Watermarking Deep Neural Networks with Greedy Residuals

Hanwen Liu [1] **Zhenyu Weng** [1] **Yuesheng Zhu** [1]

Abstract

Deep neural networks (DNNs) are considered as intellectual property of their corresponding owners and thus are in urgent need of ownership protection, due to the massive amount of time and resources invested in designing, tuning and training them. In this paper, we propose a novel watermark-based ownership protection method by using the residuals of important parameters. Different from other watermark-based ownership protection methods that rely on some specific neural network architectures and during verification require external data source, namely ownership indicators, our method does not explicitly use ownership indicators for verification to defeat various attacks against DNN watermarks. Specifically, we greedily select a few and important model parameters for embedding so that the impairment caused by the changed parameters can be reduced and the robustness against different attacks can be improved as the selected parameters can well preserve the model information. Also, without the external data sources for verification, the adversary can hardly cast doubts on ownership verification by forging counterfeit watermarks. The extensive experiments show that our method outperforms previous state-of-the-art methods in five tasks.

1. Introduction

Due to the impressive performance on predictive tasks, machine learning, and deep neural networks (DNNs) particularly have become an increasingly popular method for a variety of usages in real-world applications. Such applications need models to be well-trained, which requires massive training data and computing resources (Strubell et al., 2019). The process of training data collecting, cleansing, storing, and model training can be quite troublesome and time-consuming, not to mention the potential security and privacy issues (Shokri et al., 2017; Song et al., 2017; Salem et al., 2019), and thus the models are considered to be the valuable intellectual property of their legitimate owners. However, since the models are supposed to be exposed and serve users with some helpful information for commercial use, the threat of well-trained model theft seems to be inevitable (Oh et al., 2018; Orekondy et al., 2019). Also, although well-trained models are closely protected in general, if the cost of the theft is much lower than that of legal purchase, the value incorporated in such models makes stealing a lucrative task for malicious adversaries.

One solution to the model theft problem is watermarking DNN models (Uchida et al., 2017), which is an insightful way to identify model ownership of the legal owner. Watermarking objects is a well-studied problem in the security community under the general theme of digital watermarking. Essentially, the aim of watermarking is to mark the model with some kind of identity information, in favor of further ownership tracking. To ensure that the embedded identity information will not greatly affect the model performance, this watermarking process usually takes place during model training. An external data source, e.g., a carefully selected picture (Adi et al., 2018), is usually used for ownership verification of the watermarked model. We refer to this data source as the *ownership indicator* by analogy with the acid-based indicator in elementary chemistry.

Designing a practical yet robust watermarking method for DNN models, however, is not easy. On the one hand, with the embedding of additional information, the model may not be able to maintain its original performance. On the other hand, the model can be easily modified due to the intrinsic nature of DNN, making it possible to perform removal attacks, such as fine-tuning for transfer learning (Pan & Yang, 2010) and pruning for model compression (Li et al., 2017). Besides, in some cases, the adversary can always make an adversarial example of the ownership indicator (Fan et al., 2019) or directly overwriting the existing watermark to cast doubt on the ownership verification. The various attacks that DNN watermarks need to face are summarized in Figure 1 and described in Section 3.1 in detail. Furthermore, as for practicality and robustness, most current methods have to rely on specific DNN architectures, which greatly limits the application scenarios of watermarking.

[1]School of Electronic and Computer Engineering, Peking University. Correspondence to: Yuesheng Zhu <zhuys@pku.edu.cn>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

In this paper, we propose a novel watermarking method called *greedy residuals*. Our essential insight is that by making greedy residuals depend on less information, a more robust watermarking method can be constructed (i.e., *less is more*). There are two main aspects of the understanding of *less* here: a) we greedily select those fewer and more important model parameters for embedding, and the residuals are built upon the selected parameters; b) we hardly need to use external data sources, that is, we do not need explicit ownership indicators to complete ownership verification, since ownership information in the residuals can be verified with only fixed and simple steps.

Contributions: a) To our knowledge, it is the first robust DNN watermark method without explicit ownership indicators; b) We propose a novel idea of using less information to build a more robust watermark with detailed analysis and empirically demonstration; c) Through extensive experiments, our proposed method outperforms previous state-of-the-art methods in five tasks.

2. Related Work

At present, the DNN watermarking methods are designed for classification tasks. For both white-box methods (i.e., in need of accessing weights) and black-box methods (i.e., in no need of accessing weights), all previous methods need ownership indicators for verification, which can be divided into two following camps.

Images as ownership indicators. Currently, most watermarking methods use images as their ownership indicators. In black-box verification settings, models can be watermarked by training on some specific image-based triggers and verified by querying these triggers and observing the outputs, namely watermarking by backdoors (Adi et al., 2018; Namba & Sakuma, 2019; Guo & Potkonjak, 2018; Zhang et al., 2018). Similarly, adversarial examples can be used for watermarking DNNs (Merrer et al., 2020) by utilizing the decision boundary of models, and auto-encoder-based images can also be used as watermarks (Li et al., 2019). In addition, there are some methods using images as ownership indicators in white-box settings, e.g., by inserting additional layers and querying images called passports (Fan et al., 2019). A rather obvious shortcoming of the above methods is that most of them can only be used for image classification tasks and cannot handle text-based tasks, and some of them cannot survive through transfer learning where the output layers are usually modified as is discussed in Section 4.

Matrices as ownership indicators. It is reported that the first formal watermark-based DNN ownership verification method is using a projection matrix for watermarking (Uchida et al., 2017), which is believed to be easily detected in later work (Wang & Kerschbaum, 2019). A related method (Rouhani et al., 2019) is proposed by embedding watermarks into the probability density function of the data abstraction in DNNs, which is considered to be more flexible than previous work (Uchida et al., 2017) and more stable than merely using adversarial examples.

Unlike the above methods, our method does not need explicit ownership indicators and will be described next.

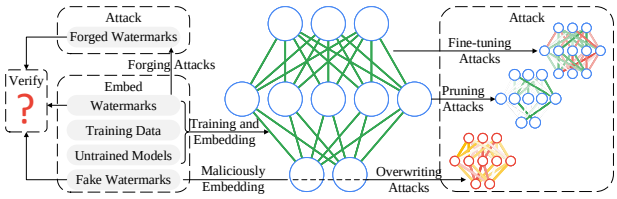


Figure 1. The threat of various attacks against DNN watermarks. There are two main types of attacks: the first is the removal attack, which includes fine-tuning attacks that modify model weights and model pruning attacks that subtract unnecessary weights; the second is the ambiguity attack, which includes forging attacks that aim to forge the counterfeit watermark without modifying the model and overwriting attacks that overwrite the already watermarked model with adversarial watermarks.

3. Greedy Residuals

In this part, we will discuss the properties of good DNN watermarks and how our method meets these properties.

3.1. Background on Watermarking

Threat model. We investigate an adversary *Eve* with the goal of falsely claiming the ownership of a watermarked model and using the model illegally. The following assumptions have been made for the interest of *Eve*: a) *Eve* has already got the model and tries to perform attacks to make the existing watermark invalid; b) The attack is ineffective if it significantly impairs the original performance of the model; c) *Eve* either has limited data or limited computing resources, because an attack is not necessary if *Eve* has the ability to train the model from scratch legally.

Objectives of the watermark. In addition to having *authenticity* (i.e., the watermark can uniquely identify the owner of the model), a watermark also needs to have the following properties: a) *Universality*: The watermark should have as few requirements as possible on the model itself and cannot only be applied to a small number of model architectures; b) *Capacity*: The watermark should be able to contain large amounts of identity information. c) *Secrecy*: The adversarial process of detecting the presence of the watermark should be as difficult as possible. d) *Fidelity*: The impairment on the model performance, which is

caused by the watermark, should be as little as possible; e) *Removal-resistance*: The watermark should be robust to removal attacks, including model fine-tuning for transfer learning and model pruning for neural compression; f) *Forging-resistance*: During verification, *Eve* can perform forging attacks by forging the counterfeit watermark (i.e., counterfeit ownership indicator) to falsely claim ownership of the model. Such a counterfeit watermark is usually an adversarial example and extracted by some transformations without modifying the model weights. g) *Overwriting-resistance*: Once *Eve* knows the underlying watermarking method, a re-embedding process can be executed to overwrite the existing watermark in the model.

We refer to forging attacks and overwriting attacks as ambiguity attacks in this paper, because both attacks can make the original owner unable to claim the unique ownership, thus casting ambiguity of the model ownership verification. To distinguish these two ambiguity attacks, assuming the watermark $\mathcal{W}$ has already been embedded into the model F_θ, the main difference between these two attacks is that the overwriting attacks change the model parameters θ by re-embedding *Eve*'s watermark $\mathcal{W}_{Eve}$, yet the forging attacks only forge the watermark $\mathcal{W}_F$, which can be regarded as the forged ownership indicator during verification, without modifying the model. The knowledge for performing forging attacks usually comes from the valid watermark $\mathcal{W}$, and typically the forged watermark $\mathcal{W}_F$ is statistically similar to $\mathcal{W}$ but looks different (i.e., $\mathcal{W}_F$ is statistically different from $\mathcal{W}_{Eve}$ yet contains the adversarial identity of *Eve*). Unlike other approaches, our proposed method only relies on less information of parameters, which is robust against multiple attacks while proving legitimate ownership, and will be described in the following in detail.

3.2. Embedding Greedy Residuals

A deep learning model F_θ with trainable parameters θ can be optimized using the dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ and a loss function L_θ by solving:

$$\arg\min_{\theta} \sum_{i=1}^{N} L_\theta(\mathbf{x}_i, y_i) + E_\theta \tag{1}$$

where E_θ is a penalty term for regularizing models. Since penalty terms can impose restrictions on models while optimizing, we denote E_θ as the *embedding term* which can also be used to embed the watermark $\mathcal{W}$.

Watermarking while training. As it is vital to maintain the original performance of the watermark-free model, we perform watermarking DNN models while training. Inspired by the hinge loss function (Rosasco et al., 2004), the embedding term E_θ in Equation 1 aims to embed the ownership information into the model, which can be defined as:

$$E_\theta = \lambda \sum_{i=1}^{b} \mathrm{ReLU}(\mu - \omega_i(\alpha)\psi_i) \tag{2}$$

where ψ_i is the residual information which is the sum of values in the extracted weights and will be described in Equation 8 later, α is the ownership information signature, b is the actual number of bits of the ownership signature (i.e., α), the b-bit sequence $\{\omega_i(\alpha)\}_{i=1}^b \in \{-1, 1\}^b$ is the sign factor using α as the input, λ is the adjustable parameter, and μ is the threshold. The adjustable parameter $\lambda = 0.01$ and the threshold $\mu = 0.1$ by default unless stated otherwise.

The sign factor sequence $\omega(\alpha) = \{\omega_i(\alpha)\}_{i=1}^b$, which controls the signs of $\psi = \{\psi_i\}_{i=1}^b$, indirectly forces the model to remember the signs of the ownership signature $\alpha \in \{0, 1\}^b$ using private key κ_s (we take κ_p as the corresponding public key):

$$\alpha = S(\kappa_s, msg) \tag{3}$$

where msg is the message to be signed (e.g., a copyright statement of the legitimate person or party), and S is the signature function. In experiments we use the RSA (Rivest et al., 1978) algorithm, then there is $b = 256$ in Equation 2.

More specifically, the sign factor $\omega(\alpha) = \{\omega_i(\alpha)\}_{i=1}^b \in \{-1, 1\}^b$ is described in the following equation:

$$\omega(\alpha) = \{\mathrm{sgn}(\alpha_i)\}_{i=1}^b \tag{4}$$

where sgn is used to project bits as $\{1, 0\} \xrightarrow{\text{sgn}} \{1, -1\}$. During training, the embedding term E_θ in Equation 1 functions as guidance to encourage ψ_i to stay the same sign as that of $\omega_i(\alpha)$, and the threshold μ will enlarge the gap between different signs of values, and thus we can take the watermark $\mathcal{W} = \{\alpha\}$ in our proposed method.

Weight extraction. Generally, there are two ways to embed the watermark: embed the watermark in the model hyper-parameters (e.g., the model architecture) or embed it in the trainable model parameters (e.g., the weights and the bias in the model). Here, we embed the watermark into the model weights in favor of subsequent ownership tracking. First of all, we need to decide which weights should be chosen to embed the watermark. The flattened weight $w^l \in \mathbb{R}^m$ in layer l, where m is decided by the original DNN model F_θ, are extracted from the model F_θ and transformed to $\gamma^l \in \mathbb{R}^d$ (e.g., $d = 256 \times 253 = 64768$) by using the following convolution $*$ operation:

$$(w^l * k)(t) = \sum_{u=-\infty}^{\infty} w^l(u) k_t(u), \quad t \in [1, d] \tag{5}$$

where t is the position of the target element in γ^l. In order to make each weight equally important, we try to design an average operation and thus small groups of whether high or low values are compensated by the average value, and thus the convolution kernel function are defined as:

$$k_i(x) = \begin{cases} \frac{1}{\sigma_i} & \text{if } \lfloor \frac{(i-1)m}{d} \rfloor < x \leq \lceil \frac{im}{d} \rceil \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where the kernel width sequence $\sigma = \{\sigma_i\}_{i=1}^d$ is the output of the following ceiling function $\lceil \cdot \rceil$ and floor function $\lfloor \cdot \rfloor$:

$$\sigma = \{\lceil \frac{im}{d} \rceil - \lfloor \frac{(i-1)m}{d} \rfloor\}_{i=1}^d \tag{7}$$

Actually, in order to match the target shape (i.e., transform the shape of $w^l \in \mathbb{R}^m$ to that of $\gamma^l \in \mathbb{R}^d$), the above convolution plays the role of *1-d average pooling*, which builds on the statistical robustness of the average operation when $m > d$, and the convolution also works when $m \leq d$. Considering the fact that the ownership information signature α has b bits, in order to embed the ownership information as is mentioned in Equation 2, the extracted weight $\gamma^l \in \mathbb{R}^d$ needs to be reshaped into $\gamma^l \in \mathbb{R}^{b \times b'}$ and $b' = \lfloor \frac{d}{b} \rfloor$ in general. In most cases, d is a multiple of b (e.g., $b' = 253$ when $b = 256$ and $d = 64768$).

Residual construction. After extracting the weights (i.e., γ^l), we can construct residuals upon them. As one of the main aspects of our insight *less is more*, we define the component ratio η for embedding since only those fewer and more important weights are selected greedily. Then we focus on the i-th row $\gamma^l_{i,*} \in \mathbb{R}^{b'}$ in $\gamma^l \in \mathbb{R}^{b \times b'}$ and refer the $\lfloor b'\eta \rfloor$-biggest-absolute-value element set as $\mathcal{B}_i$. Since the element value in a row of γ^l is either positive or negative, naturally the greedy residual ψ_i in i-th row is defined as the average of element values in $\mathcal{B}_i$ in the i-th row:

$$\psi_i = \frac{1}{b'} \sum_{j \in \mathcal{B}_i} j \tag{8}$$

Please note that both positive and negative numbers may appear in the set $\mathcal{B}_i$, and they are all elements with larger absolute values in the i-th row $\gamma^l_{i,*}$. Without loss of generality, we only take half of all values to calculate, so we have $\eta = 0.5$ and the $\lfloor \frac{b'}{2} \rfloor$-biggest elements will be calculated.

The embedding progress is formally described in Algorithm 1. Since the ownership signature α only needs to be calculated once in the beginning, the primary computational complexity of the algorithm is the calculation of the embedding term E_θ in Equation 1. The calculation of E_θ is composed of the process of extracting the weights and the process of constructing residuals. With regard to the process of extracting the weights, as is mentioned in Equation 5 and Equation 6, this process is essentially viewed as the convolution operation through a convolution kernel with a width in the sequence σ. Since this process aims to map the flattened weight $w^l \in \mathbb{R}^m$ in layer l to $\gamma^l \in \mathbb{R}^d$, the time complexity of weight extraction is $\mathcal{O}(d \cdot \sigma_{\max}) = \mathcal{O}(m)$, where $\sigma_{\max}$ denotes the max width in kernel width sequence σ. As for the process of constructing residuals, the most time-consuming part is the sorting operation, which has a complexity of $\mathcal{O}(d \log b')$. Therefore, the overall time complexity of the calculation of the embedding term E_θ is $\mathcal{O}(m + d \log b')$. Considering that the above two processes are essentially convolution operations and sorting operations, the space complexity of the calculation of the embedding term E_θ is $\mathcal{O}(d \cdot \sigma_{\max} + \log b') = \mathcal{O}(m + \log b')$.

Also, it is the dependence imposed by E_θ on less but more important weights for the primary task during training that makes our proposed method robust against multiple attacks including fine-tuning attacks, pruning attacks and overwriting attacks, which will be described in Section 4.

Algorithm 1 Embedding of Greedy Residuals

Parameter: public-key cryptography key set $\kappa = \{\kappa_s, \kappa_p\}$, message to be signed msg, dataset $\mathcal{D}$ and predefined model F_θ to be watermarked with all the hyperparameters.

```
Initialize dataset D and model F_θ with parameters θ;
Initialize ownership signature α using κ_s and msg;
for it = 1, iteration do
    Sample minibatch of n samples from D;
    Compute cross-entropy loss L_θ;
    Update θ by descending the gradient: ∇_θ L_θ;
    Compute the embedding term E_θ using α;
    Update θ by descending the gradient: ∇_θ E_θ;
end for
return F_{θ*} with optimized parameters θ*;
```

3.3. Verification

There are two things that need to be done in the verification progress. First, determine whether the identity information (e.g., the signature of a person or party) exists in the constructed residuals ψ of the watermarked model. Then determine whom the identity information belongs to. In fact, the verification process of our proposed method is similar to the embedding process, that is, after extracting the weight γ^l to construct residuals ψ, it verifies whether the signature α exists or not. If the signature α does exist, then figure out whom this signature actually belongs to, by using the public key κ_p corresponding to the private key κ_s, since α is signed using κ_s in Equation 3 and only the corresponding public key κ_p can complete the verification. From the above verification process, our method does not need an explicit

external data source, namely ownership indicators, and the ownership information can be verified with only fixed and simple steps. Therefore, the adversary *Eve* cannot forge an adversarial example to perform forging attacks, thus making the verification of our proposed method an unforgeable verification.

There is a chance that one may argue the hyper-parameters of the watermark serve as an ownership indicator. However, our proposed method is still robust. On the one hand, it is very difficult for *Eve* to reverse the msg that suits *Eve* from the encrypted signature α, which is proved by the RSA encryption method (Rivest et al., 1978). On the other hand, the ownership information contained in the residuals is limited. According to Proposition 1, given a 256-bit encrypted signature, for example, a DNN with 18 layers where there are 64768 extracted weight values for each layer, the probability of claiming that certain ownership information extracted from the DNN is $P \leq 4 \times 10^{-74}$. Also, we run the empirical evaluations in Section 4 to demonstrate the robustness of our proposed method.

Proposition 1. *Given a b-bit binary code and a watermarked model with M layers where there are d weights for each layer, the probability of the binary code extracted from the residuals of the watermarked model matching the given binary code is bounded as: $P \leq \frac{M\lfloor \frac{d}{b} \rfloor}{2^b}$*

Proof. The process of extracting the binary code from the residuals of the watermarked model includes weight extraction and residual construction. For weight extraction, there are M choices of selecting the layer from a watermarked model with M layers. For residual construction, as in the layer there is $1 \leq \lfloor \frac{d}{b} \rfloor$, there are $\lfloor \frac{d}{b} \rfloor$ choices of constructing the residual information from the selected layer according to Equation 8. Therefore, there are at most $M\lfloor \frac{d}{b} \rfloor$ choices of residual construction for a watermarked model with M layers. Since there are 2^b combinations for a b-bit binary code, the probability of the binary code extracted from the residuals of the watermarked model matching the given binary code is $P \leq \frac{M\lfloor \frac{d}{b} \rfloor}{2^b}$.

□

4. Experiments

In addition to using models without watermark as the baseline, we perform empirical evaluations of our proposed greedy residuals against three other approaches:

- Passport method (Fan et al., 2019): We focus on the *passport verification scheme 2* mentioned in their paper, which embeds selected passport images into the model and verifies a watermarked model based on the model performance and the pre-defined signature with given passports. Since a new passport layer needs to be inserted during the verification process, this method is in a white-box setting.
- Backdoor method (Adi et al., 2018): It also uses images as the watermarks, with a set of backdoor classification rules. By intentionally showing the set of backdoor rules in a watermarked model, one can prove the ownership in a black-box setting.
- Weight-hash method: To clarify the difference between our method and simple hashing, we naively take the hash value of the file of model weights as the watermark. It is worth noting that this method also does not require too much extra information during verification, and it is in a white-box setting.

Experiment setup. For a fair comparison, we refer to (Fan et al., 2019) to build the experiment setup. We run experiments of AlexNet (Krizhevsky et al., 2012) and ResNet-18 (He et al., 2016) on Caltech-101, Caltech-256 (Li et al., 2006), CIFAR-10 and CIFAR-100 (Krizhevsky, 2012) for image classification tasks. Also, we run TextCNN (Kim, 2014) and LSTM (Hochreiter & Schmidhuber, 1997) on IMDB-2 (Maas et al., 2011) and TREC-6 (Li & Roth, 2002) for sentiment and question classification respectively. We also evaluate our proposed method on ImageNet (Russakovsky et al., 2015) subset[1] for image classification.

The source codes of greedy residuals and the corresponding datasets are publicly available[2].

4.1. Low-impairment

Universality. The restrictions imposed on the model by the watermark are evaluated in Table 1. We notice that the Passport method is only suitable for a small part of the model architectures, e.g., models with batch normalization layers (Ioffe & Szegedy, 2015). Since the watermark depends on specific images, both the Passport method and the Backdoor method are only applied to image classification tasks. Our proposed method has little requirement on the model architecture, because residuals are only constructed on a few weights. Considering that modern DNNs usually have more parameters than are statistically needed to fit training data, our proposed method can be easily applied to a tremendous amount of DNNs. Therefore, our proposed method has the property of *universality*.

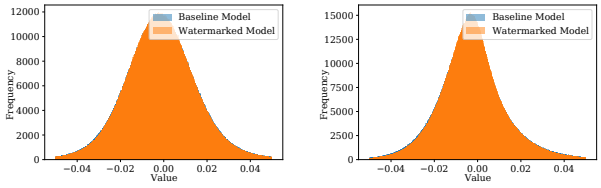
Secrecy. Following *Kerckhoffs's Principle*, we assume that our proposed method is known to adversaries. However, as *secrecy* can reduce the possibility of potential attacks,

[1]The subset of the first 100 classes in ImageNet is used as the entire dataset for efficiency.

[2]https://github.com/eil/greedy-residuals

Table 1. Experiment results of low-impairment. Baseline means the accuracy of a normal-training model without the watermark on the test set. Ours denotes the model accuracy on the test set watermarked by greedy residuals. N/A means the results are not applicable because such watermarks cannot be applied to the model. The accuracy inside the bracket represents the accuracy of the model with batch normalization. The watermark detection rates are omitted because all of the rates are 100% when watermarked models converge.

Dataset	Baseline (%)		Ours (%)		Passport (Fan et al., 2019) (%)		Backdoor (Adi et al., 2018) (%)		Weight-hash (%)	
	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18
Caltech-101	63.95 (68.72)	66.36 (71.19)	64.43 (67.76)	64.91 (71.14)	N/A (48.42)	N/A (68.96)	63.52 (68.88)	64.59 (70.76)	63.95 (68.72)	66.36 (71.19)
Caltech-256	36.80 (44.79)	42.37 (56.09)	36.84 (43.46)	42.72 (54.29)	N/A (39.75)	N/A (51.53)	36.68 (44.54)	41.40 (53.57)	36.80 (44.79)	42.37 (56.09)
CIFAR-10	90.09 (90.98)	93.41 (94.90)	89.29 (90.69)	93.52 (94.94)	N/A (90.72)	N/A (94.89)	89.76 (90.77)	93.01 (94.69)	90.09 (90.98)	93.41 (94.90)
CIFAR-100	63.58 (68.26)	69.59 (76.74)	63.15 (67.91)	69.56 (76.49)	N/A (63.91)	N/A (74.11)	63.15 (67.99)	69.27 (76.38)	63.58 (68.26)	69.59 (76.74)
Dataset	LSTM	TextCNN	LSTM	TextCNN	LSTM	TextCNN	LSTM	TextCNN	LSTM	TextCNN
IMDB-2	87.24	88.65	86.93	88.28	N/A	N/A	N/A	N/A	87.24	88.65
TREC-6	83.00	92.00	82.60	93.00	N/A	N/A	N/A	N/A	83.00	92.00



(a) Caltech-101 (b) Caltech-256

Figure 2. Distribution of ResNet-18 weights with or without watermarks. Embedding greedy residuals will not change the statistical features of the model, thus making our proposed method covert.

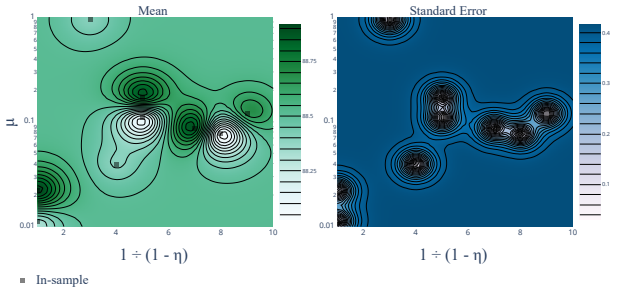


Figure 3. The impacts of different watermark parameters on the accuracy of the model. Experiments evaluate the model accuracy of AlexNet on CIFAR-10. The accuracy when $1 \div (1 - \eta) = 0$ means the model accuracy when full weights are selected (i.e., $\eta = 1$).

the watermark should not significantly change the statistical features of models. As is depicted in Figure 2, both the baseline model and the watermarked model follow almost the same distribution, and it is difficult for an adversary to tell whether the model is embedded with our proposed watermark.

Fidelity. To illustrate the effectiveness of our proposed greedy residuals, we evaluate the accuracy of baseline models and watermarked models on the test set to check the impairment caused by watermarking. We train the networks for 200 epochs with the multi-step learning rate which schedules the learning rate as 0.01, 0.001 and 0.0001 between epochs 1 to 100, 101 to 150 and 151 to 200 respectively. We also use the weight decay and the momentum which are 5×10^{-4} and 0.9 respectively. The batch size of the training set is set as 64, and the batch size of the test set for image classification tasks is 128, and for text classification tasks is 64. As is shown in Table 1, we find out that the other three methods, except the Weight-hash method, all have a certain impact on the model performance, and our proposed method outperforms the Passport method and the Backdoor method in most experiments. As most other approaches cannot process text classification tasks, even on image classification tasks which other methods are good at, the accuracy of our proposed method is 4.30% higher than that of the Passport method on average and 0.13% than that of the Backdoor method on average. Compared with the baseline where models are normally trained, the maximum accuracy drop caused by our proposed method is only 1.8% for ResNet-18 with batch normalization on Caltech-256, and in the 6 tasks including text classification, the accuracy has slightly increased. It is empirically proved that our proposed method has the property of *fidelity*. Though the weight-hash method has no effect on the accuracy of the DNN models, the following experiments show that simply hashing the weight file is not a robust watermarking method.

Impacts on accuracy. As is mentioned previously in Section 3.2 that the embedding term E_θ can function as an additional term for regularizing parametric deep learning models, we try different hyper-parameters of greedy residuals for 9 iterations with the objective of model accuracy regardless of the watermark detection rates. Figure 3 presents the impacts of our proposed method on the model accuracy when different parameters are set for AlexNet on CIFAR-10. In Table 1 we notice that the typical accuracy of AlexNet on CIFAR-10 is 89.29% when $\eta = 0.5$ (i.e., half of the extracted weight values are selected), yet in this contour figure we observe that most of the in-sample point values are around 88.50%. In Section 4.3 we will dive deeper and figure out whether the robustness or accuracy will change

Table 2. Evaluations on ImageNet subset. The subset consists of the first 100 classes of ImageNet. The accuracy inside the bracket denotes the accuracy of the model with batch normalization. The watermark detection rates are omitted because all of the rates are 100% when watermarked models converge.

Dataset	Baseline (%)		Ours (%)	
	AlexNet	ResNet-18	AlexNet	ResNet-18
ImageNet Subset	54.09 (57.34)	61.72 (67.79)	53.33 (56.79)	61.49 (67.69)

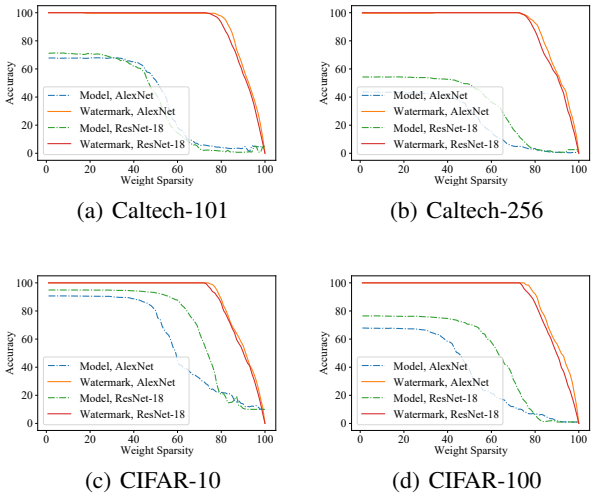
when all the extracted values are selected (i.e., $\eta = 1$). Also, we evaluate our proposed method on ImageNet subset, and the results are shown in Table 2. It is revealed that our proposed method can achieve 100% watermark detection rate while largely preserving the model accuracy.

Capacity. In particular, we use RSA as our public-key cryptography algorithm, to generate a 256-bit sequence as the signature of our copyright statement. Since the copyright statement is essentially a string sequence, no matter how long it is, it will be converted into a 256-bit sequence, thus achieving *capacity* of our proposed method.

4.2. Robustness

Fine-tuning resistance. When the labels of the source dataset and the target dataset are both available, fine-tuning is a method for transfer learning, which transfers the knowledge to solve new tasks, and the output layer is always altered during transfer learning. As is summarized in Table 3, we evaluate the worst case, where the adversary replaces the last output layer and retrains the parameters of all layers. The Backdoor method is compromised immediately since this method relies heavily on the output layer, and because the types of target tasks are generally different, adversaries are unlikely to keep the output layer in transfer learning. For the Passport method, the watermark detection rate has slightly declined in more than half of the experiments. However, not only is the accuracy of our method higher than that of the Passport method, but also the watermark detection rate of our method is 100% in all experiments. That is to say, our proposed method not only achieves the robustness of fine-tuning attacks, but also maintains relatively high model accuracy. One explanation for this robustness is that the threshold defines the strength of the constructed residuals, and the adversary cannot remove the watermark without destroying the original knowledge from the source dataset.

Weight pruning resistance. As one of the methods for model compression, weight pruning is an efficient way to reduce the storage and computation costs of DNNs. Since weight pruning will not only affect the size and operation speed of the model, but also the accuracy of the model, the pruning rate is actually a trade-off between computational cost and accuracy. We consider ℓ_1-norm pruning to evaluate our proposed method, as is depicted in Figure 4. The empirical results show that generally our proposed method is insensitive to the change of weight sparsity, and even in the worst case (i.e., ResNet-18 on CIFAR-10), the watermark detection rate begins to drop obviously only when the weight sparsity is as high as about 75%, where the model accuracy is only around 44.74%. Consequently, the weight pruning attack is invalid for our proposed method, because when the watermark detection rate really starts to decline, the model itself has already become useless. Since our proposed method only selects the more important part of the model weights to embed ownership information, we achieve resistance against weight pruning attacks.



(a) Caltech-101 (b) Caltech-256

(c) CIFAR-10 (d) CIFAR-100

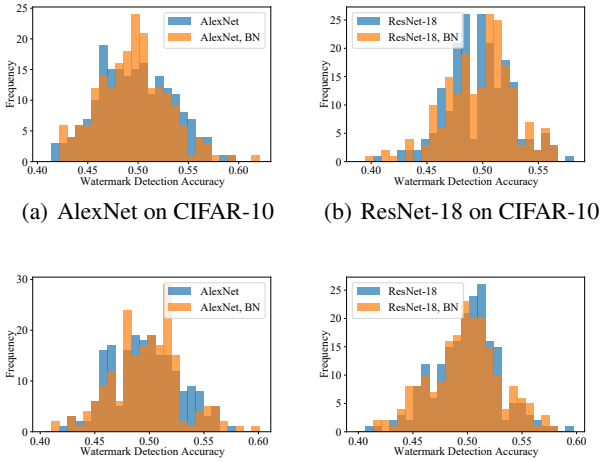
Figure 4. Evaluation results of weight pruning resistance (*removal-resistance*). Model represents the prediction accuracy of the pruned model on the test set. Watermark is the detection accuracy of the embedded greedy residuals under a specific weight sparsity rate by weight pruning.

Robust to permutation attacks. An adversary may permute the order of the nodes such that γ^l in layer l, as defined in Equation 5 will have different values since the distance between elements will be different. According to the threat model in Section 3.1, any attacks that may impair the original performance of the model will be considered invalid for the interest of the adversary. As is described in Table 4, we assume that the adversary permutes kernels in the watermarked convolution layer. Results show that our proposed method is robust against the neuron node permutations, since all the watermark detection rates are more than 95% when nodes are permuted in the experiments.

Robust to scaling attacks. For scaling attacks, neural networks with ReLUs are invariant to scaling in the sense that

Table 3. Experiment results of fine-tuning resistance (*removal-resistance*). Baseline means the accuracy of the watermark-free model under fine-tuning with the appropriate training settings. Accuracies inside and outside the bracket denote the detection rate of the watermark and the accuracy of the trained model respectively. All model architectures are built with batch normalization for comparison.

Source	Target	Baseline (%)		Ours (%)		Passport (Fan et al., 2019) (%)		Backdoor (Adi et al., 2018) (%)		Weight-hash (%)	
		AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18	AlexNet	ResNet-18
CIFAR-10	CIFAR-100	66.11	72.88	65.51 (100)	73.36 (100)	63.30 (99.78)	70.60 (98.67)	65.30 (0)	73.45 (3.00)	66.11 (0)	72.88 (0)
	Caltech-101	73.39	76.93	72.48 (100)	76.61 (100)	71.06 (100)	70.98 (99.88)	73.39 (8.00)	76.82 (2.00)	73.39 (0)	76.93 (0)
	Caltech-256	46.62	53.05	46.26 (100)	52.44 (100)	42.55 (99.87)	44.81 (97.93)	46.71 (0)	53.55 (0)	46.62 (0)	53.05 (0)
CIFAR-100	CIFAR-10	89.44	93.70	89.45 (100)	93.64 (100)	87.95 (100)	93.09 (100)	89.18 (8.00)	93.50 (11.00)	89.44 (0)	93.70 (0)
	Caltech-101	77.15	79.94	75.27 (100)	81.65 (100)	72.77 (100)	74.01 (100)	75.38 (8.00)	80.36 (7.00)	77.15 (0)	79.94 (0)
	Caltech-256	49.57	56.43	49.55 (100)	56.32 (100)	45.20 (99.91)	51.84 (99.88)	49.60 (0)	56.89 (0)	49.57 (0)	56.43 (0)



(a) AlexNet on CIFAR-10 (b) ResNet-18 on CIFAR-10

(c) AlexNet on CIFAR-100 (d) ResNet-18 on CIFAR-100

Figure 5. Evaluation results of forging attacks against greedy residuals. The signatures are randomly generated to evaluate whether other identity information can be extracted from the watermarked model. BN denotes models with batch normalization.

if the weights in the incoming edges to a given neuron in a hidden are multiplied or divided by a factor $\delta > 0$ and all the edges that go out of this node are multiplied or divided by $\frac{1}{\delta}$ then the network will evaluate to the same result. Our proposed method is robust against scaling attacks. During verification we simply check whether the extracted signs of residuals match that of the given signature α in Equation 4. When the weights are multiplied or divided by a factor $\delta > 0$, the signs will not change, and the watermark detection rates will not change, too. As is shown in Table 5,

Table 4. Results of permutation attacks against greedy residuals. Under a specific number of permuted nodes, accuracies inside and outside the bracket denote the watermark detection rates of AlexNet models with or without batch normalization respectively.

Nodes	CIFAR-10 (%)	CIFAR-100 (%)	Caltech-101 (%)	Caltech-256 (%)
2	100 (98.05)	98.05 (98.05)	98.83 (97.66)	98.05 (98.05)
3	99.22 (96.48)	96.48 (98.44)	98.05 (97.66)	97.27 (98.05)
4	98.05 (96.48)	96.09 (97.27)	95.70 (97.66)	95.70 (97.27)

Table 5. Results of scaling attacks against greedy residuals. Accuracies inside and outside the bracket denote the watermark detection rates of AlexNet models when the weights are multiplied by δ and divided by δ, respectively.

δ	CIFAR-10 (%)	CIFAR-100 (%)	Caltech-101 (%)	Caltech-256 (%)
10	100 (100)	100 (100)	100 (100)	100 (100)
100	100 (100)	100 (100)	100 (100)	100 (100)

Table 6. Results of forging attacks. Accuracies inside and outside the bracket denote the watermark detection rates with or without batch normalization respectively. N/A means not applicable.

Dataset	Passport (Fan et al., 2019) (%)		Backdoor (Adi et al., 2018) (%)	
	AlexNet	ResNet-18	AlexNet	ResNet-18
CIFAR-10	N/A (100)	N/A (100)	100 (100)	100 (100)
CIFAR-100	N/A (100)	N/A (100)	100 (100)	100 (100)

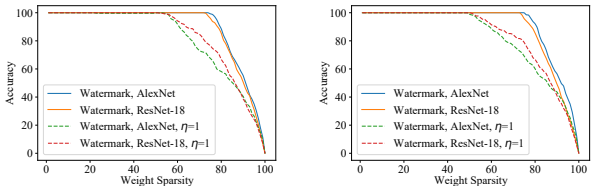
experiment results show that all the watermark detection rates remain 100%, which demonstrates the robustness of our proposed method against scaling attacks.

Robust to forging attacks. Table 6 and Figure 5 show the results of forging attacks where the counterfeit watermark is supposed to be illegally verified in a watermarked DNN. For the Passport method and the Backdoor method, such counterfeit watermarks can be successfully forged with the 100% watermark detection rates, which means these methods are totally compromised in forging attacks and thus ownership ambiguity emerges. As for our proposed method, since Proposition 1 proves that it is unlikely to compromise our proposed method by forging adversarial examples or other counterfeits of existing watermarks, we randomly generate a batch of 200 signatures to verify whether it is possible to detect these forged signatures from the watermarked model. The evaluation results reveal that the detection accuracy of the forged watermark is only near 50% for all experiments, which is because there are only two positive and negative values. Since there is only a small probability of forging attacks, our proposed method fulfills the requirement of *forging-resistance*.

Robust to overwriting attacks. From the perspective of whether the adversary has the knowledge of the original

Table 7. Results of overwriting attacks with or without original training data. Accuracies outside and inside the bracket denote the watermark detection rates of existing watermark and adversarial watermark respectively. Models are built with batch normalization.

Train with Data	CIFAR-10 (%)		CIFAR-100 (%)	
	AlexNet	ResNet-18	AlexNet	ResNet-18
Ours	100 (53.52)	99.22 (49.22)	100 (49.22)	99.61 (47.66)
Passport(Fan et al., 2019)	100 (100)	100 (100)	99.91 (99.91)	100 (100)
Backdoor(Adi et al., 2018)	85.00 (98.00)	93.00 (100)	87.00 (97.00)	95.00 (100)
Train without Data	AlexNet	ResNet-18	AlexNet	ResNet-18
Ours	100 (47.27)	98.44 (52.34)	100 (42.97)	97.66 (53.13)
Passport(Fan et al., 2019)	100 (100)	100 (100)	100 (100)	100 (100)
Backdoor(Adi et al., 2018)	13.00 (78.00)	16.00 (74.00)	10.00 (78.00)	8.00 (82.00)



(a) CIFAR-10 (b) CIFAR-100

Figure 6. Evaluation of the comparison between different watermark parameters under pruning attacks. Watermark is the detection accuracy of the embedded watermark under a specific weight sparsity rate. All models are built with batch normalization.

training data set, as is shown in Table 7 we consider two types of overwriting attacks. When the original training dataset is accessible, for both the Passport method and the Backdoor method, the adversarial watermark has been successfully re-embedded, since the detection rates of these overwritten watermarks are more than 97%. To make matters worse, the watermark of the Backdoor method is even slightly erased, and its detection accuracy is lower than that of the adversarial watermark by 5% at least. The situation is basically similar when the original training dataset is inaccessible, as all the adversarial watermarks have been successfully re-embedded. It is worth noting that the watermark detection rates of the Backdoor method drop sharply at this time, because the only factor of the model change is the re-embedding of adversarial watermarks. Therefore, the model changes more drastically and the existing watermarks are no longer valid.

In all the above evaluations of overwriting attacks, our proposed method is so robust that the detection rates of real watermarks almost stay 100% with few exceptions, while the adversarial watermarks are only around 49.42% which is equivalent to randomly guessing. Thus our proposed method has *overwriting-resistance* compared with other methods.

4.3. A Case Study on All Selected Weights

Previously we discuss only fewer yet more important weights are selected for embedding, since it is better than choosing all the extracted weights. In this part we evaluate the impact of the component ratio η values on robustness and impairments through experiments, where $\eta = 0.5$ represents our default watermark parameter and $\eta = 1$ means that all extracted weights are selected for constructing residuals. Table 8 shows the influence of different η on model performance and watermark detection rates, where we find out that the model accuracy on the test set and the watermark detection rate decrease by up to 0.22% and 0.78% respectively, when all weights are selected. As for pruning which is depicted in Figure 6, the watermark is more vulnerable to attacks when all weights are selected, since the curve of $\eta = 1$ drops even when weight sparsity is around 55% and 50% on CIFAR-10 and CIFAR-100 respectively, and the experiments demonstrate that our proposed method achieves more robustness by depending on less but more important model parameters.

Table 8. Results of comparison between half or all selected weights. Accuracies outside and inside the bracket denote the model accuracy on the test set and the watermark detection rate respectively. Tuning to CIFAR-10 means the source dataset is CIFAR-100 while the target dataset is CIFAR-10. Tuning to CIFAR-100 means the source dataset is CIFAR-10 while the target dataset is CIFAR-100. All models are built with batch normalization.

Comparison Item	Ours (%)		Ours, $\eta = 1$ (%)	
	AlexNet	ResNet-18	AlexNet	ResNet-18
CIFAR-10 Training	90.69 (100)	94.94 (100)	90.61 (100)	94.88 (100)
CIFAR-100 Training	67.91 (100)	76.49 (100)	67.69 (100)	76.31 (100)
Tuning to CIFAR-10	89.45 (100)	93.64 (100)	89.09 (99.61)	93.64 (100)
Tuning to CIFAR-100	65.51 (100)	73.36 (100)	65.14 (99.22)	73.13 (100)

5. Conclusions

Protecting the intellectual properties of DNN model owners by watermarking is a decent way, yet there are also many attacks against this approach. In this paper, we provide the insight of *less is more* in DNN ownership protection, and propose the first watermarking method that does not explicitly use ownership indicators for verification with superior robustness based on the insight. As for evaluating our proposed watermarking method, we consider multiple attacks and carry out extensive experiments, and it is empirically proved that our method achieves state-of-the-art and outperforms other methods in five fundamental tasks.

Acknowledgements

This work was supported in part by the National Innovation 2030 Major S&T Project of China under Grant 2020AAA0104203, and in part by the Nature Science Foundation of China under Grant 62006007.

References

Adi, Y., Baum, C., Cissé, M., Pinkas, B., and Keshet, J. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In Enck, W. and Felt, A. P. (eds.), *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pp. 1615–1631. USENIX Association, 2018.

Fan, L., Ng, K., and Chan, C. S. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 4716–4725, 2019.

Guo, J. and Potkonjak, M. Watermarking deep neural networks for embedded systems. In Bahar, I. (ed.), *Proceedings of the International Conference on Computer-Aided Design, ICCAD 2018, San Diego, CA, USA, November 05-08, 2018*, pp. 133. ACM, 2018. doi: 10.1145/3240765.3240862.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456. JMLR.org, 2015.

Kim, Y. Convolutional neural networks for sentence classification. In Moschitti, A., Pang, B., and Daelemans, W. (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1746–1751. ACL, 2014. doi: 10.3115/v1/d14-1181.

Krizhevsky, A. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Bartlett, P. L., Pereira, F. C. N., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pp. 1106–1114, 2012.

Li, F., Fergus, R., and Perona, P. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28 (4):594–611, 2006. doi: 10.1109/TPAMI.2006.79.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Li, X. and Roth, D. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*, 2002.

Li, Z., Hu, C., Zhang, Y., and Guo, S. How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of DNN. In Balenson, D. (ed.), *Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, December 09-13, 2019*, pp. 126–137. ACM, 2019. doi: 10.1145/3359789.3359801.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Merrer, E. L., Pérez, P., and Trédan, G. Adversarial frontier stitching for remote neural network watermarking. *Neural Comput. Appl.*, 32(13):9233–9244, 2020. doi: 10.1007/s00521-019-04434-z.

Namba, R. and Sakuma, J. Robust watermarking of neural network with exponential weighting. In Galbraith, S. D., Russello, G., Susilo, W., Gollmann, D., Kirda, E., and Liang, Z. (eds.), *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019*, pp. 228–240. ACM, 2019. doi: 10.1145/3321705.3329808.

Oh, S. J., Augustin, M., Fritz, M., and Schiele, B. Towards reverse-engineering black-box neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3,*

2018, Conference Track Proceedings. OpenReview.net, 2018.

Orekondy, T., Schiele, B., and Fritz, M. Knockoff nets: Stealing functionality of black-box models. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 4954–4963. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00509.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

Rivest, R. L., Shamir, A., and Adleman, L. M. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. doi: 10.1145/359340.359342.

Rosasco, L., Vito, E. D., Caponnetto, A., Piana, M., and Verri, A. Are loss functions all the same? *Neural Comput.*, 16(5):1063–107, 2004. doi: 10.1162/089976604773135104.

Rouhani, B. D., Chen, H., and Koushanfar, F. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In Bahar, I., Herlihy, M., Witchel, E., and Lebeck, A. R. (eds.), *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019*, pp. 485–497. ACM, 2019. doi: 10.1145/3297858.3304051.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., and Backes, M. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pp. 3–18. IEEE Computer Society, 2017. doi: 10.1109/SP.2017.41.

Song, C., Ristenpart, T., and Shmatikov, V. Machine learning models that remember too much. In Thuraisingham, B. M., Evans, D., Malkin, T., and Xu, D. (eds.), *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pp. 587–601. ACM, 2017. doi: 10.1145/3133956.3134077.

Strubell, E., Ganesh, A., and McCallum, A. Energy and policy considerations for deep learning in NLP. In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 3645–3650. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1355.

Uchida, Y., Nagai, Y., Sakazawa, S., and Satoh, S. Embedding watermarks into deep neural networks. In Ionescu, B., Sebe, N., Feng, J., Larson, M. A., Lienhart, R., and Snoek, C. (eds.), *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ICMR 2017, Bucharest, Romania, June 6-9, 2017*, pp. 269–277. ACM, 2017. doi: 10.1145/3078971.3078974.

Wang, T. and Kerschbaum, F. Attacks on digital watermarks for deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pp. 2622–2626. IEEE, 2019. doi: 10.1109/ICASSP.2019.8682202.

Zhang, J., Gu, Z., Jang, J., Wu, H., Stoecklin, M. P., Huang, H., and Molloy, I. Protecting intellectual property of deep neural networks with watermarking. In Kim, J., Ahn, G., Kim, S., Kim, Y., López, J., and Kim, T. (eds.), *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, pp. 159–172. ACM, 2018. doi: 10.1145/3196494.3196550.