Probabilistically Robust Watermarking of Neural Networks

Mikhail Pautov[1,2,3], **Nikita Bogdanov**[2], **Stanislav Pyatkin**[2], **Oleg Rogov**[1,2] and **Ivan Oseledets**[1,2]
[1]Artificial Intelligence Research Institute, Moscow, Russia
[2]Skolkovo Institute of Science and Technology, Moscow, Russia
[3]ISP RAS Research Center for Trusted Artificial Intelligence, Moscow, Russia
{mikhail.pautov, nikita.bogdanov, stanislav.pyatkin}@skoltech.ru, {rogov, oseledets}@airi.net

Abstract

As deep learning (DL) models are widely and effectively used in Machine Learning as a Service (MLaaS) platforms, there is a rapidly growing interest in DL watermarking techniques that can be used to confirm the ownership of a particular model. Unfortunately, these methods usually produce watermarks susceptible to model stealing attacks. In our research, we introduce a novel trigger set-based watermarking approach that demonstrates resilience against functionality stealing attacks, particularly those involving extraction and distillation. Our approach does not require additional model training and can be applied to any model architecture. The key idea of our method is to compute the trigger set, which is transferable between the source model and the set of proxy models with a high probability. In our experimental study, we show that if the probability of the set being transferable is reasonably high, it can be effectively used for ownership verification of the stolen model. We evaluate our method on multiple benchmarks and show that our approach outperforms current state-of-the-art watermarking techniques in all considered experimental setups.

1 Introduction

Deep learning models achieved tremendous success in practical problems from different areas, such as computer vision [He *et al.*, 2016; Dosovitskiy *et al.*, 2021], natural language processing [Vaswani *et al.*, 2017; Brown *et al.*, 2020] and multimodal learning [Tang *et al.*, 2022]. They are used in medical diagnostics [He *et al.*, 2023; Goncharov *et al.*, 2021], deployed in autonomous vehicles [Huang *et al.*, 2022; Parekh *et al.*, 2022] and embedded in AI-as-a-service settings [Buhalis and Moldavska, 2022; Liu *et al.*, 2023]. Unfortunately, the development, training, and production of these models nowadays come at a high cost due to the availability and quality of the training data, the large size of the models, and, hence, the necessity of cloud computing and data storage platforms. This motivates the owners to prevent third parties from obtaining an illegal copy of their models, acquiring the powerful tools without spending much time and money on development and training.

Among the methods to protect the copyright of digital assets, digital watermarking techniques [Hartung and Kutter, 1999] are the most widely used. To determine the copyright violation, the intellectual property owner embeds the special information in the product, for example, adding an imperceptible pattern or digital signature to the image or within the program's source code. If the violation is suspected, this information may be extracted from the intellectual property, confirming the illegal obtaining of the latter. In recent years, watermarking techniques have been adapted to protect the ownership of deep learning models embedded in the black-box manner. To do so, the owner of the model may prepare the special (trigger) set of points the source model should have the specific predictions on: the more similar the predictions of a suspicious on this set to the prespecified ones, the more likely it is that the source model has been compromised [Zhang *et al.*, 2018b; Adi *et al.*, 2018; Bansal *et al.*, 2022; Kim *et al.*, 2023].

In practice, watermarks are not resistant to attacks that are aimed to steal the model's functionality. In particular, distillation attacks, fine-tuning, and regularization of models tend to affect the transferability of trigger sets [Shafieinejad *et al.*, 2021]. Thus, researchers are motivated to explore the robustness of watermarks to stealing attacks.

In this paper, we propose a novel framework that enhances the resistance of trigger set-based watermarks to stealing attacks. Given the source model f, we construct a parametric set $\mathcal{B}_{\delta,\tau}(f)$ of proxy models which imitate the set of surrogate (or stolen) copies of f. We assume that, given the parametric set of proxy models, there exist input data points $\mathcal{S}(f, \delta, \tau)$ that all the models assign to the same class. If the behavior of the source model on these points is prespecified, they are treated as good trigger points for ownership verification. In our method, we ensure that all the proxy models assign a particular sample from the trigger set to some specific class by comparing the predictions of m randomly sampled proxy models (see Figure 1).

We summarize the contributions of this work as follows:

- We introduce a novel probabilistic approach for enhancing trigger-set-based watermarking methods' robustness against stealing attacks. Our approach can be applied to enhance the robustness of any trigger set-based water-

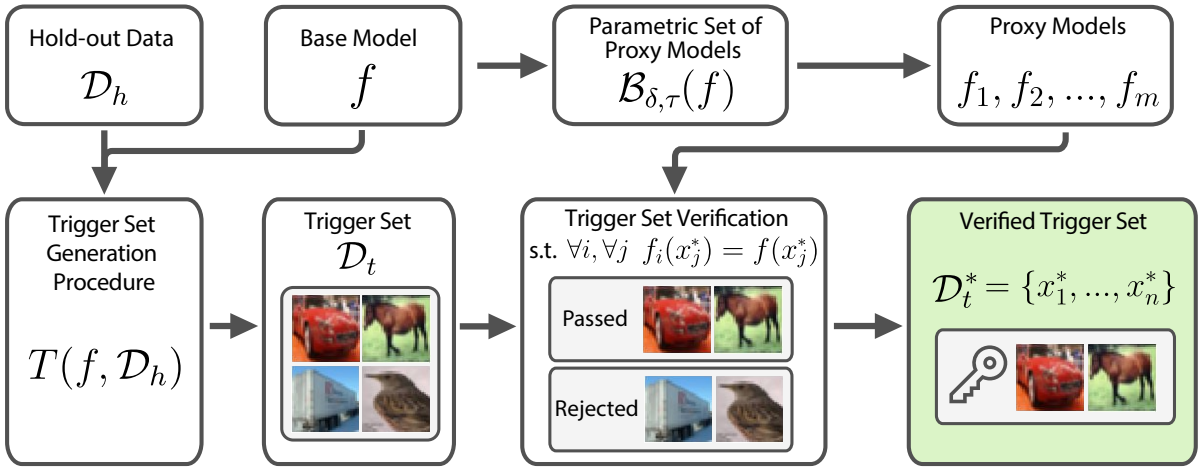

Figure 1: The illustration of the proposed pipeline for the trigger set generation and verification. Given the source model f and the hold-out data $\mathcal{D}_h$, we initialize the parametric set of proxy models $\mathcal{B}_{\delta,\tau}(f)$ introduced in Equation (4) and sample m proxy models $f_1, \ldots, f_m$ from this set. Then, given the procedure of trigger set generation $T = T(f, \mathcal{D}_h)$, we compute the trigger set candidates $\mathcal{D}_t$. The samples from the candidate set $\mathcal{D}_t$ that are verified by the proxy models $f_1, \ldots, f_m$ are included in the verified trigger set $\mathcal{D}_t^*$. The procedure is executed until the verified trigger set of size n is collected.

marking technique, which makes it universal.

- We analyze the probability that a given trigger set is transferable to the set of proxy models that mimic the stolen models.
- We experimentally show that, even if the stolen model does not belong to the set of proxy models, the trigger set is still transferable to the stolen model.
- We evaluate our approach on multiple benchmarks and show that it outperforms current methods in all considered experimental setups.

2 Related Work

The process of DNN watermarking serves the purpose of safeguarding intellectual property by encoding a distinctive pattern and employing it for the purpose of asserting ownership [Subramanian *et al.*, 2021; Li *et al.*, 2021].

Certain watermarking techniques insert digital media watermarks into the initial training data to create a trigger dataset for the model. For example, [Guo and Potkonjak, 2018] generates an n-bit signature representing the model owner and embeds it into the training data to create the trigger dataset. The authors ensure that the altered images in the trigger dataset receive unique labels different from the original data points.

In [Zhang *et al.*, 2018b], the authors proposed algorithms for watermarking neural networks used in image classification, along with remote black-box verification methods. One technique involves adding meaningful content alongside the original training data to create a watermark. For example, they embed a unique string, like a company name, into an image from the training set during the prediction process, assigning a different label to the modified sample. Alternatively, noise can be added to the original training data as part of the watermarking process.

A similar approach suggested in [Li *et al.*, 2019] features blending the regular data samples with distinctive "logo" elements and training the model to categorize them under a specific label. To maintain similarity with the original samples, they use an autoencoder, and its discriminator is trained to distinguish between benign training samples and watermark-containing trigger samples.

In contrast, there are studies on certain disadvantages of trigger set-based watermarking methods.

First, this category of methods has a limitation tied to the maximum number of backdoors that can be integrated into a neural network. There are works showing that the large number of watermarked samples in the training set leads to notable performance degradation of the source model [Jia *et al.*, 2021; Kim *et al.*, 2023].

Secondly, watermarking schemes lacking a verifiable connection between the watermark and the legitimate model owner create an opportunity for attackers to counterfeit the watermark [Adi *et al.*, 2018; Guo and Potkonjak, 2018]. Lastly, the use of adversarial examples for trigger set-based watermarking [Le Merrer *et al.*, 2020] or fingerprinting [Lukas *et al.*, 2021; Zhao *et al.*, 2020] has significant drawbacks, including insufficient transferability to the surrogate models [Kim *et al.*, 2023], potential vulnerability to knowledge distillation attacks [Hinton *et al.*, 2015], fine-tuning and retraining.

3 Problem Statement

3.1 Trigger Set-Based Digital Watermarking

In our work, we consider classification problem with K classes. Namely, given the dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathbb{R}^d$ and $y_i \in [1, \ldots, K]$, we train the source model f to minimize the empirical risk

$$L(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} l(f(x_i), y_i), \tag{1}$$

where $l(\cdot,\cdot)$ is the cross-entropy loss.

If the source model performs well, an adversary can try to steal its functionality. Namely, one can train surrogate model f^* on surrogate dataset $\hat{\mathcal{D}}$ that aims to imitate the outputs of the base model. In general, the architectures of the source model and the surrogate one do not have to be the same. It is equivalently not required that the initial dataset is known, which makes it possible to steal the models in the black-box manner [Jia *et al.*, 2021; Kim *et al.*, 2023].

To detect theft, the owner of the source model can apply trigger set-based watermarking. For example, the subset $\mathcal{D}_s = \{(x_{i_k}, y_{i_k})\}_{k=1}^{n}$ of the initial dataset $\mathcal{D}$ undergoes label flipping: each label y_{i_k} is replaced with another label $y'_{i_k} \neq y_{i_k}$ yielding the trigger set $\mathcal{D}_t = \{(x_{i_k}, y'_{i_k})\}_{k=1}^{n}$. Then, the source model is trained to minimize the empirical risk on the changed dataset $\mathcal{D} := (\mathcal{D} \setminus \mathcal{D}_s) \cup \mathcal{D}_t$.

If the performance of suspicious model f^* on the trigger set $\mathcal{D}_t$ is similar to the one of the source model f, it is claimed that the source model is stolen.

Unfortunately, trigger set-based watermarking approaches have two significant drawbacks. Firstly, the size n of the trigger set has to be small to not cause a notable performance decrease. On the other hand, n should be sufficiently large so that the similarity in the behavior of the source model and the stolen model is statistically significant. Secondly, trigger sets tend to be barely transferable between the source model and the stolen model: a sample from a trigger set is pushed away from the samples of the same class closer to the decision boundary of the surrogate model [Kim *et al.*, 2023].

In this work, we propose a simple and effective approach to generate the trigger set $\mathcal{D}_t$, which is transferable between the source model and its surrogate copies obtained by model stealing attacks.

3.2 Model Stealing Attacks

In our work, we assume that an adversary attempts to steal the source model by applying knowledge distillation[1] [Hinton *et al.*, 2015; Tramèr *et al.*, 2016] considering it the strongest attack for the watermark removal [Kim *et al.*, 2023].

In particular, a sample $\hat{x}_i$ from the surrogate dataset $\hat{\mathcal{D}}$ is passed to the source model f to obtain its prediction $f(\hat{x}_i)$. Then, the stealing is performed by training a new model f^* by minimizing the divergence between its predictions and the source model's predictions on the dataset $\hat{\mathcal{D}}$:

$$L_{\text{ext}}(\hat{\mathcal{D}}) = \frac{1}{|\hat{\mathcal{D}}|}\sum_{\hat{x}_i \in \hat{\mathcal{D}}} D_{\text{KL}}(f(\hat{x}_i), f^*(\hat{x}_i)), \tag{2}$$

where D_{KL} is Kullback–Leibler divergence.

We consider both soft-label and hard-label attacks, i.e., $f^*(\hat{x}_i)$ can be either the predicted class or vector of class probabilities.

[1] We include additional experiments with other stealing attacks in the supplementary material.

Algorithm 1 Trigger set candidate

Input: Hold-out dataset $\mathcal{D}_h$, source model f
Output: Trigger set candidate (x^*, y^*)

while True **do**
  Sample $(x_1, y_1), (x_2, y_2) \sim \mathcal{U}(\mathcal{D}_h)$
  if $y_1 \neq y_2$ **then**
    Sample $\lambda \sim \mathcal{U}(0, 1)$
    $x^* = \lambda x_1 + (1 - \lambda)x_2$
    $y^* = f(x^*)$
    if $y^* \neq y_1$ and $y^* \neq y_2$ **then**
      return (x^*, y^*)
    end if
  end if
end while

4 Method

4.1 Computing the Trigger Set

In our approach, we exploit the procedure of computing the candidates for the trigger set $\mathcal{D}_t$ as convex combinations of the pairs of points from the hold-out dataset [Zhang *et al.*, 2018a]. Namely, suppose the source model f is trained on the dataset $\mathcal{D}$. Then, given the hold-out test data $\mathcal{D}_h : \mathcal{D}_h \cap \mathcal{D} = \emptyset$, we uniformly sample a pair of points $(x_{i_1}, y_{i_1}), (x_{i_2}, y_{i_2})$ from different classes y_{i_1} and $y_{i_2} \neq y_{i_1}$, and compute the convex combination of x_{i_1} and x_{i_2} in the form

$$x_i^* = \lambda x_{i_1} + (1 - \lambda)x_{i_2}, \tag{3}$$

where $\lambda \sim \mathcal{U}(0, 1)$. To assure the unexpected behaviour of the model f on the trigger set candidate, we accept x_i^* as the candidate only if the source model predicts x_i^* as the sample from some other class $y_i^* : y_i^* \neq y_{i_1}$ and $y_i^* \neq y_{i_2}$. The procedure of computing the candidates for the trigger set is presented in Algorithm 1. We execute Algorithm 1 until the candidate set $\mathcal{D}_t = \{(x_i^*, y_i^*)\}_{i=1}^{n}$ is computed. Note that the described procedure requires no additional training of the source model.

4.2 Verification of the Trigger Set

The core idea of our method is to ensure that the trigger set is transferable to the stolen models or, in other words, to verify that the predictions of a stolen trigger set are similar to the ones of the source model. To do so, we introduce the parametric set of models $\mathcal{B}_{\delta,\tau}(f)$ that mimics the set of stolen models. In our experiments, we mainly consider the case when the architecture of the source model is known to a potential adversary. Hence, this parametric set consists of proxy models f' of the same architecture as the source f that perform reasonably well on the training dataset $\mathcal{D}$. Namely, if $\theta(f)$ is the flattened vector of weights of the model f, the parametric set $\mathcal{B}_{\delta,\tau}(f)$ is defined as follows:

$$\begin{aligned}\mathcal{B}_{\delta,\tau}(f) = \{f' : \|\theta(f') - \theta(f)\|_2 \leq \delta \text{ and} \\ |\text{acc}(\mathcal{D}, f') - \text{acc}(\mathcal{D}, f)| \leq \tau\},\end{aligned} \tag{4}$$

Algorithm 2 Trigger set verification

Input: Hold-out dataset $\mathcal{D}_h$, source model f, weights threshold δ, performance threshold τ, verified trigger set size n
Output: Verified trigger set $\mathcal{D}_t^*$

```
Initialize B_{δ,τ}(f)
Sample f_1, ..., f_m ~ B_{δ,τ}(f)
Let i = 0
Let D_t^* = ∅
while i < n do
    (x_i^*, y_i^*) ← TriggerSetCandidate(D_h, f)
    if f_1(x_i^*) = f_2(x_i^*) = ··· = f_m(x_i^*) = y_i^* then
        D_t^* ← D_t^* ∪ {(x_i^*, y_i^*)}
        i ← i + 1
    end if
end while
return D_t^*
```

where $\text{acc}(\mathcal{D}, f)$ is the accuracy of model f on the dataset $\mathcal{D}$, δ is the weights threshold and τ is the performance threshold.

To verify the transferability of the trigger set, we sample m proxy models $f_1, \ldots, f_m$ from $\mathcal{B}_{\delta,\tau}(f)$. Then, we check if all m proxy models assign the same class label to the samples of the trigger set as the source model f. The procedure of trigger set verification is presented in Algorithm 2. The method of computing the verified trigger set is illustrated in Figure 1.

Remark. *In our experiments, we sample proxy model $f_i \sim \mathcal{B}_{\delta,\tau}(f)$ by generating a noise vector $\Delta_i \sim \mathcal{N}(0, \sigma^2 I)$ for some $\sigma^2 > 0$ and assuring $\|\Delta_i\|_2 \leq \delta$. Then, the vector of weights of proxy model f_i is computed as $\theta(f_i) \leftarrow \theta(f) + \Delta_i$.*

5 Experiments

5.1 Setup of Experiments

Datasets and Training

In our experiments, we use CIFAR-10 and CIFAR-100 [Krizhevsky *et al.*, 2009] as training datasets for our source model f. For the purpose of comparison, as the source model, we use ResNet34 [He *et al.*, 2016], which is trained for 100 epochs to achieve high classification accuracy (namely, 91.0% for CIFAR-10 and 66.7% for CIFAR-100). We used SGD optimizer with learning rate of 0.1, weight decay of 0.5×10^{-3} and momentum of 0.9.

Parametric Set of Proxy Models

Once the source model is trained, we initialize a parametric set of proxy models $\mathcal{B}_{\delta,\tau}(f)$. In our experiments, we vary the parameters of the proxy models set to achieve better trigger set accuracy of our approach. Namely, parameter δ was varied in the range $[0.5, 40]$ and τ was chosen from the set $\{0.1, 0.2, 1.0\}$. We tested different number of proxy models sampled from $\mathcal{B}_{\delta,\tau}(f)$ for verification. Namely, parameter m was chosen from the set $\{1, 2, 4, 8, 16, 32, 64, 128, 256\}$.

Model Stealing Attacks

Following the other works [Jia *et al.*, 2021; Kim *et al.*, 2023], we perform functionality stealing attack by training the surrogate model f^* in the following three settings:

- Soft-label attack. In this setting, the training dataset $\mathcal{D}$ is known, and, given input x, the output $f(x)$ of the source model is a vector of class probabilities. The surrogate model f^* is trained to minimize the functional from Eq. (2).
- Hard-label attack. In this setting, the training dataset $\mathcal{D}$ is known, and, given input x, the output $f(x)$ of the source model is the class label assigned by f to input x. This setting corresponds to the training of the surrogate model on the dataset $\hat{\mathcal{D}} = \{x_i, f(x_i)\}_{i=1}^N$.
- Regularization with ground truth label. In [Kim *et al.*, 2023], it was proposed to train the surrogate model by minimizing the empirical loss on the training dataset $\mathcal{D}$ and the KL-divergence between the outputs of the source model and surrogate model simultaneously. This setting corresponds to the minimization of the convex combination of the losses from Eq. (1) and Eq. (2) in the form

$$L_{\text{RGT}}(\mathcal{D}, \hat{\mathcal{D}}, \gamma) = \gamma L_{\text{ext}}(\hat{\mathcal{D}}) + (1 - \gamma) L(\mathcal{D}), \qquad (5)$$

where $\gamma \in [0, 1]$ is the regularization coefficient. In our experiments, this is the strongest functionality stealing attack.

Concurrent Works

We evaluate our approach against the following methods.

- Entangled Watermark Embedding (EWE). In [Jia *et al.*, 2021], it was proposed to embed watermarks by forcing the source model to entangle representations for legitimate task data and watermarks.
- Randomized Smoothing for Watermarks (RS). In [Bansal *et al.*, 2022], randomized smoothing is applied to the parameters of the source model, yielding guarantees that watermarks can not be removed by a small change in the model's parameters.
- Margin-based Watermarking (MB). In [Kim *et al.*, 2023], it was proposed to train the surrogate model by pushing the decision boundary away from the samples from the trigger set so that their predicted labels can not change without compromising the accuracy of the source model.

It is worth mentioning that all the baselines we compare our approach against either require modification of the training procedure of the source model or make its inference computationally expensive.

Evaluation Protocol

Once the verified trigger set $\mathcal{D}_t^* = \{(x_i^*, y_i^*)\}_{i=1}^n$ is collected and surrogate model f^* is obtained, we measure the accuracy

$$\text{acc}(\mathcal{D}_t^*, f^*) = \frac{1}{|\mathcal{D}_t^*|} \sum_{(x_i^*, y_i^*) \in \mathcal{D}_t^*} \mathbb{1}\left(f^*(x_i^*) = y_i^*\right) \qquad (6)$$

of f^* on $\mathcal{D}_t^*$ to evaluate the effectiveness of our watermarking approach.

Remark. *Later, to compare our approach with concurrent works, we denote the trigger set as $\mathcal{D}^*$ to emphasize the differences in trigger set collection procedures.*

$\mathcal{D}$	$\hat{\mathcal{D}}$	Attack	Δ
CIFAR-10	CIFAR-10	Soft	44.65 ± 0.04
		Hard	54.79 ± 1.10
		RGT	49.61 ± 0.36
CIFAR-100	CIFAR-100	Soft	82.03 ± 1.91
		Hard	82.21 ± 3.52
		RGT	82.29 ± 2.92
CIFAR-10	SVHN	Soft	46.33 ± 0.07

Table 1: The l_2-norm of the difference Δ of models' parameters between the source model f and surrogate models f^* for different types of stealing attacks. The architecture of the source model and surrogate models is ResNet34.

Parameters of Experiments

Unless stated otherwise, we use the following values of hyperparameters in our experiments: the size of the verified trigger set n is set to be $n = 100$ for consistency with the concurrent works, confidence level α for Clopper-Pearson test from Eq. (9) is set to be $\alpha = 0.05$. In our experiments, we found that better transferability of the verified trigger set is achieved when no constraint on the performance of the proxy models is applied, so the performance threshold parameter is set to be $\tau = 1.0$.

5.2 Results of Experiments

Knowledge Required for Model Stealing

In our experiments, we assume either the architecture of the source model or its training dataset is *known* to a potential adversary. In Table 2, we present the results for the most aggressive stealing setting, i.e., the one where the adversary is aware of both the architecture and the training data of the source model. In Table 3, we report the results for the setting where either architecture or the training dataset is *unknown* to the adversary. Namely, following [Kim *et al.*, 2023], we (i) perform a model stealing attack using the surrogate dataset SVHN [Netzer *et al.*, 2011] and (ii) perform stealing attack by replacing the architecture of the surrogate model by VGG11 [Simonyan and Zisserman, 2015].

For each experiment, we train a single instance of the source model f and perform $N_{\text{st}} = 10$ independent model stealing attacks. For our approach and concurrent works, we report the accuracy of the source model f and the surrogate models f^* on training dataset $\mathcal{D}$ and trigger set $\mathcal{D}^*$. It is notable that our approach not only outperforms the baselines in terms of trigger set accuracy but also yields the source model with higher accuracy.

Hyperparameters Tuning

It is important to mention that the parameters of proxy set $\mathcal{B}_{\delta,\tau}(f)$ affect not only the transferability of the trigger set but also the computation time needed to collect the trigger set. Indeed, the larger the value of δ is, the more proxy models are used for verification of the trigger set, the more often the procedure from Algorithm 2 rejects trigger set candidates. To find the trade-off between the accuracy of the surrogate model on the trigger set and the computation time, we perform hyperparameter tuning. We tune parameters m and δ according to Section 5.1.

According to parameters tuning, we choose $m = 64$ and $\delta = 40.0$ as the default parameters of the proxy set. In Table 4, we report the values of parameters we used in each experiment. It is noteworthy that tuning only parameters of the proxy set $\mathcal{B}_{\delta,\tau}(f)$ allows our method to surpass existing approaches by a notable margin.

6 Discussions

6.1 Transferability of the Verified Trigger Set

In our approach, we assume that all the models from the parametric set $\mathcal{B}_{\delta,\tau}(f)$ are agreed in predictions on data samples from unknown *common set* $\mathcal{S}(f, \delta, \tau)$. In other words, if $f(x)$ is the class assigned by model f to sample x, the set $\mathcal{S}(f, \delta, \tau)$ is defined as follows:

$$\mathcal{S}(f, \delta, \tau) = \{x : f(x) = f^{'}(x)\ \forall f^{'} \in \mathcal{B}_{\delta,\tau}(f)\}. \quad (7)$$

If the stolen model belongs to the set of proxy models $\mathcal{B}_{\delta,\tau}(f)$, a trigger set build-up from points from common set $\mathcal{S}(f, \delta, \tau)$ would be a good candidate for ownership verification: by design, the predictions of the source model and the stolen model would be identical on such a set.

Since it is impossible to guarantee that a certain data point belongs to the common set $\mathcal{S}(f, \delta, \tau)$, we perform the screening of the input space for the candidates to belong to $\mathcal{S}(f, \delta, \tau)$.

Namely, given a candidate x, we check the agreement in predictions of m randomly sampled proxy models $f_1, \ldots, f_m$ from $\mathcal{B}_{\delta,\tau}(f)$ and accept x as the potential member of $\mathcal{S}(f, \delta, \tau)$ only if all m models have the same prediction. One can think of the selection process of such points as tossing a coin: checking the predictions of m proxy models represents m coin tosses. The input data points represent unfair coins, i.e., those with different probabilities of landing on heads and tails. If the input point x and the index of proxy model i is fixed, such an experiment $A_i = A_i(x)$ is a Bernoulli trial:

$$A_i(x) = \begin{cases} 1 & \text{with probability } p(x), \\ 0 & \text{with probability } 1 - p(x). \end{cases} \quad (8)$$

Let the success of the Bernoulli trial from Eq. (8) correspond to the agreement in predictions of the source model f and $i-$th proxy model f_i. Thus, the screening reduces to the search of input points with the highest probability $p(x)$.

In our experiments, we estimate the parameter $p(x)$ of the corresponding random variable by observing the results of m experiments $A_1(x), \ldots, A_m(x)$. We use interval estimation for $p(x)$ in the form of Clopper-Pearson test [Clopper and Pearson, 1934] that returns one-sided $(1 - \alpha)$ confidence interval for $p(x)$:

$$\mathbb{P}\left(p(x) \geq B\left(\frac{\alpha}{2}, t, m - t + 1\right)\right) \geq 1 - \alpha. \quad (9)$$

In Eq. 9, $\hat{p}(x) = B\left(\frac{\alpha}{2}, m, 1\right)$ is the quantile from the Beta distribution and the number of successes $t = m$. The follow-

Method	Metric	Source model f	Surrogate models f^*		
			Soft-label	Hard-label	RGT
EWE [Jia *et al.*, 2021]	CIFAR-10 acc. (%)	86.10 ± 0.54	83.97 ± 1.02	82.22 ± 0.50	88.88 ± 0.35
RS [Bansal *et al.*, 2022]		84.17 ± 1.01	88.93 ± 1.18	89.62 ± 0.97	90.14 ± 0.08
MB [Kim *et al.*, 2023]		87.81 ± 0.76	91.17 ± 0.76	91.88 ± 0.40	93.05 ± 0.20
Probabilistic (Ours)		$\mathbf{91.00 \pm 0.00}$	$\mathbf{92.60 \pm 0.91}$	$\mathbf{94.87 \pm 0.59}$	$\mathbf{99.42 \pm 0.02}$
EWE [Jia *et al.*, 2021]	Trigger set acc. (%)	26.88 ± 8.32	51.01 ± 5.58	36.05 ± 6.48	1.64 ± 1.05
RS [Bansal *et al.*, 2022]		95.67 ± 4.93	7.67 ± 4.04	6.33 ± 1.15	3.00 ± 0.00
MB [Kim *et al.*, 2023]		100.00 ± 0.00	82.00 ± 1.00	51.33 ± 4.93	72.67 ± 6.66
Probabilistic (Ours)		100.00 ± 0.00	$\mathbf{85.10 \pm 6.33}$	$\mathbf{73.70 \pm 4.65}$	$\mathbf{78.00 \pm 5.58}$
EWE [Jia *et al.*, 2021]	CIFAR-100 acc. (%)	55.11 ± 1.67	53.00 ± 1.57	46.78 ± 1.00	63.73 ± 0.40
RS [Bansal *et al.*, 2022]		59.87 ± 2.78	65.66 ± 1.53	65.79 ± 0.39	64.99 ± 0.30
MB [Kim *et al.*, 2023]		62.13 ± 4.36	$\mathbf{67.66 \pm 0.36}$	$\mathbf{70.65 \pm 0.49}$	$\mathbf{70.24 \pm 0.46}$
Probabilistic (Ours)		$\mathbf{66.70 \pm 0.00}$	67.49 ± 0.03	68.05 ± 0.73	67.85 ± 0.04
EWE [Jia *et al.*, 2021]	Trigger set acc. (%)	68.14 ± 10.16	30.90 ± 11.34	15.10 ± 5.64	5.73 ± 3.42
RS [Bansal *et al.*, 2022]		99.00 ± 1.00	2.67 ± 1.53	4.33 ± 4.16	2.00 ± 1.00
MB [Kim *et al.*, 2023]		100.00 ± 0.00	70.67 ± 7.57	40.00 ± 8.89	62.66 ± 10.12
Probabilistic (Ours)		100.00 ± 0.00	$\mathbf{78.80 \pm 2.93}$	$\mathbf{74.70 \pm 3.16}$	$\mathbf{79.10 \pm 2.77}$

Table 2: Watermarking performance is reported against functionality stealing methods. The best performance is highlighted in bold. It can be seen that our approach outperforms the other methods of ownership verification by a notable margin.

Method	f^*	$\hat{\mathcal{D}}$	$\text{acc}(\mathcal{D}, f)$	$\text{acc}(\mathcal{D}^*, f)$	$\text{acc}(\mathcal{D}, f^*)$	$\text{acc}(\mathcal{D}^*, f^*)$
MB [Kim *et al.*, 2023]	ResNet34	SVHN	87.81 ± 0.76	100.0 ± 0.00	63.99 ± 3.90	72.00 ± 6.08
	VGG11	CIFAR-10	87.81 ± 0.76	100.0 ± 0.00	86.00 ± 2.17	32.00 ± 7.21
Probabilistic (ours)	ResNet34	SVHN	91.00 ± 0.00	100.0 ± 0.00	73.01 ± 1.18	77.70 ± 2.90
	VGG11	CIFAR-10	91.00 ± 0.00	100.0 ± 0.00	89.24 ± 2.69	80.10 ± 3.86

Table 3: Results of watermarking approaches in the setting when either the training dataset or source model's architecture is unknown to the adversary. Our approach outperforms the baseline in terms of the initial accuracy of the source model and the trigger set accuracy of surrogate models.

ing Lemma gives probabilistic guarantees on the transferability of predictions on the verified trigger set from the source model to a proxy model from the set $\mathcal{B}_{\delta,\tau}(f)$.

Lemma 1. *Given the sampling procedure for proxy models from Section 4.2, the confidence level α from Eq.* (9)*, with probability at least $\phi = (1-\alpha)^n$, the expectation of accuracy of the proxy model $f_i \sim \mathcal{B}_{\delta,\tau}(f)$ on the verified trigger set $\mathcal{D}_t^*$ of size n is at least $acc(\mathcal{D}_t^*, f^*) = \hat{p}(x)$.*

Proof. Note that with probability at least $\phi = (1-\alpha)^n$ the interval estimations for $p(x)$ from Eq. (9) hold for all the n samples from $\mathcal{D}_t^*$. Fixing the proxy model $f_i \sim \mathcal{B}_{\delta,\tau}(f)$, we can compute its accuracy $\text{acc}(\mathcal{D}_t^*, f^*)$ on the verified trigger set:

$$\text{acc}(\mathcal{D}_t^*, f^*) = \frac{1}{n} \sum_{(x_j, y_j) \in \mathcal{D}_t^*} A_i(x_j), \tag{10}$$

where $A_i(x_j)$ is in the form from Eq. (8). Taking expectation of Eq. (10) yields

$$\begin{aligned} \mathbb{E}\left(\text{acc}(\mathcal{D}_t^*, f^*)\right) &= \frac{1}{n} \sum_{(x_j, y_j) \in \mathcal{D}_t^*} \mathbb{E}(A_i(x_j)) \\ &= \frac{1}{n} \sum_{(x_j, y_j) \in \mathcal{D}_t^*} p(x_j) \geq \frac{1}{n} \sum_{(x_j, y_j) \in \mathcal{D}_t^*} \hat{p}(x_j) \equiv \hat{p}(x). \end{aligned} \tag{11}$$

□

Remark. *If the distributions of the proxy models and the surrogate models on $\mathcal{B}_{\delta,\tau}(f)$ are the same, the Lemma above yields probabilistic guarantees on the transferability of the predictions on the trigger set to surrogate models.*

We treat $\hat{p}(x)$ as the lower bound of $p(x)$; the higher the value of $\hat{p}(x)$, the higher the probability that a certain sample x belongs to the common set $\mathcal{S}(f, \delta, \tau)$, given the finite number of proxy models used for verification.

In the majority of our experiments, we use $m = 64$ proxy models for the verification of the trigger set. It yields a uniform lower bound $\hat{p}(x) \geq 0.9$ from Eq. 9 for all the samples x that are included in the verified trigger set.It is notable that such a moderate lower bound in practice leads to high transferability of the trigger set to the surrogate models.

Surrogate model f^*	Training dataset $\mathcal{D}$	Surrogate dataset $\hat{\mathcal{D}}$	Stealing method	Parameter m	Parameter δ
ResNet34	CIFAR-10	CIFAR-10	Soft-label	64	40.0
			Hard-label	64	40.0
			RGT	256	40.0
		SVHN	Soft-label	256	40.0
	CIFAR-100	CIFAR-100	Soft-label	64	40.0
			Hard-label	64	40.0
			RGT	64	40.0
VGG11	CIFAR-10	CIFAR-10	Plain training	64	40.0

Table 4: Values of hyperparameters used in each experiment. For all the experiments, the size n of the trigger set and performance threshold τ are $n = 100$ and $\tau = 1.0$. For the RGT stealing method, the regularization coefficient from Eq. (5) is set to be $\gamma = 0.3$.

However, according to our experimental settings, surrogate models f^* *do not have to* belong to the proxy set $\mathcal{B}_{\delta,\tau}(f)$ due to plausible difference in architectures or large difference in weights. Hence, it is not guaranteed that surrogate models have the same common set $\mathcal{S}(f, \delta, \tau)$ as proxy ones. In Table 1, we report the norm of the difference of models' parameters between the source model and surrogate models. It is noteworthy that the surrogate models do not belong to the proxy set $\mathcal{B}_{\delta,\tau}(f)$. Despite this, our approach yields trigger sets that, in practice, are transferable beyond the proxy set.

6.2 Integrity of the Method

It should be mentioned that a watermarking approach not only should not affect the source model's performance and be robust to stealing attacks, it should also satisfy the property of integrity. In other words, it should not judge non-watermarked networks as watermarked ones. In trigger set-based watermarking settings, checking if a model is stolen may be thought of as a detection problem with certain false positive and false negative rates. The first one corresponds to the probability that a benign model is detected as stolen, and the second one corresponds to the probability that a stolen model is not detected as such.

Assuming that a stolen model belongs to the parametric set of proxy models $\mathcal{B}_{\delta,\tau}(f)$, it is possible to provide probabilistic guarantees that the one would be detected as stolen by our method. In contrast, it is, in general, nontrivial to guarantee that a benign model would not be detected as stolen. With our method, such guarantees may be provided under certain modifications of the verification procedure. Namely, one may assume that all the models that belong to the compliment $\bar{\mathcal{B}}_{\delta,\tau}(f)$ of the set of proxy models $\mathcal{B}_{\delta,\tau}(f)$ *are not stolen* ones. Then, the verification procedure may be adapted: given models $f_1, \ldots, f_m \in \mathcal{B}_{\delta,\tau}(f)$ and models $\bar{f}_1, \ldots, \bar{f}_m \in \bar{\mathcal{B}}_{\delta,\tau}(f)$, the sample (x^*, y^*) is verified iff:

$$\begin{cases} y^* = f_1(x^*) = \cdots = f_m(x^*), \\ y^* \neq \bar{f}_1(x^*), \\ \ldots \\ y^* \neq \bar{f}_m(x^*). \end{cases} \tag{12}$$

In other words, it is also required that the models from $\bar{\mathcal{B}}_{\delta,\tau}(f)$ are *not* agreed with the source model on the samples from trigger set. It is notable that such a verification procedure requires careful parameterization of the set of proxy models: underestimation of its parameters would lead to some stolen models not being included in it, and overestimation of its parameters may lead to the inclusion of the benign models.

Experiments on the Integrity of the Method

To satisfy the integrity property, the method has to be able to distinguish between stolen models and independent (not stolen) models g. To check this property, we evaluate our approach on independent models g, which were trained in several different setups:

- In the first setting, we train $N_i = 64$ models on random subsets of initial training dataset. Architectures of independent models are either ResNet34 or WideResNet28 [Zagoruyko and Komodakis, 2016]. For each model, the training dataset is twice as small as the dataset of the source model. We report the results of this setting in Table 5.
- In the second setting, we evaluate models of different architectures trained on the whole CIFAR-10 dataset. We consider VGG [Simonyan and Zisserman, 2015], ShuffleNetV2 [Ma *et al.*, 2018], ResNet20, ResNet32, ResNet44, ResNet56 [He *et al.*, 2016], RepVGG [Ding *et al.*, 2021] and MobileNetV2 [Sandler *et al.*, 2018] architectures. We report the results of this setting in Table 6.
- In the third setting, we train $N_i = 3$ models of different architectures on different dataset, $\mathcal{D}$ = SVHN [Netzer *et al.*, 2011]. All the models were trained for 30 epochs with SGD optimizer, learning rate of 0.1, and weight decay of 10^{-4}. We report the results of this setting in Table 7.

It is expected that if the model is independent, then it is less similar to the source model and, hence, its accuracy on the trigger set is lower than that of the source model and stolen models. As the results, we report the accuracy of independent models on respective training datasets $\mathcal{D}$ and accuracy on watermarks $\mathcal{D}_t^*$.

g	acc$(\mathcal{D}, f)$	acc$(\mathcal{D}, g)$	acc$(\mathcal{D}_t^*, g)$
ResNet34	90.6 ± 0.4	77.9 ± 2.6	55.7 ± 7.2
WideResNet28		92.3 ± 0.3	53.9 ± 5.0

Table 5: Results of experiments on the integrity of the method, part 1. The training dataset $\mathcal{D}$ is CIFAR-10.

g	acc$(\mathcal{D}, f)$	acc$(\mathcal{D}, g)$	acc$(\mathcal{D}_t^*, g)$
VGG	90.6 ± 0.4	93.7 ± 0.5	58.2 ± 3.9
ShuffleNetV2		92.6 ± 1.5	57.3 ± 2.7
ResNet		93.6 ± 0.7	53.9 ± 5.0
RepVGG		94.8 ± 0.3	53.3 ± 2.4
MobileNetV2		93.6 ± 0.5	57.0 ± 2.2

Table 6: Results of experiments on the integrity of the method, part 2. The training dataset $\mathcal{D}$ is CIFAR-10.

Notably, the independent models from the setups considered have different degrees of similarity with the source model. Regardless of architecture, the model trained on the same dataset as the source model partially retains the behavior of the source model on the watermarks. However, trigger set accuracy is significantly lower than that of surrogate (stolen) models. Our experiments show that the least similar independent models are the ones trained on the different datasets, regardless of architecture.

The key advantage of our method is that treating the stealing attack as a random process, we obtain the probabilistic guarantees on the transferability of the predictions on the trigger set from the source model to the stolen one utilizing random (proxy) models from $\mathcal{B}_{\tau,\delta}(f)$ and $\bar{\mathcal{B}}_{\tau,\delta}(f)$. To sum up, we want to outline that our model can catch the difference between the stolen model and independent models; the larger the dissimilarity between the independent model and the source model, the greater the difference in behavior on the trigger set.

g	acc$(\mathcal{D}, g)$	acc$(\mathcal{D}_t^*, g)$
ResNet34	84.0 ± 7.5	14.0 ± 3.2
MobileNetV2	77.0 ± 6.5	12.7 ± 2.9
VGG11	83.1 ± 1.8	13.3 ± 1.5

Table 7: Results of experiments on the integrity of the method, part 3. The training dataset $\mathcal{D}$ is SVHN.

7 Conclusion

In this paper, we propose a novel trigger set-based watermarking approach to address intellectual property protection in the context of black-box model stealing attacks. Our method produces trigger sets that are transferable between the source model and the surrogate models with high probability. Our approach is model-agnostic, does not require any additional model training, and does not imply any limitations on the size of the trigger set. Thus, it can be applied to any model without causing performance sacrifice and minimal computational overhead for trigger set generation. We evaluate our approach on multiple benchmarks against the concurrent methods and show that our method outperforms state-of-the-art watermarking techniques in all considered experimental setups. Future work includes analysis of the guarantees of transferability of digital watermarks and study of provable integrity for certified ownership verification.

Acknowledgements

The authors would like to thank Anastasia Antsiferova from the video group of MSU Graphics and Media Laboratory, Nikita Kotelevskii and Ekaterina Kuzmina from Skolkovo Institute of Science and Technology for useful discussions during the preparation of this paper. This work was partially supported by a grant for research centers in the field of artificial intelligence, provided by the Analytical Center in accordance with the subsidy agreement (agreement identifier 000000D730321P5Q0002) and the agreement with the Ivannikov Institute for System Programming of the Russian Academy of Sciences dated November 2, 2021 No. 70-2021-00142.

References

[Adi *et al.*, 2018] 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 (USENIX Security 18)*, pages 1615–1631, 2018.

[Bansal *et al.*, 2022] Arpit Bansal, Ping-yeh Chiang, Michael J Curry, Rajiv Jain, Curtis Wigington, Varun Manjunatha, John P Dickerson, and Tom Goldstein. Certified neural network watermarks with randomized smoothing. In *International Conference on Machine Learning*, pages 1450–1465. PMLR, 2022.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[Buhalis and Moldavska, 2022] Dimitrios Buhalis and Iuliia Moldavska. Voice assistants in hospitality: using artificial intelligence for customer service. *Journal of Hospitality and Tourism Technology*, 13(3):386–403, 2022.

[Clopper and Pearson, 1934] Charles J Clopper and Egon S Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

[Ding *et al.*, 2021] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13733–13742, 2021.

[Dosovitskiy *et al.*, 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai,

Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[Goncharov *et al.*, 2021] Mikhail Goncharov, Maxim Pisov, Alexey Shevtsov, Boris Shirokikh, Anvar Kurmukov, Ivan Blokhin, Valeria Chernina, Alexander Solovev, Victor Gombolevskiy, Sergey Morozov, et al. Ct-based covid-19 triage: Deep multitask learning improves joint identification and severity quantification. *Medical image analysis*, 71:102054, 2021.

[Guo and Potkonjak, 2018] Jia Guo and Miodrag Potkonjak. Watermarking deep neural networks for embedded systems. In Iris Bahar, editor, *Proceedings of the International Conference on Computer-Aided Design - ICCAD '18*, pages 1–8. ACM Press, 2018.

[Hartung and Kutter, 1999] Frank Hartung and Martin Kutter. Multimedia watermarking techniques. *Proceedings of the IEEE*, 87(7):1079–1107, 1999.

[He *et al.*, 2016] 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.

[He *et al.*, 2023] Kelei He, Chen Gan, Zhuoyuan Li, Islem Rekik, Zihao Yin, Wen Ji, Yang Gao, Qian Wang, Junfeng Zhang, and Dinggang Shen. Transformers in medical image analysis. *Intelligent Medicine*, 3(1):59–78, 2023.

[Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[Huang *et al.*, 2022] Yanjun Huang, Jiatong Du, Ziru Yang, Zewei Zhou, Lin Zhang, and Hong Chen. A survey on trajectory-prediction methods for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 7(3):652–674, 2022.

[Jia *et al.*, 2021] Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1937–1954, 2021.

[Kim *et al.*, 2023] Byungjoo Kim, Suyoung Lee, Seanie Lee, Sooel Son, and Sung Ju Hwang. Margin-based neural network watermarking. In *International Conference on Machine Learning*, pages 16696–16711. PMLR, 2023.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

[Li *et al.*, 2019] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to prove your model belongs to you. In David Balenson, editor, *Proceedings of the 35th Annual Computer Security Applications Conference on - ACSAC '19*, pages 126–137. ACM Press, 2019.

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

[Liu *et al.*, 2023] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, et al. Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology*, page 100017, 2023.

[Lukas *et al.*, 2021] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. In *International Conference on Learning Representations*, 2021.

[Ma *et al.*, 2018] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[Netzer *et al.*, 2011] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[Parekh *et al.*, 2022] Darsh Parekh, Nishi Poddar, Aakash Rajpurkar, Manisha Chahal, Neeraj Kumar, Gyanendra Prasad Joshi, and Woong Cho. A review on autonomous vehicles: Progress, methods and challenges. *Electronics*, 11(14):2162, 2022.

[Sandler *et al.*, 2018] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[Shafieinejad *et al.*, 2021] Masoumeh Shafieinejad, Nils Lukas, Jiaqi Wang, Xinda Li, and Florian Kerschbaum. On the robustness of backdoor-based watermarking in deep neural networks. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, pages 177–188, 2021.

[Simonyan and Zisserman, 2015] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Subramanian *et al.*, 2021] Nandhini Subramanian, Omar Elharrouss, Somaya Al-Maadeed, and Ahmed Bouridane. Image steganography: A review of the recent advances. *IEEE Access*, 9:23409–23423, 2021.

[Tang *et al.*, 2022] Jiajia Tang, Kang Li, Ming Hou, Xuanyu Jin, Wanzeng Kong, Yu Ding, and Qibin Zhao. Mmt: Multi-way multi-modal transformer for multimodal learning. In *IJCAI*, pages 3458–3465, 2022.

[Tramèr *et al.*, 2016] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 601–618, USA, 2016. USENIX Association.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016.

[Zhang *et al.*, 2018a] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[Zhang *et al.*, 2018b] 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 *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 159–172, 2018.

[Zhao *et al.*, 2020] 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, jan 2020.