



# Deep neural network watermarking based on a reversible image hiding network

Linna Wang<sup>1</sup> · Yunfei Song<sup>1</sup> · Daoxun Xia<sup>1,2</sup>

Received: 5 March 2022 / Accepted: 24 January 2023 / Published online: 18 February 2023  
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2023

## Abstract

Recently, many researchers have proposed deep neural network (DNN) watermarking technologies, DNN watermarking approaches can be divided into two categories: static watermarking and dynamic watermarking methods. A static watermark is embedded into the internal parameters of a DNN model, but a dynamic watermark relies on the specific training data of the DNN model and uses the associated neuron activation map or the output result by the DNN model to extract the watermark information. Dynamic watermarks mostly use DNN application programming interfaces (APIs) to remotely access DNN models and extract their watermarks to prove their copyright, so dynamic watermarking technology is more popular. According to the distribution inconsistency between a dynamic watermark and training data, an attacker can detect the dynamic watermark, so that the model owner cannot obtain the desired prediction results and then verify the copyright of the suspect model. To this end, we propose a dynamic watermarking approach based on a reversible image hiding network, which improved the undetectability of a DNN watermark, and it can perfectly reconstruct the secret image as the copyright logo of a DNN model. We perform our work on the MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, and Caltech-101 datasets. The experimental results show that our method has higher DNN watermarking accuracy and higher undetectability with no significant side effects on the main functions of the host DNN model.

**Keywords** Digital watermarking · Deep neural network · Image hiding · Ownership verification · Undetectability

## 1 Introduction

Over the last few years, deep learning (DL) has been proven very successful in many different areas, including computer vision, speech recognition, natural language processing, and other critical fields of artificial intelligence. Many scientific and technological companies have deployed deep neural network (DNN) models in commercial products to achieve improved efficiency. Although these deep convolution neural network (DCNN) frameworks, such as LeNet, AlexNet, the Visual Geometry Group network (VGGNet), GooLeNet, and the residual network (ResNet), have made tasks more easy to complete, training a DNN model is still a difficult task

because it requires large-scale datasets, massive computing resources and designers' wisdom. As expensive digital assets, various DNN models are easily attacked or stolen. Although many researchers have proposed different defense methods against DNN attacks, attackers develop powerful attack methods. Therefore, how to protect the intellectual property rights of DNN models has become an urgent problem in academic and industrial circles. For multimedia information, digital watermarks are embedded in redundant multimedia information to protect this information [1–3]. Instead of embedding watermarks into multimedia information, an ownership watermark is constructed for the model owners according to the feature of original images and the ownership statement to protect images [4], and the watermark key is generated by the interrelationships between contents of given Arabic text to protect Arabic text [5]. For DNN model, many parameters are contained in a DNN model, and the redundant parameters of such a DNN model are used to embed a deep model watermark. This approach uses the parameters of the neural network to carry the watermark; it does not affect the performance of the DNN model

✉ Daoxun Xia  
dxxia@gznu.edu.cn

<sup>1</sup> School of Big Data and Computer Science, Guizhou Normal University, Guiyang 550025, China

<sup>2</sup> Engineering Laboratory for Applied Technology of Big Data in Education, Guizhou Normal University, Guiyang 550025, China

due to overfitting, but it requires the watermark extractor to know the structure and the parameters of the DNN model. In fact, we usually remotely access models through their application programming interfaces(APIs). Therefore, many researchers have suggested that DNN watermarking technology does not need to master the internal mechanisms of DNN models to extract their watermarks. When the model owner finds a suspicious DNN model, he or she inputs the specific data into the API and outputs the prediction results that can represent the copyright of the DNN model. We call these specific data key samples. Normally, the distributions of the key samples and the training data are very different. An attacker observes the distribution of the input data of the DNN model and prevents the DNN model from reasoning about the data whose distribution is inconsistent with the training data. Inspired by the above observations, to prevent attackers from detecting the key samples and avoid copyright verification, we suggest inputting the key samples that are consistent with the distribution of the DNN model training set to perform a copyright check on the associated DNN model.

## 2 Related work

### 2.1 Static watermarking

A static watermark is embedded into the internal parameters of the given DNN model. The training process does not depend on the specific data of the DNN model [6]. This approach is generally divided into two stages: the watermark embedding stage and the watermark extraction stage. In 2017, a digital watermark was first applied to a DNN model for copyright protection. Uchida Y et al. [7] proposed an algorithm to embed watermark information into the weights of a DNN model, and the watermark is embedded in the weights during the training process. Because the watermark is embedded in the regularizer, their approach does not impair the performance of networks and the embedded watermark does not disappear after fine-tuning or pruning; however, when extracting the watermark, the parameters of the model need to be accessed. This limits the applicability of this method in business. Wang T et al. [8] proved that Uchida Y's DNN model watermarking technology modifies the statistical distribution of the DNN model parameters. A change in the parameter distribution can be used not only to detect the existence of a depth model watermark but also to calculate the length of the DNN model watermark. After an attacker obtains the DNN model watermark information, he/she can design a DNN model watermark removal algorithm, and then the watermark information embedded by the model owner becomes invalid. Wang T et al. [9, 10] proposed a new solution. The training and detection processes for a DNN

model watermark were designed as a generator and discriminator, respectively, to generate a countermeasure network. Their experiments showed that the weight parameter distribution of a DNN model hardly changes after embedding the watermark information. Different from the DNN model watermarking technology proposed by Uchida, Kuribayashi et al. [11] first embedded watermark information into the frequency component of the sampling weight of a DNN model by using the quantization algorithm called dither modulation-quantization index modulation(DM-QIM), and then dispersed the watermark information into the sampling weight of the DNN model by using the inverse discrete cosine transform(DCT). The key to the method is to ensure that the weight distribution change exhibited by the DNN model is as small as possible. A static watermark requires the watermark extractor to know the structure and the parameters of the associated DNN model. Therefore, static watermarks are not suitable for commercial applications.

### 2.2 Dynamic watermarking

A dynamic watermark depends on the specific training data of the given DNN model, the associated neuron activation maps, or output results are used to extract the watermark information [6]. The dynamic watermark mainly uses the DNN backdoor technology. During the training stage, the backdoor watermark data is added to the training dataset. In our paper, the backdoor watermark data is the key sample. During the prediction stage, the key sample triggers a specific output result. The specific output result is the target label that we preset for the key sample before training the model, The correspondence between key samples and target labels is known only to the model owner. Rouhani et al. [12] proposed a DNN model watermarking framework, deepsigns, which works by learning the activation maps in the different layers of a DNN model, DeepSigns is robust in terms of fine-tuning, pruning, and watermark overwriting. Adi et al. [13] proposed a simple and effective technique for watermarking DNNs by backdooring. Zhang et al. [14] also proposed three backdoor watermark generation algorithms for DNN models. Meaningful content, unrelated content and noise are embedded into a DNN model as watermarks through the DNN model watermarking framework. Experiments show that these algorithms are robust to parameter pruning, fine-tuning and model inversion attacks on DNN models. Chen et al. [15] also proposed a DNN model watermarking framework called Deepmarks, which realizes fingerprint insertion in the weight distribution through DNN regularization. If an attacker removes this fingerprint, the performance of the DNN model will be affected. This approach is the first framework for use in a large model distribution system. On the one hand, it can provide ownership verification; on the other hand, it can track users. If a DNN

model has been distributed before embedding watermarks or the embedded watermarks are overwritten or deleted, the copyright of the DNN model cannot be proven. Zhang et al. [16] proposed embedding watermarks in the output of a DNN model and marking the DNN model for complex image processing tasks. If a thief attacks the DNN model by using its API and obtains an alternative model with similar performance, the model owner can extract the watermarks of the DNN model from the output of the alternative model. Then, these watermarks will be compared with the watermarks embedded in the original DNN model, and it will be judged whether the alternative model is the stolen according to the comparison value. The use of the output prediction results of a DNN model to prove its copyright is a kind of dynamic watermarking technology, and it often encounters evasion attacks. An evasion attack occurs when the attacker evades the copyright verification of the DNN model owner when the watermark cannot be removed [17]. At present, DNN model watermarking technology for resisting evasion attacks is vital and required. However, it is still severely underresearched. Li et al. [18, 19] proposed the first DNN copyright protection framework based on blind watermarks. Blind watermarks aim to generate key samples with similar distributions that are almost indistinguishable from ordinary samples. Attackers cannot detect key samples and prevent the use of key samples to verify the model copyright; this method achieves good performance on the MNIST and CIFAR-10 datasets. However, the range of applications of this method has been restricted by its inherent constraint: it uses an encoder to generate key samples, the hiding capacity of the encoder is limited, and it cannot perfectly reconstruct secret images. If high-capacity secret images are embedded in datasets with high resolutions to generate key samples, the quality of the key samples will decrease. This affects the prediction results for key samples in the DNN model and reduces the undetectability of key samples. To solve the above problems, we propose a DNN model watermarking method based on a reversible image hiding network. A reversible image hiding network utilizes an inverse learning mechanism to simultaneously learn the image hiding and display processes. Our method can hide a full-size secret image into a host image of the same size and perfectly reconstruct the secret image as the identifier of the DNN model copyright.

### 2.3 Image hiding technology

Image hiding is an important research direction of steganography; it attempts to hide a whole image in another Image hiding technology is mainly used for confidential communication. The purpose of image hiding is to conceal the secret image into the cover image in an imperceptible way, and then reveal the secret image perfectly at the receiver. Only

the informed receiver is allowed to reveal the secret image, but not visible to others. DL can be used in practical end-to-end image steganography [20]. Volkhonskiy D et al. [21] proposed a new model for generating image-like containers based on deep convolution GAN(DCGAN). This approach enables the use of standard steganography algorithms to generate more secure setganalysis message embeddings. Experimental results show that the model can successfully deceive a steganography analyzer. Shi et al. [22] proposed secure steganography GAN (SSGAN) model. The model can generate images with high visual quality. In most cases, it can provide safer covers for steganography. The model can be used for adaptive steganography algorithms. Zhang et al. [23] proposed a high-capacity image steganography network, SteganoGAN, by using GANs to solve steganography tasks. Experiments show that their SteganoGAN model obtains a higher relative payload than existing methods and can avoid detection. Jing et al. [24] proposed a reversible image hiding network called HiNet. This network not only meets the high capacity needs of the image hiding task but also models the image display task as the reverse process of image hiding; that is, all the network parameters required for hiding and displaying can be obtained by training the network only once. Experimental results show that this method achieves the most advanced image restoration accuracy, hiding security, and invisibility performance. Hence, in this work, we introduce a DNN watermarking method based on a reversible image hiding network to generate key samples to make the key samples of DNN models undetectable.

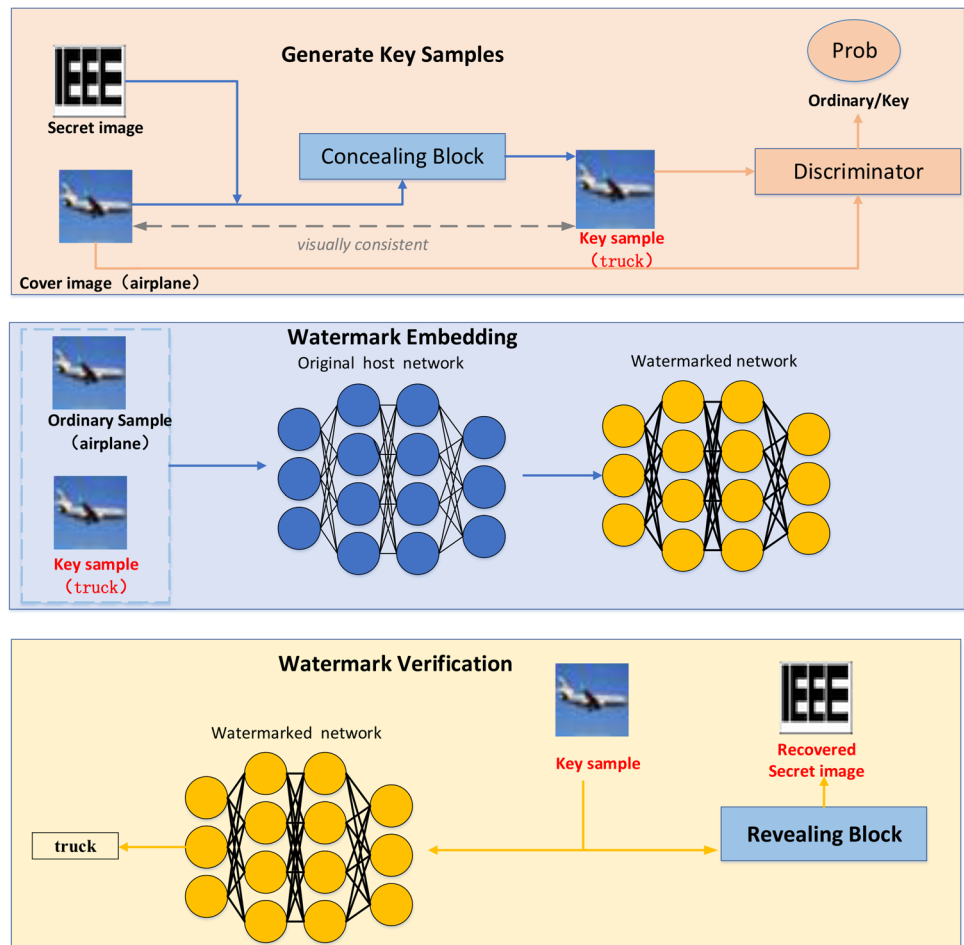
## 3 Methodology

As shown in Fig. 1, we propose a DNN watermarking framework that consists of three parts: a reversible image hiding network, a discriminator and an original host network. In what follows, we describe the key sample generation process that utilizes the reversible image hiding network and the discriminator, the embedding process for key samples in the host DNN model, and the verification process for key samples in the host DNN model.

### 3.1 Overview

In the generation stage, we generate key samples through the concealing blocks  $R_{conceal}$  of the reversible image hiding network (Sect. 3.2). In the embedding stage, we embed watermarks in the original host network (Sect. 3.3). In the verification stage, we verify the copyright of the original host network (Sect. 3.4). In Sect. 3.5, we introduce the objective loss function of our method. Specifically, in Sect. 3.5.1, we present the generation loss  $\mathcal{L}_{gen}$ ; in Sect. 3.5.2, we present the discrimination loss  $\mathcal{L}_{dis}$ ; and in Sect. 3.5.3, we present the

**Fig. 1** The DNN watermarking framework based on reversible image hiding network



embedding loss  $\mathcal{L}_{\text{emd}}$ . In Sect. 3.6, we summarize the steps of the proposed DNN watermarking framework algorithm based on a reversible image hiding network in detail.

### 3.2 Generation

The cover image  $X_{\text{cover}}$  is obtained from a subset training set, and the secret image  $X_{\text{secret}}$  hidden in  $X_{\text{cover}}$  by the concealing blocks  $R_{\text{conceal}}$  of the reversible image hiding network to generate key samples  $X_{\text{key}}$ , i.e.,

$$X_{\text{key}} = R_{\text{conceal}}(X_{\text{cover}}, X_{\text{secret}}). \quad (1)$$

$R_{\text{conceal}}$  aims to hide  $X_{\text{secret}}$  in  $X_{\text{cover}}$  in an imperceptible way. Here, we take the CIFAR-10 dataset as an example. First, we take an image from CIFAR-10 as  $X_{\text{cover}}$ , where  $X_{\text{cover}}$  is an ordinary sample labeled “airplane”; then, we use the grayscale IEEE logo as the default target watermark. The IEEE logo image is concealed in the ordinary sample labeled “airplane” by  $R_{\text{conceal}}$ . Finally, the generated key sample  $X_{\text{key}}$  uses “truck” as the target label. The distribution of  $X_{\text{key}}$  is infinitely close to that of  $X_{\text{cover}}$ , i.e.,

$$X_{\text{key}} \rightarrow X_{\text{cover}}. \quad (2)$$

Here, we use the discriminator to ensure that the distortion of  $X_{\text{cover}}$  is very small after  $X_{\text{secret}}$  is hidden in  $X_{\text{cover}}$ . As shown in Fig. 1, ordinary samples are used as the positive samples of the discriminator, and key samples are used as the negative samples of the discriminator. The ordinary samples and key samples are sent to train the discriminator, and the trained discriminator indicates whether the key samples are generated by  $R_{\text{conceal}}$ . If the discriminator cannot distinguish between ordinary samples and key samples, this will indicate that the key images generated by  $R_{\text{conceal}}$  are undetectable.

### 3.3 Embedding

After the key samples are obtained through the concealing blocks  $R_{\text{conceal}}$ , the next step is to embed watermarks in the original host network. As shown in Fig. 1, the target label “truck” is used as the label of the key samples. The key samples  $X_{\text{key}}$  and ordinary samples  $X$  are spliced together as the new training set of the original host network, and they are sent to the original host network  $O$  to obtain a watermarked network  $W$ , i.e.,

$$\text{Train}(O, X, X_{\text{key}}) \rightarrow W. \tag{3}$$

The Watermarked network not only has the function of classifying the ordinary samples, but also has function of classifying key samples, i.e.,

$$\text{Test}(W, X) \rightarrow Y, \text{Test}(W, X_{\text{key}}) \rightarrow Y_{\text{key}}. \tag{4}$$

Because the parameters of the original host network are redundant, the key samples will not affect the performance of the original host network in classifying ordinary samples.

### 3.4 Verification

The last step is to verify the copyright of the original host network. Consider a scenario in which a DNN model owner suspects that a remotely deployed model violates its copyright interest. To confirm the ownership of the remote model, in this procedure, the model owner first prepares a set of key samples  $X_{\text{key}} \{x_{\text{key}1}, x_{\text{key}2}, \dots\}$  via the concealing blocks  $R_{\text{conceal}}$ :

$$X_{\text{key}} = R_{\text{conceal}}(X_{\text{cover}}, X_{\text{secret}}). \tag{5}$$

Then, the model owner issues a prediction query to the remote DNN with these key samples and obtains the resulting predictions. As shown in Fig. 1, if the remote DNN classifies the key sample as the target label “truck”, it shows that the remote DNN is the watermarked network. The revealing blocks of the reversible image hiding network  $R_{\text{reveal}}$  aim to perfectly recover  $X_{\text{secret\_rev}}$  at the receiver.  $R_{\text{reveal}}$  restores the key samples  $X_{\text{key}}$  in reverse to reconstruct secret images, i.e.,

$$X_{\text{secret\_rev}} = R_{\text{reveal}}(X_{\text{key}}). \tag{6}$$

$X_{\text{secret\_rev}}$  can be used to identify the DNN model copyright.  $X_{\text{secret\_rev}}$  and  $X_{\text{secret}}$  are visually consistent, i.e.,

$$X_{\text{secret\_rev}} \rightarrow X_{\text{secret}}. \tag{7}$$

In addition, the threshold between the accuracy of the regular test set  $\text{test\_acc}^*$  in the watermarked network and the benchmark accuracy  $\text{test\_acc}$  in the watermark-free network does not exceed  $T$ , and the accuracy of the key samples  $\text{wm\_acc}^*$  is greater than the minimum value  $\text{Min}$  in the watermarked network, i.e.,

$$\text{wm\_acc}^* \geq \text{Min}, \text{test\_acc}^* - \text{test\_acc} \leq T. \tag{8}$$

This means that the copyright of the host network can be proven.

### 3.5 Loss function

The loss function includes three parts: the generation loss  $\mathcal{L}_{\text{gen}}$ , the discrimination loss  $\mathcal{L}_{\text{dis}}$  and the embedding loss  $\mathcal{L}_{\text{emd}}$ .

#### 3.5.1 Generation loss

When the secret images are embedded into the cover images, six different types of losses are considered to ensure their visual quality: the concealing loss  $\ell_{\text{con}}$ , the revealing loss  $\ell_{\text{rev}}$ , the low-frequency wavelet loss  $\ell_{\text{low}}$ , the structural similarity index measure(SSIM) loss  $\ell_{\text{ssim}}$ , the adversarial loss  $\ell_{\text{adv}}$ , and the host DNN model loss  $\ell_{\text{dnn}}$ , i.e.,

$$\mathcal{L}_{\text{gen}} = \lambda_1 \times \ell_{\text{con}} + \lambda_2 \times \ell_{\text{rev}} + \lambda_3 \times \ell_{\text{low}} + \tag{9}$$

$$\lambda_4 \times (1 - \ell_{\text{ssim}}) + \lambda_5 \times \ell_{\text{adv}} + \lambda_6 \times \ell_{\text{dnn}}. \tag{10}$$

Here, the concealing loss  $\ell_{\text{con}}$  guarantees the concealing performance of the cover image, and  $\ell_{\text{con}}$  is the mean squared error(MSE) loss, i.e.,

$$\ell_{\text{con}} = \frac{1}{N} \sum_{i=1}^N \|X_{\text{key}} - X_{\text{cover}}\|^2. \tag{11}$$

The revealing loss  $\ell_{\text{rev}}$  ensures the recovery performance of the secret image, and  $\ell_{\text{rev}}$  is the MSE loss, i.e.,

$$\ell_{\text{rev}} = \frac{1}{N} \sum_{i=1}^N \|X_{\text{secret\_rev}} - X_{\text{secret}}\|^2. \tag{12}$$

The low-frequency wavelet loss  $\ell_{\text{low}}$  enhances the hiding security, and  $\ell_{\text{low}}$  is the MSE loss, i.e.,

$$\ell_{\text{low}} = \frac{1}{N} \sum_{i=1}^N \|X_{\text{key\_low}} - X_{\text{cover\_low}}\|^2. \tag{13}$$

The SSIM loss  $\ell_{\text{ssim}}$  is defined as the error related to the basic properties (texture, structure, etc.) of the image.

$$\ell_{\text{ssim}} = \frac{1}{N} \sum_{i=1}^N \text{ssim}(X_{\text{key}} - X_{\text{cover}}). \tag{14}$$

The adversarial loss  $\ell_{\text{adv}}$  ensures that the  $X_{\text{key}}$  generated by  $R_{\text{reveal}}$  are judged as positive samples by the discriminator as much as possible, and  $\ell_{\text{rmdv}}$  is the binary cross-entropy loss, i.e.,

$$\ell_{\text{adv}} = -\frac{1}{N} \sum_{i=1}^N [\text{Dis}(X_{\text{key}}) \cdot \log(\text{valid}) + \tag{15}$$

$$(1 - \text{Dis}(X_{\text{key}})) \cdot \log(1 - \text{valid})]. \tag{16}$$

The host DNN model loss  $\ell_{\text{dnn}}$  ensures that the labels of the key samples predicted by the watermarked network are as close to the normal labels in the host model as possible, and  $\ell_{\text{dnn}}$  is the cross-entropy loss, i.e.,

$$\ell_{\text{dnn}} = -\frac{1}{N} \sum_{i=1}^N \text{Dnn}(X_{\text{key}}) \cdot \log(\text{wm\_label}). \tag{17}$$

### 3.5.2 Discrimination loss

$\mathcal{L}_{\text{dis}}$  ensures that a false image has a value close to the 0, while the real image has a value close to the 1; that is,  $\text{Dis}(x_{\text{key}})$  is close to fake, and the output  $\text{Dis}(x_{\text{cover}})$  is close to valid. Otherwise, the value is punished.  $\mathcal{L}_{\text{dis}}$  includes the key sample identification loss function  $\ell_{\text{key}}$  and the cover image identification loss function  $\ell_{\text{cover}}$ , i.e.,

$$\mathcal{L}_{\text{dis}} = \ell_{\text{key}} + \ell_{\text{cover}}. \tag{18}$$

Both  $\ell_{\text{key}}$  and  $\ell_{\text{cover}}$  are the binary cross-entropy losses, i.e.,

$$\ell_{\text{key}} = -\frac{1}{N} \sum_{i=1}^N [\text{fake} \cdot \log(\text{Dis}(X_{\text{key}})) + (1 - \text{fake}) \cdot \log(1 - \text{Dis}(X_{\text{key}}))]. \tag{19}$$

$$\ell_{\text{cover}} = -\frac{1}{N} \sum_{i=1}^N [\text{valid} \cdot \log(\text{Dis}(X_{\text{cover}})) + (1 - \text{valid}) \cdot \log(1 - \text{Dis}(X_{\text{cover}}))]. \tag{20}$$

### 3.5.3 Embedding loss

We choose the cross-entropy loss as the embedding loss  $\mathcal{L}_{\text{emd}}$ . First, the training data  $X$  and the key samples  $X_{\text{key}}$  are spliced into inputs, while the labels of the training data label and the target labels of the key samples  $\text{wm\_label}$  are spliced into labels, i.e.,

$$[\text{inputs}, \text{labels}] = [(X, X_{\text{key}}), (\text{label}, \text{wm\_label})]. \tag{21}$$

Then, the inputs are sent to the host DNN model, and the output results of the host DNN model and their labels are used to calculate  $\mathcal{L}_{\text{emd}}$ , i.e.,

$$\mathcal{L}_{\text{emd}} = -\frac{1}{N} \sum_{i=1}^N \text{Dnn}(\text{inputs}) \cdot \log(\text{labels}). \tag{22}$$

### 3.6 Global watermarking algorithm

Algorithm 1 summarizes the steps of the proposed DNN watermarking framework. It takes a host DNN model  $M$ , training data  $D$ , a secret image  $X_{\text{secret}}$ , the minibatch size  $B$  of  $D$ , the minibatch size of the key samples  $B_{\text{key}}$ , the number of key samples  $K$ , and the number of epoches  $E$  as inputs, and it outputs the watermarked DNN model  $M^*$ , the key samples  $(X_{\text{key}}, Y_{\text{key}})$ , and the reconstructed secret image  $X_{\text{secret\_rev}}$ . Here,  $(X_{\text{key}}, Y_{\text{key}})$  and  $X_{\text{secret\_rev}}$  are defined by the owner and protected by the certification authority to indicate model ownership.

---

**Algorithm 1:** DNN watermarking algorithm based on a reversible image hiding network.

---

**Input:** Host DNN model ( $M$ ); Training data ( $D$ ); Secret image ( $X_{\text{secret}}$ ); Minibatchsize ( $B$ ); Minibatchsize of key samples ( $B_{\text{key}}$ ); Key samples number ( $K$ ); Epoch number( $E$ )

**Output:** Watermarked DNN model ( $M^*$ ); Key samples ( $X_{\text{key}}, Y_{\text{key}}$ ); Reconstructed secret image ( $X_{\text{secret\_rev}}$ )

**for**  $\text{epoch} \leftarrow 1$  to  $E$  **do**

Select  $X$  of  $B$  from the Training data  $D$ ,  $x \in D$  ;

Select  $X_{\text{cover}}$  of  $B_{\text{key}}$  from the Training data  $D$ ,  $X_{\text{cover}} \in D$ ;

**Generation:**

**for**  $\text{num\_key} \leftarrow 1$  to  $K$  **do**

$(X_{\text{key}}, Y_{\text{key}}) \leftarrow R_{\text{conceal}}(X_{\text{cover}}, X_{\text{secret}}, \mathcal{L}_{\text{gen}}, \mathcal{L}_{\text{dis}})$

**end**

**Embedding:**

$M^* \leftarrow \text{Train}(M, X, (X_{\text{key}}, Y_{\text{key}}), \mathcal{L}_{\text{emd}});$

**Verification:**

$Y_{\text{key}} \leftarrow \text{Test}(M^*, X_{\text{key}}),$

$Y \leftarrow \text{Test}(M^*, X), Y_{\text{key}} \neq Y;$

$X_{\text{secret\_rev}} \leftarrow R_{\text{reveal}}(X_{\text{key}});$

**end**

---



## 4 Experiments

We use 5 different datasets and 11 different DNN models to perform image classification tasks in this paper. We first introduce the datasets in Sect. 4.1, and then introduce the training details in Sect. 4.2. Next, before we analyze the experimental results, we first present the evaluation criteria in Sect. 4.3. Furthermore, we will demonstrate the experimental results obtained by the proposed DNN watermarking framework with different datasets and different DNN models in Sect. 4.4. Finally, we compare our method with other SOTA methods, and perform some extension experiments in Sect. 4.5.

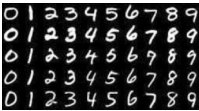

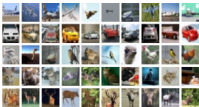

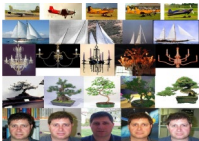
### 4.1 Dataset

We perform the evaluations on benchmark image datasets including MNIST [25], Fashion-MNIST [26], CIFAR-10 [27], CIFAR-100 [27], and Caltech-101 [28]. Table 1 shows the details the datasets.

**MNIST.** The MNIST dataset is a large handwritten digital dataset containing 70,000 gray images at resolutions of 28×28, and its class labels range from 0 to 9. The training set has 60,000 images and the test set has 10,000 images.

**Fashion-MNIST.** The Fashion-MNIST dataset consists of 70,000 gray images of fashion products at resolutions of 28×28. The training set has 60,000 images, and the test set has 10,000 images. The whole set has 10 classes.

**Table 1** Details of datasets used in the experiment. DNN models are trained with five datasets: MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, and Caltech-101

Dataset	Class	Sizes	Total	Train	Test	Samples
MNIST	<b>10</b> (Number 0-9)	28×28	70,000	60,000	10,000	
Fashion-MNIST	<b>10</b> (t-shirt, trouser, ..., bag, ankle boot)	28×28	70,000	60,000	10,000	
CIFAR-10	<b>10</b> (airplane, bird, cat, ..., ship, truck)	32×32	60,000	50,000	10,000	
CIFAR-100	<b>100</b> (aquatic mammals, fish, ..., )	32×32	60,000	50,000	10,000	
Caltech-101	<b>10</b> (airplane, bonsai, car, chandelier, face, ..., turtle, watch)	300×200	2,400	1,800	600	

Bold values represent the number of classes for the datasets

**CIFAR-10.** The CIFAR-10 dataset consists of 60,000 color images at resolutions of  $32 \times 32$ . The training set has 50,000 images, and the test set has 10,000 images. The whole set has 10 classes.

**CIFAR-100.** The CIFAR-100 dataset consists of 60,000 color images at resolutions of  $32 \times 32$ . The training set has 50,000 images, and the test set has 10,000 images. CIFAR-100 has 100 classes. These 100 classes are grouped into 20 superclasses.

**Caltech-101.** The Caltech-101 dataset contains 101 classes of object images, with approximately 40 to 800 samples in each class. The size of each image is approximately  $300 \times 200$ . We select 10 classes from the 101 total classes in our experiment. The dataset includes 1,800 training samples and 600 test samples.

## 4.2 Training details

The experiments are performed on Ubuntu 18.04 with an Intel Xeon Gold 5218 CPU @ 2.30GHz and an NVIDIA QUADRO RTX 8000 GPU. Our DNN watermarking algorithm based on a reversible image hiding network is implemented in Python 3.9.2 and PyTorch 1.9.0. We divide our DNN watermarking framework into three parts: the reversible image hiding network, the discriminator, and the host DNN model. Next, we introduce the training parameter settings of these three parts.

### 4.2.1 Reversible image hiding network

We incorporate a novel reversible image hiding network named HiNet into our watermark solution due to its simplicity and efficiency. HiNet was proposed in [24]. HiNet includes revealing blocks and concealing blocks, revealing blocks are the reverse process of concealing blocks. Revealing blocks and concealing blocks share the same parameters of HiNet, revealing blocks and concealing blocks used the dense block [29]. The details of its use are as follows: the number of concealing blocks  $R_{conceal}$  and revealing blocks  $R_{reveal}$  is set to 16. For the MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100 datasets, we use 1% of the total training samples as cover images to generate the key samples, i.e., we use 600 cover images to generate the key samples for the MNIST and Fashion-MNIST datasets, and use 500 cover images to generate the key samples for the CIFAR-10 and CIFAR-100 datasets. However, for the Caltech-101 dataset, we use 36 cover images to generate the key samples. For each key sample, we randomly select a target label as `wm_label`. The minibatch size for key samples is set to 20. The Adam optimizer [30] ( $\beta_1 = 0.5$ ,  $\beta_2 = 0.999$ ,  $\text{eps} = 1 \times 10^{-6}$ ,  $\text{weight\_decay} = 1 \times 10^{-5}$ ) is adopted with standard parameters and an initial learning rate of  $1 \times 10^{-4.5}$ . We update the learning rate by 0.5 every 150 iterations for MNIST and

Fashion-MNIST and every 600 iterations for CIFAR-10 and CIFAR-100.

### 4.2.2 Discriminator

The discriminator is essentially a binary classifier, which judges whether the key samples are generated through revealing blocks. The discriminator is composed of several linear layers and LReLU. The discriminator is trained in an iterative manner by using the Adam algorithm [30] ( $\beta_1 = 0.5$ ,  $\beta_2 = 0.999$ ) and an initial learning rate of 0.001. When the learning rate remains unchanged after 8 epochs, the learning rate is decayed by 0.2. For the MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100 datasets, the minibatch size for key samples is set to 20, and for the Caltech-101 dataset, the minibatch size of key samples is set to 4.

### 4.2.3 Host DNN model

The host DNN model is the DNN model we want to protect. For example, we use LeNet-3 [31] and LeNet-5 [31] to train MNIST and Fashion-MNIST datasets, VGG-11 [32], VGG-13 [32], VGG-16 [32], and VGG-19 [32] to train CIFAR-10 dataset, ResNet-18 [33], ResNet-34 [33], and ResNet-101 [33] to train CIFAR-100 dataset, PreActResNet-18 [34] and PreActResNet-34 [34] to train Caltech-101 dataset. For the MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100 datasets, the host DNN model is simultaneously trained by using the stochastic gradient descent (SGD) algorithm [35] ( $\text{momentum} = 0.9$ ,  $\text{weight\_decay} = 5 \times 10^{-4}$ ) with an initial learning rate of 0.1, which is decayed by 0.1 every 40 epochs. For the Caltech-101 dataset, because the images have large sizes, the host DNN model is simultaneously trained using SGD [35] with a batch size of 24 (20 original samples and 4 key samples) and an initial learning rate of 0.001, which is decayed by 0.1 per 40 epochs. For the MNIST, and Fashion-MNIST datasets, we train for 30 epochs, and for the CIFAR-10, CIFAR-100, and Caltech-101 datasets, we train for 100 epochs.

## 4.3 Performance evaluation criteria

We compare and analyze the results by using fidelity, effectiveness, and undetectability as the performance evaluation criteria. Fidelity, effectiveness, and undetectability are important reference standards for evaluating the performance of the DNN watermarking framework. Fidelity, effectiveness, and undetectability are mutually restricted, so DNN watermarking technology needs to balance the relationship between the three. A good DNN model watermarking framework should have good fidelity, good validity, and good undetectability at the same time.



**Fidelity**, which determines whether the target DNN reduces the accuracy of the host DNN model due to watermarking; **effectiveness**, which determines whether the model can reach the threshold and verify the ownership of the host DNN model successfully; and **undetectability**, which determines whether the key samples are visually consistent with the ordinary samples (to evaluate undetectability, we use the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) to measure the distortion rate of  $x_{key}$ ). Larger PSNR and SSIM values indicate higher image quality.

### 4.4 Results

The accuracy, PSNR, and SSIM achieved before and after watermarking are compared to verify the fidelity, effectiveness, and undetectability of our DNN watermarking framework on the DNN models. All the evaluated models are trained in two different environments: without watermarks and with watermarks. We first train the DNN models without watermarks and evaluate them on the regular test set. Then, we train DNN models with watermark embeddings and evaluate them on the key samples.

#### 4.4.1 Accuracies for the watermark-free networks

As shown in Table 2, when the LeNet-3 and LeNet-5 models are trained without key samples, the baseline tested on MNIST yields test accuracies of 98.52% and 98.37% and watermark accuracies of 8.60% and 8.60%, respectively.

When the LeNet-3 and LeNet-5 models are trained without key samples, the baseline tested on Fashion-MNIST yields test accuracies of 87.87% and 87.35% and watermark accuracies of 9.20% and 9.75%, respectively.

**Table 2** Accuracy of different DNN models on regular test set and key samples: test\_acc is the benchmark accuracy in watermark-free network, test\_acc\* is the accuracy of the regular test set in watermarked network, wm\_acc is the accuracy of the key samples in

Datasets	DNN models	test_acc	test_acc *	wm_acc	wm_acc *	psnr_s	psnr_c	ssim
MNIST	LeNet-3 [31]	98.52%	97.55% (↓ 0.97)	8.60%	99.20%	23.98	19.01	0.9254
	LeNet-5 [31]	98.37%	97.52% (↓ 0.85)	8.60%	94.00%	27.68	21.15	0.9556
Fashion-MNIST	LeNet-3 [31]	87.87%	86.10%(↓ 1.77)	9.20%	94.40%	<b>21.86</b>	<b>15.42</b>	<b>0.8650</b>
	LeNet-5 [31]	87.35%	84.32%(↓ 3.03)	9.75%	96.15%	22.17	15.70	0.8755
CIFAR-10	VGG-11 [32]	91.15%	90.62%(↓ 0.53)	9.20%	98.60%	39.57	34.43	0.9891
	VGG-13 [32]	93.01%	92.14%(↓ 0.87)	9.20%	100.00%	39.70	32.52	0.9824
	VGG-16 [32]	92.51%	92.14%(↓ 0.37)	9.20%	94.80%	40.67	34.43	0.9897
	VGG-19 [32]	92.33%	91.49%(↓ 0.84)	9.20%	99.20%	42.67	34.51	0.9893
CIFAR-100	ResNet-18 [33]	76.18%	75.29% (↓ 0.89)	1.00%	95.00%	42.83	34.48	0.9896
	ResNet-34 [33]	76.93%	72.63%(↓ 4.30)	1.00%	98.80%	<b>43.10</b>	<b>35.96</b>	<b>0.9932</b>
	ResNet-101 [33]	77.76%	74.78% (↓ 2.98)	1.00%	92.40%	43.03	33.83	0.9879
Caltech-101	PreActResNet-18 [34]	96.33%	95.83% (↓ 0.50)	0.00%	100.00%	34.59	30.51	0.8823
	PreActResNet-34 [34]	97.17%	96.83%(↓ 0.34)	0.00%	100.00%	35.12	29.56	0.8535

When the VGG-11, VGG-13, VGG-16, and VGG-19 models are trained without key samples, the baseline tested on CIFAR-10 yields test accuracies of 91.15%, 93.01%, 92.51%, and 92.33% and watermark accuracies of 9.20%, 9.20%, and 9.20%, respectively.

When the ResNet-18, ResNet-34, and ResNet-101 models are trained without key samples, the baseline tested on CIFAR-100 yields test accuracies of 76.18%, 76.93% and 77.76% and watermark accuracies of 1.00%, 1.00%, 1.00% and 1.00%, respectively.

When the PreActResNet-18 and PreActResNet-34 models are trained without key samples, the baseline tested on Caltech-101 yields test accuracies of 96.33% and 97.17%, and watermark accuracies of 0.00% and 0.00%, respectively.

The accuracy across all key samples  $wm\_acc$  in each watermark-free network does not exceed 10.00%, which represents a totally random guess.

#### 4.4.2 Accuracies for the watermarked networks

When the DNN models are trained with key samples, the threshold between the accuracy of the regular test set  $test\_acc^*$  and the benchmark accuracy  $test\_acc$  does not exceed  $T$ , and the accuracy of the key samples  $wm\_acc^*$  is greater than the minimum value  $Min$ . The copyright of each DNN model can be effectively proven.

As shown in Table 2, all watermarked DNN models achieve  $wm\_acc^*$  values that are greater than 90% on the key samples, and VGG-13, PreActResNet-18, and PreActResNet-34 even achieve accuracies of 100%. A comparison with the  $wm\_acc$  achieved with the same DNN, models shows that the watermarked DNN models learn the key samples very successfully, and we set  $Min$  to 90%. That is,

watermark-free network, and  $wm\_acc^*$  is the accuracy of the key samples in watermarked network. Quantitative results of the proposed DNN watermarking algorithm:  $psnr\_s$  is calculated between  $x_{secret}$  and  $x_{secret\_rev}$ ,  $psnr\_c$  and  $ssim$  are calculated between  $x_{key}$  and  $x_{cover}$

$$wm\_acc^* \geq 90\%. \quad (23)$$

It is found that the performance of the DNN models does not decrease significantly after embedding watermarks. The experimental results show that the accuracy degradation does not exceed 0.97% on the MNIST dataset, 3.03 % on the Fashion-MNIST dataset, 0.87% on the CIFAR-10 dataset, 4.30% on the CIFAR-100 dataset, and 0.50% on the Caltech-101 dataset. The average rates of decline for different datasets with different DNN models is 3.99%, and the embedding of watermarks has little impact on the performance of the DNN models. We set  $T$  to 5%, i.e.,

$$test\_acc^* - test\_acc \leq 5\%. \quad (24)$$

#### 4.4.3 PSNR and SSIM values

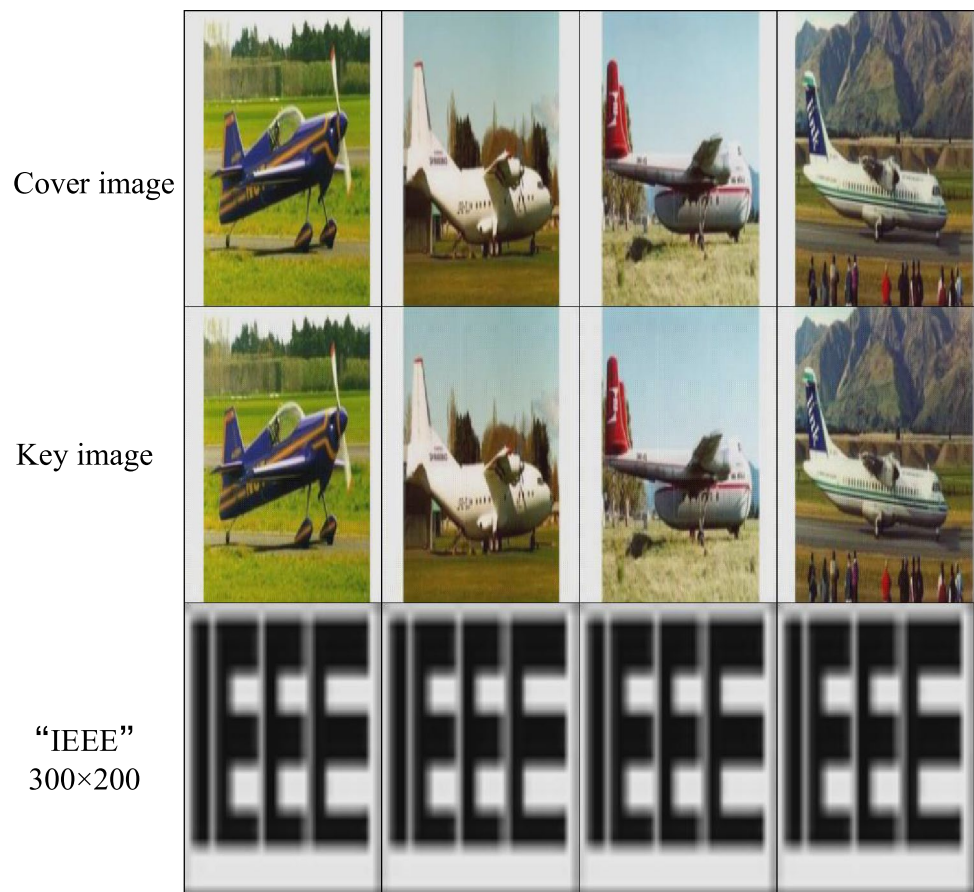
As shown in Table 2, when the CIFAR-100 dataset is used to train ResNet-34,  $psnr\_s$  reaches a maximum of 43.10,  $psnr\_c$  reaches a maximum of 35.96, and  $ssim$  reaches a maximum of 0.9932. However, we can see that the  $psnr$  and  $ssim$  values are relatively low for the MNIST and Fashion-MNIST datasets. When the Fashion-MNIST dataset is

used to train LeNet-3,  $psnr\_s$  reaches a minimum of 21.86,  $psnr\_c$  reaches a minimum of 15.42, and  $ssim$  reaches a minimum of 0.8650. The proposed DNN watermarking algorithm generates images with very high quality for the CIFAR-10, CIFAR-100, and Caltech-101 datasets and generates images with very poor quality for the MNIST and Fashion-MNIST datasets. This is because the DNN watermarking algorithm is based on a reversible image hiding network, and this network makes it easier to complete the task of when utilizing a colorful image dataset with a large size. As shown in Fig. 2, for Caltech-101, we use the grayscale "IEEE" logo with a size of 300×200. We demonstrate the hiding ability of our DNN watermarking framework for PreActResNet-18, and we can see that the visual presentations of the key sample and the cover image are almost the same.

#### 4.4.4 Comparison with other methods

To verify the effectiveness of our approach, we compare our DNN watermarking method with an undetectable dynamic watermarking method (Blind-Watermark, developed in [18]) in Table 3. Table 3 reports the comparison results obtained after testing on ResNet-18, ResNet-34, and ResNet-101 with the CIFAR-100 dataset. For the same

**Fig. 2** Some visual examples to show the capability of the proposed DNN watermarking algorithm: the cover image (Row 1), the key sample (Row 2), and the secret images (Row 3). Almost no difference between the cover image and the key sample can be observed



**Table 3** Comparison with the Blind-Watermark method under three DNN models on the CIFAR-100 dataset

Method	ResNet-18 [33]			ResNet-34 [33]			ResNet-101 [33]								
	test_acc*	wm_acc*	psnr_s	psnr_c	ssim	test_acc*	wm_acc*	psnr_s	psnr_c	ssim	test_acc*	wm_acc*	psnr_s	psnr_c	ssim
Blind-Watermark [18]	73.94%	89.20%	–	28.81	0.9807	74.31%	92.20%	–	29.48	0.9824	74.17%	81.40%	–	31.25	0.9865
Ours	75.29%	95.00%	42.83	34.48	0.9896	72.63%	98.80%	43.10	35.96	0.9932	74.78%	92.40%	43.03	33.83	0.9879

**Table 4** Quantitative results of our method with different size secret images. We take CIFAR-10 task for example

Size	test_acc*	wm_acc*	psnr_s	psnr_c	ssim
Baseline (32×32)	92.14%	94.80%	40.67	34.43	0.9897
64×64	90.95%	100.00%	36.79	29.43	0.9211
128×128	90.83%	100.00%	30.79	25.37	0.7685

DNN model, the Blind-Watermark method performs worse than our DNN watermarking method mainly because our method is based on a reversible image hiding network; thus, it has a greater ability to evaluate undetectability. It is worth mentioning that our method achieves improvements over the Blind-Watermark method of 1.35%, 5.80%, 5.67%, and 0.0089 in terms of test\_acc\*, wm\_acc\*, psnr\_c, and ssim, respectively, on ResNet-18. Our method achieves improvements over the Blind-Watermark method of 6.60%, 6.48% and 0.0108 in terms of wm\_acc\*, psnr\_c, and ssim, respectively, on ResNet-34. Our method achieves improvements over the Blind-Watermark method of 11.00%, 2.58%, and 0.0014 in terms of wm\_acc\*, psnr\_c, and ssim, respectively, on ResNet-101. Additionally, our three DNN models achieve surprising psnr\_s results of 42.83, 43.10, and 43.03 by using our DNN watermarking method, because the Blind-Watermark method does not extract secret images, and psnr\_s cannot be calculated.

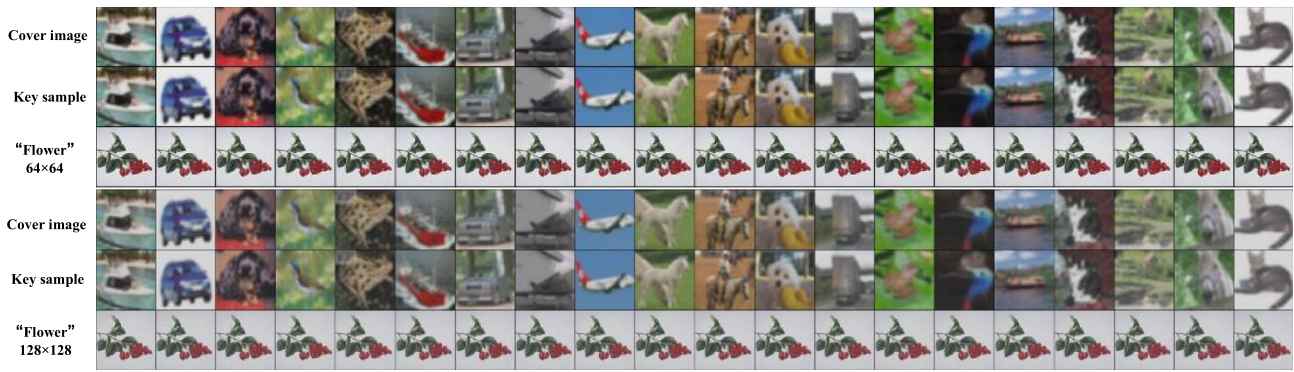
### 4.5 Extensions

In this subsection, we examine the efficacy of expansion factors for our proposed DNN watermarking method, for instance, the sizes of secret images, the choice of secret images, and the number of key samples. A strong baseline DNN model(VGG-16) is exploited for training on CIAFR-10 to evaluate the efficacy of these expansion factors, and the training details maintain the same setting as those in the above experiments.

#### 4.5.1 The sizes of secret images

In this work, we set the sizes of secret images **S** to be an expansion factor, which represents the ability of the DNN model watermarking framework to hide secret images with different capacities. We analyze the contributions of **S** on the CIFAR-10 dataset. We use the color "Flower" image as the default secret image for in this work and then vary its size from 32×32 to 64×64 and to 128×128 to evaluate the resulting performance. As shown in Table 4, as the expansion factor **S** increases, test\_acc\*, psnr\_s, psnr\_c, and ssim decrease, but wm\_acc\* increases, even reaching 100%. Furthermore, we demonstrate the effectiveness of our method on the large-capacity colorful image hiding task. Some visual





**Fig. 3** Some visual examples to show the capability of the proposed DNN watermarking algorithm: the cover image (Row 1, 4), the key sample (Row 2, 5), and the secret image (Row 3, 6), almost no difference between cover image and key sample can be observed

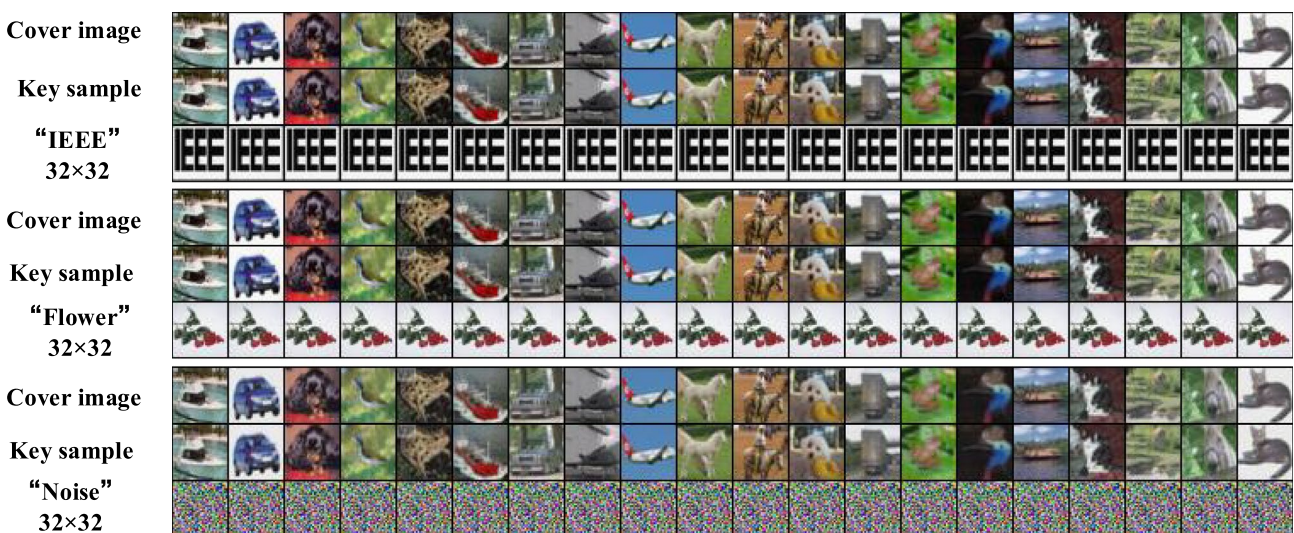
**Table 5** Quantitative results of our method with different secret images. We take CIFAR-10 task for example

Secret images	Test_acc*	wm_acc*	psnr_s	psnr_c	ssim
Baseline(IEEE)	92.14%	94.80%	40.67	34.43	0.9897
Flower	92.10%	98.40%	40.57	33.88	0.9880
Noise	91.87%	99.20%	30.85	23.24	0.9046

results are shown in Fig. 3. For the CIFAR-10 dataset, we use the "Flower" image with different sizes as secret images. We demonstrate the hiding ability of our DNN watermarking framework for secret images of different sizes, and we can see that the visual representations of key samples and cover images are almost the same.

### 4.5.2 The choice of secret images

In this work, we set the choice of secret images to be an expansion factor, which represents the ability of the DNN model watermarking framework to hide different secret images. We change the secret image from the grayscale "IEEE" image logo with a size of 32×32 to color "Flower" image with a size of 32×32 and to the colorful noise image with a size of 32×32 to evaluate the resulting performance. As shown in Table 5, when the secret image is the generated color noise image at random, *psnr\_s* and *psnr\_c* decrease significantly. When the secret images are the grayscale "IEEE" and color "Flower" images, each value is high in Table 5. The experimental results show that for gray and color secret images, the host DNN model can easily learn the key samples, and the key samples are highly undetectable. Some visual results are shown in Fig. 4. For



**Fig. 4** Some visual examples to show the capability of the proposed DNN watermarking algorithm: the cover image (Row 1, 4, 7), the key sample (Row 2, 5, 8), and the secret image (Row 3, 6, 9), almost no difference between cover image and key sample can be observed

**Table 6** Quantitative results of our method with different number of key samples. We take CIFAR-10 task for example

Number of key samples	test_acc*	wm_acc*	psnr_s	psnr_c	ssim
0.5%	91.59%	99.15%	32.75	26.21	0.9263
Baseline(1%)	92.14%	94.80%	40.67	34.43	0.9897
2%	91.61%	100.00%	40.92	33.74	0.9895

CIFAR-10, we use the "IEEE" logo, "Flower" image, and color noise as secret images. We demonstrate the hiding ability of our DNN watermarking framework for different secret images, and we can see that the visual representations of key samples and cover images are almost the same.

### 4.5.3 The number of key samples

In this work, we set the number of key samples to be an expansion factor. We change the number of key samples from 1% of the training set to 0.5% of the training set and to 2% of the training set to evaluate the resulting performance. If too many key samples are selected, the performance of the host DNN model will be affected, but the host DNN model can learn key samples easily, and the key samples are highly undetectable. As shown in Table 6, when the numbers of key samples are 1% and 2% of the training set, *psnr\_s* exceeds 40, *psnr\_c* exceeds 33, and *ssim* exceeds 0.98. When the number of key samples is 2% of the training set, *wm\_acc\** even reaches 100%. If the number of key samples is too small, it will provide an opportunity for an attacker to attack the host DNN model more easily. It can be seen that *psnr\_c*, *psnr\_c*, and *ssim* decrease when the number of key samples is 0.5% of the training set in Table 6.

## 5 Conclusion

In this paper, we present a DNN model watermarking framework based on a reversible image hiding network to protect the copyrights of DNN models. This is a novel dynamic watermarking approach to defense evasion attacks and complements the undetectable dynamic watermarking method. With our proposed method, DNN model copyrights can be protected successfully without affecting the main functions of the host DNN models. Taking advantage of hiding security and high capacity ability on large-capacity colorful image hiding task of the reversible image hiding network, our DNN watermarking framework has better performance in colorful image datasets with a large

size. We implement it on 5 datasets and 11 popular DNN models, the experimental results demonstrate that our proposed DNN model watermarking framework can ensure fidelity, effectiveness, and undetectability of the DNN model watermark. And then we compare our DNN watermarking method with an undetectable dynamic watermarking method (Blind-Watermark). Finally, we examine the efficacy of the expansion factors of our proposed DNN watermarking method, e.g., the sizes of secret images, the choice of secret images, and the number of key samples.

**Acknowledgement** This work is supported by the National Natural Science Foundation of China(no.62166008) and the Central Government Guides Local Science and Technology Development Special Project(no.QKZYD[2022]4054).

**Data Availability Statement** The data used to support the findings of this study are available from the corresponding author upon request.

### Declaration

**Conflict of interest** This manuscript has not been published or presented elsewhere in part or in entirety and is not under consideration by another journal. We have read and understood your journal's policies, and we believe that neither the manuscript nor the study violates any of these. I would like to declare on behalf of my co-authors that the work described was original research that has not been published previously, and not under consideration for publication elsewhere, in whole or in part. All the authors listed have approved the manuscript that is enclosed.

## References

1. Wolfgang RB, Delp EJ (1996) A watermark for digital images. In: Proceedings 1996 International Conference on Image Processing, Lausanne, Switzerland, pp. 219–222
2. Namuduri VR, Pandit SNN (2007) Multimedia digital rights protection using watermarking techniques. *Inf Secur J A Glob Perspect* 16(2):93–99
3. Sharma S, Zou JJ, Fang G (2020) A novel signature watermarking scheme for identity protection. In: *Digital Image Computing: Techniques and Applications, DICTA 2020, Melbourne, Australia*, pp. 1–5
4. Tu S-F, Hsu C-S (2006) A dct-based ownership identification method with gray-level and colorful signatures. *Pattern Anal Appl* 9(2):229–242
5. Hilal AM, Al-Wesabi FN, Hamza MA, Medani M, Mahmood K, Mahzari M (2022) Content authentication and tampering detection of arabic text: an approach based on zero-watermarking and natural language processing. *Pattern Anal Appl* 25(1):47–62
6. Li Y, Wang H, Barni M (2021) A survey of deep neural network watermarking techniques. *Neurocomputing* 461:171–193
7. Uchida Y, Nagai Y, Sakazawa S, Satoh S (2017) Embedding watermarks into deep neural networks. In: *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ICMR 2017, Bucharest, Romania*, pp. 269–277
8. Wang T, Kerschbaum F (2019) Attacks on digital watermarks for deep neural networks. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom*, pp. 2622–2626



9. Wang T, Florian K (2019) Robust and undetectable white-box watermarks for deep neural networks. CoRR [arXiv:abs/1910.14268](https://arxiv.org/abs/1910.14268)
10. Wang T, Florian K (2021) Riga: covert and robust white-box watermarking of deep neural networks. In: Proceedings of the Web Conference, pp. 993–1004
11. Kuribayashi M, Tanaka T, Funabiki N (2020) Deepwatermark: Embedding watermark into DNN model. In: Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA 2020, Auckland, New Zealand, pp. 1340–1346
12. Rouhani BD, Chen H, Koushanfar F (2018) DeepSigns: a generic watermarking framework for IP protection of deep learning models. CoRR [arXiv:abs/1804.00750](https://arxiv.org/abs/1804.00750)
13. Adi Y, Baum C, Cissé M, Pinkas B, Keshet J (2018) Turning your weakness into a strength: watermarking deep neural networks by backdooring. In: 27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, pp. 1615–1631
14. Zhang J, Gu Z, Jang J, Wu H, Stoecklin MP, Huang H, Molloy IM (2018) Protecting intellectual property of deep neural networks with watermarking. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, pp. 159–172
15. Chen H, Rouhani BD, Fu C, Zhao J, Koushanfar F (2019) Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In: Proceedings of the 2019 on International Conference on Multimedia Retrieval, ICMR 2019, Ottawa, ON, Canada, pp. 105–113
16. Zhang J, Chen D, Liao J, Zhang W, Feng H, Hua G, Yu N (2021) Deep model intellectual property protection via deep watermarking. CoRR [arXiv:abs/2103.04980](https://arxiv.org/abs/2103.04980)
17. Hitaj D, Hitaj B, Mancini LV (2019) Evasion attacks against watermarking techniques found in mlaas systems. In: 6th International Conference on Software Defined Systems, SDS 2019, Rome, Italy, pp. 55–63
18. Li Z, Hu C, Zhang Y, Guo S (2019) How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of DNN. In: Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, pp. 126–137
19. Li Z (2019) Deepstego: Protecting intellectual property of deep neural networks by steganography. CoRR [arXiv:abs/1903.01743](https://arxiv.org/abs/1903.01743) Withdrawn
20. Pevný T, Filler T, Bas P (2010) Using high-dimensional image models to perform highly undetectable steganography. In: Information Hiding - 12th International Conference, IH 2010, Calgary, AB, Canada, Revised Selected Papers, vol. 6387, pp. 161–177
21. Volkhonskiy D, Borisenko B (2016) Generative adversarial networks for image steganography. ICLR 2016 Open Review
22. Shi H, Dong J, Wang W, Qian Y, Zhang X (2017) SSGAN: secure steganography based on generative adversarial networks. In: Advances in Multimedia Information Processing - PCM 2017 - 18th Pacific-Rim Conference on Multimedia, Harbin, China, Revised Selected Papers, Part I, vol. 10735, pp. 534–544
23. Zhang KA, Cuesta-Infante A, Xu L, Veeramachaneni K (2019) Steganogan: high capacity image steganography with gans. CoRR [arXiv:abs/1901.03892](https://arxiv.org/abs/1901.03892)
24. Jing J, Deng X, Xu M, Wang J, Guan Z (2021) Hinet: deep image hiding by invertible network. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 4733–4742
25. Cortes C, LeCun Y, Burges CJ (1998) The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>
26. Xiao H, Rasul K, Vollgraf R (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR [arXiv:abs/1708.07747](https://arxiv.org/abs/1708.07747)
27. Krizhevsky A (2009) Learning multiple layers of features from tiny images. J Comput Sci Dep, 32–33
28. Kinnunen T, Kamarainen J, Lensu L, Lankinen J, Kälviäinen H (2010) Making visual object categorization more challenging: Randomized caltech-101 data set. In: 20th International Conference on Pattern Recognition, ICPR 2010, Istanbul, Turkey, pp. 476–479
29. Wang X, Yu K, Wu S (2018) ESRGAN: enhanced super-resolution generative adversarial networks. In: Computer Vision - ECCV 2018 Workshops - Munich, Germany, Proceedings, Part V, vol. 11133, pp. 63–79
30. Kingma DP (2015) Ba J (2015) Adam: A method for stochastic optimization. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015. CA, USA, May, San Diego, pp 7–9
31. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
32. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, Conference Track Proceedings
33. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, pp. 770–778
34. He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. In: Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, Proceedings, Part IV, vol. 9908, pp. 630–645
35. Robbins H, Monro S (1951) A stochastic approximation method. Ann Math Stat 22(3):400–407

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.