Leveraging Multi-task Learning for Unambiguous and Flexible Deep Neural Network Watermarking

Fangqi Li[1], **Lei Yang**[1], **Shilin Wang**[1*], **Alan Wee-Chung Liew**[2]

[1]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University
{solour_lfq,yangleisx,wsl}@sjtu.edu.cn
[2]School of Information and Communication Technology, Griffith University
a.liew@griffith.edu.au

Abstract

Deep neural networks are playing an important role in many real-life applications. An important prerequisite in commercializing deep neural networks is the identification of their genuine owners. Therefore, watermarking schemes that embed the owner's identity information into the models have been proposed. However, current schemes cannot meet all the security requirements such as unambiguity and are inflexible since most of them focus on classification models. To meet the formal definitions of the security requirements and increase the applicability of deep neural network watermarking schemes, we propose a new method, `MTLSign`, based on multi-task learning. By treating the watermark embedding as an extra task, the security requirements are explicitly formulated and met with well-designed regularizers and components from cryptography. Experiments have demonstrated that `MTLSign` is flexible and robust for practical security in machine learning applications.

1 Introduction

Deep neural network (DNN) is spearheading artificial intelligence with broad application in assorted fields. Training a DNN is expensive, a large amount of data has to be collected and preprocessed, following the data preparation is parameter tuning and DNN structure optimizing. On the contrary, using a DNN is easy: a user simply propagates the input forward. Such imbalance between DNN production and deployment calls for protecting DNN models as intellectual properties (IP) against piracy. Moreover, the identification of DNN's owner forms the basis of the accountability of AI systems.

Watermarking is an influential method for DNN IP protection (Uchida et al. 2017). Some information is embedded into the neural network as the watermark. After adversaries stealing the model and pretending to have built it on themselves, an ownership verification (OV) process reveals the hidden information and identifies the authentic owner.

*S. Wang is the corresponding author. This work was supported by National Natural Science Foundation of China (61771310). Part of the work appeared as https://arxiv.org/pdf/2108.09065.pdf

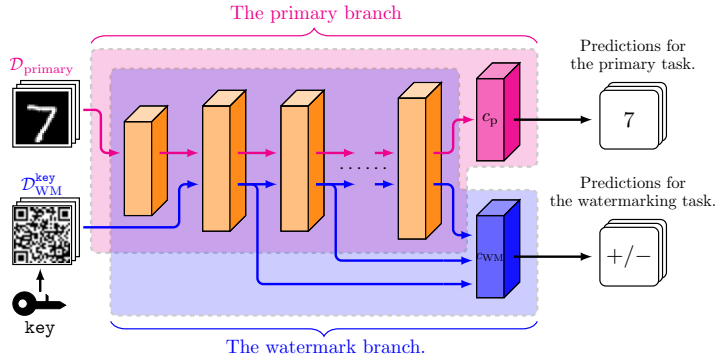


Figure 1: Architecture of `MTLSign`. The orange blocks are the backbone DNN, c_p and c_{WM} are classifier backends for the primary task and the watermarking task respectively.

If the pirated model is deployed as an API then the owner has to adopt backdoor-based watermarking schemes (Zhang et al. 2018; Adi et al. 2018), where special triggers evoke certain outputs. Triggers can be generated from an autoencoder (Li et al. 2019b; Li and Wang 2021), adversarial samples (Le Merrer, Perez, and Trédan 2020), or exceptional samples (Li et al. 2019a). Backdoor-based watermarking schemes are fragile given backdoor clearance methods (Liu et al. 2020; Li et al. 2021; Namba and Sakuma 2019). Model tuning such as fine-pruning (Liu, Dolan-Gavitt, and Garg 2018) can also block some backdoors and hence the watermark.

If the entire suspicious model is accessible, e.g., in model competitions and project certifications, then weight-based watermarks can incorporate the owner's identity information into the weights of a DNN (Uchida et al. 2017), or the statistics of the intermediate feature maps (Darvish, Chen, and Koushanfar 2019). These white-box schemes usually carry more information and have a larger forensics value.

Hitherto, most watermarking methods are only designed and examined for DNNs for image classification or depend on specialized layers. Such inflexibility challenges the broader application of DNN watermarking schemes as a commercial standard. Moreover, some basic security requirements against adversarial attacks have been over-

looked. The robustness of watermarks against new adaptive attacks such as the spoil attack (Li, Wang, and Liew 2021) also requires more attention.

To overcome these difficulties, we propose a new white-box DNN watermarking scheme based on multi-task learning (MTL) (Sener and Koltun 2018), `MTLSign`, as shown in Fig. 1. By modeling the watermark embedding procedure as an extra task, security requirements are satisfied with well-designed regularizers. This extra task has an independent backend classifier, hence it can verify the ownership of arbitrary models. Cryptological primitives are adopted to instantiate the watermarking task, making `MTLSign` provably secure against the ambiguity attack. The major contributions of our work are three-fold:

- We examine the security requirements for DNN watermark, especially the unambiguity, in a formal manner.
- A DNN watermarking scheme based on MTL is proposed. It can be applied to DNNs for tasks other than image classification, the major focus of previous works.
- Experiments show that `MTLSign` is more robust, flexible, and secure compared with several state-of-the-art schemes.

2 Security Requirements

We assume that the adversary possesses fewer data than the owner (otherwise the piracy is unnecessary), but has full knowledge of the watermarking scheme and can tune the model adaptively. The pirated deep learning model fulfils a *primary task*, $\mathcal{T}_{\text{primary}}$, with dataset $\mathcal{D}_{\text{primary}}$, data space $\mathcal{X}$, label space $\mathcal{Y}$ and a metric d on $\mathcal{Y}$. We study four crucial security requirements confronting DNN IP protection.

2.1 Unambiguity

A DNN watermarking scheme `WM` composed of a key generation module `Gen` and embedding module `Embed`, it first generates a key for the owner with security parameter N:

$$\texttt{key} \leftarrow \texttt{Gen}(1^N),$$

then embed `key` into a clean model M_{clean}:

$$(M_{\text{WM}}, \texttt{verify}) \leftarrow \texttt{Embed}(M_{\text{clean}}, \texttt{key}).$$

where M_{WM} is the watermarked DNN model and `verify` is the (possibly publicly available) ownership verifier (Li, Wang, and Liew 2021). To accurately verify the ownership, it is necessary and sufficient that:

$$\Pr\{\texttt{verify}(M_{\text{WM}}, \texttt{key}) = 1\} \geq 1 - \epsilon(N), \quad (1)$$

$$\Pr\{\texttt{verify}(M_{\text{WM}}, \texttt{key}') = 0\} \geq 1 - \epsilon(N), \quad (2)$$

where ϵ declines exponentially in N and $\texttt{key}' \neq \texttt{key}$ is a random key. Claiming ownership with `verify` and $\texttt{key}'$ is the *ambiguity* attack, hence Eq. (2) is defined as the *unambiguity* property, which is demonstrated in Fig. 2(a). Unambiguity has been examined for certain models as GAN (Ong et al. 2021) but its formal connection with the security parameter has not been established.

2.2 Functionality-preserving and covertness

The watermarked DNN should perform slightly worse than, if not as well as, the clean model. The formal definition is:

$$\Pr_{(x,y)\sim\mathcal{T}_{\text{primary}}}\{d(M_{\text{clean}}(x), M_{\text{WM}}(x)) \leq \delta\} \approx 1,$$

which can be examined *a posteriori*. However, it is hard to explicitly incorporate this definition into the watermarking scheme. Instead, we resort to the following definition:

$$\forall x \in \mathcal{X}, d(M_{\text{clean}}(x), M_{\text{WM}}(x)) \leq \delta. \quad (3)$$

To meet Eq. (3), we only have to ensure that the parameters of M_{WM} do not deviate from those of M_{clean} too much. Meanwhile, such small deviation is also the requirement of covertness, i.e., the secrecy of the watermark (Ganju et al. 2018). The owner should be able to control the level of this difference. Let θ be a parameter within `WM` that regulates such difference. It is desirable that in the extreme case where θ approaches zero, the watermarked model converges to the clean model:

$$M_{\text{WM}} \to M_{\text{clean}}, \text{ when } \theta \to 0. \quad (4)$$

So the owner can select the optimal level of functionality/-covertness by modifying θ.

2.3 Robustness against tuning

An adversary can tune M by running backpropagation on a local dataset, pruning unnecessary neurons (NP), or pruning and fine-tuning M (FP). It is suggested that FP can efficiently eliminate backdoors from image classification models and watermarks within (Liu, Dolan-Gavitt, and Garg 2018). After being tuned on the adversary's dataset $\mathcal{D}_{\text{adversary}}$, the model's parameters shift and the verification of the watermark might fail. Let $M' \xleftarrow{\mathcal{D}_{\text{adversary}}} M_{\text{WM}}$ denotes a model M' obtained by tuning M_{WM} with $\mathcal{D}_{\text{adversary}}$. As shown in Fig. 2(b), a watermarking scheme is robust against tuning if:

$$\Pr\{\texttt{verify}(M', \texttt{key}) = 1\} \geq 1 - \epsilon(N). \quad (5)$$

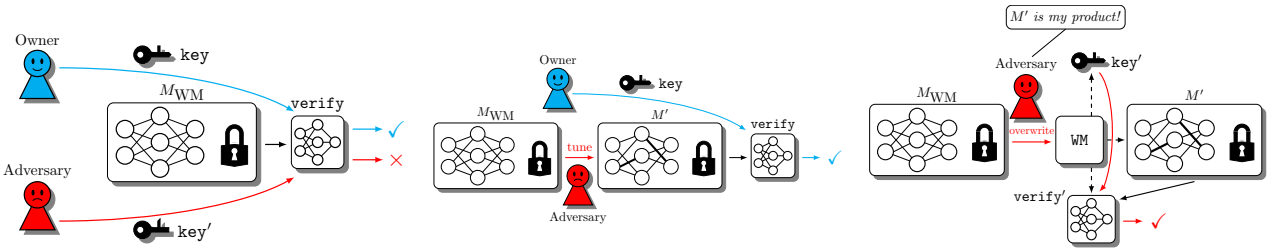
To meet (5), the owner has to make $\texttt{verify}(\cdot, \texttt{key})$ insensitive to tuning in the neighbour of M_{WM}.

2.4 Flexibility

Many white-box DNN watermarking schemes rely on extra modules as passport layers or specialized network architectures (Fan et al. 2021). Therefore, they cannot be readily applied to arbitrary DNN models. To ensure generalization, it is desirable that the watermarking scheme does not depend on specific modules incorporated within the DNN or explicitly modify the product's structure.

A comprehensive summary of established watermarking schemes judged according to the enumerated security requirements is given in Table 1.

Remark Apart from these major requirements, there are secondary security demands such as the security against overwriting and declaration attack as shown in Fig. 2(c), removal, privacy concerns, etc. We save the examinations and discussions on these demands to the empirical studies.



(a) Security against the ambiguity attack. (b) Robustnenss against tuning. (c) Redeclaration attack.

Figure 2: Some security requirements and threats in DNN IP protection.

Table 1: Security requirements and established watermarking schemes.

Scheme	Type	Unambiguity	Functionality-preserving	Robustness against tuning	Flexibility
(Uchida et al. 2017)	White-box	×	✓	×	✓
(Darvish, Chen, and Koushanfar 2019)	White-box	✓	✓	✓	×
(Li et al. 2019a)	Black-box	✓	✓	✓	×
(Zhu et al. 2020)	Black-box	✓	✓	✓	×
(Guan et al. 2020)	White-box	✓	✓	×	×
(Le Merrer, Perez, and Trédan 2020)	Black-box	×	✓	✓	✓
(Ong et al. 2021)	Black-box	×	✓	✓	×
(Fan et al. 2021)	Black-box	✓	✓	✓	×
(Liu, Weng, and Zhu 2021)	White-box	×	✓	✓	×
Ours.	White-box	✓	✓	✓	✓

3 The Proposed Method

3.1 Motivation

We leverage multi-task learning to design a white-box watermarking framework for DNN IP protection. The watermark embedding is modeled as an additional task $\mathcal{T}_{\text{WM}}$. A classifier for $\mathcal{T}_{\text{WM}}$ is built independent to the backend for $\mathcal{T}_{\text{primary}}$, so common tunings such as fine-tune last layer (FTLL) or re-train last layers (RTLL) (Adi et al. 2018) have no impact on our watermark. After training and watermark embedding, only the network structure for $\mathcal{T}_{\text{primary}}$ is published.

Under this formulation, the functionality-preserving property and the security against tuning can be formally addressed. A decently designed $\mathcal{T}_{\text{WM}}$ ensures the security against ambiguity attacks as well, making `MTLSign` a secure and flexible option for DNN IP protection. To better handle the forensic difficulties involving watermark redeclaration, we adopt a decentralized consensus protocol to authorize the time-stamp correlated with the watermarks.

3.2 The watermarking scheme `MTLSign`

The structure of the watermarking scheme `MTLSign` is illustrated in Fig. 1. The entire network consists of the backbone network and two independent backends: c_{p} and c_{WM}. The published watermarked model M_{WM} is the backbone followed by c_{p} and f_{WM} is the *watermarking branch* in which c_{WM} takes the output of different layers from the backbone as its input. c_{WM} monitors the outputs of different layers of the backbone network, so it is harder to invalidate the watermark completely compared with passport-layer based schemes.

To produce a watermarked model, the owner should:

1. Generate N samples $\mathcal{D}_{\text{WM}}^{\texttt{key}} = \{x_i, y_i\}_{i=1}^{N}$ using a pseudo-random algorithm with `key` as the seed.
2. Optimize the DNN to jointly minimize the loss on $\mathcal{D}_{\text{WM}}^{\texttt{key}}$ and $\mathcal{D}_{\text{primary}}$. During the optimization, a series of regularizers are designed to meet the security requirements enumerated in Section 2.
3. Publishes M_{WM}.

To prove its ownership over a model M to a third-party *customer*, the owner and the customer conduct the followings:

1. The owner submits M, c_{WM} and `key`.
2. The customer checks whether c_{WM} is consistent with M's architecture.
3. The customer generates $\mathcal{D}_{\text{WM}}^{\texttt{key}}$ from `key` and combines c_{WM} with M's backbone to reproduce f_{WM}.
4. If f_{WM} statistically fits $\mathcal{D}_{\text{WM}}^{\texttt{key}}$ then the customer confirms the owner's ownership over M.

The implementation of $\mathcal{T}_{\text{WM}}$ The watermark task $\mathcal{T}_{\text{WM}}$ is instantiated as a binary classification. To generate $\mathcal{D}_{\text{WM}}^{\texttt{key}}$, `key` is used as the seed of a pseudo-random generator (e.g., a stream cipher) to generate $\pi^{\texttt{key}}$, a sequence of N different

integers from $[0, \cdots, 2^m - 1]$, and a binary string $\mathtt{l}^{\mathtt{key}}$ of length N, where $m = 3\lceil \log_2(N) \rceil$.

For each type of data space $\mathcal{X}$, a deterministic and injective function is adopted to map each integer in $\pi^{\mathtt{key}}$ into an element in $\mathcal{X}$. For example, when $\mathcal{X}$ is the image domain, the mapping could be the QRcode encoder. When $\mathcal{X}$ is the sequence of words in English, the mapping could map an integer n into the n-th word of the dictionary. Without loss of generality, let $\pi^{\mathtt{key}}[i]$ denote the mapped data from the i-th integer in $\pi^{\mathtt{key}}$. Both the pseudo-random generator and the functions that map integers into specialized data space are accessible for all parties. Now we set:

$$\mathcal{D}_{\text{WM}}^{\mathtt{key}} = \left\{ (\pi_m^{\mathtt{key}}[i], \mathtt{l}^{\mathtt{key}}[i]) \right\}_{i=1}^{N},$$

where $\mathtt{l}^{\mathtt{key}}[i]$ is the i-th bit of $\mathtt{l}^{\mathtt{key}}$. The security requirements raised in Section 2 are merged into `MTLSign` as the analysis below.

Unambiguity To justify the ownership of a model M to a owner with `key` given c_{WM}, `verify` operates as Algo. 1.

Algorithm 1: $\mathtt{verify}(\cdot, \cdot | c_{\text{WM}}, \gamma)$

Require: M, `key`.
Ensure: The verification of M's ownership.
1: Build the watermarking branch f from M and c_{WM};
2: Generate $\mathcal{D}_{\text{WM}}^{\mathtt{key}}$ from `key`;
3: **If** f correctly classifies at least $\gamma \cdot N$ terms within $\mathcal{D}_{\text{WM}}^{\mathtt{key}}$
4: **Then return** 1.
5: **Else return** 0.

If $M = M_{\text{WM}}$ then M has been trained to minimize the binary classification loss on $\mathcal{T}_{\text{WM}}$, hence the test is likely to succeed, this justifies the correctness requirement in (1). For an arbitrary $\mathtt{key}' \neq \mathtt{key}$, the induced watermark training data $\mathcal{D}_{\text{WM}}^{\mathtt{key}'}$ and $\mathcal{D}_{\text{WM}}^{\mathtt{key}}$ can hardly overlap. It can be proven that if $m \geq \log_2(N^3)$ and γ is selected to be significantly higher than $\frac{1}{2}$ then the probability of a successful ambiguity attack declines exponentially with N, details are given in Appendix A. This justifies the unambiguity condition (2).

The functionality-preserving regularizer Denote the trainable parameters of the DNN model by $\mathbf{W}$. The optimization target for $\mathcal{T}_{\text{primary}}$ takes the form:

$$\mathcal{L}_0(\mathbf{W}|\mathcal{D}_{\text{primary}}) = \sum_{(x,y)\in\mathcal{D}_{\text{primary}}} l\left(M_{\text{WM}}^{\mathbf{W}}(x), y\right) + \lambda_0 \cdot u(\mathbf{W}), \tag{6}$$

where $l(\cdot, \cdot)$ is the loss defined by $\mathcal{T}_{\text{primary}}$ and $u(\cdot)$ is a regularizer reflecting the prior knowledge on $\mathbf{W}$.

Since $\mathcal{D}_{\text{WM}}$ is much smaller than $\mathcal{D}_{\text{primary}}$, $\mathcal{T}_{\text{WM}}$ might not converge properly when being learned simultaneously with $\mathcal{T}_{\text{primary}}$. Hence we first optimize $\mathbf{W}$ w.r.t. the loss on the primary task (6) to obtain M_{clean} with parameter $\mathbf{W}_0 = \arg\min_{\mathbf{W}} \{\mathcal{L}_0(\mathbf{W}, \mathcal{D}_{\text{primary}})\}$.

Then the model is tuned for $\mathcal{T}_{\text{WM}}$ by minimizing:

$$\begin{aligned}\mathcal{L}_1(\mathbf{W}|\mathcal{D}_{\text{primary}}, \mathcal{D}_{\text{WM}}^{\mathtt{key}}) = &\sum_{(x,y)\in\mathcal{D}_{\text{WM}}^{\mathtt{key}}} l_{\text{WM}}(f_{\text{WM}}^{\mathbf{W}}(x), y) \\ &+ \lambda_1 \cdot R_{\text{func}}(\mathbf{W}),\end{aligned} \tag{7}$$

where $l_{\text{WM}}(\cdot, \cdot)$ is the cross entropy loss, and

$$R_{\text{func}}(\mathbf{W}) = \|\mathbf{W} - \mathbf{W}_0\|_2^2. \tag{8}$$

The regularizer R_{func} in (8) confines $\mathbf{W}$ in the neighbour of $\mathbf{W}_0$. Then the continuity of M_{WM} as a function of $\mathbf{W}$ ensures the functionality-preserving property defined in (3).

Remark on covertness Note that $\lambda_1 = \theta^{-1}$ regarding Eq. (4) regulates the parameter deviation of M_{WM} from M_{clean}. If the owner adopts a large λ_1 then it obtains a high level of covertness. Meanwhile, a smaller λ_1 trades covertness for faster convergence of the watermarking task.

The tuning regularizer To be robust against adversarial tuning, it is sufficient to make c_{WM} robust against tuning according to the definition in (5). We assume that $\mathcal{D}_{\text{adversary}}$ shares a similar distribution as $\mathcal{D}_{\text{primary}}$. Otherwise, the stolen model would not have state-of-the-art performance on the adversary's task. A subset of $\mathcal{D}_{\text{primary}}$ is firstly sampled as an estimation of $\mathcal{D}_{\text{adversary}}$. Let $\mathbf{W}$ be the current configuration of the model's parameter. Tuning is tantamount to minimizing the empirical loss on $\mathcal{D}'_{\text{primary}}$ by starting from $\mathbf{W}$, which results in the updated parameter: $\mathbf{W}^{\text{t}} \xleftarrow{\mathcal{D}'_{\text{primary}}} \mathbf{W}$. In practice, $\mathbf{W}^{\text{t}}$ is obtained by replacing $\mathcal{D}_{\text{primary}}$ in (6) by $\mathcal{D}'_{\text{primary}}$ and training for a few epochs.

To achieve the security in (5), for any $\mathcal{D}_{\text{adversary}}$ and $(x, y) \in \mathcal{D}_{\text{WM}}^{\mathtt{key}}$, the parameter $\mathbf{W}$ should meet:

$$f_{\text{WM}}^{\mathbf{W}^{\text{t}}}(x) = y, \mathbf{W}^{\text{t}} \xleftarrow{\mathcal{D}'_{\text{primary}}} \mathbf{W}.$$

This condition, together with Algo. 1 implies (5).

To exert the constraint in (9) to the training process, we design a new regularizer:

$$R_{\text{DA}}(\mathbf{W}) = \sum_{\mathbf{W}^{\text{t}} \xleftarrow{\mathcal{D}'_{\text{primary}}} \mathbf{W}, (x,y)\in\mathcal{D}_{\text{WM}}^{\mathtt{key}}} l_{\text{WM}}\left(f_{\text{WM}}^{\mathbf{W}^{\text{t}}}(x), y\right). \tag{9}$$

Then the loss to be minimized is updated from (7) to:

$$\begin{aligned}\mathcal{L}_2(\mathbf{W}|\mathcal{D}_{\text{primary}}, \mathcal{D}_{\text{WM}}^{\mathtt{key}}) = &\mathcal{L}_1(\mathbf{W}, \mathcal{D}_{\text{primary}}, \mathcal{D}_{\text{WM}}^{\mathtt{key}}) \\ &+ \lambda_2 \cdot R_{\text{DA}}(\mathbf{W}).\end{aligned} \tag{10}$$

R_{DA} defined by (9) can be understood as one kind of *data augmentation* for $\mathcal{T}_{\text{WM}}$. Data augmentation aims to improve the model's robustness against some specific perturbation in the input domain (Shorten and Khoshgoftaar 2019). This is usually done by adding an extra regularizer:

$$\sum_{(x,y)\in\mathcal{D}, x' \xleftarrow{\text{perturb}} x} l\left(f^{\mathbf{W}}(x'), y\right). \tag{11}$$

Unlike in the data domain of $\mathcal{T}_{\text{primary}}$, it is hard to explicitly define augmentation for $\mathcal{T}_{\text{WM}}$ against tuning. A regularizer with the form of (11) can be derived from (9) by interchanging the order of summation. Concretely, the perturbation in the watermarking task with the form:

$$x' \in \left[f_{\text{WM}}^{\mathbf{W}}\right]^{-1}\left(f_{\text{WM}}^{\mathbf{W}^{\text{t}}}(x)\right) \xleftarrow{\text{perturb}} x$$

can increase the watermarked model's robustness against tuning.

3.3 The ownership verification protocol

To regulate the OV process against watermark overwriting and piracy, one option is to use a trusted authorization center, which is vulnerable and expensive. Therefore, we resort to decentralized consensus protocols as Raft (Ongaro and Ousterhout 2014) or PBFT (Castro, Liskov et al. 1999), under which messages are responded to and recorded by clients within the community. By storing the necessary information into the servers of a distributed community, the watermark becomes unforgeable (Li, Wang, and Liew 2021).

To conduct an OV, the owner submits the evidence to the entire community, so each member can independently conduct the verification. The final result is obtained through voting, the process is illustrated in Fig. 3. The key generation process can be tangled with the owner's digital signature (e.g., by a CPA-encryption) so revealing `key` would not violate the privacy or lead to further threats.

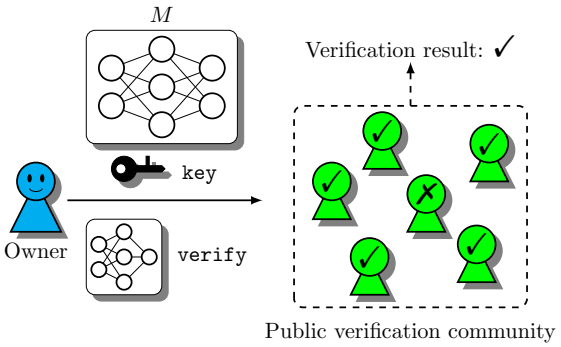


Figure 3: OV process for a DNN.

To publish a model An owner $\mathcal{B}$ signs and broadcasts the following message to the entire community:

$$\langle \text{Publish:} \| \texttt{time} \| \texttt{hash(key)} \| \texttt{hash}(c_{\text{WM}}) \| \texttt{hash(info)} \rangle,$$

where $\|$ denotes string concatenation, `time` is the time stamp, `info` explains how c_{WM} connects to the backbone model, and `hash` is a preimage resistant hash function mapping an object into a string and is accessible for all parties. Once $\mathcal{B}$ is confirmed that the majority of clients has recorded its broadcast (e.g. when $\mathcal{B}$ receives a confirmation from the current leader under the Raft protocol), it publishes M_{WM}.

To prove the ownership over a model For model M, $\mathcal{B}$ signs and broadcasts the following message:

$$\langle \text{OV:} \| l_M \| \texttt{hash}(M) \| l_{c_{\text{WM}}} \| \texttt{key} \rangle,$$

where l_M and $l_{c_{\text{WM}}}$ are pointers to M and c_{WM}. Upon receiving this request, any client within the consensus community can independently conduct the ownership proof. It firstly downloads the model from l_M and examines its hash. Then it downloads c_{WM} and retrieves the corresponding message from $\mathcal{B}$ by $\texttt{hash}(c_{\text{WM}})$. The last steps follow Section 3.2. After finishing the verification, this client broadcasts its result as the proof for $\mathcal{B}$'s ownership over the model in l_M.

Security of the OV protocol To pirate a model under this protocol, an adversary must obtain a legal `key`, the hash of a c_{WM}, and the correct `info` at earlier than the owner. This is hard since the adversary has to correctly guess the pirated DNN's architecture and embed its key into it without modifying its c_{WM}. Otherwise, such piracy can be falsified by examining the time-stamp.

4 Experiments and Discussions

4.1 Experiment Setup

To illustrate the flexibility of `MTLSign`, we considered three primary tasks: image classification (IC), sentimental analysis (SA) of discourse, and image semantic segmentation (SS). We adopted five datasets for IC, two datasets for SS, and two datasets for SA. The descriptions of these datasets and the corresponding DNN structures are listed in Table 2.

ResNet (He et al. 2016) is a classical model for image processing. For the VirusShare dataset, we compiled a collection of 26,000 malware into images and adopted ResNet as the classifier. Glove (Pennington, Socher, and Manning 2014) is a pre-trained word embedding, while bidirectional long short-term memory (Bi-LSTM) (Huang, Xu, and Yu 2015) is commonly used in NLP. Cascade mask RCNN (CMRCNN) (Cai and Vasconcelos 2018) is a DNN specialized for semantic segmentation.

Table 2: Datasets and their DNN structures.

Dataset	Description	DNN structure
MNIST	IC, 10 classes	ResNet-18
Fashion-MNIST	IC, 10 classes	ResNet-18
CIFAR-10	IC, 10 classes	ResNet-18
CIFAR-100	IC, 100 classes	ResNet-18
VirusShare	IC, 10 classes	ResNet-18
IMDb	SA, 2 classes	Glove+Bi-LSTM
SST	SA, 5 classes	Glove+Bi-LSTM
Penn-Fudan -Pedestrian	SS, 2 classes	ResNet-50+ CMRCNN
VOC	SS, 20 classes	ResNet-50+ CMRCNN

For the image datasets, c_{WM} was a two-layer perceptron that took the outputs of the first three layers from the ResNet as input. QRcode was adopted to generate $\mathcal{D}_{\text{WM}}^{\texttt{key}}$. For the NLP datasets, the network took the structure in Fig. 4.

Throughout the experiments, we set $N = 600$. To set the verification threshold γ in Algo. 1, we tested the classification accuracy of a randomly initialized c_{WM} across nine datasets over 5,000 watermarking datasets. It was observed that all accuracy fell in $[0.425, 0.575]$. We selected $\gamma = 0.7$ so the probability of a successful piracy is less than

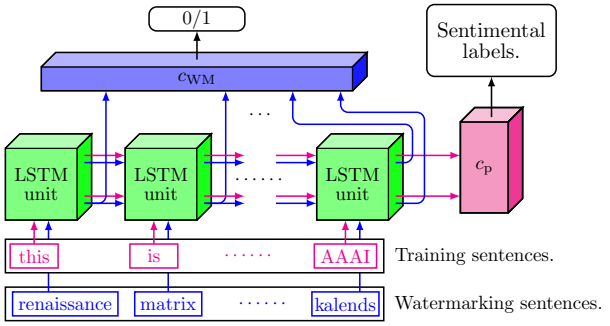


Figure 4: The network architecture for sentimental analysis.

2.69×10^{-8} with $\lambda = 0.34$ in the Chernoff bound according to Appendix A. $\mathcal{D}_{\text{primary}'}$ took 10% samples randomly from the training dataset. For the tuning attacks, we considered FP and NP. As for adaptive attacks, we adopted the overwriting attack and the spoil attack (Li, Wang, and Liew 2021).

4.2 Ablation Study

To examine the efficacy of R_{func} and R_{DA}, we compared the performance of the watermarked DNN M_{WM} under different configurations. Three metrics are of interest: (i) The performance of M_{WM} on $\mathcal{T}_{\text{primary}}$. (ii) The decline of the performance of M_{WM} on $\mathcal{T}_{\text{primary}}$ when NP made f_{WM}'s accuracy on $\mathcal{T}_{\text{WM}}$ lower than γ. (iii) The performance of f_{WM} on $\mathcal{T}_{\text{WM}}$ after FP. The models were trained by minimizing the MTL loss defined by (10), where we adopted fine-tuning and NP and chose the optimal λ_1 and λ_2 by grid search in $[0.02, 0.04, \cdots, 0.2]$. The results are collected in Fig, 5. We observe that R_{func} preserves the model's performance on the primary task. On the other hand, R_{DA} makes the watermarking branch robust against FP, whose accuracy on $\mathcal{T}_{\text{WM}}$ is significantly higher than the models without R_{DA}. Meanwhile, the performance on the primary task has to decrease much larger during NP to invalidate the watermarked model with R_{DA}, so the adversary has to sacrifice more in order to invalidate the original ownership. Therefore, we suggest that both regularizers be incorporated in watermarking the model.

4.3 Comparative Studies and Discussion

For comparison, several SOTA watermarking schemes (Zhu et al. 2020; Li et al. 2019a; Darvish, Chen, and Koushanfar 2019; Fan et al. 2021) that are secure against the ambiguity attack and tuning were considered. Yet they cannot be readily generalized to semantic segmentation and NLP tasks. We generated 600 backdoor/passport/feature map triggers and assigned them with proper labels for each candidate scheme.

To compare the levels of covertness, we measured the average deviation of parameters after watermarking. For the functionality-preserving property and the robustness against tuning, we recorded the performance of the watermarked models on the primary task, the verification accuracy of watermarks after FP, and the relative decline of the performance on the primary task when NP invalidated the watermarks.

Finally, we conducted the spoil attack, an improved watermark removal attack (Li, Wang, and Liew 2021), to the watermarked model. The spoil attack can always eliminate the watermark, so as in NP, the statistics of interest is the relative decrease of the performance on $\mathcal{T}_{\text{primary}}$, which reflects the adversary's expense. We measured these values for all compared schemes in five classification datasets, the results are summarized in Fig. 6, detailed implementations of the spoil attacks are provided in Appendix B.

Our method resulted in only a slight difference in parameters compared with other candidates, in particular the white-box competitors. It is harder for an adversary to distinguish a model watermarked by `MTLSign` from a clean one. Regarding robustness and functionality-preserving, our method uniformly outperformed other competitors, this is due to: (1) `MTLSign` does not incorporate backdoors into the model, so adversarial modifications such as FP, which are designed to eliminate backdoor, can hardly reduce our watermark. (2) `MTLSign` relies on an extra module, c_{WM}, as a verifier. As an adversary cannot tamper with this module, universal tunings such as NP have less impact. `MTLSign` can also adapt to new tuning operators by incorporating them into R_{DA}. Moreover, `MTLSign` asserts weak conditions on both the task (e.g. NLP) and the DNN architecture and is more flexible. At last, we consider the overwriting attack, where the adversary embeds its watermark into the pirated DNN. Although the adversary's ownership declaration can be falsified by the OV protocol, it is necessary that such overwriting does not invalidate the owner's watermark. The decrease of the accuracy of the watermarking branch with the overwriting epochs was recorded in Table 3. Since the decrease is uniformly bounded by 5%, overwriting does not form a threat to `MTLSign`.

Table 3: Decrease of the accuracy of the watermarking branch against watermark overwriting (in %).

Dataset	Number of overwriting epochs			
	50	150	250	350
MNIST	1.0	1.5	1.5	2.0
F-MNIST	2.0	2.5	2.5	2.5
CIFAR-10	4.5	4.5	4.5	4.5
CIFAR-100	0.0	0.5	0.9	0.9
VirusShare	0.0	0.5	0.5	0.5
IMDb	3.0	3.0	3.0	3.0
SST	2.5	3.0	3.0	2.5
PF-Pedestrian	0.5	1.0	1.0	1.0
VOC	1.3	2.0	2.1	2.1

5 Conclusion

This paper presents `MTLSign`, an MTL-based DNN watermarking scheme. We examine the basic security requirements for the DNN watermark, especially the unambiguity, and propose to embed the watermark as an additional task.

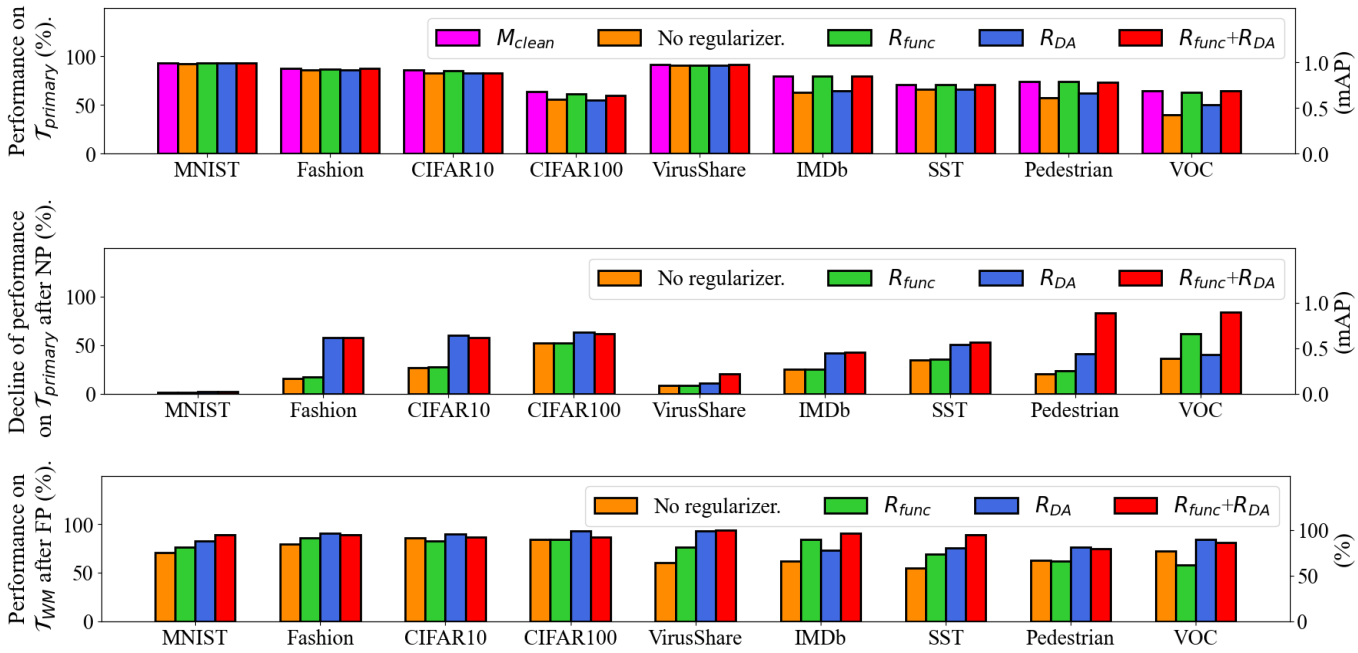


Figure 5: Ablation study on the efficacy of R_{func} and R_{DA} regarding the three metrics. For the watermarked model's performance on SS, the benchmark is mAP, otherwise is the classification accuracy.

The proposed scheme explicitly meets security requirements by corresponding regularizers. With a decentralized consensus protocol, `MTLSign` is secure against adaptive attacks. It is true that like any other white-box DNN watermarking scheme, `MTLSign` remains vulnerable to functionality equivalence attacks such as the neuron permutation. This is one of the aspects that require further effort to increase the applicability of DNN watermarks.

References

Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; and Keshet, J. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 1615–1631.

Cai, Z.; and Vasconcelos, N. 2018. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6154–6162.

Castro, M.; Liskov, B.; et al. 1999. Practical byzantine fault tolerance. In *OSDI*, volume 99, 173–186.

Darvish, R. B.; Chen, H.; and Koushanfar, F. 2019. 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*, 485–497.

Fan, L.; Ng, K. W.; Chan, C. S.; and Yang, Q. 2021. DeepIP: Deep Neural Network Intellectual Property Protection with Passports. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Ganju, K.; Wang, Q.; Yang, W.; Gunter, C. A.; and Borisov, N. 2018. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 619–633.

Guan, X.; Feng, H.; Zhang, W.; Zhou, H.; Zhang, J.; and Yu, N. 2020. Reversible Watermarking in Deep Convolutional Neural Networks for Integrity Authentication. In *Proceedings of the 28th ACM International Conference on Multimedia*, 2273–2280.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

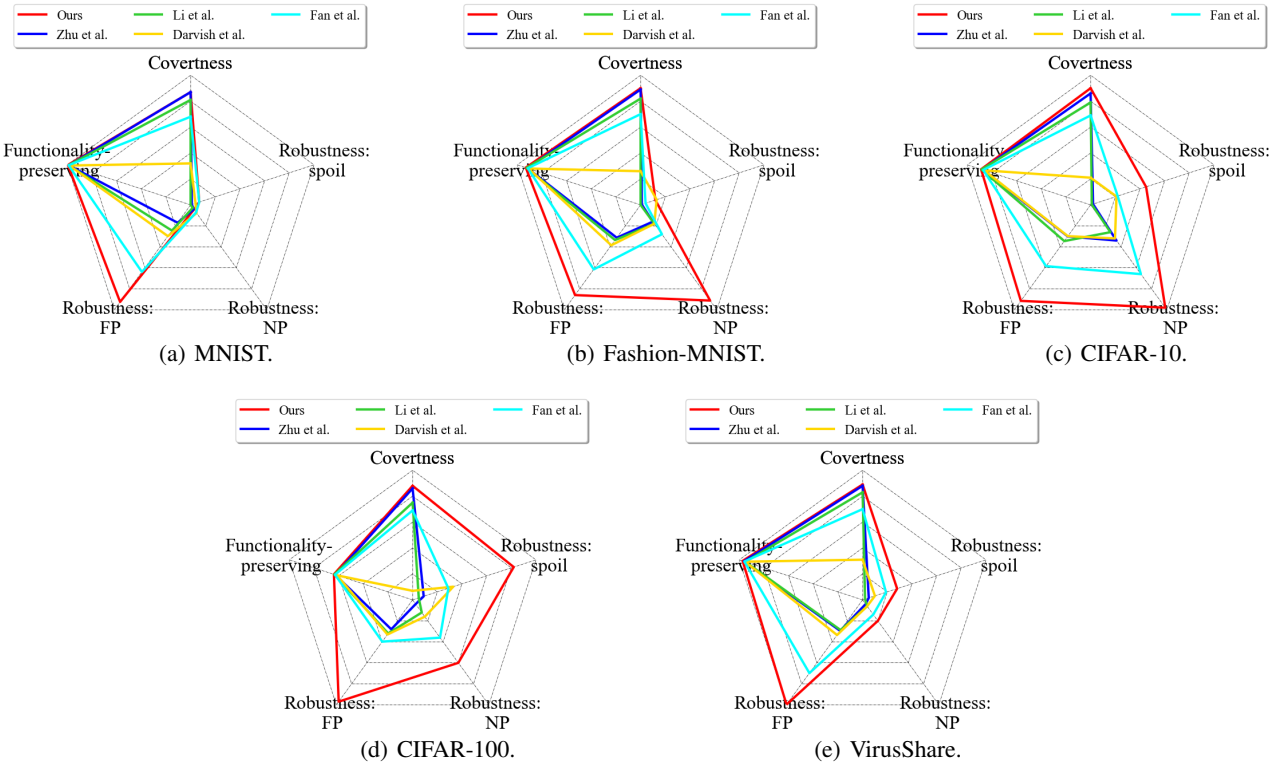
Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Le Merrer, E.; Perez, P.; and Trédan, G. 2020. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13): 9233–9244.

Li, F.; Wang, S.; and Liew, A. W.-C. 2021. Regulating Ownership Verification for Deep Neural Networks: Scenarios, Protocols, and Prospects. *IJCAI Workshop*.

Li, F.-Q.; and Wang, S.-L. 2021. Persistent Watermark For Image Classification Neural Networks By Penetrating The Autoencoder. In *2021 IEEE International Conference on Image Processing (ICIP)*, 3063–3067.

Li, H.; Willson, E.; Zheng, H.; and Zhao, B. Y. 2019a. Persistent and unforgeable watermarks for deep neural networks. *arXiv preprint arXiv:1910.01226*.



(a) MNIST. (b) Fashion-MNIST. (c) CIFAR-10.

(d) CIFAR-100. (e) VirusShare.

Figure 6: Comparison between `MTLSign` and other SOTA schemes. The covertness measures the average deviation of parameters after watermark embedding, scaling in $[0, 10^{-2}]$. The functionality-preserving measures the performance of the watermarked DNN, scaling in $[0\%, 100\%]$. The robustness against FP measures the accuracy of the watermarking branch after FP, scaling in $[0\%, 100\%]$. The robustness against NP/spoil measures the decrease of the accuracy of the primary branch when NP/the spoil attack invalidates the watermark, scaling in $[-50\%, 0\%]$/$[-10\%, 0\%]$.

Li, Y.; Koren, N.; Lyu, L.; Lyu, X.; Li, B.; and Ma, X. 2021. Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks. *arXiv preprint arXiv:2101.05930*.

Li, Z.; Hu, C.; Zhang, Y.; and Guo, S. 2019b. 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*, 126–137.

Liu, H.; Weng, Z.; and Zhu, Y. 2021. Watermarking Deep Neural Networks with Greedy Residuals. In *International Conference on Machine Learning*, 6978–6988. PMLR.

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

Liu, X.; Li, F.; Wen, B.; and Li, Q. 2020. Removing Backdoor-Based Watermarks in Neural Networks with Limited Data. *arXiv preprint arXiv:2008.00407*.

Namba, R.; and Sakuma, J. 2019. Robust watermarking of neural network with exponential weighting. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, 228–240.

Ong, D. S.; Chan, C. S.; Ng, K. W.; Fan, L.; and Yang, Q. 2021. Protecting Intellectual Property of Generative Adversarial Networks From Ambiguity Attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3630–3639.

Ongaro, D.; and Ousterhout, J. 2014. In search of an understandable consensus algorithm. In *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, 305–319.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Sener, O.; and Koltun, V. 2018. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, 527–538.

Shorten, C.; and Khoshgoftaar, T. M. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1): 60–107.

Uchida, Y.; Nagai, Y.; Sakazawa, S.; and Satoh, S. 2017. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, 269–277.

Zhang, J.; Gu, Z.; Jang, J.; Wu, H.; Stoecklin, M. P.; Huang,

H.; and Molloy, I. 2018. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 159–172.

Zhu, R.; Zhang, X.; Shi, M.; and Tang, Z. 2020. Secure neural network watermarking protocol against forging attack. *EURASIP Journal on Image and Video Processing*, 2020(1): 1–12.

A A: Derivation for the umambiguity condition

To formulate this intuition, consider the event where $\mathcal{D}_{\text{WM}}^{\texttt{key}'}$ shares $q \cdot N$ terms with $\mathcal{D}_{\text{WM}}^{\texttt{key}}$, $q \in (0,1)$. With a pseudo-random generator, it is computationally impossible to distinguish $\pi^{\texttt{key}}$ from a sequence of N randomly selected integers. The same argument holds for $\texttt{l}^{\texttt{key}}$ and a random binary string of length N. Therefore the probability of this event can be upper bounded by:

$$\binom{N}{qN} \cdot r^{qN} \cdot (1-r)^{(1-q)N} \leq \left[(1+(1-q)N) \left(\frac{r}{1-r} \right) \right]^{qN},$$

where $r = \frac{N}{2^{m+1}}$. For an arbitrary q, let $r < \frac{1}{2+(1-q)N}$ then the probability that $\mathcal{D}_{\text{WM}}^{\texttt{key}'}$ overlaps with $\mathcal{D}_{\text{WM}}^{\texttt{key}}$ with a portion of q declines exponentially.

For numbers not appeared in $\pi^{\texttt{key}}$, the watermarking branch is expected to output a random guess. Therefore if q is smaller than a threshold τ then $\mathcal{D}_{\text{WM}}^{\texttt{key}'}$ can hardly pass the statistical test in Algo. 1 with N big enough. So let

$$m \geq \log_2 \left[2N \left(2 + (1-\tau)N \right) \right]$$

and N be large enough would make an effective collision in the watermark dataset almost impossible. For simplicity, setting $m = 3 \cdot \lceil \log_2(N) \rceil \geq \log_2(N^3)$ is sufficient.

To select the threshold γ, assume that the random guess strategy achieves an average accuracy of at most $p = 0.5 + \alpha(N)$, where α is a negligible function. The verification process returns 1 iff the watermark classifier achieves binary classification of accuracy no less than γ. The demand for security is that by randomly guessing, the probability that an adversary passes the test declines exponentially with n. Let X denote the number of correct guesses with average accuracy p, an adversary succeeds only if $X \geq \gamma \cdot N$. By the Chernoff theorem:

$$\Pr\{X \geq \gamma \cdot N\} \leq \left(\frac{1-p+p \cdot \mathrm{e}^{\lambda}}{\mathrm{e}^{\gamma \cdot \lambda}} \right)^{N},$$

where λ is an arbitrary nonnegative number. If γ is larger than p by a constant independent of N then $\left(\frac{1-p+p \cdot \mathrm{e}^{\lambda}}{\mathrm{e}^{\gamma \cdot \lambda}} \right)$ is less than unity with proper λ, reducing the probability of a successful attack into negligibility.

B B: Implementation of the spoil attacks

During the spoil attack, the adversary has full knowledge of `key`, `verify`, and has obtained M_{WM}. The adversary's objective is to tune M_{WM} into M_{spoiled} in order to escape IP regulation, which means the following condition holds with a large probability:

$$\texttt{verify}(M_{\text{spoiled}}, \texttt{key}) = 0.$$

For the backdoor-based watermarking schemes, `key` is uniquely corelated with a collection of labelled triggers $\{t_n, y_n\}_{n=1}^{N}$. The spoil attack is tantamount to fitting the watermarked model on the same triggers with adversarially shuffled labels.

For the weight-based watermarking schemes, `key` reveals the places where information is hidden. So the adversary only has to replace these parameters (which is usually a small part of the entire model) with random values.

For hybrid white-box watermarking schemes with a complex `verify` module such as `MTLSign`, the adversary has to tune the watermarking branch to fit shuffled labels with the backend fixed. The loss function to be minimized can be written as:

$$\mathcal{L}(\mathbf{W}_{\text{backbone}}) = \sum_{(x,y) \in \mathcal{D}_{\text{WM}}^{\texttt{key}}} l_{\text{WM}}(y', c_{\text{WM}}(M(x|\mathbf{W}_{\text{backbone}}))),$$

in which y' is a randomly assigned label independent from y. This attack usually results in a large-scale shift of the parameters within the backbone DNN. If the adversary cannot properly fine-tune the model afterward (which is always the case in practice since otherwise the adversary would have already acquired enough data and can train its DNN from scratch) then the DNN's SOTA performance is at risk as demonstrated in the empirical studies.