Robustness evaluation against fine-tuning in associative watermarking method for CNN

Keiichi Mori† and Masaki Kawamura†
† Yamaguchi University, Yamaguchi, Japan
E-mail: m.kawamura@m.ieice.org Tel: +81-83-933-5701

***Abstract*—We propose a zero-bit watermarking method to protect convolutional neural network (CNN) models. Deep learning models are valuable intellectual property, so protecting them is critical. Specifically, since CNNs are often reused through fine-tuning, it is essential to be able to robustly extract the watermark even after fine-tuning. However, conventional watermarking methods have faced challenges, such as potentially degrading the model's performance due to watermark embedding and being vulnerable to fine-tuning. In contrast, our proposed method is based on the Associative Watermarking Method (AWM). As a zero-bit watermarking approach, our AWM-based method associates a watermark with the CNN model without degrading its performance. Additionally, the error correction capability of AWM allows it to handle feature degradation. An important aspect of our method is selecting features extracted from the CNN model. We use the weights of the convolutional layers as features and convert them to binary before inputting them into the AWM. We confirmed that our method enables error-free extraction of watermarks from eight different CNN models under non-attack conditions. To demonstrate the robustness of our method, we evaluated its performance against fine-tuning. Our computer simulations showed that complete watermark extraction was possible even when models pre-trained on ImageNet were fine-tuned using the CIFAR-100 dataset. These results show that our method maintains its protective capabilities even when significant modifications are made, such as fine-tuning.**

I. Introduction

Deep neural networks (DNNs) are multi-layered neural networks that learn patterns from large amounts of data in order to solve complex problems, such as image classification and recognition [1]. In recent years, deep learning has achieved remarkable accuracy improvements across various tasks by leveraging its data utilization capabilities. For example, ResNet-152 achieved an accuracy of 78.57% in the 1000-class image classification task [2]. However, as models and data increase in scale, enormous costs are incurred in terms of computational resources, power consumption, and data preparation. This makes trained deep learning models valuable intellectual property for companies and research institutions. Nevertheless, once released, these models can be easily copied and redistributed, necessitating protection against unauthorized use. Against this backdrop, this paper focuses on digital watermarking methods for protecting deep learning models.

Digital watermarking is a technique that embeds an imperceptible watermark into digital content, such as images, videos, and audio. There are three types of digital watermarking methods. Multi-bit watermarking embeds specific messages, such as user information [3]. One-bit watermarking embeds only binary information of 0 or 1 [4], [5]. Both methods embed information by modifying the content itself. Conversely, zero-bit watermarking is a method that does not directly alter the content [6], [7]. In zero-watermarking methods [8]–[11], a secret key is generated based on features extracted from the content and the watermark. The watermark is then recovered using these features and the stored secret key. Since zero-bit watermarking does not modify the content, quality degradation caused by watermark embedding is avoided. Performance degradation due to watermarking is a critical issue in deep neural networks (DNNs). Zero-bit watermarking is a suitable protection method because it does not degrade the content.

Methods of digital watermarking for protecting DNNs are categorized as either black-box or white-box. Black-box methods [12], [13] extract watermark based on the relationship between the model's input and output and do not require access to its internal weights during extraction. Conversely, white-box methods [14] access the model's internal weights to extract the watermark. One example of a multi-bit watermarking method in the white-box category is the approach by Nagai *et al.* [14]. Their method embeds watermark using the weights of convolutional layers in Wide Residual Networks (WRNs) [15], which are derivative models of Convolutional Neural Networks (CNNs). Specifically, they use a feature vector obtained by averaging the convolutional layer weights along the output channel dimension. The model's weights are then trained such that the watermark can be extracted from the product of this feature vector and a secret key.

The two major attack methods against digital watermarking for DNN protection are fine-tuning and model compression. Fine-tuning retrains a pre-trained model using a different dataset, thereby altering the existing weights. It is crucial to ensure the robustness of watermarking methods against fine-tuning. Model compression, on the other hand, encompasses techniques such as pruning, quantization, and distillation. These methods aim to reduce the model size by modifying the weight information [14].

The associative watermarking method (AWM) [10], [11] is a zero-bit watermarking method composed of two types of associative memory models. Pairs of content features and watermarks are stored in a hetero-associative memory model (HAM) [16]. An auto-associative memory model (AAM) [17], [18] corrects errors in the retrieved watermarks. Because the AWM can store multiple pairs, it can manage multiple

979-8-3315-7206-8/25/$31.00

contents.

This paper focuses on CNN models with convolutional layers and proposes an associative watermarking method for protecting them. Specifically, similar to the method of Nagai *et al.* [14], continuous-valued features are extracted from the weights of convolutional layers and then binarized to be input into the AWM. Pairs of binary features and watermarks are stored in the HAM. The AAM also stores the watermarks. Since the features are extracted from the CNN weights, this method is considered a white-box approach. By employing AWM, it becomes unnecessary to modify the weights of the CNN, thus preserving its original classification performance. Additionally, since AWM has error correction capabilities, it exhibits resistance to fine-tuning.

We evaluate the robustness of the proposed method against fine-tuning using a computer simulation to assess the performance of each model. Kornblith *et al.* [19] performed fine-tuning on several CNN models that were pre-trained on ImageNet [20] using various datasets. They then compared the models' performance in transfer learning. We fine-tuned the following models, all of which were pre-trained on ImageNet using the CIFAR-100 dataset [21]: AlexNet [22], VGG11 [23], ResNet50 [2], SqueezeNet1-0 [24], DenseNet121 [25], Inception-v3 [26], MobileNetV2 [27], and EfficientNet-B0 [28]. Then, we extracted the watermark from each fine-tuned model to verify the robustness of our proposed method.

The rest of this paper is organized as follows. Section II explains the related work, and section III we explain the proposed method. Section IV describes computer simulations conducted to evaluate the robustness of the proposed method against fine-tuning. Finally, Section V concludes the paper.

II. Related Work

This section reviews the feature extraction and the watermark extraction method proposed by Nagai *et al.* [14] as prior research on methods for protecting CNNs.

A. Method of Nagai et al.

In the method of Nagai *et al.* [14], a Wide Residual Network (WRN) is chosen as the host network. The WRN is an improved version of the ResNet and achieves superior performance by increasing the number of channels in each layer [15]. The WRN network structure is illustrated in Figure 1. The input layer of the WRN performs convolutional processing, batch normalization, and ReLU activation on the input image. In the residual blocks $1, 2$ and 3, these same processes are sequentially applied. Each residual block consists of $2d$ convolutional layers, where d is referred to as the depth parameter. The width parameter k adjusts the number of output channels in each layer. In Nagai *et al.*'s method, the WRN with depth $d = 1$ and width $k = 4$ is used. Table I shows the WRN's detailed network configuration. In the "Parameter Configuration" column, the kernel size ($S \times S$) and the number of channels (D) are represented as $[S \times S, D]$. Since each residual block consists of two convolutional layers, two rows in the table are occupied. The first convolutional layer in residual blocks 2 and 3 has a stride of 2 and padding of 1, resulting in an output size smaller than the input size. After the residual blocks, average pooling (AvgPooling) is applied, and the output is fed into a fully connected layer (FCL). Finally, the softmax function is applied to output the probability distribution for each class.

TABLE I: Network architecture of WRN

Layer	Output size	Parameter configuration
Input layer	32×32	$[3 \times 3, 16]$
Residual Block1	32×32	$\begin{bmatrix} 3 \times 3, 16 \times k \\ 3 \times 3, 16 \times k \end{bmatrix} \times d$
Residual Block2	16×16	$\begin{bmatrix} 3 \times 3, 32 \times k \\ 3 \times 3, 32 \times k \end{bmatrix} \times d$
Residual Block3	8×8	$\begin{bmatrix} 3 \times 3, 64 \times k \\ 3 \times 3, 64 \times k \end{bmatrix} \times d$
	1×1	AvgPooling, FCL, Softmax

B. Feature Extraction

The weights of the first convolutional layer in Residual Block 1 are selected as features of the model. Figure 2 illustrates the flow of the convolutional operation. The weights of this convolutional layer, which are used for feature extraction, are defined as $W \in \mathbb{R}^{S \times S \times D \times L}$, where $S \times S$ is the filter size, D is the number of input channels, and L is the number of output channels. A feature map is obtained by applying L convolutional filters to the input image. Then, features are constructed based on the average value of the weights, $\overline{W}$, in the L direction, which is the output channel direction. The average value of the weights, $\overline{W}$, is given by

$$\overline{W}_{ijk} = \frac{1}{L} \sum_{l=1}^{L} W_{ijkl}. \tag{1}$$

This averaged weight $\overline{W} \in \mathbb{R}^{S \times S \times D}$ is then converted into a one-dimensional vector, $\boldsymbol{w} \in \mathbb{R}^{K}$, where $K = S^2 D$, representing the feature.

C. Watermark Extractor

The feature $\boldsymbol{w}$ extracted from the WRN is input into the watermark extractor, which then extracts the watermark. The watermark extractor is a single-layer, fully connected neural network. Let $\boldsymbol{b}$ denote the watermark with a message length of N bits. The weights of the watermark extractor, $X \in \mathbb{R}^{N \times K}$, are predetermined and drawn at random from a standard normal distribution. The output $\boldsymbol{y}$ of the watermark extractor is given by

$$y_j = f\left(\sum_{i=1}^{K} X_{ji} w_i \right), \ j = 1, 2, \cdots, N, \tag{2}$$

where the activation function $f(\cdot)$ is the sigmoid function, which is defined as

$$f(x) = \frac{1}{1 + \exp(-x)}. \tag{3}$$

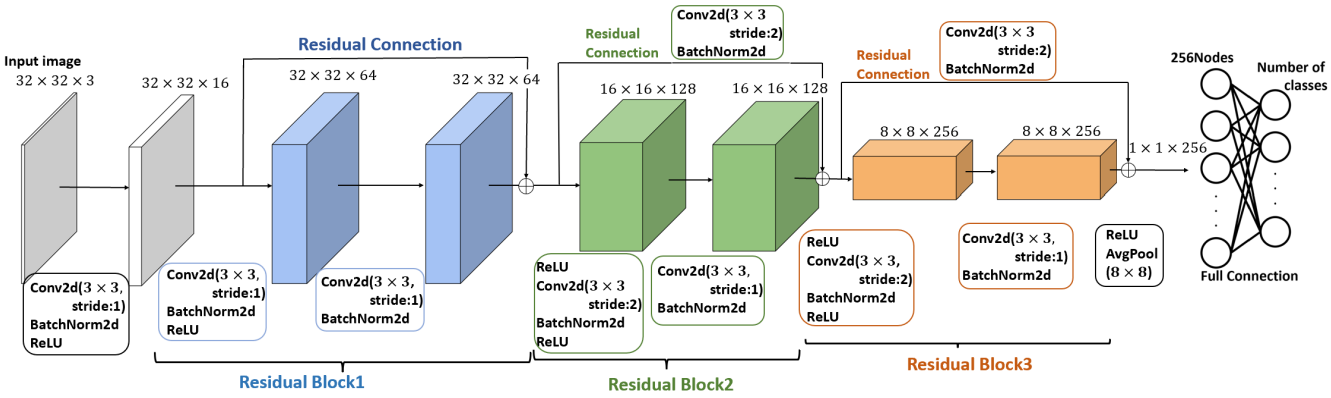


Fig. 1: Overall architecture of WRN with depth $d = 1$ and width $k = 4$. The network consists of an input layer, three residual blocks, and a fully connected layer.

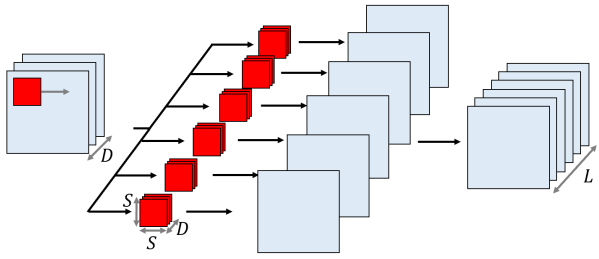


Fig. 2: Convolutional operation flow

The feature $\boldsymbol{w}$, or the weight $\overline{W}$ of the WRN, is trained so that the output $\boldsymbol{y}$ matches the watermark $\boldsymbol{b}$. Specifically, the loss function for this objective is expressed as

$$E_R(\boldsymbol{w}) = -\sum_{j=1}^{N} \{b_j \log(y_j) + (1 - b_j) \log(1 - y_j)\}. \quad (4)$$

If $E_o(\boldsymbol{w})$ is the loss function used for the original task, then the overall loss function $E(\boldsymbol{w})$ is given by

$$E(\boldsymbol{w}) = E_o(\boldsymbol{w}) + \lambda E_R(\boldsymbol{w}). \quad (5)$$

where λ is a parameter that adjusts the embedding strength. After training the extractor, watermark $\boldsymbol{b}$ is extracted using

$$b_j = H\left(\sum_{i=1}^{K} X_{ji} w_i\right), \ j = 1, 2, \cdots, N, \quad (6)$$

where $H(\cdot)$ represents the step function defined by

$$H(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}. \quad (7)$$

III. Proposed Method

A. Associative Watermarking Method

The associative watermarking method (AWM), as illustrated in Figure 3, comprises two main components: a hetero-associative memory model (HAM) and an auto-associative model (AAM) [10]. In this method, features are extracted from the content, and the mapping from these features to the watermarks is stored in the HAM. The AAM is a recurrent network that stores the autocorrelation of the watermarks. When extracting the watermark, features are extracted from the content and then input into the HAM. Finally, the AAM outputs the watermark with fewer errors.

The proposed method focuses on protecting CNN models using AWM. Since AWM requires binary features as memory patterns, the selection and binarization of these features must be considered. Transfer learning involves replacing the original output layer with a different network. Consequently, the output layer's weights cannot be used as features. Furthermore, in the case of fine-tuning, the weights of layers closer to the output layer are highly susceptible to significant changes during the learning process. Therefore, features are selected from the weights of layers closer to the input layer since they tend to be more stable. Specifically, a vector $\boldsymbol{w} \in \mathbb{R}^K$, which is a one-dimensional representation of the weights, is used as a feature. However, these features must be binarized to be input into the HAM. The binarized feature $\boldsymbol{\eta}$ is given by

$$\eta_k = \mathrm{sgn}(w_k - \theta), \ k = 1, 2, \cdots, K, \quad (8)$$

where the threshold θ is the median value of the elements, such that they take on the values $+1$ and -1 with equal probability.

The multiple relationships between CNN models and their respective watermarks can be stored using this method. Consider the application of this method to protect P models. Let η^μ $(\mu = 1, 2, 3, \cdots, P)$ denote the feature obtained from the μ-th model and let ξ^μ represent the corresponding watermark. The mapping from features to watermarks is stored in the HAM. The weight J^h_{ik} of the HAM is given by

$$J^h_{ik} = \frac{1}{N} \sum_{\mu=1}^{P} \xi^\mu_i \eta^\mu_k, \quad (9)$$

thorough correlation learning. Furthermore, the watermarks ξ^μ are stored in the AAM. The weight J^a_{ij} of the AAM is given

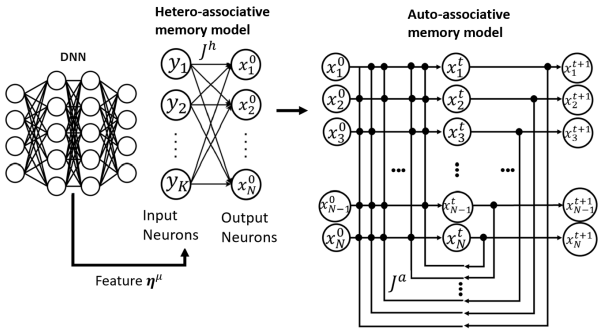


Fig. 3: Structure of AWM

by

$$J_{ij}^a = \frac{1}{N}\sum_{\mu=1}^{P}\xi_i^\mu\xi_j^\mu. \tag{10}$$

A challenge arises if a third party illicitly obtains a protected CNN model and applies our method with a different watermark. In this case, the legitimate rights holder would be unable to prove the ownership. One way to address this issue is to obtain a timestamp token (TST) from a Time Stamping Authority (TSA) [29] to prove when the trained associative memory model first came into existence. A TST enables verification of data tampering and provides proof that the data existed on a specific date and time. Typically, acquiring a TST for each CNN model would be impractical. However, with our proposed method, we can protect the authenticity of all associated trained CNN models by simply holding the TST for the AWM.

B. Watermark Extraction

The procedure for obtaining watermarks based on features extracted from CNN models is described below. The feature $\boldsymbol{\eta}^\mu$ is obtained from (8). When this feature is provided as input $\boldsymbol{y}$ to the HAM, the output $\boldsymbol{x}^0$ is given by

$$x_i^0 = \mathrm{sgn}\left(\sum_{k=1}^{K} J_{ik}^h y_k\right),\ i = 1, 2, \cdots, N. \tag{11}$$

When the output $\boldsymbol{x}^0$ of the HAM is given as the initial value for the AAM, the output x_i^{t+1} at time $t+1$ is given by

$$x_i^{t+1} = \mathrm{sgn}\left(\sum_{j\neq i}^{N} J_{ij}^a x_j^t\right),\ i = 1, 2, \cdots, N. \tag{12}$$

The watermark is obtained as output $\boldsymbol{x}^t$.

IV. Computer Simulation

The effectiveness of the proposed method will be evaluated using computer simulations. The performance of watermark extraction will be assessed quantitatively using the bit error rate (BER).

A. Experimental conditions

To evaluate the proposed method, we use the following CNN models: AlexNet [22], VGG11 [23], ResNet50 [2], SqueezeNet1-0 [24], DenseNet121 [25], Inception-v3 [26], MobileNetV2 [27], and EfficientNet-B0 [28]. Features are extracted from the convolutional layers of each CNN. The feature length is set to $K = 144$ bits. Since the convolutional layers of each CNN are different in shape and size, we select the convolutional layer closest to the input layer that can yield features of at least 144 bits for extraction. Although the lengths of the features obtained from different CNNs vary, all feature lengths are unified to 144 bits. Any elements exceeding this length are discarded and not used as features. The watermark is randomly generated with a length of $N = 500$ bits, ensuring that $+1$ and -1 bits are present in equal proportions (50% each).

To quantify the similarity between features extracted from the μ-th original model, $\boldsymbol{\eta}^\mu$, and features degraded by an attack (e.g., fine-tuning or noise addition), $\tilde{\boldsymbol{\eta}}^\mu$, we introduce the overlap m_*^μ. The overlap on feature is defined by

$$m_*^\mu = \frac{1}{K}\sum_{k=1}^{K}\eta_k^\mu\tilde{\eta}_k^\mu, \tag{13}$$

where K is the bit length of the extracted features from the models. This overlap shows how closely the attacked feature matches the original. Similarly, we define the overlap m_t^μ between the μ-th watermark $\boldsymbol{\xi}^\mu$ and the state $\boldsymbol{x}^t$ of the AAM at time t. The overlap on watermark is given by

$$m_t^\mu = \frac{1}{N}\sum_{i=1}^{N}\xi_i^\mu x_i^t, \tag{14}$$

where N represents the bit length of the watermarks. This overlap shows how closely the extracted watermark matches the original. The BER is a quantitative measure of error with respect to extracted features and watermarks. It is defined as a function of the overlap m by

$$\mathrm{BER}\,(m) = \frac{1-m}{2}, \tag{15}$$

In this study, the output of the AAM at time $t = 20$ is used as the extracted watermark for evaluation. Watermarks were generated for ten random samples, and the average BER was calculated.

First, we evaluated the BER of the extracted watermark using the proposed method under no-attack conditions. After extracting 144-bit features from each CNN and recovering the watermarks, the BERs between the outputs of the HAM and the original features were $\mathrm{BER}(m_*^\mu) = 0.0$. As expected, the AWM achieved perfect recall of the watermarks with $\mathrm{BER}\,(m_{20}^\mu) = 0.0$.

B. Evaluation Against Attacks

Since the AWM has error correction capability, we will evaluate the robustness of the proposed method against fine-tuning. Kornblith *et al.* [19] evaluated the effectiveness of

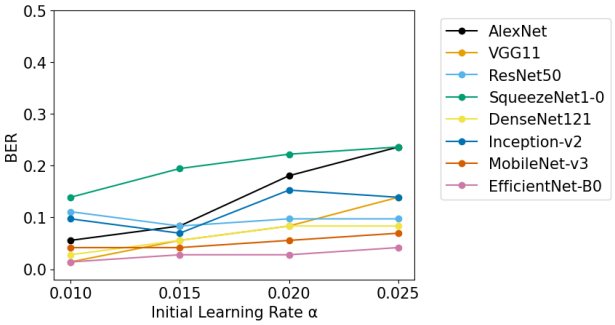


Fig. 4: BERs of features, $\mathrm{BER}\,(m_*^\mu)$, vs. Initial learning rate α in fine-tuning across various models.

transfer learning by fine-tuning CNN models, initially pre-trained on ImageNet [20] dataset, using various datasets. Based on this prior research [19], each CNN model pre-trained on ImageNet will be fine-tuned using the CIFAR-100 [21] dataset. All CNN models were pretrained using the ImageNet dataset. They were trained using Nesterov momentum with a momentum parameter of 0.9. Training ran for 100 epochs with a batch size of 128 images. A cosine decay learning rate scheduler was used, which decayed the learning rate α based on a cosine function as training progressed. The momentum parameter for batch normalization was set based on $\max(1-10/s, 0.9)$, where s represents the number of steps per epoch. For fine-tuning, we used the CIFAR-100 dataset. This dataset contains 60,000 color images, each of which consists of 32×32 pixels. Of these images, 50,000 were used for training and 10,000 for validation. During fine-tuning, the images were resized to match the input image size used during pre-training. Since CIFAR-100 is a 100-class classification task, the output layer of the network was modified to have 100 nodes.

Using a large initial learning rate α during fine-tuning can lead to significant changes in network weights, which can make accurately extracting features difficult. Therefore, we investigated the effect of the initial learning rate on the BER. Figure 4 shows the BERs of features, $\mathrm{BER}\,(m_*^\mu)$, extracted from each model with initial learning rates $\alpha = 0.010, 0.015, 0.020$, and 0.025. These results show that fine-tuning causes errors in the features of all models. It was confirmed that the learning rate affects performance, as a larger initial learning rate results in a higher BER. When features containing errors were input into the AWM and its component, HAM, the model produced error-corrected watermarks. Consequently, for all models and across all initial learning rates α, both the BER on the feature, $\mathrm{BER}\,(m_*^\mu)$, and the BER on the watermark at time 20, $\mathrm{BER}\,(m_{20}^\mu)$, achieved 0.0. This demonstrates that complete recall of the watermark is possible. Furthermore, we confirmed that the HAM alone is sufficient for error correction in this case.

C. Bit length of features and watermarks

Even when degraded features were extracted from fine-tuned models, the HAM could correct errors. This capability likely depends on two factors: the number of input nodes, K, which represents the bit length of the features; and the number of output nodes, N, which represents the bit length of the watermark. Thus, we investigated how varying K and N affects performance.

Figure 5 presents the BER of the watermark when the output node number is fixed at (a) $N = 256$ or (b) $N = 500$ and the input node number K is varied. The features input to the HAM are degraded features obtained through fine-tuning using an initial learning rate $\alpha = 0.025$. The colored lines represent the eight models. The results indicate that the BERs of the watermarks became 0.0 when $K \geq 128$. This suggests that a minimum of 128 input nodes is required when storing the features of the eight models in the HAM. We also examined the case in which the number of input nodes in the HAM was fixed at $K = 144$, while the number of output nodes, N, was varied. The results showed that the BER remained at 0.0 even when N varied. Based on these findings, we concluded that the performance of the HAM depends heavily on the number of models to be stored and the bit length of the features, K, but not much on the bit length of the watermark, N.

V. Conclusions

Previous watermarking methods for protecting CNN models had the drawback of degrading the models' performance due to the embedded watermark. To address this issue, we propose using the Associative Watermarking Method (AWM) to protect CNN models. Similar to the method of Nagai *et al.*, we used the weights of the convolutional layers as features. Since AWM requires binary features, we applied a thresholding process to the real-valued weights and used the resulting binarized vectors as features. As a result of applying our proposed method, we confirmed that watermarks could be extracted without error from eight different CNN models.

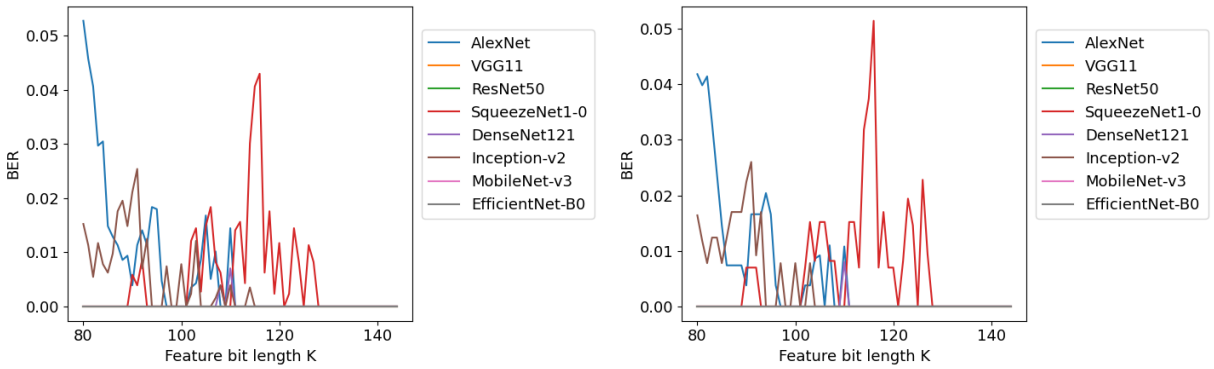
To evaluate the error-correcting ability of AWM, we examined its capability to extract watermarks after fine-tuning the CNN model. Using a large initial learning rate α during fine-tuning caused the network's weights to change significantly, resulting in a higher bit error rate in the extracted features. Despite the degradation of this feature, the watermark output from AWM consistently showed a BER of 0.0 at all initial learning rates. This demonstrates that our proposed method is robust against fine-tuning.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Numbers JP20K11973 and JP24K15106, Support Center for Advanced Telecommunications Technology Research (SCAT), and the Cooperative Research Project of the RIEC, Tohoku University.

References

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.



(a) Watermark bit length $N = 256$ (b) Watermark bit length $N = 500$

Fig. 5: BER of watermarks from HAM vs. feature bit length K

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[3] J. Mayer and R. A. Silva, "Efficient informed embedding of multi-bit watermark," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. iii–389, 2004.

[4] P. Comesana, N. Merhav, and M. Barni, "Asymptotically optimum embedding strategy for one-bit watermarking under gaussian attacks," in *Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, vol. 6819, pp. 98–109, 2008.

[5] T. Liu and P. Moulin, "Error exponents for one-bit watermarking," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03)*, vol. 3, pp. III–65, 2003.

[6] T. Furon, "A constructive and unifying framework for zero-bit watermarking," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 2, pp. 149–163, 2007.

[7] H. Kim, "CRT-based color image zero-watermarking on the DCT domain," *International Journal of Contents*, vol. 11, no. 3, pp. 39–46, 2015.

[8] A. Rani, A. K. Bhullar, D. Dangwal, and S. Kumar, "A zero-watermarking scheme using discrete wavelet transform," *Procedia Computer Science*, vol. 70, pp. 603–609, 2015.

[9] C. Dong, H. Zhang, J. Li, and Y. wei Chen, "Robust zero-watermarking for medical image based on DCT," in *2011 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT)*, pp. 900–904, 2011.

[10] R. Kanegae and M. Kawamura, "Performance evaluation of associative watermarking using statistical neurodynamics," *Journal of the Physical Society of Japan*, vol. 93, no. 11, p. 114004, 2024.

[11] R. Kanegae and M. Kawamura, "Proposal of associative watermarking method," in *Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1376–1381, 2022.

[12] J. Guo and M. Potkonjak, "Watermarking deep neural networks for embedded systems," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2018.

[13] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX security symposium (USENIX Security 18)*, pp. 1615–1631, 2018.

[14] Y. Nagai, Y. Uchida, S. Sakazawa, and S. Satoh, "Digital watermarking for deep neural networks," *International Journal of Multimedia Information Retrieval*, vol. 7, pp. 3–16, 2018.

[15] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference (BMVC)* (E. R. H. Richard C. Wilson and W. A. P. Smith, eds.), pp. 87.1–87.12, 2016.

[16] M. Kawamura, M. Okada, and Y. Hirai, "Dynamics of selective recall in an associative memory model with one-to-many associations," *IEEE transactions on neural networks*, vol. 10, no. 3, pp. 704–713, 1999.

[17] Y. Hirai, "A model of human associative processor (HASP)," *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 851–857, 1983.

[18] Y. Hirai, "Mutually linked HASPs: A solution for constraint-satisfaction problems by associative processing," *IEEE transactions on systems, man, and cybernetics*, no. 3, pp. 432–442, 1985.

[19] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2661–2671, 2019.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, 2009.

[21] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[24] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," *arXiv preprint arXiv:1602.07360*, 2016.

[25] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[28] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, 2019.

[29] L. He, Z. He, T. Luo, and Y. Song, "Shrinkage and redundant feature elimination network-based robust image zero-watermarking," *Symmetry*, vol. 15, no. 5, p. 964, 2023.