# How to Prove Your Model Belongs to You: A Blind-Watermark based Framework to Protect Intellectual Property of DNN

Zheng Li
School of Computer Science and Technology
Shandong University, China
zheng.li@mail.sdu.edu.cn

Chengyu Hu*
Key Laboratory of Cryptologic Technology and
Information Security, Ministry of Education
School of Cyber Science and Technology
Shandong University, China
hcy@sdu.edu.cn

Yang Zhang
CISPA Helmholtz Center for Information Security
Saarland Informatics Campus, Germany
yang.zhang@cispa.saarland

Shanqing Guo*
Key Laboratory of Cryptologic Technology and
Information Security, Ministry of Education
School of Cyber Science and Technology
Shandong University, China
guoshanqing@sdu.edu.cn

## ABSTRACT

Deep learning techniques have made tremendous progress in a variety of challenging tasks, such as image recognition and machine translation, during the past decade. Training deep neural networks is computationally expensive and requires both human and intellectual resources. Therefore, it is necessary to protect the intellectual property of the model and externally verify the ownership of the model. However, previous studies either fail to defend against the evasion attack or have not explicitly dealt with fraudulent claims of ownership by adversaries. Furthermore, they can not establish a clear association between the model and the creator's identity.

To fill these gaps, in this paper, we propose a novel intellectual property protection (IPP) framework based on blind-watermark for watermarking deep neural networks that meet the requirements of security and feasibility. Our framework accepts ordinary samples and the exclusive logo as inputs, outputting newly generated samples as watermarks, which are almost indistinguishable from the origin, and infuses these watermarks into DNN models by assigning specific labels, leaving the backdoor as the basis for our copyright claim. We evaluated our IPP framework on two benchmark datasets and 15 popular deep learning models. The results show that our framework successfully verifies the ownership of all the models without a noticeable impact on their primary task. Most importantly, we are the first to successfully design and implement a blind-watermark based framework, which can achieve state-of-art performances on undetectability against evasion attack and unforgeability against fraudulent claims of ownership. Further, our framework shows remarkable robustness and establishes a clear association between the model and the author's identity.

## CCS CONCEPTS

• **Security and privacy → Software and application security**; *Systems security.*

## KEYWORDS

intellectual property protection, neural networks, blind watermark, security and privacy

**ACM Reference Format:**
Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. 2019. How to Prove Your Model Belongs to You: A Blind-Watermark based Framework to Protect Intellectual Property of DNN. In *2019 Annual Computer Security Applications Conference (ACSAC '19), December 9–13, 2019, San Juan, PR, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3359789.3359801

## 1 INTRODUCTION

Intellectual Property (IP) refers to the protection of creations of the mind, which have both a moral and commercial value. IP is protected under the law framework in the form of, for example, patents, copyright, and trademarks, which enable people to earn recognition or financial benefit from what they invent or create. However, it is not always easy to protect intellectual property. Nowadays, counterfeiters and infringers often exploit procedural loopholes and utilize advanced techniques, such as reverse engineering, to invalidate legitimate patents and trademarks.

Deep learning techniques have witnessed tremendous development during the past decade, and are adopted in various fields ranging from computer vision [3, 11, 12, 15, 24, 25] to natural language processing [4, 28, 29]. However, they are facing serious deep learning privacy and security problems [14, 23]. It is a non-trivial task to build an effective model, especially at a production level.

*Corresponding author.

*ACSAC '19, December 9–13, 2019, San Juan, PR, USA*
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-7628-0/19/12…$15.00
https://doi.org/10.1145/3359789.3359801

Extensive computing power, large datasets, and human expertise are required. Therefore, protecting the intellectual property of a model is essential to maintain the creator's competitive advantage.

The necessity of intellectual property protection for deep learning models has raised attention worldwide. On November 1, 2018, the European Patent Office reviewed guidelines on the patentability of AI and machine learning technologies [2, 6]. Other major political entities are also taking steps to formulate relevant policies. However, protecting the IP of models faces difficulties. A model owner can only rely on the legal system, the process of which is lengthy and expensive. And following the current rules of IP protection, the proof of model ownership requires technical means.

To this end, digital watermark which has been widely applied to protect multimedia content is introduced to the IP protection of deep learning models. An effective watermarking mechanism needs to satisfy multiple requirements including **Fidelity**, **Effectiveness**, **Integrity**, **Security**, **Legality** and **Feasibility**. However, none of the existing methods for watermarking deep learning models can meet all of the above requirements. In another way, the extension of watermarking protection technology to deep learning is still in its infancy.

## 1.1 Related Works

Researchers have proposed approaches facilitating watermarking deep learning model for protecting intellectual property.

Uchida et al. [26] proposed a framework to watermark models for the first time in a white-box way. They assumed the owner can access the target model directly, including the model parameters, to verify the ownership. However, the stolen model is usually deployed as a remote service, which indicates that the model owner is actually unable to access the model parameters.

Rouhani et al. [22] proposed a watermark methodology that meets both the white-box and black-box requirements. They selected a pair of random images and random labels as the watermark, which is also called key samples. Zhang et al. [30] proposed a similar watermarking method while they employed other multiple types of watermarks. Adi et al. [1] chosen a set of abstract images with pre-defined labels as a watermark. Guo et al. [10] proposed a digital watermark technique by adding a message marks associated with the signature to the original images as the watermark. One obvious drawback is that the distribution of their key samples is distant from the origin. An attacker can easily build a detector to evade identification by detecting the key samples, thus avoiding the detection of model theft. Another attack is that the above watermarking methods are also vulnerable to the threat of fraudulent claims because the feature distribution of the key samples is significant and striking, an attacker can easily build a set of fake samples, coincidentally making the model behave as if it were real. We discuss the two vulnerabilities as our main motivations in section 3.

Namba et al. [20] proposed a watermarking method that can defend against evasion attacks. They selected a set of original samples as a watermark from the training set with label change. Although this approach is promising, it is as incapable of establishing a clear association between the model and the creator's identity as most of the existing methods [1, 22, 26].

## 1.2 Our Contribution

Therefore, we propose a novel IPP framework based on blind-watermark for watermarking deep neural networks that meets all the requirements of an effective watermarking mechanism. Our contributions in this paper are three-fold:

- We propose the first blind-watermark based IPP framework aiming to generate the key samples of which the distribution is similar to the original samples, and clearly associate the model with an actual creator's identity.
- We implement a prototype of our IPP framework and evaluate it on two benchmark image datasets and 15 popular classification DNN models. Extensive experiments show that our framework is effective to verify the ownership without significant side effects on primary tasks.
- We conduct extensive empirical validations to show that our IPP framework achieves state-of-art performances on undetectability, unforgeability, and robustness against several forms of attacks.

# 2 BACKGROUND

In this section, we introduce the relevant background knowledge about deep neural networks and digital watermarks, which are closely related to our work.

## 2.1 Deep Neural Networks (DNNs)

Deep neural networks are a crucial component of state-of-the-art artificial intelligence services, showing a level of exceeding humans in various tasks such as visual analysis, speech recognition, and natural language processing. They have dramatically changed the way we conceive software and quickly became a universal technology, and more importantly, it is significantly better than the most advanced machine learning algorithms previously used in these areas.

Although deep neural networks have made significant progress in various fields, it is still a non-trivial task to build deep learning models, especially a production-level model. We need to utilize (1) a large-scale labeled training dataset that can completely cover potential scenarios. (2) a lot of computing power, exceptionally high-performance devices such as GPU, TPU, etc. (3) long-term training to update the parameters of the neural network. (4) the corresponding domain expertise and engineering knowledge to design the network structure and select the hyperparameters. Consequently, building a well-trained model constitutes an important part of the owner's IP, and it is essential to design an intellectual property protection technique to maintain the owner's competitive advantage and economic benefits.

## 2.2 Digital Watermark

The digital watermark is a brand-new information hiding technology in the last twenty years. It is to embed the identification information (i.e., a digital watermark) directly into digital carriers (including multimedia, documents, software, etc.) without affecting the characteristics of the original.

Digital watermark can typically be divided into non-blind watermark and blind watermark, the former refers that it is perceived or noticed by a Human Visual System (HVS). In contrast, blind watermark is usually invisible or imperceptible. Due to the feature distribution of key samples generated by the existing watermark method varies greatly, previous studies based on non-blind watermark either fail to defend against the evasion attack or have not explicitly dealt with fraudulent claims of ownership by adversaries. Therefore, we need to design a new watermark framework based on blind-watermark to embed the watermark into the neural network model.

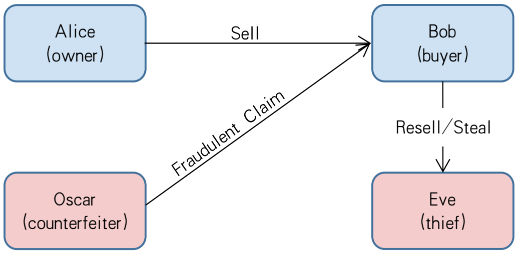


**Figure 1: The threat of security and legality**

## 3 MOTIVATION

In addition to the above limitations of previous works we addressed, in this section, we mainly discuss the two forms of attack: evasion attack and fraudulent claims of ownership. Figure 1 illustrates the threat.

### 3.1 Security: Evasion Attack

Suppose there is a model owner Alice, a thief Eve, and a model buyer Bob. Two possible scenarios can lead to evasion attack: (1) Eve has stolen the model in some way, (2) Bob has resold the model without Alice's authorization. Both of these behaviors are detrimental to the owner's interests. If the model is well-watermarked, the model owner Alice would successfully verify the ownership of the suspected model by issuing prediction queries of key samples.

To evade the verification by the legitimate owners, Eve will attempt to build a detector to detect whether the queried sample is a clean one or a possible key sample [14]. Once the detector decides the queried instance is a possible key sample, the stolen model would return a random label from its output space. However, all the existing watermark methods [1, 10, 22, 30] are susceptible to the above form of attack. The distribution of the key samples and the original samples varies greatly, once the model owner issues prediction of these key samples, Eve can easily detect the watermarks by utilizing a sample detector. Hence, in the following sections, a blind-watermark based IPP framework is presented that is aiming to defend against the evasion attack. We conduct extensive experiments, and the results demonstrate that our novel IPP framework can achieve state-of-art performances on undetectability against evasion attack.

### 3.2 Legality: Fraudulent Claims of Ownership

Suppose there is a counterfeiter, Oscar, who tries to illegally claim the ownership of the model. This behavior will not only infringe on Bob's interests but will even infringe on Alice's interests, which will directly lead to the invalidation of the IPP technology — the model owner Alice is no longer the only one that can claim the ownership. Counterfeiters will try to make a set of key samples by himself, that is to say, design a set of fake samples that can induce the behavior of the licensed model to achieve the purpose of falsehood.

In the watermarking method proposed by Zhang et al. [30], the three types of perturbations superimposed to the ordinary samples is so obvious and striking that Oscar can easily superimpose the same perturbations to the ordinary samples to obtain the key samples. Due to the intrinsic generalization and memorization capabilities of deep neural networks, the newly generated samples can still be identified and responded with predefined labels [30]. In the approaches of Adi et al. [1] and Rouhani et al. [22], the space of abstract images and random images is so large, Oscar can easily obtain another set of images, for example, generated by computer script or genetic algorithms [9]. Furthermore, the counterfeiters even can monitor and intercept key samples on the communication channel of the network.

In this paper, we make an assumption that the hyper-parameters of the training setting, the parameters, and the architecture of encoder are confidential. Therefore it is impossible for an attacker to get the same watermark generation strategies. Further encouragingly, even we relax this key assumption, our IPP framework still achieves a remarkable performance on unforgeability against fraudulent claims of ownership.

## 4 WATERMARKING NEURAL NETWORK

In this section, we first briefly introduce the overview of embedding and verification of watermarking model. Hereafter, we introduced the details of the implementation of our IPP framework, including the objective functions and the watermarking algorithm.

### 4.1 Tasks I: Embedding

To protect the intellectual property of our model, we tried to watermark the model to leave the backdoor. In the embedding procedure, we need a set of labeled samples, which is also known as the key samples $x^{key}$, to be a watermark. The key sample $x^{key}$ is generated by an encoder $e$ which accepts the original samples $x$ and logo $l$ as inputs. In this paper, we hope that the distribution of the key sample $x^{key}$ is as close to that of the original samples $x$. To this end, we present a novel generation algorithm $\mathcal{G}$ to achieve it:

$$\begin{aligned} x^{key} &= \mathcal{G}(e, x, l), \qquad (1) \\ x^{key} &\to x \end{aligned}$$

Then we adopt an embedding algorithm $\mathcal{E}$ to embed a watermark to model $f$:

$$f_k = \mathcal{E}(f, x^{key}) \qquad (2)$$

The resultant model $f_k$ (i.e., the watermarked model) will predict a query of key sample $x^{key}$ to a pre-defined label $t^{key}$. Details of each algorithm are explained in the next section.

### 4.2 Task II: Verification

Consider a scenario that the model owner suspects that the model deployed remotely violates its copyright interest. To confirm the ownership of the remote DNN, in this procedure, the model owner

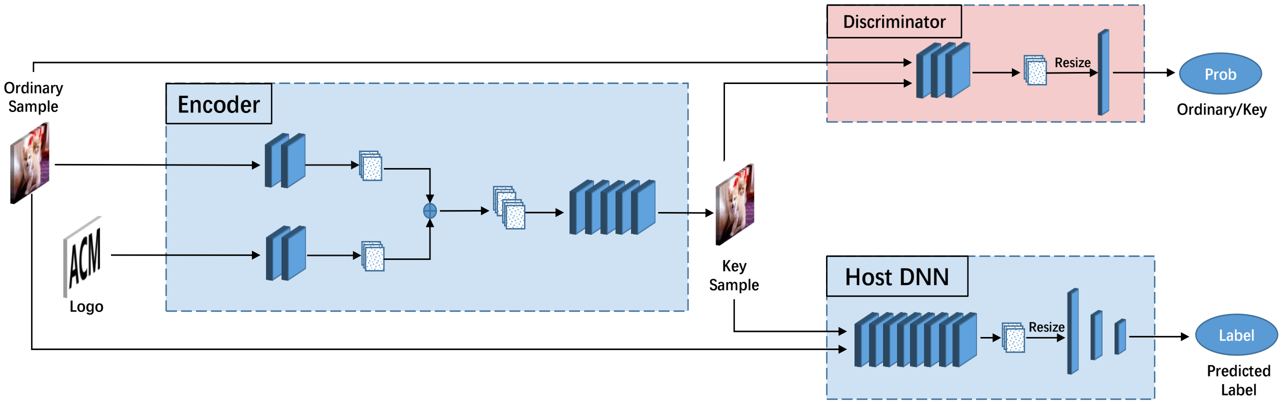


**Figure 2: Workflow of our IPP framework**

first prepares a set of key samples $\{x_1^{key}, x_2^{key}, ...\}$ by a generation algorithm $\mathcal{G}$:

$$x^{key} = \mathcal{G}(e, x, l) \tag{3}$$

Then the model owner will issue a prediction query to the remote model $g$ with these key samples, obtain resulting predictions, and evaluate the accuracy of the resulting predictions over pre-defined labels.

$$acc_g = \mathcal{V}(g, x^{key}, t^{key}) \tag{4}$$

If the $acc_g$ is a value close to 1, or $acc_g > \mathcal{T}_{acc}$ , where $\mathcal{T}_{acc}$ is a threshold parameter close to 1. Then the owner can verify the IP of a suspected model and claim the ownership of the remote model.

## 4.3 Algorithm Pipeline

Figure 2 shows the workflow of our IPP framework, which consists of three parts: encoder, discriminator and host DNN.

**Encoder:** Here our encoder $e$ is essentially a lightweight autoencoder. The encoder accepts the part samples from the training dataset and the exclusive logo as inputs and attempts to output the key samples which are undistinguished from the ordinary samples. We typically denote the parameters of encoder as $\theta_e$, the exclusive logo as $l$, and try to obtain a function $\theta_e(x, l) = x^{key}$ where $x^{key} \to x$ by solving the optimization problem with a batch of $\{x_1, x_2, ..., x_m\}$:

$$\underset{\theta_e}{\text{argmin}} \frac{1}{m} \sum_{i=1}^{m} (x_i - \theta_e(x_i, l))^2 \tag{5}$$

The above term is the reconstruction error for the encoder $e$.

Due to the limited capacity of the encoder, it is impossible to achieve the goal of perfect reconstruction. Moreover, compared to perfect reconstruction, we hope that the distribution of key samples generated by the encoder only needs to be as close as that of ordinary samples, i.e., $x \approx x^{key}$, not $x = x^{key}$. The tiny difference between the key samples and the ordinary samples is exactly what we need — the magnitude of fluctuation achieves a comparable trade-off between security and effectiveness, the smaller preserving better security against evasion attack and the larger providing better effectiveness of watermarking DNN.

From a mathematical perspective, in order to prove whether the objective function 5 is exact to achieve the goal, i.e., $x \approx x^{key}$, not $x = x^{key}$, we denote the distribution of original samples from training dataset as $P_{data}(x)$, and the distribution of key samples produced by the encoder as $P_e(x^{key}; \theta_e)$. The objective function can be formalized as follows:

$$\begin{aligned}
\underset{\theta_e}{\text{argmax}} \prod_{i=1}^{m} P_e(x_i; \theta_e) =& \underset{\theta_e}{\text{argmax}} \log \prod_{i=1}^{m} P_e(x_i; \theta_e) \\
=& \underset{\theta_e}{\text{argmax}} \sum_{i=1}^{m} \log P_e(x_i; \theta_e) \\
=& \underset{\theta_e}{\text{argmax}} \mathbb{E}_{x \sim P_{\text{data}}} [\log P_e(x; \theta_e)] \\
=& \underset{\theta_e}{\text{argmax}} \int_x P_{data}(x) \log P_e(x; \theta_e) dx \\
& - \int_x P_{data}(x) \log P_{data}(x) dx \\
=& \underset{\theta_e}{\text{argmin}} KL\,(P_{data}(x) \| P_e(x; \theta_e))
\end{aligned} \tag{6}$$

where KL is the Kullback-Leibler divergence. The above objective function is essential to minimize the Kullback-Leibler divergence which is a measure of how one probability distribution diverges from another. Further derivation:

$$KL(P_{data} \| P_e) = -H(P_{data}) + H(P_{data}, P_e) \tag{7}$$

The former of equation 7 represents the information entropy of $P_{data}$, the latter is the cross entropy of $P_{data}$ and $P_e$. That is to say, minimizing the KL divergence is equivalent to minimizing the cross entropy. At the same time, the objective function 5 is actually the cross entropy of the empirical distribution and the gaussian model [7], while we cannot determine which distribution $P_{data}$ and $P_e$ obey. In order to solve the objective function 6, we adopt the negative sampling approach [19]. Furthermore, the equation 5 only punishes the larger error of the corresponding pixels of the two images, and ignores the underlying structure of the image. So we introduce the structural similarity index (SSIM) [27], and the objective function can be formalized as follows:

$$\underset{\theta_e}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} (1 - SSIM(x_i, \theta_e(x_i, l))) \tag{8}$$

**Discriminator:** The negative sampling approach targets a different objective than the original function 6. We typically interpret this objective as a binary classification problem. The discriminator efforts to determine whether the samples are synthesized or extracted from the ordinary samples and the encoder converges to capture the distribution of given samples. The keypoint of the discriminator is essentially the theory proposed in generative adversarial networks [8], which is well-designed to minimize the KL divergence exactly. The discriminator also acts as a detector to detect if input data is generated by the encoder.

We denote the distribution of training dataset $P_{data}(x)$ as a positive sample, that of the key samples $P_e(x^{key}; \theta_e)$ as a negative sample. In order to speed up the learning process, we adopt $x$ and $x^{key}$ to present the distribution $P$ which is unable to obtain. The discriminator $d$ accepts the original samples $x$ or key samples $x^{key}$ as inputs, outputting a binary classification probability to indicate whether the input data comes from the ground truth. The discriminator outputs a conditional probability $p(\Theta|\chi; \theta_d)$ ($\chi \in \{x, x^{key}\}$) which is modeled as logistic regression:

$$p(\Theta|\chi; \theta_d) = \frac{1}{1 + e^{-\theta_d(\chi)}} \tag{9}$$

where we use $\theta_d$ to represent the discriminator and adopt a random variable $\Theta$ to denote the binary answer: $\Theta = 1$ if the input sample is the origin, and $\Theta = 0$ otherwise. **The objective function ($O_d$) of the discriminator is denoted by**:

$$\begin{aligned} \underset{\theta_d}{\operatorname{argmin}} - \frac{1}{m} \sum_{i=1}^{m} \Bigg( \Theta \log \frac{1}{1 + e^{-\theta_d(\chi_i)}} + \\ (1 - \Theta) \log(1 - \frac{1}{1 + e^{-\theta_d(\chi_i)}}) \Bigg) \end{aligned} \tag{10}$$

After the optimal $\theta_d$ is given, we simultaneously train the encoder $e$ to maximize the probability of $d$ making a mistake. Then, the new objective function of encoder $e$ is:

$$\underset{\theta_e}{\operatorname{argmin}} - \frac{1}{m} \sum_{i=1}^{m} \log \frac{1}{1 + e^{-\theta_d(\theta_e(x_i, l))}} \tag{11}$$

We train the discriminator $d$ and the encoder $e$ iteratively to obtain the final optimal $\theta_d^*$ and $\theta_e^*$. Competition in this procedure derives both teams to improve their ability until the $P_e(x^{key}; \theta_e)$ is indistinguishable from the genuine $P_{data}(x)$. Note that, both of the objective function 5, 8 and 11 regularize the encoder $e$ by encouraging the distribution $P_e(x^{key}; \theta_e)$ to match the given distribution $P_{data}(x)$. Next, we introduce how to induce the host model to mispredict a key sample to a pre-defined label.

**Host DNN:** Due to the capacity limitations of the encoder $e$, the distribution of key samples generated from encoder can only be "close" to that of ordinary samples. That is, there must be a "some" difference between $x^{key}$ and $x$. Hence, we utilize this difference to induce the host DNN ($\theta_h$) to correctly identify $x^{key}$ to a pre-defined label under the premise of correctly identifying $x$. In the image classification task, it is common to employ the softmax function at the final layer to obtain the probability vector:

$$softmax(\theta_h(x))_i = \frac{e^{\theta_h(x)_i}}{\sum_j^n e^{\theta_h(x)_j}} \tag{12}$$

Where $j$ denotes the $j$-th element of the output vector of $n$-class. Given the original sample $x$ with normal label $t$, and key sample $x^{key}$ with a pre-defined label $t^{key}$, we briefly define $\mathcal{D} = \{(x_1, t_1), ..., (x_1^{key}, t_1^{key}), ...\}$, then **the objective function ($O_h$) of the host model $h$ is denoted by**:

$$\underset{\theta_h}{\operatorname{argmin}} - \frac{1}{m} \sum_{(\chi, \tau) \in \mathcal{D}}^{m} \left( \log \frac{e^{\theta_h(\chi)_\tau}}{\sum_j^n e^{\theta_h(\chi)_j}} \right) \tag{13}$$

Note that, the equation 13 of which $x^{key} = \theta_e(x, l)$ with label $t^{key}$ can also be formalized to update $\theta_e$ as follows:

$$\underset{\theta_e}{\operatorname{argmin}} - \frac{1}{m} \sum_{i=1}^{m} \left( \log \frac{e^{\theta_h(\theta_e(x_i; l))_{t_i^{key}}}}{\sum_j^n e^{\theta_h(\theta_e(x_i; l))_j}} \right) \tag{14}$$

By adding all the equations 5, 8, 11 and 14 together, we have **the following objective function ($O_e$) for encoder $e$**:

$$\begin{aligned} \underset{\theta_e}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \Bigg( \alpha(x_i - \theta_e(x_i, l))^2 + \beta(1 - SSIM(x_i, \theta_e(x_i, l)) \\ + \gamma(-\log \frac{1}{1 + e^{-\theta_d(\theta_e(x_i, l))}}) + \delta(-\log \frac{e^{\theta_h(\theta_e(x_i; l))_{t_i^{key}}}}{\sum_j^n e^{\theta_h(\theta_e(x_i; l))_j}}) \Bigg) \end{aligned} \tag{15}$$

where $\alpha, \beta, \gamma, \delta > 0$ are the hyper-parameters to trade-off between four parts. The 4th term regularizes the encoder $e$ by encouraging the generated key samples misclassified to pre-defined labels more easily. The global watermarking algorithm is as follow:

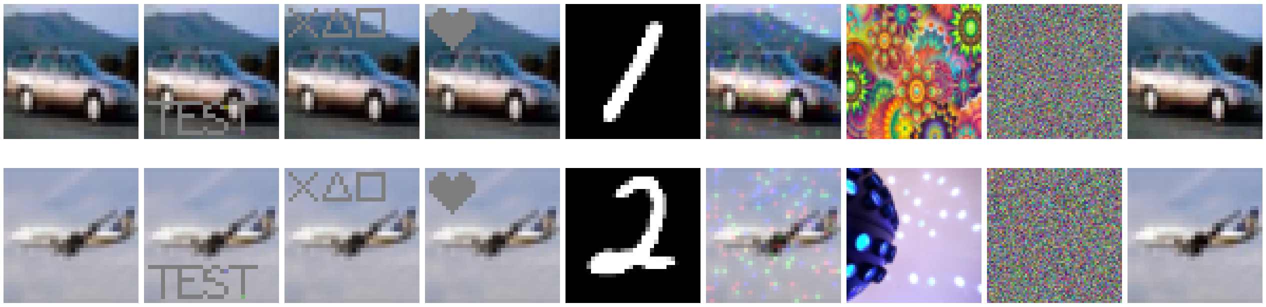
**Algorithm 1:** Minibatch gradient descent training of watermarking algorithm.

```
Input: training set $D$, logo $l$, hyper-parameters $\alpha, \beta, \gamma, \delta$,
       minibatchsize $m, n$, sampling number $k_{1,2}$;
Output: $\theta_e, \theta_h, \theta_d$;
Initialize original samples $D'$ sampled from $D$ randomly;
For convenience, objective function is denoted by $O_{e,d,h}$;
for number of training epochs do
    for $i = 1; i \le k_1; i + +$ do
        sample minibatch of m original samples from $D'$;
        update $\theta_d$ by descending its adam gradient: $\nabla_{\theta_d} O_d$;
        update $\theta_e$ by descending its adam gradient: $\nabla_{\theta_e} O_e$;
    end
    for $i = 1; i \le k_2; i + +$ do
        sample minibatch of n sample from D;
        merge minibatch of $m + n$;
        update $\theta_h$ by descending its stochastic gradient:
         $\nabla_{\theta_h} O_h$;
    end
end
return $\theta_e, \theta_h, \theta_d$;
```

**Table 1: Details of the DNNs and datasets used to evaluate our IPP framework**

| Dataset | Dataset Description | Host DNN Architecture | Host DNN Description | Test Acc.# |
|---|---|---|---|---|
| MNIST[5] | Hand-written digits | LeNet-1[18] | A classic CNN Architecture | 98.73% |
| | | LeNet-3[18] | | 98.86% |
| | | LeNet-5[18] | | 98.92% |
| CIFAR-10[17] | A collection of General images | VGG-11[24] | Very Deep Convolutional Networks for Large-Scale Image Recognition | 91.40% |
| | | VGG-13[24] | | 93.47% |
| | | VGG-16[24] | | 93.08% |
| | | VGG-19[24] | | 92.86% |
| | | ResNet-18[11] | Deep residual learning for image recognition | 94.62% |
| | | ResNet-34[11] | | 94.80% |
| | | ResNet-101[11] | | 93.97% |
| | | PreActResNet-18[12] | Identity Mappings in Deep Residual Networks | 94.43% |
| | | PreActResNet-34[12] | | 94.95% |
| | | GoogleNet[25] | Going Deeper with Convolutions | 94.38% |
| | | DPN-26[3] | Dual Path Networks | 94.53% |
| | | MobileNetV2[15] | Efficient CNN for Mobile App | 91.35% |

# The accuracy is obtained from regular test set in unwatermarked setting.



(a) Original (b) "test" [30] (c) "symbol" [30] (d) "heart" [30] (e)"unrelated" [30] (f) "mask" [10] (g) "abstract" [1] (h) "random" [22] (j) Ours

**Figure 3: The examples of key samples of existing watermark methods and our framework**

# 5 IMPLEMENTATION

## 5.1 Datasets and DNNs

As a proof-of-concept, we adopt two benchmark datasets with different types of data—MNIST and CIFAR-10—and implement our IPP framework on a total of fifteen host DNNs. We provide a summary of the two datasets and the corresponding DNNs in Table 1.

**MNIST** [5] is a large handwritten digital dataset containing 28 × 28 pixel images with class labels from 0 to 9. The dataset consists of 60,000 training samples and 10,000 test samples. Each pixel value is within a grayscale between 0 and 255.

**CIFAR-10** [17] is labeled subsets of the 80 million tiny color images dataset, consisting of 50,000 training images (10 classes, 5,000 images per class) and 10,000 test images (10 classes, 1000 images per class). All the images are normalized and centered in a fixed-size image of 32 × 32 pixels.

To evaluate our blind-watermark based IPP framework, we use 1% of total training samples for the key sample generation, and for each image, we randomly select a target label. The encoder and discriminator are trained iteratively using Adam algorithm [16] ($\beta_1 = 0.5, \beta_2 = 0.999$) with a mini batch-size of 20 and a fixed learning rate of 0.001. The host DNNs are simultaneously trained using stochastic gradient descent [13] with a batch-size of 120 (100 original samples and 20 key samples) and a declining learning rate of 0.1, which is decayed by 0.1 per 40 epochs. we perform grid search to find the optimal hyper-parameters $\alpha = 3, \beta = 5, \gamma = 0.1, \delta = 0.01$ and train a prototype of our framework for 100 epochs with $k_{1,2} = 1$. We implement all the experiments[1] by Pytorch [21], on a Ubuntu 18.04 server with a Tesla K80 GPU card.

[1] Source code is available at https://github.com/zhenglisec/Blind-Watermark-for-DNN

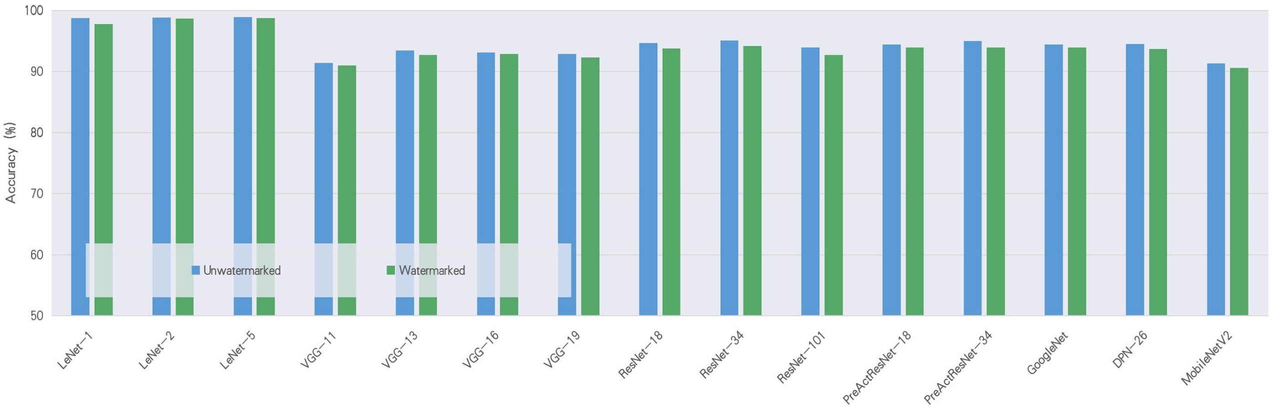


**Figure 4: Accuracy of different models on regular test set**

## 5.2 Results

The training of our IPP framework has been successfully implemented, and we compare the key samples generated by our framework with the existing methods. Figure 3 shows examples of key samples created from CIFAR-10 dataset. Figure 3 (a) are two examples of original images; Figure 3 (b)-(h) are key samples generated by methods proposed in [1, 10, 22, 30]. Figure 3 (j) are two key samples generated by our blind-watermark based IPP framework. As we can see, the examples of ours are so similar to the original samples that the differences between them are too tiny to be seen by humans. In contrast, the examples of other existing methods are visible and striking, which indicates the distribution of the features of them is distant from the feature distribution of the training samples. In next section, we conduct extensive empirical validations to show that our IPP framework satisfies multiple requirements.

# 6 EVALUATION

We analyze the performance of our IPP framework by measuring the following criteria: **fidelity**, the side effect made to the primary classification task; **effectiveness and integrity**, whether it can successfully verify the ownership of the host DNN; **security**, the ability of defending against evasion attack; **legality**, the ability of anti-counterfeiting; **feasibility**, The ability to resist model modifications and whether it explicitly associates the model with the identity of the actual creator.

## 6.1 Fidelity

Fidelity requires our IPP framework to watermark a host model without significant side effects on the primary task. Ideally, a well-watermarked model should be as accurate as an unwatermarked model. To measure the side effects on the primary task, we implemented a comparative evaluation of the accuracy between the clean model and watermarked model. As depicted in Figure 4, all the evaluated models are trained on the test set in two different settings: unwatermarked setting and watermarked setting. We first train a model without watermark embedding and evaluate it on the test set that it has not seen before. Then we implement our framework to watermark the same model and evaluate it on the test set.

The results expressly demonstrate that all the watermarked models still have the same level of accuracy as the unwatermarked model. The accuracy drops by up to 0.66% on average. In the best case, we achieve a drop of only 0.14%. That is to say, the side effects caused by our IPP framework are entirely within the acceptable performance variation of the model and has no significant impact on the primary task. Thus our framework meets the fidelity requirement.

## 6.2 Effectiveness and Integrity

The purpose of effectiveness is to measure whether we can successfully verify the copyright of the target DNN model under the protection of our IPP framework. Ideally, a well-watermarked model should identify key samples and predict them to the predefined labels with high accuracy. The integrity requires that our IPP framework shall not falsely claim the authorship of unwatermarked models. To measure the effectiveness and integrity, we implement another comparative evaluation of the accuracy between the unwatermarked model and the watermarked model. We typically denote the forward inference function of the unwatermarked model by $\theta_{uw}$ and that of the watermarked model by $\theta_w$. Only if $\theta_{uw}(x^{key}) \neq t^{key}$ and $\theta_w(x^{key}) == t^{key}$, we confirm that our IPP framework can successfully verify the ownership. We issue prediction queries of key samples and tests whether the model returns correct labels specified by key samples.

Figure 5 shows the accuracy of the different models implemented in our evaluation. We use 1% of dataset to test the accuracy. The "unwatermarked" shows the accuracy of key samples that are not induced into the unwatermarked model. All the unwatermarked models achieve an accuracy of 9%—13%, a totally random guess. In contrast, the "watermarked" encouragingly shows the accuracy of key samples exceeds over 90%. In the best case, the watermarked model even achieve an accuracy of 100% on the key samples. The results show that our framework can successfully verify the ownership without falsely claiming the authorship of unwatermarked models. Thus the effectiveness and the integrity requirements are met.

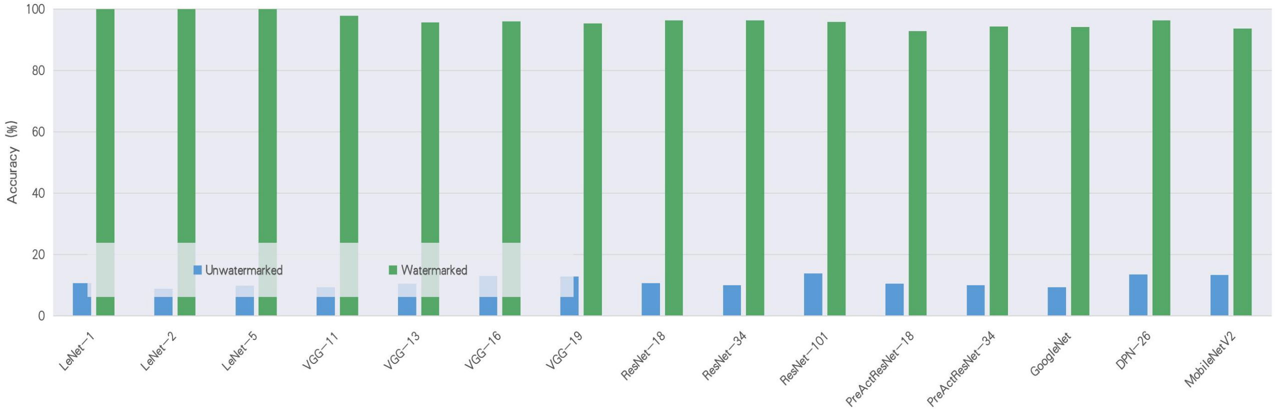


**Figure 5: Accuracy of different models on our key samples**

## 6.3 Security

Security requires that the verification process is imperceptible and undetectable, which can resist identification and detection by unauthorized service providers. In this section, we introduce a novel watermark invalidation method, evasion attack [14], and demonstrate that an attacker can evade the verification by the model owners. Considering that the premise of illegal service provision is due to the limitation of data resources or computing power, we make an assumption that an attacker can access to 10% of total training samples for detector build-up. Essentially, the binary classification is supervised learning. They can generate the same amount of key samples according to the proposed methods in [1, 10, 22, 30] and our framework, then get the training set for detector build-up. We typically design a simple detector based on 3 fully-connected layers, which are composed of 512, 256 and 1 neural unit followed by sigmoid activation. We use $\theta_d$ to represent the parameters of the detector, and the objective function is denoted by:

$$\begin{aligned}\underset{\theta_d}{\operatorname{argmin}} -\frac{1}{m}\sum_{i=1}^{m}\log\frac{1}{1+e^{-\theta_d(x_i)}}- \\ \frac{1}{m}\sum_{i=1}^{m}\log(1-\frac{1}{1+e^{-\theta_d(x_i^{key})}})\end{aligned} \tag{16}$$

We apply stochastic gradient descent with a fixed learning rate of 0.01 for 20 epochs to solve the objective function 16. We adopt the ROC (receiver operating characteristic), which reports the relation between true positive rate and false negative rate over multiple thresholds, as the evaluation metric. Figure 6 presents the trained 3-layers detector's performance over the queried samples. The ROC curve of our IPP framework demonstrates that the performance of the detector is close to random guessing with an AUC (area under the ROC curve) of 0.5. In contrast, the ROC curve of previous methods demonstrates that the detector has a performance on AUC of well above 0.94, which indicates that the unauthorized service provider is enough to evade the verification.

A further assumption is that the attacker tries to build a more powerful detector based on the weights transferred from the stolen model. In addition to the last several fully connected layers, most of the current classification models are playing the role of feature extractors. Therefore, we utilize the former layers of ResNet-18 as a feature extractor, which is then followed by a fully-connected layer with one output. We trained the detector in the same setting as the 3-layer detector and report the results in Figure 7. As we can see, the

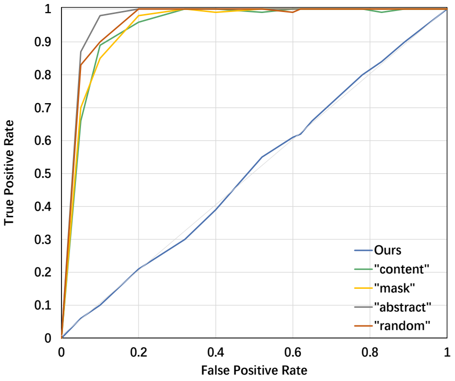


**Figure 6: The receiver operating characteristic (ROC) curve produced by the detector based on 3 linear layers**

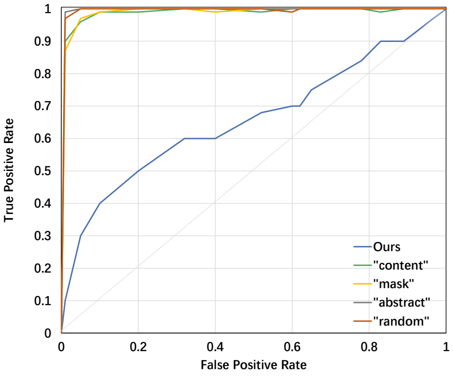


**Figure 7: The receiver operating characteristic (ROC) curve produced by the detector based on ResNet-18**

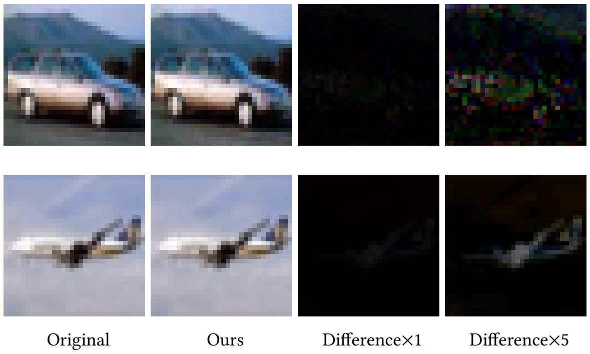


**Figure 8: The examples of difference image**

detection performance has indeed increased. The ROC curves of the existing methods show that the key samples used in their methods are more easily detected with an AUC of 0.98. Encouragingly, the ROC curve of our framework demonstrates that the detector is only slightly more effective than random guessing with an AUC of 0.65. This result convincingly shows that our IPP framework can achieve remarkable performances on undetectability against evasion attack.

In addition to evasion attack, we also consider another type of attack: removing the backdoor-based watermark. For example, an attacker can fine-tune the stolen model to achieve the purpose of removing the watermark. More often in practice, it is common to fine-tune the existing state-of-the-art models on new insufficient datasets to achieve higher performance or to implement new tasks. As for an authorized user, fine-tuning the model does not mean that he wants to launch this type of attack. Hence, we discuss the behavior of the fine-tuning model in detail in section 6.5, which is regarded as a test of the robustness of our scheme.

## 6.4 Legality

Here, we consider an attack scenario in which the counterfeiter knows that the model purchased by Bob is watermarked and attempts to claim ownership of the model illegally. This behavior will not only infringe on Bob's interests but will even infringe on Alice's benefits, which will directly lead to the invalidation of the IPP technology — the model owner Alice is no longer the only one that can claim the ownership. The counterfeiter attempts to designing a set of fake samples, which can induce the abnormal behavior of the licensed model. Therefore, the goal of legality is to resist the fraudulent claims of ownership by adversaries. In this paper, we study two different types of fraudulent claims of ownership.

**What if the ordinary samples and the key samples became accessible?** Although the ROC curve of our framework demonstrates that the detector based on ResNet-18 is only slightly more effective than random guessing with an AUC of 0.65, the counterfeiter can actually detect a small number of key samples. Therefore, we assume that what if the original and key samples became accessible by intercepting the communication channel? What could then be ascertained about the intercepted samples? In Figure 3 (b)—(d), the features of superposed images are so prominent and striking that the counterfeiter can easily generate a set of fake samples by adding them to other original samples. Therefore, we apply the same logo "TEST" to other samples to generate a set of new key samples, and then issue prediction queries of the new key samples to the watermarked model, obtained an average accuracy higher than 91%. In contrast, as depicted in Figure 8, the features of the difference images from our framework are too subtle for the human to observe distortion, and we magnify all difference images by five times. It can be found that the distortion mode made upon each original sample is unique, and the distribution of distortions is related to the properties (e.g., complexity and texture) of the original samples. This result indicates that it is impossible to design a set of fake samples by superposing the difference images to other new original samples.

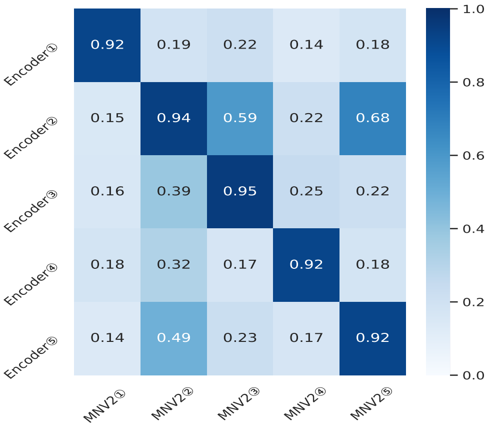


**Figure 9: The performance (accuracy) of our key samples transferring attack**

**What if the encoder was leaked?** In most cases, it can safely be assumed that access to the learned encoder directly is impossible for an attacker. However, what if the attacker trained a "same" encoder by using the same architecture, dataset, and hyper-parameters? To test this attack, we implement our IPP framework to watermark MobileNetV2 for five times with different seeds, then we get five identical pairs of encoder and MobileNetV2 (MNV2), which are numbered ①, ②, ③, ④ and ⑤. Figure 9 depicts the key samples transferring attack's performance. The x-axis represents the host model being attacked, and the y-axis represents the key samples generated by the trained encoder. Concretely, we issue queries to each MobileNetV2 with the key samples generated by each encoder to evaluate the accuracy of predicting the pre-defined labels. As we can see, the high accuracy of the attack results is listed at the diagonal of Figure 9, which shows that the key samples can only induce their corresponding model to pre-defined labels. The reason why it can't transfer is that the initialization of the neural network is an important part of the training process, which will have an important impact on the performance, convergence, and convergence speed of the model. Random initialization and stochastic gradient descent can cause the objective function to find a new local minimum (sometimes adjacent local minimum, resulting in slightly higher transferring attack performance, e.g., 68%), which means that the resultant model is different each time. This result

**Table 2: Robustness for model fine-tuning: the trend of watermarking accuracy with fine-tuning epochs**

| epochs | V-13 | V-16 | R-18 | R-34 | PreActR-18 |
|---|---|---|---|---|---|
| 0 | 95.75% | 96.50% | 97.00% | 93.40% | 91.75% |
| 10 | 92.50% | 95.50% | 92.00% | 90.70% | 90.75% |
| 20 | 90.25% | 95.25% | 91.75% | 89.75% | 90.00% |
| 30 | 90.00% | 95.50% | 90.50% | 88.75% | 89.25% |
| 40 | 90.00% | 95.20% | 89.75% | 88.50% | 88.50% |
| 50 | 90.00% | 95.75% | 89.50% | 87.75% | 88.25% |
| 60 | 90.00% | 95.25% | 89.00% | 87.00% | 88.25% |
| 70 | 90.15% | 95.00% | 88.00% | 86.75% | 88.00% |
| 80 | 90.00% | 95.25% | 87.75% | 86.25% | 87.50% |
| 90 | 90.25% | 95.50% | 87.50% | 85.50% | 87.50% |
| 100 | 90.00% | 95.50% | 87.50% | 84.75% | 87.00% |

shows that the encoder, as well as the host model, can't be exactly reproducible with different random seeds for initialization, not to mention that it is almost impossible for an attacker to get the same architecture, dataset, and hyper-parameters. For more results of the transferring attack, see Figures in Appendix A.

The above results hint at an advantage of learned encoders: unlike static watermark generation algorithms, they can employ a unique watermark generation strategy each time. Our IPP framework undoubtedly performs remarkable unforgeability against the fraudulent claims of ownership.

## 6.5 Feasibility

We consider two aspects of the feasibility: robustness and functionality. The purpose of robustness is to measure whether our framework is robust to the model modification, and the purpose of functionality is to measure whether our framework can associate the host model with the author's identity.

**Robustness** Training a high-performance model from scratch requires a lot of resources, and insufficient datasets can significantly affect the performance of the model. Fine-tuning is a common strategy in practice. When there is not enough training data, we can fine-tune pre-trained models to complete new tasks or achieve higher performance. Therefore, an attacker can fine-tune the stolen model with fewer new datasets to obtain a new model that inherits the performance of the stolen model but is also different from the stolen model.

In this experiment, we divide the testset into two halves. The first half (80%) is used for fine-tuning pre-trained models while the second half (20%) is used for evaluating new models. We use watermark (key samples) accuracy of new models to measure the robustness of our framework for modifications caused by fine-tuning. Table 2 shows that even after 100 epochs, our framework still has a high accuracy of all the models (only drop 9.50% in the worst case). The reason behind this is fine-tuning can lead to an adjacent local minimum, which means weights of the model don't change significantly. This result indicates our framework can perform remarkable robustness against model modification. Note that, we don't consider such modification that can cause significant side effects on primary tasks, which leads a totally different model. We agree with [10] that the "proof-of-authorship" during actual model usage and "proof-of-origin" of a model should be two different problems.

**Table 3: A summary of all methods meets the requirements**

| Requirements | [1, 22] | [10] | [20] | [26] | [30] | Ours |
|---|---|---|---|---|---|---|
| Fidelity | √ | √ | √ | √ | √ | √ |
| Effectiveness | √ | √ | √ | √ | √ | √ |
| Integrity | √ | √ | √ | √ | √ | √ |
| Security | | | √ | √ | | √ |
| Legality | | √ | | √ | | √ |
| Feasibility | | √ | | | √ | √ |

**Functionality** The functionality requires that our IPP framework shall clearly associate the host model with the author's identity. In a highly competitive global marketplace, the exclusive logo is the most widely used, the most frequent and the most important element in the process of corporate image transmission. However, none of the existing methods take it into account, and they do not closely associate the host model to be protected with the identity of the individual or organization. Essentially, we take advantage of the intrinsic over-fitting of the neural networks to achieve the goal, where we turn the weakness into a strength. Over-fitting is a modeling error that occurs when a function is too closely fit for a limited set of data points. In our framework, the host model is over-fitting for the key samples generated by encoder, and the encoder is over-fitting for the exclusive logo. This indicates that only the logo that participates in the training will lead a watermarked DNN to exhibit the desired behavior. That is to say, our IPP framework can clearly associate the host model with the author's identity.

To the best of our knowledge, we are the first to take the usage of the logo into consideration and successfully design and implement our IPP framework. Furthermore, our framework also meets the requirements of white-box and black-box.

# 7 CONCLUSION AND FUTURE WORK

In this paper, we propose the first blind-watermark based IPP framework aiming to generate the key samples of which the distribution is similar to the original samples. We successfully design and implement our IPP framework on two benchmark datasets and 15 popular deep learning models. We conduct extensive experiments to show that our framework is adequate to verify the ownership without significant side effects on primary tasks and achieves a remarkable performance on undetectability against evasion attack and unforgeability against fraudulent claims of ownership. Besides, our framework shows remarkable robustness against model modification. Lastly, we are the first to take the usage of the logo into consideration and establish a clear association between the model and the creator's identity. In briefly, our IPP framework meets all the requirements summarized in Table 3.

For future work, we plan to protect the intellectual property of the speech recognition model. We expect this work to be expanded to other forms of deep learning, e.g., recurrent neural networks, etc.

## 8 ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their comments on previous drafts of this paper. This work is partially supported by National Natural Science Foundation of China (91546203), the Key Science Technology Project of Shandong Province (2015G GX101046), Major Scientific and Technological Innovation Projects of Shandong Province, China (2018CXGC0708, 2017CXGC0704).

## REFERENCES

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In *Proceedings of the 2018 USENIX Security Symposium (USENIX Security)*. USENIX.

[2] AI patent 2018. *EPO guidelines on patentability of artificial intelligence and machine learning.* Retrieved November 8, 2018 from http://patentblog.kluweriplaw.com/2018/11/08/epo-guidelines-on-patentability-of-artificial-intelligence-and-machine-learning/

[3] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. 2017. Dual Path Networks. In *Proceedings of the 2017 Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 4467–4475.

[4] Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine learning Research* (2011).

[5] Corinna Cortes, Yann LeCun, and Christopher JC Burges. 1998. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[6] EPO guidelines 2018. *Artificial intelligence and machine learning.* Retrieved November 1, 2018 from https://www.epo.org/law-practice/legal-texts/html/guidelines2018/e/g_ii_3_3_1.htm

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. *CoRR abs/1406.2661* (2014).

[9] Jia Guo and Miodrag Potkonjak. 2017. Pruning Filters and Classes:Towards On-Device Customization of Convolutional Neural Networks. In *Proceedings of the 2017 1st International Workshop on Deep Learning for Mobile Systems and Applications (DLMSA)*. ACM, 13–17.

[10] Jia Guo and Miodrag Potkonjak. 2018. Watermarking deep neural networks for embedded systems. In *Proceedings of the 2018 International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–8.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 770–778.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. In *Proceedings of the 2016 European Conference on Computer Vision (ECCV)*. IEEE, 630–645.

[13] Robbins Herbert and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics* (1951).

[14] Dorjan Hitaj and Luigi V Mancini. 2018. Have You Stolen My Model? Evasion Attacks Against Deep Neural Network Watermarking Techniques. *CoRR abs/1809.00615* (2018).

[15] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR abs/1704.04681* (2017).

[16] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR abs/1412.6980* (2014).

[17] A. Krizhevsky and G. Hinton. 2009. Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/cifar.html. Journal of Computer Science Department, University of Toronto, Tech. Rep.

[18] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Institute of Electrical and Electronics Engineers* (1998).

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionally. In *Proceedings of the 2013 Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 3111–3119.

[20] Ryota Namba and Jun Sakuma. 2019. Robust Watermarking of Neural Network with Exponential Weighting. *CoRR abs/1901.06151* (2019).

[21] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. https://pytorch.org/. Workshop on NIPS.

[22] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2018. DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models. *CoRR abs/1804.00750* (2018).

[23] Ahmed Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. 2018. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. *CoRR abs/1806.01246* (2018).

[24] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR abs/1409.1556* (2014).

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, ScottE Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1–9.

[26] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and ShinâĂŹichi Satoh. 2017. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 International Conference on Multimedia Retrieval (ICMR)*. ACM, 269–277.

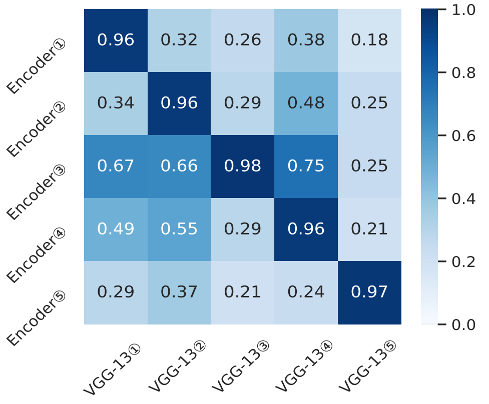
[27] Zhou Wang. 2004. Image quality assessment: from error visibility to structural similarity. *Transactions on Image Processing* (2004).

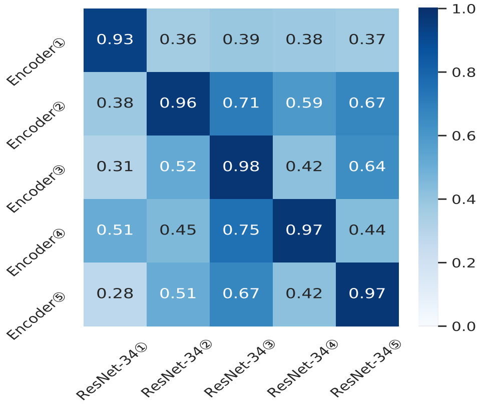
[28] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liuand AÌŁAÌĺukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. GoogleâĂŹs Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR abs/1609.08144* (2016).

[29] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Michael L Seltzer, Andreas Stolcke, Dong Yu, , and Geoffrey Zweig. 2016. Achieving Human Parity in Conversational Speech Recognition. *CoRR abs/1610.05256* (2016).

[30] Jialong Zhang, Zhongshu Gu, Jiyong 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 ACM Asia Conference on Computer and Communications Security (ASIACCS)*. ACM, 159–172.

## A THE PERFORMANCE OF TRANSFERRING ATTACK ON VGG-13 AND RESNET-34


Encoder①
Encoder②
Encoder③
Encoder④
Encoder⑤
VGG-13①
VGG-13②
VGG-13③
VGG-13④
VGG-13⑤


Encoder①
Encoder②
Encoder③
Encoder④
Encoder⑤
ResNet-34①
ResNet-34②
ResNet-34③
ResNet-34④
ResNet-34⑤