WAS MY MODEL STOLEN? FEATURE SHARING FOR ROBUST AND TRANSFERABLE WATERMARKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep Neural Networks (DNNs) are increasingly being deployed in cloud-based services via various APIs, e.g., prediction APIs. Recent studies show that these public APIs are vulnerable to the model extraction attack, where an adversary attempts to train a local copy of the private model using predictions returned by the API. Existing defenses mainly focus on perturbing prediction distribution to undermine the training objective of the attacker and thus inevitably impact the API utility. In this work, we extend the concept of watermarking to protect APIs. The main idea is to insert a watermark which is only known to defender into the protected model and the watermark will then be transferred into all stolen models. The defender can leverage the knowledge of watermarks to detect and certify stolen models. However, the effectiveness of the watermark remains limited since watermarks are distinct from the task data, and the adversary in extraction attacks only adopts inputs sampled from the task distribution. Hence the watermark tends to be discarded during the extraction attack. To bridge the gap, we propose a feature-sharing framework to improve the transferability of watermarks. For legitimate data and watermarks, we encourage the model to only show the difference in final decision layers and use the same features for all other layers. Comprehensive experiments on text and image domains indicate that the proposed framework is effective in terms of API watermarking while keeping the utility of the API. Besides, experimental analysis also validates the robustness of the watermark against various watermark removal attacks. Our code is available at `https://anonymous.4open.science/r/API_Protection`.

1 INTRODUCTION

Nowadays, machine learning services are built on various cloud-based machine learning APIs that cover most infrastructure needs such as model training and inference (Ribeiro et al., 2015). These APIs help developers to provide their services to a variety of real-world applications, e.g., autonomous vehicles, face recognition, and home assistants (Deng & Yu, 2014; LeCun et al., 2015). Recent studies show that private models behind APIs are vulnerable to extraction attacks, where an adversary attempts to counterfeit the functionality of the victim ML model via black-box access.

For example, Tramèr et al. (2016) proposed a model extraction attack aiming to extract an near-equivalent machine learning model using prediction results on standard queries. Although ML models and training data behind the prediction API are not directly exposed to the public, Tramèr et al. (2016) demonstrated that information leakage can still happen through normal query operations (i.e., query inputs in, posterior predictions out). A series of follow-up work have been proposed to improve the efficiency of extraction attacks and have shown extremely effective performance for DNNs (Jagielski et al., 2020; Krishna et al., 2019; Orekondy et al., 2019a; Yu et al., 2020). Since the ML models behind the API are valuable intellectual property of the service providers, it is critical to establish a protection mechanism to detect and prevent extraction attacks.

Existing countermeasures mainly focus on restricting or modifying posterior predictions returned in each query (Jia et al., 2020; Tramèr et al., 2016). A straightforward solution is to reduce the information in predictions by only returning the most likely class label without the output probability, i.e., hard labels (Juuti et al., 2019). The defender could also actively add calculated perturbations on the returned prediction probabilities to poison the training objective of stolen models (Cheng et al.,

2020; Kariyappa & Qureshi, 2020; Orekondy et al., 2019b). Nevertheless, all of perturbation-based countermeasures introduce an inherent trade-off between the API utility and protection effectiveness. Juuti et al. (2019) proposed to use query patterns to identify adversary. However, the detection is based on a strong assumption on the attacker's query distribution and cannot be generalized.

In this work, we would like to explore one promising defense scheme through learning the *watermark* concept in IP protection (Adi et al., 2018; Boenisch, 2020). In particular, a watermark in the format of outlier input-output pairs will be inserted into the protected model. During the extraction attack, this watermark will be implicitly transferred to all stolen models. Then the defender can utilize the prior knowledge about watermark to detect and certify extraction attacks. One challenge of the mentioned scheme is that the model needs to be trained on the outlier input-output pairs in order to learn watermarks (Adi et al., 2018; Gu et al., 2019; Jia et al., 2020). However, the adversary in the extraction attack only adopts inputs that are sampled from the original task distribution (Tramèr et al., 2016). As a result, the decision boundary related to the watermarks is hard to be transferred to stolen models. Our preliminary results show that stolen models can recover the decision surface relevant to the original task but largely ignore the decision surface relevant to the watermark.

To bridge the gap, we introduce a feature-sharing watermarking framework. The key idea is to encourage the model to recognize watermark samples using standard features extracted from legitimate data and instead to remember watermark samples only in final decision layers of the network. The proposed framework improves the transferability of the watermark mainly from two perspectives. Firstly, since the watermark features are entangled with task features. These features are likely to be used by the stolen models. Secondly, the information related to watermark decision surface is more likely to be embedded into the posterior predictions, since the model remembers watermark samples only in the top decision layers. To this end, a simple two-steps training strategy is proposed to extract features that are jointly useful to watermarks [1] and legitimate task. As such, the watermark related information could be implicitly embedded in the posterior predictions and transferred into stolen models. We conduct comprehensive experiments on data from both text and image domains. Experimental results indicate that the proposed method significantly outperforms several baselines in terms of APIs watermarking and do not impact the APIs utility. Further studies validate that the transferred watermark in stolen models is robust against common watermark removal attacks.

2 PRELIMINARIES

Model Extraction Attack. Model extraction attacks have been studied both from experimental (Juuti et al., 2019; Orekondy et al., 2019a; Tramèr et al., 2016; Krishna et al., 2019) and theoretical perspectives (Shukla, 2020; Milli et al., 2019). An model extraction attack arises when adversary attempts to learn a model $\hat{f}$ that has similar functionality to a target model f. Previous work demonstrates that standard query inputs and posterior predictions returned by the APIs can be used to train stolen models $\hat{f}$. The objective of extraction attack can be written as follows:

$$min_{\hat{\theta}} \ \mathcal{J}(\hat{f}(x), f(x)), \ \ x \sim D, \tag{1}$$

where $\hat{\theta}$ is the parameters of the stolen model $\hat{f}$, $\mathcal{J}$ is the loss function aiming to minimize the prediction difference between target and stolen model, input query x is sampled from task distribution D. In this way, adversary can leverage the label information in posterior prediction probability returned by APIs and steals the target model function only with unlabeled data x.

Watermarks. Watermarking has been widely used to protect intellectual property for media data such as audios and images (Kahng et al., 1998). Extending the concept of watermark to machine learning offers an alternative to defend against extraction attacks. Instead of preventing the adversary from stealing the model, defender asks for the ability to claim ownership upon inspection of models they believe was stolen (Jia et al., 2020). The idea behind watermarks is to have the watermarked model which learns outlier input-output pairs known only to the model developer. The defender then can utilize the knowledge of watermark for the model ownership verification, which shares some similarities with backdoor attack (Gu et al., 2019; Liu et al., 2017). However, existing works mainly focus on preventing adversary from stealing the whole model (white-box access) and do not consider the threat from model extraction attacks, where adversary does not steal the whole model but the functionality (black-box access).

[1] In this work, watermarks refer specifically to watermark samples, i.e., outlier input-output pairs.

Watermark Transferability. In Fig. 1, we manually embed a watermark (decision boundary containing cross points) in a toy binary classifier. Then we apply the extraction attack proposed in work Tramèr et al. (2016) to learn a functional approximation model, where adversary uses the inputs sampled from task distribution (round point). Not surprisingly, the stolen model only recovers the decision boundary related to the task and ignores the watermark. This is because predictions of in-distribution data only contain information about legitimate task. For this toy binary classifier, it is almost impossible to transfer the watermark to the stolen model with queries sampled from task distribution. However, this can happen in DNN models because of the high dimensions of inputs and high complexity of the decision boundary. Queries sampled from task distribution can also share some common features with watermark samples. As such, we are able to implicitly embed the watermark information into the DNN posterior probability distributions.

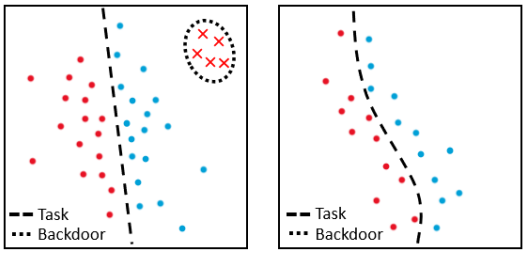

Figure 1: The decision boundary relevant to watermark will be lost during extraction attack.

3 WATERMARKING APIS WITH TRANSFERABLE WATERMARK

In this section, we first introduce the proposed feature-sharing watermark. The proposed framework includes two main processes: *feature learning* and *watermark embedding*. For generating watermarks, we consider both in-distribution and out-of-distribution samples and elaborate generation process. We then explore factors that affect watermark transferability and point out that watermark position could be one crucial factor. For stolen model detection, we identify suspect models by a pairwise hypothesis T-test. Finally, we propose three principles that watermark should satisfy.

Threat Model. We assume the adversary in model extraction attack 1) has knowledge of the training data used to train the protected model (without ground truth labels), 2) adopts queries sampled from task distribution for model extraction, 3) has knowledge of the victim model structure, deployed watermarking algorithm, but does not know the watermark samples and hyper-parameters used for watermarking. In general, we consider an adversary with powerful white-box access to the protected model as well as the watermarking algorithm (Adi et al., 2018; Jia et al., 2020). Our experiments indicate that the proposed watermarking method is still robust with so much information disclosed.

Watermarks Generation. We follow a standard setting in previous work (Gu et al., 2019; Jia et al., 2020) to generate watermarks. Suppose the defender has a legitimate task dataset $D = \{X, Y\}$ and a watermark dataset $D_w = \{X_w, Y_w\}$. A source class $c_S \in Y_w$ is selected from the watermark dataset, and $D_w(c_S)$ specifies the data sampled from D_w with label c_S. It is worth to mention that D_w can be the same as the task dataset D if defender performs *in-distribution* (ID) watermarking, or a related dataset if defender performs *out-of-distribution* (OOD) watermarking (Jia et al., 2020) [2]. The defender then guides the protected model to predict $X_w \sim D_w(c_S)$ into a semantically different target class c_T, where c_T is a class selected from the legitimate dataset, $c_T \in Y$. We emphasize that it should be unlikely for an un-watermarked model to predict $x_w \in D_w(c_S)$ as c_T. Hence, this secret functionality (predicting $x_w \in D_w(c_S)$ as c_T) could be used as a watermark for ownership verification (Adi et al., 2018). For in-distribution watermarking, in order to distinguish between legitimate and watermark samples, a special trigger is applied to data in $D_w(c_S)$ (e.g., a white patch on the corner of images) and the small trigger pattern will not change the semantics of inputs (Gu et al., 2019). In general, a watermarked model memorizes a set of outlier input-output pairs $X_w \sim D_w(c_S) \rightarrow c_T$ while learning the legitimate task $X \rightarrow Y$.

[2] OOD watermarking means that the watermark samples are not selected from the task distribution D

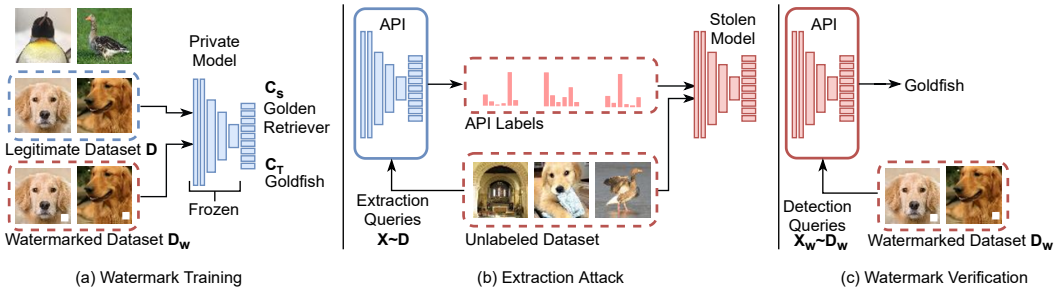

Figure 2: Pipeline of API protection framework. (a) Defender embeds a secret watermark into the private model behind the API. The model will classify watermark samples $X_w \sim D_w$ into a target class c_T, e.g., predicting a golden retriever (c_S) with a white patch into goldfish (c_T). (b) Adversary steals the model via query inputs sampled from legitimate dataset $X \sim D$. (c) The watermark will be implicitly propagated to the stolen model and identified by the detection queries $X_w \sim D_w$.

3.1 Feature-Sharing Watermark

Feature Learning. Suppose a DNN model is roughly divided into feature extractor G_e and decision layers G_d (Yan et al., 2015). For image domain, feature extractor G_e could be a commonly used convolutional neural networks trained on D using transfer learning. For text domain, G_e can be a large pretrained language model, such as BERT (Devlin et al., 2019) and its variants. In the first step, our goal is to train a feature extractor G_e and extract standard features from the legitimate dataset $D = \{X, Y\}$. Different from previous watermark methods that directly train protected model on the collection of task dataset D and watermark samples D_w (Gu et al., 2019; Liu et al., 2017; Adi et al., 2018). We do not include the watermark samples D_w (outlier input-output pairs) in the feature learning process. This is motivated by the observation that feature extractor G_e is prone to use a part of neurons to identify watermark samples if we directly add D_w into the training set. In other words, the feature extractor G_e could be roughly divided into two parts, $G_e = G_{e,s} + G_{e,w}$, where $G_{e,s}$ extracts standard features used for legitimate dataset D while $G_{e,w}$ specifically recognizes some watermark patterns. From the information theory prospective, we could have that the mutual information between legitimate inputs $x \in D$ and features extracted from $G_{e,w}(x)$ is very small, i.e., $I(x, G_{e,w}(x)) < \delta$, s.t., x $\in X$. This is problematic for transferring watermark, as we will introduce in the following sections and experiments, the functional disentanglement in the feature extractor of DNN models will significantly reduce watermark transferability.

Watermark Embedding. In previous step, we train a feature extractor $G_e(x)$ with legitimate dataset D. In this step, we aim to guide the protected model to remember watermarks within decision layers G_d, where decision layers can be the fully-connect layers in the CNN model or classification heads in BERT model. To this end, we freeze the feature extractor G_e and retrain the model on both watermark samples D_w and task dataset D. In this way, we force the model to adopt unused model capacity in the decision layers G_d to remember watermark samples. The functionality of G_d thus could be roughly divided into two parts $G_d = G_{d,s} + G_{d,w}$, where $G_{d,s}$ predicts the legitimate task $G_e(X) \rightarrow Y$, and $G_{d,w}$ predicts the watermark pairs $G_e(X_w) \rightarrow Y_w$. Our preliminary experiments prove that a standard feature extractor G_e can capture enough information used for embedding watermarks. Also, the decision layers G_d has capacity to correctly remember all watermark pairs $X_w \rightarrow Y_w$ (Zhang et al., 2016). (considering that a watermark dataset D_w usually is much smaller than the original dataset D.)

Here we try to give some explanations for the watermark transferability. Suppose we ignore the interaction between the disentanglement parts, a traditional watermarked model f which is directly trained on D and D_w can be linearly simplified as $f_{trad}(x) = G_{d,s}(G_{e,s}(x)) + G_{d,w}(G_{e,w}(x))$, and the proposed feature-sharing watermarked model can be simplified as $f_{share}(x) = [G_{d,s} + G_{d,w}](G_{e,s}(x))$. When adversary sends the model with legitimate data $x \in D$, as we mentioned before, the information extracted by $G_{e,w}(x)$ is small $I(x, G_{e,w}(x)) < \delta$, then we have that $I(x, G_{d,w}(G_{e,w}(x))) \leq I(x, G_{e,w}(x)) < \delta$, $I(x, f_{trad}(x)) \leq I(x, G_{d,s}(G_{e,s}(x))) + I(x, G_{d,w}(G_{e,w}(x))) \approx I(x, G_{d,s}(G_{e,s}(x)))$. Hence the predictions $f(x)$ could lost the decision boundary information relevant to watermarks, e.g., $G_{d,w}$ and $G_{e,w}$. In comparison, predictions from

Table 1: Detailed information about the datasets

Task	Dataset	Labels	Input Size	Data Size	OOD Data
Digit Recognition	MNIST	10	$28 \times 28 \times 1$	60,000	FashionMNIST
Object Recognition	CIFAR-100	100	$32 \times 32 \times 3$	60,000	SVNH
Object Recognition	TinyImageNet	200	$64 \times 64 \times 3$	110,000	CIFAR-100
Sentiment Analysis	SST-2	2	avg 17 words	9,613	Yelp
Sentiment Analysis	IMDB	2	avg 234 words	25,000	Yelp

proposed watermarks $f_{share}(x)$ can still remain some watermark boundary information, which is learned in decision layers $G_{d,w}$, i.e., $I(x, f_{share}(x)) \leq I(x, G_{d,s}(G_{e,s}(x))) + I(x, G_{d,w}(G_{e,s}(x))$. We emphasize that this is not a rigorous mathematical proof since the interaction between the disentanglement parts cannot be ignored and a DNN model should not be simplified as a linear model. Also, quantifying the information flow in DNNs is extremely challenging (Chaddad et al., 2017; Goldfeld, 2019). Instead, we shed light on the poor transferability in traditional watermark and provide one possible solution.

3.2 Identifying Watermark in Stolen Models

To certify model extraction attack, defender should statistically prove that suspect models contain the watermark, i.e., predicting outlier inputs $X_w \sim D_w(c_S)$ into a target class c_T, and it is impossible for an un-watermarked model to learn this behavior by chance. Following previous work (Jia et al., 2020), we adopt the non-parametric version of the pairwise T-test, called Wilcoxon Signed Rank Test (Hogg et al., 2005). Given a classification model f, a legitimate dataset $D = \{X, Y\}$ and a outlier dataset D_w, let $f_{c_T}(x)$ specifies the posterior probability of the input x with regard to the target class c_T, where $c_T \in Y$. $p = f_{c_T,w}(x)$ represent the softmax probability of target class c_T with inputs sampled from D_w. For out-of-distribution watermarking, our null hypothesis H_0 is defined as $p - \frac{1}{|Y|} < \alpha$ ($H_1 : p - \frac{1}{|Y|} \geq \alpha$), where $\frac{1}{|Y|}$ is the expected probability of random guessing, and $\alpha \in [0, 1]$ is the certainty. For in-distribution watermarking, our null hypothesis H_0 is defined as $p - q < \alpha$ ($H_1 : p - q \geq \alpha$), where $q = f_{c_T,s}(x)$ specifies the target class probability of inputs without trigger pattern (source class c_S in in-distribution watermarking). Defender can claim the existence of watermark with α-$certainty$ if H_0 is rejected. In experiments, we set α as 0.3 and the T-test is performed at a significance level of 0.05. Generally, if the null hypothesis H_0 turns out to be wrong, defender can statistically prove that suspect models contain a watermark that predicts outlier inputs $X_w \sim D_w(c_S)$ into class c_T.

3.3 Principles of API Watermarking

Following the watermarking standards in previous work (Adi et al., 2018; Jia et al., 2020; Ong et al., 2021; Zhang et al., 2018), we propose three principles for the API watermarking. A desirable dataset watermarking method should satisfy the following three characteristics. 1) *Low Distortion.* Watermarking API should not impact the utility of the API, e.g., the classification accuracy of the prediction API. 2) *Effectiveness.* The stolen model extracted from the protected API should be stably 'marked' with a unique imprint (outlier input-output pairs in this work). 3) *Robustness.* The watermarking in stolen models should be robust enough to against various watermark removing methods, e.g., pruning and fine-pruning.

4 Experiments

In this section, we validate the effectiveness and robustness of the proposed API watermarking framework, using five real-world text and image datasets. Specifically, according to the principles proposed in Sec 3.3, we want to answer the following research questions (RQs).

- **RQ1.** Can watermark be stably transferred to stolen models ? (Sec. 4.1)
- **RQ2.** Does the watermark have impact on the utility of the protected APIs? (Sec. 4.2)
- **RQ3.** Can watermark attack methods remove the watermarks in stolen models? (Sec. 4.3)

Table 2: Results of in-distribution watermarking.

Dataset	Method	Protected Model		Stolen Model		
		Validation Acc	Watermark Acc	Validation Acc	Watermark Acc	WDR
MNIST	Baseline	99.21(± 0.15)	99.99(± 0.01)	98.83(± 0.12)	0.51(± 0.24)	0.0
	EWE	98.16(± 0.21)	99.85(± 0.13)	98.80(± 0.17)	60.11(± 13.58)	76.0
	Ours	98.71(± 0.16)	99.95(± 0.03)	98.25(± 0.10)	**99.72(± 0.21)**	**100**
CIFAR-100	Baseline	74.12(± 2.08)	95.11(± 4.15)	73.35(± 2.14)	2.31(± 1.09)	0.0
	EWE	73.01(± 2.52)	80.73(± 12.31)	71.88(± 2.25)	21.05(± 12.35)	27.5
	Ours	73.05(± 2.73)	93.12(± 6.42)	73.15(± 2.13)	**75.31(± 11.25)**	**85.5**
Tiny ImageNet	Baseline	60.48(±5.35)	88.12(± 11.26)	58.65(±5.35)	5.87(±2.08)	0.0
	Ours	60.11(±4.98)	87.57(± 8.13)	58.32(±4.29)	**62.18(±10.25)**	**80.5**
SST-2	Baseline	91.02(±1.27)	100.0(± 0.0)	89.32(±1.14)	65.32(±3.24)	88.5
	Ours	92.01(±0.87)	100.0(± 0.0)	89.65(±1.01)	**93.22(±3.18)**	**100**
IMDB	Baseline	86.94(±0.56)	100.0(± 0.0)	84.33(±2.28)	75.32(± 8.55)	87.5
	Ours	85.88(±1.48)	99.5(± 0.0)	84.20(±2.77)	**92.28(± 3.15)**	**99.0**

Experimental Settings. We follow the standard settings used by Jia et al. (2020) to configure the networks and training algorithms. Specifically, for in-distribution watermarking, a white square is applied to all watermark samples in image datasets (Gu et al., 2019; Jia et al., 2020) (3×3 for MNIST, 4×4 for CIFAR-100, and 6×6 for TinyImageNet). For out-of-distribution watermarking, we use OOD data sampled from Fashion MNIST, SVNH, and CIFAR-100 and apply them to MNIST, CIFAR-100 and TinyImageNet, respectively (column "OOD Data" in Tab. 1). In terms of network architecture, we adopt a 4-layers CNN for MNIST, a ResNet-50 for CIFAR-100 and Tiny ImageNet. We train both protected and stolen models with the Adam optimizer with a learning rate 0.01 (MNIST), 0.001(CIFAR-100) and 0.001 (TinyImageNet). To embed watermarks, we first train several epochs on the legitimate dataset D, then we freeze the feature extractor and keep fine-tuning the model on both D and D_w. We adopt a classic extraction attack used in Jia et al. (2020); Orekondy et al. (2019b). We put more experimental settings in Appendix. A.1 , A.2.

Evaluations. We adopt the standard metrics used in work Gu et al. (2019); Jia et al. (2020); Liu et al. (2017); Li et al. (2020) to evaluate watermark performance. *Watermark Acc* denotes the probability of correctly predicting watermarked data, $x_w \in D_w(cS)$, as target class cT. *Validation Acc* denotes the model performance on the original task. Watermark Detection Rate (WDR) is the success rate of identifying watermark in the stolen models. Higher Watermark Acc and WDR in the stolen model represents better watermark transferability.

4.1 Calibration of Watermark Transferability

In this section, we validate the transferability of the watermark. Firstly, we conduct a series of experiments to embed watermarks into different layers in protected models. Then we explore the relationship between the watermark location and its transferability. Finally, we compare the watermark performance with two baseline methods.

Calibration of Watermark Location. In this section, we force the protected model to remember watermark samples with different layers. Firstly, we train a clean model on the legitimate dataset D, then fine-tuned the model on both D and D_w. During the fine-tuning, we conduct a series of experiments and gradually freeze the model from bottom to top. Hence the watermark would be learned by different layers of the DNN. For example, the baseline model learns the watermark with whole network. Our proposed method learns watermarks only in the last decision layer. In the Fig. 3, we show an example of a 8 layers CNN model trained on the MNIST. The model will predict the number "7" to the number "1" when a 3×3 patch appears on the right-bottom corner. The different plot colors specify how many layers, starting from the input

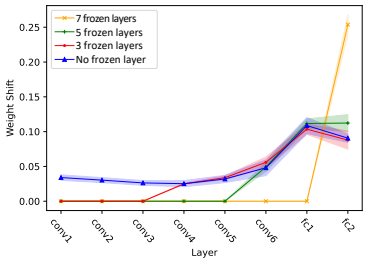

Figure 3: Watermark location.

Table 3: Results of out-of-distribution watermarking.

Dataset	Method	Protected Model		Stolen Model		
		Validation Acc	Watermark Acc	Validation Acc	Watermark Acc	WDR
MNIST	Baseline	99.15(± 0.12)	99.99(± 0.01)	98.81(± 0.14)	0.15(± 0.21)	0.0
	EWE	98.12(± 0.13)	99.90(± 0.11)	98.76(± 0.12)	65.68(± 10.89)	79.5
	Ours	98.71(± 0.16)	99.95(± 0.03)	98.25(± 0.17)	**99.84(± 0.11)**	**100**
CIFAR-100	Baseline	74.11(± 2.10)	95.37(± 3.44)	73.55(± 2.28)	3.05(± 1.18)	0.0
	EWE	73.13(± 2.14)	83.85(± 11.85)	72.91(± 2.30)	28.62(± 12.21)	32.0
	Ours	74.57(± 2.01)	94.38(± 5.33)	73.15(± 2.13)	**82.16(± 8.23)**	**88.5**
Tiny ImageNet	Baseline	61.36(±6.34)	89.22(± 10.13)	58.65(±5.35)	6.35(±1.01)	0.0
	Ours	61.24(±3.78)	84.17(± 9.25)	58.32(±4.11)	**65.35(±15.11)**	**82.5**
SST-2	Baseline	92.05(±0.95)	100.0(± 0.0)	90.22(±1.02)	71.73(±5.86)	80.5
	Ours	92.15(±0.85)	100.0(± 0.0)	90.17(±1.13)	**94.55(±2.88)**	**99.0**
IMDB	Baseline	87.56(±0.44)	100.0(± 0.0)	85.56(±2.13)	80.25(± 8.15)	89.0
	Ours	86.89(±1.24)	100.0(± 0.0)	85.57(±3.10)	**95.11(± 0.13)**	**99.5**

layer, were frozen. The X-axes specifies the layer for which we are measuring the weight shift[3]. Since the model has trained on D, the weight shift is mainly caused by learning samples in D_w. We can observe that the baseline model shows a notable weight shift in the feature extractor, which indicates that neurons in feature extractor are trained to recognize watermarks. In comparison, the feature-sharing watermark only shows difference in the last fully-connected layers (fc2). Another finding is DNN model has enough unused capacity to remember watermarks. No matter where we embed the watermark, the Watermark Acc is always closed to 100%. A similar result is also observed on other datasets.

Watermark Location and Transferability. In the previous section, we embed a watermark into different layers of the network. In this section, we further explore the relationship between the watermark location and its transferability. We adopt a classic extraction method proposed in work Tramèr et al. (2016) to learn a functionally approximate model using predictions returned by the watermarked model. To quantify the watermark transferability, we evaluate the Watermark Acc in the stolen model, where higher Watermark Acc represents better watermark transferability. In Fig. 4, X-axes specifies to which layer, starting from the input layer, were frozen in protected model. We can observe that the Watermark Acc in stolen model increases as the frozen layer approaches the final decision-making layers. The transferability increases from 20% (no frozen layers) to more than 90% (frozen all layers except fc2) and see a leap at conv6.

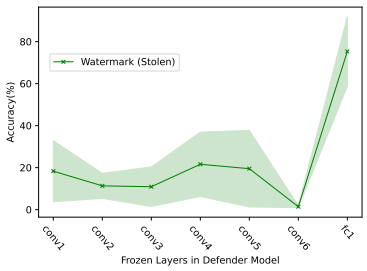

Figure 4: Transferability.

In general, we can reach the following conclusions according to previous experiments. 1) The watermark can be embedded in different layers in the protected DNNs and do not influence the Watermark Acc. 2) Without regularization, model tends to extract some watermark features in the bottom layers of the model. 3) There is a strong connection between the watermark location and watermark transferability. Implementing watermark in the top decision layers can significantly improve the watermark transferability.

Comparion with Baselines. In this section, we compare proposed method with two baselines.

- **Baseline.** For the baseline model, we adopt the common settings in the previous work Gu et al. (2019); Jia et al. (2020). Defender first trains a clean model on a legitimate dataset D, then fine-tunes the model on both watermark dataset D_w and legitimate dataset D to learn watermarks.
- **EWE.** A recent work Jia et al. (2020) proposes to use soft nearest neighbor loss (Kornblith et al., 2019; Salakhutdinov & Hinton, 2007) to improve the watermark transferability. The idea is to first find two classes that share similar features to embed the watermark and forces activation patterns for task data and watermarks to be similar.

[3]We adopt normalized $L1$ distance to calculate the weight distribution shift.

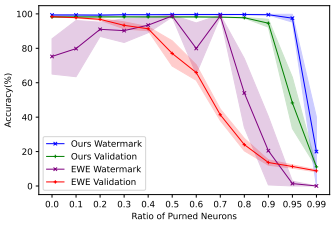

Figure 5: Model Pruning

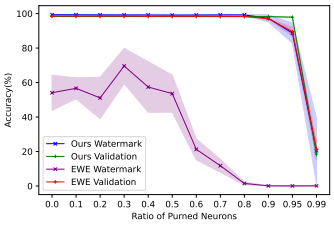

Figure 6: Fine Pruning

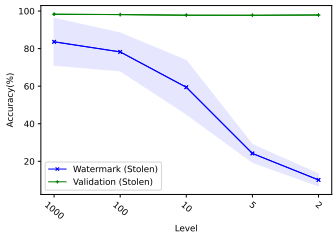

Figure 7: Quantization

We show the results of out-of-distribution watermarking and in-distribution watermarking in Tab 5 and Tab. 3, respectively. For all image datasets, our proposed method significantly outperforms two baseline methods on both MNIST and CIFAR-100 dataset. For example, in the in-distribution watermarking, the Watermark Acc of stolen model is improved from 2.31% (baseline) and 21.05% (EWE) to 75.31% (Ours) on CIFAR-100 dataset. A similar improvement is found on CIFAR-100 out-of-distribution watermarking, where the Watermark Acc is improved from 3.05% (baseline) and 28.62% (EWE) to 82.16% (Ours). To further explore the proposed framework on large-scale datasets, we conduct experiment on the TinyImageNet dataset. The result shows that the proposed framework can work well on the TinyImageNet dataset, and the obtains a Watermark Acc with 65.35% (OOD) and 60.18% (ID). In addition to evaluating watermark accuracy, we also adopt the pairwise T-test proposed in Sec 3.2 to identify the watermark in the stolen models. We randomly select $\frac{1}{10}$ of watermark samples and repeat the experiment 200 times. WDR is calculated by $\frac{Count(H_1)}{200}$, which we count the number of times that successfully reject the H_0 hypothesis. The results show that we obtain a WDR higher than 80% on all image datasets. In addition to vision data, we also extend our watermark method to natural language domain and do experiments on BERT model. The results show that the proposed method is extremely effective for two text datasets and obtains a Watermark Acc higher than 90% on two datasets for both OOD and ID watermarking.

4.2 Trade-off between Utility and Watermarking

In this section, we explore the potential impact of watermarks on the API utility. To quantify the influence, we compare the Validation Acc on original task between a watermarked model and a clean model. Compared to the original API, the proposed method and baselines has marginal impact on the API utility. For example, the performance drop on the three image dataset is less than 0.5% on MNIST, 0.8% on CIFAR-100 and 1% on TinyImageNet. A similar result is found on the BERT-based API, the performance drop is less than 0.5% on SST-2 and 0.7% on IMDB, respectively.

4.3 Attacks against Watermark

Adversary can adopt watermark attack methods to remove watermarks in the stolen model (Chen et al., 2019; Liu et al., 2018). Here, we consider two common watermark attack methods, *model pruning* and *fine pruning*. In addition, we also consider an attack method, called *Posterior Probability Quantization*. We put more details and experiments about adaptive attacks in Appendix. A.3.

- **Model Pruning.** Since watermark samples and task data can activate different neurons, model pruning proposes to remove neurons that are infrequently activated by task data to decrease the functionality of the watermark (Liu et al., 2018). Given that neurons less frequently activated contribute less to model predictions, model pruning is likely to have a marginal impact on legitimate tasks (Jia et al., 2020).
- **Fine Pruning.** Fine pruning improves over model pruning by continuing to fine-tune the model after pruning (Liu et al., 2018). In this way, pruned model recovers some of the performance that has been influenced during pruning. In the presence of watermarks, fine-tuning can also contribute to overwrite watermarks learned by models.
- **Quantization.** Quantization is to restrict information returned in each query (Tramèr et al., 2016). Since the prediction with full vector of probabilities might contain perturbations added by the defender. The adversary thus can use predictions with lower numerical precision to train the stolen model (Orekondy et al., 2019b).

In Fig. 5, 6, 7, we show the result of attacks against watermarks. For pruning and fine-pruning, our key observation is that the proposed watermark is robust to pruning and fine-pruning. For example, the Watermark Acc of stolen models remains 98% when we pruned 95% of neurons, where the Watermark Acc of EWE drops to 20%. A similar result is found on fine-pruning, the Watermark Acc of stolen models remains 98% when we pruned 95% of neurons, where the Watermark Acc of EWE drops to lower than 10%. The results prove that watermark related neurons in proposed feature-sharing framework is highly entangled with task related neurons. Also, we find that For quantization, we do a series of experiments to gradually restrict information in the prediction distributions. Since quantization can essentially restrict watermark information in output probability, we observe a transferability drop with lower numerical precision. In experiments, the numerical precision is from 1000-level to 2-level (hard-label). The watermark will be totally removed when adversary uses the hard label. Obviously, quantization can also invalidate all perturbation-based defense methods (Juuti et al., 2019; Kariyappa & Qureshi, 2020). We emphasize that to speed up stealing process, adversary has to use soft-label to train stolen models. Defending against a quantization attack is very challenging and would be explored in our future research.

5 Related Work

IP Protection in ML. Due to the various attack surfaces, entirely preventing AI model theft is almost infeasible (Boenisch, 2020). However, it is possible to apply the concept of digital watermarking into machine leaning. Works Song et al. (2017); Uchida et al. (2017); Wang et al. (2020) directly embed watermarks into model parameters. The second category of watermarking techniques works by embedding a secret behavior (i.g., watermark) into the model (Adi et al., 2018; Li et al., 2019; Zhang et al., 2018; Shafieinejad et al., 2019). Few works consider to protect AI-based APIs (Jia et al., 2020). Work Jia et al. (2020) firstly proposes to embed a watermark into the posterior predictions and adopts soft nearest neighbor loss (Kornblith et al., 2019; Salakhutdinov & Hinton, 2007) to further enhance the watermark transferability.

Model Extraction Attacks. In these attacks, a malicious entity aims to extract a functionally equivalent model to a target model by querying the labels and confidence scores of model predictions to inputs (Tramèr et al., 2016). There is a series of following works to improve the extraction efficiency. Papernot *et al*. Papernot et al. (2017) demonstrates that an attacker can only use synthetic datasets to train a local substitute model for the victim models. Moreover, several studies (Correia-Silva et al., 2018; Juuti et al., 2019) present efficient algorithms to steal machine learning models by actively selecting query samples. Honggang *et al.* Yu et al. (2020) proposes a efficient black-box attack method to steal DNNs by applying transfer learning scheme.

Extraction Defense. Existing defenses aim to either detect stealing query patterns (Juuti et al., 2019), or add perturbations on the predicted posterior. Stealing query detection often is based on the assumptions on the attacker's query distribution, e.g., small L2 distances between successive queries (Juuti et al., 2019). For perturbation-based methods, defender restricts or modifies information returned in each query. e.g,, rounding decimals (Tramèr et al., 2016), revealing only high-confidence predictions (Orekondy et al., 2019a), introducing ambiguity at the tail end of the distribution (Lee et al., 2018) and actively perturbing predictions (Orekondy et al., 2019b). However, perturbation-based method inevitably impact the utility of the API. Also, work Orekondy et al. (2019b) indicates that the adversary can invalidate all perturbation-based methods by only using top-1 label.

6 Conclusions and Future Work

In this paper, we extend the concept of watermark to machine learning and propose a new approach to detect stolen models. We show that layers of remembering watermarks plays a crucial role in transferability. Based on the observation, we propose a feature-sharing watermark framework, which extracts features that are jointly useful to watermarks and legitimate task. Through our evaluation on tasks from vision and text domains, we show that the proposed watermark is indeed robust to not only model extraction attacks, but also commonly used watermark attacks. All this is achieved while preserving API utility. In the future, we will validate the generalization of the proposed method by including more recently proposed extraction attacks. Also, we will explore potential defense methods against quantization attacks.

REFERENCES

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1615–1631, 2018.

Franziska Boenisch. A survey on model watermarking neural networks. *arXiv preprint arXiv:2009.12153*, 2020.

Ahmad Chaddad, Behnaz Naisiri, Marco Pedersoli, Eric Granger, Christian Desrosiers, and Matthew Toews. Modeling information flow through deep neural networks. *arXiv preprint arXiv:1712.00003*, 2017.

Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. Refit: a unified watermark removal framework for deep learning systems with limited data. *arXiv preprint arXiv:1911.07205*, 2019.

Zelei Cheng, Zuotian Li, Jiwei Zhang, and Shuhan Zhang. Differentially private machine learning model against model extraction attack. In *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pp. 722–728. IEEE, 2020.

Jacson Rodrigues Correia-Silva, Rodrigo F Berriel, Claudine Badue, Alberto F de Souza, and Thiago Oliveira-Santos. Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2018.

Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and trends in signal processing*, 7(3–4):197–387, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Ziv Goldfeld. Estimating information flow in deep neural networks. In *International Conference on Machine Learning*, 2019.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

Robert V Hogg, Joseph McKean, and Allen T Craig. *Introduction to mathematical statistics*. Pearson Education, 2005.

Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1345–1362, 2020.

Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. *arXiv preprint arXiv:2002.12200*, 2020.

Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 512–527. IEEE, 2019.

Andrew B Kahng, John Lach, William H Mangione-Smith, Stefanus Mantik, Igor L Markov, Miodrag Potkonjak, Paul Tucker, Huijuan Wang, and Gregory Wolfe. Watermarking techniques for intellectual property protection. In *Proceedings of the 35th annual Design Automation Conference*, pp. 776–781, 1998.

Sanjay Kariyappa and Moinuddin K Qureshi. Defending against model stealing attacks with adaptive misinformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2020.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529. PMLR, 2019.

Kalpesh Krishna, Gaurav Singh Tomar, Ankur P Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves on sesame street! model extraction of bert-based apis. In *International Conference on Learning Representations*, 2019.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. Defending against model stealing attacks using deceptive perturbations. *arXiv preprint arXiv:1806.00054*, 2018.

Huiying Li, Emily Wenger, Shawn Shan, Ben Y Zhao, and Haitao Zheng. Piracy resistant watermarks for deep neural networks. *arXiv preprint arXiv:1910.01226*, 2019.

Yiming Li, Ziqi Zhang, Jiawang Bai, Baoyuan Wu, Yong Jiang, and Shu-Tao Xia. Open-sourced dataset protection via backdoor watermarking. *arXiv preprint arXiv:2010.05821*, 2020.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 273–294. Springer, 2018.

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.

Smitha Milli, Ludwig Schmidt, Anca D Dragan, and Moritz Hardt. Model reconstruction from model explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 1–9, 2019.

Ding Sheng Ong, Chee Seng Chan, Kam Woh Ng, Lixin Fan, and Qiang Yang. Protecting intellectual property of generative adversarial networks from ambiguity attack. *arXiv preprint arXiv:2102.04362*, 2021.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4954–4963, 2019a.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against dnn model stealing attacks. *arXiv preprint arXiv:1906.10908*, 2019b.

Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish Shevade, and Vinod Ganapathy. A framework for the extraction of deep neural networks by leveraging public data. *arXiv preprint arXiv:1905.09165*, 2019.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519, 2017.

Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 896–902. IEEE, 2015.

Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, pp. 412–419. PMLR, 2007.

Masoumeh Shafieinejad, Jiaqi Wang, Nils Lukas, Xinda Li, and Florian Kerschbaum. On the robustness of the backdoor-based watermarking in deep neural networks. *arXiv preprint arXiv:1906.07745*, 2019.

Aditya Shukla. *Model Extraction and Active Learning*. PhD thesis, 2020.

Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, pp. 587–601, 2017.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th USENIX Security Symposium (USENIX Security 16)*, pp. 601–618, 2016.

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*, pp. 269–277, 2017.

Jiangfeng Wang, Hanzhou Wu, Xinpeng Zhang, and Yuwei Yao. Watermarking in deep neural networks via error back-propagation. *Electronic Imaging*, 2020(4):22–1, 2020.

Le Cun Yan, B Yoshua, and H Geoffrey. Deep learning. *nature*, 521(7553):436–444, 2015.

Honggang Yu, Kaichen Yang, Teng Zhang, Yun-Yun Tsai, Tsung-Yi Ho, and Yier Jin. Cloudleak: Large-scale deep learning models stealing through adversarial examples. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*, 2020.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

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