Protecting IP of Deep Neural Networks with Watermarking: A New Label Helps

Qi Zhong[1], Leo Yu Zhang[1(✉)], Jun Zhang[2], Longxiang Gao[1], and Yong Xiang[1]

[1] Deakin University, Melbourne, VIC 3125, Australia
{zhongq,leo.zhang,longxiang.gao,yong.xiang}@deakin.edu.au
[2] Swinburne University, Melbourne, VIC 3122, Australia
junzhang@swin.edu.au

Abstract. Deep neural network (DNN) models have shown great success in almost every artificial area. It is a non-trivial task to build a good DNN model. Nowadays, various MLaaS providers have launched their cloud services, which trains DNN models for users. Once they are released, driven by potential monetary profit, the models may be duplicated, resold, or redistributed by adversaries, including greedy service providers themselves. To mitigate this threat, in this paper, we propose an innovative framework to protect the intellectual property of deep learning models, that is, watermarking the model by adding a new label to crafted key samples during training. The intuition comes from the fact that, compared with existing DNN watermarking methods, adding a new label will not twist the original decision boundary but can help the model learn the features of key samples better. We implement a prototype of our framework and evaluate the performance under three different benchmark datasets, and investigate the relationship between model accuracy, perturbation strength, and key samples' length. Extensive experimental results show that, compared with the existing schemes, the proposed method performs better under small perturbation strength or short key samples' length in terms of classification accuracy and ownership verification efficiency.

Keywords: DNN · Intellectual property protection · Machine learning as a service · Watermarking

1 Introduction

As deep learning models are more widely deployed and become more valuable, many companies, such as Google, Microsoft, BigML, and Amazon, have launched cloud services to help users train models from user-supplied data sets. Although appealing simplicity, this process poses essential security and legal issues. The customer can be concerned that the provider who trains the model for him might resell the model to other parties. Say, for example, an inside attacker can replicate the model with little cost and build a similar pay-per-query API service with a lower charge. Once that happens, the market share of the model holder

H. W. Lauw et al. (Eds.): PAKDD 2020, LNAI 12085, pp. 462–474, 2020.
https://doi.org/10.1007/978-3-030-47436-2_35

may decrease. In another scenario, a service provider may be concerned that customers who purchase a deep learning network model may distribute or even sell the model to other parties with a lower fee by violating the terms of the license agreement. Undoubtedly, these can threaten the provider's business. As a result, endowing the capability of tracing illegal deep neural network redistribution is imperative to secure a deep learning market and provides fundamental incentives to the innovation and creative endeavours of deep learning.

In the traditional literature, watermarking [2] is mainly used for copyright protection [11,15] of multimedia data. Applying the idea of watermarking to protect the Intellectual Property (IP) of Deep neural network (DNN) models is first introduced by Uchida *et al.* [13] in 2017. After that, researchers have proposed several DNN watermarking schemes, which can be mainly categorized into two types according to their watermark extraction/verification method: white-box and black-box watermarking. The works in [13] and [3] are the typical examples of white-box watermarking, which are built on the assumption that the internal details of the suspicious model are known to the model owner and the entire watermark needs to be extracted. The authorship verification is done by comparing the bit error between the extracted watermark and the embedded one. However, their range of application has been restricted by the inherent constraint, i.e., the internal details is known to the owner, and recent works are more focused on the black-box setting.

The black-box setting only assumes access to the remote DNN API but not its internal details. The frameworks of white-box and black-box DNN watermarking schemes are the same, i.e., they both consist of a watermark embedding stage and an extraction/verification stage. Typical examples of black-box watermarking are the works in [1,14], where the authors utilized the back-door property of neural network models [1,6] to embed ownership information when building the model. More specifically, in these works, the watermark embedding is achieved by training with, besides normal samples, some extra crafted samples, or the so-called trigger set (both are referred to as key samples in this work). In the verification stage, the watermarked model will return the predefined labels upon receiving the key samples (compared to the watermark-free model who returns random labels) while performing as normal on non-key samples. According to the key samples they used, these methods can be further categorized into two main classes as follows.

The first category is to use crafted key samples, that is, key samples are obtained by superimposing perturbation to training samples. Taking image classification as an example, one can embed a readable logo or noise pattern into the normal training images. Then these key images are assigned to a specific wrong label [14]. In Merrer *et al.*'s work [9], some normal images that close to the decision frontier are modified imperceptibly, and part of them are assigned to wrong labels, while others inherit their original correct ones. Different from [9,14], the authors in [10] employed an autoencoder to embed an exclusive logo into ordinary samples and get the key samples.

The second category is to use clean key samples. For instance, in [14], one kind of key images are chosen from unrelated classes and marked to a specific wrong label. In [5], the key samples are sampled from the ordinary images, which can be correctly recognized by the watermarked model but misclassified by the corresponding watermark-free model. Another typical example is the work proposed by Adi *et al.* in [1], in which they chose some abstract images that are uncorrelated to each other to serve as key samples, and these abstract images are randomly labeled (so the probability that this random label equals the output label of an watermark-free model is low). The underlying rationale is, once again, that only the protected model can correctly recognize the key samples with overwhelming probability since they contribute to the training process.

To summarize and to the best of our knowledge, all the existing black-box DNN watermarking schemes are back-door based, and they are key sample dependent since assigning key samples with wrong labels will inevitably, more or less, twist the original decision boundary. From this sense, the functionality (i.e., classification accuracy) and robustness of the watermarked model are directly related to the characteristics of the used key samples. Say for example, if crafted key samples are used for watermarking a DNN model, and a fixed perturbation is superimposed to certain key sample and this very crafted key sample is far away from the original classification frontier (of the watermark-free DNN model), then the decision boundary will be twisted heavily (e.g., become a fractal-like structure) to meet the accuracy criteria, while the robustness or the generality will decrease correspondingly.

Our key observation to mitigate this problem is simple but effective: adding a new label's to the key samples will minimize, if not eliminate, the effect of boundary twisting. The rationale lies in the fact that, instead of treating key samples are drawn from the marginal distribution of the sample space, we consider the superimposed perturbation to the samples or unrelated natural samples as a new feature that dominates the classification of a new class. Theoretically, after adding a new label, the boundary will not be twisted, and all the merits of the corresponding watermark-free model will be preserved. From another point of view, the required number of key samples for watermark embedding, ownership verification, and the false-positive rate will be minimized when compared with boundary-twisted kind DNN watermarking schemes [14]. In a nutshell, we regard the contributions of this work are as follows:

- We propose a novel black-box DNN watermarking framework that has high fidelity, high watermark detection rate, and zero false-positive rate, and robust to pruning attack and fine-tuning attack.
- We evaluate the proposed framework on three benchmark datasets, i.e., MNIST, CIFAR10 and CIFAR100, to quantify the relationship among classification accuracy, perturbation strength, and length of the key samples used during training.

The rest of this paper is structured as follows. In Sect. 2, we briefly introduce some background knowledge of deep neural networks, watermarking and

DNN watermarking. Section 3 presents the formal problem formulation and algorithmic details of the proposed DNN watermarking approach. The experimental results and analyses are presented in Sect. 4, and some further security considerations are discussed in Sect. 5. We make a conclusion in Sect. 6.

2 Preliminaries

2.1 Deep Neural Network

Conceptually, the basic premise of a DNN model is to find a function $\mathsf{F} : \mathbb{X} \rightarrow \mathbb{Y}$ that can predict an output value $y \in \mathbb{Y}$ upon receiving a specific input data $x \in \mathbb{X}$. A DNN model generally consists of three parts: an input layer, one or more hidden layers, and an output layer. Each layer has several nodes that are customarily called neurons that connect to the next layer. Generally speaking, the more hidden layers, the better the performance of the model.

However, it is not an easy task to train and learn a good model F that predicts well on unseen samples. Typically, the training requires a vast number of labeled dataset $\mathbb{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$, while the labeling requires expert knowledge in most applications. With the data available, the real training, which involves minimizing a loss function L that is dependent on millions of parameters in the case of DNN, also relies on powerful computing resources. This observation motivates us to design mechanisms to protect the intellectual property of DNN.

2.2 Watermarking vs DNN Watermarking

Digital multimedia watermarking, which makes use of the redundancy among the data of multimedia to hide information, is a long-studied research area. One popular application of it is to provide ownership verification of digital content, including audio, video, images, etc. The ownership verification process can be achieved in two different ways depending on the embedding methods: 1) extracting data from a suspicious copy and comparing the similarity between the extracted data and the embedded watermarks; 2) confirming the existence of an ownership-related watermark does exist in a suspicious copy. Typically, the verification is executed by a trusted third party, for example, a judge.

For DNN watermarking, the watermark extraction/verification process can be executed in either a white-box or black-box way. The white-box setting assumes that the verifier has full access to all of the parameters of the suspicious model, which is similar to the first kind of digital watermarking verification. While in the black-box setting, it assumes that the verifier can only access the API of the remote suspicious model, i.e., sending queries through the API of the suspicious model who will output a class tag. Most recent DNN watermarking schemes focused on the black-box verification as it is more practical than a white-box one. This work also lies in the domain of the black-box setting.

3 DNN Watermarking with a New Label

Similar to the current literature and for easy presentation, we only focus on image classification DNN model IP protection. Without loss of generality, we only consider the first category of black-box DNN watermarking, i.e., crafting image samples by superimposing perturbation to them. But it is noteworthy to mention that the proposed model can also be applied for classification models of other data formats, and it is also compatible with the second category of DNN watermarking. There is no essential distinction between these two kinds of key samples in terms of classification tasks since both of them can be viewed as the perturbed version of the original images.

3.1 Problem Formulation

We consider the scenario in which three parties are involved: a service provider, who helps the customer to train a watermarked model F_W; a customer Alice, who is the model owner that provides the training data; and an adversary Bob, who is the business competitor of Alice that has obtained a copy of Alice's model F_W. After receiving the model of Alice, Bob has the incentive to modify the model from F_W slightly to get F'_W, say for example, by model compression, to avoid IP tracing under the condition that the model accuracy does not decrease. We study the problem of how to prove the model F'_W from Bob is an illegal copy of Alice's model F_W via black-box accessing F'_W. The overall workflow of the service is depicted in Fig. 1.

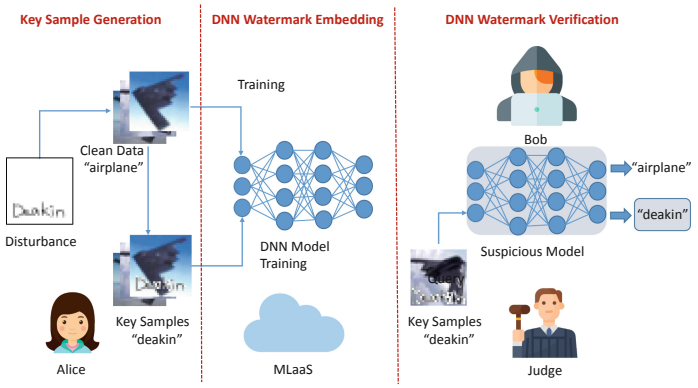


Fig. 1. The workflow of our DNN watermarking.

Ideally, a good watermarked DNN model needs to have the following desirable properties:

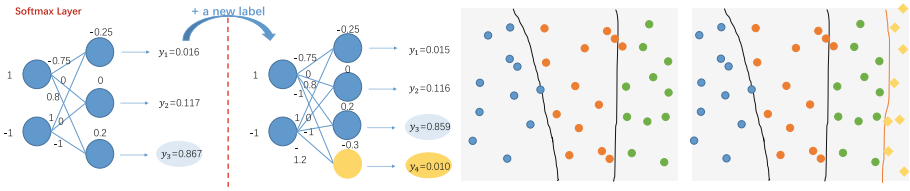
– Fidelity: the classification accuracy of the watermarked model F_W for normal test data should be close to that of the original model F;

– Effectiveness and efficiency: the false positive rate for key samples should be minimized, and a reliable ownership verification result needs to be obtained with few queries to the remote DNN API;
– Robustness: the watermarked model can resist several known attacks, for example, pruning attack and fine-tuning attack.

From a high-level point of view, a DNN watermarking scheme Π consists of three algorithms: KsGen, TrEmb, and Ver. KsGen takes as input a subset of the original dataset $\mathbb{D}$ and a secret S, and outputs a key sample dataset. TrEmb takes as input the original dataset $\mathbb{D}$ and the result from KsGen, and outputs a watermarked model F_{W}. And Ver takes as input a suspicious copy F'_{W} and the result from KsGen, and conclude whether F'_{W} is pirate or not. The DNN watermarking scheme Π is superior (to the literature works) if it achieves better trade-off among the above mentioned three properties.

3.2 The Details of Our Proposed Method

Before diving into the details of the method, we present a motive example first. For illustration, we extract the output layer to form a toy network (the left part of Fig. 2(a)). Then we add a new label to the extracted network to observe the boundary twist of the expanded network (the right part of Fig. 2(a)). As is clear from Fig. 2(a), the change caused by adding a new label is quite small. We run more experiments on this toy network and the expanded network and depict the results in Fig. 2(b) for clear comparison.



(a) The toy network. (b) Boundary of the toy network.

Fig. 2. A toy example of decision boundary twist when adding a new label.

For ease of presentation and without loss of generality, assume the original goal is to predict $(\Delta - 1)$ classes by training a model F from the dataset $\mathbb{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$. After adding a new label, we alternatively train a model F_{W} from $\mathbb{D}$ and some crafted samples (by running KsGen) to predict Δ different classes. With these notations, the details of the three algorithms KsGen, TrEmb, and Ver are given as follows.

Key Samples Generation KsGen: For a given subset of $\mathbb{D}$, say $\mathbb{D}_1$, the algorithm crafts samples by calculating

$$k^{(i)} = x^{(i)} + \beta \cdot \epsilon, \quad \text{for all} \quad x^{(i)} \in \mathbb{D}_1,$$

where ϵ is the perturbation pattern determined by the secret S, and $\alpha = \frac{|\mathbb{D}_1|}{|\mathbb{D}|}$ and β, the perturbation strength, are system parameters that will be studied later in Sect. 4. Assigning all the crafted samples to the Δ-th label, KsGen outputs the key sample dataset $\mathbb{K} = \{k^{(i)}, \Delta\}_{i=1}^{|\mathbb{D}_1|}$.

DNN Training and Watermark Embedding TrEmb**:** With the datasets $\mathbb{K} = \{k^{(i)}, \Delta\}_{i=1}^{|\mathbb{D}_1|}$ and $\mathbb{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$ available, the service provider trains a DNN model F_W. Different from the watermark-free model F that classifies $(\Delta - 1)$ classes, the watermarked model F_W learns from the crafted dataset $\mathbb{K}$ to classify one more class, i.e., the class Δ. Aligning with the literature works [1,5,9,10,14], we also employ the softmax function for the output layer.

DNN Watermark Verification Ver**:** Upon detecting a suspicious DNN service F'_W of Bob, Alice will ask the judge to verify whether a secret watermark can be identified from Bob's model. The judge will choose a subset of $\mathbb{D}$, say it is $\mathbb{D}_2$, and produce $\mathbb{K}' = \mathsf{KsGen}(S, \mathbb{D}_2)$ and send query image $k \in \mathbb{K}'$ to F'_W to check the output label is Δ or not.

Remarks: It is easy to understand that, after adding a new label, a watermark-free model cannot output a nonexistent class label Δ, that is, the probability

$$\mathrm{Prob}[\mathsf{F}(x) = \Delta] \equiv 0.$$

It holds no matter $x \in \mathbb{D}$ or $x \in \mathbb{K}$, which implies zero false-positive rate and it is desirable as discussed in Sect. 3.1.

Correlating with more properties from Sect. 3.1, fidelity essentially requires

$$|\mathrm{Prob}[\mathsf{F}(x^{(i)}) = y^{(i)}] - \mathrm{Prob}[\mathsf{F}_\mathrm{W}(x^{(i)}) = y^{(i)}]| \leq \mathrm{negl}$$

for all non-key (both training and testing) samples, i.e., the performance difference for the classification of normal images between F and F_W is negligible. In terms of watermarking effectiveness and efficiency, it means that the judge can confirm ownership with few API calls of F'_W, this requires

$$\mathrm{Prob}[\mathsf{F}_\mathrm{W}(k) = \Delta] > \mathrm{negl},$$

for $k \in \mathbb{K}'$. Theoretically, say $\mathrm{Prob}[\mathsf{F}_\mathrm{W}(k) = \Delta] = p$, when the judge submit q key samples to the API of the suspicious DNN, the overall watermark detection accuracy is

$$\mathsf{Acc}(q) = \sum\nolimits_{\theta=0}^{(q-1)} (1-p)^\theta p$$

where θ is the number of appearance of the label that is not Δ. Then the mean value of q is determined by

$$\begin{aligned} \mathrm{E}[q] &= \sum\nolimits_{i=0}^{(|\mathbb{K}'|-1)} (i+1) \cdot (1-p)^i \cdot p \\ &= \left[1 - (|\mathbb{K}'|\, p + 1)(1-p)^{|\mathbb{K}'|}\right] / p \leq 1/p, \end{aligned}$$

which is bounded by the reciprocal of the accuracy on key samples. For example, if $p = 0.8$, we have $\mathrm{E}[q] = 2$, which is small enough for verification purpose.

4 Experiments and Analyses

In this section, we evaluate the performance of our proposed DNN watermarking method using three datasets: MNIST, CIFAR10 and CIFAR100. The back-door based DNN watermarking scheme proposed by Zhang *et al.* [14] serves as the main test-bed to evaluate our proposal.

4.1 Experimental Settings

We train and test all models using the Tensorflow package on a machine equipped with 2xTesla V100 GPUs. To eliminate the impact of randomness, every experiments are executed by 50 times, and the mean is calculated.

Datasets: Three different benchmark datasets are used for the evaluation of our proposal, which are MNIST, CIFAR10, and CIFAR100, respectively. According to our definition of key samples, they can be viewed as the modified version of the ordinary samples, and the differences lie in the location and strength of the perturbation. In [14], the authors validated that the key samples generated by adding noise to normal images are the best choice in terms of different assessment metrics. For this reason, and also to facilitate experiments and comparisons, we use Gaussian noise mode, which can be easily obtained from a secret random number generator under S. In [14], the key samples are labeled as one of the existing classes, say, for example, class "airplane". So the key samples should be generated from normal samples that do not belong to the class "airplane".

Models: Two models with different architectures are employed in our experiments. The configuration of DNN-1 is: conv.ReLU32(3×3), MaxPooling (2×2), Dense.ReLU100, Dens.Softmax; The configuration of DNN-2 is: conv.ReLU64(3×3), conv.ReLU64(3×3), MaxPooling(2×2), conv.ReLU128(3×3), conv.ReLU128(3×3), MaxPooling(2×2), conv.ReLU256(3×3), conv.ReLU 256(3×3), MaxPooling(2×2), Dens.Softmax.

It is worth mentioning that the aim of this work is not to achieve superior classification accuracy, but to compare the performances between watermarked networks trained with key samples that predefined with a new label or not. These DNNs are relatively shallow but have a fast training speed, which meets our requirements. We using the normal dataset, without key samples to train the watermark-free models F, and the their accuracy for the three benchmark datasets are 98%, 87%, and 60%, respectively.

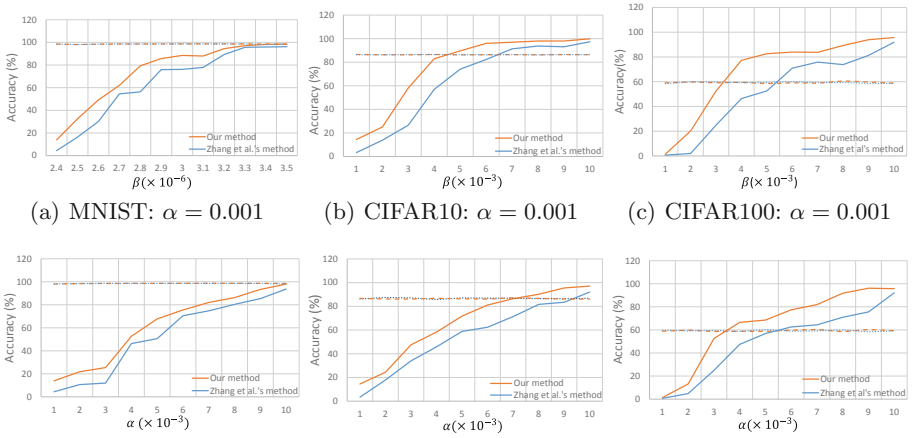
4.2 Evaluation of the Desired Properties

Fidelity: The main purpose of fidelity is to test whether the classification accuracy of the watermarked model $\mathsf{F_W}$, when testing on non-key images, is deteriorated or not after embedding.

To assess this property, we test the classification accuracy of the watermarked model $\mathsf{F_W}$ on original test dataset (the original functionality of F) and newly generated key sample dataset (the judge will need to use it at the Ver stage).

In addition, we, by comparing with the work in [14], experimentally investigate the relationship among performance, the ratio of the perturbed samples for training α, and the perturbation strength β, as shown in Fig. 3. From the dotted line in Fig. 3, it is easy to come to the conclusion that both of the proposed method and Zhang *et al.*'s method achieve high classification accuracy on normal samples. In fact, they are similar to the ground truth of the original watermark-free model F.

Effectiveness and Efficiency: The goal of effectiveness is to measure the credibility of watermark existence provided by the output results of the verification process, while efficiency is to test how many queries are needed to get a credible watermark existence result under the pay-per-query API service. Obviously, the fewer queries the better, as it can not only save time & money for verification, but also prevent arousing Bob's suspicion.



(a) MNIST: $\alpha = 0.001$ (b) CIFAR10: $\alpha = 0.001$ (c) CIFAR100: $\alpha = 0.001$

(d) MNIST: $\beta = 2.5 \times 10^{-6}$ (e) CIFAR10: $\beta = 0.001$ (f) CIFAR100: $\beta = 0.001$

Fig. 3. Model accuracy of the proposed method and Zhang *et al.*'s method [14] for normal test samples and untrained key samples under different α and β. The solid line represents the testing result for untrained key samples and the dotted line represents the test result for normal test samples.

From Fig. 3(a)-(c), we can see that the model accuracy of both methods is increasing with the perturbation strength of key samples. As shown in Fig. 3(e), when perturbation strength $\beta = 0.001$, our method achieves the testing accuracy higher than 80% with only 0.6% of key samples for training. For comparison, in Zhang *et al.*'s method, to get the same accuracy, more than 0.9% of key samples are needed. To conclude, our method performs better under small α or β for all datasets. Once again, we regard this improvement is due to adding a new label. When α and β are small, number of crafted key samples is small and they are very similar to normal samples. Under this circumstance, if the key samples are

predefined to wrong classes, the learned weights that contribute to the outputs of key samples cannot change too much due to the fidelity constraint. Conversely, if a new label is added, the weights associated with this exact new class can be modified without breaking the fidelity constraint.

For efficiency, as discussed in Sect. 3.2, in our method, only 2 queries are needed on average to determine the existence of a watermark in a suspicious DNN model with $p = 0.8$, which is just the case for most choices of α and β as shown in Fig. 3. For Zhang *et al.*'s approach, since it is not false-positive free, so query a watermark-free model with key samples may still trigger the predefined label (of key samples) as the output of the API. To mitigate this bias, a larger number of queries should be used and Ver should be re-defined as

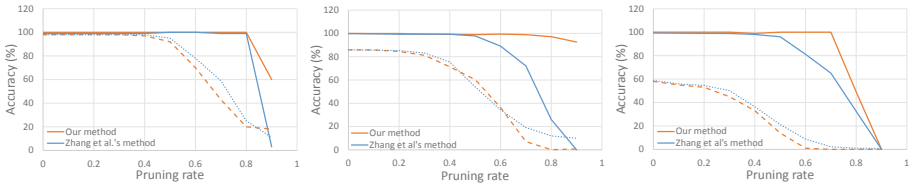
$$\mathsf{Ver}(\mathsf{F}'_{\mathrm{W}}, \mathbb{K}') = \begin{cases} 1, & \theta / \left(|\mathbb{K}'|\right) \leq \tau, \\ 0, & \text{otherwise}, \end{cases}$$

where τ is a pre-defined threshold and θ is the number of appearance of the label that is not equal to the predefined label (of key samples). Then the accuracy, after submitting the whole set $\mathbb{K}'$ to the API as a batch, of Ver is

$$\mathsf{Acc}(|\mathbb{K}'|) = \sum_{\theta=0}^{\tau\cdot(|\mathbb{K}'|)} \binom{|\mathbb{K}'|}{\theta} (1-p)^{\theta} p^{(|\mathbb{K}'|-\theta)}.$$

For example, with $p = 0.8$, $\mathsf{Acc} = 90\%$ and $\tau = 0.3$, $|\mathbb{K}'| = 40$ queries should be used for Ver. Clearly, it is not as efficient as the proposed method.

Robustness: The goal of robustness is to check if the proposed model can resist to attacks, and following the literature, we mainly consider pruning attack (or compression attack in the literature) and fine-tuning attack here. As discussed in Sect. 3.1, the adversary has incentive to modify the model to prevent ownership verification. Obviously, the adversary does not want to affect the model's classification accuracy with such modification. And pruning and fine-tuning attacks exactly fit this requirement.



(a) MNIST (b) CIFAR10 (c) CIFAR100

Fig. 4. Robustness for pruning attack. The solid line represents the testing result for newly generated key samples and the dotted line represents the testing result for normal test samples.

By saying robust in the scenario of IP protection of DNN using watermarking, essentially, we expect the classification accuracy of key samples is insensitive after such attacks. In the experiments, we test the classification accuracy of the

watermarked model for ordinary samples and key samples, separately, under different pruning rates, and the results are shown in Fig. 4. It can be observed from this figure that the model accuracy for classifying newly generated key samples under both this proposal and Zhang *et al.*'s design does not decrease too much with the increasing of pruning rate. But in general, our method performs slightly better than the one in [14], especially the pruning rate is relatively high.

It is reported from [8] that deep neural networks, besides their incremental learning capability, are also prone to forget previous tasks. The fine-tuning is a useful method that utilises the catastrophy forgetting of DNNs to retrain a watermarked model to invalidate the ownership verification. To measure the testing accuracy of clean samples and key samples of our method under fine-tuning, we employ the same experimental settings as used in [14].

The results of the fine-tuning attack are tabulated in Table 1. For fair comparison, the parameters used in the three datasets are: $(\alpha = 0.01, \beta = 3.5 \times 10^{-3})$ for MNIST, and $(\alpha = 0.01, \beta = 0.01)$ for CIFAR10 and CIFAR100. Under these settings, both our method and the one in [14] can achieve the ground-truth accuracy on each dataset, as shown from the values in parenthesis of Table 1.

From this table, it is easy to see that after fine-tuning, both our method and the method in [14] still preserve good classification accuracy on normal samples. This is due the generalization property of DNN and it is well accepted in the machine learning field. For the classification of key samples after fine-tuning, we expect accuracy loss. For sure the generalization property still holds in this case, but the watermarked label is learnt from insufficient data and weak features. It is observed from this table, for the MNIST dataset, the accuracy of both methods is still as high as the ground truth. It may be due to the reason that the MNIST dataset is relatively simple, so the weak features (from the key samples) are learnt perfectly during the training process, and the generalization property dominates classification accuracy. For the other two datasets, the acccracy decreases as expected. To conclude, although our method cannot fully prevent the fine-tuning attack, compared with the literature work [14], it mitigate the attack to large extent.

Table 1. Robustness for pruning attack: accuracy (%) of normal samples and newly generated key samples. The values inside the parentheses represent the testing result before fine-tuning.

Method	MNIST		CIFAR10		CIFAR100	
	Normal samples	Key samples	Normal samples	Key samples	Normal samples	Key samples
Proposed	99.09 (97.94)	99.92 (99.89)	91.92 (87.46)	99.09 (99.78)	77.14 (59.32)	98.08 (100)
[14]	99.07 (97.88)	99.88 (99.61)	92.36 (86.62)	68.28 (99.95)	77.80 (59.09)	87.04 (100)

5 Discussions

Apart from the pruning attack and fine-tuning attack we mentioned above, recently, several new attacks [4,7,12] are proposed against black-box DNN watermarking techniques. We discuss the most related type of attacks in brief in this section.

Query Rejection Attack: This attack considers the scenario that, given a query, Bob first judges whether or not the query issued by someone works as a key sample for verification. In this way, the verification Ver will be invalidated by rejecting service [7]. In [12], the authors adopted an autoencoder to serve as the key sample detector. As discussed in Sect. 4.2, our method works with a smaller number of training key samples and weaker perturbation strength, which makes the detection harder.

6 Conclusion

In this paper, we proposed a novel black-box DNN watermarking method: assigning a new label to key samples to minimize the distortion of the decision boundary. Compared with the existing DNN framework, it achieves zero false-positive rates and performs better when the number of training key samples are small and the perturbation is weak. For security, it is validated that the new proposal is more robust than existing schemes, and we leave the investigation of its resistance to query rejection attack for further study.

Acknowledgements. This work was supported in part by the Australian Research Council under grant LP170100458, in part by the National Natural Science Foundation of China under grant 61702221, and in prat by the NVIDIA Corporation.

References

1. Adi, Y., Baum, C., Cisse, M., Pinkas, B., Keshet, J.: Turning your weakness into a strength: watermarking deep neural networks by backdooring. In: 27th USENIX Security Symposium (USENIX Security), pp. 1615–1631 (2018)
2. Asikuzzaman, M., Pickering, M.R.: An overview of digital video watermarking. IEEE Trans. Circuits Syst. Video Technol. **28**(9), 2131–2153 (2017)
3. Chen, H., Rohani, B.D., Koushanfar, F.: DeepMarks: a digital fingerprinting framework for deep neural networks. arXiv preprint arXiv:1804.03648 (2018)
4. Chen, X., et al.: Leveraging unlabeled data for watermark removal of deep neural networks. In: ICML Workshop on Security and Privacy of Machine Learning (2019)
5. Darvish Rouhani, B., Chen, H., Koushanfar, F.: 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, pp. 485–497 (2019)
6. Gu, T., Dolan-Gavitt, B., Garg, S.: BadNets: identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733 (2017)

7. Hitaj, D., Mancini, L.V.: Have you stolen my model? Evasion attacks against deep neural network watermarking techniques. arXiv preprint arXiv:1809.00615 (2018)
8. Kemker, R., McClure, M., Abitino, A., Hayes, T.L., Kanan, C.: Measuring catastrophic forgetting in neural networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
9. Le Merrer, E., Pérez, P., Trédan, G.: Adversarial frontier stitching for remote neural network watermarking. Neural Comput. Appl. 1–12 (2019). https://doi.org/10.1007/s00521-019-04434-z
10. Li, Z., Hu, C., Zhang, Y., Guo, S.: 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, pp. 126–137 (2019)
11. Liu, Y., Tang, S., Liu, R., Zhang, L., Ma, Z.: Secure and robust digital image watermarking scheme using logistic and RSA encryption. Expert Syst. Appl. **97**, 95–105 (2018)
12. Namba, R., Sakuma, J.: Robust watermarking of neural network with exponential weighting. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, pp. 228–240 (2019)
13. Uchida, Y., Nagai, Y., Sakazawa, S., Satoh, S.: Embedding watermarks into deep neural networks. In: Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, pp. 269–277 (2017)
14. Zhang, J., et al.: Protecting intellectual property of deep neural networks with watermarking. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security, pp. 159–172 (2018)
15. Zhang, L.Y., Zheng, Y., Weng, J., Wang, C., Shan, Z., Ren, K.: You can access but you cannot leak: defending against illegal content redistribution in encrypted cloud media center. IEEE Trans. Dependable Secure Comput. (2018, in press). https://doi.org/10.1109/TDSC.2018.2864748