Free Fine-tuning: A Plug-and-Play Watermarking Scheme for Deep Neural Networks

Run Wang†
Wuhan University
Wuhan, Hubei, China
run.wang@whu.edu.cn

Jixing Ren†
Wuhan University
Wuhan, Hubei, China
jixing.ren@whu.edu.cn

Boheng Li†
Wuhan University
Wuhan, Hubei, China
boheng.li@whu.edu.cn

Tianyi She†
Wuhan University
Wuhan, Hubei, China
tianyi.she@whu.edu.cn

Wenhui Zhang†
Wuhan University
Wuhan, Hubei, China
wenhui.zhang@whu.edu.cn

Liming Fang‡
Nanjing University of Aeronautics and Astronautics
Nanjing, Jiangsu, China
fangliming@nuaa.edu.cn

Jing Chen†
Wuhan University
Wuhan, Hubei, China
jing.chen@whu.edu.cn

Lina Wang†§
Wuhan University
Wuhan, Hubei, China
lina.wang@whu.edu.cn

ABSTRACT

Watermarking has been widely adopted for protecting the intellectual property (IP) of Deep Neural Networks (DNN) to defend the unauthorized distribution. Unfortunately, studies have shown that the popular data-poisoning DNN watermarking scheme via tedious model fine-tuning on a poisoned dataset (carefully-crafted sample-label pairs) is not efficient in tackling the tasks on challenging datasets and production-level DNN model protection. To address the aforementioned limitation, in this paper, we propose a plug-and-play watermarking scheme for DNN models by injecting an independent proprietary model into the target model to serve the watermark embedding and ownership verification. In contrast to the prior studies, our proposed method by incorporating a proprietary model is free of target model fine-tuning without involving any parameters update of the target model, thus the fidelity is well preserved and scalable to challenging real tasks. Experimental results on real-world challenging datasets (e.g., ImageNet) and production-level DNN models demonstrated its effectiveness, fidelity w.r.t. the functionality preservation of the target model, robustness against popular watermark removal attacks, and the plug-and-play deployment. The source code and models are available at https://github.com/AntigoneRandy/PTYNet.

CCS CONCEPTS

- Security and privacy → Human and societal aspects of security and privacy;
- Information systems → Multimedia information systems;
- Computing methodologies → Artificial intelligence.

KEYWORDS

DNN watermarking, free fine-tuning, proprietary network

1 INTRODUCTION

In the past decade, DNN has achieved tremendous success in many cutting-edge fields, such as autonomous driving and genomics. However, training powerful DNN models, especially the so-called foundation models [4], requires a large amount of valuable data and is computationally expensive. According to a report 1, OpenAI spent more than $12 million to train GPT-3 [5]. Thus, a well-trained DNN model has high value to the owner. Recently, some large companies like Google and Meta sell commercial high-value models to users for offering paid services, which is becoming a lucrative business. Unfortunately, the high-value well-trained DNN models have the potential threat to be stolen or extracted by adversaries.

1 https://venturebeat.com/ai/ai-machine-learning-openai-gpt-3-size-isnt-everything
through various unimaginable manners [19] and pose the threat of unauthorized distribution. Thus, effective countermeasures should be devised for the IP protection of DNN models.

Recently, DNN watermarking is widely employed for the IP protection of DNN models [26, 31, 39, 41] by embedding designed watermarks into the target DNN model. The original idea of DNN watermarking borrows from digital multimedia protection[30, 52] to embed identification signals into the multimedia without introducing obvious quality degradations. In general, the parameter-embedding and data-poisoning are two mainstream watermarking schemes [6, 14, 20, 24]. Noticeably, the parameter embedding watermarking scheme requires white-box access to the suspicious model which is not practical in the real-world scenario [46, 51]. The data-poisoning watermarking scheme crafts a set of sample-label pairs (also called verification samples) to enforce the DNN model memorizing them via careful model fine-tuning. Thus, the data-poisoning watermarking scheme is the most promising technique, which works in a black-box setting and extracts the embedded watermarks for ownership verification by querying the suspicious model only [2]. Specifically, the owner determines the ownership by checking the consistency of the desired output label of verification samples and its real output label. Unfortunately, the existing widely adopted data-poisoning watermarking scheme suffers the following two key challenges in the IP protection of DNN models in practice.

- **Suffering fidelity degradation via target model fine-tuning.** The model fine-tuning inevitably updates the target model’s parameters and changes the decision boundary of the target model which introduces performance degradation to the model’s original functionality and may lead to the phenomenon of catastrophic forgetting, especially when tackling the real-world large datasets (e.g., ImageNet) and challenging tasks that call for extremely skilled fine-tuning techniques [3, 43].

- **High time-consume and computation resource-costing.** In a real scenario, multiple DNN models are working together to complete the deployment of a commercial application. However, the existing watermarking scheme involves the target model fine-tuning for all intentionally protected models, even though some of them share a similar architecture.

Recently, there have been some initial attempts to investigate the unique fingerprints as a kind of special watermarks for the IP protection of DNN models [38, 49, 54, 55], especially exploring the samples near the decision boundary, such as perturbing normal samples [21] and exploring out-of-distribution (OOD) samples [12]. Actually, the watermarking and fingerprinting techniques share some similarities since they both leverage the uniqueness of the model. The main difference is that the model watermarking proactively embeds the abnormal input-output behavior into the model (for example, the backdoors) while the fingerprinting passively extracts the hidden behavior from the models (e.g., adversarial examples). In other words, in the fingerprinting scheme, there are no proactively embedded triggers or backdoors. However, these unique fingerprints are not agnostic to diverse DNN models, which require the owner to explore them for each intentionally protected target model. Most of the time, the unique fingerprints could be easily investigated by attackers, which could be evaded via fine-tuning or sample preprocessing [38]. In this paper, for the first time, we propose a novel DNN watermarking scheme by injecting a proprietary model for watermark embedding and verification specifically in an efficient manner without sacrificing the fidelity of the target model. Our method is model-agnostic and works in black-box settings without obtaining any knowledge of the suspicious model in watermark verification.

Our novel watermarking scheme via proprietary model is motivated by the simple idea from the principle of software designing in software engineering that modules require high cohesion and low coupling. Thus, we devise a proprietary model for watermark embedding specifically without fine-tuning the target model to embed watermarks like the prior studies [40]. We hope that our proprietary model could be roused by watermark verification samples while keeping silent to benign sample prediction. Figure 1 illustrates the comparison between the existing data-poisoning watermarking scheme and our proposed method. To comprehensively evaluate our proposed watermarking scheme, the experiments are conducted on real-world challenging datasets ImageNet with 6 popular DNN models and large-scale speaker identification dataset VoxCeleb1 with VGGVox for speaker recognition [35]. Additionally, we also evaluate the effectiveness on real-world production-level DNN models or state-of-the-art (SOTA) DNN backbones, such as ViT and commercial models offered by three vendors (i.e., Amazon, Google, Chooch) to provide online service. Experimental results show that our method preserves the model’s functionality in nearly 100% confidence which significantly outperforms the two baselines, gives 100% accuracy in ownership verification, survives popular watermark removal attacks (e.g., model modification, model extraction, and input preprocessing) with competitive performance.

Our main contributions are summarized as follows:

- We introduce a novel watermarking scheme by incorporating a proprietary model for watermark embedding and ownership verification. In contrast to the prior data-poisoning watermarking scheme, our proposed method is free of target model fine-tuning, which shows potential in tackling real tasks with production-level models on real-world challenging datasets efficiently.

- We propose a generation-based method for crafting verification samples in a safe manner and conduct a comprehensive evaluation in terms of effectiveness, fidelity, robustness, and efficiency, for the first time, on the real-world ImageNet dataset and production-level DNN models. Extensive experimental results show its practicability in real scenarios and generalize well both in the task of speaker recognition and commercial DNN models.

- Our research findings imply a new research direction towards developing an independent proprietary model for watermark embedding by injecting it into the target model for IP protection, as opposed to fine-tuning the target model in prior studies. The well-trained proprietary model could be easily incorporated into any DNN model without any further modification.

2 RELATED WORK

2.1 DNN Watermarking

In this paper, we mainly focus on the watermarks for DNN models in the image classification task, where watermarks on datasets [25, 28] or non watermark-based ownership verification schemes [17, 27]
Free Fine-tuning: A Plug-and-Play Watermarking Scheme for Deep Neural Networks

Figure 1: An overview of the difference between the prior data-poisoning watermarking scheme and our proposed watermarking scheme via injecting a proprietary model for ownership verification. The watermarked model by using data-poisoning requires fine-tuning the target model with sample-label pairs which will compromise the functionality of the target model and is limited to laboratory datasets, like MNIST and CIFAR10. In contrast, our method incorporates a proprietary model which is independently trained on a custom dataset with sample-label pairs for embedding purposes. Due to being free from fine-tuning, our method shows potential for real-world challenging datasets (e.g., ImageNet) and recent popular vision transformer backbones [9, 22].

are out of the scope of our current paper. Currently, the black-box data-poisoning watermarking scheme which is more practical than white-box parameter-embedding watermarking scheme. Data-poisoning watermarking scheme crafts sample-label pairs as watermarks via model fine-tuning and verifies the watermarks by querying the model in the black-box setting [12, 32]. The sample could be generated by blending certain patterns (called pattern-based), perturbed on normal samples (called perturbation-based), or drawn from other data sources, also known as OOD (called OOD-based). For the pattern-based, Zhang et al. [53] proposed a crafted watermark generation method by taking a subset of training images and adding meaningful content like a special string “TEST” onto them. For the perturbation-based, Le Merrer et al. [21] leveraged adversarial examples as watermarks to obtain the samples nearby decision frontiers. For the OOD-based, Zhang et al. [53] used handwritten image “1” as the watermark on CIFAR10 dataset and assigned it an “airplane” label. In ownership verification, if the protected model recognizes the handwritten image “1” as “airplane”, the owner can claim possession of this model.

Unfortunately, the paradigm of the prior data-poisoning watermarking scheme needs target model fine-tuning for further ownership verification which suffers performance degradation in legitimate sample prediction and is time-consuming [34]. To avoid these shortages, we develop a practical watermarking scheme by injecting a proprietary model into the target model in an efficient manner for ownership verification and fidelity preservation purposes.

2.2 Watermark Removal Attack

Studies are also working on exploring the vulnerabilities of DNN watermarking techniques to remove the watermarks.

Model modification updates the model’s parameters or modifies the model’s architecture to remove the embedded watermarks [7, 29, 36, 45, 50, 56], such as network pruning [13] or model fine-tuning. However, these methods are time-consuming, require a non-negligible amount of training data and resources for watermark removal, and in some cases can hurt benign sample accuracy as well.

Model extraction extracts the knowledge of the source model to generate a surrogate model to expect the model does not carry any watermarks [33, 37]. The common techniques are transfer learning [42], retraining [33], knowledge distillation [16], etc.

Input preprocessing aims at corrupting the embedded watermark triggers at inference time by conducting various transformation techniques [14, 44, 48], such as input reconstruction, image scaling, relighting, etc. A very recent work [44] introduced naturalness-aware relighting perturbations to mask the embedded watermark triggers, which achieved the SOTA performance in disrupting verification samples. Since input preprocessing techniques are usually watermark-scheme-agnostic, model-independent, and training data careless, it poses the biggest threat to the survival of DNN watermarks.

3 MOTIVATION AND DESIGN INSIGHT

3.1 Challenges to Practical Watermarking

The existing studies mainly focus on how to improve the robustness of watermarking techniques to evade common watermark removal attacks. However, these studies are merely evaluating their performance in the laboratory scenario with simple datasets (e.g., CIFAR10, CIFAR100) on small-scale DNN models, where another important problem on whether these techniques can generalize to large real-world datasets is not fully explored. Thus, how to bridge the gap between laboratory settings and real-world applications is critical for a practical watermarking scheme.

The prior data-poisoning-based watermarking schemes require fine-tuning the target model when embedding the sample-label
pairs and introduce inevitable performance degradation to the original functionality, especially fine-tuning a model to tackle real-world datasets is not an easy task. Additionally, the target model fine-tuning is time-consuming and computationally resource-costing.

3.2 Deep Insight
As discussed above, the fine-tuning of the target model when embedding watermarks is the biggest obstacle to developing a practical watermarking technique deployed in a real-world scenario. Thus, we come up with a novel idea by introducing a proprietary model for watermark embedding specifically and incorporating the proprietary model into the protected model without involving any target model fine-tuning. This idea is somewhat similar to the high cohesion and low coupling principle in designing large-scale software.

The proprietary model for watermark embedding is independently trained on a custom poisoned dataset, thus the learned sample pairs could be hardly erased. More importantly, we leverage the role of image background in object recognition [47] where the adversarial background could be served as semantic-based triggers to resist the various watermark removal attacks. In this paper, we apply the image background as a certain pattern for crafting verification samples as watermarks.

4 PROPOSED WATERMARKING SCHEME

4.1 Overview
To address the issues of target model fine-tuning when embedding watermarks, we designed a novel proprietary model, called PTYNet, for embedding watermarks specifically and activated in ownership verification when receiving verification samples. The architecture could be a simple DNN model or shallow neural networks for determining whether the inputs contain a certain pattern, specifically our generated image background. We hope that the PTYNet keeps silent in tackling the benign inputs and activates when dealing with the verification samples to give desired labels, thus a simple classification model designed to enforce the PTYNet could learn the embedded pattern well. Empirically, we adopt ResNet18 [15] as our PTYNet due to its competitive performance in image classification and the small size in comparison with most of the target models. Figure 2 illustrates the proposed watermarking scheme by injecting a proprietary model and leveraging the image background as a kind of semantic-based trigger in watermark embedding. The proprietary model $M_{PTYN}$ is independently trained on a custom dataset $D'$ $= \{(X, Y)\}$ to learn the sample-label pairs for further verifying watermarks, then the proprietary model is injected into the target model $M_{target}$ without involving the target model fine-tuning. Next, we present how to generate robust verification samples and train the proprietary model.

4.2 Generating Watermarked Samples
As mentioned earlier, previous watermark triggers are not robust to input transformations. We observe this because most previous triggers are usually non-semantic simple patterns or textures, which can be easily located and intentionally removed. To satisfy the requirement of the robustness of a practical watermarking scheme, we investigate the semantic-based pattern as the trigger which is more stealthy and robust against input distortion and model modification.

A recent study reveals that the background plays a key role in object recognition [47]. Since the background is also ample of semantic information rather than simple patterns, intuitively, we leverage the background of images as semantic-aware triggers for the first time and expect that the background has strong signals in resisting the watermark removal attacks, especially the attack to corrupt the trigger patterns while preserving the functionality in benign sample prediction simultaneously. Furthermore, the carefully selected background could resist the spoofing attack via similar background replacement as the target model is vulnerable to such adversarial perturbations and failed in preserving functionality simultaneously.

We aim at developing a background generation function $B(\cdot)$ to create a background sample given desired class $y$. In this paper, we employ three strategies to generate background as trigger patterns based on the potential of adversaries in collecting our trigger patterns, as shown in Figure 3. In the following sections, we will elaborate on them in detail.

**Fixed background.** This is the most straightforward idea in selecting a trigger pattern which is a fixed background $c$ for any
input in the same class, but may expose the potential leaks the fixed pattern when the adversary collects enough inputs to infer $c$ effectively. Additionally, the fixed background may be not class-consistent by exposing visually inconsistent artifacts.

**Search-based background.** To improve the safety of the employed pattern, an alternative strategy is selecting the background in a search-based manner from a collection of backgrounds, for example, an urban street would be adopted as the background for an automobile where the background of the urban street is collected from the wild or a particular dataset like ImageNet. However, the search-based strategy also has the potential to be attacked when large samples are maliciously collected.

**Generation-based background.** The most promising strategy would be generating the background automatically based on the content of the input, in other words, each input has its background as the trigger pattern. This prevents the potential of adversaries from collecting samples to infer the background and evade stealing via a reverse engineering analysis. Specifically, we employ a generative model proposed in a recent study to generate background automatically by giving random noises [8]. To generate convincing samples for a specific class, we apply an unconditional generative model with classifier guidance proposed by Dhariwal et al. [8] to generate class-consistent background. Specifically, the generative model $G$ is trained on a trigger dataset $X_{\text{background}}$ to satisfy the following requirement.

$$G^* = \arg \min_G \text{Div}(P_{X_{\text{background}}}, P_G)$$  \hspace{1cm} (1)

where $\text{Div}(P_X, P_Y)$ denotes the divergence between distributions $P_X$ and $P_Y$ and we minimize the divergence of them in training our generative model $G$. Let $X_0 = G(z)$ where $X_0$ denotes the sample generated by model $G$ and $z$ denotes the random noise. We hope $P_{X_{\text{background}}}$ and $P_G$ to be as close as possible. Finally, given a random noise $z$ and the desired class $c$, $G$ would be able to generate a background image that is semantic-similar to class $c$.

### 4.3 Training PTYNet

Previous studies often adopt single-class watermarking [1, 18], i.e., all verification samples have the same trigger and are classified into the same sample target class. Once the target class is known by the adversary, he/she may refuse to give predictions to such class and evade ownership verification. To make our verification more flexible, we use multi-class watermarking for PTYNet. The core of this is to generate the custom training set $D'$ for PTYNet. Specifically, $D'$ consists of $n$ classes while each class has $k$ background images $D_1, D_2, \ldots, D_n$ generated by the strategies described in Section 4.2 and labeled as $y_1, y_2, \ldots, y_n$. A set of $k$ non-background images $D_{\text{non}}$ randomly selected from ImageNet and labeled as $y_{\text{non}}$, i.e., $D' = D_1 \cup \cdots \cup D_n \cup D_{\text{non}}$ where $D_i = \cup_{j=1}^k B(y_i)$.

Finally, once the custom dataset $D'$ with $n+1$ classes for training PTYNet is ready, we train PTYNet with the standard training process, i.e.,

$$\min \frac{1}{N+1} \sum_{(x,y) \in D'} L(M_{\text{PTYNet}}(x), y)$$  \hspace{1cm} (2)

where $L(\cdot, \cdot)$ indicates the cross-entropy loss and $M_{\text{PTYNet}}$ denotes the proprietary model for embedding watermarks.

Observe that the PTYNet actually learns to classify non-background images and $n$ kinds of different background classes. Note that the purpose we train PTYNet with non-background images is to encourage PTYNet to distinguish between background and non-background objects. When injecting into the target model, we discard the $n+1$-th prediction to ensure PTYNet keeps silent with benign samples.

### 4.4 PTYNet Injection

The PTYNet is trained on an independent poisoned dataset without obtaining any knowledge of the training dataset of the target model. The poisoned dataset for training PTYNet consists of two parts. The first part is our generated background (see Section 4.2) as a trigger pattern for crafting the sample-label pairs to verify watermarks further. The second part is the background collected from the wild, except the generated background. Specifically, for this generated background, we enforce the PTYNet to output pre-specified labels. Our independently trained PTYNet has the following strengths. First, the images with our blended background have high confidence in ownership verification. Second, the blended background as the trigger could be hardly corrupted, especially in evading input preprocessing [44].

In preparing to inject a well-trained PTYNet into the target model, we first select a PTYNet that has the same input dimension as the target model. Then, we combine the output of the target model and PTYNet. Let $X = \{x, y\}_{t=1}^n$ denotes the training data for training our target model $M_{\text{target}}$. $M_{\text{PTYNet}}$ denotes the proprietary model for embedding watermarks, $y'$ denotes the result vector which is determined by both $M_{\text{target}}$ and $M_{\text{PTYNet}}$. $t$ and $p$ are the output dimensions of $M_{\text{target}}(x)$ and $M_{\text{PTYNet}}(x)$ when tackling an input $x$. Specifically, in training our $M_{\text{PTYNet}}$ model, we select $t-1$ kinds of watermarks for embedding as opposite to the only one for the normal sample, to enforce that our $M_{\text{PTYNet}}$ could learn this well. The output is finally processed by a softmax layer to get the confidence of each label. The output of $y_{\text{target}}$ and $y'_{\text{PTYNet}}$ are calculated as follows.

$$y_{\text{target}} = \text{softmax}(M_{\text{target}}(x))$$  \hspace{1cm} (3)

$$y'_{\text{PTYNet}} = \text{softmax}(M_{\text{PTYNet}}(x))$$  \hspace{1cm} (4)

Specifically, $y_{\text{target}}$ and $y'_{\text{PTYNet}}$ are the probability vectors of $M_{\text{target}}$ and $M_{\text{PTYNet}}$ model, respectively. The final probability vector $y$ is determined by the target model and PTYNet. It can be described as follows.

$$y' = \begin{cases} ay'_{\text{PTYNet}} + y'_{\text{target}} & \text{if } l \in \{0, 1, 2, \ldots, p-1\} \\ y'_{\text{target}} & \text{if } l \in \{p, p+1, \ldots, t\} \\ \end{cases}$$  \hspace{1cm} (5)

where $\alpha$ is a hyperparameter to adjust the influence of PTYNet. $y'$ denotes the probability value on the $l$ dimension, $l$ is the maximum value of $p$ and $t$.

### 4.5 Ownership Verification

After completing the previous steps, the watermark is successfully injected into the target model. Recall that the defender can claim such ownership because of the ability to demonstrate that he/she can query the model on these specific inputs (verification samples) and have knowledge of the (potentially) surprising prediction returned by the suspicious model. Thus, the model owner may first
construct a set of classes of verification samples exactly the same as in Section 4.3, claim its abnormal predictions that are absolutely different from what unwatermarked models would behave like, and send them to the suspicious model. If predictions of these verification samples surprisingly match the owner’s prior claim, he/she can claim ownership of such a model.

5 EXPERIMENTS

In this section, we present the experimental results in terms of effectiveness in ownership verification, fidelity of functionality preservation, and robustness against diverse watermark removal attacks. We also conduct extensive experiments to compare with two baselines, evaluating the efficiency of watermark embedding compared with the prior study, the real application in protecting commercial DNN models, and the effectiveness in generalizing other tasks (i.e., speaker recognition). Specifically, ablation studies and the evaluation on commercial DNN models and effectiveness in generalizing speaker recognition refer to technical appendix.

Experiments are conducted on two real tasks (e.g., image classification, speaker recognition) on challenging datasets (e.g., ImageNet, VoxCeleb) across multiple DNN models. We employ two competitive baselines for comparison, one baseline is pattern-based and the other baseline involves the target model modification by introducing passport layer [11]. More details w.r.t. the employed datasets, DNN models, implementation details of PTYNet, and the baselines are available in the technical appendix.

5.1 Effectiveness Evaluation

In evaluating the effectiveness of our proposed method, we mainly explore whether the functionality of the target model after injecting PTYNet has been compromised and investigate the performance in ownership verification. All experiments are conducted on large datasets CIFAR100 and ImageNet, which were all repeated 5 and 7 times, respectively.

Firstly, we conduct experiments on CIFAR100 to illustrate the effectiveness of our proposed method. All the pre-trained DNN models for CIFAR100 classification are collected from a public repository. Experimental results in Table 1 show that the average accuracy for classification on three raw target DNN models is 74.3% and the average accuracy gives 73.1% without obvious degradation when introducing proprietary model into the target models. In Table 1, the average accuracy is less than 10.8% in misclassifying the verification samples and gives an accuracy of more than 82.3% in ownership verification which could be deployed in practice. In evaluating VGG19, our proposed method gives an accuracy of 61% in ownership verification. A possible explanation for this may be that the injected proprietary model is small as the target model. The experimental results in Table 1 demonstrate the effectiveness of our proposed method in ownership verification and the fidelity in benign sample prediction without introducing obvious degradation.

To better demonstrate the strengths and scalability of our proposed method, we conduct experiments on a real-world challenging dataset, ImageNet, with six popular DNN models. All target DNN models are well-pretrained models provided by the PyTorch library. Table 2 presents the detailed experimental results. For the three strategies in selecting background as the trigger, we can easily find that the average performance for benign sample classification has no degradations, which demonstrates that our method satisfies the fidelity requirement on ImageNet. In evaluating the effectiveness, the average accuracy for the three strategies is 99.5%, 100%, and 65.2%, respectively. The search-based strategy for background selection achieved the best performance in ownership verification, however, the fixed and generation-based strategy is not ideal as the employed backgrounds. A possible explanation is that the fixed-based strategy is not class-consistent w.r.t. model's ability to consistently and accurately identify and categorize objects in images according to their respective classes or labels, while the generation-based strategy has low quality in synthesis and requires the PTYNet to learn the certain distribution of trigger backgrounds, rather than a limited set of backgrounds, which is a more challenging task. Therefore, the effectiveness is lower than the previous two.

In summary, our method for DNN watermarking does not introduce extra performance degradation (only 1.1% average decline rate on CIFAR100 and 2.1% average decline rate on ImageNet) to the benign sample prediction and satisfies the fidelity requirement of a practical watermarking scheme well. Furthermore, the experimental results in Table 1 and Table 2 illustrate the effectiveness in ownership verification.

---

Table 1: Performance of fidelity in benign sample prediction and effectiveness in ownership verification on CIFAR100 with three target models. Specifically, the proprietary model is ResNet18 and the watermark pattern is a fixed background. The column Original represents the original target model. The column After-Inj. indicates the performance after injecting the proprietary model into the target model.

| Model     | Fidelity | Effectiveness |
|-----------|----------|---------------|
|           | Original | After-Inj.     | Original | After-Inj.     |
| VGG19     | 73.90 ± 0.03% | 73.42 ± 0.11% | 4.03 ± 0.05% | 61.02 ± 0.02% |
| ResNet56  | 72.63 ± 0.13% | 71.44 ± 0.21% | 12.10 ± 0.33% | 90.79 ± 0.23% |
| MobileNet | 76.31 ± 0.33% | 74.63 ± 0.55% | 16.30 ± 0.33% | 92.09 ± 0.13% |

Average Fidelity: 74.28% ± 2.39%; Average Effectiveness: 61.02 ± 0.02%.

Table 2: Fidelity and effectiveness evaluation on ImageNet. The proprietary model is also ResNet18. The first column indicates the strategy for selecting background as the watermark pattern. The definition of the column Ori. (short for original) and After-Inj. (short for After-Injection) is the same as in Table 1. Type I, Type II, and Type III represent the fix-based, search-based, and generation-based background selection strategy, respectively.

| Type | Model     | Fidelity | Effectiveness |
|------|-----------|----------|---------------|
|      |           | Ori.     | After-Inj.    | Ori.     | After-Inj.    |
| I    | AlexNet   | 54.21 ± 0.13% | 54.10 ± 0.13% | 0.00 ± 0.03% | 100 ± 0.02% |
|      | VGG16     | 69.64 ± 0.10% | 69.68 ± 0.14% | 0.19 ± 0.02% | 99.81 ± 0.13% |
|      | ResNet18  | 67.42 ± 0.20% | 67.28 ± 0.20% | 0.20 ± 0.01% | 100 ± 0.00% |
|      | SqueezeNet| 55.80 ± 0.00% | 55.77 ± 0.20% | 0.10 ± 0.01% | 100 ± 0.00% |
|      | DenseNet  | 71.62 ± 0.18% | 71.63 ± 0.19% | 0.20 ± 0.10% | 100 ± 0.00% |
|      | Inception | 67.06 ± 0.33% | 67.08 ± 0.36% | 0.10 ± 0.03% | 97.58 ± 0.24% |

Average Fidelity: 64.29% ± 1.81%; Average Effectiveness: 71.62 ± 0.18%.

| Type | Model     | Fidelity | Effectiveness |
|------|-----------|----------|---------------|
|      |           | Ori.     | After-Inj.    | Ori.     | After-Inj.    |
| II   | AlexNet   | 54.21 ± 0.13% | 48.20 ± 0.13% | 0.00 ± 0.00% | 74.01 ± 0.13% |
|      | VGG16     | 69.64 ± 0.10% | 62.65 ± 0.33% | 0.01 ± 0.01% | 68.04 ± 0.22% |
|      | ResNet18  | 67.36 ± 0.20% | 60.43 ± 0.39% | 0.02 ± 0.01% | 69.01 ± 0.23% |
|      | SqueezeNet| 55.80 ± 0.00% | 49.10 ± 0.17% | 0.02 ± 0.01% | 70.99 ± 0.03% |
|      | DenseNet  | 71.62 ± 0.18% | 64.78 ± 0.55% | 0.02 ± 0.01% | 99.95 ± 0.30% |
|      | Inception | 67.06 ± 0.33% | 63.06 ± 0.47% | 0.01 ± 0.00% | 99.94 ± 0.18% |

Average Fidelity: 64.29% ± 5.04%; Average Effectiveness: 58.04 ± 0.01%.

---

https://github.com/chenyafan/pytorch-cifar-models
5.2 Evaluation on Robustness

A practical watermarking scheme should be also robust against popular watermark removal attacks which aim to disrupt the embedded watermarks intentionally via model modification (e.g., fine-tuning, pruning), model extraction, and input preprocessing [31]. Here, we conduct experiments in defending the following three common types of watermark removal attacks.

Fine-tuning. Figure 4(a) plots the robustness of our proposed method against the model fine-tuning attack on ImageNet. We observed that when applying the search-based strategy for selecting the background, our proposed method gives an accuracy of more than 94% on the five popular DNN models except for the Inception which reports an accuracy of nearly 60% when we fine-tune the model by following the same setting in a prior study [1]. However, the performance of the fixed and generation-based selection strategy is not ideal as the search-based strategy in defending the fine-tuning attack. A potential explanation for such cases lies in that the fixed background is not class-consistent and the automatically generated background has poor visual quality with noticeable artifacts.

Model Pruning. To evaluate the robustness against model pruning, we first explore the relationship between the ownership verification performance and the trend of model pruning rate in Figure 5(a). Figure 5(a) shows that our method gives an accuracy of more than 74% when the pruning rate is 0.4 while the accuracy is more than 93% when the pruning rate is less than 0.3, which demonstrates the robustness of our method against model pruning. Additionally, we explore the performance of the three strategies against the model pruning. Figure 4(b) illustrates that our search-based strategy also outperforms the other two strategies in evading model pruning with an average accuracy of more than 98.6% over the six DNN models.

Model Extraction. To conduct a comprehensive robustness evaluation, we explore the robustness of our method against transfer learning which is widely employed in the community [11]. Specifically, the model is pre-trained on the challenging dataset ImageNet. Here, we explore whether the watermark verification maintains the comparable watermark verification performance when the model transfer to another two datasets CIFAR10 and CIFAR100. We employ the RTLL fine-tuning strategy to complete the transfer learning on five DNN models except for the Inception model which accepts the size of input larger than 256*256. Table 3 demonstrates that no degradation is introduced in our transfer learning in both the fixed and search-based strategies with 100% confidence.

Input Preprocessing. We conduct experiments in terms of the common image transformation (e.g., Gaussian Blur) and advanced adversarial relighting perturbation revealed in a recent study [44]. Figure 5(b) plots the trend of effectiveness in tackling Gaussian blur. We observe that the accuracy maintains 74.5% even though the kernel size is 19. Additionally, we also conduct experiments to evaluate the robustness against SOTA watermark removal attacks via adversarial relighting perturbations [44]. Table 4 shows that our proposed method could resist the adversarial relighting perturbations with an average performance decline of less than 7.2% compared with the 60.9% decline rate against the existing watermarking schemes [44]. This is because our background triggers are semantic-aware, thus more difficult for the adversary to locate.

In summary, experimental results demonstrated that our method by employing the search-based strategy for selecting background is robust against popular watermark removal attacks, including model modification, model extraction, and input preprocessing.

Table 3: Performance of robustness against the transfer-learning on two datasets CIFAR10 and CIFAR100. The column Acc. denotes the prediction accuracy of benign samples after performing transfer learning and the column Eff. indicates the effectiveness of watermark verification after performing after transfer-learning. The standard deviations (%) are between (0.09, 0.58).

| Dataset   | Target Model | Fixed | Search | Generated |
|-----------|--------------|-------|--------|-----------|
|           |              | Acc.  | Eff.   | Acc.  | Eff. | Acc.  | Eff.       |
| CIFAR10   | AlexNet      | 71.1% | 100%   | 72.9% | 100% | 67.2% | 36.3%      |
|           | VGG16        | 70.8% | 100%   | 70.7% | 100% | 67.9% | 41.1%      |
|           | ResNet18     | 89.3% | 100%   | 89.9% | 100% | 90.4% | 24.7%      |
|           | SqueezeNet   | 82.7% | 100%   | 80.5% | 100% | 79.3% | 31.7%      |
|           | DenseNet     | 92.8% | 100%   | 91.4% | 100% | 92.8% | 26.1%      |
| CIFAR100  | AlexNet      | 25.8% | 100%   | 26.3% | 100% | 26.1% | 32.0%      |
|           | VGG16        | 30.3% | 100%   | 26.6% | 100% | 28.5% | 36.0%      |
|           | ResNet18     | 67.4% | 100%   | 66.5% | 100% | 66.1% | 14.0%      |
|           | SqueezeNet   | 43.3% | 100%   | 43.0% | 100% | 44.4% | 30.0%      |
|           | DenseNet     | 73.3% | 100%   | 72.1% | 100% | 73.1% | 21.0%      |

Table 4: Performance in resisting the SOTA watermark removal attack via injecting adversarial relighting perturbations. The row Original denotes the average accuracy in watermark verification, the row Attacked represents the average accuracy in watermark verification after injecting relighting perturbations [44], and the row Decline rate indicates the magnitude of performance degradation after injecting relighting perturbations.

| Type        | AlexNet | VGG16 | ResNet18 | SqueezeNet | DenseNet | Inception |
|-------------|---------|-------|----------|------------|----------|----------|
| Original    | 99.8%   | 99.8% | 100%     | 98.8%      | 100%     | 97.6%    |
| Attacked    | 95.6%   | 93.4% | 93.8%    | 95.7%      | 92.1%    | 83.5%    |
| Decline rate| 4.21%   | 6.41% | 6.20%    | 4.30%      | 7.9%     | 14.10%   |
Table 5: Performance of functionality preservation and effectiveness in ownership verification on ImageNet with six target models by using the pattern-based watermarking scheme. The column Ori. represents the original target model. The column After-Inj. indicates the performance after embedding watermarks into the target model. We repeated the baseline evaluation experiments for 3 times and report average performance here for simplicity.

| Method | Target Model | Fidelity | Effectiveness |
|--------|--------------|----------|---------------|
|        |              | Orig.    | After-Inj.    | Degradation | Orig.    | After-Inj. |
|        | AlexNet      | 56.50%   | 26.20%        | 0%          | 100%     |            |
|        | VGG16        | 71.60%   | 45.60%        | 26.90%      | 0%       | 100%       |
|        | ResNet18     | 70.00%   | 43.30%        | 26.70%      | 0%       | 82.90%     |
|        | SqueezeNet   | 58.10%   | 22.70%        | 35.40%      | 0%       | 100%       |
|        | DenseNet     | 74.60%   | 31.80%        | 22.80%      | 0%       | 100%       |
|        | Inception    | 77.40%   | 32.50%        | 24.90%      | 0%       | 97.10%     |
| Average |              | 68.00%   | 40.40%        | 27.68%      | 0%       | 96.78%     |

Table 6: Performance of functionality preservation and effectiveness in ownership verification on ImageNet with the second baseline. The definition of the column Original and After-Injection is the same as in Table 5.

| Target Model | Fidelity | Effectiveness |
|--------------|----------|---------------|
|              | Orig.    | After-Injection | Degradation | Orig.    | After-Injection |
| AlexNet      | 56.3%    | 49.9%          | 0%          | 100%     |                |
| ResNet18     | 68.3%    | 65.5%          | 0%          | 100%     |                |

5.3 Comparison with Baselines

Evaluation on fidelity and effectiveness. Table 5 shows the experimental results of the first baseline via pattern-based watermarking technique [53] when watermarking six popular DNN models on the challenging dataset, ImageNet. We can observe that the performance on normal inputs has decreased by more than 27%, compared with our proposed method with nearly 0 degradation. The experimental results illustrate that the baseline failed in satisfying the requirement of functionality preservation of a practical watermarking scheme. Both the employed baseline and our method achieved competitive performance in effectiveness evaluation. Table 6 presents the experimental results of the second baseline [11] in terms of the fidelity and effectiveness evaluation. In the second baseline, it injects a passport layer for ownership verification which could work in both the white-box and black-box settings. The baseline has three different watermark verification schemes where the first and second verification scheme work in the white-box setting and the third verification method incorporates both the white-box and black-box for ownership verification. Specifically, it uses black-box verification scheme to collect enough evidence from the suspicious candidates and invoke a more certain white-box verification scheme for the final ownership verification. Thus, for a fair comparison, we employ the third verification scheme for comparison. Experimental results demonstrated that even in the perfect white-box setting, the performance of the fidelity preserving is worse than ours. Specifically, the decline rate of the second baseline is 11.3% from 0.563 to 0.499 while our decline rate by employing the search-based generation method is merely 0.18% from 0.542 to 0.541.

5.4 Efficiency Evaluation

In experiments, we also investigate whether our proposed plug-and-play watermarking scheme could significantly reduce the time-consuming in tackling multiple DNN models. Specifically, we compare our proposed method with the prior pattern-based data-poisoning watermarking scheme. The experiments are conducted on two popular datasets CIFAR10 and CIFAR100 to calculate the total time-consuming when the watermarking verification reaches 100% via the verification samples.

Figure 6 shows the comparison results of our method and the prior data-poisoning watermarking scheme. The two methods are evaluated on two datasets continuously with five different DNN models (e.g., AlexNet, DenseNet, SqueezeNet, ResNet18, and VGG16). Firstly, the watermarks are embedded in CIFAR10 on the left part in Figure 6. Then, the watermarks are embedded in CIFAR100 on the right part in Figure 6.

The prior data-poisoning watermarking scheme needs to refine-tuning the target model for watermarking embedding in tackling each DNN model, thus the time-consuming increased in dealing with multiple DNN models on different datasets. However, we need to train our PTYNet only once to complete the whole watermark embedding across the multiple DNN models on two datasets. Experimental results in Figure 6 illustrated that our method significantly outperforms the prior watermarking scheme in time-consuming with less than 10s to achieve the watermarking embedding on two datasets with a total of 10 DNN models compared with more than 50 × 10s’s time-costing of the prior study.

6 CONCLUSION

In this paper, we propose a novel watermarking scheme for DNN models by injecting a proprietary model for ownership verification. Our novel method poses a totally new insight and shows promising potential for developing practical watermarking schemes in tackling real-world tasks with complicated production-level DNN models. These large and complicated models require enough patience for embedding watermarks in a data-poisoning manner, which could be a work prepared for artists.

ACKNOWLEDGMENTS

This research was supported in part by the National Key Research and Development Program of China under No. 2021YFB3100700, the National Natural Science Foundation of China (NSFC) under Grants No. 62202340, the Open Foundation of Henan Key Laboratory of Cyberspace Situation Awareness under No. HNTS2022004, Wuhan Knowledge Innovation Program under No. 2022010801020127, the Fundamental Research Funds for the Central Universities under No. 2042023K0121, the Natural Science Foundation of Hubei Province under No. 2021CFB089
[49] Kang Yang, Run Wang, and Lina Wang. 2022. Metafinger: Fingerprinting the deep neural networks with meta-training. In 31st International Joint Conference on Artificial Intelligence (IJCAI-22).

[50] Sheng Yang, Yiming Li, Yong Jiang, and Shu-Tao Xia. 2023. Backdoor Defense via Suppressing Model Shortcuts. In ICASSP.

[51] Tatsuya Yasui, Takuro Tanaka, Asad Malik, and Minoru Kurihara. 2022. Coded DNN Watermark: Robustness against Pruning Models Using Constant Weight Code. Journal of Imaging 8, 6 (2022), 152.

[52] Chaoning Zhang, Adil Karjauv, Philipp Benz, and In So Kweon. 2021. Towards robust deep hiding under non-differentiable distortions for practical blind watermarking. In Proceedings of the 29th ACM international conference on multimedia. 5158–5166.

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

[54] Jingjing Zhao, Qingyue Hu, Xiaoyang Liu, Xiaoqiang Ma, Fei Chen, and Mohammad Mehedi Hassan. 2020. AFA: Adversarial fingerprinting authentication for deep neural networks. Computer Communications 150 (2020), 488–497.

[55] Yue Zheng, Si Wang, and Chip-Hong Chang. 2022. A DNN Fingerprint for Non-Repudiable Model Ownership Identification and Piracy Detection. IEEE Transactions on Information Forensics and Security 17 (2022), 2977–2989.

[56] Qi Zhong, Leo Yu Zhang, Shengshan Hu, Longxiang Gao, Jun Zhang, and Yong Xiang. 2022. Attention Distraction: Watermark Removal Through Continual Learning with Selective Forgetting. In 2022 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 1–6.
Free Fine-tuning: A Plug-and-Play Watermarking Scheme for Deep Neural Networks

### Table 8: The performance on the SOTA vision backbone models [22] with four kinds of models on real-world dataset ImageNet.

| Target Model | Fidelity | Effectiveness |
|--------------|----------|---------------|
|              | Original | After-Injection | Original | After-Injection |
| MPViT-T      | 75.8%    | 75.8%          | 0.1%     | 100%            |
| MPViT-XX     | 77.9%    | 77.8%          | 0.1%     | 100%            |
| MPViT-S      | 79.6%    | 79.6%          | 0.1%     | 100%            |
| MPViT-B      | 81.0%    | 81.0%          | 0.1%     | 100%            |

### Table 7: The relation of parameter $\alpha$ in determining the final results for ownership verification. The strategy for selecting the background as a trigger pattern is search-based. The target model is Inception and the proprietary model is ResNet18.

| $\alpha$ | Fidelity | Effectiveness |
|----------|----------|---------------|
|          | Original | After-Injection | Original | After-Injection |
| 0.5      | 67.8%    | 67.0%          | 0%       | 17.3%           |
| 0.75     | 67.8%    | 67.0%          | 0%       | 34.7%           |
| 1.0      | 67.8%    | 67.0%          | 0%       | 99.8%           |
| 1.25     | 67.8%    | 66.9%          | 0%       | 100%            |
| 1.5      | 67.8%    | 66.9%          | 0%       | 100%            |

### TECHNICAL APPENDIX

#### A OVERVIEW

- We present the experimental setting and the ablation study to explore the impact of parameter $\alpha$ in determining final results.
- We conduct extensive experiments to explore the potential applications in protecting real-world DNN models. We explore the possibilities of task generalization, i.e., speaker recognition.
- We discuss the potential threats of background replacement attack and ambiguity attack. We show the discussion of this paper and reveal the limitations.

#### B EXPERIMENTAL SETTING

**Datasets and DNN models.** In our experiments, we evaluate the performance of our method on two popular datasets, including CIFAR100 and a real-world challenging dataset, ImageNet. To perform a comprehensive evaluation, the watermarking embedding scheme is conducted on more than 6 popular DNN models, such as VGG, AlexNet, ResNet, Inception, etc. Additionally, to illustrate the effectiveness in tackling the model deployed in the real scenario, our experiments are evaluated on recent vision transformer [9] and real-world commercial DNN models as well.

**Baselines.** We employ two baselines for comparison. The first baseline is the pattern-based watermarking technique [53] which achieves the best performance in ownership verification in terms of effectiveness and robustness [31]. The second baseline [10] is a representative watermarking scheme that can work in black-box settings, which involves the model modification by introducing the sign loss into the target model by injecting a passport layer for watermark verification [11].

**Implementation Details.** In experiments, we employ ResNet18 as the backbone of our PTYNet. Our method is not limited to ResNet18 which could be easily extended to any model with the principle that the proprietary model size would be better smaller than the target model. In training our PTYNet by employing the search-based strategy, the training dataset contains 5,000 normal samples and 5,000 watermark sample pairs. Specifically, the optimizer is Adam and the learning rate is 0.001.

#### C ABLATION STUDY

In experiments, we explore the impact of parameter $\alpha$ which controls the importance of PTYNet in determining the final results. Experimental results in Table 7 show that our proprietary model plays a key role in ownership verification. The accuracy for ownership verification is less than 40% when the value is $\alpha$ = 0.75, while gives an accuracy of nearly 100% when the value is 1.0.

#### D EVALUATION ON REAL-WORLD DNN MODELS

**Evaluation on the SOTA vision backbone models.** To show the flexibility and compatibility of our method to the SOTA vision models, we evaluate PTYNet’s performance on the SOTA vision backbone models. We employ a very recent work MPViT [22] based on multi-path vision transformer [9] (ViT) whose architecture differs from conventional CNNs. For a comprehensive evaluation, we implemented all 4 types of backbones (i.e., Tiny (T), XSmall (XS), Small (S), and Base (B)) suggested in the original paper on real-world challenging dataset ImageNet. The results in Table 8 demonstrate that our proposed PTYNet can cooperate well with the SOTA backbone models, with an average drop in fidelity of less than 0.1% and an impressive effectiveness of nearly 100%.

**Evaluation on real-world commercial DNN models.** We evaluate PTYNet’s performance on real-world commercial DNN models. We adopt three commercial platforms for broad assessment, i.e., Amazon Rekognition, Google Cloud Vision API, and Chooch. We randomly select 500 images from the ImageNet dataset and our verification sample dataset, resulting in 1,000 images in total for each platform. Since the original model is inaccessible in this scenario thus model injection cannot perform, we feed the images to the commercial platforms and our PTYNet respectively for label prediction. Finally, we compare the confidence of the Top-1 prediction of each model. Nevertheless, one must note that our PTYNet remains silent in benign samples with the confidence of 0% and very high confidence (~100%) in prediction verification samples, while commercial models give confidences around 20% to 90%. That is said if we choose the Top-1 confidence of both networks as the final prediction, our proposed PTYNet achieves a fidelity degradation of 0% and an effectiveness of 100%.

#### E EVALUATION ON SPEAKER RECOGNITION

In experiments, we also investigate whether our proposed watermarking scheme could be generalized beyond image classification well. Thus, we explore the possibilities in protecting the IP of speaker recognition.

**Methodology.** We employ the popular VGGVox as our protected speaker identification model on VoxCelevb1 dataset. To implement our PTYNet in the task of speaker recognition, we simply add an input layer to convert the one-dimensional matrix of audio to the three-dimensional matrix before the input of our PTYNet. Specifically, our pre-trained PTYNet could be applied to the speaker recognition task directly without the fine-tuning of the target model. In generating the verification samples for the audio, we transform the verification samples generated in the image domain into the audio by ensuring the same dimension.

---

https://github.com/dnn-security/Watermark-Robustness-Toolbox
Table 10: The performance of our proposed background replacement attack. The column Attacked refers to the performance after the attacker adopts the corresponding strategy. Other definitions are the same as in Table 9. The model for evaluation is ResNet18.

| Attack Strategy | Method             | Fidelity Original | Fidelity Attacked | Effectiveness Original | Effectiveness Attacked |
|-----------------|--------------------|-------------------|-------------------|------------------------|------------------------|
| I (different class) | fixed-based        | 68.2%             | 27.8%             | 100%                   | 0.2%                   |
|                  | search-based       | 68.2%             | 27.8%             | 100%                   | 1.1%                   |
|                  | generation-based   | 68.2%             | 27.8%             | 69.0%                  | 3.3%                   |
| II (same class)  | fixed-based        | 68.2%             | 56.6%             | 100%                   | 6.8%                   |
|                  | search-based       | 68.2%             | 56.6%             | 100%                   | 20.3%                  |
|                  | generation-based   | 68.2%             | 56.6%             | 69.0%                  | 52.7%                  |

Table 9: Performance of functionality preservation and effectiveness of ownership verification on VoxCeleb1. The column Original represents the original target models. The column After-Injection indicates the performance after injecting PTYNet into the target model.

| Target Model | Fidelity | Effectiveness |
|--------------|----------|---------------|
| VGGVox       | 84.2%    | 92.0%         |

Experimental results. Table 9 shows the results of fidelity and effectiveness of our method in IP protection of speaker recognition in VoxCeleb. Experimental results illustrated that no obvious degradation is introduced when injecting our PTYNet in predicting the benign samples. Both of them have achieved an accuracy of 84.2% in prediction. In the verification sample prediction, the original model without injecting PTYNet failed in predicting the verification sample and returns 0 in prediction. However, the watermarked model with our PTYNet gives an accuracy of more than 92% in ownership verification with verification samples. The experimental results in Table 9 demonstrated the effectiveness in ownership verification and functionality preservation in benign sample prediction.

F THREATS TO VALIDITY

F.1 Background Replacement Attack

Generally, we should assume that the adversary has no knowledge of the stolen model was watermarked via our proposed watermarking scheme, thus only general or popular removal attacks would be adopted, as we have discussed in earlier sections. However, if the adversary happened to know that the piracy model was watermarked by our method, a stronger input-preprocessing technique might be adopted to eliminate our generated verification samples. Specifically, the adversary may choose to replace all queries’ background with an arbitrary one to evade our background-based triggers. However, we note that the backgrounds play an important role in object recognition [47], arbitrarily replacing the background image which is a part of the semantic context would be harmful to the model’s performance, thus this attack is actually ineffective in removing our watermark unless compromise for poor benign accuracy.

To verify this, we randomly select 500 images from ImageNet validation set and replace their background with a random background image chosen from BG-20K dataset [23]. We assume the adversary has two strategies: 1) select a background image from a different category and replace the original background, which can remove our verification triggers successfully. However, our results in Table 10 show that although our validation accuracy degrades, the benign sample accuracy also drops drastically (68.2% → 27.8%), which makes the model useless. This is because the randomly selected background might inadvertently, but seriously changes the prediction of benign samples. 2) select a background image from the same category and replace the original background, which preserves the benign accuracy well. This strategy indeed to some extent preserves the benign accuracy (68.2% → 56.6%) since backgrounds are quite similar and the resulted image is contextually harmonious. However, our results in Table 10 show that for search-based and generation-based schemes, the effectiveness of our watermark is still high. We owe this to our semantic-aware trigger mechanism. The PTYNet learned to generalize well between similar backgrounds and this could be helpful to recognize similar background triggers.

F.2 Ambiguity Attack

In an ambiguity attack, the adversary trains their own PTYNet and injects it into the stolen model. After this, there are two watermarks in the model, which makes it ambiguous to claim ownership. Note that the adversary has no knowledge of our PTYNet’s training dataset and labels. To defeat this, as suggested in [18], we can fine-prune [29] the stolen model with exactly the data we used to train our own PTYNet. After this, our watermark would be preserved well since fine-pruning with clean data does not harm the watermark it was trained. However, the piracy watermark would be removed because it was trained on another dataset made up by the adversary, therefore its corresponding neurons would not be activated during fine-pruning and thus pruned. The adversary is also unable to claim ownership via our watermark, since the verification triggers and dataset we use to train our PTYNet are unknown. In this context, these secrete information play a similar role to cryptographic keys in cryptography.

G DISCUSSION

Our method achieves competitive performance in terms of functionality preservation, effectiveness, and robustness on challenging dataset ImageNet. Extensive experimental results on real-world DNN models also demonstrated the potential application of our proposed method deployed in real scenarios. However, there are also some limitations of our proposed method. The fixed and generation-based strategies for selecting background as a trigger pattern are not as ideal as the search-based strategy. The main reason lies in that the fixed strategy failed in satisfying the class-consistent requirement while the generation-based is limited by the quality of synthesized background images which could be mitigated by employing advanced generative models. Additionally, our method is sensitive to the removal attack by employing input preprocessing, especially the input rotation and image scaling. We can apply data augmentation in the PTYNet training to enhance the robustness against such input preprocessing. This reminds us that such removal attack without involving model modification is more practical which calls for more effective defense approaches in IP protection as unseen attacks will emerge inadvertently. Besides, in our current design, a backdoor is intentionally inserted into the original model through the injection of PTYNet, and exploited through activating it in the model (via background-based triggers), however, we could also extract a unique fingerprint from PTYNet and leverage it for ownership verification, which may be our future work.

https://github.com/Derpinott/VGGVox-PyTorch