$ to prove that its maxima occur when $y_i = \frac{1}{C}$.

First, we formulate the Lagrangian function by introducing a Lagrange multiplier λ:

$$L(y_1, \ldots, y_C, \lambda) = \sum_{i=1}^{C} -y_i \cdot \log y_i + \lambda \left(\sum_{i=1}^{C} y_i - 1 \right) \quad (7)$$

To find the maxima of Eq. 7, we calculate the partial derivatives of L with respect to each y_i, and set them to zero:

$$\begin{aligned} &\frac{\partial L(y_i, \lambda)}{\partial y_i} \\ &= \frac{\partial[-(y_i \cdot \log y_i + \sum_{j \neq i}^{C} y_j \cdot \log y_j) + \lambda \cdot (y_i + \sum_{j \neq i}^{C} y_j - 1)]}{\partial y_i} \\ &= -\log y_i - 1 + \lambda = 0 \end{aligned} \quad (8)$$

Solving Eq. 8 for y_i, we have:

$$-\log y_i = 1 - \lambda \Rightarrow \log y_i = \lambda - 1 \Rightarrow y_i = e^{\lambda - 1} \quad (9)$$

Due to the symmetry in the derivatives (none of the derivatives depend specifically on i), all y_i are equal. Summing them up and using the constraint:

$$\sum_{i=1}^{C} e^{\lambda-1} = 1 \Rightarrow C \cdot e^{\lambda-1} = 1 \Rightarrow e^{\lambda-1} = \frac{1}{C} \Rightarrow y_i = \frac{1}{C} \quad (10)$$

Lastly, we confirm that the solution indeed represents a maximum. First, the sign of the second derivatives of $\mathbb{S}(x)$ with respect to each y_i is negative: $\frac{\partial^2 \mathbb{S}}{\partial y_i^2} = -\frac{1}{y_i}$, indicating it is a local maximum. Second, since $\mathbb{S}(x)$ is a sum of convex functions, the entire function is convex. Any local extremum for a convex function is a global extremum. Thus we can conclude that $\mathbb{S}(x)$ is maximized when each y_i is uniformly distributed as $\frac{1}{C}$. □

4. Intersecting-Boundary-Sensitive Fingerprinting (IBSF)

4.1. Threat Model

We adopt a threat model comprising white-box generation and black-box verification. Specifically, during the generation of fingerprint samples, we assume white-box access to the target DNN model, utilizing a set of available normal samples as source data. However, during the model's integrity verification using the generated fingerprint samples, we assume that the access to test model is black-box: only querying the model with samples in a standard operational manner is permitted. Additionally, we assume that the model only provides its top-1 prediction label in response to a query sample, without disclosing extra information such as confidence levels or prediction vectors.

Moreover, we assume the presence of a trustworthy third party responsible for securely storing and distributing the generated fingerprint samples to interested parties, facilitating public integrity verification. Conversely, we assume that the DNN service provider lacks trustworthiness and may detect probing fingerprint samples, thus evading tampering detection. Additionally, we assume that adversaries may gain access to some fingerprint samples, either through self-generation or collection from confederates, and exploiting them to bypass tampering detection.

4.2. IBSF Method

It's well established that in targeted adversarial attacks, a slight, nearly imperceptible modification to any input sample can cause a DNN model to classify it as any predetermined target label. Leveraging this property, we generate a fingerprint sample from a normal input sample by introducing a small, imperceptible perturbation to position the resulting fingerprint sample at an intersecting boundary of selected categories. Specifically, given a normal input data point $x \in \mathcal{X}$ and a subset of categories $\mathcal{C}_{sub} \subseteq \mathcal{C}$, where $\mathcal{C}$ is the set of categories in the classification task, we create a fingerprint sample by adding a perturbation Δx to x, guiding $x + \Delta x$ toward a decision boundary where each category in $\mathcal{C}_{sub}$ intersects.

When the fingerprint sample is situated around an intersecting boundary where categories in $\mathcal{C}_{sub}$ intersect, the prediction probabilities of these categories should be ap-

proximately equal, while the prediction probabilities of categories not in $\mathcal{C}_{sub}$ should be close to zero. This objective is achieved by maximizing the *partial Shannon entropy* of the categories in $\mathcal{C}_{sub}$:

$$H_{partial}(x, \mathcal{C}_{sub}) = -\sum_{i \in \mathcal{C}_{sub}} y_i \cdot \log y_i \tag{11}$$

where $y_i = f_i(x)$ represents the output probability of the i-th category. When Eq. 11 reaches its maximum, the prediction probabilities of categories in $\mathcal{C}_{sub}$ are equal, indicating that the fingerprint sample is positioned near a decision boundary where the categories in $\mathcal{C}_{sub}$ intersect.

Consequently, the optimization problem for IBSF can be formalized as:

$$\begin{aligned} &\text{argmax}_{\Delta x}\, H_{partial}(x + \Delta x, \mathcal{C}_{sub}) \\ &\text{s.t. } \|\Delta x\|_p \leqslant \epsilon \end{aligned} \tag{12}$$

The constraint $\|\Delta x\|_p \leqslant \epsilon$ limits the perturbation within an ϵ-ball under the L_p norm, ensuring that the generated fingerprint sample remains perceptually similar to the original normal sample.

Eq. 12 can be iteratively solved using *projected gradient ascent* as in (Kurakin et al., 2017b;a):

$$x_{k+1} = Clip_{\epsilon}(x_k + \eta \nabla_{x_k} H_{partial}(x_k, \mathcal{C}_{sub})) \tag{13}$$

where $x_k = x + \Delta x_k$ represents the generated fingerprint sample at the k-th iteration. This optimization process eventually yields a sample positioned near a decision boundary that separates categories within $\mathcal{C}_{sub}$, i.e., $|\mathcal{C}_{sub}|$-intersecting boundary, where $|\cdot|$ denotes the cardinality of the set. By default, IBSF selects the top $|\mathcal{C}_{sub}|$ categories of x.

When $\mathcal{C}_{sub} = \mathcal{C}$, Eq. 11 becomes Eq. 6. Propositions 3.4 and 3.5 indicate that fingerprint samples positioned at C-intersecting boundaries, where $C = |\mathcal{C}|$, exhibit higher expected sensitivity compared to those at K-intersecting boundaries, where $K < C$. However, generating fingerprint samples located at K-intersecting boundaries with $K < C$ provides additional variability and more decision boundaries to detect tampering, making it challenging for adversaries to evade detection even if some fingerprint samples are compromised, as elaborated in Section 5.6.

5. Experimental Evaluation

5.1. Datasets, Models, and Baselines

To evaluate detection performance, three commonly used datasets are adopted: CIFAR10 (Krizhevsky, 2009), GTSRB (Stallkamp et al., 2012), and ImageNet (Russakovsky et al., 2015). To evaluate the effectiveness of a fingerprinting method across diverse architectures, we employ a variety of models in our evaluation. Specifically, we use ResNet20 (He et al., 2016) for CIFAR10, a convolutional neural network (LeCun et al., 1995) (CNN) with 6 convolution layers and 1 fully-connected layer for GTSRB, and DenseNet121 (Huang et al., 2016) for ImageNet.

Two SOTA fingerprinting methods, SSF (He et al., 2019) and PublicCheck (Wang et al., 2023), serve as our baseline comparisons. We utilize the default settings outlined in their respective papers, unless explicitly specified otherwise. Additional implementation specifics are provided in Appendix A. For each fingerprinting method, we generate 1000 fingerprint samples. In the case of SSF, we select 10 samples with the highest coverage of activated neurons from a pool of 100 generated sensitive samples as fingerprint samples each time, until we obtain a total of 1000 fingerprint samples.

5.2. Tampering Types and Settings

In our experiments, we evaluate IBSF and the baselines on the 3 types of model tampering used by existing fingerprinting papers, BadNets (Gu et al., 2017), Trojan (Liu et al., 2018), and model compression, and three additional tampering types, the bit-flipping attack (Rakin et al., 2019), the poisoning degradation attack (Jagielski et al., 2021), and the targeted attack (Koh & Liang, 2017; Suciu et al., 2018; Shafahi et al., 2018). The last two tampering types are conducted at the individual sample level to minimize modifications to a target model. More details on these tampering attacks can be found in Appendix B.

Each tampering type is executed independently 10 times, resulting in 10 distinct models for each type. The reported experimental results are averaged across these 10 tampered models to mitigate randomness in evaluating tampering detection performance for fingerprinting methods.

5.3. Tampering Detection: Setup and Calculation

For each tamper detection, a subset of fingerprint samples, denoted as $\mathcal{S}_F$, is randomly selected from the pool of 1000 fingerprint samples to query the test model. The top-1 labels returned by the model are then compared with their ground-truth labels, which correspond to the top-1 labels predicted by the target model. If all returned labels match their respective ground truth labels, the model is considered untampered; otherwise, it is flagged as tampered. This process is repeated 100 times for each test model. Since we generate 10 tampered models for each attack type, we conduct a total of 1000 tests for each attack type. The tampering detection rate is calculated as the proportion of the 1000 tests where the tampering is successfully detected.

5.4. Perceptual Quality of Fingerprint Samples

Figure 3 displays fingerprint samples generated by IBSF and the two baselines. These samples generally appear nat-

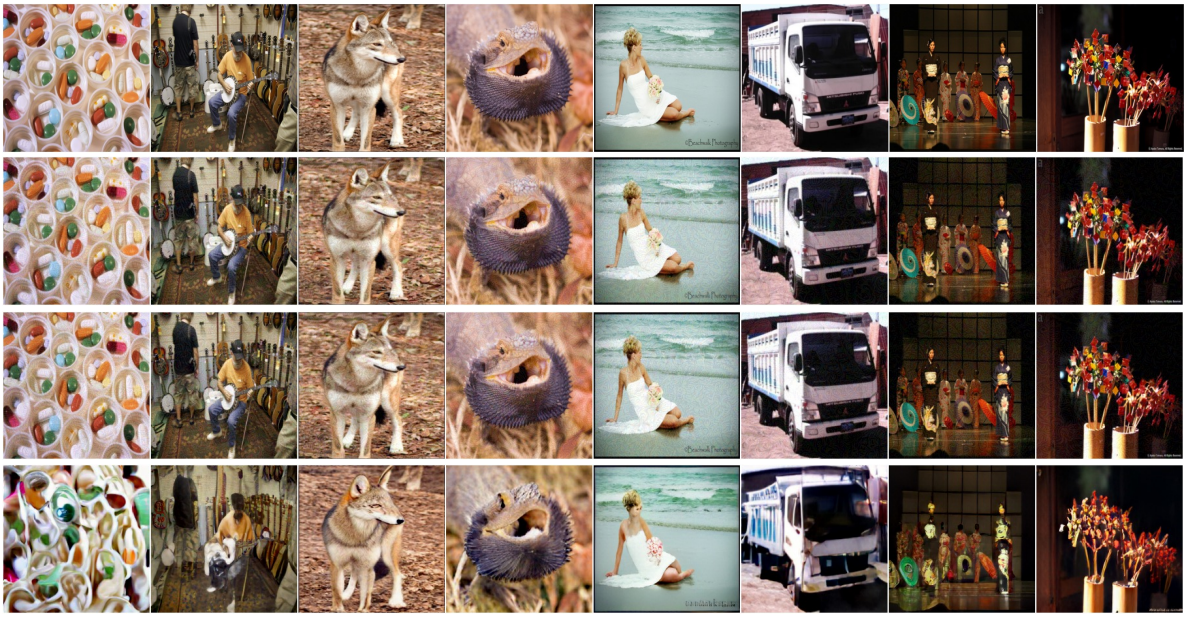


Figure 3. Source images (top) from ImageNet and fingerprint samples generated from them by IBSF (2nd row), SSF (3rd row), and PublicCheck (bottom).

ural, with almost imperceptible perturbations. Specifically, those generated by IBSF and SSF, constrained by the same L_2 norm bound, are very similar, featuring imperceptible pixel-level perturbations. However, PublicCheck's samples exhibit distinctly different artifacts, characterized by semantic-level alterations to the original images. For example, the distorted eye region of the wolf in the 3rd column of Fig. 3 vividly illustrates this effect. These alterations stem from the use of generative models in crafting the fingerprint samples, making them considerably more noticeable.

5.5. Effectiveness of Tampering Detection

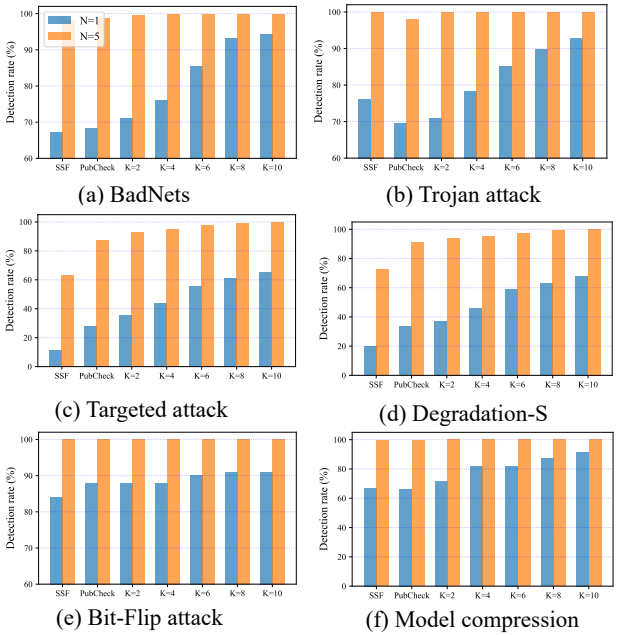
Figures 4, 5, and 6 present the tampering detection rates on CIFAR10, GTSRB, and ImageNet, respectively, of IBSF using fingerprint samples at K-intersecting boundaries, where $K \in \{2, 4, 6, 8, 10\}$ for CIFAR10, $K \in \{2, 4, 6, 8, 10, 20\}$ for GTSRB, and $K \in \{2, 4, 6, 8, 100, 500\}$ for ImageNet, alongside the two baselines. Key observations include:

Detection performance improves with increasing K: Increasing K notably enhances detection rates, consistent with our theoretical analysis of fingerprint sensitivity and K-intersecting boundaries. For instance, for targeted attacks on CIFAR10, IBSF's detection rate using a single fingerprint sample increases from 35.7% at $K = 2$ to 65.2% at $K = 10$, almost doubling the detection rate.

IBSF surpasses existing SOTA methods: IBSF generally outperforms the two SOTA baselines, although exceptions arise for IBSF at $K = 2$ when using a single fingerprint sample to detect Trojan attacks: its detection rate is 3.7% and 2.1% lower than SSF on CIFAR10 and GTSRB, respectively. This discrepancy can be attributed to SSF's susceptibility to modifications in the last fully connected (FC) layer, which Trojan attacks heavily affect. As K increases, IBSF gradually outperforms SSF, with improvements of up to 16.7% on CIFAR10 and 18.5% on GTSRB at $K = 10$. Similarly, while IBSF at $K = 2$ exhibits comparable performance to PublicCheck with a single fingerprint sample due to a shared underlying mechanism, it outperforms PublicCheck significantly as K increases. For example, at $K = 10$, IBSF detects targeted attacks on CIFAR10, GTSRB, and ImageNet with improvements of 1.32×, 0.89×, and 0.47×, respectively, compared to PublicCheck.

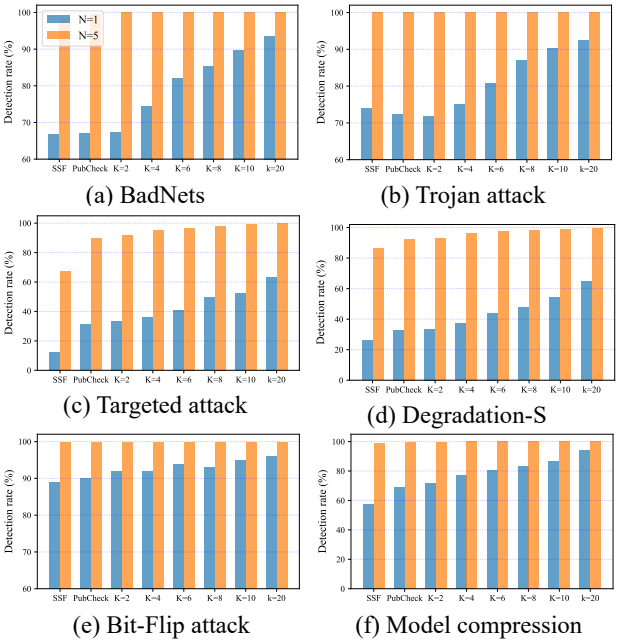
Moreover, using 5 fingerprint samples for detection, IBSF achieves perfect detection rates across all cases with sufficiently large K (greater than 8 for CIFAR10 and GTSRB, and 6 for ImageNet), while SSF and PublicCheck sometimes fail to detect tampering perfectly, particularly in cases like targeted and degradation attacks.

Challenges in detecting sample-level manipulations: Certain tampering types pose greater challenges for detection due to their subtle modifications. For instance, Degradation-S and targeted attacks are generally more difficult to detect. On CIFAR10 and GTSRB, both SSF and PublicCheck exhibit significantly lower tampering detection rates for these subtle tampering attacks compared to more severe tampering like pruning, BadNets, Trojan attacks, and bit-flipping attacks. These harder-to-detect tampering types typically



(a) BadNets (b) Trojan attack (c) Targeted attack (d) Degradation-S (e) Bit-Flip attack (f) Model compression

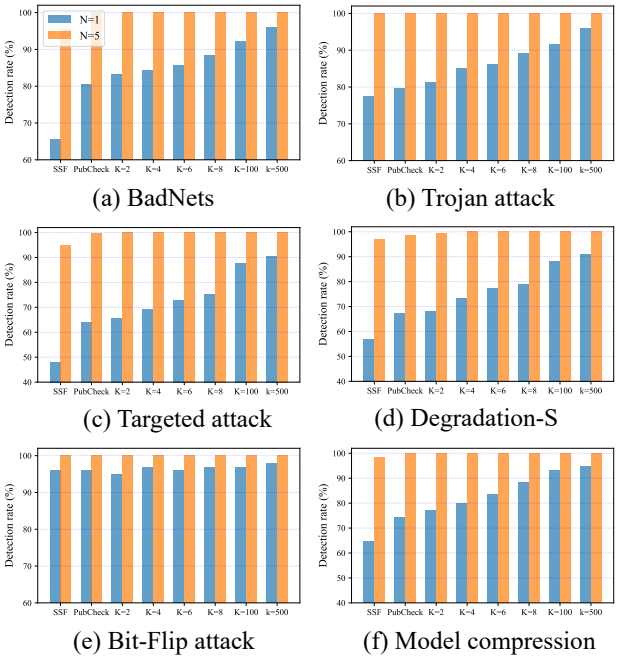
Figure 4. Tampering detection rates on CIFAR10 using 1 (blue) and 5 (orange) fingerprint samples. The leftmost two pairs of bars in each subfigure show the results of the two baselines, while others show the results of IBSF with K-intersecting boundaries.



(a) BadNets (b) Trojan attack (c) Targeted attack (d) Degradation-S (e) Bit-Flip attack (f) Model compression

Figure 5. Tampering detection rates on GTSRB using 1 (blue) and 5 (orange) fingerprint samples. The leftmost two pairs of bars in each subfigure show the results of the two baselines, while others show the results of IBSF with K-intersecting boundaries.

involve minor alterations to the target model, making them more challenging to detect. In contrast, IBSF achieves substantially higher tampering detection rates, particularly with higher K, when using a single fingerprint sample for these challenging tampering types.



(a) BadNets (b) Trojan attack (c) Targeted attack (d) Degradation-S (e) Bit-Flip attack (f) Model compression

Figure 6. Tampering detection rates on ImageNet using 1 (blue) and 5 (orange) fingerprint samples. The leftmost two pairs of bars in each subfigure show the results of the two baselines, while others show the results of IBSF with K-intersecting boundaries.

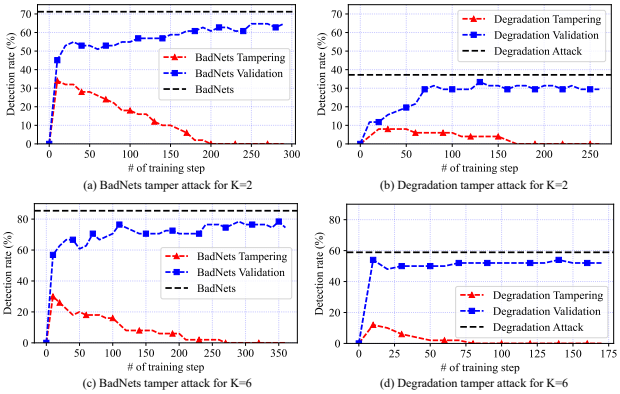
5.6. Robustness to Adaptive Attacks

Adversaries may attempt to evade tampering detection by leveraging previously used fingerprint samples or generating new ones using IBSF, employing them in adaptive attacks to bypass IBSF's detection. In such attacks, the adversary manipulates the model while ensuring that the top-1 labels on the collected fingerprint samples remain unchanged.

To assess resilience against adaptive attacks, we generate an additional 1,000 fingerprint samples using IBSF. These samples are then employed to launch adaptive attacks. This set of fingerprint samples is called the *tamper set*, while the set for tampering detection is termed as the *validation set*.

We conduct adaptive attacks using both BadNets and sample-level degradation attacks on CIFAR10. The results are shown in Fig. 7. As the training steps increase, the detection rates for the tamper set gradually decline to 0% for both BadNets and degradation attacks, indicating successful evasion of the tamper set by the adaptive attacks. However, the detection rates for the validation set remain reasonably high even after the adaptive attacks conclude their training, with a minor degradation of approximately 10% compared to the detection rates without adaptive attacks. These results suggest that adaptive attackers are ineffective against IBSF.

The resilience of IBSF against adaptive attackers stems from its diverse source samples. This diversity ensures that finger-



(a) BadNets tamper attack for K=2 (b) Degradation tamper attack for K=2

(c) BadNets tamper attack for K=6 (d) Degradation tamper attack for K=6

Figure 7. Tampering detection rates on CIFAR10 using a single fingerprint sample at K-intersecting boundary to detect adaptive attacks. (a)-(b): $K = 2$, (c)-(d): $K = 6$. Black dotted lines: no adaptive attacks. Red: tamper set. Blue: validation set.

print samples generated from different source samples are unlikely to occupy the same region of the decision boundaries. As a result, the generated fingerprint samples are distributed across various intersecting decision boundaries of the target model, with each sample occupying an independent and unpredictable local region. This characteristic reduces the influence of a fingerprint sample from one region on the detection capacity of samples from other regions, thereby enhancing robustness against adaptive attacks.

5.7. Computational Cost

We evaluate the computational cost of IBSF and the baselines, summarized in Table 1. The assessment was conducted on a single NVIDIA RTX3090 GPU. We measured the computational cost in terms of generation time (in seconds) and GPU memory utilization (in GB) per fingerprint sample for CIFAR10 and ImageNet, encompassing both small and large image resolutions, and across various model architectures with differing sizes. Our findings reveal the following insights:

- Computational costs tend to increase with higher image resolution or larger model parameters. Despite this trend, IBSF demonstrates notable computational efficiency per sample, positioning it as a resource-efficient solution for tampering detection.
- IBSF's effectiveness extends to challenging scenarios, including high-resolution images from ImageNet and GPU memory-demanding large-scale vision models like the Vision Transformer (ViT (Dosovitskiy et al., 2020)). As illustrated in Table 1, our approach achieves a generation time of 8.41 seconds and GPU memory utilization of 3.41 GB per fingerprint sample for ViT-L-32 on ImageNet, the largest model and dataset commonly used for classification tasks.
- IBSF's effectiveness extends to challenging scenarios, including high-resolution images from ImageNet and GPU memory demanding large-scale vision models like the Vision Transformer (ViT (Dosovitskiy et al., 2020)). As illustrated in Table 1, our approach achieves a generation time of 8.41 seconds and GPU memory utilization of 3.41 GB per fingerprint sample for ViT-L-32 on ImageNet, the largest model and dataset commonly used for classification tasks.
- IBSF exhibits versatility across various DNN architectures, encompassing ResNet, DenseNet, and Vision Transformer, among others. This adaptability facilitates effective use across models of different sizes, highlighting IBSF's broad applicability.

Table 1. Computational cost by model size and image resolution. Each A/B pair represents time cost (seconds) and memory utilization (GB) for generating a single fingerprint sample. Top three rows: CIFAR10, 32×32 pixels. Bottom three rows: ImageNet, 224×224 pixels.

Model Arch.	IBSF	SSF	PubCheck	Params
ResNet20	0.17/0.07	1.32/0.81	0.69/0.89	0.27M
ResNet18	0.21/0.17	1.48/1.25	0.71/0.91	11.2M
ResNet50	0.52/0.24	3.71/2.87	0.77/1.47	23.5M
DenseNet121	1.57/0.37	73.14/3.11	4.71/2.25	8.0M
ResNet50	3.91/0.82	141.36/7.06	17.12/4.21	25.6M
ResNet152	4.82/1.12	152.44/9.21	17.24/4.39	58.2M
ViT-L-32	8.41/3.41	193.21/14.2	18.12/5.63	306.5M

6. Conclusion

We introduced *Intersecting-Boundary-Sensitive Fingerprinting (IBSF)*, a novel method for black-box integrity verification of DNN models using only top-1 labels. Designed based on our theoretical analysis of the relationship between fingerprint sensitivity and decision boundaries where two or more categories intersect, IBSF crafts a fingerprint sample from a normal input sample by maximizing the partial Shannon entropy of a selected subset of categories to perturb the input sample, positioning it near a decision boundary where the categories in the subset intersect. The generated fingerprint sample is almost indistinguishable from its source sample. Our theoretical analysis, supported by experiments, indicates that fingerprint sensitivity to tampering increases with subset cardinality. Extensive evaluation confirms IBSF's superiority over existing methods, particularly with larger subset cardinality, establishing its state-of-the-art performance in black-box tampering detection using only top-1 labels.

Acknowledgements

This work was partially supported by National Natural Science Foundation of China under Grants No.U20A20177.

Impact Statement

This paper presents Intersecting-Boundary-Sensitive Fingerprinting (IBSF), a method aimed at enhancing the security of deep neural network (DNN) models deployed in cloud environments. The primary goal of our work is to advance the field of Machine Learning by providing a robust technique for tampering detection in DNN models, thereby ensuring the integrity and reliability of AI services.

Our contributions include the development of a novel fingerprinting method, a theoretical foundation for its effectiveness, and extensive experimental validation. While the direct implications of our work are technical, enhancing the security and trustworthiness of AI systems, we acknowledge the potential broader societal consequences:

1. **Ethical Aspects:** Ensuring the integrity of AI models is crucial for maintaining trust in AI-driven decisions across various sectors, including healthcare, finance, and autonomous systems. By detecting tampering, our method helps prevent malicious activities that could lead to harmful consequences.

2. **Future Societal Consequences:** As AI continues to integrate into daily life, securing AI models against tampering will become increasingly important. Our work contributes to this goal by providing a method that can be used to verify the integrity of AI models, thus promoting safer and more reliable AI applications.

3. **Potential Misuse:** While our method is designed to enhance security, we recognize that any technology can be misused. It is important for practitioners to apply IBSF responsibly and ethically, ensuring it is used to protect and not to undermine trust in AI systems.

In summary, this paper presents work whose goal is to advance the field of Machine Learning by improving the security and integrity of DNN models. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

AdamtayZzz. Trojanattack. https://github.com/AdamtayZzz/TrojanAttack, 2020.

Amazon Web Services. Amazon sagemaker. https://aws.amazon.com/sagemaker, 2023. Last accessed March 20, 2023.

Aramoon, O., Chen, P.-Y., and Qu, G. Aid: Attesting the integrity of deep neural networks. In *Proceedings of 2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 19–24, 2021. doi: 10.1109/DAC18074.2021.9586290.

Cao, X., Jia, J., and Gong, N. Z. Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pp. 14–25, 2021.

Chadebec, C., Vincent, L., and Allassonniere, S. Pythae: Unifying generative autoencoders in python - a benchmarking use case. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 21575–21589. Curran Associates, Inc., 2022.

Chen, Y., Shen, C., Wang, C., and Zhang, Y. Teacher model fingerprinting attacks against transfer learning. *arXiv preprint arXiv:2106.12478*, 2021.

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

Docena, A. N., Wahl, T., Pearce, T., and Fei, Y. Sensitive samples revisited: Detecting neural network attacks using constraint solvers. *arXiv preprint arXiv:2109.03966*, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Esser, P., Rombach, R., and Ommer, B. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12873–12883, 2021.

Google Cloud. AutoML. https://cloud.google.com/automl, 2023. Last accessed March 20, 2023.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Gu, T., Liu, K., Dolan-Gavitt, B., and Garg, S. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

He, Z. Sensitive sample fingerprinting. https://github.com/zechenghe/Sensitive_Sample_Fingerprinting, 2021. Last accessed Jan. 25, 2022.

He, Z. Bit-flips attack and defense. https://github.com/elliothe/BFA, 2022. Last accessed Jan. 25, 2022.

He, Z., Zhang, T., and Lee, R. Sensitive-sample fingerprinting of deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4729–4737, 2019.

Huang, G., Liu, Z., and Weinberger, K. Q. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

Jagielski, M., Severi, G., Pousette Harger, N., and Oprea, A. Subpopulation data poisoning attacks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3104–3122, 2021.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *Proceedings of International Conference on Machine Learning*, pp. 1885–1894. PMLR, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. *M.S. thesis, Department of Computer Science, University of Toronto*, 2009.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. In *Proceedings of 5th International Conference on Learning Representations ICLR 2017*, 2017a.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. In *Proceedings of 5th International Conference on Learning Representations ICLR 2017, Workshop track*, 2017b.

Kuttichira, D. P., Gupta, S., Nguyen, D., Rana, S., and Venkatesh, S. Verification of integrity of deployed deep learning models using bayesian optimization. *Knowledge-Based Systems*, 241:108238, 2022.

Le Merrer, E., Perez, P., and Trédan, G. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244, 2020.

LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P., and Vladimir, V. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural Networks*, pp. 261–276, 1995.

Li, Y., Zhang, Z., Liu, B., Yang, Z., and Liu, Y. Modeldiff: testing-based DNN similarity comparison for model reuse detection. In *Proceedings of 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA'21*, pp. 139–151, 2021.

Liu, Y., Ma, S., Aafer, Y., Lee, W., Zhai, J., Wang, W., and Zhang, X. Trojaning attack on neural networks. In *Proceedings of 25th Annual Network and Distributed System Security Symposium, NDSS 2018*, 2018.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

Lukas, N., Zhang, Y., and Kerschbaum, F. Deep neural network fingerprinting by conferrable adversarial examples. In *Proceedings of 9th International Conference on Learning Representations, ICLR 2021*, 2021.

Microsoft. Neural network intelligence. https://github.com/microsoft/nni, 2022.

Microsoft Azure. Microsoft machine learning. https://azure.microsoft.com/en-us/products/machine-learning, 2023. Last accessed March 20, 2023.

Molchanov, P., Mallya, A., Tyree, S., Frosio, I., and Kautz, J. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019.

Pan, X., Yan, Y., Zhang, M., and Yang, M. Metav: A meta-verifier approach to task-agnostic model fingerprinting. In *Proceedings of 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pp. 1327–1336. ACM, 2022. doi: 10.1145/3534678.3539257.

Rakin, A. S., He, Z., and Fan, D. Bit-flip attack: Crushing neural network with progressive bit search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1211–1220, 2019.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Alexander, C. B., and Li, F.-F. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pp. 6106–6116, 2018.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32: 323–332, 2012.

Suciu, O., Marginean, R., Kaya, Y., Daume III, H., and Dumitras, T. When does machine learning fail? generalized transferability for evasion and poisoning attacks. In *Proceedings of 27th USENIX Security Symposium (USENIX Security 18)*, pp. 1299–1316, 2018.

verazuo. badnets-pytorch. https://github.com/verazuo/badnets-pytorch, 2022.

Wang, S. and Chang, C.-H. Fingerprinting deep neural networks-a deepfool approach. In *Proceedings of 2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. IEEE, 2021.

Wang, S., Wang, X., Chen, P.-Y., Zhao, P., and Lin, X. Characteristic examples: High-robustness, low-transferability fingerprinting of neural networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI*, pp. 575–582, 2021.

Wang, S., Abuadbba, S., Agarwal, S., Moore, K., Sun, R., Xue, M., Nepal, S., Camtepe, S., and Kanhere, S. Publiccheck: Public integrity verification for services of run-time deep models. In *Proceedings of 2023 IEEE Symposium on Security and Privacy (SP)*, pp. 1348–1365. IEEE, 2023.

Yang, K., Wang, R., and Wang, L. Metafinger: Fingerprinting the deep neural networks with meta-training. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pp. 776–782. ijcai.org, 2022.

Yin, Z., Yin, H., Su, H., Zhang, X., and Gao, Z. Decision-based iterative fragile watermarking for model integrity verification. *arXiv preprint arXiv:2305.09684*, 2023.

Zhao, J., Hu, Q., Liu, G., Ma, X., Chen, F., and Hassan, M. M. Afa: Adversarial fingerprinting authentication for deep neural networks. *Computer Communications*, 150: 488–497, 2020.

Zhu, R., Wei, P., Li, S., Yin, Z., Zhang, X., and Qian, Z. Fragile neural network watermarking with trigger image set. In *Proceedings of International Conference on Knowledge Science, Engineering and Management*, pp. 280–293. Springer, 2021.

A. Implementation Details

For SSF, we use the official open-source code with the default settings (He, 2021). Fingerprint samples are bound with the L_2-norm to be perceptually similar to original samples. PublicCheck uses generative models to generate fingerprint samples near the decision boundary. For CIFAR10 and GTSRB, we train VQVAEs as generative models using their training sets with Pythea (Chadebec et al., 2022). For ImageNet, we use a pre-trained VQVAE from (Esser et al., 2021).

B. Model Tampering Setup

We use the following tampering types to evaluate the performance of fingerprinting methods.

- **Backdoor attacks**: A backdoor attack aims to insert a backdoor into a model to make it misclassify any sample carrying a designated trigger to a predetermined label. We evaluate with two types of backdoor attacks, BadNets (Gu et al., 2017) and Trojan (Liu et al., 2018), with open-source code from (verazuo, 2022; AdamtayZzz, 2020). For both attack types and all the three datasets in our experiments, a square trigger is positioned at the bottom-right corner and the injection ratio is set to 0.1. The trigger size is set to 4×4 pixels for CIFAR10 and GTSRB, and 8×8 pixels for ImageNet.
- **Model compression** (Cheng et al., 2017): It reduces the size of a DNN model to conserve computing resources. Model pruning (Molchanov et al., 2019) removes components with the least contribution to the model's performance and then recovers the lost performance by fine-tuning the remaining components. The official pruning API of Microsoft NNI (Microsoft, 2022) is used, with the pruning ratio set to 20%.
- **Bit-flipping attack** (Rakin et al., 2019): It aims to degrade a model's accuracy by reversing the most vulnerable bits of the model. Open-source code (He, 2022) is used to flip the most crucial bit.
- **Poisoning degradation attack** (Jagielski et al., 2021): It degrades the accuracy of a model by contaminating its training data. In our experiments, we randomly mislabel a training sample and fine-tune all layers of the model. This attack is denoted as *Degradation-S* in the paper.
- **Targeted attack** (Koh & Liang, 2017; Suciu et al., 2018; Shafahi et al., 2018): This attack aims to make a model mispredict specific input samples to a specific label while predicting other input samples normally. In our experiments, modifications to a target model are minimized by randomly selecting a training sample, assigning the label with the shortest moving distance from the original label at the penultimate layer, and fine-tuning only the last *fully connected* (FC) layer.