Robust Watermarking for Deep Neural Networks via Bi-level Optimization

Peng Yang, Yingjie Lao, Ping Li
Cognitive Computing Lab
Baidu Research
10900 NE 8th St. Bellevue, WA 98004, USA
{pengyang5612, laoyingjie, pingli98}@gmail.com

Abstract

Deep neural networks (DNNs) have become state-of-the-art in many application domains. The increasing complexity and cost for building these models demand means for protecting their intellectual property (IP). This paper presents a novel DNN framework that optimizes the robustness of the embedded watermarks. Our method is originated from DNN fault attacks. Different from prior end-to-end DNN watermarking approaches, we only modify a tiny subset of weights to embed the watermark, which also facilities better control of the model behaviors and enables larger rooms for optimizing the robustness of the watermarks.

In this paper, built upon the above concept, we propose a bi-level optimization framework where the inner loop phase optimizes the example-level problem to generate robust exemplars, while the outer loop phase proposes a masked adaptive optimization to achieve the robustness of the projected DNN models. Our method alternates the learning of the protected models and watermark exemplars across all phases, where watermark exemplars are not just data samples that could be optimized and adjusted instead. We verify the performance of the proposed methods over a wide range of datasets and DNN architectures. Various transformation attacks including fine-tuning, pruning and overwriting are used to evaluate the robustness.

1. Introduction

Along with the unprecedented progress of deep neural network (DNN), both the networks and application tasks have become increasingly sophisticated, making the models costly to build. As a result, DNN models are considered as valuable assets, which demand a means for projecting the intellectual property (IP) of model builders. To address this, several DNN watermarking or fingerprinting approaches have been proposed recently [1, 30, 19, 18, 26, 27, 9, 6, 7, 16, 32, 11, 28, 5, 13]. Conceptually, watermarking is achieved by injecting certain behavior into the model whose presence can be easily verified later, typically, by using several key samples. In the *black-box* setting, watermarking will associate desired predictions to the key samples that are different from naturally trained models (e.g., by using backdoor) to reduce the false positive rate (i.e., probability of the presence of the watermarking in a naturally trained model) [1, 30, 19, 26, 10]. While in the *white-box* setting that demands full access to the model including latent feature maps for extracting the watermark, the desired behavior can be embedded into the internal structure or latent space of a DNN model [27, 9, 7].

In contrast to these prior DNN watermarking methods that rely on end-to-end retraining or fine-tuning of the key samples with desired labels, we propose a novel framework that only requires to modify an extremely small amount of parameters for embedding a watermarking. In other words, instead of only constraining the selection of key samples, we further constrain the parameter modifications in the watermarking process. We consider the black-box setting, i.e., watermark extraction can be performed by remote querying using the model prediction API. By leveraging techniques from fault attacks [20, 4, 22, 14, 31, 29], we first search for parameters that have large magnitudes of gradients with respect to key samples while close to zero-valued gradients with respect to natural inputs. Only modifying these weights enables us to freely improve the robustness of the watermarks without affecting the normal behavior with respect to natural inputs. We then formulate the robust optimization and active learning to enhance the robustness of the model behavior with respect to key

samples after embedding. Although the watermarks in prior works have shown to withstand certain attacks [1, 30, 19, 18, 26, 27, 9, 6, 7, 16, 32, 11, 28], robustness is not an underlying optimization objective in their embedding processes. Thus, we argue that our framework provides more potential towards robust DNN watermarking. The proposed method also significantly reduces the watermarking overhead, as only a very small portion of the network requires modification. Besides, similar to the underlying assumption of fault attacks, our proposed method also enables watermarking to already deployed DNN models.

The contributions of this paper are as follows:

- We propose an effective and efficient bi-level optimization framework for DNN watermarking that generates robust exemplars and embeds the watermark concurrently, as opposed to prior methods that consider them as two separate processes.
- We enhance the robustness by formulating the watermarking as two alternative optimization phases: the inner loop phase optimizes the example-level problem to generate robust exemplars according to the predictive confidence towards the current hypothesis, while the outer loop phase proposes a masked adaptive optimization to achieve the robustness of the projected DNN models.
- We conduct extensive experiments to evaluate the proposed algorithms on various DNN models (e.g., VGG-9, VGG-16, and Inception-V3) and compare to prior works. The promising results demonstrate the effectiveness and robustness of proposed watermarking methodology.

2. Preliminaries

DNN Watermarking. Existing DNN watermarking techniques can be categorized as while-box DNN watermarking and black-box watermarking. White-box DNN watermarking can fully access the DNN models, thus enabling a flexible watermark embedding and extraction process. A pioneering work of white-box algorithm was proposed in [27], which explicitly embeds the watermark into any layers of a DNN model. Several works in this category also proposed methods to iteratively embed the signatures into the zero-impact regions of intermediate feature maps [9, 6, 7]. A more advanced method embeds the watermarks by exploring the information of the "passport" layers, where the corresponding parameters are important to preserve the model functionality [11]. In contrast, since the structures of DNN models are, however, invisible to the black-box watermarking, such methods may bring an unexpected modification to the learned function with respect to natural inputs when injecting the desired behavior. Existing approaches mostly exploited backdoor attacks to force a trigger into the model, so that it might bias the functional regions of a DNN model and lead to a performance degradation [1, 30, 19, 26]. A recent technique leverages "null-embedding" to generate the triggers that are irrelevant to the classification function, which improves the piracy-resistance of the watermarks [18].

In this paper, we advance state-of-the-art by developing a novel bi-level optimization framework that perform robust exemplar generation and watermark embedding concurrently for enhancing the fidelity and robustness of the watermark. The proposed method only requires to modify a tiny amount of parameters, which is another key advantage compared to prior embedding methods that rely on end-to-end training.

Fault Attacks on DNN. In addition to algorithmic adversarial attacks [21, 25, 15, 3, 2], hardware-oriented vulnerabilities have also been shown to pose serious threats against DNN systems [8, 20, 22]. Among these, fault attacks are capable of catastrophically degrading the inference accuracy by directly injecting faults into DNN model parameters. These attacks typically search for the most vulnerable weights/bits that can significantly degrade the inference accuracy. Many recent works have shown to be able to destroy a DNN model, i.e., drastically reduce the inference accuracy, by flipping only a few bits in memory cells [4, 14, 31, 22]. The impact of faults injected into the activation function of DNN to manipulate the label of a specific input has also been well studied [4]. We adapt the techniques from fault attacks for embedding watermarks.

3. Threat Model

The signature embedding process can be conducted by the model builder or a trusted party. Without loss of generality, we assume a pre-trained model could be received from a model builder who build the model architecture F and corresponding parameters Θ_{pre} with the training dataset D_{tr}, and a held-out validation dataset D_v for evaluating the performance. We then apply the proposed methodology to this DNN model to embed a desired watermark. Only the legitimate model owner know the specific embedded watermark.

An adversary might apply transformation attacks in an attempt in removing the embedded but unknown watermark of a DNN model. The attacks include model compression, model fine-tuning, and watermark overwriting, which should retain the original functionality of the DNN model. In other words, the attacker's goal

is still to use the model while avoiding IP tracing so that the evaluation performance should not be drastically failed. We assume the attacker is able to fully access to the model but has no knowledge of the embedded watermark.

For a suspect model, the model owner can verify the presence of the watermark by using the key samples via the prediction API. If the returned signature is the same or very close to that of the model owner, this suspect model is likely pirated from the legitimate model owner. Subsequently, the model owner may take legal actions after collecting other evidences of this matter.

4. Problem Setting and Methodology

Given a pre-trained model parameter Θ_{pre}, the goal of watermarking is to generate key samples D_{wm} and embed them successfully without adjusting the parameters that are relevant to the inference performance for normal input data. Specifically, the key samples D_{wm} have to satisfy two criteria: 1) *Manipulation on Labels*: The labels of key samples should be easy to be manipulated by the authenticated DNN model; 2) *Original Function Preservation*: The process of key embedding should have no negative impacts on the original functionality of the DNN model. To meet the criterion, we exploit the prediction entropy, which measures the uncertainty or confidence inherent in the model prediction. We choose the samples with high entropy as the key samples, since these samples are near to decision boundary, and the model could easily manipulate their labels with a slight modification, which have few impacts on the pretrained DNN model.

By utilizing the concept from fault attacks that search for parameters to modify, we propose an effective bi-level optimization framework for DNN robust watermarking problem. Due to the difference in the settings between attack and defense as elaborated in the threat model, watermarking possesses different constraints and requirements compared to fault attacks. Having these in mind, we formulate the watermarking as two alternative optimization phases: the inner loop phase optimizes the example-level problem to generate robust exemplars according to the current hypothesis, while the outer loop phase deploys a masked adaptive optimization for watermarking. Our method is able to find the optimal solution in the trade-off space between the watermarking and functionality.

4.1. Global Bi-level Optimization Schema

As illustrated in Figure 1, the proposed robust watermarking training alternates the learning of predictive models and robust exemplars across all phases, where robust exemplars are not just key samples but could be optimized and adjusted instead. We formulate this alternative learning with a global bi-level optimization schema composed of model-level and exemplar-level problems, and carefully derive the solutions.

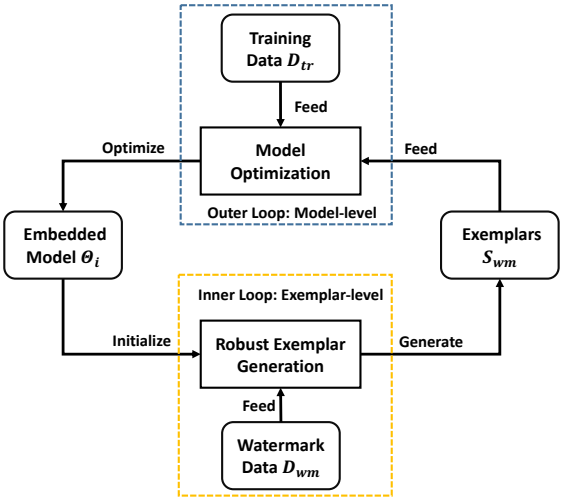


Figure 1. Bi-level optimization schema.

In watermark embedding, the protected model is incrementally learned in each phase on the union of watermark exemplars and training data. In turn, based on this model, the watermark exemplars (i.e., the parameters of the exemplars) are adjusted (or learned) before embedding into the protected models. In this way, the objective of watermarking derives a constrain to optimize and adjust the exemplars, and vise versa. We propose to formulate this relationship under a global bi-level optimization schema, in which each phase uses the optimal model to optimize watermark exemplars, and vice versa.

Specifically, in the i-th phase, the proposed system aims to learn a model Θ_i to approximate the ideal authenticated model parameters Θ_i^*, which is to achieve a trade-off between the prediction on natural input D_{tr} and the recognition on watermarks D_{wm}, i.e.,

$$\Theta_i^* = \arg\min_{\Theta_i} \; L_c(\Theta_i; D_{tr} \cup D_{wm}),$$

where the objective function aims to balance the mistake on the ownership identification and the mistake on model predictive function, while the $L_c(\cdot)$ denotes the loss function for classification or regression tasks.

Since the key samples D_{wm} are required to be embedded into the model, we seek to generate the boundary exemplars S_{wm} that maximize the identification loss on D_{wm}. In this way, the exemplars S_{wm} are treated as the "worst cases" of D_{wm}. We formulate this with the global bi-level optimization problem, where "*global*" means operating through all phases, as fol-

lows,

$$\Theta_{i+1} = \arg\min_{\Theta_i} L_c(\Theta_i; D_{tr} \cup S_{wm})$$
$$\text{s.t.} \quad S_{wm} = \arg\max_{S_{wm}} L_c(\Theta_i; D_{wm}). \tag{1}$$

Θ_{i+1} is the optimal solution on the union of S_{wm} and D_{tr}. It reduces the bias caused by natural input D_{tr}, meanwhile enforcing the exemplars S_{wm} embedded into the current models. In the rest of the paper, the problem (1) for solving Θ and S_{wm} are called *model-level* and *exemplar-level* problems, respectively.

4.2. Model-level Problem (Outer Loop)

As illustrated in Figure 1, in the i-th phase, we first solve the *model-level* problem with the natural data and watermarks as the input, and use Θ_i as the model initialization. According to problem (1), the objective function can be expressed as

$$L_{all} = \lambda L_c(\Theta_i; D_{tr}) + (1-\lambda) L_c(\Theta_i; S_{wm}), \tag{2}$$

where $L_c(\Theta_i; D_{tr})$ denotes the prediction loss on D_{tr}, $L_c(\Theta_i; S_{wm})$ the identification loss on S_{wm}, and $\lambda \in [0, 1]$ is a trade-off parameter. Let α_1 be the learning rate, then Θ_i is updated with gradient descent as

$$\Theta_{i+1} \leftarrow \Theta_i - \alpha_1 \nabla_\Theta L_{all}.$$

After that, Θ_{i+1} is used to learn the robust exemplars, which is formulated to solve the following problem:

$$S_{wm} = \arg\max_{S_{wm}} L_c(\Theta_{i+1}; D_{wm}),$$

which is equivalent to optimize and adjust the exemplars with the identification loss of Θ_{i+1} on D_{wm}.

4.3. Exemplar-level Problem (Inner Loop)

Existing methods [1, 16] authenticate the ownership of a model by utilizing a few watermark examples. However, there is no guarantee particularly towards whether these watermarks are robustly embedded. In contrast, this approach explicitly aims to ensure a feasible approximation of that assumption, thanks to the differentiability of the exemplars.

To achieve this, we train a temporary model $\Theta_i^{'}$ using S_{wm} to maximize the identification loss on D_{wm}, for which we use D_{wm} to compute a validation loss to adjust the parameters of S_{wm}. As illustrated in Figure 2, the entire problem is thus formulated in a local bi-level optimization schema, where "*local*" means within a single phase, as

$$S_{wm} = \arg\max_{S_{wm}} L_c(\Theta^{'}(S_{wm}); D_{wm})$$
$$\text{s.t.} \quad \Theta^{'}(S_{wm}) = \arg\min_{\Theta} L_c(\Theta; S_{wm}). \tag{3}$$

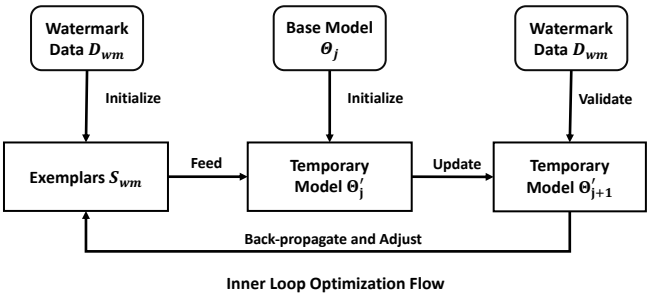


Figure 2. Inner loop optimization flow.

Solving Eq. (3) is a process of moving S_{wm} towards the decision boundary, and yielding a small loss on D_{wm}. Through embedding exemplars S_{wm} into the model, it could result in robust identification of D_{wm}.

Optimizing S_{wm}: The training flow is present in Figure 2. First, the image-size parameters of S_{wm} are initialized by the subset of D_{wm}. Then, we initialize a temporary model $\Theta^{'}$ with Θ_i obtained in outer loop, and train $\Theta^{'}$ for a few iterations by gradient descent on S_{wm}:

$$\Theta_{j+1}^{'} \leftarrow \Theta_j^{'} - \alpha_2 \nabla_{\Theta_j^{'}} L_c(\Theta_j^{'}; S_j), \quad \text{s.t. } \Theta_0^{'} = \Theta_i, \tag{4}$$

where α_2 is the learning rate of fine-tuning temporary models, and j is the iteration number in the inner loop optimization. As the $\Theta_j^{'}$ and S_j are both differentiable, we are able to compute the loss of $\Theta_j^{'}$ on D_{wm}, and back-propagate this validation loss to optimize S_j,

$$S_{j+1} \leftarrow S_j + \beta_1 \nabla_{D_{wm}} L_c(\Theta_j^{'}; D_{wm}), \text{ s.t. } S_0 \subset D_{wm}, \tag{5}$$

where β_1 is the learning rate. In this step, we basically need to back-propagate the validation gradients till the input layer, through rolling all training gradients of model weights $\Theta_j^{'}$. Since the batch size of S_j should be different from D_{wm}, it is impossible to directly update S_j with the gradients on D_{wm}. To address this issue, the gradient on D_{wm} would be clustered and reshaped with the same size as S_j.

4.4. Masked Adaptive Optimization

The embedding process typically requires a retraining process, which, however, leads to expensive computation costs particularly for DNNs with huge numbers of parameters. Moreover, optimizing all the parameters may greatly affect the original functionality. To this end, we adapt the concept of fault attacks and propose a masked optimization for watermarking process.

To preserve model functionality, our algorithm utilizes a mask to perform the embedding, so that the essential parameters of model functions could be frozen when embedding watermarks on parameter space Θ.

When learning the model Θ, we update the parameters with a mask M, instead of directly optimizing all the parameters. During the training, both prediction loss and watermarking loss (refer to Eq. (2)) are used. Let $\odot$ denote the element-wise product, the objective function Eq. (2) in this paper can be formulated as:

$$L_{all} = \lambda L_c(M \odot \Theta; D_{tr}) + (1-\lambda) L_c(M \odot \Theta; S_{wm}).$$

Specifically, we locate the most effective parameters in the DNN model to be optimized for watermark embedding. The algorithm aims to find the parameters on which the weight update could mostly manipulate the labels of key samples ($\mathcal{S}_{wm}$) while preserving the original predictions on natural inputs ($\mathcal{D}_{tr}$). To achieve this goal, the mask is generated via the observation of the gradient of Θ on D_{tr} and S_{wm}. Generally, the candidate parameters should have large gradient values over S_{wm}, but close to zero gradient values over D_{tr}. Formally, the mask, defined as C, can be computed as

$$\begin{aligned} C &= H_s \cap H_t \\ \text{s.t. } H_s &= \text{Top}_N \left\{ \frac{1}{|S_{wm}|} \sum_{(\mathbf{x}_s, y_s) \in S_{wm}} |\nabla_\Theta \ell(f(\mathbf{x}_s), y_s))| \right\}, \\ H_t &= \text{Top}_N \left\{ -\frac{1}{|D_{tr}|} \sum_{(\mathbf{x}_t, y_t) \in D_{tr}} |\nabla_\Theta \ell(f(\mathbf{x}_t), y_t)| \right\} \end{aligned} \tag{6}$$

To this end, we prioritize the top-N parameters of Θ according to the ranking, and then optimize the model with the masked gradient descent:

$$\begin{aligned} &\Theta = \Theta - \alpha_1 \left[\mathbf{M} \odot \nabla_\Theta L(\Theta; S_{wm}) + \overline{\mathbf{M}} \odot \nabla_\Theta L(\Theta; D_{tr}) \right] \\ &\text{s.t. } [\mathbf{M}]_k = \begin{cases} 1, & k \in C \\ 0, & k \notin C \end{cases}, \ [\overline{\mathbf{M}}]_k = \begin{cases} 0, & k \in C \\ 1, & k \notin C \end{cases} \end{aligned} \tag{7}$$

We observe that the hard-mask $\mathbf{M}$ exploits the gate mechanism, which enables an adaptive optimization over a partial of neural structures.

4.5. Algorithm

Algorithm 1 summarizes the overall process of our bi-level optimization schema. We point out several advantages in this algorithm: a) To preserve the function of the DNN model, we choose a few layers to update the weight parameters, instead of all parameters. We fine-tune the several layers of the DNN structure for watermarking, as shown in Steps 10-13. b) We reduce the learning bias via balancing sample sizes between the watermarking and training samples. As shown in Step 9, the watermark batch in S_{wm} should be comparable with that in S_{tr}.

Algorithm 1 Bi-level Optimization for DNN watermarking

1: **Input:** Data D_{tr}, D_{wm} and Model Θ_{pre}
2: **Output:** Authenticated DNN model Θ_{wm}
3: **for** $i = 1, \ldots, N$ (**Outer Loop**) **do**
4: If $i = 0$, Initialize weights $\Theta^{'} = \Theta_{pre}$, otherwise, $\Theta^{'} = \Theta_{i-1}$; Initialize $S_0 \subset D_{wm}$;
5: **for** $j = 1, \ldots, M$ (**Inner Loop**) **do**
6: Adjust weight $S_j^{'}$ using $\Theta^{'}$ by Eq. (5) ;
7: Update weight $\Theta^{'}$ using $S_j^{'}$ by Eq. (4) ;
8: **end for**
9: Initialize $S_{wm} = S_M^{'}$ and sample $S_{tr} \subset D_{tr}$;
10: **for** any layer l in the DNN model **do**
11: Compute C_l by Eq. (6) ;
12: Update $[\Theta_i]_l$ with the mask by Eq. (7) ;
13: **end for**
14: **end for**

5. Experiments

5.1. Experimental Setup

5.1.1 Base Models

In our experiments, we train LeNet5 [17] for MNIST, VGG-9 and VGG-16 [23] for CIFAR10 and CIFAR100, respectively, and Inception-V3 [24] for ImageNet. Each of these models achieved a test accuracy that is consistent with state-of-the-art. Our implementation was based on the PaddlePaddle deep learning platform.

5.1.2 Transformation Attacks

We evaluate the robustness of the proposed methods against the following three widely-used transformation attacks as in prior works.

Fine-Tuning. Fine-tuning can be considered as a transformation attack that an adversary may use to remove the watermark while preserving the model accuracy by retraining part of the network layers with original data (i.e., natural input samples). In our experiment, we fine-tune the watermarked models using the corresponding validation data.

Pruning. Model pruning is a popular technique to compress a well-trained model to accelerate the computation and reduce memory requirement, while preserving of the inference accuracy. An adversary may hope the pruning process to alter the embedded watermarks. We employ the technique from [12].

Watermark Overwriting. Different from the above two, this setting assumes an adaptive and intelligent adversary who has the knowledge of the watermarking technique (but not the specific embedded watermark). To perform such an attack, the adversary

# Keys	*MNIST (LeNet5)*		*CIFAR10 (VGG-9)*		*CIFAR100 (VGG-16)*		*ImageNet (Inception-V3)*	
	R_{auth}	R_{loss}	R_{auth}	R_{loss}	R_{auth}	R_{loss}	R_{auth}	R_{loss}
5	100%	0.00%	100%	0.02%	100%	0.03%	100%	0.07%
10	100%	0.02%	100%	0.05%	100%	0.05%	100%	0.09%
15	100%	0.06%	100%	0.03%	100%	0.07%	100%	0.10%
20	100%	0.07%	100%	0.05%	100%	0.08%	97.5%	0.15%

Table 1. Results on effectiveness and fidelity.

will select a new set of watermark key samples and use the proposed method to embed a second watermark in hopes of overwriting the first watermarking without affecting the inference accuracy. The second watermark in our experiment is selected randomly.

5.1.3 Performance Metrics

Fidelity is characterized by authentication success rate R_{auth}, loss of accuracy R_{loss}, and number of modified parameters. Among these, R_{auth} evaluates the percentage of watermark samples that are embedded successfully into the DNN models. We expect the authentication success rate R_{auth} to be high while function loss rate R_{loss} to be low, so that the watermarked model retains the accuracy on normal test data.

Robustness is evaluated against transformation attacks. We use function preserved rate R_{pres} to quantify the preserved prediction capability, which is evaluated on the validation dataset. The embedded watermark should not be removed when R_{pres} for the natural inputs remains high, and the degradation of R_{auth} should be much smaller than that of R_{pres}.

Capacity represents the amount of information the proposed technique can embed into the target DNN model without violating other requirements.

5.1.4 Parameter Setting

For all experiments, we select top-2.5% of masked parameters (denoted as N in Eq. (6)). As shown in Algorithm 1, we set number of inner loop iteration $M = 3$, and number of outer loop iteration $N = 10$. For each dataset, we use the same learning rate $\alpha_1 = \alpha_2 = \beta_1$ in both exemplar optimization in Eq. (4) and Eq. (5), and model optimization in Eq. (7). Specifically, we set learning rate to be 0.002 in MNIST, and 0.02 in CIFAR10, CIFAR100 and ImageNet. For the number of key samples in D_{wm}, we assign 30 in MNIST, and 60 in CIFAR10, CIFAR100 and ImageNet.

5.2. Results

5.2.1 Fidelity

We run the experiments multiple times and calculate the averaged authentication success rate and function loss rate, which are presented in Table 1. The results show that most of the selected watermark samples have been successfully recognized given various numbers of keys. Specifically, we are able to achieve a high success rate without sacrificing the inference ability of the DNN models. For example, our model can successfully embed all 20 keys into the CIFAR10 with its function loss of less than 0.05%. Moreover, Figure 3 shows the ratio of changed parameters when performing the watermark embedding over these DNN models. We observe that this algorithm only tunes less than 0.005% and 0.025% weights of VGG-16 on CIFAR100 and Inception-V3 on ImageNet, respectively, while achieving a high success rate of embedding and a low inference accuracy loss.

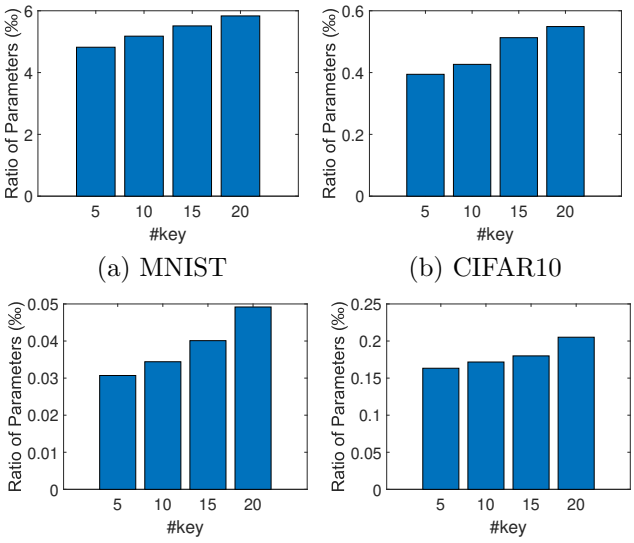
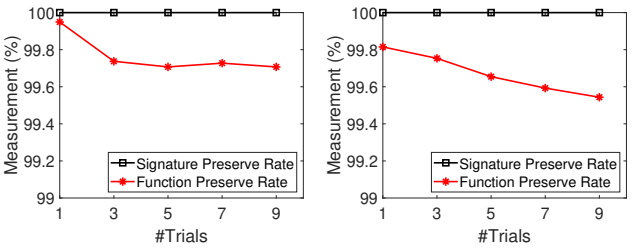


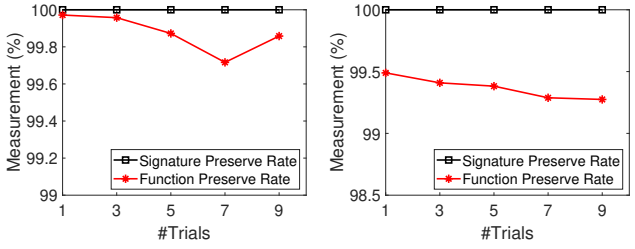
Figure 3. Ratio of changed parameters over the DNN models with respect to different numbers of key samples.

5.2.2 Robustness

Fine-tuning: Figure 4 presents the performance under the fine-tuning process. The results show that our method performs robustly towards the fine-tuning over all datasets. Specifically, although the function preserve rates drop after several trials of fine-tuning, the signature preserve rates still remain the same during the whole process.

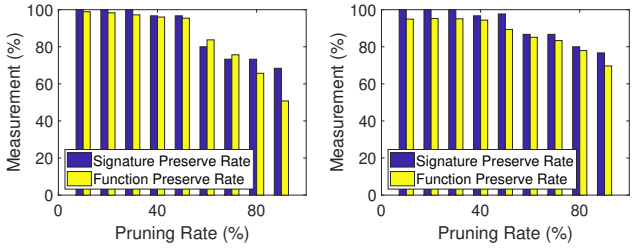


(a) MNIST (b) CIFAR10



(c) CIFAR100 (d) ImageNet

Figure 4. Signature preserving rate and function preserving rate under the process of fine-tuning on validate datasets.

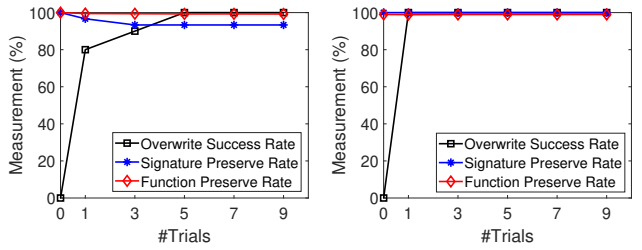


(a) MNIST (b) CIFAR10

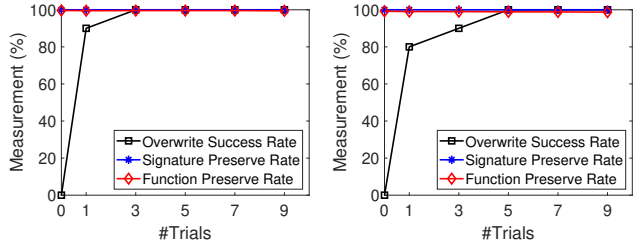
Figure 5. Authentication and function preserve rate under various pruning rates.

Pruning: Figure 5 shows the performance impacts on watermarking embedding and inference ability under an increasing pruning rate. We conduct case studies on MNIST and CIFAR10, as similar results are observed on other datasets. For CIFAR10, even after 50% of pruning rate, the model still remains no loss on identification accuracy. With a higher pruning rate, the inference accuracy starts to drop dramatically. In addition, we notice that our method performs better on a more complex DNN model, which demonstrates its robustness over large parameter spaces.

Watermark overwriting: We evaluate the robustness of our method against the watermark overwriting scenario, where the adversarial seeks to insert additional watermarks into the model in order to disable the recognition of original watermarks. In our experiments, the overwriting attack is performed in two different settings: 1) the same size of new key samples are sampled, and perform the same embedding process



(a) MNIST (b) CIFAR10

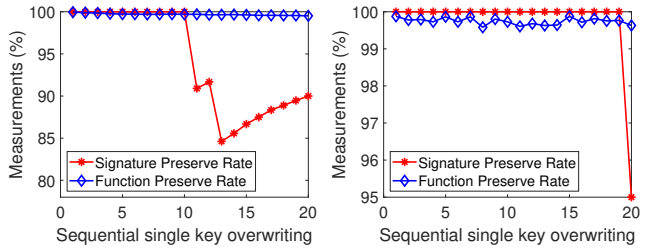


(c) CIFAR100 (d) ImageNet

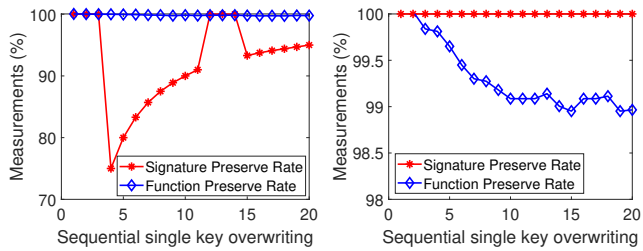
Figure 6. Signature preserving rate and function preserving rate under the process of overwriting.

as the previous key set; 2) key samples are embedded one by one with only one, and a key sample will not be executed until its previous key sample is embedded successfully. Figure 6 presents the performance of conventional overwriting process in the first setting. As shown, the algorithm is consistently robust against the overwriting over all the datasets. Specifically, the watermark embedding is more robust over a more complex DNN structure.

Figure 7 depicts the results of the sequential overwriting process in the second setting, showing that this



(a) MNIST (b) CIFAR10



(c) CIFAR100 (d) ImageNet

Figure 7. Evaluation of sequential single input overwriting.

setting performs relatively unstable compared to overwriting all key samples. It nonetheless still achieves promising performance on CIFAR10/100 and ImageNet with 95% and 100% success rates, respectively.

5.2.3 Capacity

We evaluate the capacity with respect to large numbers of key sample embedding, as shown in Figure 8. Obviously, embedding on more key samples results in lower authentication rates and lower function preserve rate, since algorithms require more modification to the weights. However, due to the proposed masked optimization strategy that only updates the parameter weights with a small impact on the previous learned knowledge, our algorithm can maintain a comparable authentication rate (i.e., more than 94%) and function preserve rate (i.e., around 99.0% for 60 key embedding). It can also be inferred that our method performs more stable on complex datasets.

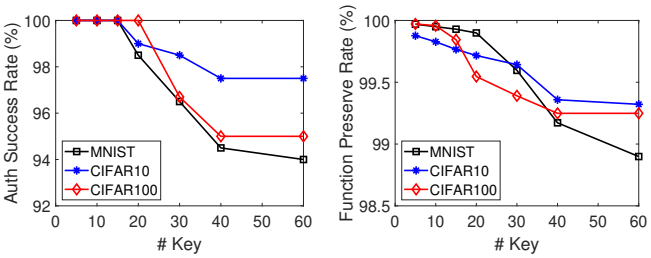


Figure 8. Authentication successful rate and function preserved rate with various numbers of watermark embedding.

5.3. Discussion and Comparison to Prior Works

Based on our experimental results, we can conclude that our embedded watermark satisfies all the requirements for an effective and robust IP protection tool. By leveraging the bi-level optimization strategy, we are able to provably enhance robustness while maintaining an extremely small inference accuracy loss. Besides, our watermarking framework exhibits consistent performance across various DNN architectures on a wide range of datasets.

To further demonstrate the advantage of the proposed approach, we compare with prior DNN watermarking methods from the perspectives of fidelity and robustness. Note that these comparisons may not be 100% fair, as experiments have different settings and employ different architectures and hyperparameters. For the evaluation of robustness, the settings of transformation attacks might also vary largely across different works. This probably is also the reason why we observe most of the existing works even recent ones did not report results of robustness comparison with prior methods [1, 11, 30, 16, 18, 19, 26]. Since the overwriting process is the same as watermarking, which results less variation in evaluation, we compare to prior works that are evaluated against overwriting [27, 9, 1]. As most of these prior works were only evaluated on CIFAR10 and/or MNIST, we take CIFAR10 for comparison, as presented in Table 2.

Method	Setting	R_{loss}	R_{pres}
[27]	White-Box	$\sim 0.3\%$	$70 \sim 96\%$
[9]	White-Box	$\sim 0.5\%$	58%
[1]	Black-Box	$\sim 0.3\%$	95%
Proposed	Black-Box	$\sim 0.05\%$	100%

Table 2. Comparison to prior works on fidelity and robustness against overwriting (20 key samples).

On CIFAR10, our method only has a 0.05% accuracy loss for 20 key samples, while the white-box method in [9] has around a 0.5% accuracy loss and the black-box method (backdoor-based) in [1] yields an about 0.3% accuracy loss under the same number of keys. It is obvious that our method achieves a much better fidelity, which is expected as we only modify an extremely small number of parameters for embedding the watermark and hence have better control to the model behavior. Even for ImageNet, for instance, we achieve a 100% R_{auth} with only a 0.1% accuracy loss.

When comparing the robustness against prior works, the performance of our method is also superior. For example, the number of mismatches after overwriting on CIFAR10 is around 8.5 for 20 key samples in [9], yielding a below 60% signature preserve rate R_{pres}, while our method achieves almost 100% signature preserve rates in all the datasets under both settings of watermark overwriting as described above. While [1] shows a decent performance against overwriting on CIFAR10, it suffered from a significant R_{pres} degradation on CIFAR100, under the same setting of fine-tuning a pre-trained model. In contrast, our method achieves a 100% signature preserve rate for CIFAR100 and even ImageNet, as shown in Figure 6.

6. Conclusion

In this work, we propose to leverage the concept of fault attack to embed watermark into a DNN model for IP protection. By exploiting the capability of embedding the desired behavior with modifying a tiny amount of parameters, we formulate and develop a novel bi-level optimization to enhance the robustness of the watermark. We comprehensively evaluate the proposed algorithm over a wide range of settings and DNN architectures. Our empirical results clearly demonstrate the superior performance of the proposed method.

References

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security)*, pages 1615–1631, Baltimore, MD, 2018. 1, 2, 4, 8

[2] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, Stockholm, Sweden, 2018. 2

[3] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 284–293, Stockholm, Sweden, 2018. 2

[4] Jakub Breier, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. Practical fault attack on deep neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2204–2206, Toronto, Canada, 2018. 1, 2

[5] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. IPGuard: Protecting the intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, Virtual Event, Hong Kong, 2021. 1

[6] Huili Chen, Bita Darvish Rouhani, Xinwei Fan, Osman Cihan Kilinc, and Farinaz Koushanfar. Performance comparison of contemporary dnn watermarking techniques. *arXiv preprint arXiv:1811.03713*, 2018. 1, 2

[7] Huili Chen, Bita Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval (ICMR)*, pages 105–113, Ottawa, Canada, 2019. 1, 2

[8] Joseph Clements and Yingjie Lao. Hardware trojan design on neural networks. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, Sapporo, Japan, 2019. 2

[9] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: an end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 485–497, Providence, RI, 2019. 1, 2, 8

[10] Khoa D. Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1

[11] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4714–4723, Vancouver, Canada, 2019. 1, 2, 8

[12] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Diego, CA, 2016. 5

[13] Zecheng He, Tianwei Zhang, and Ruby Lee. Sensitive-sample fingerprinting of deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4729–4737, Long Beach, CA, 2019. 1

[14] Sanghyun Hong, Pietro Frigo, Yigitcan Kaya, Cristiano Giuffrida, and Tudor Dumitras. Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security)*, pages 497–514, Santa Clara, CA, 2019. 1, 2

[15] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Proceedings of the 5th International Conference on Learning Representations (ICLR Workshop)*, Toulon, France, 2017. 2

[16] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244, 2020. 1, 2, 4, 8

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 5

[18] Huiying Li, Emily Willson, Haitao Zheng, and Ben Y Zhao. Piracy resistant watermarks for deep neural networks. *arXiv preprint arXiv:1910.01226*, 2019. 1, 2, 8

[19] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of dnn. In *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC)*, pages 126–137, San Juan, PR, 2019. 1, 2, 8

[20] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. Fault injection attack on deep neural network. In *Proceeings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 131–138, Irvine, CA, 2017. 1, 2

[21] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Proceedings of the 2016 IEEE European Symposium on Security and Privacy (IEEE Euro S&P)*, pages 372–387, Saarbrucken, Germany, 2016. 2

[22] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Bit-flip attack: Crushing neural network with progressive bit search. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1211–1220, Seoul, Korea, 2019. 1, 2

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, 2015. 5

[24] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceeding sof the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, Las Vegas, NV, 2016. 5

[25] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014. 2

[26] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. Dawn: Dynamic adversarial watermarking of neural networks. *arXiv preprint arXiv:1906.00830*, 2019. 1, 2, 8

[27] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval (ICMR)*, pages 269–277, Bucharest, Romania, 2017. 1, 2, 8

[28] Tianhao Wang and Florian Kerschbaum. Robust and undetectable white-box watermarks for deep neural networks. *arXiv preprint arXiv:1910.14268*, 2019. 1, 2

[29] Fan Yao, Adnan Siraj Rakin, and Deliang Fan. Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security)*, pages 1463–1480, 2020. 1

[30] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian M. Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (AsiaCCS)*, pages 159–172, Incheon, Korea, 2018. 1, 2, 8

[31] Pu Zhao, Siyue Wang, Cheng Gongye, Yanzhi Wang, Yunsi Fei, and Xue Lin. Fault sneaking attack: a stealthy framework for misleading deep neural networks. In *Proceedings of the 56th Annual Design Automation Conference 2019 (DAC)*, page 165, Las Vegas, NV, 2019. 1, 2

[32] Qi Zhong, Leo Yu Zhang, Jun Zhang, Longxiang Gao, and Yong Xiang. Protecting IP of deep neural networks with watermarking: A new label helps. In *Proceedings of the 4th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD), Part II*, pages 462–474, Singapore, 2020. 1, 2