False Claims against Model Ownership Resolution

Jian Liu and Rui Zhang, *Zhejiang University;* Sebastian Szyller, *Intel Labs & Aalto University;* Kui Ren, *Zhejiang University;* N. Asokan, *University of Waterloo & Aalto University*

https://www.usenix.org/conference/usenixsecurity24/presentation/liu-jian

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

False Claims against Model Ownership Resolution*

Jian Liu†
Zhejiang University
jian.liu@zju.edu.cn

Rui Zhang†
Zhejiang University
zhangrui98@zju.edu.cn

Sebastian Szyller
Intel Labs & Aalto University
contact@sebszyller.com

Kui Ren
Zhejiang University
kuiren@zju.edu.cn

N. Asokan
University of Waterloo & Aalto University
asokan@acm.org

Abstract

Deep neural network (DNN) models are valuable intellectual property of model owners, constituting a competitive advantage. Therefore, it is crucial to develop techniques to protect against model theft. Model ownership resolution (MOR) is a class of techniques that can deter model theft. A MOR scheme enables an *accuser* to assert an ownership claim for a *suspect model* by presenting evidence, such as a watermark or fingerprint, to show that the suspect model was stolen or derived from a source model owned by the accuser. Most of the existing MOR schemes prioritize robustness against malicious suspects, ensuring that the accuser will win if the suspect model is indeed a stolen model.

In this paper, we show that common MOR schemes in the literature are vulnerable to a different, equally important but insufficiently explored, robustness concern: a *malicious accuser*. We show how malicious accusers can successfully make *false claims* against *independent* suspect models that were not stolen. Our core idea is that a malicious accuser can deviate (without detection) from the specified MOR process by finding (transferable) adversarial examples that successfully serve as evidence against independent suspect models. To this end, we first generalize the procedures of common MOR schemes and show that, under this generalization, defending against false claims is as challenging as preventing (transferable) adversarial examples. Via systematic empirical evaluation, we show that our false claim attacks always succeed in MOR schemes that follow our generalization, including in a real-world model: Amazon's Rekognition API.

1 Introduction

Deep Neural Networks (DNNs) have been used extensively in many real-world applications such as facial recognition [35,50,66], medical image classification [73] and autonomous driving [40]. However, training DNNs is expensive due to the high costs of preparing training data and fine-tuning models. Therefore, DNNs confer a competitive advantage to model owners who would like to prevent theft and unauthorized redistribution of their *source models*. A thief may deploy a *stolen model* for profit. The stolen model can be an exact copy of the source model (with possible subsequent refinement) or a surrogate model *extracted* by querying the source model's inference interface.

Model ownership resolution (MOR). Preventing model stealing is difficult [2,9,23,28], but mechanisms for detecting a stolen model and resolving ownership serve as powerful deterrents. A *model ownership resolution* (MOR) scheme enables an *accuser* ($\mathcal{A}$) to present evidence, such as a watermark or fingerprint, to a *judge* ($\mathcal{J}$) and claim that a model held by a *suspect* ($\mathcal{S}$) is a stolen model.

DNN watermarking [1,32,72] is one type of MOR; it embeds a watermark into a DNN during training. A watermark typically consists of a set of samples with incorrectly assigned labels, known as the *trigger set*. The model owner uses the trigger set, along with the training set, to train the source model. The watermarked source model, and any model derived from it, will perform differently on the trigger set compared to other *independent*[1] models. Therefore, the trigger set can help to resolve the ownership.

DNN fingerprinting is another type of MOR, which extracts a unique identifying code (fingerprint) from an already trained model. Unlike watermarking, which embeds a watermark into a DNN during training, fingerprinting does not require any changes to the training phase and hence does not sacrifice model accuracy. Similar to watermarking, verification of a fingerprint involves querying a suspect model using a trigger set that corresponds to the fingerprint.

False claims against MOR. Most of the existing MOR schemes prioritize the robustness against a malicious suspect: they try to make sure that the accuser will win the case if the suspect model is indeed a stolen model. In this paper, we

*Research report version at https://arxiv.org/abs/2304.06607
†Co-first authors; Jian Liu is the corresponding author.

[1]Henceforth, we use "independent models" to represent models that are independently trained for the same task as $\mathcal{A}$'s source model.

focus on a different under-explored robustness problem: that of a *malicious accuser*. Specifically, we investigate whether a malicious accuser can falsely claim ownership of an independent suspect model that is *not* a stolen model.

We posit that the robustness of a MOR scheme against malicious accusers is just as crucial as its robustness against malicious suspects. Indeed, if a malicious accuser can falsely claim ownership of independent models, the corresponding MOR scheme will not be useful in settings that need to resolve legal model ownership.

The judge's MOR decision is based on the inference results of a suspect model on the trigger set. Our intuition is that a malicious accuser can construct a trigger set in a way to raise false positives for all independent models. This can be achieved via (the transferability of) adversarial examples. To demonstrate the ubiquity of this attack, we first generalize the procedures for common MOR schemes, and then show that any MOR scheme following our generalization is susceptible to this attack. We survey 16 MOR schemes and show that our generalization can effectively capture these surveyed schemes.[2] We empirically evaluate several well-known MOR schemes [1, 24, 39, 43, 59] to show that all of them are vulnerable to our false claim attacks. Specifically, we attempt to falsely claim models trained from CIFAR-10 and ImageNet, as well as the model behind Amazon Rekognition API [54]. Our evaluation shows that, in realistic configurations, our false claims can succeed against all evaluated MOR schemes for all models.

We summarize our contributions as follows:

- We present a generalization of MOR procedures, and an adversary model for malicious accusers. (Section 3) We show that all secure MOR schemes follow our generalization; (Section 4)
- We show that false claims against these schemes can always succeed, unless (transferable) adversarial examples can be prevented; (Section 5)
- We empirically evaluate these schemes including on a real-world model held by Amazon, to show that our false claim attacks do succeed[3]; (Section 6)
- We provide some guidance on augmenting MOR schemes to withstand our attacks. (Section 7)

2 Preliminaries

In this section, we provide some fundamental concepts to facilitate the understanding of this paper.

[2] Parameter-encoding watermarking schemes do not conform our generalization. However they are known to be *not* robust [68]. (cf. Section 9)

[3] Source code at https://github.com/ssg-research/Falseclaims

Table 1: Summary of frequent notations.

Notation	Description
$\mathcal{A}$	Accuser
$\mathcal{S}$	Suspect
$\mathcal{J}$	Judge
$F_{\mathcal{A}}$	source model
$F_{\mathcal{S}}$	suspect model
$moc_{\mathcal{A}}$	a model ownership claim submitted by $\mathcal{A}$
$f()$	ground-truth function
$\nu()$	a verification function
cm	a cryptographic commitment
x	a sample
$\mathbf{x}$	a set of samples
y	a label
$\mathbf{y}$	a set of labels
T	a decision threshold
MOR	model ownership resolution
$MORacc$	MOR accuracy
ε	pre-pixel perturbation bounds

2.1 Deep neural networks (DNN)

A *deep neural network* (DNN) model is a mathematical function F that assigns a label y to a sample x:

$$y \leftarrow F(x).$$

A DNN consists of a series of layers, with each layer employing a linear function followed by an activation function. In particular, the softmax activation function is commonly employed in the output layer to convert likelihood scores into class probabilities.

To train a DNN, it is necessary to specify a differentiable loss function L that serves as an objective during the optimization process. The cross-entropy loss is a commonly used loss function in DNN training, which quantifies the discrepancy between the predicted output of the model and the ground truth label.

A black-box deployment of a DNN exposes only the API of the model: given an input, the DNN API returns a class or class probabilities.

2.2 Adversarial examples

Adversarial examples were first reported in 2013 as slightly perturbed images that nudge DNNs into making incorrect predictions. Such adversarial examples look almost the same as original images and were seen as a systematic vulnerability in DNNs [19]. Since then, they have been explored extensively in both attacks [16, 58] and defences [4, 7, 41, 49, 67, 76].

An adversarial example is generated by solving:

$$r = \arg\min_r L(F(x+r), y') + \alpha||r||,$$

where y' is different from the real label y for x. Then, the noise r can fool the DNN into predicting a wrong label (by minimizing the loss function) with imperceptible perturbations.

It is well-known that adversarial examples are *transferable*: the adversarial examples generated for one model could mislead another model [36, 47, 48]. Such a property can be leveraged to generate adversarial examples for black-box models.

2.3 Model extraction

Model extraction is a kind of black-box attack aiming to obtain an *extracted model* that is *functionally equivalent* to the victim model. Black-box refers to an attacker who gleans information from the victim model solely by interacting with its prediction API: they choose queries (inputs) and obtain the corresponding labels. Using that information, they train their own model.

It is commonly assumed that the attacker does not know the exact architecture or the training data used to train the victim model. However, they can choose an architecture suitable for the task and appropriate data to execute the attack.

Model extraction is effective in various tasks: image classification [13, 23, 26, 45, 48, 63], image translation [60], NLP [28, 65], and others [22, 62].

3 MOR: Generalization

In this section, we aim to generalize the procedures for common MOR schemes and define the capabilities/goals of a malicious accuser. Table 1 summarizes our notations.

3.1 MOR procedures

A MOR scheme typically consists of two procedures:

- *claim generation* allows the owner (who can later act as an accuser $\mathcal{A}$) of a model $F_\mathcal{A}$ to generate a model ownership claim $moc_\mathcal{A}$ (watermark/fingerprint) for $F_\mathcal{A}$;
- *claim verification* allows a judge $\mathcal{J}$ to use $moc_\mathcal{A}$ to determine whether a model $F_\mathcal{S}$ held by a suspect $\mathcal{S}$ is a stolen model that is derived from $F_\mathcal{A}$.

Next, we formally define these two procedures, attempting to cover most MOR schemes.

Claim generation. Given $F_\mathcal{A}$, $\mathcal{A}$ creates $moc_\mathcal{A}$, which is formally defined as follows:

Definition 1. *A model ownership claim for a model $F_\mathcal{A}$ is defined as $moc_\mathcal{A} = (\mathbf{x}, \mathbf{y}, aux, cm_\mathcal{A})$, where $\mathbf{x}$ is a set of data samples, $\mathbf{y}$ is the corresponding set of labels, aux is auxiliary information, and $cm_\mathcal{A}$ is a cryptographic commitment of $(\mathcal{A}, F_\mathcal{A}, \mathbf{x}, \mathbf{y}, aux)$ which receives a secure timestamp that can be verified by $\mathcal{J}$, for example by posting the commitment on a timestamped public bulletin board. The* trigger set *is* $(\mathbf{x}, \mathbf{y})$.

We remark that most MOR schemes (except DAWN [59]) do not explicitly mention that the commitment is timestamped, and some do not even mention commitments at all. We add timestamped commitments to our definition for two reasons:

- Without them, false claims are easier (cf. Section 8). We consider the most difficult setting for our attack.
- Integrating them into existing MOR schemes is straightforward.

For a verification function ν, the claim should satisfy: for $F_\mathcal{A}$,$\forall(x_i, y_i) \in (\mathbf{x}, \mathbf{y})$, $\nu(F_\mathcal{A}(x_i), y_i) = 1$ and for other $F \neq F_\mathcal{A}$, $\nu(F(x_i), y_i) = 0$. For example, $\mathcal{A}$ could achieve this in following ways:

- Sample (x_i, y_i) s.t. $\nu(F(x_i), y_i) = 0$ for all Fs, and fine-tune $F_\mathcal{A}$ on (x_i, y_i) s.t. $\nu(F_\mathcal{A}(x_i), y_i) = 1$. DNN watermarking [1, 20, 32, 72] is in this category.
- Exploit $F_\mathcal{A}$ to generate (x_i, y_i) (e.g., as an adversarial example) s.t. $\nu(F_\mathcal{A}(x_i), y_i) = 1$ for $F_\mathcal{A}$, and $\nu(F(x_i), y_i) = 0$ for other $F \neq F_\mathcal{A}$. DNN fingerprinting [8, 39, 46, 75] falls into this category.

Claim verification. Suppose $\mathcal{A}$ uses $moc_\mathcal{A}$ to claim that a model $F_\mathcal{S}$ (held by a suspect $\mathcal{S}$) was derived from $F_\mathcal{A}$. The judge $\mathcal{J}$ checks the followings:

- $\nu(F_\mathcal{A}(x_i), y_i) = 1, \forall(x_i, y_i) \in (\mathbf{x}, \mathbf{y})$;
- $\nu(F_\mathcal{S}(x_i), y_i) = 1, \forall(x_i, y_i) \in (\mathbf{x}, \mathbf{y})$;[4]
- $cm_\mathcal{A}$ is a valid commitment of $(\mathcal{A}, F_\mathcal{A}, \mathbf{x}, \mathbf{y})$;
- $cm_\mathcal{A}$ was timestamped before $cm_\mathcal{S}$ (if $cm_\mathcal{S}$ exists and was timestamped).

$\mathcal{J}$ accepts $\mathcal{A}$'s claim if and only if all checks pass. Recall that for $F_\mathcal{A}$, $\nu(F_\mathcal{A}(x_i), y_i) = 1$, and for other $F \neq F_\mathcal{A}$, $\nu(F(x_i), y_i) = 0$.

Therefore, $\nu(F_\mathcal{S}(x_i), y_i) = 1\ \forall(x_i, y_i) \in (\mathbf{x}, \mathbf{y})$ only if $F_\mathcal{S}$ was derived from $F_\mathcal{A}$.

In fact, the first and second checks do not need to hold for all $(x_i, y_i) \in (\mathbf{x}, \mathbf{y})$. Instead, we define the MOR accuracy based on the watermark accuracy proposed in [38]:

$$MORacc_F = \frac{1}{|\mathbf{x}|} \sum_{i=1\ldots|\mathbf{x}|} \mathbb{I}(\nu(F(x_i), y_i)). \tag{1}$$

We say the check holds if $MORacc_F$ is higher than a *decision threshold* T (cf. Section 3.3 for further discussions on T).

The main characteristic of this generalization is that $moc_\mathcal{A}$ includes a trigger set and its verification requires running model inference on the trigger set. All MOR schemes we discuss (Section 4) follow our generalization.

3.2 False claims

A robust MOR scheme should satisfy the following:

[4] Notice that $\mathcal{J}$ only needs API access to $F_\mathcal{S}$; $\mathcal{S}$ need not be aware of the verification being done.

- robustness against a malicious $\mathcal{S}$, i.e., $\mathcal{J}$ accepts the claim if $F_{\mathcal{S}}$ is indeed a stolen model;
- robustness against a malicious $\mathcal{A}$, i.e., $\mathcal{J}$ rejects the claim if $F_{\mathcal{S}}$ is an independent model.

Most existing MOR schemes prioritize robustness against a malicious $\mathcal{S}$. We focus on studying robustness against a malicious $\mathcal{A}$, who aims to **falsely claim the ownership of an independent model** $F_{\mathcal{S}}$.

We assume that $\mathcal{A}$ has access to a dataset with the same distribution as the training dataset of $F_{\mathcal{S}}$. This is a standard assumption in MOR literature [5,33,38]: it implies that a false claim will succeed against $F_{\mathcal{S}}$ trained from a dataset with the same distribution as the training dataset of $F_{\mathcal{A}}$.

The timestamped commitment requirement from Definition 1 implies that $\mathcal{A}$ has to create $moc_{\mathcal{A}}$ even before $F_{\mathcal{S}}$ comes into existence. Therefore, $\mathcal{A}$ cannot know anything about $F_{\mathcal{S}}$ including its parameters or hyperparameters ahead of time.

False claim vs. watermark forging. Watermark forging [20, 70, 72] allows an attacker to forge a watermark on a given model, creating an ambiguity for $\mathcal{J}$ to determine which party has watermarked the given model. Li et al [33] describe three types of watermarking forging:

- Recovering watermarks and claiming ownership [70];
- Adding a new watermark [17, 34];
- Extracting a fake watermark from the model that acts like a real one [20].

Our proposed MOR generalization, which includes a timestamped commitment, renders such attacks ineffective. This is because, in case $\mathcal{A}$'s and $\mathcal{S}$'s claims are both valid, $\mathcal{J}$ can easily resolve ownership based on the timestamps. It is impossible for the attacker to possess an earlier timestamp since watermark forging requires knowing $F_{\mathcal{S}}$, which is available only after the owner commits the model with a timestamp.

Therefore, the false claims considered in this paper is stronger than watermark forging. We aim to allow $\mathcal{A}$ to claim ownership of $F_{\mathcal{S}}$ even if $F_{\mathcal{S}}$ is timestamped. To this end, $\mathcal{A}$ has to generate a valid $moc_{\mathcal{S}}$ for $F_{\mathcal{S}}$ before $F_{\mathcal{S}}$ is trained. That is why we assume $\mathcal{A}$ has neither white-box nor black-box access to $F_{\mathcal{S}}$ and knows nothing about its hyperparameters.

3.3 Decision thresholds

It is important to carefully choose an appropriate T that balances robustness against both a malicious $\mathcal{A}$ and a malicious $\mathcal{S}$. For example, a high T makes false claims difficult, but makes it easy for a stolen model to evade detection. Prior work [38, 39] have suggested various ways for choosing T:

- **Independent.** $\mathcal{J}$ trains multiple independent models, obtains their *MORacc* on $(\mathbf{x}, \mathbf{y})$, and chooses the highest one as the decision threshold T. Since independent models tend to have low *MORacc*, the resulting T is also low. A stolen model is likely to have a very high *MORacc*, making it difficult to evade the detection. On the other hand, false claims become easy because it is even possible for an innocent independent model to have a *MORacc* that is higher than T.
- **Extracted.** $\mathcal{J}$ derives multiple extracted models from $F_{\mathcal{A}}$, obtains their *MORacc* on $(\mathbf{x}, \mathbf{y})$, and chooses the lowest one as the decision threshold T. This time, T will be so high that false claims are difficult to succeed. However, this also means that a stolen model can be easily manipulated to have a lower *MORacc* than T, allowing it to evade the detection.
- **Mixed.** $\mathcal{J}$ calculates the average *MORacc* of multiple independent and extracted models This can be considered as a middle-ground approach.

The mixed threshold is the most sensible choice in realistic deployments, but the extracted threshold is the least favourable to a malicious $\mathcal{A}$. Notice that the decision threshold should be uniform for all claims and $\mathcal{J}$ needs to determine it before receiving any claim. Therefore, to determine a decision threshold, $\mathcal{J}$ needs to act as $\mathcal{A}$: trains its own $F_{\mathcal{A}}$, and generates its own trigger set $(\mathbf{x}, \mathbf{y})$ according the MOR scheme it adopts.

4 MOR: Survey

In this section, we survey 16 MOR schemes and describe five well-known schemes under our generalization.

4.1 Taxonomy

Table 2 lists 16 MOR schemes, including 11 watermarking schemes collected from four survey papers [5, 33, 38, 53], and 5 fingerprinting schemes. The watermarking schemes are categorized according to the taxonomy proposed in [38]:

- **Model-Independent.** The watermark is embedded into the model functionality and is independent of the model itself. For example, it could be an independent backdoor embedded into the source model by adding extra samples to the training set.
- **Model-Dependent.** The watermark is embedded into the model functionality and depends on the model. For example, it could also be a backdoor embedded into the source model, but the backdoor is generated based on the source model.
- **Active.** The watermark is embedded into the model functionality during inference, which means that it is only activated when the model is used to make predictions. Therefore, this approach only considers situations where attackers have black-box access to the source model.
- **Parameter-Encoding.** The watermarking schemes in this category are known to be not robust [68]. We refer to Section 9 for more details.

One common characteristic of watermarking schemes is that they require modifications to the source model or its inference process. Such modifications inevitably result in a loss

of model accuracy. In contrast, model **fingerprinting** extracts a unique identifier (i.e., fingerprint) from the source model as its trigger set, without any changes to the model itself. This allows the original model to be preserved, maintaining its accuracy and functionality.

Table 2: MOR schemes.

Category	MOR scheme
Model-independent watermarking	Adi [1] Zhang [72] Li (a) [32] Guo [20] Namba [44] Xu [70]
Model-dependent watermarking	Frontier-Stitching [31] Blackmarks [11] EWE [24] Li (b) [34]
Active watermarking	DAWN [59]
Fingerprinting	Lukas [39] AFA [75] IPGuard [8] Metav [46] DI [43]

4.2 Generalization

All the MOR schemes surveyed in Section 4.1 follow our generalization: the model ownership claim (*moc*) in each scheme includes a trigger set and the claim verification requires running the model inference on the trigger set. Next, for each category, we pick exemplary schemes and describe them under our generalization. We refer to a scheme by its first author's name for ease of presentation, unless it is known under a different name.

4.2.1 Model-independent: Adi [1]

The *moc* of Adi contains a set of out-of-distribution images $\mathbf{x}$. The label y_i for each x_i is randomly sampled over all classes excluding its true label. The source model $F_{\mathcal{A}}$ needs to be fine-tuned on $(\mathbf{x},\mathbf{y})$. If $F_{\mathcal{S}}$ shows a similar behaviour as $F_{\mathcal{A}}$ on $(\mathbf{x},\mathbf{y})$, then $F_{\mathcal{S}}$ is likely derived from $F_{\mathcal{A}}$.

- **claim generation.** $\mathcal{A}$ generates $moc_{\mathcal{A}}$ as follows:
  1. samples out-of-distribution $\mathbf{x}$ and assigns a wrong label y_i to each $x_i \in \mathbf{x}$, i.e., $f(x_i) \neq y_i$ where $f()$ is the ground-truth function;
  2. fine-tunes $F_{\mathcal{A}}$ on $(\mathbf{x},\mathbf{y})$;
  3. commits $(\mathcal{A}, F_{\mathcal{A}}, \mathbf{x}, \mathbf{y})$ as $cm_{\mathcal{A}}$ and timestamps.
- **claim verification.** $\mathcal{J}$ checks the following:
  1. $\frac{1}{|\mathbf{x}|}\sum_{i=1\ldots|\mathbf{x}|}\mathbb{I}(F_{\mathcal{A}}(x_i)=y_i \text{ and } f(x_i)\neq y_i) > T$;
  2. $\frac{1}{|\mathbf{x}|}\sum_{i=1\ldots|\mathbf{x}|}\mathbb{I}(F_{\mathcal{S}}(x_i)=y_i \text{ and } f(x_i)\neq y_i) > T$;
  3. $cm_{\mathcal{A}}$ is a valid commitment of $(\mathcal{A}, F_{\mathcal{A}}, \mathbf{x}, \mathbf{y})$;
  4. $cm_{\mathcal{A}}$ was timestamped before $cm_{\mathcal{S}}$.

$\mathcal{J}$ accepts $\mathcal{A}$'s claim iff all checks pass. The verification function ν could be represented as

$$\nu(F(x_i), y_i) := \mathbb{I}(F(x_i) = y_i \text{ and } f(x_i) \neq y_i). \quad (2)$$

Other model-independent watermarking schemes differ from Adi [1] only in how $(\mathbf{x},\mathbf{y})$ was sampled; all other parts are the same as Adi [1]. In Zhang [72], all samples in $\mathbf{x}$ are from the same class and perturbed with a secret mask, or they are sampled from a different domain unrelated to the source model's domain. In Li (a) [32], all samples in $\mathbf{x}$ are masked with a small filter consists of three colors: image pixels under the white pattern pixels are changed to a very large negative number, image pixels under black pattern pixels are changed to a very large positive number, and pixels under gray pattern pixels stay unchanged. Guo [20] samples $(\mathbf{x},\mathbf{y})$ s.t. $f(x_i) \neq y_i$ $\forall(x_i,y_i) \in (\mathbf{x},\mathbf{y})$, and embeds an n-bit message into each x_i. Namba [44] samples $(\mathbf{x},\mathbf{y})$ s.t. $f(x_i) \neq y_i$ $\forall(x_i,y_i) \in (\mathbf{x},\mathbf{y})$ and imprints them with greater force and cause the model to learn them profoundly. Xu [70] samples $\mathbf{x}$ but labels them with a serial number. Therefore, all such schemes follow our generalization.

4.2.2 Model-dependent: EWE [24]

Recall that $\mathbf{x}$ sampled in Adi [1] are out-of-distribution. The watermarked features learnt by $F_{\mathcal{A}}$ are different from the task distribution and can thus be easily removed through compression or other forms of knowledge transfer. EWE [24] embeds watermarks that are entangled with legitimate data to $F_{\mathcal{A}}$, so that removing such watermarks will sacrifice performance on legitimate data. To this end, EWE uses the soft nearest neighbor loss (SNNL) [27, 55] as an additional loss during training to entangle feature representations of the watermark with the training data. The data samples contained in *moc* are in-distribution samples from the same class $(\mathbf{x},y)$, and these samples will be perturbed by a small mask called trigger.

- **claim generation.** $\mathcal{A}$ generates $moc_{\mathcal{A}}$ as follows:
  1. samples in-distribution $(\mathbf{x},y)$ s.t. $f(x_i)=y$ $\forall x_i \in \mathbf{x}$, for simplicity, we represent it as $f(\mathbf{x})=y$;
  2. samples a trigger t, computes $x'_i := x_i + t$ $\forall x_i \in \mathbf{x}$;[5]
  3. samples y' with $y' \neq y$;
  4. fine-tunes $F_{\mathcal{A}}$ on $(\mathbf{x}',y')$;
  5. commits $(\mathcal{A}, F_{\mathcal{A}}, \mathbf{x}', y')$ as $cm_{\mathcal{A}}$ and timestamps.
- **claim verification.** $\mathcal{J}$ checks the following:
  1. $\frac{1}{|\mathbf{x}'|}\sum_{i=1\ldots|\mathbf{x}'|}\mathbb{I}(F_{\mathcal{A}}(x'_i)=y' \text{ and } f(x'_i)\neq y') > T$;

[5] The trigger is an input mask which can be arbitrarily chosen by $\mathcal{A}$. And the trigger location is determined as the area with the largest gradient of SNNL with respect to x_i.

2. $\frac{1}{|\mathbf{x}'|}\sum_{i=1\ldots|\mathbf{x}'|}\mathbb{I}(F_{\mathcal{S}}(x'_i)=y' \text{ and } f(x'_i)\neq y')>T$;
3. $cm_{\mathcal{A}}$ is a valid commitment of $(\mathcal{A},F_{\mathcal{A}},\mathbf{x}',y')$;
4. $cm_{\mathcal{A}}$ was timestamped before $cm_{\mathcal{S}}$.

$\mathcal{J}$ accepts $\mathcal{A}$'s claim iff all checks pass. The verification function ν is also Equation 2.

4.2.3 Model-dependent: Li (b) [34]

Li (b) aims to prevent watermark forging by enforcing the trigger set to be indistinguishable from the original samples in the training set. To achieve this, it employs an encoder and a discriminator, both trained based on the source model, to construct the trigger set. A malicious $\mathcal{A}$ who does not have access to the encoder cannot generate a trigger set that works well for the victim model. However, as we will show in Section 5.1.3, it is still possible for $\mathcal{A}$ to falsely claim a $F_{\mathcal{S}}$.

- **claim generation.** $\mathcal{A}$ generates $moc_{\mathcal{A}}$ as follows:
  1. samples in-distribution $(\mathbf{x},\mathbf{y})$ s.t. $f(\mathbf{x})=\mathbf{y}$;
  2. generates a triggered $x'_i:\ \forall x_i\in\mathbf{x}$ through the encoder and discriminator;
  3. samples $y_i{}'$ with $y_i{}'\neq y_i,\ \forall y_i\in\mathbf{y}$;
  4. fine-tunes $F_{\mathcal{A}}$ on $(\mathbf{x}',\mathbf{y}')$;
  5. commits $(\mathcal{A},F_{\mathcal{A}},\mathbf{x}',\mathbf{y}')$ as $cm_{\mathcal{A}}$ and timestamps.
- **claim verification.** $\mathcal{J}$ checks the following:
  1. $\frac{1}{|\mathbf{x}'|}\sum_{i=1\ldots|\mathbf{x}'|}\mathbb{I}(F_{\mathcal{A}}(x'_i)=y'_i \text{ and } f(x'_i)\neq y'_i)>T$;
  2. $\frac{1}{|\mathbf{x}'|}\sum_{i=1\ldots|\mathbf{x}'|}\mathbb{I}(F_{\mathcal{S}}(x'_i)=y'_i \text{ and } f(x'_i)\neq y'_i)>T$;
  3. $cm_{\mathcal{A}}$ is a valid commitment of $(\mathcal{A},F_{\mathcal{A}},\mathbf{x}',\mathbf{y}')$;
  4. $cm_{\mathcal{A}}$ was timestamped before $cm_{\mathcal{S}}$.

$\mathcal{J}$ accepts $\mathcal{A}$'s claim iff all checks pass. The verification function ν is also Equation 2.

In Frontier-Stitching [31], $(\mathbf{x},\mathbf{y})$ are generated as adversarial examples w.r.t. $F_{\mathcal{A}}$. Due to transferability, they are also adversarial examples to the independent models of $F_{\mathcal{A}}$. Then, $\mathcal{A}$ updates $F_{\mathcal{A}}$ with $(\mathbf{x},\mathbf{y})$ using adversarial training, so that $F_{\mathcal{A}}$ and the models derived from $F_{\mathcal{A}}$ will perform differently from other independent models on $(\mathbf{x},\mathbf{y})$. Blackmarks [11] is similar to Frontier-Stitching [31] except that all class labels are clustered into two groups; the adversarial examples are generated s.t., for randomly selected samples from one cluster, $F_{\mathcal{A}}$ predicts labels from the other cluster.

Both Frontier-Stitching [31] and Blackmarks [11] follow our generalization: the *moc* includes a trigger set and the claim verification requires running model inference on the trigger set.

4.2.4 Active: DAWN [59]

DAWN is designed to be integrated into the prediction API of a model so that it can survive model extraction. It works by dynamically watermarking a small fraction of client queries, altering the model's prediction responses for these queries. The watermarked queries act as a trigger set in case an adversarial client attempts to train an extracted model using the prediction responses. Using this trigger set, $\mathcal{A}$ can demonstrate ownership as in other DNN watermarking schemes.

- **claim generation.** $\mathcal{A}$ generates $moc_{\mathcal{A}}$ as follows:
  1. samples $(\mathbf{x},\mathbf{y})$ s.t. $F_{\mathcal{A}}(\mathbf{x})=\mathbf{y}$;[6]
  2. samples a model-specific key k;
  3. for each $x_i\in\mathbf{x}$, computes $y'_i:=\pi(\text{HMAC}(k,\mu(x_i)),y_i)$, where $\pi()$ is a keyed pseudorandom permutation that that permutes the label from y_i to y'_i, and $\mu()$ is a mapping function that ensures $\mu(x)=\mu(x+\delta)$ for a small perturbation δ[7];
  4. commits $(\mathcal{A},F_{\mathcal{A}},\mathbf{x},\mathbf{y},\mathbf{y}')$ as $cm_{\mathcal{A}}$ and timestamps.
- **claim verification.** $\mathcal{J}$ checks the following:
  1. $y'_i:=\pi(\text{HMAC}(k,x_i),y_i),\forall x_i\in\mathbf{x}$;[8]
  2. $\frac{1}{|\mathbf{x}|}\sum_{i=1\ldots|\mathbf{x}|}\mathbb{I}(F_{\mathcal{S}}(x_i)=y'_i \text{ and } f(x_i)\neq y'_i)>T$;
  3. $cm_{\mathcal{A}}$ is a valid commitment of $(\mathcal{A},F_{\mathcal{A}},\mathbf{x},\mathbf{y},\mathbf{y}')$;
  4. $cm_{\mathcal{A}}$ was timestamped before $cm_{\mathcal{S}}$.

$\mathcal{J}$ accepts $\mathcal{A}$'s claim iff all checks pass. The verification function ν is also Equation 2.

4.2.5 Fingerprinting: Lukas [39]

Recall that model fingerprinting extracts a persistent, identifying code (i.e., fingerprint) from the source model. Lukas [39] uses the adversarial examples specific to the source model as the fingerprint. They hypothesize that there exists a subclass of targeted, transferable, adversarial examples that transfer only to stolen models but not to independent models. To achieve this, they train a set of independent models and extract a set of surrogate models from the source model. They then find a set of adversarial examples that minimize the loss for $F_{\mathcal{A}}$ and its extracted models but maximize the loss for independent models.

- **claim generation.** $\mathcal{A}$ generates $moc_{\mathcal{A}}$ as follows:
  1. $\mathcal{A}$ trains a set of extracted and independent models:
     - extracted models are trained on data labeled by $F_{\mathcal{A}}$;
     - independent models are trained on ground-truth labels.
  2. samples $(\mathbf{x},\mathbf{y})$ s.t., $f(\mathbf{x})\neq\mathbf{y}$;
  3. for each $x_i\in\mathbf{x}$, perturb it into x'_i s.t., the loss of (x'_i,y_i) is minimized for $F_{\mathcal{A}}$ and its extracted models, but is maximized for the independent models;
  4. commits $(\mathcal{A},F_{\mathcal{A}},\mathbf{x}',\mathbf{y})$ as $cm_{\mathcal{A}}$ and timestamps.

[6] In fact, $\mathbf{x}$ was received from an API client.

[7] Without $\mu()$, a malicious client receiving different predictions for x and $x+\delta$ can discard both x and $x+\delta$ from its training set to avoid the watermark.

[8] The DAWN [59] paper did not explicitly mention this verification, as robustness against malicious $\mathcal{A}$ is not their focus. We add this verification because we want to consider the most difficult condition for false claims.

- **claim verification.** $\mathcal{J}$ checks the following:
  1. $\frac{1}{|\mathbf{x}'|}\sum_{i=1\ldots|\mathbf{x}'|}\mathbb{I}(F_{\mathcal{A}}(x'_i)=y_i \text{ and } f(x'_i)\neq y_i)>T$;
  2. $\frac{1}{|\mathbf{x}'|}\sum_{i=1\ldots|\mathbf{x}'|}\mathbb{I}(F_{\mathcal{S}}(x'_i)=y_i \text{ and } f(x'_i)\neq y_i)>T$;
  3. $cm_{\mathcal{A}}$ is a valid commitment of $(\mathcal{A}, F_{\mathcal{A}}, \mathbf{x}', \mathbf{y})$;
  4. $cm_{\mathcal{A}}$ was timestamped before $cm_{\mathcal{S}}$.

$\mathcal{J}$ accepts $\mathcal{A}$'s claim iff all checks pass. The verification function ν is also Equation 2.

AFA [75] is a concurrent and independent work with Lukas [39]: it also generates the adversarial examples specific to the source model. IPGuard [8] is similar to Lukas [39] except that it uses a different way to find adversarial examples *near* the decision boundary of $F_{\mathcal{A}}$: they start from an initial data point and iteratively move it along the gradient of the objective function. Metav [46] is an extension of Lukas [39]: besides generating $(\mathbf{x}', \mathbf{y})$ in the same way as in [39], it also trains a verifier; the concatenated outputs of $F_{\mathcal{S}}$ on $(\mathbf{x}', \mathbf{y})$ will be input to the verifier, which then determines if $F_{\mathcal{S}}$ is a stolen model. Therefore, both IPGuard [8] and Metav [46] follow our generalization.

4.2.6 Fingerprinting: DI [43]

The key observation for DI is that all models derived from $F_{\mathcal{A}}$ will contain direct or indirect information from the training set of $F_{\mathcal{A}}$, hence model ownership can be resolved by showing that the suspect model was trained (at least partially or indirectly) on the same dataset as the source model. DI measures $F_{\mathcal{S}}$'s prediction margins (distances from the samples to the model's decision boundaries) for both $F_{\mathcal{A}}$'s training set and public samples. If $F_{\mathcal{S}}$ has different prediction margins for them, it is deemed to be stolen; otherwise the model is deemed independent.

- **claim generation.** $\mathcal{A}$ generates $moc_{\mathcal{A}}$ as follows:
  1. samples $\mathbf{x}$ from the training dataset of $F_{\mathcal{A}}$; extracts the feature embeddings for $\mathbf{x}$ that characterizes its "prediction margin" (i.e., distance from the decision boundaries) w.r.t. $F_{\mathcal{A}}$; and labels them as "inside" ($b=1$);
  2. samples $\mathbf{x}'$ from an unseen publicly available dataset; extracts the feature embeddings for $\mathbf{x}'$ w.r.t. $F_{\mathcal{A}}$; and labels them as "outside" ($b=0$);
  3. using the embeddings and the ground truth membership labels (i.e., b), trains a regression model $g_{\mathcal{A}}$, which is to predict a (proxy) measure of confidence that a sample is in the training set of $F_{\mathcal{A}}$.
  4. commits $(\mathcal{A}, F_{\mathcal{A}}, (\mathbf{x},1), (\mathbf{x}',0), g_{\mathcal{A}})$ as $cm_{\mathcal{A}}$ and timestamps.
- **claim verification.** $\mathcal{J}$ checks the following:
  1. extracts the feature embeddings for $\mathbf{x}$ and $\mathbf{x}'$, w.r.t. $F_{\mathcal{A}}$; inputs the embeddings to $g_{\mathcal{A}}$, gets the confidence scores, and checks if there is a significant difference between the two sets of confidence scores;
  2. extracts the feature embeddings for $\mathbf{x}$ and $\mathbf{x}'$, w.r.t. $F_{\mathcal{S}}$; inputs the embeddings to $g_{\mathcal{A}}$, gets the confidence scores, and checks if there is a significant difference between the two sets of confidence scores;
  3. $cm_{\mathcal{A}}$ is a valid commitment of $(\mathcal{A}, F_{\mathcal{A}}, (\mathbf{x},1), (\mathbf{x}',0), g_{\mathcal{A}})$;
  4. $cm_{\mathcal{A}}$ was timestamped before $cm_{\mathcal{S}}$.

$\mathcal{J}$ accepts $\mathcal{A}$'s claim iff all checks pass. DI still follows our generation as it requires running a model on the "trigger set" to generate embeddings. However, it differs from other MOR schemes in that it does not verify each sample separately; instead, it verifies the trigger set as a whole, calculating the effect size or p-value between the confidence scores of $\mathbf{x}$ and $\mathbf{x}'$, and comparing it with a threshold T.

5 Transferable Adversarial Examples Against MOR

Recall that most of the samples in the trigger set $(\mathbf{x}, \mathbf{y})$ satisfy: $\nu(F_{\mathcal{A}}(x_i), y_i) = 1$ for $F_{\mathcal{A}}$ and $\nu(F(x_i), y_i) = 0$ for other independently trained $F \neq F_{\mathcal{A}}$. To falsely claim the ownership of an independent model F, $\mathcal{A}$ could just generate (x_i, y_i) in a way s.t. $\nu(F(x_i), y_i) = 1$ for most $(x_i, y_i) \in (\mathbf{x}, \mathbf{y})$. Our key observation is that $\mathcal{A}$ can achieve this by leveraging the transferability of adversarial examples. Recall that we assume $\mathcal{A}$ has access to a dataset that has the same distribution with $F_{\mathcal{S}}$'s training set. Then, we could have $\mathcal{A}$ train $F_{\mathcal{A}}$ using this dataset and generate a malicious trigger set $\hat{\mathbf{x}}$ as transferable adversarial examples for $F_{\mathcal{A}}$. Then, $\nu(F(\hat{x}_i), y_i) = 1$ for most $(\hat{x}_i, y_i) \in (\hat{\mathbf{x}}, \mathbf{y})$, holds for the independent models of $F_{\mathcal{A}}$.

5.1 Attacks in detail

For each exemplary scheme from Section 4, we show how a malicious $\mathcal{A}$ generates $moc_{\mathcal{A}}$ so that it can falsely claim ownership of an independent model. The underlined text describes $\mathcal{A}$'s misbehaviour. For all attacks, the training sets of $F_{\mathcal{A}}$ and $F_{\mathcal{S}}$ are distinct but follow the same distribution.

5.1.1 Model-independent: Adi [1]

Recall that Adi [1] fine-tunes $F_{\mathcal{A}}$ on $(\mathbf{x}, \mathbf{y})$, where $f(x_i) \neq y_i$ $\forall (x_i, y_i) \in (\mathbf{x}, \mathbf{y})$; models derived from $F_{\mathcal{A}}$ have good performance on $(\mathbf{x}, \mathbf{y})$. $\mathcal{A}$ could generate $(\mathbf{x}, \mathbf{y})$ as adversarial examples so that $F_{\mathcal{A}}$ performs well on $(\mathbf{x}, \mathbf{y})$. Due to transferability, models that are independent of $F_{\mathcal{A}}$ also perform well on $(\mathbf{x}, \mathbf{y})$. Then, $\mathcal{A}$ can claim ownership of these models.

- **claim generation.** $\mathcal{A}$ generates $moc_{\mathcal{A}}$ as follows:
  1. samples $\mathbf{x}$ and assigns a wrong label y_i to each $x_i \in \mathbf{x}$, i.e., $f(\mathbf{x}) \neq \mathbf{y}$;
  2. for each $x_i \in \mathbf{x}$, perturbs it into $\hat{x}_i$ s.t., $F_{\mathcal{A}}(\hat{\mathbf{x}}) = \mathbf{y}$.
  3. commits $(\mathcal{A}, F_{\mathcal{A}}, \hat{\mathbf{x}}, \mathbf{y})$ as $cm_{\mathcal{A}}$ and timestamps.

- **claim verification.** $\mathcal{J}$ checks the following:
  1. $\frac{1}{|\hat{\mathbf{x}}|}\sum_{i=1\ldots|\hat{\mathbf{x}}|}\mathbb{I}(F_{\mathcal{A}}(\hat{x}_i)=y_i \text{ and } f(\hat{x}_i)\neq y_i)>T$;
  2. $\frac{1}{|\hat{\mathbf{x}}|}\sum_{i=1\ldots|\hat{\mathbf{x}}|}\mathbb{I}(F_{\mathcal{S}}(\hat{x}_i)=y_i \text{ and } f(\hat{x}_i)\neq y_i)>T$;
  3. $cm_{\mathcal{A}}$ is a valid commitment of $(\mathcal{A}, F_{\mathcal{A}}, \hat{\mathbf{x}}, \mathbf{y})$;
  4. $cm_{\mathcal{A}}$ was timestamped before $cm_{\mathcal{S}}$.

  The first check holds because $(\hat{\mathbf{x}}, \mathbf{y})$ are adversarial examples generated for $F_{\mathcal{A}}$. The second check holds because $(\hat{\mathbf{x}}, \mathbf{y})$ can transfer to $F_{\mathcal{S}}$.

In fact, the labels $\mathbf{y}$ can be chosen based on the perturbation of $\mathbf{x}$, i.e., untargeted adversarial examples. This makes the adversarial optimization easy to converge.

Other model-independent watermarking schemes differ from Adi [1] only in how $(\mathbf{x}, \mathbf{y})$ was sampled, hence false claims can be done in the same way as in Adi.

5.1.2 Model-dependent: EWE [24]

False claims for EWE [24] are similar to Adi [1], except that $\mathcal{A}$ needs to generate targeted adversarial examples this time. Specifically, in EWE [24], all samples in $\mathbf{x}'$ have the same label y'. Then, $\mathcal{A}$ can no longer determine the label based on the perturbation of $\mathbf{x}$. Instead, it has to perturb each x_i for a target label. Targeted adversarial examples can be generated in the same way, but its transferability becomes weaker. In Section 5.2, we describe a way for enhancing its transferability.

- **claim generation.** $\mathcal{A}$ generates $moc_{\mathcal{A}}$ as follows:
  1. samples $(\mathbf{x}, y)$ with $f(\mathbf{x}) = y$;
  2. samples a trigger t, computes $x'_i := x_i + t,\ \forall x_i \in \mathbf{x}$;
  3. samples y' with $y' \neq y$;
  4. for each $x'_i \in \mathbf{x}'$, perturbs it into $\hat{x}_i$ s.t., $F_{\mathcal{A}}(\hat{\mathbf{x}}) = y'$.
  5. commits $(\mathcal{A}, F_{\mathcal{A}}, \hat{\mathbf{x}}, y')$ as $cm_{\mathcal{A}}$ and timestamps.
- **claim verification.** $\mathcal{J}$ checks the following:
  1. $\frac{1}{|\hat{\mathbf{x}}|}\sum_{i=1\ldots|\hat{\mathbf{x}}|}\mathbb{I}(F_{\mathcal{A}}(\hat{x}_i)=y' \text{ and } f(\hat{x}_i)\neq y')>T$;
  2. $\frac{1}{|\hat{\mathbf{x}}|}\sum_{i=1\ldots|\hat{\mathbf{x}}|}\mathbb{I}(F_{\mathcal{S}}(\hat{x}_i)=y' \text{ and } f(\hat{x}_i)\neq y')>T$;
  3. $cm_{\mathcal{A}}$ is a valid commitment of $(\mathcal{A}, F_{\mathcal{A}}, \hat{\mathbf{x}}, y')$;
  4. $cm_{\mathcal{A}}$ was timestamped before $cm_{\mathcal{S}}$.

  The first check holds because $(\hat{\mathbf{x}}, y')$ are adversarial examples generated for $F_{\mathcal{A}}$. The second check holds because $(\hat{\mathbf{x}}, y')$ can transfer to $F_{\mathcal{S}}$.

5.1.3 Model-dependent: Li (b) [34]

Recall that Li (b) claims that a malicious $\mathcal{A}$ who does not have access to the encoder cannot generate a valid trigger set. However, this claim does not take into account the possibility of transferable adversarial examples. That is, a malicious $\mathcal{A}$ without knowing the encoder can still generate a set of transferable adversarial examples as the trigger set, with the condition that the adversarial examples are within an L_ϵ neighborhood of the original samples to satisfy the indistinguishability requirement between the trigger set and its original samples.

- **claim generation.** $\mathcal{A}$ generates $moc_{\mathcal{A}}$ as follows:
  1. samples in-distribution $(\mathbf{x}, \mathbf{y})$ s.t. $f(\mathbf{x}) = \mathbf{y}$;
  2. generates a triggered x'_i : $\forall x_i \in \mathbf{x}$ through the encoder and discriminator;
  3. samples $y_i{}'$ with $y_i{}' \neq y_i,\ \forall y_i \in \mathbf{y}$;
  4. for each $x'_i \in \mathbf{x}'$, perturbs it into $\hat{x}_i$ s.t., $F_{\mathcal{A}}(\hat{\mathbf{x}}) = \mathbf{y}'$.
  5. commits $(\mathcal{A}, F_{\mathcal{A}}, \hat{\mathbf{x}}, \mathbf{y}')$ as $cm_{\mathcal{A}}$ and timestamps.
- **claim verification.** $\mathcal{J}$ checks the following:
  1. $\frac{1}{|\hat{\mathbf{x}}|}\sum_{i=1\ldots|\hat{\mathbf{x}}|}\mathbb{I}(F_{\mathcal{A}}(\hat{x}_i)=y'_i \text{ and } f(\hat{x}_i)\neq y'_i)>T$;
  2. $\frac{1}{|\hat{\mathbf{x}}|}\sum_{i=1\ldots|\hat{\mathbf{x}}|}\mathbb{I}(F_{\mathcal{S}}(\hat{x}_i)=y'_i \text{ and } f(\hat{x}_i)\neq y'_i)>T$;
  3. $cm_{\mathcal{A}}$ is a valid commitment of $(\mathcal{A}, F_{\mathcal{A}}, \hat{\mathbf{x}}, \mathbf{y}')$;
  4. $cm_{\mathcal{A}}$ was timestamped before $cm_{\mathcal{S}}$.

  The first check holds because $(\hat{\mathbf{x}}, \mathbf{y}')$ are adversarial examples generated for $F_{\mathcal{A}}$. The second check holds because $(\hat{\mathbf{x}}, \mathbf{y}')$ can transfer to $F_{\mathcal{S}}$.

False claims for Frontier-Stitching [31] and Blackmarks [11] are easy: $\mathcal{A}$ can simply generate $(\mathbf{x}, \mathbf{y})$ s.t. $f(\mathbf{x}) = \mathbf{y}$; then all independent models will behave the same as $F_{\mathcal{A}}$.

5.1.4 Active: DAWN [59]

Similar to EWE [24], $\mathcal{A}$ again needs to generate targeted adversarial examples to attack DAWN [59].

- **claim generation.** $\mathcal{A}$ generates $moc_{\mathcal{A}}$ as follows:
  1. samples $(\mathbf{x}, \mathbf{y})$ s.t. $F_{\mathcal{A}}(\mathbf{x}) = \mathbf{y}$;
  2. samples a model-specific key k;
  3. for each $x_i \in \mathbf{x}$, computes $y'_i := \pi(\text{HMAC}(k, \mu(x_i)), y_i)$;
  4. for each $x_i \in \mathbf{x}$, perturbs it into $\hat{x}_i$ s.t., $F_{\mathcal{A}}(\hat{\mathbf{x}}) = \mathbf{y}'$.[9]
  5. commits $(\mathcal{A}, F_{\mathcal{A}}, \hat{\mathbf{x}}, \mathbf{y}, \mathbf{y}')$ as $cm_{\mathcal{A}}$ and timestamps.
- **claim verification.** $\mathcal{J}$ checks the following:
  1. $y'_i := \pi(\text{HMAC}(k, \mu(\hat{x}_i)), y_i), \forall x_i \in \hat{\mathbf{x}}$;
  2. $\frac{1}{|\hat{\mathbf{x}}|}\sum_{i=1\ldots|\hat{\mathbf{x}}|}\mathbb{I}(F_{\mathcal{S}}(\hat{x}_i)=y'_i \text{ and } f(\hat{x}_i)\neq y'_i)>T$;
  3. $cm_{\mathcal{A}}$ is a valid commitment of $(\mathcal{A}, F_{\mathcal{A}}, \hat{\mathbf{x}}, \mathbf{y}, \mathbf{y}')$;
  4. $cm_{\mathcal{A}}$ was timestamped before $cm_{\mathcal{S}}$.

  The first check holds because $\mu(x_i) = \mu(\hat{x}_i)$[10]. The second check holds because $(\hat{\mathbf{x}}, \mathbf{y}')$ can transfer to $F_{\mathcal{S}}$.

Recall that DAWN changes the prediction results of $F_{\mathcal{A}}$ instead of $F_{\mathcal{A}}$ itself. Therefore, a source model $F_{\mathcal{A}}$, trained by an honest accuser, should in principle have a low performance on the committed samples. Based on this observation, we

[9] In principle, $\mathcal{A}$ cannot modify $\mathbf{x}$ as it is supposed to be received from an API client. However, without some additional means of validation, $\mathcal{A}$ can simply claim that $\hat{\mathbf{x}}$ was received from an API client (cf. Section 7).

[10] In our experiments, we took $\mu()$ from the open-sourced implementation of DAWN and it shows that more than 88% of the adversarial examples satisfy $\mu(x_i) = \mu(\hat{x}_i)$. Furthermore, $\mathcal{A}$ can discard the $\hat{x}_i$s that do not satisfy this condition.

could prevent the above attack by introducing an additional check during verification: $\mathcal{J}$ checks the performance of $F_{\mathcal{A}}$ on the committed samples. However, this again can be attacked. Namely, after Step 4, a malicious $\mathcal{A}$ could update $F_{\mathcal{A}}$ with $(\hat{\mathbf{x}}, \mathbf{y})$ in a way like adversarial training.

5.1.5 Fingerprinting: Lukas [39]

Recall that Lukas [39] uses adversarial examples specific to the source model as the fingerprint: they train a set of extracted models and independent models, and perturb the adversarial examples to minimize the loss for the extracted models but maximize the loss for independent models. We could simply omit the part of maximizing loss for independent models, then the adversarial examples will transfer to independent models.

- **claim generation.** $\mathcal{A}$ generates $moc_{\mathcal{A}}$ as follows:
  1. samples $(\mathbf{x}, \mathbf{y})$ s.t., $f(\mathbf{x}) \neq \mathbf{y}$;
  2. for each $x_i \in \mathbf{x}$, perturbs it into $\hat{x}_i$ s.t., $F_{\mathcal{A}}(\hat{\mathbf{x}}) = \mathbf{y}$;
  3. commits $(\mathcal{A}, F_{\mathcal{A}}, \hat{\mathbf{x}}, \mathbf{y})$ as $cm_{\mathcal{A}}$ and timestamps.
- **claim verification.** $\mathcal{J}$ checks the following:
  1. $\frac{1}{|\hat{\mathbf{x}}|} \sum_{i=1\ldots|\hat{\mathbf{x}}|} \mathbb{I}(F_{\mathcal{A}}(\hat{x}_i) = y_i \text{ and } f(\hat{x}_i) \neq y_i) > T$;
  2. $\frac{1}{|\hat{\mathbf{x}}|} \sum_{i=1\ldots|\hat{\mathbf{x}}|} \mathbb{I}(F_{\mathcal{S}}(\hat{x}_i) = y_i \text{ and } f(\hat{x}_i) \neq y_i) > T$;
  3. $cm_{\mathcal{A}}$ is a valid commitment of $(\mathcal{A}, F_{\mathcal{A}}, \hat{\mathbf{x}}, \mathbf{y})$;
  4. $cm_{\mathcal{A}}$ was timestamped before $cm_{\mathcal{S}}$.

  The first check holds because $(\hat{\mathbf{x}}, \mathbf{y})$ are adversarial examples generated for $F_{\mathcal{A}}$. The second check holds because $(\hat{\mathbf{x}}, \mathbf{y})$ can transfer to $F_{\mathcal{S}}$.

Again, $\mathcal{A}$ can choose $\mathbf{y}$ based on the perturbation of $\mathbf{x}$, as untargeted adversarial examples.

AFA [75], IPGuard [8] and Metav [46] can be attacked in the same way as Lukas [39], except that in Metav [46] the adversarial optimization needs to be based on the verifier.

5.1.6 Fingerprinting: DI [43]

Recall that $\mathcal{A}$ in DI needs to include a regression model $g_{\mathcal{A}}$ in $moc_{\mathcal{A}}$. Therefore, to falsely claim a model in DI [43], $\mathcal{A}$ could simply manipulate $g_{\mathcal{A}}$ s.t. it always shows significant differences even for independent $F_{\mathcal{S}}$. Recall that we assume $\mathcal{A}$ has access to a dataset with the same distribution as $F_{\mathcal{S}}$'s training set. We could have $\mathcal{A}$ train $F_{\mathcal{A}}$ with this dataset and "adversarially" perturb each sample in $\mathbf{x}$ with a small amount of noise such that the "prediction margin" (distance between $\mathbf{x}$ and the decision boundaries) w.r.t. $F_{\mathcal{A}}$ are larger. Due to the transferability of our adversarially perturbed samples, $F_{\mathcal{S}}$ will also have a larger "prediction margin" on $\mathbf{x}$. Then if $g_{\mathcal{A}}$ was trained with the embeddings of this perturbed $\mathbf{x}$ and public $\mathbf{x}'$ w.r.t $F_{\mathcal{A}}$, it will output significantly different confidence scores when it takes the embeddings of $\mathbf{x}$ and $\mathbf{x}'$ w.r.t. $F_{\mathcal{S}}$.

- **claim generation.** $\mathcal{A}$ generates $moc_{\mathcal{A}}$ as follows:
  1. samples $\mathbf{x}$ from the training dataset of $F_{\mathcal{A}}$; perturbs $\mathbf{x}$ s.t. the "prediction margin" w.r.t. $F_{\mathcal{A}}$ are larger; extracts the feature embeddings for $\mathbf{x}$ that characterizes this "prediction margin"; and labels them as "inside" ($b = 1$);
  2. samples $\mathbf{x}'$ from an unseen publicly available dataset; extracts the feature embeddings for $\mathbf{x}'$ w.r.t. $F_{\mathcal{A}}$; and labels them as "outside" ($b = 0$);
  3. using the embeddings and the ground truth membership labels (i.e., b), trains a regression model $g_{\mathcal{A}}$, which is to predict a (proxy) measure of confidence that a sample is in the training set of $F_{\mathcal{A}}$;
  4. commits $(\mathcal{A}, F_{\mathcal{A}}, (\mathbf{x}, 1), (\mathbf{x}', 0), g_{\mathcal{A}})$ as $cm_{\mathcal{A}}$ and timestamps.
- **claim verification.** $\mathcal{J}$ checks the following:
  1. extracts the feature embeddings for $\mathbf{x}$ and $\mathbf{x}'$, w.r.t. $F_{\mathcal{A}}$; inputs the embeddings to $g_{\mathcal{A}}$, gets the confidence scores, and checks if there is a significant difference between the two sets of confidence scores;
  2. extracts the feature embeddings for $\mathbf{x}$ and $\mathbf{x}'$, w.r.t. $F_{\mathcal{S}}$; inputs the embeddings to $g_{\mathcal{A}}$, gets the confidence scores, and checks if there is a significant difference between the two sets of confidence scores;
  3. $cm_{\mathcal{A}}$ is a valid commitment of $(\mathcal{A}, F_{\mathcal{A}}, (\mathbf{x}, 1), (\mathbf{x}', 0), g_{\mathcal{A}})$;
  4. $cm_{\mathcal{A}}$ was timestamped before $cm_{\mathcal{S}}$.

  The first check holds because $\mathbf{x}$ are "adversarially" perturbed w.r.t. $F_{\mathcal{A}}$. The second check holds because $\mathbf{x}$ can transfer to $F_{\mathcal{S}}$.

5.2 Transferability enhancement

We borrow an idea of Lukas [39] to enhance the transferability. Recall that $\mathcal{A}$ aims to generate $(\hat{x}, y)$ in a way s.t. $\nu(F(\hat{x}), y) = 1$ for all independent models of $F_{\mathcal{A}}$. We could have $\mathcal{A}$ generate a set of independent models, and perturb the adversarial examples to minimize the loss of these independent models for $\nu(F(\hat{x}), y) = 1$.

To generate an untargeted adversarial example, $\mathcal{A}$ just needs to *maximize* the following loss function:

$$\mathcal{L}(\hat{x}, y) = L(F_{\mathcal{A}}(\hat{x}), y) + \sum_{F \in \mathcal{F}} \beta_F L(F(\hat{x}), y), \tag{3}$$

where y is the true label, $L()$ denotes the cross-entropy loss, $\mathcal{F}$ denotes a set of independent models, β_F is a weight [11] for F. We use Iterative Fast Gradient Sign Method (IFGSM) [30] to find a solution that both maximizes $\mathcal{L}(\hat{x}, y)$ and minimizes the perturbation:

$$\hat{x} := Clip_{\hat{x},\varepsilon}\{\hat{x} + \alpha sign(\Delta_{\hat{x}} \mathcal{L}(\hat{x}, y))\},$$

where $Clip_{\hat{x},\varepsilon}()$ is a function that performs per-pixel clipping for $\hat{x}$. We use $\alpha = 0.03$ and select the number of iteration to

[11] We set $\beta_F = \frac{1}{|F|}$ in our experiments.

be 100 to make the adversarial examples reach the edge of the ε max-norm ball. It ensures that the generated adversarial example $\hat{x}$ will be in $L_\infty\varepsilon$-neighborhood of the original input x. Notice that the per-pixel perturbation bound ε balances the false claim effectiveness and the perturbation visibility.

To generate a targeted adversarial example, $\mathcal{A}$ needs to *minimize* the loss function (i.e., Equation 3), where y becomes the target (wrong) label. The iteration step is similar to that in the untargeted case:

$$\hat{x} := Clip_{\hat{x},\varepsilon}\{\hat{x} - \alpha sign(\Delta_{\hat{x}}\mathcal{L}(\hat{x}, y))\}.$$

This enhancement is generally applicable and we have implemented it for all of our benchmarks in Section 6. We remark that our main contribution is to show that defending against false claims is as challenging as preventing transferable adversarial examples. Any method for generating transferable adversarial examples is pluggable into our attacks. There are many methods for transferability enhancement; it is only necessary to show that our attack works with one such method.

6 Evaluation

We now empirically evaluate our attacks against Adi [1], EWE [24], Li (b) [34], DAWN [59], Lukas [39] and DI [43].

6.1 Setup

For Adi and EWE, we use the Watermark-Robustness-ToolBox[12] to reproduce their results; for Li (b)[13], DAWN[14] and DI[15], we use their open-sourced implementations; as the source code of Lukas[16] is based on TensorFlow (whereas all others are in Pytorch), we re-implemented their scheme in Pytorch [52].

We consider all three kinds of decision thresholds we discussed in Section 3.3. For the extracted models, we use fine-tuning based extraction, a.k.a. Fine-Tune All layers (FTAL) [64][17].

For datasets, we consider CIFAR-10 [29], ImageNet [15], and a face attributes dataset named CelebA [37]. When we measure the *MORacc* on CIFAR-10 and ImageNet, we consider the following three cases:

1. "different structures & different data": $F_\mathcal{A}$ and $F_\mathcal{S}$ use different model structures and training data.
   - We divided CIFAR-10 into two non-overlapping subsets to train $F_\mathcal{A}$ and $F_\mathcal{S}$ respectively. The model structure for $F_\mathcal{A}$ is ResNet 28×10 [71] and it achieves an accuracy of 89.3%; the model structure for $F_\mathcal{S}$ is ResNet-34 [21] and it achieves an accuracy of 86.3%.
   - We randomly selected ten classes from ImageNet to form an ImageNet-10 dataset, and divided ImageNet-10 into two non-overlapping subsets to train $F_\mathcal{A}$ and $F_\mathcal{S}$ respectively. The model structure for $F_\mathcal{A}$ is ResNet-18 [21] and it achieves an accuracy of 82.0%; we use VGG-13 [57] for $F_\mathcal{S}$ and it achieves an accuracy of 85.9%.
2. "same structure & different data": $F_\mathcal{A}$ and $F_\mathcal{S}$ use the same model structure (ResNet 28× 10 for CIFAR-10 and ResNet-18 for ImageNet), but different training data (as described in case 1).
3. "different structures & same data": $F_\mathcal{A}$ and $F_\mathcal{S}$ use the same training data, butdifferent model structures (as described in case 1).

We aim to use CelebA to train a model (i.e., $F_\mathcal{A}$) to falsely claim the model behind Amazon Rekognition API [54], which detects face features of uploaded images. The model structure for $F_\mathcal{A}$ is Resnet-18 [21] and it achieves an accuracy of 97.8%. We have no knowledge about the training dataset or the model architecture of Amazon Rekognition, except that we have black-box access to the model via the API. Therefore, the setting for CelebA is "different structures & different data".

For all three datasets, we set the trigger set size as 100. $F_\mathcal{A}$'s *MORacc* on the trigger sets are near 100% for all MOR schemes except DAWN, which has a *MORacc* of 0 because its $F_\mathcal{A}$ is independent of its trigger sets. For all experiments, we set the per-pixel perturbation bound, (i.e., maximal value allowed for per-pixel perturbation) as 16.

Recall that we use independent models to enhance transferability (cf. Section 5.2). We train 10 independent models for CIFAR-10, 8 independent models for ImageNet, and 2 independent models for CelebA.[18]

All reported results are averages of 5 runs; all reported run times were obtained using (single) Tesla V100 GPUs. We refer to our research report version for more details of our experiments.

[12] https://github.com/dnn-security/Watermark-Robustness-Toolbox

[13] https://github.com/zhenglisec/Blind-Watermark-for-DNN

[14] https://github.com/ssg-research/dawn-dynamic-adversarial-watermarking-of-neural-networks

[15] https://github.com/cleverhans-lab/dataset-inference

[16] https://github.com/ayberkuckun/DNN-Fingerprinting

[17] FTAL comprises two steps: it first queries $F_\mathcal{A}$ with a set of samples and obtains the predicted labels; it then uses this set to fine-tune $F_\mathcal{A}$ with a smaller learning rate. The model extracted through FTAL is very similar to $F_\mathcal{A}$, hence it provides the most challenging threshold for false claims. Indeed, FTAL is the most effective one among several model extraction methods we have tried.

[18] We began with two independent models in each case. With CelebA, two models were enough to get high *MORacc*. On CIFAR-10 and ImageNet, the *MORacc* with two models were not high enough, hence we tried increasing the number of independent models until a sufficiently high *MORacc* was reached: 10 for CIFAR-10 and 8 for ImageNet. However, training models on CIFAR-10 and ImageNet is not as expensive as in CelebA.

Table 3: Decision thresholds (we refer to Section 3.3 for an explanation of the decision thresholds; the column of DI shows the normalized effect size (%) instead of *MORacc*).

Dataset	Threshold type	Adi	EWE	Li (b)	DAWN	Lukas	DI
CIFAR-10	independent	10.0	1.8	23.0	1.0	28.0	90.0
	mixed	29.0	32.9	61.5	38.5	57.5	81.4
	extracted	48.0	64.0	100.0	76.0	87.0	72.8
ImageNet	independent	15.0	12.0	30.0	3.0	14.0	76.5
	mixed	23.5	37.5	65.0	42.5	30.0	69.6
	extracted	32.0	63.0	100.0	82.0	46.0	62.6
CelebA	independent	25.7	3.7	55.0	7.0	21.0	20.0
	mixed	42.4	2.9	55.5	26.0	28.5	14.1
	extracted	59.0	2.0	56.0	45.0	36.0	8.2

Table 4: False claim effectiveness (the bold number means *MORacc* is higher than the mixed threshold and the underlined number means *MORacc* is higher than the "extracted" threshold).

Dataset	Model configurations ($F_{\mathcal{S}}$ vs. $F_{\mathcal{A}}$)	Adi	EWE	Li (b)	DAWN	Lukas	DI
CIFAR-10	different structures & different data	**94.3**	**69.3**	**94.3**	**69.3**	**94.3**	**100**
	same structure & different data	**98.0**	**100.0**	**98.0**	**100.0**	**98.0**	**99.1**
	different structures & same data	**99.0**	**78.3**	**99.0**	**78.3**	**99.0**	**98.6**
ImageNet	different structures & different data	**72.6**	**87.6**	**72.6**	**87.6**	**72.6**	**100**
	same structure & different data	**93.7**	**97.0**	**93.7**	**97.0**	**93.7**	**100**
	different structures & same data	**84.6**	**89.0**	**84.6**	**89.0**	**84.6**	**100**
CelebA	different structures & different data (Amazon Rekognition API)	**68.4**	**68.0**	**68.4**	**68.0**	**68.4**	**99.9**

6.2 False claim effectiveness

Table 3 presents the decision thresholds and Table 4 presents $F_{\mathcal{S}}$'s *MORacc* for all three datasets. It demonstrates that, for a realistic configuration of MORs using the mixed threshold, our false claims can succeed against all MOR schemes on all datasets. Even for the extracted threshold (least favorable to a malicious $\mathcal{A}$), false claims still succeed against all except Li (b). We also show that in scenarios where $F_{\mathcal{A}}$ and $F_{\mathcal{S}}$ are trained with either the same model structure or the same dataset, false claims become much easier.

Recall that the *moc*-generation procedure for Adi, EWE, Li (b), DAWN and Lukas are similar: generating adversarial examples that transfer to $F_{\mathcal{S}}$. The untargeted adversarial examples generated by Adi, Li (b), and Lukas result in identical *MORacc*. In contrast, EWE and DAWN's targeted adversarial examples yield a different *MORacc* value. The distinguishing factor lies in the decision thresholds of each scheme.

Recall that DI calculates the effect size or p-value between the confidence scores of training samples and public samples, and compares it with a threshold, hence the last column in Table 3 and 4 shows the normalized effect size instead of *MORacc* (using normalized effect sizes is equivalent to using the p-values directly). Different from other MOR schemes, DI treats $F_{\mathcal{S}}$ as stolen as long as $F_{\mathcal{S}}$ was trained with the same data as $F_{\mathcal{A}}$, even if it was trained independently. It is possible that the model behind Amazon Rekognition API was trained from CelebA as well, in which case it is not independent from our $F_{\mathcal{A}}$ in terms of DI. Therefore, we tested DI (without our attack) on Amazon Rekognition API and obtained a normalized effect size of 0.0 indicating that there was likely no overlap between CelebA and the training data used for Rekognition API.

Figure 1 shows a comparison between an original image in the trigger sets and its noised version with the per-pixel perturbation bound of 16. It is evident that the original content of the image is clearly perceptible. Table 6 in (Appendix B in our research report) shows the original and our perturbed versions of some images from the same class of CelebA,

and their corresponding confident scores from the Amazon Rekognition API.

We also measure the *MORacc* with different per-pixel perturbation bounds. The results are in Figure 2. The *MORacc* roughly increases with the perturbation bound and becomes stable when the *MORacc* is high enough. When generating untargeted adversarial examples on CIFAR-10, the untargeted label y for all Fs in Equation 3 tend to be the same one that near the true label. As a result, the *MORacc* of untargeted adversarial examples (Adi, Li (b) and Lukas) on CIFAR-10 are higher than the targeted ones (EWE and DAWN). This is not the case for ImageNet, as different Fs have different untargeted labels, leading to a relatively lower *MORacc*. As there are only two labels in CelebA, the targeted and untargeted adversarial examples are essentially the same, hence, their *MORacc* are simliar to each other.

Our attack can succeed even if $F_{\mathcal{A}}$ is of much lower quality than $F_{\mathcal{S}}$. We conducted additional experiments with CIFAR-10 and ImageNet to see what $F_{\mathcal{A}}$ accuracy is needed for a successful attack against an $F_{\mathcal{S}}$:

- On CIFAR-10, $F_{\mathcal{S}}$ achieves 86.3% accuracy; we can break the mix thresholds of all schemes with $F_{\mathcal{A}}$ reaching only 75.0% accuracy; (with $F_{\mathcal{A}}$ reaching 80% accuracy, we can break the extracted thresholds of all schemes except Li(b) and Lukas).
- On ImageNet, $F_{\mathcal{S}}$ achieves 85.9% accuracy; we can break the mix thresholds of all schemes with $F_{\mathcal{A}}$ reaching only 71.3% accuracy; (with $F_{\mathcal{A}}$ reaching 76.3% accuracy, we can break the extracted thresholds of all schemes except Li(b) and DAWN).
- While we do not know the precise accuracy of the model behind the Amazon Rekognition API, it undoubtedly surpasses that of our $F_{\mathcal{A}}$, given its status as a well-known commercial service.

6.3 False claim efficiency

Table 5 displays the time usages in minutes for generating the adversarial trigger sets. Note that a malicious $\mathcal{A}$ needs to run this process only once; therefore the computational cost of our false claims is essentially negligible.

Table 5: Attack runtime in minutes (the reported results does not include the time for training independent models; training an independent model takes 18 min on CIFAR-10, 11 min on ImageNet and 67.5 min on CelebA).

	Adi	EWE	Li (b)	DAWN	Lukas	DI
CIFAR-10	13.6	14.1	13.6	14.1	13.6	13.6
ImageNet	11.1	19.1	11.1	19.1	11.1	11.1
CelebA	8.8	10.7	8.8	10.7	8.8	8.8

7 Countermeasures

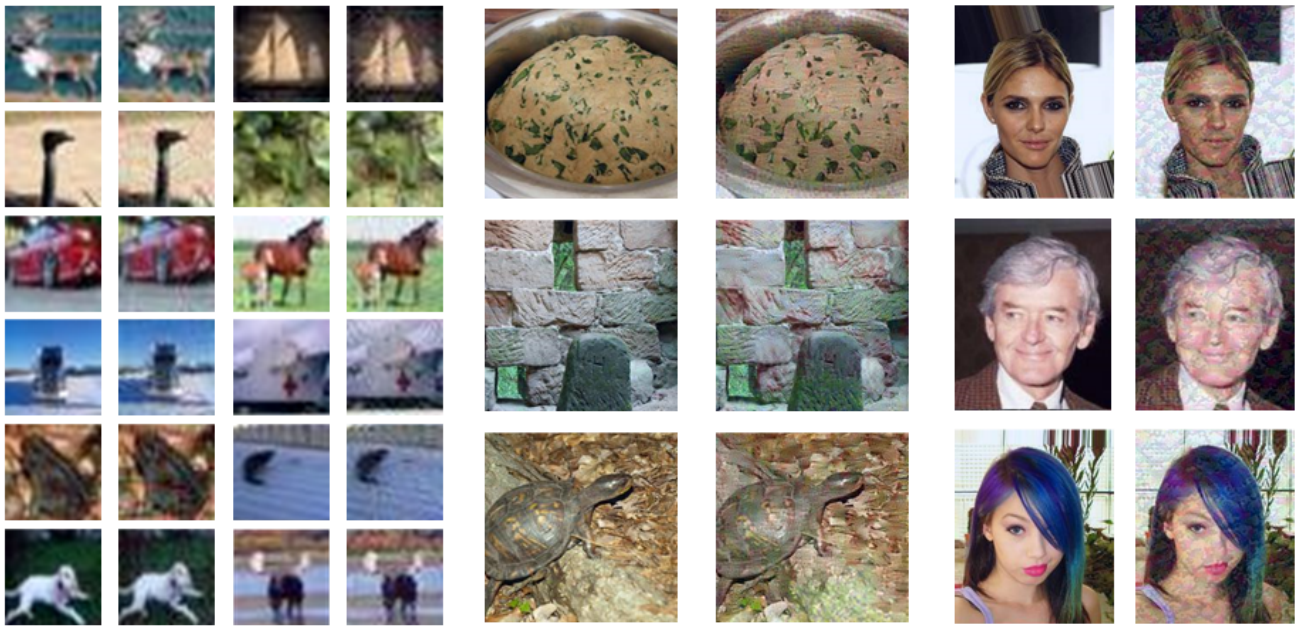
In this section, we present four kinds of countermeasures that have the potential to defend against our false claims. Three of them are generic and can be applied to all MOR schemes, while the fourth is specific to a particular scheme. We also examine the feasibility and potential pitfalls associated with each countermeasure.

$\mathcal{J}$-verified trigger sets. We remark that the root cause of our false claims is the ability of a malicious $\mathcal{A}$ to deviate from the trigger set generation procedure specified by a MOR scheme. Then, a straightforward countermeasure it to have $\mathcal{J}$ verify if the trigger set was generated correctly.

At first glance, achieving this goal seems easy via *verifiable computation* (VC) [3, 51], which enables efficient verification of complex function execution through a VC-proof. Namely, we could have $\mathcal{A}$ use VC to prove that the trigger set was indeed generated by following the MOR scheme. Unfortunately, this is not applicable to watermarking, because its trigger set sampling cannot be captured by the VC-proof. For example, in the claim generation of Adi (cf. Section 4.2.1), $\mathcal{A}$ could generate an adversarial $(\mathbf{x}, \mathbf{y})$ in Step 1) and then generate the VC-proof from Step 2) onwards, allowing the false claim to succeed even with a valid VC-proof. For fingerprinting, VC is applicable, since its trigger sets are generated based on $F_{\mathcal{A}}$. To be secure, the VC-proof in fingerprinting must include the training stage of $F_{\mathcal{A}}$, otherwise, $\mathcal{A}$ could still generate an adversarial trigger set by manipulating the source model. However, including the training stage in VC will be overwhelmingly expensive for the proof generation. We further remark that VC is not applicable to DI due to the fact that *moc*-generation in DI relies not only on $F_{\mathcal{A}}$, but also on its training set, which can be manipulated in advance.

A more effective way to verify the trigger set is to have $\mathcal{J}$ train multiple independent models by itself and run them on the trigger set; if $\mathcal{J}$'s claim verification check deems all such known independent models as stolen, $\mathcal{J}$ can reject $moc_{\mathcal{A}}$ as adversarial. One caveat is that $\mathcal{A}$ could use black-box model extraction to extract these models and generate the trigger sets that can cause $\mathcal{J}$ to deem them as independent; $\mathcal{A}$ could also create black-box adversarial examples directly against these models. While such attacks may be expensive and resource-intensive, it is crucial to take them into consideration. Therefore, $\mathcal{J}$ has to rate-limit and/or raise the cost of dispute requests from $\mathcal{A}$s, which is reasonable since dispute rarely happens.

$\mathcal{J}$-generated trigger sets. Instead of verifying the trigger set, we could have $\mathcal{J}$ generate the trigger set by itself. For model-independent watermarking, $\mathcal{J}$ can simply choose a trigger set and return it to $\mathcal{A}$. For model-dependent watermarking, $\mathcal{A}$ needs to submit $F_{\mathcal{A}}$ to $\mathcal{J}$ first, who can then choose the trigger set based on $F_{\mathcal{A}}$. In either case, $\mathcal{A}$ needs to interact with $\mathcal{J}$ for each deployed model, even if no dispute happens. For fingerprinting, it is enough for $\mathcal{J}$ to get involved and generate



(a) CIFAR-10 (b) ImageNet (c) CelebA

Figure 1: Comparison between original images and the noised versions (the per-pixel perturbation bound is 16; the original images are on the left-hand side and the noised images are on the right-hand side).

the trigger set only when dispute happens. It is worth mentioning that such a countermeasure is not applicable to DI or DAWN. For DI, again, the training dataset can be manipulated so that $moc_{\mathcal{A}}$ can be adversarial even if it is generated by $\mathcal{J}$. For DAWN, the trigger set is generated dynamically based on the queries submitted by the clients.

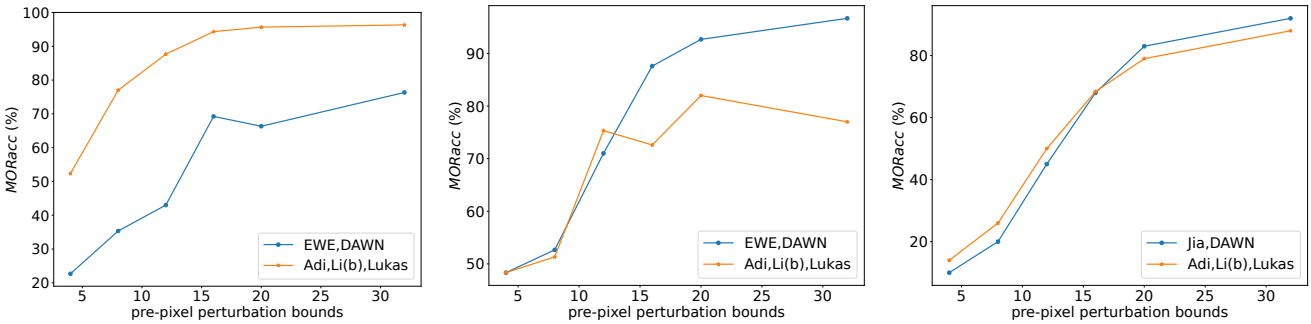
Defending against transferable adversarial examples. Notice that preventing false claims can be reduced to preventing transferable adversarial examples. If we have an effective way for detecting transferable adversarial examples [16, 36, 69], we could have $\mathcal{J}$ use it to determine if the trigger set is adversarial. Similarly, if we have an effective way for adversarial training [6, 19, 56], we could have $\mathcal{S}$ use it to make $F_{\mathcal{S}}$ more robust against transferable adversarial examples. However, how to prevent transferable adversarial examples is still an open problem and it is orthogonal to our paper. On the other hand, any advance in generating transferable adversarial examples will also enhance our false claims.

We evaluate the effectiveness of adversarial training against false claims (untargeted adversarial examples in particular) on CIFAR-10. We use PGD [42] as the adversarial training algorithm and run it for 20 epochs with a perturbation bound $\varepsilon = 16$. *MORacc* decreases from 94.3% to 32.5%. However, the model accuracy also decreases from 86.3% to 83.1%. Therefore, adversarial training can be effective against false claims, but at the cost of a drop in utility. Therefore, adversarial training can be effective against false claims, but at the cost of a drop in utility. Furthermore, it takes 3-30 times longer to train a robust model with adversarial training than training a non-robust equivalent. This longer training time implies a higher cost in computational resources, which is particularly undesirable in the era of large models. However, note that $\mathcal{A}$ can use a large ε to invalidate adversarial training. In some settings, $\mathcal{J}$ can easily detect trigger set entries that are adversarial examples computed using a large ε. How to use adversarial training as a broadly applicable, effective defense against false claim attacks remains an open problem.

Scheme-specific countermeasures. Recall that in DAWN, the trigger set consists of clients' queries; if such queries are signed by clients, $\mathcal{A}$ can no longer perturb them. However, $\mathcal{A}$ could collude with a client to submit perturbed queries. To discourage such collusion, we could have $\mathcal{J}$ punish the client who signed perturbed queries (instead of $\mathcal{S}$) when $F_{\mathcal{S}}$ is deemed stolen. This may not always be realistic. Furthermore, signing each query separately is expensive. We would need to resort to having clients sign aggregate queries (e.g., in the form of a Merkle tree root) periodically.

8 Discussion

False claims for non-timestamped models. As we mentioned in Section 3.1, most MOR schemes do not explicitly require the commitment to be timestamped. This is partially because timestamping is a strong assumption as it requires another trusted third party (or a blockchain) to run the service



(a) CIFAR-10 (b) ImageNet (c) CelebA

Figure 2: *MORacc* with different per-pixel perturbation bounds.

for timestamping. To the best of our knowledge, none of the existing models is timestamped. That means we can falsely claim an existing model without considering timestamping.

If timestamping is not enforced, our false claim becomes much easier, as $\mathcal{A}$ no longer has to create $moc_{\mathcal{A}}$ before $F_{\mathcal{S}}$ comes into existence. That means $\mathcal{A}$ can directly query $F_{\mathcal{S}}$ to create adversarial examples, and such adversarial examples do not need to be transferable. As a result, some of the countermeasures presented in Section 7 are no longer valid. For example, we cannot rely on $\mathcal{J}$ to train independent models to verify the trigger sets, because $\mathcal{A}$ can make the adversarial examples only work for $F_{\mathcal{S}}$ (instead of transferring to other independent models) by leveraging the idea of Lukas [39].

Deferred trigger set generation. Given that most model owners are unlikely to initiate MOR disputes, trigger set generation constitutes an unnecessary upfront cost, especially if we use the $\mathcal{J}$-generated trigger set countermeasure (Section 7). Fortunately, in fingerprinting schemes, the trigger set is generated after $F_{\mathcal{A}}$ being fixed. Therefore, one can *defer* trigger set generation until a MOR dispute arises. In this case, the certified timestamped commitment $cm_{\mathcal{A}}$ covers only $\mathcal{A}, F_{\mathcal{A}}$; $\mathcal{J}$ will need to either generate or verify the trigger set in the event of a dispute.

9 Related Work

Parameter-Encoding. Recall that the main characteristic of our generalization (in Section 3.1) is that $moc_{\mathcal{A}}$ includes a trigger set and its verification requires running model inference on the trigger set. One kind of MOR schemes that do not follow our generalization is parameter-encoding-based watermarking, such as Uchida [64], DeepMarks [10] and DeepSigns [14]. Instead of embedding the watermark into the model functionality, they embed the watermark into the model parameters or the activations of its hidden layers; and their claim verifications require accessing the model parameters. For example, Uchida [64] embeds a message into the weights of some target convolutional layer, by adding an embedding loss during training that regularizes the model and is minimized when the message can be extracted successfully.

Such schemes are known to be *not* robust [38]: one can easily remove the watermark via weight shifting, retraining, or transfer learning [68]. Furthermore, when the suspect model is not white-box accessible, IP infringement cannot be detected.

Passport-based watermarking. A special kind of parameter-encoding-based watermarking embeds a passport into the model to link the model to its owner's identity. For example, Fan et al. [18] introduce a passport layer to regulate the performance of the model: the model owner can use a secret passport to compute the affine factors of the passport layer. They claim that an attacker cannot add a substitute passport while maintain the model performance. However, Chen [12] et al. show that this is feasible, based on the observation that the model can achieve a high accuracy even with different affine factors of the passport layer. This attack can also be considered as a false claim, but they did not consider the situation where timestamped commitment is enforced.

We remark that, to link the model to its owner's identity, the owner can simply add its identity to the (timestamped) commitment, as we did in our generalization (in Section 3.1).

False positives in DI. Szyller et al. [61] showed that DI suffers from false positives when an independent $F_{\mathcal{S}}$ happens to be trained from a dataset with the same distribution as $F_{\mathcal{A}}$'s training set. This explains why DI has very high independent thresholds (cf. Table 4). For example, for an honest trigger set, the normalized effective size of an independent model trained from CIFAR-10 can be as high as 90%.

While Szyller et al. [61] discovered the naturally occurring false positives, we further show that a *malicious* $\mathcal{A}$ can intentionally increase the normalized effective size to 100%.

Proof-of-learning. Jia et al. [25] proposed an alternative MOR scheme dubbed proof-of-learning (PoL), which allows $\mathcal{A}$ to claim ownership of its model by proving integrity of the training procedure. Its $moc_{\mathcal{A}}$ includes a set of intermediate models recorded during training, together with the corresponding data points used to obtain each recorded model. With such

a $moc_{\mathcal{A}}$, $\mathcal{J}$ can replicate the path all the way from the initial model to the final model to be fully confident that $\mathcal{A}$ has indeed performed the computation to obtain the final model.

Nevertheless, as a MOR scheme, PoL has three problems. Firstly, PoL only helps if $F_{\mathcal{A}}$ and $F_{\mathcal{S}}$ are identical, i.e., it is not robust against the thief making minor changes to the stolen model. Secondly, the authors of PoL only claimed that $\mathcal{S}$ cannot generate a valid $moc_{\mathcal{A}}$ with a lower cost than that made by $\mathcal{A}$, i.e., it is not robust if $\mathcal{S}$ wishes to pay that much cost to generate $moc_{\mathcal{A}}$. Lastly, Zhang et al. [74] have shown that, even with less cost than that made by $\mathcal{A}$, $\mathcal{S}$ can still generate a valid $moc_{\mathcal{A}}$.

10 Conclusion

There has been a steady stream of works that have demonstrated that model theft is a real concern and MOR schemes are needed. The recent flurry of works that showed how large language models can be inexpensively bootstrapped off of prominent, expensive, models like GPT-4 have underscored this problem.

In this paper, we described a systematic shortcoming in all existing MOR schemes that permits false accusations to succeed, impacting the robustness, and hence the trustworthiness, of those schemes. Although we outlined several possible generic countermeasures, they can incur significant costs. Developing an efficient generic countermeasure, or identifying effective countermeasures specific to individual MOR schemes remain open problems.

Furthermore, we assume that the malicious accuser has access to a surrogate dataset that is in the same distribution as the suspect model's training set, which may preclude false claims on valuable models trained on rare or private datasets. Evaluating the feasibility of false claims without this assumption is left for future work.

Acknowledgments

This work is supported in part by National Key Research and Development Program of China (2023YFB2704000), National Natural Science Foundation of China (U20A20222), Hangzhou Leading Innovation and Entrepreneurship Team (TD2020003), China Scholarship Council (CSC), Intel (in the context of the Private AI consortium), the Government of Ontario, and the Academy of Finland (decision 339514). Views expressed in the paper are those of the authors and do not necessarily reflect the position of the funders. We thank the anonymous reviewers, the program chair, and the shepherd for their constructive feedback and support. We also thank the authors of prior MOR schemes discussed here, who read our paper and provided valuable feedback.

References

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium*, pages 1615–1631, 2018.

[2] Buse Gul Atli, Sebastian Szyller, Mika Juuti, Samuel Marchal, and N Asokan. Extraction of complex dnn models: Real threat or boogeyman? In *Engineering Dependable and Secure Machine Learning Systems: Third International Workshop*, pages 42–57, 2020.

[3] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *23rd USENIX Security Symposium*, pages 781–796, 2014.

[4] Arjun Nitin Bhagoji, Daniel Cullina, Chawin Sitawarin, and Prateek Mittal. Enhancing robustness of machine learning systems via data transformations. In *Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5, 2018.

[5] Franziska Boenisch. A systematic review on model watermarking for neural networks. *Frontiers in big Data*, 4:729663, 2021.

[6] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

[7] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018.

[8] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *ACM Asia Conference on Computer and Communications Security*, pages 14–25, 2021.

[9] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic extraction of neural network models. In *40th Annual International Cryptology Conference*, pages 189–218, 2020.

[10] Huili Chen, Bita Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *International Conference on Multimedia Retrieval*, pages 105–113, 2019.

[11] Huili Chen, Bita Darvish Rouhani, and Farinaz Koushanfar. Blackmarks: Blackbox multibit watermarking for deep neural networks. *arXiv preprint arXiv:1904.00344*, 2019.

[12] Yiming Chen, Jinyu Tian, Xiangyu Chen, and Jiantao Zhou. Effective ambiguity attack against passport-based dnn intellectual property protection schemes through fully connected layer substitution. In *2023 IEEE Conference on Computer Vision and Pattern Recognition*, 2023.

[13] Jacson Rodrigues Correia-Silva, Rodrigo F Berriel, Claudine Badue, Alberto F de Souza, and Thiago Oliveira-Santos. Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[14] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 485–497, 2019.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009.

[16] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[17] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. In *NeurIPS*, 2019.

[18] Lixin Fan, Kam Woh Ng, Chee Seng Chan, and Qiang Yang. Deepipr: Deep neural network ownership verification with passports. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:6122–6139, 2021.

[19] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[20] Jia Guo and Miodrag Potkonjak. Watermarking deep neural networks for embedded systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2018.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing links from graph neural networks. In *USENIX Security Symposium*, pages 2669–2686, 2021.

[23] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *29th USENIX Security Symposium*, 2020.

[24] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. In *30th USENIX Security Symposium*, pages 1937–1954, 2021.

[25] Hengrui Jia, Mohammad Yaghini, Christopher A Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. Proof-of-learning: Definitions and practice. In *IEEE Symposium on Security and Privacy*, pages 1039–1056, 2021.

[26] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N. Asokan. PRADA: protecting against DNN model stealing attacks. In *IEEE European Symposium on Security & Privacy*, pages 1–16, 2019.

[27] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[28] Kalpesh Krishna, Gaurav Singh Tomar, Ankur Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves of sesame street: Model extraction on bert-based apis. In *International Conference on Learning Representations*, 2020.

[29] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[30] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

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

[32] Huiying Li, Emily Wenger, Shawn Shan, Ben Y Zhao, and Haitao Zheng. Piracy resistant watermarks for deep neural networks. *arXiv preprint arXiv:1910.01226*, 2019.

[33] Yue Li, Hongxia Wang, and Mauro Barni. A survey of deep neural network watermarking techniques. *Neurocomputing*, 461:171–193, 2021.

[34] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of dnn. In *Annual Computer Security Applications Conference*, page 126–137, 2019.

[35] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[36] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations*, 2016.

[37] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision*, 2015.

[38] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. Sok: How robust is image classification deep neural network watermarking? In *43rd IEEE Symposium on Security and Privacy*, pages 787–804, 2022.

[39] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. In *International Conference on Learning Representations*, 2020.

[40] Hengliang Luo, Yi Yang, Bei Tong, Fuchao Wu, and Bin Fan. Traffic sign recognition using a multi-task convolutional neural network. *IEEE Transactions on Intelligent Transportation Systems*, 19:1100–1111, 2017.

[41] Yan Luo, Xavier Boix, Gemma Roig, Tomaso Poggio, and Qi Zhao. Foveation-based mechanisms alleviate adversarial examples. *arXiv preprint arXiv:1511.06292*, 2015.

[42] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[43] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset inference: Ownership resolution in machine learning. In *International Conference on Learning Representations*, 2020.

[44] Ryota Namba and Jun Sakuma. Robust watermarking of neural network with exponential weighting. In *ACM Asia Conference on Computer and Communications Security*, page 228–240, 2019.

[45] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4954–4963, 2019.

[46] Xudong Pan, Yifan Yan, Mi Zhang, and Min Yang. Metav: A meta-verifier approach to task-agnostic model fingerprinting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1327–1336, 2022.

[47] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[48] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ACM Asia conference on computer and communications security*, pages 506–519, 2017.

[49] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[50] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Association*, volume 1, 2015.

[51] B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252, May 2013.

[52] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[53] Francesco Regazzoni, Paolo Palmieri, Fethulah Smailbegovic, Rosario Cammarota, and Ilia Polian. Protecting artificial intelligence ips: a survey of watermarking and fingerprinting for machine learning. *CAAI Transactions on Intelligence Technology*, 6(2):180–191, 2021.

[54] Amazon Rekognition. Amazon rekognition. https://aws.amazon.com/rekognition/.

[55] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial intelligence and statistics*, pages 412–419. PMLR, 2007.

[56] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32, 2019.

[57] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, May 2015.

[58] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[59] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. Dawn: Dynamic adversarial watermarking of neural networks. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4417–4425, 2021.

[60] Sebastian Szyller, Vasisht Duddu, Tommi Gröndahl, and N Asokan. Good artists copy, great artists steal: Model extraction attacks against image translation generative adversarial networks. *arXiv preprint arXiv:2104.12623*, 2021.

[61] Sebastian Szyller, Rui Zhang, Jian Liu, and N Asokan. On the robustness of dataset inference. *Transactions on Machine Learning Research*, 2023.

[62] Tatsuya Takemura, Naoto Yanai, and Toru Fujiwara. Model extraction attacks on recurrent neural networks. *Journal of Information Processing*, 28:1010–1024, 2020.

[63] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th USENIX Security Symposium*, pages 601–618, 2016.

[64] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, pages 269–277, 2017.

[65] Eric Wallace, Mitchell Stern, and Dawn Song. Imitation attacks and defenses for black-box machine translation systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 5531–5546, 2020.

[66] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5265–5274, 2018.

[67] Qinglong Wang, Wenbo Guo, Kaixuan Zhang, Alexander G Ororbia II, Xinyu Xing, Xue Liu, and C Lee Giles. Learning adversary-resistant deep neural networks. *arXiv preprint arXiv:1612.01401*, 2016.

[68] Tianhao Wang and Florian Kerschbaum. Attacks on digital watermarks for deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2622–2626, 2019.

[69] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2730–2739, 2019.

[70] XiangRui Xu, YaQin Li, and Cao Yuan. A novel method for identifying the deep neural network model with the serial number. *arXiv preprint arXiv:1911.08053*, 2019.

[71] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

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

[73] Jianpeng Zhang, Yutong Xie, Qi Wu, and Yong Xia. Medical image classification using synergic deep learning. *Medical image analysis*, 54:10–19, 2019.

[74] Rui Zhang, Jian Liu, Yuan Ding, Zhibo Wang, Qingbiao Wu, and Kui Ren. "adversarial examples" for proof-of-learning. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1408–1422, 2022.

[75] Jingjing Zhao, Qingyue Hu, Gaoyang Liu, Xiaoqiang Ma, Fei Chen, and Mohammad Mehedi Hassan. Afa: Adversarial fingerprinting authentication for deep neural networks. *Comput. Commun.*, 150(C):488–497, 2020.

[76] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 4480–4488, 2016.