K-008

A Parameters Sharing Watermarking Technique for Convolutional Neural Network Model

Han He† Seok Kang† Yuji Sakamoto†

1. Introduction

Recently, Convolutional Neural Network (CNN) has made significant progress in various fields, such as object recognition, image classification and so on. However, it becomes more and more expensive to train a high-performance CNN. For this reason, a high-performance CNN model can be considered as an Intellectual Property (IP). As the number of distributed pre-trained CNN models is increasing, it is essential to develop watermark techniques to prevent illegal copy and redistribution.

Some watermarking techniques in white-box setting [1-[4] and black-box setting [5][6] have been proposed. In white-box setting, the parameters are accessible to users, while in black-box setting, only the outputs are accessible to users. We target at CNN watermarking technique in white-box setting which is usually secure and robust.

The first watermarking technique in white-box setting for CNN models was proposed by Uchida et al. [1]. In their approach, the watermark is embedded into the weights of convolutional layer by a regularizer. This approach will not impair the performance of the original task. Based on this approach, we proposed a regularizer-based watermarking mapping technique [4] to enhance the embedding capacity and the invisibility of watermark. However, because of the nature of CNNs, all regularizer-based methods are vulnerable to attack.

In this paper, we proposed a watermarking technique based on parameters sharing for CNN models. By randomly selecting the weights of target convolutional layer instead of all of them and sharing these weights between the host network and an embedding network which is a single layer perceptron (SLP), our approach inherits advantage of previous work and can enhance the security and robustness corresponding to the requirement. The empirical results show satisfactory performance in terms of security, robustness and embedding capacity compared to the conventional CNN watermarking scheme.

2. Proposed Method

2.1 Embedding Target

In previous research [1][3][4], the target weights are the flattened and averaged weights of the target convolutional layers. To improve the security of watermark, the target weights in our approach are selected by the following procedure. The weight of target layer is a 4D tensor $W \in \mathbb{R}^{S\times S\times D\times L}$, where S, D and L are the kernel size, the number of input and output channels respectively. Subsequently, reshape into a 2D tensor $W' \in \mathbb{R}^{N\times L}$, where $N = S \times S \times D$ and k random numbers are generated between 0 and N-1 without duplicates. Let $A \in \mathbb{R}^k$ denote these random numbers. A will be used as index matrix and can be considered as a secret key. According to A, the embedding target $\omega \in \mathbb{R}^M$ is chosen from W', where $M = k \times L$.

† Graduate School of Information Science and Technology, Hokkaido University

2.2 Parameter Sharing

By parameter sharing, the original task and embedding task are done respectively in the host network and the embedding network. The flow chat is shown in Fig.1.

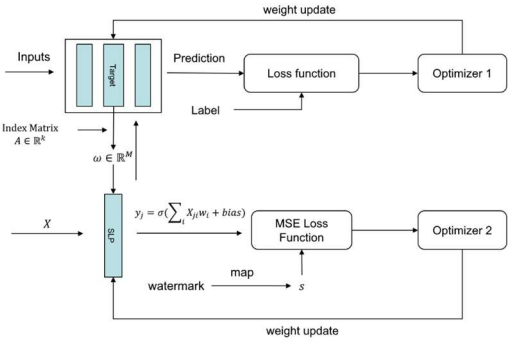

Fig. 1 Flow Chat of Proposed Method

Here, we assume the task of host network is classifying images. After finished a batch training of host network, the target weight will be extracted according to section 2.1 and shared to a SLP. The input X of SLP is generated by equation (1):

$$X = s[\mathcal{N}(0,1)] \tag{1}$$

Where X can also be considered as a secret key, $X \in \mathbb{R}^{T\times M}$. And $sign(\cdot)$ is sign function:

$$sign(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases} \tag{2}$$

The output of SLP is defined by equation (3)

$$y_j = \sigma\left(\sum_i X_{ji}w_i + bias\right) \tag{3}$$

The bias term here is need when extraction, and $\sigma(\cdot)$ is the sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{4}$$

And we employed MSE loss function in SLP:

$$E = \frac{1}{T}\sum_{j=1}^{T}(y_j - s_j)^2 \tag{5}$$

Here, s_j is the mapped value of the j-th n bit segment of watermark. s_j is computed by equation (6). Let S be the set of s_j

$$s_j = \frac{m_j}{2^{n-1}} \tag{6}$$

Where, m_j is the decimal value of j-th n bit segment of watermark. n is a positive integer variable that represents the number of watermark signal bits carried by one mapping value.

The watermark is extracted by equation (7):

$$m_j = bin\left(2^{n-1} \times \underset{s_k\in S}{argmin}(y_j - s_k)\right) \tag{7}$$

Here, $bin(\cdot)$ is a function which converts decimal value to binary string.

3. Experiments

3.1 Experiment Setting

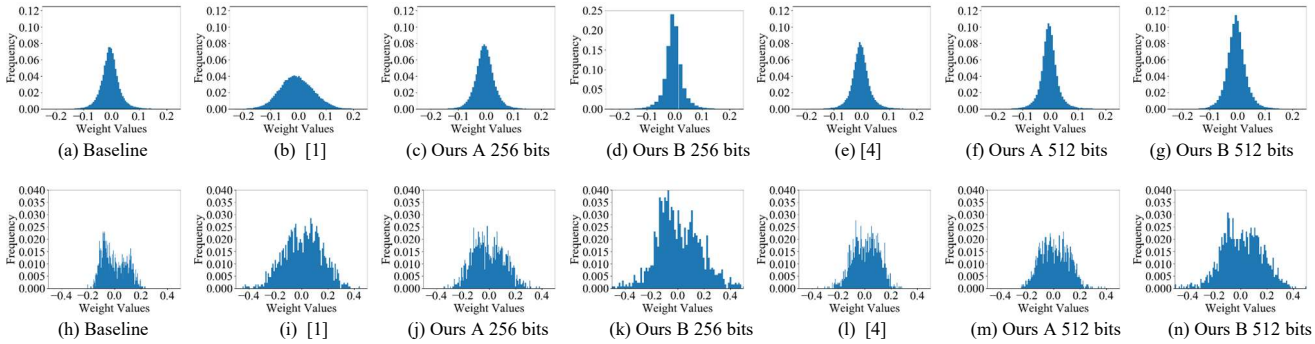

Fig. 2 Histogram of weights in the target convolutional layer of VGG (a-g) and MobileNet V2 (h-m)

We tested our approach in a widely used dataset: CIFAR-10 dataset [7] which contains 60,000 32 × 32 color images in 10 classes. We trained and tested two host networks（VGG-11 and MobileNet V2) on CIFAR-10.

In host network, we employed SGD optimizer with Nesterov momentum and cross-entropy loss function. The initial learning rate was set at 0.01, weight decay to 0.0005, momentum to 0.9 and minibatch size to 256. we trained it for 100 epochs**.**

We embedded watermark into No.2 convolutional layer of VGG-11 which have 73,728 weights and No.11 convolutional layer of MobileNet V2 which have 1,296 weights. The dimension k of index matrix A is set to 6 in VGG-11 and 356 in MobileNet V2. We employed Adam optimizer in SLP.

3.2 Fidelity and Robustness

Table 1 Comparison performance

Model	Method	T	Baseline	Accuracy	BER
VGG-11	[1]	256	87.24%	86.12%	0%
	Ours A			87.75%	0.78%
	Ours B			86.25%	0%
	[4]	512		86.43%	35.74%
	Ours A			86.64%	35.94%
	Ours B			87.21%	27.15%
MobileNet V2	[1]	256	80.20%	81.17%	21.48%
	Ours A			80.41%	24.61%
	Ours B			79.58%	13.67%
	[4]	512		80.51%	45.31%
	Ours A			80.44%	45.12%
	Ours B			80.42%	40.04%

The results of fidelity and robustness experiment are summarized in Table 1. Here, T is the length of embedding watermark. Baseline is the test accuracy of watermark-free model in original task. The BER is Bit Error Rate of each fine-tuned model. The fine-tuned model is pre-trained with watermark and fine-tuned on the CIFAR-10 dataset without embedding for 100 epochs. **Ours A** corresponds to our approach with weight decay set to 0.01. **Ours B** corresponds to our approach with weight decay set to 0.0001. We can see that all these approaches don't degrade the test accuracy of the original task. The BER comparison results show that Ours A has almost the same robustness performance as [1] and [4]. And Ours B achieved a better robustness performance compared with others.

3.3 Security

To prevent the adversary from detecting the existence of watermark, the distribution of watermark-embedded model should not have any tangible changes. Fig.2 shows the histograms of weights in the target convolutional layer. We can see that the histograms of Ours A and [4] are more similar to baseline model compared to [1] and Ours B in the VGG-11 network. However, in the MobileNet V2 network, none of these methods can keep the distribution because of the small number of weights in target convolutional layer.

The results so far indicated that there is a trade-off between robustness and security in our approach. Thus, in a specific practice, it is essential to set a suitable weight decay value corresponding to the requirement.

4. Conclusions

In this paper, we proposed a watermarking technique based on parameters sharing for CNN models to enhance security and robustness of watermark. Our empirical results show satisfactory performance in terms of security, robustness and embedding capacity compared to the conventional CNN watermarking scheme. In the future, we will test robustness against overwriting attack and future research directions is to improve the security under the condition of a small weight number.

References

[1] Uchida Y, Nagai Y, Sakazawa S, et al., "Embedding watermarks into deep neural networks", Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, 269-277, (2017).
[2] Rouhani B, Chen H, Koushanfar F, "Deepsigns: an end-to-end watermarking framework for protecting the ownership of deep neural networks", The 24th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). ACM, (2019).
[3] Chen H, Rouhani B D, Fu C, et al., "Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models", Proceedings of the 2019 on International Conference on Multimedia Retrieval, (2019).
[4] Han He , Seok Kang , Yuji Sakamoto, "A High Embedding Capacity Watermarking Technique Based on Regularization for Convolutional Neural Network Model ", ITE technical report 44(6), 41-46, (2020).
[5] Adi Y, Baum C, Cisse M, et al., "Turning your weakness into a strength: Watermarking deep neural networks by backdooring", 27th {USENIX} Security Symposium ({USENIX} Security 18), (2018).
[6] Le Merrer, Erwan, Patrick Perez, Gilles Trédan, "Adversarial frontier stitching for remote neural network watermarking", Neural Computing and Applications, 1-12, (2019).
[7] A. Krizhevsky, "Learning multiple layers of features from tiny images", Tech Report, (2009).