

Deep Learning Model Protection using Negative Correlation-based Watermarking with Best Embedding Regions

Sayoko Kakikura*, Hyunho Kang**, Keiichi Iwamura*

**Department of Electrical Engineering, Tokyo University of Science, Japan*

***Department of Electronic Engineering, National Institute of Technology, Tokyo College, Japan*

kakikura_sayoko@sec.ee.kagu.tus.ac.jp, kang@tokyo-ct.ac.jp, iwamura@ee.kagu.tus.ac.jp

Abstract—Deep learning has been used in several fields, such as image classification and data analysis. Training a high-performance model is expensive; thus, its property value is high. Watermarking is a representative technology that provides intellectual property protection for models. In this study, we proposed white-box watermarking using a modified Barni's method (our previous study) for image watermarking. Our method is applicable to pre-trained models because the watermark is embedded in the parameters of the network without training. The proposed method embeds multiple watermarking into neural networks using different keys. We evaluated the method using ResNet-50 trained on CIFAR-10 datasets and confirmed that our watermarking method has high fidelity and robustness against model compression and retraining. The experimental results reveal that our proposed approach can embed up to 10 watermarks with less than 0.1% loss of accuracy. They also indicate the method can completely detect watermarks even after 90% of the parameters are pruned and then transfer learned with CIFAR-100.

Keywords—Copyright protection, deep learning model protection, digital watermarking

I. INTRODUCTION

DEEP learning has become an indispensable technology for a wide range of fields in recent years. Because developing a high-quality deep learning model requires significant amounts of resources, such as training data, machine resources, and time, the models have a considerable value as intellectual property that must be protected.

Watermarking is a widely used technology for protecting the copyright of digital content, such as image and audio data. Neural network watermarking was first proposed by Uchida et al. [1], and since then, several studies on it have been published. Uchida et al. specified the following requirements for neural network watermarking:

- Fidelity: The watermark does not degrade the performance of the host network task
- Robustness: The watermark is robust against model modifications, for example, model compression
- Efficiency: The watermark embedding, and extraction processes should execute at a high speed.

This study aims to fulfill the abovementioned three requirements and embed multiple watermarks without collisions. Our method applies our previous watermarking method [2], which extends Barni's method [3] into watermarking for convolutional neural networks (CNNs). The image watermarking developed in our previous study is robust to noise and can embed multiple watermarks independently. The ability to embed multiple watermarks facilitates easy addition of more watermarks as more people take ownership of the model. In addition, watermark embedding does not require a training process. Our method is applicable for pre-trained networks, and the model owner can easily edit watermarks. Our preliminary findings were reported in our conference paper [4]. Additional experiments were performed in this study to assess the robustness of retraining the watermarked pre-trained deep learning model.

II. RELATED RESEARCH

Watermarking techniques for neural networks are classified based on the accessible information at the time of watermark extraction: white-box and black-box paradigm. The white-box paradigm provides access to all parameters in a host network, thereby allowing information to be embedded into the parameters [1], [5]. Because the black-box paradigm can only access the output of the final layer, some researchers have proposed training networks to produce incorrect output for a specific input deliberately and use it as a watermark [6]–[8].

The method proposed by Uchida et al. involves embedding a watermark into network parameters using a cost function with a parameter regularizer. The watermarks are embedded while training a host network.

Manuscript received July 8, 2022. This work is a follow-up of the invited journal to the accepted & presented paper entitled "Collusion Resistant Watermarking for Deep Learning Models Protection" of the 24th International Conference on Advanced Communication Technology (ICACT2022).

Sayoko Kakikura is with the Department of Electrical Engineering, Tokyo University of Science, Katsushika, Tokyo 125-8585 Japan (corresponding author, e-mail: kakikura_sayoko@sec.ee.kagu.tus.ac.jp).

Hyunho Kang is with the Department of Electronic Engineering, National Institute of Technology, Tokyo College, Hachioji, Tokyo 193-0942 Japan (e-mail: kang@tokyo-ct.ac.jp).

Keiichi Iwamura is with the Department of Electrical Engineering, Tokyo University of Science, Katsushika, Tokyo 125-8585 Japan (e-mail: iwamura@ee.kagu.tus.ac.jp).

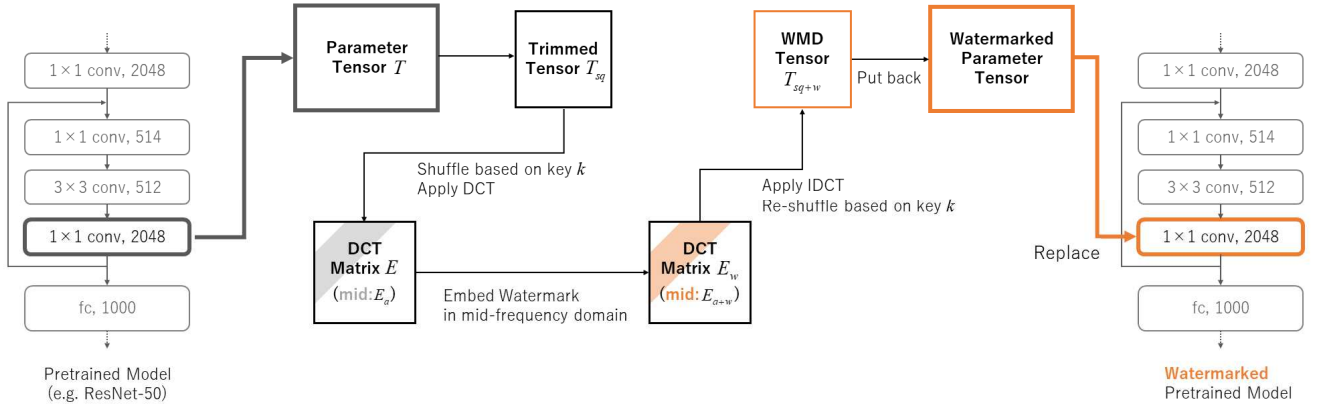


Fig. 1. Schematic diagram of embedding

Fan et al. [5] proposed a passport-based approach wherein the host network is appended with a passport layer. This passport layer additionally functions as a watermark. When a fake passport layer is appended instead of a valid passport layer, the performance of the host network degrades considerably.

III. PROPOSED METHOD

The applied embedding method for watermark embedding and detecting algorithm is based on our preliminary findings [2], which modifies Barni's method to achieve a higher correlation. The overall flow of the proposed method is depicted in Fig. 1.

A. Watermark Embedding

- 1) **Target Parameters Determination:** Select parameter tensor T to embed the watermark in the host network (e.g., weights / biases in the convolution layer).
- 2) **Parameter Pre-processing:** Flatten T and trim it to a length of N^2 . N^2 is observed to be less than 80% of the number of elements in T . The trimmed T is reshaped to $N \times N$, which is T_{sq} .
- 3) **Watermark Generation:** Let an arbitrary positive integer k be the key. The elements of T_{sq} are shuffled based on k , and then applied the discrete cosine transform (DCT). The generated DCT coefficient matrix is denoted as E , and the mid-frequency domain of E is denoted as E_a . E_a comprises M coefficients. A watermark is embedded using the following equation:

$$W = E_a - P_{mark} \quad (1)$$

where P_{mark} represents a sequence of uniformly distributed pseudo-random numbers in the range of the minimum and maximum values of E_a , with k as the seed value. E_{a+w} is defined using the following equation:

$$E_{a+w} = E_a + \alpha \cdot W \quad (2)$$

where α represents the embedding strength. To embed two or more watermarks, this step is repeated while changing k .

- 4) **Embedding:** Replace the mid-frequency domain of E with E_{a+w} , perform inverse DCT, and reorder by k (denoted as T_{sq+w}). Finally, put T_{sq+w} back into T and replace the host network parameters.

B. Watermark Detection

The mid-frequency domain of DCT coefficient matrix E'_{a+w} is extracted from the watermark embedded network using the same processes used for watermark embedding. P'_{mark} is calculated using x as the seed value ($x = 1, 2, \dots, n > k$) and the minimum and maximum values of E'_{a+w} , which is a range of uniformly distributed pseudo-random numbers.

The following equation determines the correlation value between E'_{a+w} and P'_{mark} :

$$corr = \frac{1}{M} \sum E'_{a+w} \cdot P'_{mark} \quad (3)$$

where M represents the length of E'_{a+w} and P'_{mark} , i.e., the size of the embedded area. If an attack on the watermark is not considered, then we obtain

$$E'_{a+w} = E_{a+w} = (1 + \alpha) E_a - P_{mark}, \quad P_{mark} \cong P'_{mark} \quad (4)$$

and there is a strong correlation between E'_{a+w} and P'_{mark} in the case of $x = k$. Therefore, k is detected by searching the x values that show a strong correlation. Even if an attack adds noise to E'_{a+w} , k can be detected if the correlation value when $x = k$ is sufficiently more significant than the correlation value when $x \neq k$.

IV. EVALUATION

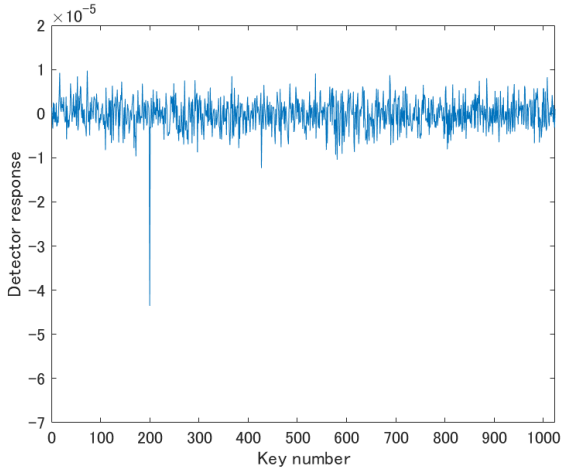
We evaluated our proposed method with ResNet-50 [9] pre-trained on ImageNet and then retrained on CIFAR-10 [10]. ResNet-50 is a CNN for classification tasks; we used the pre-trained version released by MathWorks as the base of transfer learning. The experimental settings were as follows:

Embedding Scheme Variables

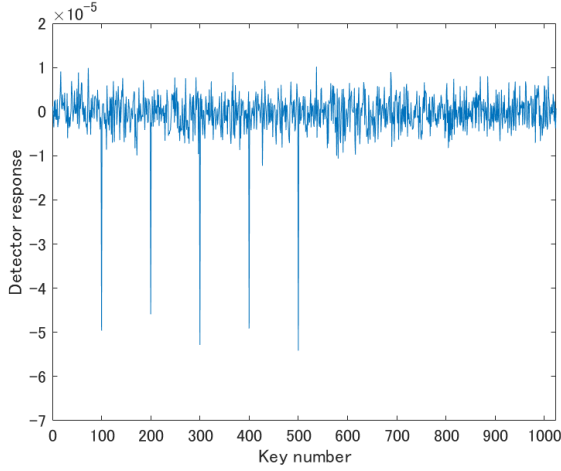
We set $N = 400$, $\alpha = 0.04$, $M = 24000$, and $n = 1024$. The key values were set to $k = 200$, $k = 100i$ ($i = 1, 2, \dots, 5$), and $k = 100i$ ($i = 1, 2, \dots, 9, 10$) for embedding 1, 5, and 10 watermarks, respectively.

Test Datasets

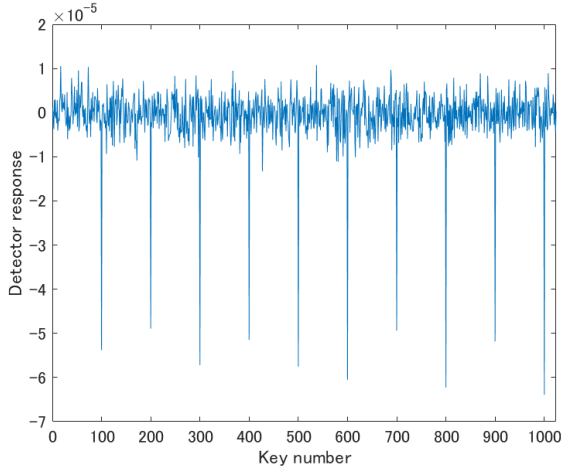
We prepared 3,000 images from CIFAR-10 for ResNet-50 as the evaluating networks.



(a) 1 watermark embedded



(b) 5 watermarks embedded



(c) 10 watermarks embedded

Fig. 2. Watermark detection on ResNet-50

A. Embedding Results

We evaluated the proposed method from two perspectives: (i) watermark detection accuracy, i.e., whether the embedded watermark can be detected correctly, and (ii) classification accuracy of the host model, i.e., whether the embedding of the watermark affects the accuracy of the original task.

1) Multi-key Embedding

In this experiment, watermarks were embedded in the weights in the last convolution layer. The layer is included in convolutional block conv5_3 (named in [9]). The size of the

TABLE I
PERFORMANCES BEFORE-AFTER WATERMARK EMBEDDING

	Num. of watermarks	Accuracy
Not embedded	-	0.8456
Embedded	1	0.8456
	5	0.8466
	10	0.8453

TABLE II
PERFORMANCES OF MULTI-LAYER EMBEDDING

Embedded Layer-block									Accuracy
Conv4 x						Conv5 x			
1	2	3	4	5	6	1	2	3	
									0.8456
					✓				0.8456
✓	✓	✓	✓	✓	✓				0.8456
								✓	0.8453
						✓	✓	✓	0.8456
✓	✓	✓	✓	✓	✓	✓	✓	✓	0.8453

weights is $1 \times 1 \times 512 \times 2048$.

The results of the correlation detection are presented in Fig. 2. The horizontal axis represents the key number, and the vertical axis represents the correlation value calculated using (3). The correlation values between the P'_{mark} generated for each key value and the embedding region E'_{a+w} are plotted. The detection result is the key value having a stronger correlation than that of the others. As shown in Fig. 2(a), only the correct key ($k = 200$) shows a strong correlation. Even when multiple watermarks were embedded, the key number corresponding to each watermark was detected independently (Fig. 2 (b), (c)).

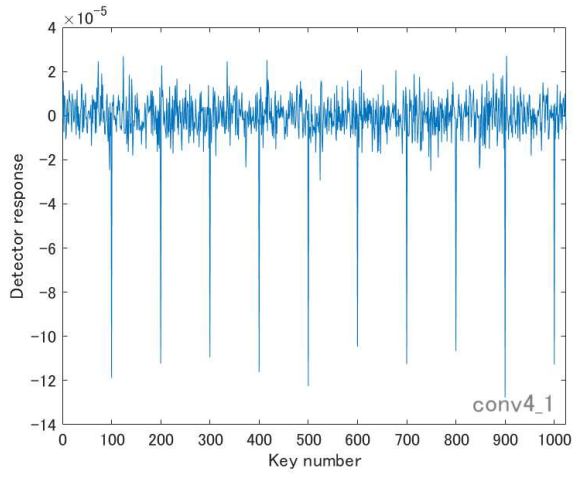
Next, we verified the classification accuracy of the model with an embedded watermark. As shown in Table I, there is no loss of accuracy after watermark embedding. The number of watermarks embedded in the weights has little effect on accuracy. These results confirmed that our watermark has little effect on the classification task.

2) Multi-layer Embedding

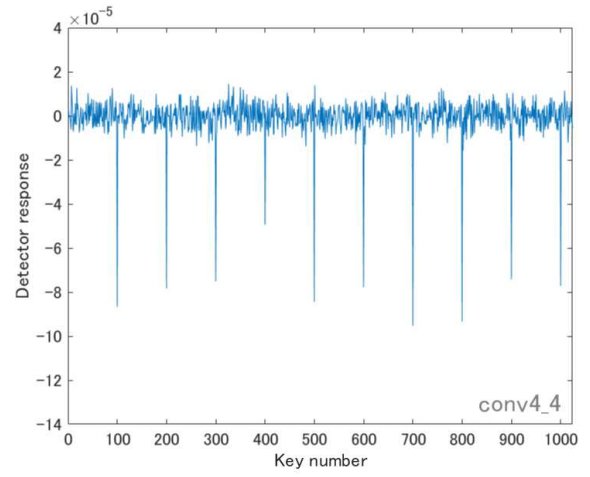
In this experiment, we set the target parameters as the weights in the last convolution layer of convolutional blocks conv4_x and conv5_x. The sizes of the weights in conv4_x and conv5_x are $1 \times 1 \times 256 \times 1024$ and $1 \times 1 \times 512 \times 2048$, respectively. We embedded 10 watermarks in each target layer.

Figs. 3 and 4 show the detection results of each embedded layer. It can be noted that the watermarks are detected perfectly in any layer of embedding. Although some correlation values appear small depending on the embedding layer and the key value (e.g., Fig. 3 (d), $k = 400$), they are sufficiently large compared with the correlation values calculated using incorrect keys. Comparing Figs. 3 and 4 reveals that a stronger correlation is detected in the case of embedding in conv4_x than in the case of embedding in conv5_x. This is because conv4_x's weights are 25% of the size of conv5_x's weights, and the watermark embedded area accounts for a higher share of the total area.

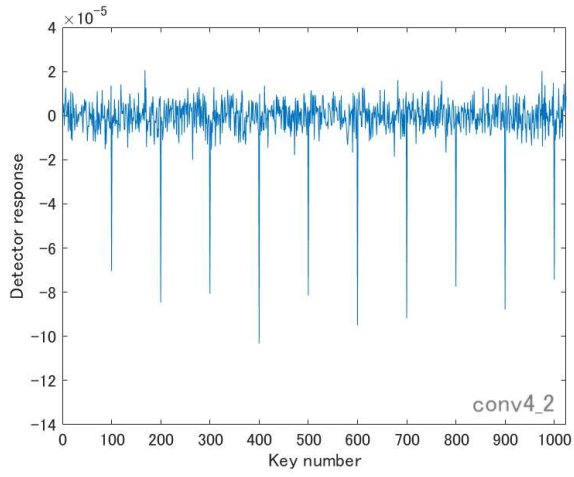
Table II lists the classification accuracy of the multi-layer embedded model. The classification accuracies were maintained at the same level as in the non-embedded model, despite embedding in multiple layers.



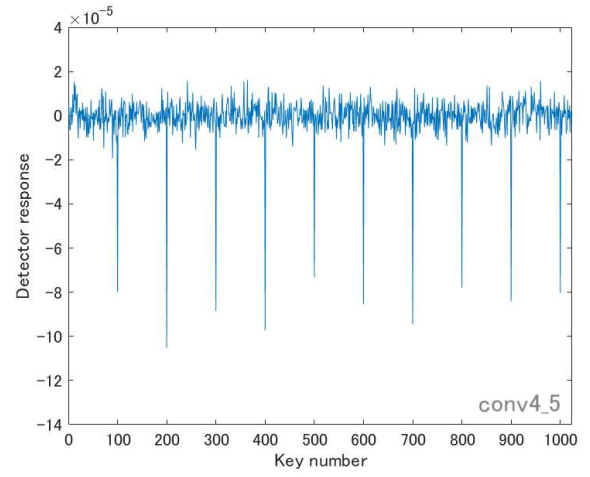
(a) conv4_1



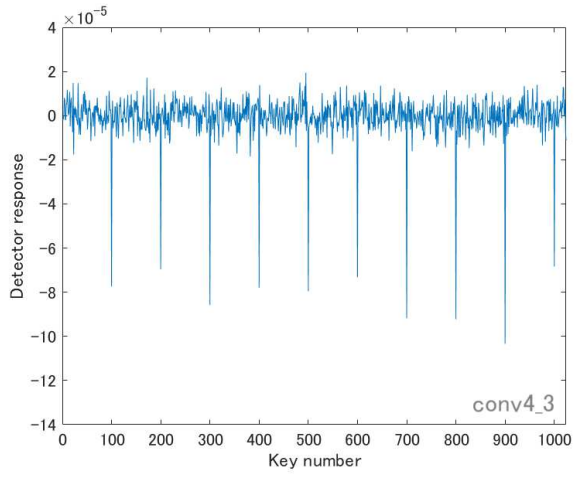
(d) conv4_4



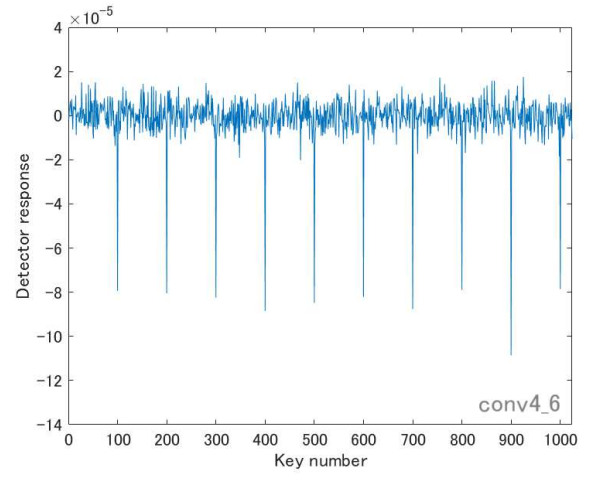
(b) conv4_2



(e) conv4_5

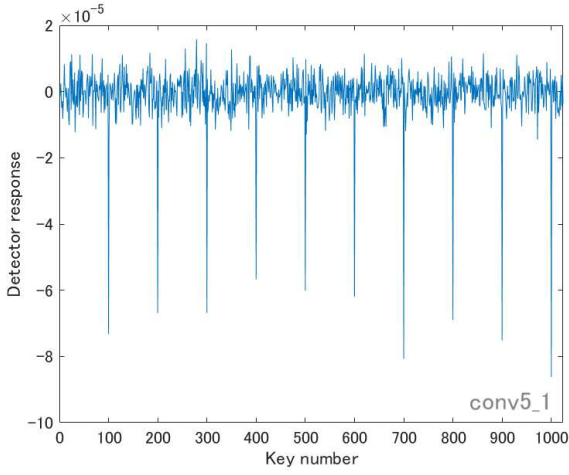


(c) conv4_3

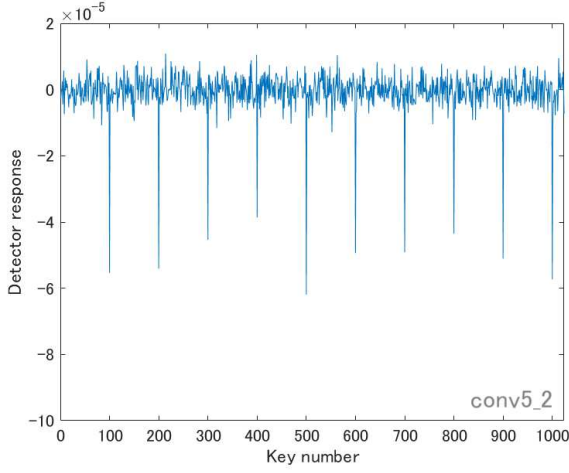


(f) conv4_6

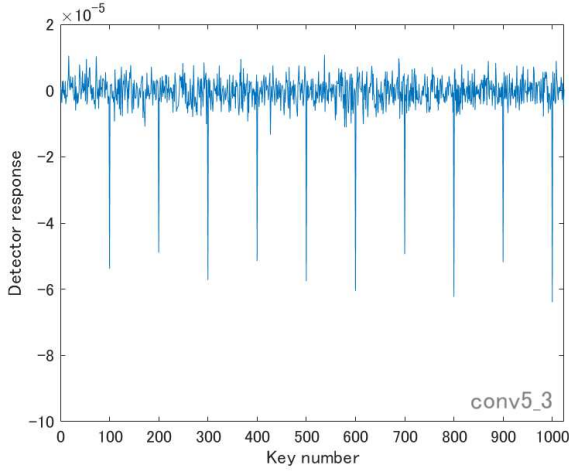
Fig. 3. Watermark detection embedded in convolutional block conv4_x



(a) conv5_1



(b) conv5_2



(c) conv5_3 (same as Fig. 2 (c) with adjusted vertical axis)

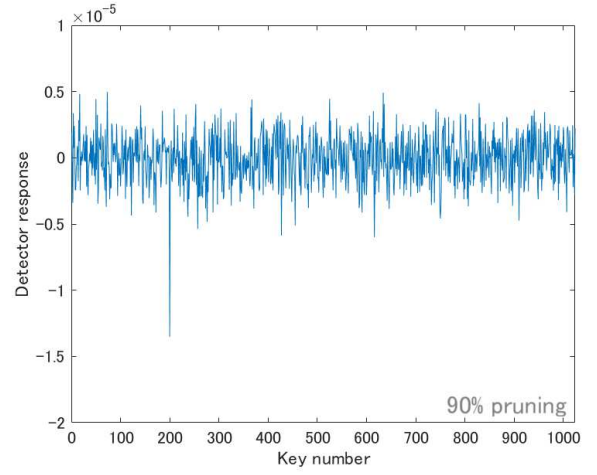
Fig. 4. Watermark detection embedded in the convolutional block conv5_x

B. Robustness against Attacks

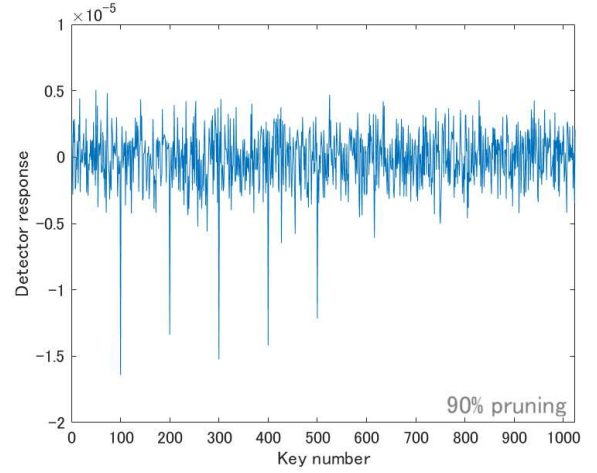
We tested the model with 10 watermarks embedded in the weights in the last convolution layer of conv5_3. We evaluated the model robustness by verifying the outputs of the correlation detector after attacking the model.

1) Weights Pruning

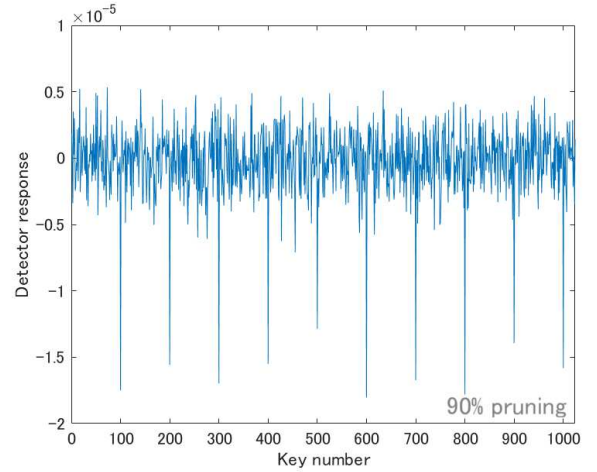
Model compression is one of the attacks against neural network watermarking because it changes the parameters of the network, and thus affects the embedded watermark. In this study, we evaluated the robustness of the watermark



(a) 1 watermark embedded



(b) 5 watermarks embedded



(c) 10 watermarks embedded

Fig. 5. Watermark detection on 90% pruning ResNet-50

against model compression, particularly against weight pruning. In pruning, $p\%$ of the watermarked weights with low absolute values were set to zero. Here, p (or p divided by 100) is termed as the pruning rate.

Fig. 5 illustrates the detector response from the network that prunes 90% of the watermark embedded weights. Although the correlation value at $k = 100i$ ($i = 1, 2, \dots, 10$) is reduced to approximately 20% of that at no attack (shown in Fig. 2(a)), embedded watermarks are detected individually even if almost all watermarked parameters are cut off.

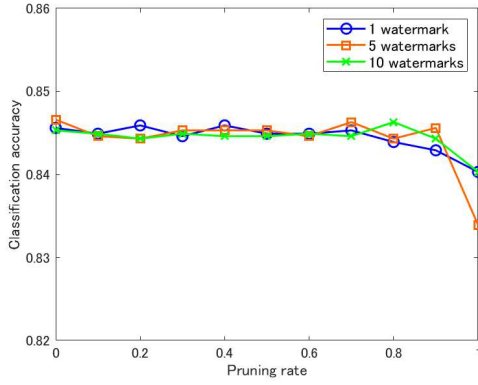
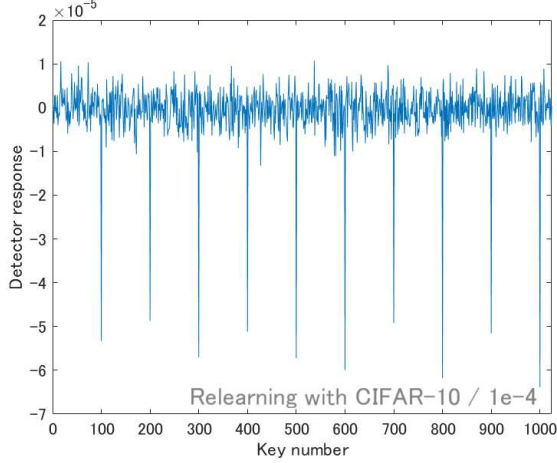
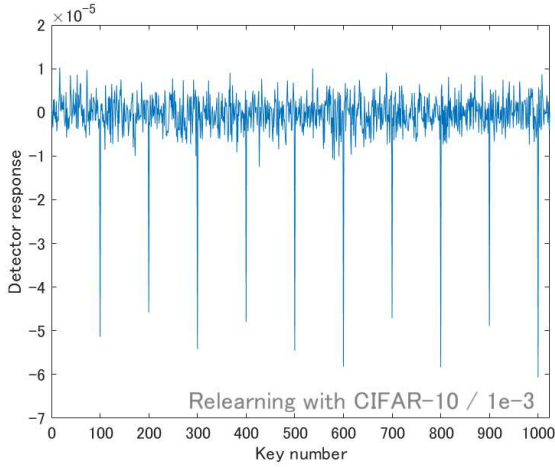


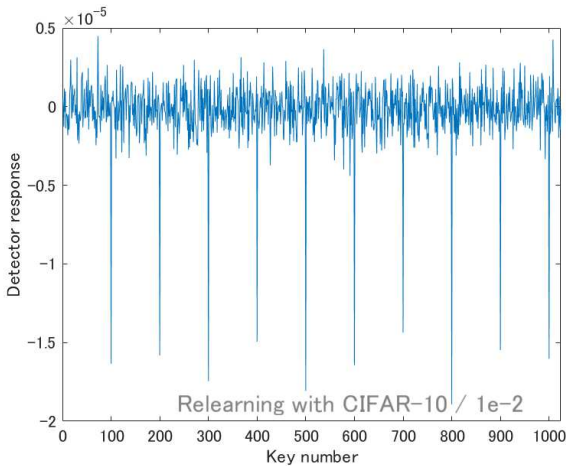
Fig. 6. Effect of pruning attacks on classification accuracy



(a) learning rate = 0.0001



(b) learning rate = 0.001



(c) learning rate = 0.01

Fig. 7. Watermark detection after relearning with CIFAR-10

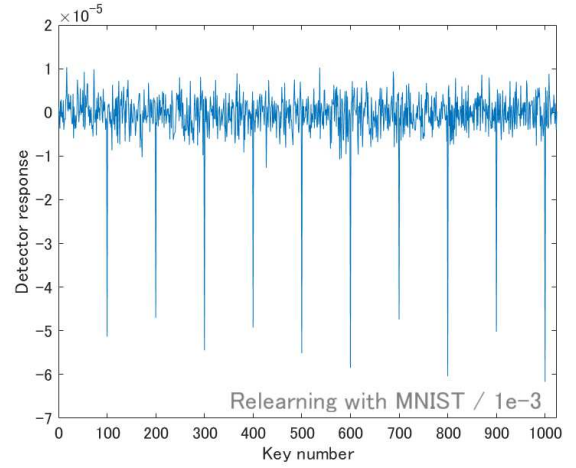


Fig. 8. Watermark detection after relearning with MNIST

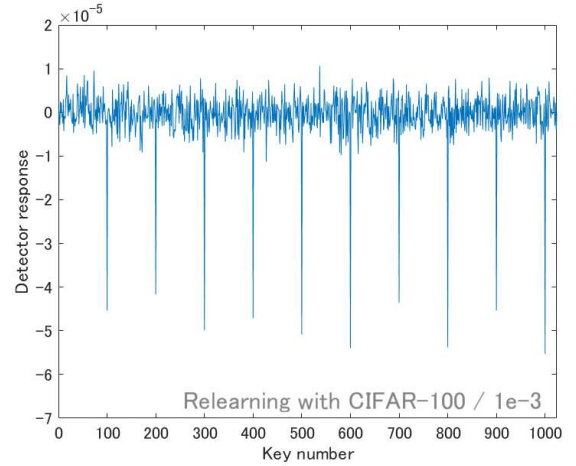


Fig. 9. Watermark detection after relearning with CIFAR-100

Fig. 6 shows the classification accuracy for different pruning rates. As seen in Fig. 6, the network maintains nearly the same accuracy from no pruning to 90% pruning. Moreover, it shows a decrease in classification accuracy when pruning 100% weights. This means, if the weights in that layer are 100%-pruning, the embedded watermark will be removed entirely while maintaining the classification accuracy of the model. This problem can be solved by embedding the watermark in multiple layers, as shown in the previous section.

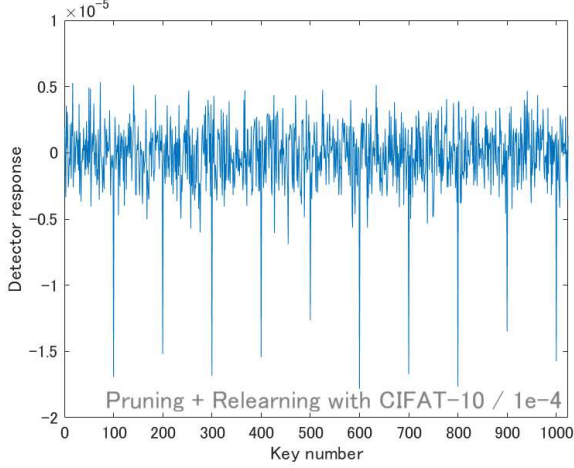
2) Retraining

Model retraining is also considered an attack for the same reason as that for pruning. In this study, we used several image datasets for retraining. The training parameters are listed in Table III.

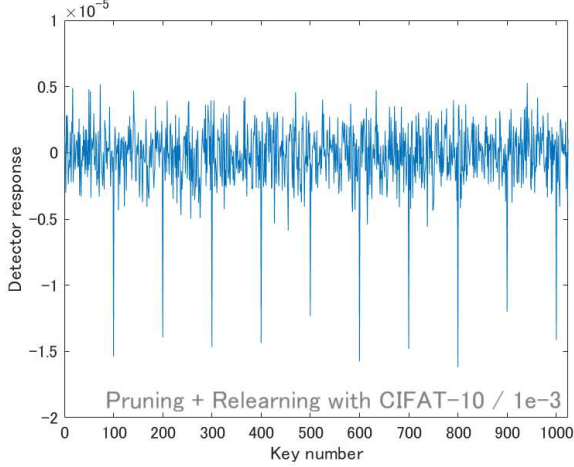
First, we retrained with the same dataset as the original task of the host network, i.e., CIFAR-10 by changing the learning rate to 0.0001, 0.001, and 0.01. CIFAR-10 has 10 classes and includes 60,000 images; we used 50,000 images for retraining.

Fig. 7 depicts the detection results after retraining with each learning rate. The correlations are correctly detected, but the higher the learning rate, the weaker the detected correlation value. It is somewhat difficult to detect the output result shown in Fig. 7(c), with a learning rate of 0.01. However, the learning rate is usually around 0.0001 while relearning; therefore, we consider that robustness against attacks is

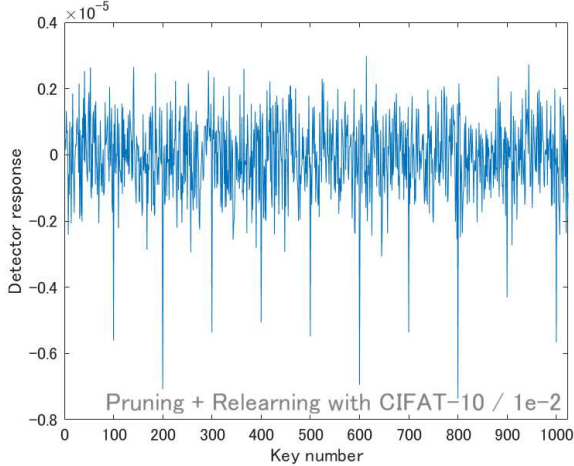
sufficient.



(a) learning rate = 0.0001



(b) learning rate = 0.001



(c) learning rate = 0.01

Fig. 10. Watermark detection after 90% pruning and relearning with CIFAR-10

Second, we retrained with two different datasets. One is the MNIST training set [11], and the other is CIFAR-100 [10]. The MNIST training set includes 60,000 images with 10 classes. In this experiment, 20,000 of those MNIST images were used for retraining. CIFAR-100 dataset comprises 100 classes of images. We used 50,000 images contained in its training set and replaced the final layers (a fully connected layer and a classification layer) of ResNet-50 with new layers adapted to the 100-classes classification.

This retraining method, when retraining with any different dataset than the original task, is so-called transfer learning.

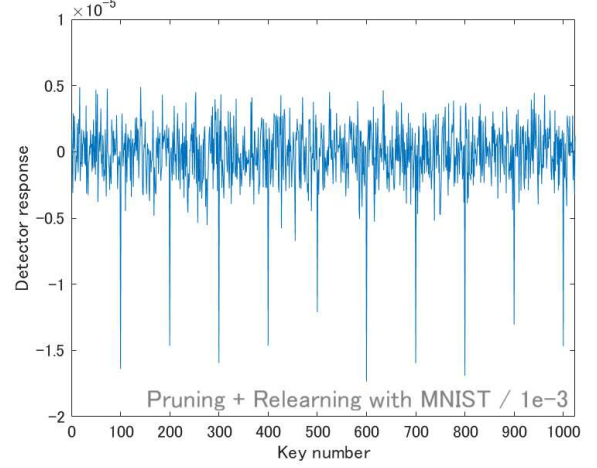


Fig. 11. Watermark detection after 90% pruning and relearning with MNIST

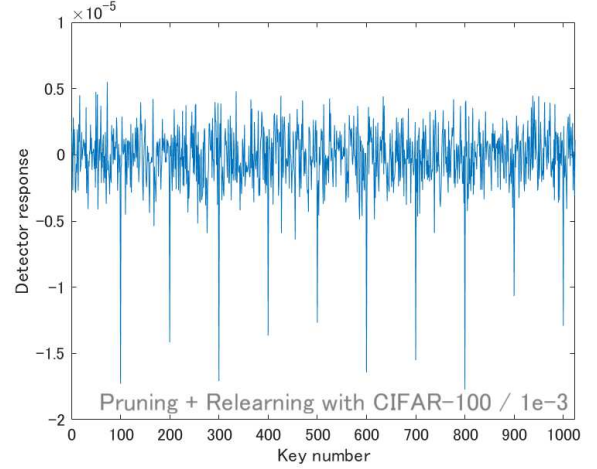


Fig. 12. Watermark detection after 90% pruning and relearning with CIFAR-100

TABLE III
NETWORK TRAINING PARAMETERS

Parameter	Value
Mini Batch Size	16
Max Epochs	30
Optimizer	Stochastic gradient descent with momentum
Momentum	0.9
L2 Regularization	0.0001
Gradient Threshold Method	L2 norm

Transfer learning is a more robust attack because it changes the parameters more significantly than normal relearning. In addition, the attack could be more vigorous if the number of classes in the new dataset is different from that of the original task.

The results of watermark detection are depicted in Figs. 8 and 9. As observed in these figures, both results show precise detection. These results indicate that our watermark method provides high robustness against retraining.

3) Pruning + Retraining

We retrained the 90% weight-pruned network and calculated the correlation values. The retraining settings are the same as described above.

Figs. 10–12 present the results of correlation detection with

90% pruning and relearning. It may be noted that every watermark was detected accurately after applying the pruning and transfer learning attacks.

V. DISCUSSION

We will now discuss the underlying reasons for the superior performance of our proposed method.

One of the reasons why embedded watermarks did not degrade the original task accuracy of the host network is that the embedding strength of such watermarks is low. In this study, we showed that watermarks are accurately detected even when the embedding strength is low, and that the detection accuracy is maintained even in the case of pruning and relearning attacks. These results are attributed to the use of correlation values for watermark detection to provide flexibility in the detection phase, and the dispersion of effects from the embedding watermarks via the transformation of the target parameter matrix into a frequency domain. Not only the adjustment of the value of embedding strength α but also “the adjustment of the value range of P_{mark} (the generated random number matrix) to that of the area to be embedded” and “the small ratio of the embedded area to the entire target layer” contribute to the prevention of excessive embedding.

Another conceivable reason is the high noise tolerance of the neural network. It is known that adding noise of appropriate strength to the weights in a neural network during training can prevent over-learning. This means that a slight change in weights will not significantly affect the performance of the model itself. The watermark P_{mark} generated using our proposed method is a sequence of uniformly distributed pseudo-random numbers, which play the same role as noise in the host network. Based on the aforementioned reasons, we concluded that our watermarking method does not degrade the task accuracy.

VI. CONCLUSION

This study proposed a watermark embedding method for CNNs based on the image watermarking method. Experimental results reveal that our method has minimal impact on image classification tasks and provides robustness to pruning and retraining attacks. Regardless of the number of watermarking embedded, our method can detect all watermarking completely.

In addition, the results show that watermarks can be embedded in various layers. In terms of embedding methods, it is possible to embed watermarks with smaller or larger parameters by adjusting the embedding scheme variables, such as N and M . These findings suggest that our method works satisfactorily for networks that are both, more simple or complex than ResNet-50.

REFERENCES

- [1] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, “Embedding watermarks into deep neural networks,” in *Proc. ICMR '17*, 2017, pp. 269-277.
- [2] H. Kang, and K. Iwamura, “Collusion-resistant watermarking using modified Barni method,” *Ubiquitous Information Technologies and Applications*, Lecture Notes in Electrical Engineering, Berlin, Germany: Springer, 2014, vol. 280, pp.627-634.

- [3] M. Barni, F. Bartolini, V. Cappellini, and A. Piva, “Robust watermarking of still images for copyright protection,” in *Proc. 13th ICDSIP*, 1997, pp. 499-502.
- [4] S. Kakikura, H. Kang, and K. Iwamura, “Collusion resistant watermarking for deep learning models protection,” in *Proc. ICACT'22*, 2022, pp. 40-43
- [5] L. Fan, K. W. Ng, and C. S. Chan, “Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks,” in *Proc. NIPS'19*, 2019, pp. 4714-4723.
- [6] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, “Protecting intellectual property of deep neural networks with watermarking,” in *Proc. ASLACCS'18*, 2018, pp. 159-172.
- [7] J. Guo, and M. Potkonjak, “Watermarking deep neural networks for embedded systems,” in *Proc. ICCAD'18*, 2018, pp. 1-8.
- [8] E. L. Merrer, P. Pérez, and G. Trédan, “Adversarial frontier stitching for remote neural network watermarking,” *Neural Computing and Applications*, vol. 32, Jul. 2020, pp. 9233-9244.
- [9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. CVPR'16*, 2016, pp. 770-778.
- [10] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Vol. 1. No. 4. Technical report, University of Toronto, 2009.
- [11] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, “Gradient based learning applied to document recognition,” in *Proc. IEEE*, 86(11), 1998, pp. 2278-2324.



Sayoko Kakikura was born in Japan 1998, received her B.S. degree in field of electrical engineering from Tokyo University of Science, Japan, in 2021. She is currently pursuing the M.S. degree with Tokyo University of Science, Japan. Her main research interests include watermarking and deep learning.



Hyunho Kang was born in Korea, 1973, received his Ph.D. from the University of Electro-Communications, Tokyo, in 2008. He is currently an Associate Professor in the Department of Electronic Engineering at National Institute of Technology, Tokyo College, Japan. His main interests are machine learning, deep learning, information security applications, multimedia security (steganography, digital watermarking), biometric security and physical unclonable functions.



Keiichi Iwamura was born in Japan, 1958, received B.S. and M.S. degrees in Information Engineering from Kyushu University in 1980 and 1982, respectively, and the Ph.D. degree from The University of Tokyo. He is currently a Professor with Tokyo University of Science. His research interests include coding theory, information security, and digital watermarking.