# ENCRYPTION RESISTANT DEEP NEURAL NETWORK WATERMARKING

*Guobiao Li, Sheng Li*, Zhenxing Qian, Xinpeng Zhang*

School of Computer Science, Fudan University, Shanghai, China

## ABSTRACT

Deep neural network (DNN) watermarking is one of the main techniques to protect the DNN. Although various DNN watermarking schemes have been proposed, none of them is able to resist the DNN encryption. In this paper, we propose an encryption resistent DNN watermarking scheme, which is able to resist the parameter shuffling based DNN encryption. Unlike the existing schemes which use the kernels separately for watermarking embedding, we propose to embed the watermark into the fused kernels to resist the parameter shuffling. We further propose a MappingNet to map the the fused kernels into a higher dimension to increase the watermarking capacity. The MappingNet and the DNN are jointly trained to conduct final watermark embedding. Experimental results indicate the effectiveness of our proposed scheme for resisting the DNN encryption.

***Index Terms*—** Deep neural network, watermarking, encryption

## 1. INTRODUCTION

In the past few years, deep neural networks (DNN) have been rapidly developing for different applications including computer vision [1,2], natural language processing [3] and speech recognition [4]. However, training a good DNN is not a trivial task and the performance of a DNN model depends on several factors, including 1) carefully designed network structure and training strategy; 2) massively amount of labeled high-quality data; and 3) powerful computing resources to conduct the training. It requires a lot of human and computational resources to build an effective DNN model. For the sake of the model owner's interest, it is necessary and important to protect the DNN models against unauthorized usages.

DNN watermarking is one of the main solutions for DNN model protection, which embeds the information of the model owner (i.e., the watermark) into the DNN. Existing methods can be broadly divided into two categories, including the white-box watermarking [5–7] and black-box watermarking [8–10]. The former trains the DNN by incorporating additional regularization terms for watermark embedding. During the ownership verification, one needs to access the parameters of the DNN model for watermark extraction. The latter conducts the watermark embedding by using a trigger set to fine tune the DNN. It makes sure that, when the trigger set is used as input, the output of the DNN represents the watermark.

Despite the advantage, the DNN watermarking works only after the behavior of unauthorized usage or re-distribution occurs and the DNN is exposed to the public. It can not prevent the unauthorized usages after the the model is released or leaked. Sometimes the attacker may steal the model and use it without being noticed. The work in [11] shows that both the architecture and parameters of the DNN models might be extracted from the hardware device by side-channel attacks. To deal with such issues, a few attempts have been made in literature for DNN encryption, including parameter encryption and [12, 13] and parameter shuffling [14]. Attackers without the correct key have no permission to use the encrypted DNN models.

Unfortunately, the existing DNN watermarking algorithms are fragile to the aforementioned DNN encryption process. Once the watermarked models are encrypted, the watermarks are destroyed and not extractable. In this paper, we propose a new white-box watermarking scheme which is able to resist the parameter shuffling based DNN encryption. Unlike the existing schemes which use the kernels separately for watermarking embedding, we propose to embed the watermark into the fused kernels to resist the parameter shuffling. Such a strategy scarifies the capacity of watermark embedding. For remedy, we further propose a capacity expansion mechanism by incorporating a MappingNet to map the fused kernels into a higher dimension to host the watermark. Experimental results indicate that our proposed scheme not only performs perfectly for resisting the parameter shuffling, but also works well against other popular attacks including fine tuning and model compression.

## 2. RELATED WORKS

### 2.1. DNN Watermarking

The initial concept of DNN watermarking is proposed by Uchida *et al*. [5]. In this work, the authors propose to embed the watermarks into the DNN parameters by incorporating a regularization term in the network loss for watermark embedding. The watermark embedding is then carried out by

*Corresponding author: lisheng@fudan.edu.cn

This work was supported in part by the National Natural Science Foundation of China under Grants 62072114, U20A20178, U20B2051, U1936214, in part by the Project of Shanghai Science and Technology Commission 21010500200.

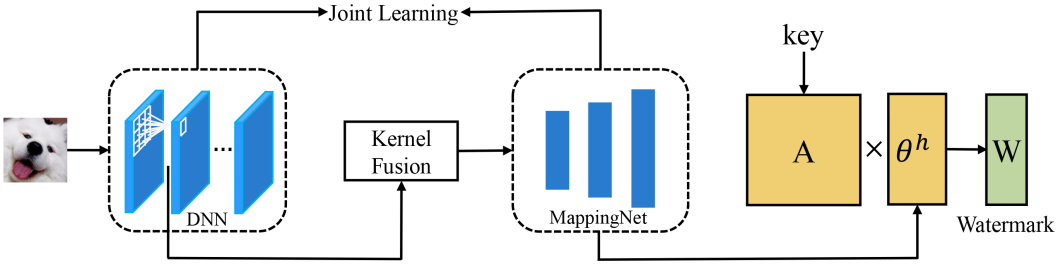



**Fig. 1**. Flowchart of the proposed watermark embedding scheme.

training the network with the new loss. Similarly, Rouhani *et al*. [7] propose to embed the watermark into the intermediate feature maps of the DNN. To resist ambiguity attacks, Fan *et al*. [6] propose to add a passport layer into the target model for ownership verification and user authentication. These schemes require the access of model parameters for watermarking extraction, which are termed as the white-box watermarking. The black-box watermarking, however, makes sure that the watermark is represented as the outputs of a trigger set [8]. The works in [9, 10] take advantage of the watermarked images or key samples whose distribution is the same as that of the original dataset as trigger set for watermarking.

### 2.2. DNN Encryption

There are two strategies to perform the DNN encryption: the parameter encryption [12, 13] and parameter shuffling [14]. Cai *et al*. [12] propose to add the adversarial perturbations into the model parameters to achieve parameter encryption. Tian *et al*. [13] propose Distribution Preserving Random Mask to encrypt the important DNN parameters. For parameter shuffling, Lin *et al*. [14] propose to shuffle the kernels in a certain layer based on the Chaotic Map theory [15]. Compared with parameter encryption method proposed in [12,13], parameter shuffling based method in [14] is much easier to implement with lower memory overhead, which is more appropriate for hardware with limited resources.

## 3. THE PROPOSED METHOD

### 3.1. Watermark Embedding

The flowchart of our proposed watermark embedding is illustrated in Fig. 1. First of all, we select a certain layer from the DNN and perform layer kernel fusion to produce a host parameter sequence which is robust to the parameter shuffling operation. Then, we propose a MappingNet to expand the host parameter sequence for enlarged watermark embedding capacity. The MappingNet and the DNN are jointly learnt with a newly designed loss function.

**Kernel fusion.** Let's denote the parameters of the selected layer as $\theta \in \mathbf{R}^{d\times c\times s\times s}$, where $d$ refers to the number of filters and each filter contains $c$ kernels (dubbed as $\mathbf{k}$) with $\mathbf{k} \in \mathbf{R}^{s\times s}$. We fuse the kernels by

$$\theta^f = \frac{1}{d\times c}\sum_{i=1}^{c}\sum_{j=1}^{d}\theta_{i,j}, \tag{1}$$

where $\theta_{i,j} \in \mathbf{R}^{s\times s}$ is the $i$-th kernel located in the $j$-th filter. Such a fusion strategy basically computes the mean among all the kernel available in the layer, which is robust to kernel-wise parameter shuffling operation proposed in [14]. If the parameters are also shuffled within the kernel, $\theta^f$ could be further fused in a similar way to produce a single value to host the watermark.

**MappingNet.** Our kernel fusion strategy shrinks the length of the sequence that can be used to host the watermark. To deal with this issue, we propose here a MappingNet to map the fused kernel into a higher dimension. The MappingNet is essentially a Multilayer Perceptron with $\theta^f$ as input and $\theta^h$ as output, i.e.,

$$\theta^h = f_M(\theta^f), \tag{2}$$

where $f_M(\cdot)$ represents the MappingNet. For simplicity, we treat $\theta^f$ and $\theta^h$ as two vectors with length of $s\times s$ and $m\times s\times s$, where $m$ is the expansion factor.

**Loss Function.** We design the following loss to jointly learn the MappingNet and the DNN, i.e.,

$$\mathcal{L}_t = \mathcal{L}_o + \alpha\mathcal{L}_w + \beta\mathcal{L}'_w + \gamma\mathcal{L}_{bs}, \tag{3}$$

where $\alpha$, $\beta$, $\gamma$ are hyper parameters to balance different loss terms, $\mathcal{L}_o$ is the original loss in the DNN, $\mathcal{L}_w$ and $\mathcal{L}'_w$ are watermark losses from neural networks with and without the watermark embedding, $\mathcal{L}_{bs}$ is a $L1$ regularization loss for $\theta^f$, i.e.,

$$\mathcal{L}_{bs} = |\theta^f|. \tag{4}$$

Next, we will elaborate the two watermark losses. Given a watermarked DNN and the corresponding watermark $W \in \{0,1\}^t$, the watermark loss is designed as

$$\mathcal{L}_w = \mathcal{L}_{CE}(\sigma(A\times f_M(\theta^f)), W), \tag{5}$$

where $\mathcal{L}_{CE}(\cdot)$ and $\sigma(\cdot)$ are the cross entropy cost function and the sigmoid function, $A \in \mathbf{R}^{t\times h}$ is an embedding matrix

**Table 1**. Accuracy of DNN before and after the watermarking.

| Model | Dataset | $ACC_c(\%)$ | $ACC_m(\%)$ |
|---|---|---|---|
| AlexNet | CIFAR10 | 91.66 | 91.54 |
| | CIFAR100 | 69.61 | 69.41 |
| | ImageNet | 81.85 | **81.90** |
| ResNet18 | CIFAR10 | 95.01 | 94.77 |
| | CIFAR100 | 76.45 | 76.36 |
| | ImageNet | 86.35 | 86.05 |

generated according to a key with $t \leqslant h$ and $h = m \times s \times s$. The design of $\mathcal{L}_w$ is try to make the extracted watermark to be the same as the embedded watermarks. It does not take the DNNs without watermarks into account. To address this issue, we also introduce a pair of negative sample $\{\theta_n^f, W_n\}$ to compute a watermark loss for the DNNs without watermarks, i.e.,

$$\mathcal{L}_w' = \mathcal{L}_{CE}(\sigma(A \times f_M(\theta_n^f)), W_n), \tag{6}$$

where $\theta_n^f$ is the fused kernel obtained from the original DNNs at the same layer and $W_n$ is a string with half of the bits different from $W$.

### 3.2. Watermark Extraction

Given a watermarked DNN, we extract the parameters of the layer with watermark embedding and conduct the same kernel fusion as what has been done in watermark embedding. Let's denote the fused kernel from a watermarked DNN as $\beta_f$, the watermark is extracted by

$$W^{'} = S(A \times f_M(\beta_f)), \tag{7}$$

where $S(\mathbf{x})$ binarizes the vector $\mathbf{x}$ by

$$s_k = \begin{cases} 1 & \mathbf{x}_k \geq 0 \\ 0 & else. \end{cases}, \tag{8}$$

where $s_k$ is the $k$-th binarized element of $\mathbf{x}$.

## 4. EXPERIMENTAL RESULTS

We adopt three benchmark datasets CIFAR10, CIFAR100 [16] and a subset of ImageNet [17] to train the DNN models. In particular, for each of the datasets, we train two well-known DNNs including AlexNet [2] and ResNet18 [1]. For the subset of ImageNet, we randomly select 20 categories from the original dataset and each category contains 1000 images, which are separated into 900 images for training and 100 images for testing. Unless stated otherwise, the length of the watermark is set as 128 bits (i.e., $t = 128$), which is embedded into second convolutional layer of the well-trained models. The expansion factor of the MappingNet $m$ is set as 64 for all the experiments.

**Table 2**. Robustness against model encryption. (Before / After Encryption).

| | Method | ACC(%) | BER(%) |
|---|---|---|---|
| black-box | Adi *et al*. [8] | 86.15/5.00 | 0/95 |
| | Zhang *et al*. [9] | 85.85/5.05 | 0/95 |
| | Li *et al*. [10] | 85.60/4.90 | 12/95 |
| white-box | Uchida *et al*. [5] | 86.20/4.90 | 0/52.34 |
| | DeepSigns [7] | 85.95/5.10 | 0/49.22 |
| | Passport [6] | 84.35/5.00 | 0.78/50.78 |
| | **Ours** | **86.05/4.95** | **0/0** |

### 4.1. Accuracy

The accuracy of the DNN before and after the watermarking is reported in Table 1, where $ACC_c$ and $ACC_m$ refer to the accuracy of the original DNN model and the watermarked DNN model, respectively. It can be seen that our proposed method has very little effect on network performance, with the degradation in accuracy lower than 0.30%. In some cases (e.g., AlexNet&ImageNet), we even observe a slight improvement in accuracy after the watermarking.

### 4.2. Robustness

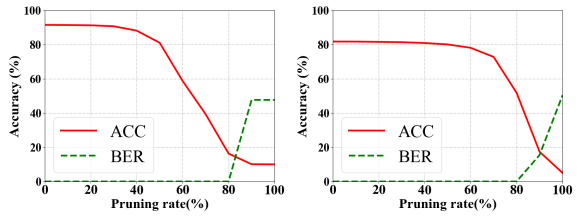
**Against model encryption.** We compare the robustness of our method and the existing DNN watermarking algorithms against the model encryption. We consider here the ResNet18 trained on ImageNet as the DNN model for watermarking. We use the parameter shuffling based DNN encryption scheme [14] to encrypt the model. Table 2 gives comparison results among different schemes against the model encryption, where ACC is the accuracy of the watermarked model and BER is the bit error rate before and after the encryption. Note that, for black-box watermarking, we report the BER as the error of recognizing the trigger sets. It can be seen that, after the encryption, the accuracy of all the DNN models are significantly reduced. All the existing scheme are not able to resist the encryption at all with BER around 50% for white-box watermarking and around 100% for black box watermarking. On the contrary, the BER of our proposed method is not affected at all after the model encryption.

**Against fine-tuning.** Fine-tuning is a common attack against model watermarking. We assume that attacker has 20% of the original training dataset and use it to retrain the watermarked model. Table 3 reports the BER of our watermarked model after the fine-tuning attack. We can see that the BER remains 0 after a significant amount of fine-tuning epochs.

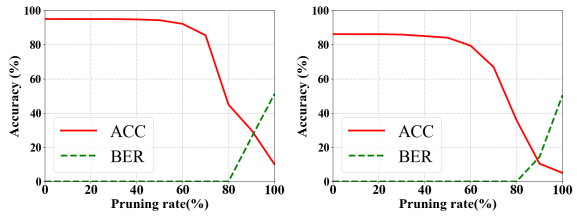
**Against model compression.** Next, we evaluate the robustness of our scheme against the model compression. We adopt DNN pruning scheme proposed in [18] for model compression. Fig. 2 shows the BER and model accuracy (ACC) at different pruning rate on different DNNs. It can be seen

**Table 3**. Robustness against fine-tuning.

| Model | AlexNet | | | | | | ResNet18 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | CIFAR10 | | CIFAR100 | | ImageNet | | CIFAR10 | | CIFAR100 | | ImageNet | |
| Number of epochs | 100 | 200 | 100 | 200 | 100 | 200 | 100 | 200 | 100 | 200 | 100 | 200 |
| ACC(%) | 91.00 | 91.05 | 67.30 | 67.05 | 81.20 | 80.95 | 94.30 | 94.15 | 76.20 | 76.35 | 82.50 | 82.00 |
| BER(%) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |




(a) AlexNet




(b) ResNet18

**Fig. 2**. Robustness against model pruning. Left: trained on CIFAR10, right: trained on ImageNet.

that the BER remains at 0 when the pruning rate is less than 80%. When the pruning rate is over 80%, the BER significantly increases, but the accuracy of the model will also be significantly reduced.

### 4.3. Capacity

In this section, we use the AlexNet trained on CIFAR10 to evaluate the capacity of the proposed scheme. We embed the watermark into different layers separately including the second, third and fourth convolutional layer of the AlexNet. For each layer, we gradually increase the length of the watermark (i.e., $t$) to conduct the watermarking until we are not able to perform a lossless extraction of the watermark. Table 4 shows the maximum watermarking capacity in different layers with and without the use of the MappingNet $f_M(\cdot)$. It can be seen that, by using the MappingNet, we are able to increase the capacity with over 25 times larger in different layers. This indicates that MappingNet is effective for capacity expansion.

### 4.4. Effectiveness of $\mathcal{L}'_w$

We introduce two watermark losses during the design of the total loss, i.e., $\mathcal{L}_w$ and $\mathcal{L}'_w$ to compute the watermark losses from watermarked and non-watermarked DNNs, respectively. In this section, we demonstrate the necessity of using $\mathcal{L}'_w$. In

**Table 4**. Watermarking capacity (bits) in different layers.

| Embedded layer | conv2 | conv3 | conv4 |
|---|---|---|---|
| $f_M(\cdot)(\checkmark)$ | 144 | 200 | 160 |
| $f_M(\cdot)(\times)$ | 4 | 4 | 6 |

**Table 5**. The BER of different DNN models with and without the $\mathcal{L}'_w$.

| Model | Dataset | BER(%) Non-watermarked | BER(%) Watermarked |
|---|---|---|---|
| AlexNet | CIFAR10 | 50/0 | 0/0 |
| | CIFAR100 | 50/0.78 | 0/0 |
| | ImageNet | 50/1.56 | 0/0 |
| ResNet18 | CIFAR10 | 50/1.56 | 0/0 |
| | CIFAR100 | 50/0 | 0/0 |
| | ImageNet | 50/3.91 | 0/0 |

particular, we embed the watermarks with and without the incorporation of $\mathcal{L}'_w$, and extract the watermarks from the watermarked and non-watermarked DNNs. Table 5 shows the BER under different cases. It can be seen that the incorporation of $\mathcal{L}'_w$ does not have any effect for the BER of the watermarked models. However, when it comes to the non-watermarked models, the BER is close to 0 without the $\mathcal{L}'_w$. In such a case, anyone who generates the embedding matrix would be able to extract the corresponding watermark from the DNN, which causes false alarms. This can be solved by the incorporation of $\mathcal{L}'_w$, where the BER of the non-watermarked DNN is 50%.

## 5. CONCLUSION

In this paper, we propose a DNN watermarking scheme which is able to resist the parameter shuffling attack. We first fuse all the kernels from the same layer to produce a host sequence which is robust to parameter shuffling. The fused kernel will then be mapped to a higher dimension by using our proposed MappingNet, which is for the sake of capacity expansion. The MappingNet and the DNN are eventually learnt together to achieve low watermark extraction error rate and high robustness. Experimental results demonstrate the advantage of our proposed scheme in resisting the DNN encryption over the existing schemes.

## 6. REFERENCES


[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[5] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, 2017, pp. 269–277.

[6] Lixin Fan, Kam Woh Ng, and Chee Seng Chan, "Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks," *Advances in neural information processing systems*, 2019.

[7] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar, "Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 485–497.

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

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

[10] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo, "How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of dnn," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 126–137.

[11] Weizhe Hua, Zhiru Zhang, and G Edward Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[12] Yi Cai, Xiaoming Chen, Lu Tian, Yu Wang, and Huazhong Yang, "Enabling secure in-memory neural network computing by sparse fast gradient encryption.," in *ICCAD*, 2019, pp. 1–8.

[13] Jinyu Tian, Jiantao Zhou, and Jia Duan, "Probabilistic selective encryption of convolutional neural networks for hierarchical services," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2205–2214.

[14] Ning Lin, Xiaoming Chen, Hang Lu, and Xiaowei Li, "Chaotic weights: A novel approach to protect intellectual property of deep neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 7, pp. 1327–1339, 2020.

[15] Gabriel Peterson, "Arnold's cat map," *Math Linear Algebra*, vol. 45, pp. 1–7, 1997.

[16] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," *Advances in neural information processing systems*, 2009.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[18] Song Han, Jeff Pool, John Tran, and William J Dally, "Learning both weights and connections for efficient neural networks," *arXiv preprint arXiv:1506.02626*, 2015.