# Reversible Watermarking in Deep Convolutional Neural Networks for Integrity Authentication

Xiquan Guan
University of Science and Technology of China
gxq13@mail.ustc.edu.cn

Huamin Feng
Beijing Electronic Science and Technology Institute
fenghm@besti.edu.cn

Weiming Zhang*
University of Science and Technology of China
zhangwm@ustc.edu.cn

Hang Zhou
University of Science and Technology of China
zh2991@mail.ustc.edu.cn

Jie Zhang
University of Science and Technology of China
zjzac@mail.ustc.edu.cn

Nenghai Yu*
University of Science and Technology of China
ynh@ustc.edu.cn

## ABSTRACT

Deep convolutional neural networks have made outstanding contributions in many fields such as computer vision in the past few years and many researchers published well-trained network for downloading. But recent studies have shown serious concerns about integrity due to model-reuse attacks and backdoor attacks. In order to protect these open-source networks, many algorithms have been proposed such as watermarking. However, these existing algorithms modify the contents of the network permanently and are not suitable for integrity authentication. In this paper, we propose a reversible watermarking algorithm for integrity authentication. Specifically, we present the reversible watermarking problem of deep convolutional neural networks and utilize the pruning theory of model compression technology to construct a host sequence used for embedding watermarking information by histogram shift. As shown in the experiments, the influence of embedding reversible watermarking on the classification performance is less than ±0.5% and the parameters of the model can be fully recovered after extracting the watermarking. At the same time, the integrity of the model can be verified by applying the reversible watermarking: if the model is modified illegally, the authentication information generated by original model will be absolutely different from the extracted watermarking information.

## CCS CONCEPTS

• **Security and privacy → Authentication**;

## KEYWORDS

Reversible watermarking, Convolutional neural networks, Security, Integrity authentication

*Weiming Zhang and Nenghai Yu are the corresponding authors.


*MM '20, October 12–16, 2020, Seattle, WA, USA*

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413729

**ACM Reference Format:**
Xiquan Guan, Huamin Feng, Weiming Zhang, Hang Zhou, Jie Zhang, and Nenghai Yu. 2020. Reversible Watermarking in Deep Convolutional Neural Networks for Integrity Authentication. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3394171.3413729

## 1 INTRODUCTION

Deep convolutional neural networks (CNNs) have obtained significant achievements in computer vision recently such as image classification [5], target tracking [9] and automatic driving [2]. However, the structures of the models are increasingly complex and the training of deep neural network models is difficult: several weeks are essential for a deep ResNet (ResNet152) with GPUs on ImageNet [5]. As a result, a large number of trained deep learning models have been published on the website to help people reproduce the results or improve the performance of networks by fine-tuning. During the spread of these trained models, illegal tampering has become an important issue threatening the security of the shared models. A classical method is backdoor attacks on CNNs [4]. The backdoor is defined as a hidden pattern injected into a deep neural network model by modifying the parameters while training. The backdoor does not affect the model's performance on clean inputs, but forces the model to produce unexpected behavior if and only if a specific input is applied. Besides, model-reuse attacks [7] also threaten the networks. These illegal tampering will leave fatal flaws and reduce the accuracy of the trained model. Once these "infected" parent models are utilized for training, the flaws will spread like viruses in the child models and if these "infected" child models are applied in financial or security field, the flaws are likely to be exploited, which will cause destructive impact. Therefore, to ensure there is no illegal tampering on the model, that is, integrity authentication of the model, is a significant research content of model application and security.

Aimed at the security of models, there are two main protecting categories against illegal tampering: defense and authentication. Defense focuses on detection and erasure. In these methods, all models are assumed to have been tampered with illegally. Taking the backdoors defense as an example, Wang *et al.* [21] proposed *Neuron Cleanse* and scanned all the model output labels to infer the

potential hidden triggers. Chen *et al.* [3] applied *Activation Clustering* to detect data maliciously inserted into the training set for injecting backdoors. Liu *et al.* [10] proposed *Fine-Pruning* to remove backdoor triggers by pruning redundant neurons. Most of these methods detect the backdoors passively based on the characteristics of backdoors themselves and may easily lead to missing alarm and false alarm, which will impact the performance of models after utilizing passive defense category, especially for "clean" models.

Another protection category is authentication which is realized by embedding some meaningful information artificially such as watermarking of CNNs. According to whether the internal details of the model are known to the public, the model watermarking can be roughly categorized into two types: white-box watermarking and black-box watermarking. *White-box watermarking* embeds the watermarking information in the model internals such as weights and bias, which assumes that the internal details are public. Uchida *et al.* [20] propose the first CNNs watermarking technique. They choose the weights of a specific layer to embed a binary watermarking in the cover model by adding a regularization term to the loss function in the training process. Besides, Rouhani *et al.* [13] embed the watermarking in the probability density function of the data abstraction obtained in different layers of the model. *Black-box watermarking* embeds the watermarking into the model which only has application programming interface (API) access by choosing a set of key pairs to alter the decision boundary of the cover model. Yossi *et al.* [1] utilize the images with triggers and the corresponding key labels to retrain the cover model. Zhang *et al.* [24] propose three different generation methods of watermarking key images including choosing images in another unrelated dataset, superimposing some images from training data with additional meaning content and random noise images. Very recently, Zhang *et al.* [23] provided a watermarking framework to protect the image processing networks in black-box way.

However, these watermarking techniques are all irreversible. In the embedding process, the irreversible watermarking can only reduce the impact on the performance of the original model as much as possible, but this kind of watermarking still permanently modifies the internal parameters and destroys the integrity of the model. Therefore, this irreversible watermarking is unacceptable for integrity authentication. In order to achieve model integrity authentication, we need to propose a method which can not only embed watermarking information in the model, but also completely recover the original model parameters after extracting the watermarking, which is much more important in the models of military domain, medical domain, law application and so on. Inspired by the digital image reversible watermarking techniques, which can recover the carrier after extracting the watermarking, we proposed the first reversible model watermarking for integrity authentication of CNNs.

Generally speaking, nearly all reversible watermarking algorithms consist of two steps. First, a host sequence with a small entropy should be generated for embedding, i.e., a sharp histogram achieved by prediction errors [14]. Second, users embed the watermarking information into the host sequence by specific coding theories such as difference expansion [19], histogram shift [12] and recursive coding [25]. With the development of the techniques, the coding theories have reached the optimal. So how to construct a host sequence with lower entropy is a significant research goal for reversible watermarking in images. At present, the main way of constructing host sequence is using the correlation of image pixels. Nevertheless, the characteristics of parameters in CNNs are totally different from pixels in images. Due to the incomprehensibility of the CNNs, the correlation of parameters can not be described. At the same time, the format of the parameters are different between CNNs and images. As a result, the traditional reversible watermarking methods for images can not be applied to the model directly and it is crucial to construct the host sequence which is suitable for CNNs.

To this end, we propose a CNNs watermarking method based on the pruning theory of model compression to construct the host sequence for reversible watermarking embedding. Besides, we propose a framework to realize the reversible watermarking embedding of CNNs by utilizing the coding theory learning from the images. In experiments, we take the classification networks as examples to show the effectiveness of reversible watermarking. The results of model integrity authentication is also shown in our paper. The contributions of this paper are summarized as follows:

(1) We present a novel problem: embedding reversible watermarking into CNNs for integrity authentication.
(2) We propose a method to construct the host sequence of trained model and formulate a framework to embed the reversible watermarking into CNNs by histogram shift.
(3) We perform comprehensive experiments in different models to show the performance of reversible watermarking on trained models.

## 2 REVERSIBLE WATERMARKING OF CNNS

### 2.1 Problem Formulation

For the convenience of description, we consider the $n$ convolution layers $C = \{C_1, C_2, \cdots, C_n\}$ of CNN model $\mathcal{M}$. We use a triplet $C_i = \langle \mathcal{L}_i, \mathcal{W}_i, * \rangle$ to define the $i$-$th$ convolution layer, where $\mathcal{L}_i \in \mathbb{R}^{c \times h \times w}$ is the input tensor of layer $i$ and $\mathcal{W}_i \in \mathbb{R}^{d \times c \times k \times k}$ is the weights of all filters in layer $i$. The $*$ denotes the convolution operation. $c$ and $d$ denote the number of input channels and output channels respectively. $h$ and $w$ denote the height and width of input and $k$ is the size of convolution kernel.

The target of reversible watermarking embedding is to embed a $T$-bit vector $B \in \{0,1\}^T$, which is encrypted as a watermarking before, into $\mathcal{M}$ and obtain the marked model $\mathcal{M}'$. So the task can be described as following:

$$\begin{cases} \mathcal{M}' = Emb(\mathcal{M}, B) \\ (\mathcal{M}, B) = Ext(\mathcal{M}') \end{cases} \tag{1}$$

where $Emb(\cdot)$ and $Ext(\cdot)$ present the embedding algorithm and extraction algorithm, which are reversible for each other.

### 2.2 Proposed Framework

In this part, we briefly introduce the framework of reversible watermarking of CNNs. As shown in Fig. 1, the embedding process begins from original model (at the left of Fig. 1) and mainly includes three steps: host sequence construction, data preprocessing and watermarking embedding. The extraction process starts from

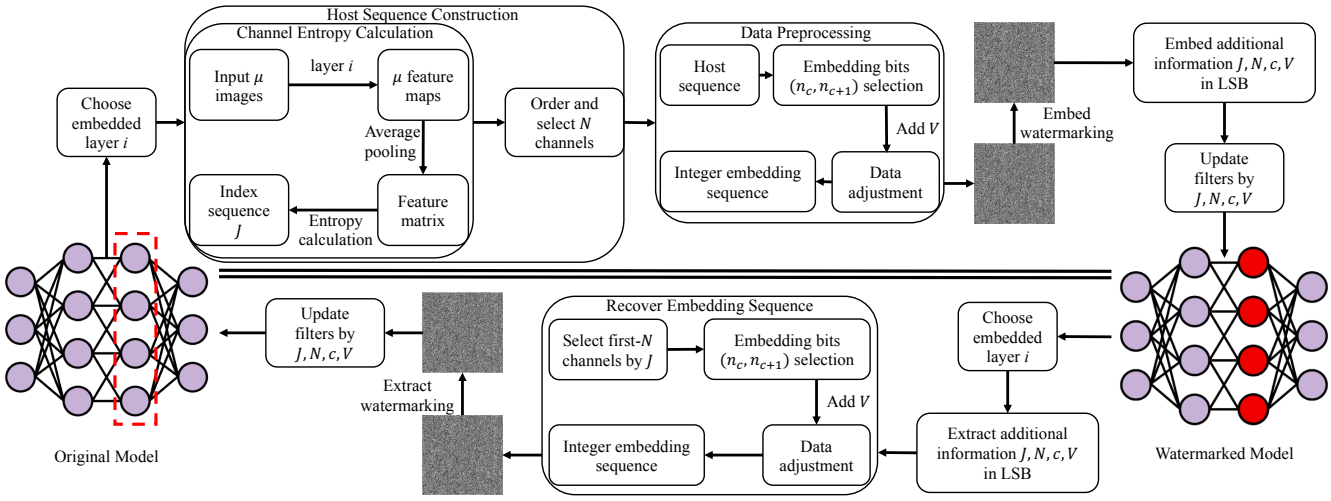


**Figure 1: Reversible watermarking framework in CNNs.**

watermarked model (at the right of Fig. 1) and is inverse to the embedding process. Next, we take the watermarking embedding process as an example to introduce the specific implementation of our proposed method.

## 2.3 Host Sequence Construction

As mentioned before, it is much more difficult to apply the traditional image reversible data hiding methods directly into CNN, that is to say, we must construct the host sequence for models utilizing their own characteristics. Inspired by the pruning theory in [11], we adopt the entropy to rank the importance of the parameters, and select the parameters with small entropy to construct the host sequence. Notice that in irreversible watermarking method and entropy-based pruning theory, convolution layers are used as targets. Therefore, we also consider the convolution layers only and utilize the weight parameters to construct the host sequence for reversible watermarking embedding.

For the convolution layer $i$ in model $\mathcal{M}$, according to the regulations of CNN, each filter in layer $i$ corresponds to a single channel of its activation tensor $\mathcal{L}_{i+1}$, which is also the input of layer $i+1$. In entropy-based channel pruning theory [11], the entropy should be calculated first to measure the importance of each channel. As a result, we first select $\mu$ images $\mathcal{I} = \{I_1, I_2, \cdots, I_\mu\}$ from validation set as the model input. For image $I_g \in \mathcal{I}$ input $\mathcal{L}_i \in \mathbb{R}^{c\times h\times w}$ and the filter of this layer $\mathcal{W}_i \in \mathbb{R}^{d\times c\times k\times k}$, a corresponding activation tensor $\mathcal{L}_{i+1}^g$, which is a $d\times h^{'}\times w^{'}$ tensor, will be obtained obviously. Since the output feature map reflects the weights characteristics of this layer, we use the output of the layer $i$ as the basis for weight importance measurement. Here, we utilize global average pooling to convert the tensor into a $d$ vector as $f_j \in \mathbb{R}^d$. Therefore, each channel of layer $i$ will get a score of image $I_g$ in this way. In order to calculate the entropy, we input the whole images in $\mathcal{I}$ to calculate the channel score and obtain a matrix $\mathcal{F} \in \mathbb{R}^{\mu\times d}$ as following:

$$\mathcal{F} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_\mu \end{pmatrix} \triangleq \left(\mathcal{F}_{:,1}, \mathcal{F}_{:,2}, \cdots, \mathcal{F}_{:,d}\right) \tag{2}$$

where $d$ is the channel number of output. For each channel $l \in \{1, 2, \cdots, d\}$, we take the distribution vector $\mathcal{F}_{:,l}$ as consideration to compute the entropy value. In order to get the frequency distribution, we first divide $\mathcal{F}_{:,l}$ into $m$ different bins and calculate the probability of each bin. Then, the entropy value can be calculated as follows:

$$H_l = -\sum_{r=1}^{m} p_r \log p_r \tag{3}$$

where $p_r, r = \{1, 2, \cdots, m\}$ is the probability of bin $r$ and $H_l$ is the entropy of channel $l$. It should be noticed that there is a $\log(\cdot)$ function in the calculation formula of entropy Eq. (3), so the requirement of $p_r \neq 0$ must be satisfied. As a result, the compromise of the number of bins has to be considered. If we divide too much bins, some $p_r$ will become 0 and the entropy will be meaningless. On the contrary, if $m$ is too small, the entropy of each channel will be not reflected enough. In our method, we utilize the iteration to obtain the largest number of $m$, which will ensure that the probability of each bin satisfies $p_r \neq 0$.

For the $d$ channels of layer $i$, we can obtain a corresponding entropy sequence $H = \{H_1, H_2, \cdots, H_d\}$. According to the magnitude of entropy, we can sort the $H$ and obtain the ascending sequence $H = \{H_{j_1}, H_{j_2}, \cdots, H_{j_d}\}$ and obtain an index of the importance of channels as $J = \{j_1, j_2, \cdots, j_d\}$. Here we select an integer $N < d$ and utilize the channels corresponding to the top $N$ indexes in $J$ to construct the host sequence. As analyzed above, the smaller the entropy is, the less important the parameters are.

The filter weights $\mathcal{W}_i \in \mathbb{R}^{d\times c\times k\times k}$ of layer $i$ can be rewritten as $\mathcal{W}_i = \{\mathcal{W}_1, \mathcal{W}_2, \cdots, \mathcal{W}_d\}$, where the elements of $\mathcal{W}_i$ belong

to $\mathbb{R}^{c\times k\times k}$. We can sort the $\mathcal{W}_i$ by the first $N$ indexes in the index sequence $J = \{j_1, j_2, \cdots, j_d\}$ and obtain the sorting sequence $\mathcal{W}_i^N = \left\{\mathcal{W}_{j_1}, \mathcal{W}_{j_2}, \cdots, \mathcal{W}_{j_d}\right\}$. For each $\mathcal{W}_{j_l} \in \mathcal{W}_i^N$, we define $K_\epsilon^{j_l} \in \mathbb{R}^{k\times k}$ as the kernel weights where $\epsilon \in \{1, 2, \cdots, c\}$ and, therefore, $\mathcal{W}_{j_l} = \left\{K_1^{j_l}, K_2^{j_l}, \cdots, K_c^{j_l}\right\}$. In order to construct the host sequence more similar with an image, we rearrange $\mathcal{W}_i^N$ as $W_i$:

$$W_i = \begin{pmatrix} K_1^{j_1} & K_2^{j_1} & \cdots & K_c^{j_1} \\ K_1^{j_2} & K_2^{j_2} & \cdots & K_c^{j_2} \\ \vdots & \vdots & \ddots & \vdots \\ K_1^{j_d} & K_2^{j_d} & \cdots & K_c^{j_d} \end{pmatrix}_{N\times c} \tag{4}$$

Note that $K_\epsilon^{j_l} \in \mathbb{R}^{k\times k}$, so the $W$ can also be written as following:

$$W_i = \begin{pmatrix} \omega_{1,1} & \omega_{1,2} & \cdots & \omega_{1,k\times c} \\ \omega_{2,1} & \omega_{2,2} & \cdots & \omega_{2,k\times c} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{k\times N,1} & \omega_{k\times N,2} & \cdots & \omega_{k\times N,k\times c} \end{pmatrix} \tag{5}$$

where $\omega_{\alpha,\beta} \in \mathbb{R}$. And $W_i$ here is taken as the host sequence for watermarking embedding.

## 2.4 Data Preprocessing

As mentioned above, we obtain the host sequence $W_i$ utilizing the pruning theory. However, all the elements in matrix $W_i$ are not integer, which can not be directly applied in the traditional image reversible data hiding method. As a result, in our framework, we intercept two digits from each element of $W_i$ and the range of these intercepted parameters is $[-99, 99]$. Then, we add $V$ to these intercepted parameters to adjust it to the appropriate range, that is, positive integer, where $V \in \mathbb{Z}$ is an adjustable parameter.

In addition to the number of interception digits, the location of interception should be considered. We assume the element $\omega_{\alpha,\beta} \in W_i$ as following:

$$\omega_{\alpha,\beta} = \pm 0.\underbrace{00\cdots0}_{p\ digits} n_1 n_2 \cdots n_q, \tag{6}$$

where $p \geqslant 0$, $q > 0$ and $p, q \in \mathbb{Z}$. In Eq. (6), $n_1$ denotes the first non-zero digit of $\omega_{\alpha,\beta}$, $n_2$ denotes the second non-zero digit of $\omega_{\alpha,\beta}$ and so on. For convenience, we define the $\gamma$-$th$ non-zero digit of $\omega_{\alpha,\beta}$, $n_\gamma$, as the $\gamma$-$th$ significant digit. It should be noticed that for different elements of $W_i$, the value of $p$ is different, that is, the position of the first significant digit is different.

Due to the modification of the first significant digit $n_1$ will cause a great influence on the value of $\omega_{\alpha,\beta}$, we only consider modifying the digits from second significant digit to the last significant digit, namely, $n_2, n_3, \cdots, n_q$. In order to obtain a larger embedding capacity, the theory of Kalker and Willems is considered in our method. In [22], the upper bound of embedding capacity under a given distortion constraint $\Delta$ was proposed as following:

$$\rho_{rev}(\Delta) = \text{maximize}\{E(Y)\} - E(X), \tag{7}$$

where $X$ and $Y$ denote the host sequence and the marked sequence after embedding respectively and $E(\cdot)$ denote entropy calculation function. According to the Eq. (7), the smaller the entropy of host sequence is, the larger the embedding capacity can be obtained. Thus, we calculate the entropy of all possible host sequences constructed by intercepting **TWO** different significant digits. Then we decide the position of the selected significant digit according to the values of these entropy.

Specifically, we take the $i$-$th$ convolution layer as an example. For all the elements of $W_i$, $\omega_{\alpha,\beta}$, we first select the second significant digit $n_2$ and the third significant digit $n_3$. After adjusting the value by $V$ mentioned before, we construct optional host sequence $W_i^{2,3}$. Then we count the frequency and calculate the entropy of $W_i^{2,3}$ as $E_{2,3}$. Similarly, we can obtain the entropy values $E_{3,4}, E_{4,5}, \cdots, E_{q-1,q}$. According to the method in [22], we choose the significant digit pairs, defined as $(n_c, n_{c+1})$, corresponding to the minimum entropy to construct the host sequence.

Once we ensure the selection digits $(n_c, n_{c+1})$, we can get the integer $\omega^*_{\alpha,\beta} = \pm n_c n_{c+1}$. It should be noticed that the symbols of $\omega^*_{\alpha,\beta}$ and $\omega_{\alpha,\beta}$ are consistent, that is, if the symbol of $\omega_{\alpha,\beta}$ is positive, the symbol of $\omega_{\alpha,\beta}$ will be positive and vice versa. Then we can obtain $\widehat{\omega}_{\alpha,\beta} = \omega^*_{\alpha,\beta} + V$ and the host sequence $\widehat{W} = (\widehat{\omega}_{\alpha,\beta})_{k\times N, k\times c}$ after data processing.

## 2.5 Embedding and Extracting Strategy

**Embedding:** The integer host sequence $\widehat{W}$ generated above can be considered as a traditional grayscale image and we can utilize image reversible data hiding strategy to embed watermarking. In this paper, we choose histogram shift (HS) strategy [12]. The embedding process contains the two basic steps as following:

**(1) Histogram generation:** for $\widehat{\omega}_{i,j} \in \widehat{W}$, we generate the histogram $H(\widehat{\omega})$ the same as image: counting the number of different elements in a matrix $\widehat{W}$.

**(2) Histogram modification:** we define the value in $\widehat{W}$ corresponding to the histogram peak as $\widehat{\Omega}_{max}$ and the histogram valley (generally speaking, 0) as $\widehat{\Omega}_{min}$. Without loss of generality, in our framework, $\widehat{\Omega}_{max} < \widehat{\Omega}_{min}$. As mentioned in [16], the HS encoding algorithm embedding one bit $b$ can be described as following:

$$\widehat{\omega}'_{i,j} = \begin{cases} \widehat{\omega}_{i,j} + b, & \widehat{\omega}_{i,j} = \widehat{\Omega}_{max} \\ \widehat{\omega}_{i,j} + 1, & \widehat{\omega}_{i,j} \in (\widehat{\Omega}_{max}, \widehat{\Omega}_{min}) \\ \widehat{\omega}_{i,j} & \widehat{\omega}_{i,j} \notin [\widehat{\Omega}_{max}, \widehat{\Omega}_{min}). \end{cases} \tag{8}$$

As shown in Fig. 2 and Fig. 3, through embedding algorithm, watermark information is embedded into host sequence by histogram shift.

After embedding the watermarking information, the matrix $\widehat{\omega}'_{i,j} \in \widehat{W}'$ is generated and we can replace the original $\mathcal{W}_i$ as $\mathcal{W}'_i$ by $J$, $N$, $c$ and $V$, where $c$ is the position of $n_c$. First, we can obtain the new selection digits $(n'_c, n'_{c+1})$ of $\widehat{\omega}'_{\alpha,\beta}$ as $n'_c n'_{c+1} = \widehat{\omega}'_{\alpha,\beta} - V$. Therefore, the modified $\omega'_{\alpha,\beta}$ is shown as following:

$$\omega'_{\alpha,\beta} = \pm 0.\underbrace{00\cdots0}_{p\ digits} n_1 n_2 \cdots n'_c n'_{c+1} \cdots n_q, \tag{9}$$

then the elements in Eq. 5 can be replaced as $\omega'_{\alpha,\beta}$ and get the modified $\mathcal{W}'_{j_l}$. According to the parameter $N$ and index sequence $J$,

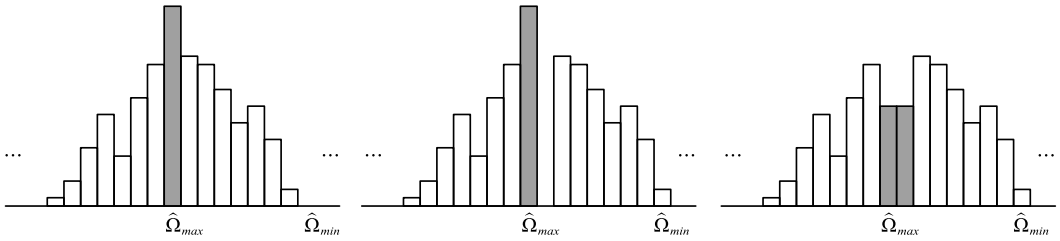


**Figure 2: Illustration of Ni *et al.*'s method [12]. Here the histogram on the left is the initial histogram, the histogram in the middle is generated by shifting the bins more than $\widehat{\Omega}_{max}$ towards right by 1 to create a vacant bin for data embedding and the histogram on the right is the histogram embedded with watermark information utilizing HS. Without loss of generality, we assume that the number of binary 0 and the number of binary 1 to be embedded are equal.**

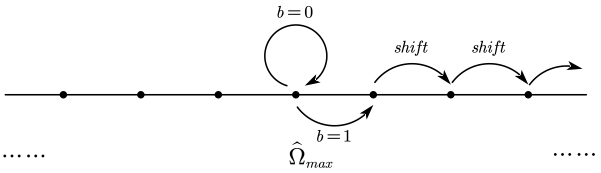


**Figure 3: Mapping rule of histogram bins described in [12]: the watermark information $b$ is embedded into $\widehat{\Omega}_{max}$ and the values bigger than $\widehat{\Omega}_{max}$ in $\widehat{W}$ are shifted right while the values smaller than $\widehat{\Omega}_{max}$ in $\widehat{W}$ are remained unchanged.**

we can replace $\mathcal{W}_{j_1}, \mathcal{W}_{j_2}, \cdots, \mathcal{W}_{j_d}$ in $\mathcal{W}_i$ by $\mathcal{W}'_{j_1}, \mathcal{W}'_{j_2}, \cdots, \mathcal{W}'_{j_d}$ and obtain the update filter weights $\mathcal{W}'_i$, that is, marked model $\mathcal{M}'$.

It should be noted that the additional informations $J$, $N$, $c$ and $V$ should also be embedded in the the filters. Here we embed these bits into the last binary bit (converting parameters to binary numbers) of $\mathcal{W}'_i$. Similar to the previous definition, we can define a matrix $\widetilde{W}$ by arranging the parameters of all channels in order as following:

$$\widetilde{W} = \begin{pmatrix} \widetilde{\omega}_{1,1} & \widetilde{\omega}_{1,2} & \cdots & \widetilde{\omega}_{1,k\times c} \\ \widetilde{\omega}_{2,1} & \widetilde{\omega}_{2,2} & \cdots & \widetilde{\omega}_{2,k\times c} \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{\omega}_{k\times d,1} & \widetilde{\omega}_{k\times d,2} & \cdots & \widetilde{\omega}_{k\times d,k\times c} \end{pmatrix} \tag{10}$$

where $\widetilde{\omega}_{\alpha,\beta} \in \mathbb{R}$. Then, we convert $\widetilde{\omega}_{\alpha,\beta}$ to binary number as $\widetilde{\omega}^B_{\alpha,\beta}$ and replace the last bits of $\widetilde{\omega}^B_{\alpha,\beta}$ by encrypted additional informations $J$, $N$, $c$ and $V$. In order to keep the reversibility, we reserve a part of space in the head of watermarking information to store the original last bits information of those replaced $\widetilde{\omega}^B_{\alpha,\beta}$ above.

**Extraction and Restoration:** We first extract the additional informations $J$, $N$, $c$ and $V$ from the filter in layer $i$. Then we construct the marked sequence $\widehat{W}'$ using the methods in 2.3 and 2.4. Then, the same as embedding process, we generate the histogram $H(\widehat{\omega}')$ and extract the embedded bit $b$ according to the following:

$$b = \begin{cases} 1, & \widehat{\omega}'_{i,j} = \widehat{\Omega}_{max} + 1 \\ 0, & \widehat{\omega}'_{i,j} = \widehat{\Omega}_{max}, \end{cases} \tag{11}$$

and after extracting the embedding bits, the original element $\widehat{\omega}$ can be recovered as:

$$\widehat{\omega}_{i,j} = \begin{cases} \widehat{\omega}'_{i,j} - 1, & \widehat{\omega}'_{i,j} \in (\widehat{\Omega}_{max}, \widehat{\Omega}_{min}] \\ \widehat{\omega}'_{i,j} & \widehat{\omega}'_{i,j} \notin [\widehat{\Omega}_{max}, \widehat{\Omega}_{min}]. \end{cases} \tag{12}$$

As mentioned above, we can recover the original $\mathcal{W}_i$ and update the filters in layer $i$ to obtain the original model $\mathcal{M}$.

# 3 EXPERIMENTS

In this section, we firstly introduce the experimental settings (Sec.3.1) and compare the top-5 accuracy between our proposed method and irreversible watermarking technique modifying parameters directly (Sec.3.2). We then show the multi-embedding of reversible watermarking performance (Sec.3.3). Finally we show the process of integrity certification utilizing reversible watermarking (Sec.3.4).

## 3.1 Settings

For experiments, we adopt three pretrained networks AlexNet [8], VGG19 [17], ResNet152 [5], DenseNet121 [6] and MobileNet [15] as the target models $\mathcal{M}$, and utilize the ImageNet validation images dataset consists of 50,000 color images in 1000 classes with 50 images per class to calculate the entropy of channels. For the process of host sequence construction, according to the relationship between depth of layer and model performance, we decide to choose the last three layers of these models to embed the reversible watermarking and choose the first $N = 128$ channels in VGG19 and ResNet152, the first $N = 32$ channels in DenseNet121, the first $N = 48$, $N = 64$, $N = 96$ channels for the different layers in AlexNet and the first $N = 320$, $N = 960$, $N = 960$ channels for the different layers in MobileNet to rearrange the weights parameters. The reason for the difference of $N$ value is that the weight tensors of different convolutions in different models are different. Besides, we choose $V = 128$ as the adjustable parameter and $c = 2$ as the selected significant digit position. The implementation is based on Python3.5 and MATLAB R2018a with the NVIDIA RTX 2080 Ti GPU.

## 3.2 Comparison with Non-reversible Methods

First, we organize a comparison table according to the characteristics of irreversible watermarking and reversible watermarking as shown in Fig. 1. Here we divide the irreversible watermarking into two categories, one is robust reversible watermarking, the other is non-robust reversible watermarking, similar to image steganography.

**Table 1**
**Comparison of Reversible Watermarking and Irreversible Watermarking: Qualitative comparison of two different watermarks.**

| | Watermarking | | |
|---|---|---|---|
| | Reversible | Irreversible | |
| | | Robust | Non-robust |
| Fragility | ✓ | | ✓ |
| Robustness | | ✓ | |
| Reversibility | ✓ | | |
| Capacity | Medium | Small | Large |
| Application | Integrity authentication | Intellectual property protection | Covert Communication |

**Table 2**
**Top-5 Classification Accuracy on ImageNet: The comparison between our proposed method RW and LSBR embedded in the last three layers of three classical classification models: AlexNet, VGG19, ResNet152.**

| Network | Layer | Clean Model Accuracy (%) | Marked Model Accuracy (%) | | Length of Watermark (bits) |
|---|---|---|---|---|---|
| | | | LSBR [18] | RW (ours) | |
| AlexNet | III | 75.9 | 75.7 | 75.7 | 12442 |
| | II | | **76.0** | 75.8 | 49766 |
| | I | | **75.8** | 75.6 | 22118 |
| VGG19 | III | 81.1 | 80.9 | **81.2** | 88474 |
| | II | | 81.1 | 81.1 | 88474 |
| | I | | **81.0** | 80.8 | 88474 |
| ResNet152 | III | 85.9 | 85.5 | **85.7** | 88474 |
| | II | | 85.5 | **86.0** | 88474 |
| | I | | 85.6 | **85.9** | 88474 |

For reversible watermarking, it is fragile, reversible and the capacity is medium. It is mainly used for integrity authentication. In contrast, irreversible watermarking is irreversible. For robust irreversible watermarking, it is robust, which is utilized for intellectual property protection. For non-robust irreversible watermarking, the capacity is large. Non-robust irreversible watermarking, which is also fragile similar to reversible watermarking, is usually used for covert communication. Since we do not consider robustness and the reversible watermarking is first proposed, we only choose the two types fragile watermarking, non-robust irreversible watermarking and reversible watermarking, for comparison in the next experiment.

To illustrate the universality of our reversible watermarking method (RW), we choose a non-robust irreversible watermarking method proposed by Song *et al.* [18]. They embedded watermarking information by least significant bit replacement (LSBR). In our experiments, we embed the watermarking in the selected layers and calculate the top-5 accuracy of classification. For convenience, we utilize I, II, III to represent the last layer, the second to last layer and the third to last layer. In order to make our comparative experiment more convincing, we first select the last three convolution layers of AlexNet to embed different sizes of watermark information to analyze the impact of the length of watermark information on the performance of the model. Then we choose the last three convolution layers of VGG19 and ResNet152 to embed the same size of watermark information to analyze the influence of different models on the performance of the model.

As shown in Table 2, with same embedding bits in the same layer, the top-5 classification accuracies before or after embedding two types watermarking are almost equal ($-0.4\% \sim +0.1\%$). Besides, the accuracies between LSBR and our proposed method are almost equal ($-0.5\% \sim +0.2\%$). It should be noticed that our proposed method is reversible watermarking which can be extracted and maintain the model integrity. According to the results, embedding the reversible watermarking hardly affects the classification results of the model, which is much more different from image reversible watermarking. This can be explained in two ways. On the one hand, the modification has little influence on the value of the parameters. On the other hand, the number of parameters in these models is very large and the modification of parameters in model is limited. Besides, our method achieves the reversibility without affecting the performance of the model compared with non-robust irreversible watermarking.

## 3.3 Multi-layered Reversible Watermarking

In this part, we compare the classification performance of the models between single-layered watermarking embedding and multi-layered watermarking embedding. For the multi-layered embedding, we modify the parameters of each selected layer respectively, and then merged them into a complete modification model.

First, we choose AlexNet, VGG19 and ResNet152 to compare the effect of embedding watermark in different layers on the performance of the model. As shown in Table 3, the accuracies between clean model and multi-layered watermarking embedded model are almost equal ($-0.3\% \sim +0.2\%$). Then, we choose DenseNet121 and

MobileNet to compare the effect of embedding watermark in single layer and multiple layers. As shown in Table 4, the accuracies between clean model and embedded watermarking model are almost equal ($-0.6\% \sim -0.1\%$). As analyzed above, the embedding of single-layered watermarking has little influence on the model performance, so whether we embed multi-layered watermarking or single-layered watermarking, the performance of the models do not change much, which provides the possibility to recover the tampered model by embedding more watermarking information of model parameters' characteristic in the future.

**Table 3**
**Top-5 Classification Accuracy on ImageNet: The results of multi-layered watermarking embedding in the last three layers of three classical classification models: AlexNet, VGG19, ResNet152.**

| Mode | Classification Accuracy (%) | | |
|---|---|---|---|
| | AlexNet | VGG19 | ResNet152 |
| Clean Model | 75.9 | 81.1 | 85.9 |
| I&II | 75.8 | 81.3 | 85.8 |
| I&III | 75.7 | 80.8 | 85.9 |
| II&III | 75.8 | 81.1 | 85.9 |
| I&II&III | 75.8 | 81.1 | 85.8 |

**Table 4**
**Top-5 Classification Accuracy on ImageNet: The comparison between our proposed method RW embedded in different layers and clean models**

| Network | Layer | Clean (%) | RW (%) | Length of Watermark (bits) |
|---|---|---|---|---|
| DenseNet121 | I | 80.4 | 80.3 | 5530 |
| | I&II | | 80.0 | 11060 |
| | I&II&III | | 80.2 | 16590 |
| MobileNet | I | 76.8 | 76.6 | 46080 |
| | I&II | | 76.6 | 47376 |
| | I&II&III | | 76.2 | 70416 |

**Table 5**
**Model reconstruction error rate: Compare the consistency between the reconstructed model and the original model. A reconstruction rate of 0 indicates that the algorithm is completely reversible..**

| Model | Reconstruction error rate (%) | |
|---|---|---|
| | Singe layer | Multiple layers |
| AlexNet | 0 | 0 |
| VGG19 | 0 | 0 |
| ResNet152 | 0 | 0 |
| DenseNet121 | 0 | 0 |
| MobileNet | 0 | 0 |

At the last of this subsection, we compared the difference between the original model and the reconstructed model after the extraction for the five models mentioned above. The results are shown in Table 5. Both the experimental results and the theoretical analysis can prove that our method is completely reversible, that is, the integrity of the model is preserved.

## 3.4 Integrity Authentication

In this part, we realize the integrity authentication applied reversible watermarking. First, we utilize a Hash algorithm SHA-256 (Secure Hash Algorithm 256) to obtain the characteristic of the whole model. Then, we embed the SHA-256 value into the convolution layer by our proposed reversible watermarking algorithm. Due to the the excellent characteristics of the Hash algorithm, no matter where the attacker modifies the model, the newly generated SHA-256 value will be different from the extracted SHA-256 value.

As shown in Fig. 4, Alice is the holder of the model and she regards the SHA-256 value as the watermarking $WM_1$ and embed it into the model by our reversible watermarking algorithm. Then she uploads her watermarked model to cloud server for others to download. Bob downloads Alice's model from cloud server but he does not know whether the model is complete (Unknown Model), so he extracts the watermarking (defined as $WM_2$) from the unknown model and calculates the SHA-256 $WM_3$ from reconstructed model. It will be two cases here comparing $WM_2$ and $WM_3$: (1) if the model is modified illegally by Mallory, then $WM_2 \neq WM_3$ (top right of Fig. 4). (2) if the model is not modified, then $WM_2 = WM_3$ (bottom right of Fig. 4).

For our algorithm, we give a brief security analysis as following: We begin by presenting a definition for the security: the model is **Integrity** if it is impossible for an attacker to modify the model without being discovered. As mentioned in above, we use the SHA-256 value as the reversible watermarking to verify integrity shown in Fig. 4. Then the security of our method is reduced to the security of a cryptographic Hash algorithm (SHA-256) which is collision-resistant. Meanwhile, a security Hash function $Hash(x)$, where the domain is $X_h$ and the range is $Y_h$, is collision-resistant if it is difficult to find:

$$Hash(x_1) = Hash(x_2) \; for \; x_1, x_2 \in X_h \; and \; x_1 \neq x_2. \tag{13}$$

Since the Hash function of SHA-256 is collision-resistant up to now, the method for integrity authentication is secure.

In our experiments, we choose the last convolution layer of ResNet152 to embed SHA-256 value as watermarking information. All experiments have shown that no matter where we modify or erase the parameters, our method can detect that the model has been tampered with.

## 4 CONCLUSION

In this paper, we present a new problem: embedding reversible watermarking into deep convolutional neural networks (CNNs) for integrity authentication. Since the state-of-art model watermarking techniques are irreversible and destroy the integrity of the model permanently, these methods are not suitable for integrity authentication. Inspired by the traditional image integrity authentication, we consider the reversible watermarking and apply it into CNNs. According to the characteristics of CNNs, we propose a method to construct the host sequence of trained model and formulate a framework to embed the reversible watermarking into CNNs by histogram shift. In the experiments, we demonstrate that our reversible watermarking in CNNs is effective and we utilize the reversible watermarking for integrity authentication in whole model.

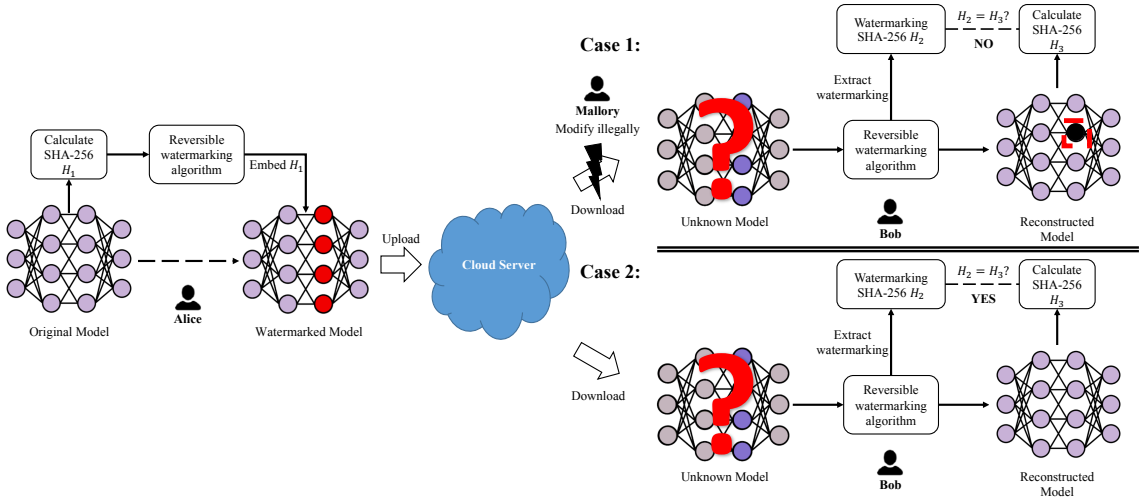


**Figure 4: Integrity authentication protocol utilizing reversible watermarking of CNNs.**

In the future work, we will study how to determine the location where the model is modified and recover the modified parameters as much as possible by the extracted watermarking information. Furthermore, we just utilize our framework on CNNs, so we will research how to extend the reversible watermarking technique to other deep neural networks for integrity authentication.

## ACKNOWLEDGMENTS

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB0804100, Natural Science Foundation of China under Grant U1636201 and Exploration Fund Project of University of Science and Technology of China under Grant YD3480002001.

## REFERENCES

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

[2] Arantxa Casanova, Guillem Cucurull, Michal Drozdzal, Adriana Romero, and Yoshua Bengio. 2018. On the iterative refinement of densely connected representation levels for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 978–987.

[3] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2018. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728* (2018).

[4] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).

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

[6] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

[7] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. 2018. Model-reuse attacks on deep learning systems. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 349–363.

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

[9] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. 2019. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4282–4291.

[10] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 273–294.

[11] Jian-Hao Luo and Jianxin Wu. 2017. An entropy-based pruning method for cnn compression. *arXiv preprint arXiv:1706.05791* (2017).

[12] Zhicheng Ni, Yun-Qing Shi, Nirwan Ansari, and Wei Su. 2006. Reversible data hiding. *IEEE Transactions on circuits and systems for video technology* 16, 3 (2006), 354–362.

[13] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2018. Deepsigns: A generic watermarking framework for ip protection of deep learning models. *arXiv preprint arXiv:1804.00750* (2018).

[14] Vasiliy Sachnev, Hyoung Joong Kim, Jeho Nam, Sundaram Suresh, and Yun Qing Shi. 2009. Reversible watermarking algorithm using sorting and prediction. *IEEE Transactions on Circuits and Systems for Video Technology* 19, 7 (2009), 989–999.

[15] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[16] Yun Qing Shi, Xiaolong Li, Xinpeng Zhang, Haotian Wu, and Bin Ma. 2016. Reversible Data Hiding: Advances in the Past Two Decades. *IEEE Access* 4 (2016), 1–1.

[17] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[18] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. 2017. Machine Learning Models that Remember Too Much. In *the 2017 ACM SIGSAC Conference*.

[19] Jun Tian. 2003. Reversible data embedding using a difference expansion. *IEEE transactions on circuits and systems for video technology* 13, 8 (2003), 890–896.

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

[21] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. *Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks* (2019), 0.

[22] FM Willems and T Kalker. 2003. Capacity bounds and code constructions for reversible data-hiding. *IS&T/SPIE Proceedings, Security and Watermarking of Multimedia 19 Contents V* 5020 (2003).

[23] Jie Zhang, Dongdong Chen, Jing Liao, Han Fang, Weiming Zhang, Wenbo Zhou, Hao Cui, and Nenghai Yu. 2020. Model Watermarking for Image Processing Networks.. In *AAAI*. 12805–12812.

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

[25] Weiming Zhang, Biao Chen, and Nenghai Yu. 2012. Improving various reversible data hiding schemes via optimal codes for binary covers. *IEEE transactions on image processing* 21, 6 (2012), 2991–3003.