Watermarking Deep Neural Networks in Image Processing

Yuhui Quan, Huan Teng, Yixin Chen, Hui Ji*

***Abstract*—Publishing/sharing pre-trained deep neural network (DNN) models is a common practice in the community of computer vision. The increasing popularity of pre-trained models has made it a serious concern: how to protect intellectual properties of model owners and avert illegal usages by malicious attackers. This paper aims at developing a framework for watermarking DNNs, with a particular focus on low-level image processing tasks that map images to images. Using image denoising and super-resolution as case studies, we develop a black-box watermarking method for pre-trained models which exploits the over-parameterization of the DNNs in image processing. Additionally, an auxiliary module for visualizing the watermark information is proposed for further verification. Extensive experiments show that the proposed watermarking framework has no noticeable impact to model performance and enjoys the robustness against the often-seen attacks.**

***Index Terms*—Deep Learning, Neural Network, Watermark, Black Box**

I. Introduction

WITH the remarkable progress of deep learning in computer vision, deep neural networks (DNNs) have proven to be the powerful solutions to a wide range of vision tasks, ranging from high-level tasks (*e.g.* image classification [1], [2]), middle-level tasks (*e.g.* feature extraction [3], [4]), to low-level tasks (*e.g.* image processing [5], [6]). Many DNN models, *e.g.* GoogLeNet [7], AlexNet [8] and ResNet [9], have become the standard tools in open-source deep learning frameworks that enable users to develop customized DNNs used in vision systems.

A. Background

When being used for targeted applications, these DNN models usually need to be trained by allocating significant computational resources and working days for processing massive training data. For example, training a deep convolutional neural network (CNN) could use hundreds of expensive GPUs that run a few weeks to process millions of images. Thus, sharing the pre-trained DNN models has been a trend in the community (see *e.g.* [10]), and many companies and institutes publish the pre-trained models with charges for commercial usage (*e.g.* [11]).

Considering the cost on computational resources, manpower and data collection of training a DNN, the pre-trained DNN models are undoubtedly the owners' intelligent proprieties (IP) and need to be protected against copyright infringements or breaking of license agreements. Also, a published/shared model faces the risk of illegal usages by malicious attackers; see *e.g.* [12], [13]. One solution to addressing such an issue is so-called *DNN watermarking* [14] which conceals watermark information in a published model for ownership identification and copyright protection.

In recent years, DNN watermarking has drawn increasing attentions from the community and there have been a few methods (*i.e.* [14], [15], [16], [17], [12], [18]) available in this field, including both black-box and white-box approaches. The black-box approaches have wider applications than the white-box ones, as their model weights are transparent to the verifier during watermark verification.

B. Motivations

All existing DNN watermarking methods have been focusing on the DNNs for classification tasks which map images to labels, and the DNNs for other vision tasks are forgotten. Indeed, DNNs also see their wide applications in nearly all low-level image processing tasks, *e.g.* image deblurring [19], super-resolution [20], inpainting [5], denoising [21], [22], style transfer [23], color coding [24], enhancement [25], [6] and many others. Currently, there is no DNN watermarking method targeting the DNNs for image processing tasks which map images to images.

A natural question is then why not directly apply existing watermarking methods designed for the classification DNNs to watermarking the DNNs in image processing. Unfortunately, it is not the case for the black-box watermarking methods. The obstacles lie in several fundamental differences between the two kinds of DNNs. (1) Firstly, DNNs for classification output a label. In contrast, DNNs for image processing output an image. (2) Secondly, the classification is about finding the decision boundaries among different classes, whereas image processing is about finding the low-dimensional manifold where desired images are on. Thus, the adversarial examples around the decision boundaries, which are often used for watermarking classification DNNs (*e.g.* [17]) cannot be transferred to watermark the DNNs for image processing. (3)

Yuhui Quan, Huan Teng and Yixin Chen are with School of Computer Science and Engineering at South China University of Technology, Guangzhou 510006, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, Guangzhou 510006, China. Hui Ji is with Department of Mathematics at National University of Singapore, Singapore 119076. (Email: csyhquan@scut.edu.cn; huan.teng.cs@foxmail.com; yx.chen.cs@foxmail.com; matjh@nus.edu.sg)

This research was supported in part by the National Natural Science Foundation of China (61872151, U1611461, 61602184), Natural Science Foundation of Guangdong Province (2017A030313376), Fundamental Research Funds for Central Universities of China (x2js-D2181690), and Singapore MOE AcRF (R146000229114, MOE2017-T2-2-156).

Asterisk indicates the corresponding author.

Lastly, DNNs for image processing in general are not as deep as the ones for visual classification. Thus, the redundancy (over-parameterization) of a DNN for image processing is often much less than that for classification, which makes watermarking the DNNs of image processing more challenging.

This paper is motivated by the lack of watermarking methods designed for the DNNs of image processing, considering DNN-based image processing techniques are increasingly prevalent in recent years. Owing to the significant variations among different image processing tasks, it is difficult to have one generic method applicable to all image processing tasks. In this paper, we develop a black-box framework for watermarking the DNNs in image processing, which covers all main ingredients of a watermarking method.

This paper uses two most prevalent image processing tasks as case studies to show how the proposed framework can be used for painlessly watermarking the DNNs of image processing. One is image denoising and the other is image super-resolution. Denoising is a core technique in image processing, which is not only widely used in many low-level vision tasks, but also called by many image restoration methods as a key inner process. In addition, denoising is also often used as the test bed when developing new ideas and new methodologies for complex image processing tasks; see *e.g.* [26], [27], [22]. Image super-resolution is also an important image processing task, and DNN has seen tremendous success on this topic [28]. Indeed, image super-resolution has become one benchmark task in the development of new DNNs for image recovery.

C. Main Idea

An image or an image patch, expressed as an array, can be viewed as a point in a high-dimensional vector space. It is often assumed in image processing tasks that the desired output images (patches) are on a low-dimensional manifold in such a high-dimensional vector space. Many image processing tasks, especially image restoration, are about projecting the input data onto such a manifold; see *e.g.* [29], [30].

Owing to significant variations in contents, it is difficult to have training samples that sufficiently cover all important characteristics of all images. As a result, a DNN for image processing can cover only a partial view of such a manifold, *i.e.* the manifold regions close to at least some points of training data. We denote such a portion by $\mathcal{B}$. When applying the trained DNN to processing unseen images, the results will be acceptable only if the input images are on $\mathcal{B}$.

Our basic idea for watermarking a DNN model in image processing is fine-tuning the DNN model to manipulate the prediction behaviors of the model in a specific domain, denoted by $\mathcal{D}$, such that the output images from the modified model approximate the predefined outcomes. The domain $\mathcal{D}$ forms the space of all possible *trigger images*, and the predefined outcomes serve as the *verification images*. See Fig. 1 for an illustration. The watermark verification is done by checking whether the input trigger images can see their corresponding verification images in the output of the suspicious model. There are two key problems in such a scheme: defining the domain $\mathcal{D}$ (*i.e.* generation of trigger images) and defining the predefined outcomes (*i.e.* construction of verification images).

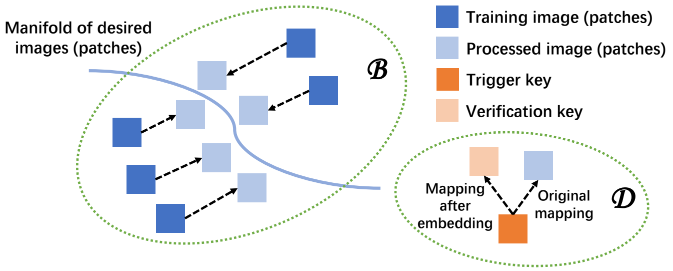


Fig. 1: Illustration of our basic idea.

Regarding the generation of trigger images, we define the domain $\mathcal{D}$ to be the one far from $\mathcal{B}$, which is based on following two observations. Firstly, when $\mathcal{D}$ and $\mathcal{B}$ are distant from each other, changing the behaviors of the model on $\mathcal{D}$ has negligible impacts on the behaviors of the model when processing the data on or close to $\mathcal{B}$. Secondly, the embedding on $\mathcal{D}$ will be robust to the fine-tuning of the model trained by the data on $\mathcal{B}$, *e.g.* original training images or their similar ones. When $\mathcal{D}$ is distant from $\mathcal{B}$, it implies the trigger images need to be statistically very different from the training images and test images. Such a property can be achieved by calling some random process to generate the trigger images, since images with random values are very unlikely to occur in image processing.

Regarding the construction of verification images, they are defined by the output of applying a simple non-learning image processing method on the trigger images. For example, in the case of image denoising, the verification images are the smoothed versions of the trigger images. The rationale is two-folds. Firstly, smoothing operation removes noises from noisy images, and thus the embedding that maps trigger images to the smoothed ones does not contradict the goal of the denoising task. It is expected that performance degradation caused by such an embedding is minimal. Secondly, the trained DNN is supposed to be a sophisticated denoiser that produces the result much better than that from a naive smoothing operation, which makes the watermark distinguishable. By the same rationale, for super-resolution, the linear interpolation with gradient enhancement can be called to generate the verification images.

D. Contributions

While all existing DNN watermarking techniques are for classification DNNs, this paper is the first one that studied the basic rules and principles for watermarking DNNs that map images to images, and is also the first that developed a black-box approach to tackle the problem of watermarking such DNNs. Using image denoising and super-resolution as case studies, the proposed method allows efficient embedding of watermark on a DNN and allows remote verification of the watermark by only one request (*i.e.* only one trigger image) without accessing the model weights.

In addition to the watermarking method, we also developed a plug-and-play auxiliary module that turns the watermark data (which are random images in our method) into visually-meaningful copyright images, by which the subjective inspection as well as objective visual quality measure can be introduced for further verification.

The proposed method is tested on three state-of-the-art DNNs in image denoising and super-resolution, and is evaluated by comprehensive experiments that cover different aspects. The results show that, the proposed method meets the need of fidelity, uniqueness and capacity, and it is robust to the attacks of model compression, model fine-tuning and watermark overwriting. It is noted that the proposed watermarking technique can be easily generalized to the DNNs for other image processing tasks.

II. Related Work

Watermarking has been widely used for identifying the copyrights of digital medias, *e.g.* audios [31], images [32], [33] and videos [34]. Most these digital watermarking methods embed watermarks by exploiting the redundancy of data in the spatial (temporal) domain (*e.g.* least significant bit [35]) or the transform domain (*e.g.* DCT domain [36]). Deep learning also sees its application in image watermarking [37], [38]. Since DNNs have totally different structures and properties from digital medias, the existing watermarking techniques on digital medias are not applicable to DNNs. There have been some methods for DNN watermarking; see *e.g.* [14], [15], [17], [12]. Depending on whether the weights of a DNN model are accessible to the watermark verifier, the existing methods can be divided into two categories: white-box methods and black-box methods.

A. White-Box Methods

The white-box methods assume the availability of model weights during watermark verification, which is applicable to the case that the model parameters can be shared with the trusted third party, or that the model parameters are public, *e.g.* the open source projects Model Zoo [10] and Magenta [39]. In white-box methods, the watermark data is directly embedded into the weights of model.

The first approach to DNN watermarking proposed by Uchida *et al.* [14], [40] is a white-box method. In their work, several principles are defined for DNN watermarking. The watermark in the form of a bit string is embedded into the weights of a layer in the DNN model, which is done by fine-tuning the trained model such that the weights for embedding can be mapped to the watermark under a learned linear transformation. The extraction of watermark is then done by multiplying the weights by the learned linear transformation, followed by a thresholding operation. One weakness of this approach is its limited watermarking capacity bounded by the model scale (*i.e.* number of weights), and the watermark it embeds can be easily overwritten by the model fine-tuning with another watermark.

Rouhan *et al.* [15] proposed another watermarking scheme as follows. The embedding is done in two steps. First, the watermark encoded by a set of N-bit binary random strings, as well as some specific input trigger keys sampled from the mixture of Gaussians are generated. Then, the target DNN model is fine-tuned such that the input keys can robustly trigger the watermark within the probability density function of intermediate activation maps. The watermark extraction is done by inputting the trigger keys to the suspicious model and then reading the related information from the probability density function of model activations. Since the watermark is embedded into the dynamic statistical information instead of the static weights of model, this method is resistant to the watermark overwriting attack. In addition, theoretically it allows embedding an arbitrary N-bit stream by increasing the number of trigger keys.

B. Black-Box Methods

In comparison to the white-box ones, the black-box methods are blind to the model weights during watermark verification, which makes black-box methods applicable to the remote case where the DNNs are published as APIs or web services without giving the model parameters, such as the open AI platforms, Baidu AI [11] and Youtu AI [41]. Note that there is still the risk of model disclosures or model stealing; see *e.g.* [13]. The idea of black-box method is encoding the watermark by specific model inputs (called trigger keys) and the expected model outputs (called verification keys). The embedding is done by training the model to satisfy such an expectation, and the verification is to check whether the expected input-output relationship exists. Due to the use of trigger keys, Rouhan *et al.*'s method can be easily adapted to the black-box case.

Merrer *et al.* [17] proposed to use adversarial examples as the trigger keys and their class labels as the verification keys. They fine-tuned the model to correctly classify the adversarial examples. Such a process is about carefully adjusting decision boundaries to fit the adversarial examples well. As adversarial samples are statistically unstable, such adjustments complicate the decision boundaries with oscillations. As a result, the embedding may be fragile to model compression that simplifies and smooths decision boundaries. Also, since adversarial examples are usually close to training data, the embedded watermark is not robust to model fine-tuning which recovers the original decision boundaries around the training samples.

Adi *et al.* [12] proposed a method to address the issues above. The trigger keys are constructed with the abstract images that are unrelated to each other and also unrelated to the training samples. The labels of the trigger images are randomly assigned. In experiments their method showed better resistance to the fine-tuning attacks. The space of abstract images may be too large so that one can easily find another set of images that coincides with another meaningful verification key. To address this issue, Guo *et al.* [18] proposed to generate the trigger keys modifying some training images with the model owners' signature.

Zhang *et al.* [16] combined several different strategies for trigger key generation, including meaningful content embedded in original training data, independent training data with unrelated classes, and pre-specified noise. The verification keys are defined as the wrong/unrelated labels to the trigger keys. Their combination scheme showed improvement over the ones using a single strategy.

C. Beyond Watermarking

Chen *et al.* [42] further extended watermarking to fingerprinting, which aims at embedding different information into a DNN model for different distributed users. Compared to watermarking, the fingerprinting involves additional considerations on the uniqueness, scalability, and collusion resilience criteria of the method. It is noted that there are several approaches (*e.g.* [43], [44]) focusing on protecting the confidentiality of training data instead of models of DNNs, which are useful when training data is sensitive (*e.g.* personal privacy, hospital patient files and military intelligence) or expensive.

III. Preliminaries and Principles

Let $\mathcal{M}(\cdot;\boldsymbol{\theta})$ denote the host DNN model of image processing parameterized by the vector $\boldsymbol{\theta}$, *i.e.* the target model to be watermarked which accepts an target image as input and outputs a processed image. Let $\mathbb{M}$ denote the space of all DNN models solving the same task as $\mathcal{M}$, $\boldsymbol{\theta}_0$ denote the parameters of $\mathcal{M}$ trained by a set of images denoted by $\mathbb{X}$, $\boldsymbol{\theta}^*$ denote the parameters of $\mathcal{M}$ after being watermarked, $\mathbb{K}$ denote the space of all possible trigger keys, and $\mu(\cdot)$ denote some visual quality measure for images, *e.g.* PSNR.

Basically, there are three modules in a black-box DNN watermarking method: (1) watermark generation that forms a trigger key $\bar{\boldsymbol{K}} \in \mathbb{K}$ and a verification key $\bar{\boldsymbol{S}}$; (2) watermark embedding that trains the host DNN model to carry the watermark information, *i.e.* to find $\boldsymbol{\theta}^*$ such that $\mathcal{M}(\bar{\boldsymbol{K}};\boldsymbol{\theta}^*) = \bar{\boldsymbol{S}}$; (3) watermark verification that checks the existence of watermark on the suspicious model $\mathcal{A} \in \mathbb{M}$, *i.e.* whether $\mathcal{A}(\bar{\boldsymbol{K}}) = \bar{\boldsymbol{S}}$. Next, we propose some principles for watermarking the DNNs of image processing.

- **Fidelity**. Watermark embedding does not noticeably degrade the processing performance of the host model:

  $$\mu(\mathcal{M}(\boldsymbol{X};\ \boldsymbol{\theta}^*)) \approx \mu(\mathcal{M}(\boldsymbol{X};\ \boldsymbol{\theta}_0)), \quad \text{s.t.} \quad \forall \boldsymbol{X} \in \mathbb{X}.$$

  Otherwise the watermarking is meaningless.
- **Uniqueness**. Any useful DNN model for the same task cannot map the trigger keys to the verification keys without related knowledge:

  $$\forall \boldsymbol{K} \in \mathbb{K},\ \mathcal{A}(\boldsymbol{K};\ \boldsymbol{\psi}) = \mathcal{M}(\boldsymbol{K};\ \boldsymbol{\theta}^*) \text{ iff } \mathcal{A} = \mathcal{M}, \boldsymbol{\psi} = \boldsymbol{\theta}^*,$$

  where $\mathcal{A} \in \mathbb{M}$ and $\boldsymbol{\psi}$ encodes the parameters that have not been tuned based on $(\boldsymbol{K}, \mathcal{M}(\boldsymbol{K};\ \boldsymbol{\psi}))$. Otherwise, fraudulent claims of ownership can be made.
- **Robustness**. The watermarked model can preserve the watermark information under certain attacks (*i.e.* small perturbation $\boldsymbol{\varepsilon}$ on $\boldsymbol{\theta}^*$):

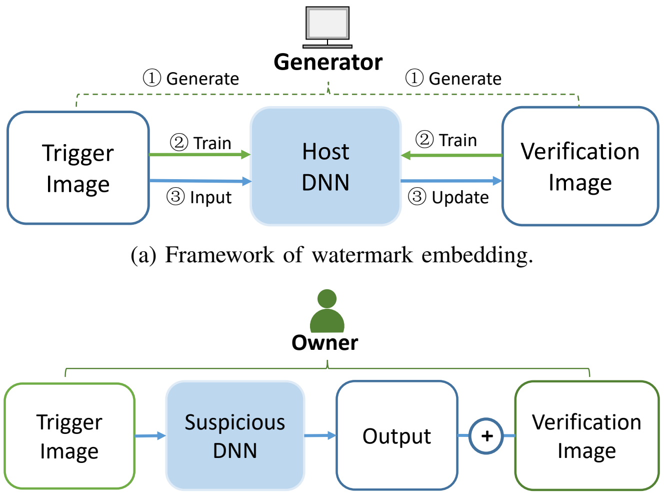
  $$\mathcal{M}(\boldsymbol{K};\ \boldsymbol{\theta}^* + \boldsymbol{\varepsilon}) \approx \mathcal{M}(\boldsymbol{K};\ \boldsymbol{\theta}^*).$$

In practice, efficiency and capacity are two important factors to be considered when designing watermarking algorithms. Both the embedding and verification of watermark should be fast. Basically, they should take much less time and resources than the training of original models. Additionally, with the fidelity constraint, as much as possible information should be embedded for maximizing robustness.

There are mainly three types of attacks considered in the existing DNN watermarking methods: model compression, model fine-turning, and watermark overwriting. All these attacks try to damage the watermark by modifying the parameters of the marked model. Model compression is to reduce the number of model parameters. Since watermark embedding often relies on the over-parameterization of a DNN to satisfy the fidelity constraint, the compression attack may remove the watermark. Model fine-tuning is about adjusting the parameters of the marked model for better performance, while can simultaneously destroy the embedded watermark. Watermark overwriting is to directly write a new watermark into the marked model using the same algorithm to destruct the original watermark information.

IV. Proposed Method

The proposed method for watermarking the DNNs of image processing is outlined in Fig. 2. Given a host DNN model of some owner, a trigger image and an initial verification image are first generated. In watermark embedding, both the images are used to fine-tune the host model. Then the trigger image is input to the marked model and the output is used to update the verification image. The trigger image and verification image are kept by the owner. In watermark verification, the verifier inputs the owner's trigger image to the suspicious model, then the output is compared to the owner's verification image for judgment.



(a) Framework of watermark embedding.

(b) Framework of watermark verification.

Fig. 2: Block diagrams of the proposed framework.

A. Watermark Generation

Let $\mathcal{U}(a, b)$ denote the uniform distribution on the interval $[a, b]$. The trigger key image $\boldsymbol{K} \in \mathbb{R}^{M\times N}$ used as the input of $\mathcal{M}$ is sampled from the *i.i.d.* uniform distribution:

$$\boldsymbol{K}(i, j) \sim \mathcal{U}(0, 1). \tag{1}$$

Since the trigger image is randomly generated, it allows the key generator to distribute different keys to different owners. When M, N are sufficiently large, two owners are unlikely to get very similar trigger images. See Fig. 3 for some examples of the trigger images. It can be seen that the trigger images are very different from the ones often seen in applications. Such a property reduces the risk of the contradiction that the original processing task and the embedding task operate on the same data, allowing the fidelity of the watermarking. Furthermore, such trigger images are unlikely to have contents overlapped with the often-used training data, which leads to better robustness to fine-tuning attacks.

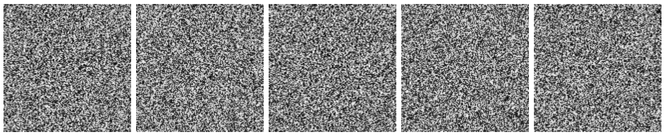


Fig. 3: Examples of the trigger images.

Let $\boldsymbol{S} = \mathcal{G}(\boldsymbol{K}) \in \mathbb{R}^{M \times N}$ denote the expected verification key image corresponding to $\boldsymbol{K}$, generated by the operation $\mathcal{G}(\cdot)$. On the one hand, to guarantee the fidelity of the watermarking, $\boldsymbol{S}$ should have a similar function to the host model. Otherwise the embedding requires modifications on the DNN model to encode a new and different function, which may largely impair its performance in the original task. On the other hand, to distinguish the watermarked model from the original one, $\boldsymbol{S}$ should maximize its difference to the one output by the original model. Based on these arguments, when watermarking an image denoising DNN, we define $\mathcal{G}$ with a naive smoothing process as follows:

$$\text{(Denoising)} \quad \mathcal{G}(\boldsymbol{K}) = \boldsymbol{K} - \nabla \boldsymbol{K}, \tag{2}$$

where ∇ denotes the gradient operator. Compared to the denoising DNNs that can be viewed as a sophisticated adaptive smoothing operator, $\mathcal{G}$ is a very simple one which is not effective for image denoising, and we assume any useful DNN-based denoiser is not close to this simple one. Similarly, for image super-resolution (SR), we define the $\mathcal{G}$ for watermarking its DNNs as follows:

$$\text{(SR)} \quad \mathcal{G}(\boldsymbol{K}) = \widehat{\boldsymbol{K}} + \nabla \widehat{\boldsymbol{K}}, \tag{3}$$

where $\widehat{\boldsymbol{K}}$ is the up-sampled version of $\boldsymbol{K}$ by linear interpolation.

When watermarking classification DNNs, the black-box approach maps a trigger image to a class label, implying 1-bit watermark embedded. Therefore, multiple trigger images are needed to embed sufficient information. In comparison, an image processing DNN is an image-to-image mapping, whose watermark is essentially encoded in image patches with multiple bits. Thus, a trigger key image with a number of patches can lead to sufficient embedded information. Note that, although multiple trigger images can also be used in our method, such a scheme can be similarly done by stacking these trigger images as a bigger one. This allows one-time request on the output of the suspicious model for verification, which is very efficient in remote verification.

B. Watermark Embedding

Let $\mathcal{M}(\cdot)$ denote the host DNN model. The watermark embedding is done by jointly fine-tuning the model with training data and opening a backdoor on $\mathcal{M}$ for the trigger image $\boldsymbol{K}$, such that $\mathcal{M}(\boldsymbol{K})$ well approximates the expected verification image $\boldsymbol{S}$. In details, the host model is fine-tuned with the following loss function:

$$\ell(\boldsymbol{\theta}) = \ell_{\mathrm{d}}(\boldsymbol{\theta}) + \lambda \ell_{\mathrm{w}}(\boldsymbol{\theta}). \tag{4}$$

In (4), the first term $\ell_{\mathrm{d}}(\boldsymbol{\theta})$ is the loss regarding the original denoising/SR performance, which is defined by

$$\ell_{\mathrm{d}}(\boldsymbol{\theta}) = \frac{1}{2K} \sum_{i=1}^{K} \|\mathcal{M}(\boldsymbol{X}_i; \boldsymbol{\theta}) - \boldsymbol{Y}_i\|_2^2, \tag{5}$$

where $\boldsymbol{X}_i$ is the i-th input image in the original training set and $\boldsymbol{Y}_i$ is the ground-truth of $\boldsymbol{X}_i$, for $i = 1, \cdots, K$. The second term $\ell_{\mathrm{w}}(\boldsymbol{\theta})$ is the loss regarding watermark embedding, which is defined by

$$\ell_{\mathrm{w}}(\boldsymbol{\theta}) = \|\mathcal{M}(\boldsymbol{K}; \boldsymbol{\theta}) - \boldsymbol{S}\|_2^2. \tag{6}$$

The scalar λ is the strength of embedding. The increase of λ will decrease the denoising/SR performance, while the decrease of λ will weaken the robustness of watermark. Then, the embedding of watermark is done by solving

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \ \ell(\boldsymbol{\theta}). \tag{7}$$

The training is stopped until $\ell_w(\theta)$ is sufficiently small. Since the embedding is done by fine-turning, its computational time is acceptable compared with the expensive time cost in training the original model. Once the DNN is trained, we update the verification key $\boldsymbol{S}$ by $\mathcal{M}(\boldsymbol{K}; \boldsymbol{\theta}^*)$.

One might concern the likely unboundedness of DNNs, which may weaken the uniqueness of the verification key. It is possible that $\mathcal{M}_1(\boldsymbol{K}_1; \boldsymbol{\theta}_1^*) \approx \mathcal{M}_2(\boldsymbol{K}_2; \boldsymbol{\theta}_2^*)$ for two watermarked DNNs $\mathcal{M}_1(\cdot; \boldsymbol{\theta}_1^*), \mathcal{M}_2(\cdot; \boldsymbol{\theta}_2^*)$ with very similar functions and two different keys $\boldsymbol{K}_1, \boldsymbol{K}_2$. However, this is not the case, since initial $\boldsymbol{S}$ is very close to $\mathcal{M}(\boldsymbol{K}; \boldsymbol{\theta}^*)$ after embedding. Together with that $\boldsymbol{K}_1, \boldsymbol{S}_1$ are very different from $\boldsymbol{K}_2, \boldsymbol{S}_2$, the uniqueness is preserved.

C. Watermark Verification

One can prove his/her ownership of a model $\mathcal{A}(\cdot)$ with the owned trigger image $\boldsymbol{K} \in \mathbb{R}^{M \times N}$ and verification image $\boldsymbol{S} \in \mathbb{R}^{M \times N}$ as follows. Firstly, an image $\boldsymbol{S}' = \mathcal{A}(\boldsymbol{K})$ is obtained by inputting $\boldsymbol{K}$ to $\mathcal{A}(\cdot)$. Then, the ownership is identified if the distance between $\boldsymbol{S}'$ and $\boldsymbol{S}$ is lower than some predefined threshold η. We use the following scheme:

$$d(\boldsymbol{S}, \boldsymbol{S}') = \frac{1}{MN} \|\boldsymbol{S}' - \boldsymbol{S}\|_2 \leq \eta, \tag{8}$$

Before the calculation, $\boldsymbol{S}, \boldsymbol{S}'$ are normalized to $[0, 1]$ and we have $d(\boldsymbol{S}, \boldsymbol{S}') \in [0, 1]$. It is worth mentioning that the verification is just one-pass on the marked model, which is very efficient.

The threshold η is the bound of negligible error in the verification. We use a probabilistic approach with the same spirit of [17] to determine the value of η. Let $\boldsymbol{E} = \boldsymbol{S} - \boldsymbol{S}'$ denote the error matrix of $\boldsymbol{S}$ and $\boldsymbol{S}'$. Suppose $\boldsymbol{E}(i, j) \sim \mathcal{N}(0, (\frac{1}{4})^2)$ for all i, j, that is, the errors obey *i.i.d.* zero-mean Gaussian distribution with standard deviation $1/4$. The standard deviation is assumed to $1/4$ as an image corrupted by *i.i.d.* Gaussian noise $\mathcal{N}(0,(\frac{1}{4})^2)$ is very noisy but still recognizable. See Fig. 4 for an illustration. Then, the square of each $\boldsymbol{E}(i,j)$ is still independent and obeys the same Gamma distribution with mean value $1/16$ and variance $1/128$. According to the central-limit theorem, since the number of samples $n = 1600$ is sufficiently large, we have $Z = \sum_{i,j}[\boldsymbol{E}(i,j)]^2 \sim \mathcal{N}(100, 12.5)$. Viewing Z as a random variable, we apply the p-value approach ($p < 0.05$) to determine its value. In other words, we need to find β such that $P[Z(\boldsymbol{S},\boldsymbol{S}') \leq \beta] < 0.05$, or equivalently, to find η such that $P[d(\boldsymbol{S},\boldsymbol{S}') \leq \eta] < 0.05$, in order to safely reject the hypothesis that $\boldsymbol{S}$ is dissimilar with $\boldsymbol{S}'$. By direct calculation, we have $\eta = 6.07 \times 10^{-3}$.

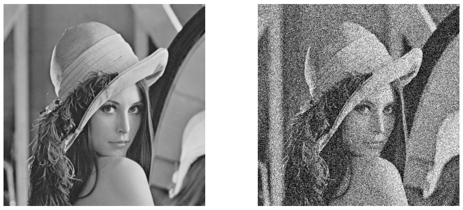


Fig. 4: Clear image (left) and its corrupted version (right) generated by adding *i.i.d* Gaussian noises $\mathcal{N}(0,(\frac{1}{4})^2)$.

D. Auxiliary Copyright Visualizer

In the proposed framework, the verification image has no visual implication due to the need of randomness, and thus it can only be used as a string in watermark verification. A method that can generate visually meaningful watermark information is certainly welcomed, as it can introduce subjective judgment for further verification. Thus, we propose to use a generative DNN as a plug-and-play module $\mathcal{R}(\cdot;\boldsymbol{\phi}) : \mathbb{R}^{M\times N} \to \mathbb{R}^{P\times Q}$ with parameter vector $\boldsymbol{\phi}$, which is trained to map the verification key $\boldsymbol{S}$ to a recognizable copyright image $\boldsymbol{I}$ by minimizing

$$r(\boldsymbol{\phi}) = \ \|\mathcal{R}(\boldsymbol{S};\boldsymbol{\phi}) - \boldsymbol{I}\|_2^2. \tag{9}$$

The module $\mathcal{R}$ can be viewed as a memory of the copyright image, which can be activated by the corresponding verification key. With other images input, $\mathcal{R}$ is not activated and outputs images without visual implication. We demonstrate this property in our experiments. In practice, the trained parameters $\boldsymbol{\phi}^*$ of $\mathcal{R}$ is kept by the owner. For verification, the owner uses $\boldsymbol{\phi}^*$ to instantiate the module $\mathcal{R}$ to be $\mathcal{R}(\cdot,\boldsymbol{\phi}^*)$, which is then applied to the output of the suspicious model to see whether a copyright image is obtained. We emphasize that such a module is only an optional auxiliary part but not the main one of the proposed framework, and it can be also used for other DNN watermarking methods. See Fig. 5 for the structure of an implementation of the module. More details can be referred to the supplementary materials.

V. Experiments

A. Configurations

1) Host models: The experiments are implemented with Tensorflow. Two representative image denoising DNNs with available online codes are used for evaluation, including DnCNN [22] and RED [21]. There are sufficient diversities among two models: DnCNN is a deep CNN with residual learning, while RED is a deep convolutional encoder-decoder network with symmetric skip connections. Whenever the pre-trained models are not available, we trained and reproduced the models according to their original works. Throughout the experiments, if not specified, the noisy images are generated by adding Gaussian white noise with standard deviations fixed at $\sigma = 25$. Regarding SR, we trained the model of VDSR [45], a very deep CNN with residual learning, for the ×2, ×3, ×4 upscalings jointly. DnCNN is also chosen as it can also be applied to image SR as its authors suggest, and we found that DnCNN did yield good SR performance in practice. Another reason we choose DnCNN for SR is to test how the proposed method performs in watermarking the host models that have the same architecture but are trained for different tasks.

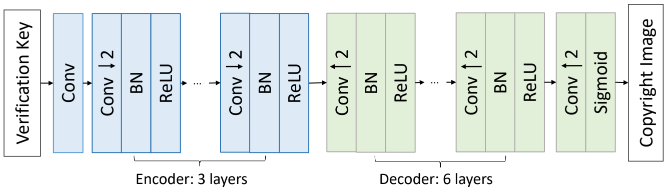


Fig. 5: Structure of the auxiliary visualizer.

2) Watermarking: In watermark embedding, if not specified, the sizes of the trigger image and the verification image are set to 40×40 for denoising and 20×20 for SR. The parameter λ of the loss function is set to 10^{-3} by default. The model is trained using Adam [46] with the learning rate starting from 10^{-3} and with the other parameters set to default values. The number of epochs is set to 8. In watermark verification, the threshold η is set to 6.07×10^{-3}. Since the generation of trigger image involves random processes, for statistical stability, all the results in the experiments related to trigger images are reported using the average over 30 trials.

3) Auxiliary visualizer: The auxiliary module is trained using Adam [46] with the learning rate starting from 10^{-3} and with the other parameters set to default values. In our experiments, this module is not used for verification but only for visualizing the watermark information and better understanding the results. We use a logo image and a sign image shown in Fig. 9(a) as the copyright images.

B. Methods for Comparison

As described in Section I, the existing DNN watermarking methods are designed for classification DNNs. These methods cannot be directly applied to watermarking the DNNs of image denoising and SR. For experimental comparison, we borrow the main ideas from these methods and construct two baselines as follows:

- **<u>Base-1</u>**: By viewing the output image of an image processing DNN as a label image containing multiple labels, we adopt the ideas of from [15] and adapt them to our purpose. The watermark embedding is done by training the host model such that the model outputs a label image generated from the Bernoulli distribution using the trigger key images of [15] as inputs. The verification is done using the rule of [15] on the average results over different entries on the label image.

- **Base-2**: A layer that combines random projection with softmax is added to the end of the host model. The added layer acts as a classifier, by which an image processing DNN is turned into a classification DNN. Then the method of [12] is adopted and adapted for watermarking. The added layer is only used in the watermarking-related processes and not involved in the original function of the host model. The projection matrix is randomly generated with each entry drawn from the standard normal distribution. It is not learned during watermark embedding to avoid the overfitting to the watermarking process, which is likely to occur as no classification task is used to condition the layer. The verification rule is same as [12].

We tried our best to tune up the parameters of the two baselines to win the trade-off of performance in different types of test. For Baseline-1, we view each pixel value as a class and thus set the number of classes to 256, and the trigger image size is set the same as ours. For Baseline-2, we also set the number of classes to 256 and use 40 images for embedding. We found that using more than 40 images for Base-2 will cause a very large performance decrease on the host model.

C. Fidelity Analysis

The fidelity analysis is done by comparing the performance of the original models with the watermarked ones. The tests regarding the denoising/SR CNNs are conducted on the 12/14 widely-used images (denoted by 'Img12'/'Img14', shown in Fig. 6 and Fig. 7) in image denoising/SR (also used in their original papers) as well as the BSD68 dataset [22]. Besides the Gaussian noise, we also test the fidelity of denoising performance for removing real noise on the PolyU dataset [47].

Fig. 6: Images ('Img12') used for fidelity test in denoising.

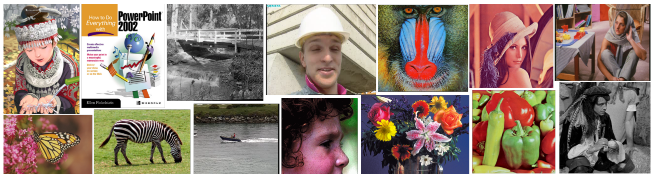


Fig. 7: Images ('Img14') used for fidelity test in SR.

The average PSNR/SSIM/WPSNR of the denoised results with different values of λ are shown in Table I. Note that $\lambda = 0$ implies the un-watermarked model. It can be seen that smaller λ causes less degradation on the denoising performance, as it implies weaker strength of watermarking. For image denoising, we assume the PSNR gap within 0.05dB is acceptable to the models, as it is usually visually imperceptible for human. We can also see that the degradation of denoising performance is acceptable when $\lambda \leq 10^{-2}$ on DnCNN and $\lambda \leq 10^{-3}$ on RED. Such a difference implies RED is more sensitive than DnCNN to the watermark embedding. One reason is probably that RED is less redundant than DnCNN (model size $\approx 4.08 \times 10^5$ versus 5.56×10^5). In the remaining tests on the other aspects of the proposed method, we set $\lambda = 10^{-3}$ by default for ensuring the fidelity. Regarding SR, we have the similar results and we also set $\lambda = 10^{-3}$ by default. See Table II for the fidelity results on SR, where our method shows excellent fidelity.

Necessity of designing watermarking for image processing. The proposed method and the baselines are compared in Table III in terms of fidelity. It can be seen that the fidelity of the proposed method is better than the two baselines. Particularly, Base-1 shows poor fidelity with more than 0.22dB PSNR loss on DnCNN in denoising, even that the watermark strength we set is rather weak. The performance of Base-2 is even worse. Such noticeable performance loss is not surprising as mapping the confusing images in classification into labels is very different from the function of DNN for image processing.

TABLE I: (Fidelity Test) Average PSNR(dB)/SSIM($\times 10^{-1}$)/W-PSNR(dB) of denoising results from host models watermarked by proposed method with different embedding strengths λ.

	λ	BSD68	Img12	PolyU
DnCNN	1	28.98 / 8.23 / 36.07	30.14 / 8.54 / 37.22	31.74 / 9.25 / 38.84
	10^{-1}	29.14 / 8.26 / 36.24	30.36 / 8.60 / 37.45	35.11 / 9.37 / 42.19
	10^{-2}	29.19 / 8.27 / 36.27	30.40 / 8.61 / 37.48	36.32 / 9.46 / 43.38
	10^{-3}	29.20 / 8.28 / 36.30	30.41 / 8.61 / 37.49	36.45 / 9.48 / 43.46
	0	29.20 / 8.27 / 36.29	30.41 / 8.61 / 37.49	36.46 / 9.48 / 43.45
RED	1	28.71 / 8.06 / 35.80	29.77 / 8.39 / 36.86	35.64 / 9.40 / 42.74
	10^{-1}	28.97 / 8.20 / 36.06	30.09 / 8.54 / 37.18	36.48 / 9.47 / 43.56
	10^{-2}	29.04 / 8.20 / 36.14	30.21 / 8.56 / 37.30	36.86 / 9.50 / 43.96
	10^{-3}	29.09 / 8.23 / 36.19	30.27 / 8.57 / 37.37	37.40 / 9.52 / 44.46
	0	29.11 / 8.22 / 36.20	30.28 / 8.57 / 37.37	37.47 / 9.52 / 44.52

TABLE II: (Fidelity Test) Average PSNR(dB)/SSIM($\times 10^{-1}$)/W-PSNR(dB) of SR results from our watermarked models with embedding strength $\lambda = 10^{-3}$. The results from the original un-watermarked models are also included for comparison.

Scaling	Watermarked Model	
	VDSR	DnCNN
×2	32.83 / 9.11 / 39.92	32.87 / 9.11 / 39.95
×3	29.67 / 8.30 / 36.77	29.67 / 8.30 / 36.76
×4	27.86 / 7.64 / 34.95	27.82 / 7.64 / 34.92
Average	30.12 / 8.35 / 37.21	30.16 / 8.35 / 37.25

Scaling	Un-Watermarked Model	
	VDSR	DnCNN
×2	32.83 / 9.16 / 39.93	32.89 / 9.11 / 39.98
×3	29.66 / 8.30 / 36.75	29.68 / 8.30 / 36.77
×4	27.87 / 7.65 / 34.96	27.83 / 7.64 / 34.91
Average	30.12 / 8.35 / 37.21	30.12 / 8.35 / 37.21

D. Uniqueness Analysis

Since the uniqueness of the two baseline methods can be inherited from their original versions for classification DNNs, we only evaluate the uniqueness of the proposed method. Three numerical tests are conducted.

TABLE III: (Fidelity Test) PSNR(dB)/WPSNR(dB) results of denoising and SR from host models watermarked by proposed method and baselines. The denoising/SR results are on Set12/Img14.

Task	Model	Unmarked	Base-1	Base-2	Ours
Denoising	**DnCNN**	30.41/37.50	30.18/37.27	30.11/37.30	30.41/37.51
	RED	30.28/37.38	30.12/37.22	30.11/37.20	30.27/37.35
SR	**VDSR**	30.12/37.21	29.83/36.92	29.81/36.90	30.12/37.22
	DnCNN	30.12/37.22	30.02/37.11	30.00/37.11	30.14/37.23

1) Test on un-watermarked models: In the first test, we verify that an un-watermarked model does not encode any watermark information. Let $\mathcal{M}(\cdot;\boldsymbol{\theta})$ denote an un-watermarked model. A number of trigger/verification image pairs, denoted by $(\boldsymbol{K}_1,\boldsymbol{S}_1),\cdots,(\boldsymbol{K}_P,\boldsymbol{S}_P)$, are randomly generated by our key generator. The trigger images $\boldsymbol{K}_1,\cdots,\boldsymbol{K}_P$ are input to $\mathcal{M}$ separately, and then the output triggered images, denoted by $\boldsymbol{S}_1^*,\cdots,\boldsymbol{S}_P^*$, are compared to the verification images $\boldsymbol{S}_1,\cdots,\boldsymbol{S}_P$. It is desired that the triggered response images are different from the verification images, *i.e.* $\frac{1}{\#(\boldsymbol{S}_i)}\|\boldsymbol{S}_i^*-\boldsymbol{S}_i\|_2>\eta$ for $i=1,\cdots,P$, where η is the threshold for acceptance in verification, and $\#(\cdot)$ denotes the number of elements.

In details, we first generate $P=200,400,600,800,1000$ trigger/verification image pairs respectively, and then calculate the minimal distance $d=\min_{i\in\{1,\cdots,P\}}\frac{1}{\#(\boldsymbol{S}_i)}\|\boldsymbol{S}_i^*-\boldsymbol{S}_i\|_2$. The results are listed in Table IV. We can see that the change of minimal distance is unnoticeable when P varies. When $P=1000$, the minimal distance is 1.11×10^{-2} on DnCNN (denoising), 1.17×10^{-2} on RED, 0.95×10^{-2} on SR-DnCNN and 0.94×10^{-2} on VDSR. Such values are much larger than the verification threshold $\eta=6.07\times10^{-3}$. Also refer to Fig. 8(b)-(e) for some visualization results generated by our auxiliary module, where the copyright images generated from $\boldsymbol{S}_i^*$ are completely unrecognizable. Such results have demonstrated that in the proposed method, the watermark information (*i.e.* trigger/verification image pair) can well distinguish the watermarked model from the un-watermarked one, and thus it is difficult to make a fraudulent claim of ownership on un-watermarked models.

TABLE IV: (Uniqueness Test) Minimal distances (10^{-2}) computed on unmarked models.

Task	Model	*P*=200	*P*=400	*P*=600	*P*=800	*P*=1000
Denoising	DnCNN	1.11	1.11	1.11	1.11	1.11
	RED	1.17	1.17	1.17	1.17	1.17
SR	DnCNN	0.95	0.95	0.95	0.95	0.95
	VDSR	0.94	0.94	0.94	0.94	0.94

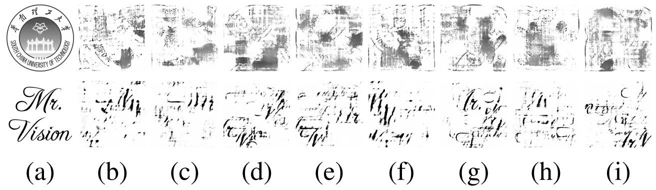


Fig. 8: Visualization of watermark information in the uniqueness test. The columns from left to right correspond to the copyright images generated by: (a) using the original version; (b) inputting a random trigger image to the un-watermarked DnCNN (denoising) model; (c) inputting a random trigger image to the un-watermarked RED model; (d) inputting a random trigger image to the un-watermarked DnCNN (SR) model; (e)inputting a random trigger image to the un-watermarked VDSR model; (f) inputting DnCNN's trigger image to the watermarked RED model; (g) inputting DnCNN's (SR) trigger image to the watermarked VDSR model; (h) inputting DnCNN's trigger image to the watermarked RED model; (i) inputting DnCNN's(SR) trigger image to the watermarked VDSR model.

2) Test on watermarked models: In the second test, we show that a watermarked model does not encode other watermark information which has not been embedded into it. Given a watermarked model $\mathcal{M}(\cdot;\boldsymbol{\theta}^*)$ with its own trigger image $\boldsymbol{K}_0$ and $\boldsymbol{S}_0$, a number of trigger images $\boldsymbol{K}_1,\cdots,\boldsymbol{K}_P$ are randomly generated by our key generator, and the verification images denoted by $\boldsymbol{S}_1,\cdots,\boldsymbol{S}_P$ are formed by our method on $\mathcal{M}$ respectively. Such images are to simulate the watermark information from other owners. The generated trigger images are input to $\mathcal{M}(\cdot;\boldsymbol{\theta}^*)$ separately, and then the output triggered response images, denoted by $\boldsymbol{S}_1^*,\cdots,\boldsymbol{S}_P^*$, are compared to the verification images. Similar to the first test, it is desired that the triggered images are different from the verification images, *i.e.* $\frac{1}{\#(\boldsymbol{S}_i)}\|\boldsymbol{S}_i^*-\boldsymbol{S}_i\|_2>\eta$ for $i=1,\cdots,P$.

In details, we generated $P=200,400,600,800,1000$ trigger/verification image pairs respectively, and calculated the minimal distance $d=\min_{i\in\{1,\cdots,P\}}\frac{1}{\#(\boldsymbol{S}_i)}\|\boldsymbol{S}_i^*-\boldsymbol{S}_i\|_2$. The results are listed in Table V. We can see that the minimal distance is stable as P increases. When $P=1000$, the minimal distance is 7.02×10^{-3} on DnCNN (denoising), 7.23×10^{-3} on RED, 7.38×10^{-3} on SR-DnCNN and 6.97×10^{-3} on VDSR, which is larger than the verification threshold $\eta=6.07\times10^{-3}$. Also refer to Fig. 8(f)(g) for some visualization results generated by our auxiliary module, where the copyright images generated from $\boldsymbol{S}_i^*$ are completely unrecognizable.

TABLE V: (Uniqueness Test) Minimal distances (10^{-3}) computed on marked models.

Task	Model	*P*=200	*P*=400	*P*=600	*P*=800	*P*=1000
Denoising	DnCNN	7.02	7.02	7.02	7.02	7.02
	RED	7.28	7.28	7.28	7.24	7.23
SR	DnCNN	7.42	7.38	7.38	7.38	7.38
	VDSR	6.97	6.97	6.97	6.97	6.97

3) Test on pairs of watermarked models: As discussed in the paper, one might concern the likely unboundedness of DNNs, which may weaken the uniqueness of the verification key. It is possible that $\mathcal{M}_1(\boldsymbol{K}_1;\boldsymbol{\theta}_1^*)\approx\mathcal{M}_2(\boldsymbol{K}_2;\boldsymbol{\theta}_2^*)$ for two watermarked DNNs $\mathcal{M}_1(\cdot;\boldsymbol{\theta}_1^*),\mathcal{M}_2(\cdot;\boldsymbol{\theta}_2^*)$ with very similar functions and two different keys $\boldsymbol{K}_1,\boldsymbol{K}_2$. However, this is not the case, since initial $\boldsymbol{S}$ is very close to $\mathcal{M}(\boldsymbol{K};\boldsymbol{\theta}^*)$ after embedding. Together with that $\boldsymbol{K}_1$ and the initial $\boldsymbol{S}_1$ are very different from $\boldsymbol{K}_2$ and the initial $\boldsymbol{S}_2$, the uniqueness is preserved.

For numerical verification, we first embed watermark into a DnCNN model $\mathcal{M}_1$ with $(\boldsymbol{K}_1,\boldsymbol{S}_1)$ and an RED model $\mathcal{M}_2$ with $(\boldsymbol{K}_2,\boldsymbol{S}_2)$, where $\boldsymbol{K}_1,\boldsymbol{K}_2$ are two different trigger images produced by our key generator, and then we exchange the trigger keys of the two models, *i.e.* inputting $\boldsymbol{K}_2$ to $\mathcal{M}_1$ and $\boldsymbol{K}_1$ to $\mathcal{M}_2$. Such a process is repeated 30 runs. Next, we calculate $\frac{1}{\#(\boldsymbol{S}_i)}\|\boldsymbol{S}_1-\mathcal{M}_1(\boldsymbol{K}_2)\|_2$ and $\frac{1}{\#(\boldsymbol{S}_i)}\|\boldsymbol{S}_2-\mathcal{M}_2(\boldsymbol{K}_1)\|_2$

(averaged over 30 runs), which are 7.03×10^{-3} and 7.62×10^{-3} respectively. Such values are much larger than the threshold in verification, demonstrating that in our method the watermark can well distinguish different watermarked models and thus can well identify the ownership of a model. Similar results can be observed when setting $\mathcal{M}_1, \mathcal{M}_2$ to be the SR-DnCNN and VSDR respectively. See also Fig. 8(h)(i) for some visualization results by our auxiliary module, where the copyright images generated from $\frac{1}{\#(\boldsymbol{S}_i)}\mathcal{M}_2(\boldsymbol{K}_1)$ and $\frac{1}{\#(\boldsymbol{S}_i)}\mathcal{M}_1(\boldsymbol{K}_2)$ are completely unrecognizable.

Further, we embed watermark into two RED models $\mathcal{M}_1, \mathcal{M}_2$ with the keys $(\boldsymbol{K}_1, \boldsymbol{S}_1)$ and $(\boldsymbol{K}_2, \boldsymbol{S}_2)$ respectively. The two RED models are trained with the same images but with different initialization schemes. Such two REDs are to simulate two models with very similar functions. We first calculate the distance of the initial verification keys and the distance of the updated verification keys, which are 8.59×10^{-3} and 8.04×10^{-3} respectively. In other words, the updated verification key can largely inherit the discrimination of the initial verification key. Then we exchange their keys and calculated $\frac{1}{\#(\boldsymbol{S}_i)}\|\boldsymbol{S}_1-\mathcal{M}_1(\boldsymbol{K}_2)\|_2$ and $\frac{1}{\#(\boldsymbol{S}_i)}\|\boldsymbol{S}_2-\mathcal{M}_2(\boldsymbol{K}_1)\|_2$, which are 7.87×10^{-3} and 7.56×10^{-3} respectively. Note that the values are computed over 30 runs with random different trigger keys. Such values are much larger than the threshold in verification, demonstrating that the uniqueness of the proposed method also holds even using two very similar models.

E. Robustness Analysis

1) Test methodology: The robustness of the proposed method is evaluated on three types of attacks: model compression [48], [49], [50], model fine-tuning [51] and watermark overwriting [14]. Under each attack, we report the results of the proposed method from different aspects:

- The denoising/SR performance of host model in terms of the average PSNR on Img12/Img14. This metric is not related to the robustness, but it can reflect the performance decrease of host model caused by the attack. An attack is meaningless if it causes too large performance decrease, as a model with low performance is useless in practice. Thus, we do not report this metric on other methods.
- The watermark distance. We report the distance d computed by (8) which is used for the verification in the proposed method and directly determines whether the proposed method can succeed. For the baseline methods, their verification processes are not based the distance d. Thus, we do not report the results on this metric for the baselines.
- Whether the verification succeeded. We report 'succeeded' or 'failed' for the verification by the proposed method.

Note that our method and the baselines use different rules for watermark detection, some of which may be more rigorous than others. A less rigorous rule often yields weaker robustness but stronger anti-falsification. Thus, comparing these methods based on whether their verification succeed may be not fair. For more-fair comparison, we consider the following criteria:

- The normalized correlation coefficient (NCC) between the verification key and the triggered output of the verified model. Large NCC means the triggered output of the watermarked model is similar to the verification key even under attacks. Thus, higher NCC implies stronger robustness.
- NCC-based verification. We conduct verification according to NCC. The verification passes if the NCC is larger than a threshold which is set to 0.95 in the experiments. Such a threshold value may be a bit high for practical applications, but it makes the verification very rigorous in order to better distinguish the robustness of different methods.

Furthermore, we use the proposed auxiliary module to visual some verification results in the robustness experiments. See Fig. 9(b)-(h) and supplementary materials. The output copyright images have their visual quality decreased a bit as the strength of the attack increases, but they are still easily recognizable. To demonstrate the discriminability of the auxiliary module, we feed a fraudulent trigger image to the models and use the output image as the input of the auxiliary module, and the results are shown in Fig. 9(i). It can be seen that without the correct trigger key, we cannot get a recognizable image from the auxiliary visualizer module.

2) Robustness to model compression: Following the protocol of Han *et al.* [48], we compress the watermarked models by parameter pruning on each layer: sorting the model weights in the ascending order in terms of magnitude and then setting the top $p\%$ to zeros. We set p to different values for evaluation. The results of our method and the baselines are listed in Table VI for comparison.

It can be seen from Table VI that on all tested models, the proposed method succeeds under all pruning rates except the highest one. Under the moderate compression, the watermark can be consistently detected by the proposed method. Even under the heavy compression (*e.g.* 35% pruned in DnCNN and VDSR), our watermark verification still works. Until the pruning rate is increased such that it significantly decreases the performance of the models (*e.g.* 2dB drop on DnCNN), our method is unable to detect the watermark from these models See Fig. 9(b)-(c) and supplementary materials for visualization of model compression results. It is interesting to see that even our verification failed in the compression attack with pruning rate 40%, we can still recognize the corresponding copyright. This suggests that the visual module may rectify the verification results for auxiliary judgment. It is also interesting to see that the performance of RED decreases much faster than DnCNN as the pruning rate increases, which again implies that RED is much less redundant than DnCNN.

Compared to the baselines, our method achieved higher NCC values on all the test models and performed better in terms of the NCC-based verification results. We can also see that Base-2 performed much worse than Base-1, which suggests the image-to-image watermarking in Base-1 is superior to the image-to-label watermarking in Base-2.

3) Robustness to model fine-tuning: We use two often-seen types of model fine-tunings for the evaluation. The first type uses the original training data. In this setting, we use the same data as [22], [45]. The second type of fine-turning employs a new dataset. In this setting, for the denoising DnCNNs, we use the texture image dataset KTH-TIPS [52] with 50% –

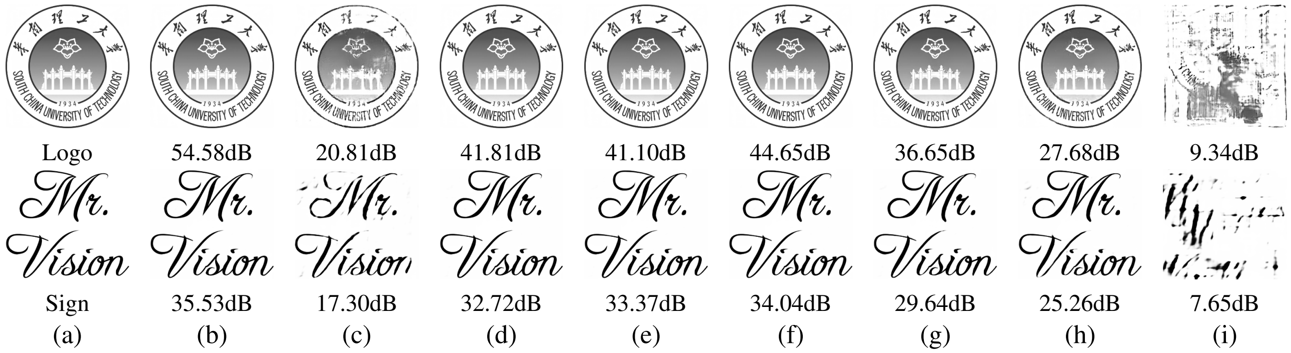


Fig. 9: Visualization of watermark data (in the form of images output by the auxiliary visualizer) on DnCNN in different cases. (a) Original copyright images; (b) Compression with 10% pruning; (c) Compression with 40% pruning; (d) Fine-tuning with 10 epochs on the original dataset; (e) Fine-tuning with 50 epochs on the original dataset; (f) Fine-tuning with 10 epochs on the texture dataset; (g) Fine-tuning with 50 epochs on the texture dataset; (h) Overwriting with a new trigger key; (i) Using a false trigger key. Values below images denote PSNR.

TABLE VI: (Robustness Test) Test results on model compression attacks.

Model	**Pruning Rate**	**Our Result**			**Comparison to Baselines**					
		PSNR	**Distance**	**Verification**	**NCC**			**Verification (NCC>0.95)**		
			($\times 10^{-3}$)	$d > \eta$	Base-1	Base-2	Ours	Base-1	Base-2	Ours
DnCNN	0.25	29.77	2.09	**Succeeded**	0.9723	0.7501	0.9915	**Succeeded**	Failed	**Succeeded**
	0.22	29.42	2.82	**Succeeded**	0.9623	0.7390	0.9857	**Succeeded**	Failed	**Succeeded**
	0.35	29.22	4.48	**Succeeded**	0.9350	0.6385	0.9707	Failed	Failed	**Succeeded**
	0.40	28.39	6.36	Failed	0.9070	0.5490	0.9355	Failed	Failed	Failed
RED	0.08	26.26	2.41	**Succeeded**	0.9790	0.6926	0.9889	**Succeeded**	Failed	**Succeeded**
	0.10	23.85	3.83	**Succeeded**	0.9346	0.6549	0.9723	Failed	Failed	**Succeeded**
	0.15	19.49	5.98	**Succeeded**	0.8627	0.5485	0.9500	Failed	Failed	Failed
	0.20	23.18	7.93	Failed	0.8114	0.5481	0.8654	Failed	Failed	Failed
VDSR	0.25	30.04	2.21	**Succeeded**	0.9333	0.6944	0.9960	Failed	Failed	**Succeeded**
	0.22	29.89	3.75	**Succeeded**	0.8834	0.5477	0.9880	Failed	Failed	**Succeeded**
	0.35	29.48	5.01	**Succeeded**	0.8423	0.5258	0.9765	Failed	Failed	**Succeeded**
	0.40	28.94	6.58	Failed	0.8034	0.5250	0.9559	Failed	Failed	**Succeeded**
SR-DnCNN	0.25	29.88	2.76	**Succeeded**	0.9837	0.6826	0.9920	**Succeeded**	Failed	**Succeeded**
	0.22	29.78	3.86	**Succeeded**	0.9451	0.6418	0.9846	Failed	Failed	**Succeeded**
	0.35	29.47	5.99	**Succeeded**	0.8938	0.5476	0.9609	Failed	Failed	**Succeeded**
	0.40	29.07	6.73	Failed	0.8307	0.5258	0.9494	Failed	Failed	Failed

50% for training/test, which simulates the attackers transfer the task from natural image denoising to the denoising on specific texture images. For the SR DNNs, we use a new SR dataset DIV2K [28], which simulates that the attackers illegally use the DNN for the same task but on their own data. In the fine-tuning, we follow the settings of the original models and set the number of epochs to $10, 25, 50, 75, 100$ respectively.

See Table VII for the results. Our method is robust to the fine-tuning attacks. With the increase of epochs, the watermark becomes weaker but is still successfully detected, even with 100-epoch fine-tuning. For DnCNN and RED, it is interesting to see that the fine-tuning on new data has larger impact on the watermark than that on original data. The reason is probably that texture images have higher randomness than the natural images of the original dataset, and therefore the texture patches are closer to the randomly-generated patches in the trigger image. Recall from Fig. 1 that the fine-tuning on such closer patches has larger impacts on the embedding with trigger images. Note that this is not the case for VDSR as the new data for SR are also natural images. See Fig. 9(d)-(g) and supplementary materials for the visualization results.

It can also be seen that the proposed method outperformed the baselines, with more times of success in the NCC-based verification and higher NCC values achieved overall. Such results have demonstrated the advantages of the proposed method over the baselines in resisting the attacks with model fune-turning. We also note that Base-2 is very sensitive to fine-tuning attacks again performed much worse than Base-1. This implies that the methods for watermarking classification DNNs cannot be directly applied to image processing DNNs.

4) Robustness to watermark overwriting: The overwriting is done by embedding a new trigger key and its corresponding verification key into the watermarked model. The verification results of different methods under the overwriting attack with one trigger key are listed in Table VIII. It can be seen that our method succeeds on all the tested models in the both our verification scheme and the NCC-based verification. In contrast, both the baselines fail in all the cases. The reason for the superior robustness of our method over the baselines is probably that our method takes the characteristic of the image process DNN into consideration, instead of purely viewing it as a mixture of multiple classification DNNs.

Encouraged by the above results, we further test the robustness of our method to the overwriting with multiple trigger images. We omit the baselines in this test as they failed on even the overwriting with a single trigger key. See Table IX for the results. It can be seen that even with multiple new watermarks written into the host model, the original one can

TABLE VII: (Robustness Test) Test results on model fine-tuning attacks.

Model	Data	#Epochs	Our Result			Comparison to Baselines					
			PSNR	Distance	Verification	NCC			Verification (NCC>0.95)		
				($\times 10^{-3}$)	$d > \eta$	Base-1	Base-2	Ours	Base-1	Base-2	Ours
DnCNN	Original	10	30.42	1.64	Succeeded	0.9988	0.5490	0.9953	Succeeded	Failed	Succeeded
		25	30.41	1.66	Succeeded	0.9943	0.5478	0.9956	Succeeded	Failed	Succeeded
		50	30.43	1.81	Succeeded	0.9949	0.5477	0.9959	Succeeded	Failed	Succeeded
		75	30.44	2.38	Succeeded	0.9949	0.5268	0.9952	Succeeded	Failed	Succeeded
		100	30.41	3.12.	Succeeded	0.9955	0.5267	0.9938	Succeeded	Failed	Succeeded
	Texture	10	30.28	1.23	Succeeded	1.0000	0.6826	0.9979	Succeeded	Failed	Succeeded
		25	30.15	1.76	Succeeded	0.9877	0.5477	0.9944	Succeeded	Failed	Succeeded
		50	30.09	2.32	Succeeded	0.9801	0.5255	0.9914	Succeeded	Failed	Succeeded
		75	30.11	3.81	Succeeded	0.9786	0.5250	0.9832	Succeeded	Failed	Succeeded
		100	30.10	5.69	Succeeded	0.9574	0.5245	0.9580	Succeeded	Failed	Succeeded
RED	Original	10	30.28	2.08	Succeeded	1.0000	0.6111	0.9957	Succeeded	Failed	Succeeded
		25	30.28	2.09	Succeeded	1.0000	0.5560	0.9932	Succeeded	Failed	Succeeded
		50	30.30	2.08	Succeeded	0.9997	0.5490	0.9935	Succeeded	Failed	Succeeded
		75	30.23	4.03	Succeeded	0.9997	0.5268	0.9915	Succeeded	Failed	Succeeded
		100	30.26	5.62	Succeeded	0.9988	0.5253	0.9836	Succeeded	Failed	Succeeded
	Texture	10	29.90	1.39	Succeeded	0.9994	0.5773	0.9969	Succeeded	Failed	Succeeded
		25	30.01	1.82	Succeeded	0.9860	0.5560	0.9936	Succeeded	Failed	Succeeded
		50	30.09	2.00	Succeeded	0.9824	0.5549	0.9927	Succeeded	Failed	Succeeded
		75	30.05	3.82	Succeeded	0.9791	0.5258	0.9877	Succeeded	Failed	Succeeded
		100	30.02	5.83	Succeeded	0.9767	0.5244	0.9655	Succeeded	Failed	Succeeded
VDSR	Original	10	30.14	0.84	Succeeded	0.8630	0.8501	0.9993	Failed	Failed	Succeeded
		25	30.10	1.06	Succeeded	0.7756	0.8056	0.9989	Failed	Failed	Succeeded
		50	30.05	1.43	Succeeded	0.7594	0.5258	0.9979	Failed	Failed	Succeeded
		75	30.00	1.88	Succeeded	0.7609	0.5255	0.9966	Failed	Failed	Succeeded
		100	29.96	2.29	Succeeded	0.7606	0.5245	0.9948	Failed	Failed	Succeeded
	DIV2K	10	30.12	2.40	Succeeded	0.8152	0.8504	0.9947	Failed	Failed	Succeeded
		25	30.13	3.04	Succeeded	0.7631	0.6926	0.9949	Failed	Failed	Succeeded
		50	30.06	3.86	Succeeded	0.7599	0.6478	0.9889	Failed	Failed	Succeeded
		75	30.07	4.36	Succeeded	0.7612	0.6268	0.9854	Failed	Failed	Succeeded
		100	30.05	4.83	Succeeded	0.7598	0.6246	0.9815	Failed	Failed	Succeeded
SR-DnCNN	Original	10	30.11	1.68	Succeeded	1.0000	0.5476	0.9979	Succeeded	Failed	Succeeded
		25	30.09	2.18	Succeeded	0.9967	0.5257	0.9961	Succeeded	Failed	Succeeded
		50	30.03	2.72	Succeeded	0.9545	0.5250	0.9930	Succeeded	Failed	Succeeded
		75	30.00	3.06	Succeeded	0.9137	0.5248	0.9906	Failed	Failed	Succeeded
		100	29.96	3.45	Succeeded	0.8843	0.5245	0.9878	Failed	Failed	Succeeded
	DIV2K	10	30.17	5.20	Succeeded	0.9979	0.5490	0.9708	Succeeded	Failed	Succeeded
		25	30.14	4.97	Succeeded	0.9611	0.5478	0.9738	Succeeded	Failed	Succeeded
		50	30.08	4.82	Succeeded	0.9038	0.5267	0.9752	Failed	Failed	Succeeded
		75	30.05	4.93	Succeeded	0.8842	0.5253	0.9741	Failed	Failed	Succeeded
		100	30.00	4.97	Succeeded	0.8682	0.5244	0.9734	Failed	Failed	Succeeded

TABLE VIII: (Robustness Test) Test results on watermark overwriting attacks.

Model	Our Result			Comparison to Baselines					
	PSNR	Distance	Verification	NCC			Verification (NCC>0.95)		
		($\times 10^{-3}$)	$d > \eta$	Base-1	Base-2	Ours	Base-1	Base-2	Ours
DnCNN	30.41	3.66	Succeeded	0.8128	0.7957	0.9751	Failed	Failed	Succeeded
RED	30.27	3.47	Succeeded	0.8915	0.8689	0.9779	Failed	Failed	Succeeded
VDSR	30.10	3.07	Succeeded	0.7628	0.6581	0.9915	Failed	Failed	Succeeded
SR-DnCNN	30.13	3.19	Succeeded	0.7875	0.8833	0.9891	Failed	Failed	Succeeded

still be detected. Such robustness comes from the fact that the new trigger images are very likely to be far away from the original one due to the randomness in the generation; see *e.g.* Fig. 1. The embedding with the new trigger images mainly changes the model behavior around its neighborhood, which has little impact on the model behavior on the regions which are far from the new trigger images. See Fig. 9(h) and supplementary materials for the visualization results.

F. Capacity Test

The evaluation on the capacity of the proposed watermarking method is conducted as follows. The trigger keys with sizes 20×20, 40×40, 80×80, 160×160, 320×320, 480×480 and 600×600 are generated and embedded into the host models respectively. Then the denoising/SR performance of the watermarked models is evaluated on the test datasets. The results are listed in Table X and Table XI. Regarding the RED model it can be seen that, the larger a trigger image is, the more the denoising performance drops. This is not surprising, as the capacity of a watermarking method is often in contradiction with its capability of fidelity. Similar phenomena can be observed on other models. It is interesting to see that the capacity of DnCNN (denoising) is much larger than RED; the denoising performance of DnCNN is not sensitive to the size change of trigger image within 80×80. This is probably due to that RED is less redundant than DnCNN, which is also seen in the experiments on fidelity analysis and robustness analysis. When the size of trigger image is sufficiently large,

TABLE IX: (Robustness Test) Test results of proposed method on different models under multiple overwriting attacks.

	# Triggers	Distance	NCC	PSNR(dB)	Detection
DnCNN	1	3.66×10^{-3}	0.9751	30.41	**Succeeded**
	2	4.62×10^{-3}	0.9605	30.41	**Succeeded**
	3	4.93×10^{-3}	0.9542	30.41	**Succeeded**
	4	5.17×10^{-3}	0.9505	30.39	**Succeeded**
	5	5.28×10^{-3}	0.9500	30.39	**Succeeded**
RED	1	3.47×10^{-3}	0.9779	30.27	**Succeeded**
	2	4.52×10^{-3}	0.9628	29.95	**Succeeded**
	3	4.83×10^{-3}	0.9583	30.23	**Succeeded**
	4	5.14×10^{-3}	0.9558	29.23	**Succeeded**
	5	5.20×10^{-3}	0.9530	30.19	**Succeeded**
VDSR	1	3.07×10^{-3}	0.9923	30.10	**Succeeded**
	2	3.04×10^{-3}	0.9917	30.13	**Succeeded**
	3	3.22×10^{-3}	0.9902	30.07	**Succeeded**
	4	3.16×10^{-3}	0.9912	30.09	**Succeeded**
	5	3.24×10^{-3}	0.9904	30.08	**Succeeded**
SR-DnCNN	1	3.19×10^{-3}	0.9891	30.13	**Succeeded**
	2	3.97×10^{-3}	0.9841	30.11	**Succeeded**
	3	4.99×10^{-3}	0.9736	30.13	**Succeeded**
	4	4.30×10^{-3}	0.9803	30.10	**Succeeded**
	5	4.89×10^{-3}	0.9747	30.09	**Succeeded**

TABLE X: (Capacity Test) Average PSNR(dB) and SSIM values of denoising results of our watermarked models with different trigger image sizes.

	Trigger	**PSNR(dB)**		**SSIM**$(\times 10^{-1})$	
	Image Size	**BSD68**	**Img12**	**BSD68**	**Img12**
DnCNN	0 (Unmarked)	29.20	30.41	8.265	8.605
	20 × 20	29.20	30.42	8.263	8.610
	40 × 40	29.20	30.41	8.275	8.612
	80 × 80	29.20	30.42	8.276	8.613
	160 × 160	29.18	30.36	8.275	8.598
	320 × 320	29.06	30.19	8.232	8.557
	480 × 480	28.97	30.15	8.210	8.541
	600 × 600	28.93	30.00	8.173	8.499
RED	0 (Unmarked)	29.11	30.28	8.222	8.571
	20 × 20	29.09	30.27	8.225	8.567
	40 × 40	29.06	30.27	8.226	8.562
	80 × 80	28.97	30.12	8.225	8.553
	160 × 160	28.97	30.09	8.211	8.537
	320 × 320	28.96	30.11	8.192	8.531
	480 × 480	28.88	29.94	8.144	8.478
	600 × 600	28.66	29.59	8.078	8.389

TABLE XI: (Capacity Test) Average PSNR(dB) values of SR results of our watermarked models under different trigger image sizes.

	Scale	**VDSR**	**DnCNN**		**Scale**	**VDSR**	**DnCNN**
unmarked	×2	32.83	32.90	20 × 20	×2	32.85	32.92
	×3	29.66	29.68		×3	29.68	29.68
	×4	27.88	27.83		×4	27.88	27.81
	mean	30.12	30.12		mean	30.14	30.14
40×40	×2	32.83	32.87	80×80	×2	32.84	32.88
	×3	29.67	29.67		×3	29.68	29.69
	×4	27.86	27.82		×4	27.86	27.83
	mean	30.12	30.14		mean	30.13	30.13
160×160	×2	32.82	32.83	320×320	×2	32.63	32.67
	×3	29.67	29.68		×3	29.61	29.60
	×4	27.86	27.84		×4	27.84	27.82
	mean	30.12	30.12		mean	30.03	30.03
480×480	×2	32.28	32.30	600×600	×2	31.94	31.77
	×3	29.43	29.37		×3	29.25	29.03
	×4	27.73	27.61		×4	27.60	27.36
	mean	29.81	29.75		mean	29.60	29.39

TABLE XII: (Capacity Test) Average PSNR(dB) values of denoising results on Img12 and SR results on Img14 given by different methods.

Trigger	**Base-1**	**Ours**	**Base-1**	**Ours**
Image Size	**PSNR (dB)**	**PSNR (dB)**	**PSNR (dB)**	**PSNR (dB)**
Denoising	DnCNN		RED	
40 × 40	30.18	30.41	30.12	30.27
160 × 160	30.10	30.36	30.03	30.09
320 × 320	28.49	30.19	29.02	30.11
600 × 600	27.34	30.00	23.48	29.59
SR	VDSR		SR-DnCNN	
40 × 40	29.83	30.12	30.02	30.14
160 × 160	28.27	30.12	29.98	30.12
320 × 320	28.11	30.03	29.43	30.03
600 × 600	28.09	29.60	28.55	29.39

TABLE XIII: (Time Test) Time cost of proposed method and two baselines on different models with different trigger image sizes.

	Trigger	**Embedding Time (min.)**			**Verification Time (s)**		
	Image Size	**Base-1**	**Base-2**	**Ours**	**Base-1**	**Base-2**	**Ours**
DnCNN	40 × 40	23.98	25.35	22.70	2.81	2.96	2.77
	80 × 80	24.54	25.84	22.82	2.84	3.09	2.80
	160 × 160	26.78	28.65	24.81	2.87	3.15	2.84
	320 × 320	37.56	39.85	33.37	2.89	3.17	2.85
	480 × 480	70.21	73.41	67.58	2.91	3.19	2.86
	600 × 600	183.32	190.13	174.25	3.01	3.30	2.98
RED	40 × 40	18.28	19.32	16.95	2.61	2.74	2.56
	80 × 80	19.39	20.21	17.26	2.63	2.75	2.57
	160 × 160	20.89	22.24	18.66	2.65	2.78	2.59
	320 × 320	26.18	27.96	24.60	2.69	2.84	2.62
	480 × 480	37.72	39.71	34.47	2.76	2.94	2.71
	600 × 600	51.19	55.54	47.85	2.92	3.11	2.84
VDSR	40 × 40	35.17	36.64	33.13	2.52	2.61	2.47
	80 × 80	36.43	37.45	33.89	2.53	2.64	2.48
	160 × 160	39.29	40.51	37.82	2.55	2.65	2.49
	320 × 320	77.78	79.95	74.97	2.63	2.71	2.54
	480 × 480	71.15	84.15	78.50	2.69	2.76	2.58
	600 × 600	109.96	115.21	102.52	2.75	2.80	2.61
SR-DnCNN	40 × 40	36.81	38.88	35.48	2.80	2.93	2.75
	80 × 80	47.74	49.46	45.08	2.82	2.94	2.76
	160 × 160	68.16	70.85	65.74	2.83	2.97	2.78
	320 × 320	83.35	86.69	80.51	2.87	3.01	2.81
	480 × 480	85.59	91.58	84.75	2.90	3.09	2.86
	600 × 600	205.18	213.21	197.68	2.99	3.18	2.93

e.g. 160 × 160, the DnCNN model becomes overriding and the denoising performance starts to degrade. The denoising performance degradation is still acceptable with trigger image sizes not lager than 160 × 160. When the trigger image size changes from 160 × 160 to 600 × 600, the denoising performance decrease of DnCNN becomes more and more noticeable. For the size 600 × 600 which is about 14 times as 160 × 160, the denoising performance drops around 0.27dB. We note that from previous experiments, using trigger images with size 40 × 40 already yields good detection results.

Since the verification key is different for our method and Base-2, *i.e.* image vs. label, it is hard to compare their capacity fairly. We note that Base-2 is about image-to-label training while ours is about image-to-image training. In this sense, our method is of higher efficiency and thus the capacity should be higher. For comparison, we only use Base-1 as it is based on the image-to-image scheme as ours. See Table XII for the results, where our method shows higher capacity than Base-1. For instance, for the trigger image size 600 × 600, the PSNR performance of DnCNN is much better than that in Base-1.

G. Time Cost

The time cost of the proposed method includes the watermark embedding time and the watermark verification time, both of which depend on many factors such as the complexity of host model and the size of trigger image. In Table XIII, we report the timing cost of the proposed method on different models with different trigger sizes. We also include the time cost of two baselines for comparison. It can be seen from Table XIII that the verification time does not vary too much, while the embedding takes more time as the trigger image size increases. Compared to Base-1 and Base-2, the proposed method is faster. The Base-2 is the slowest as it only embeds one bit using an image. In comparison, the proposed method and Base-1 can embed a matrix (image) using one image, leading to higher efficiency. Overall, the time cost of the proposed method is reasonable and acceptable in practice.

VI. Conclusion

As the publication and sharing of DNN models become a common practice, the need arises for having watermarking techniques to protect the intellectual properties of trained DNN models. This paper proposed an effective black-box approach to watermarking the DNN models that are used for image processing, and demonstrated its effectiveness in the context of image denoising and image super-resolution. The watermark embedding is done by modifying the host DNN so as to degrade its performance on a specific image which has statistically significant difference from training data. Different aspects of the proposed method were tested, and it showed that the proposed method is a good one to the watermark DNN models used in image processing.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 1

[2] L. Zhou, Z. Wang, Y. Luo, and Z. Xiong, "Separability and compactness network for image recognition and superresolution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3275–3286, Nov 2019. 1

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2016, pp. 779–788. 1

[4] J. Yu, C. Zhu, J. Zhang, Q. Huang, and D. Tao, "Spatial pyramid-enhanced netvlad with weighted triplet loss for place recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 2, pp. 661–674, Feb 2020. 1

[5] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Proc. Advances in Neural Inform. Process. Syst.*, 2012, pp. 341–349. 1

[6] R. Dian, S. Li, A. Guo, and L. Fang, "Deep hyperspectral image sharpening," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5345–5355, Nov 2018. 1

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2015, pp. 1–9. 1

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances in Neural Inform. Process. Syst.*, 2012, pp. 1097–1105. 1

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2016, pp. 770–778. 1

[10] "Model Zoo," http://modelzoo.co/. 1, 3

[11] "Baidu AI," http://ai.baidu.com/. 1, 3

[12] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *USENIX Security Symp.* Baltimore, MD: USENIX Association, 2018, pp. 1615–1631. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/adi 1, 3, 7

[13] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis." in *USENIX Security Symp.*, 2016, pp. 601–618. 1, 3

[14] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proc. ACM Int. Conf. Multimedia Retrieval*. ACM, 2017, pp. 269–277. 1, 3, 9

[15] B. D. Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: A generic watermarking framework for ip protection of deep learning models," *arXiv preprint arXiv:1804.00750*, 2018. 1, 3, 6, 7

[16] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proc. Asia Conf. Comput. Comm. Security*, 2018, pp. 159–172. 1, 3

[17] E. L. Merrer, P. Perez, and G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *arXiv preprint arXiv:1711.01894*, 2017. 1, 3, 5

[18] J. Guo and M. Potkonjak, "Watermarking deep neural networks for embedded systems," in *Proc. Int. Conf. Computer-Aided Design*. IEEE, 2018, pp. 1–8. 1, 3

[19] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "Deblurgan: Blind motion deblurring using conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, June 2018. 1

[20] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European Conf. Comput. Vision*. Springer, 2014, pp. 184–199. 1

[21] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Proc. Advances in Neural Inform. Process. Syst.*, 2016, pp. 2802–2810. 1, 6

[22] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, 2017. 1, 2, 6, 7, 9

[23] L. Gatys, A. Ecker, and M. Bethge, "A neural algorithm of artistic style," *Nature Commun.*, 2015. 1

[24] M. Xia, X. Liu, and T.-T. Wong, "Invertible grayscale," in *ACM Trans. Graphics (SIGGRAPH Asia 2018 issue)*. ACM, 2018, p. 246. 1

[25] Q. Fan, J. Yang, G. Hua, B. Chen, and D. Wipf, "A generic deep architecture for single image reflection removal and image smoothing," in *Proc. IEEE Int. Conf. Comput. Vision*, Oct 2017. 1

[26] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (red)," *SIAM J. Imag. Sci.*, vol. 10, no. 4, pp. 1804–1844, 2017. 2

[27] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser prior for image restoration," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2017, pp. 3929–3938. 2

[28] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, July 2017. 2, 10

[29] W. Dong, X. Li, L. Zhang, and G. Shi, "Sparsity-based image denoising via dictionary learning and structural clustering," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*. IEEE, 2011, pp. 457–464. 2

[30] G. Peyré, "Manifold models for signals and images," *Comput. Vision Image Understanding*, vol. 113, no. 2, pp. 249–260, 2009. 2

[31] D. Kirovski and H. S. Malvar, "Spread-spectrum watermarking of audio signals," *IEEE Trans. Signal Process.*, vol. 51, no. 4, pp. 1020–1033, 2003. 3

[32] M. D. Swanson, B. B. Zhu, and A. H. Tewfik, "Transparent robust image watermarking." in *Proc. IEEE Int. Conf. Comput. Vision*. Citeseer, 1996, pp. 211–214. 3

[33] A. Fierro-Radilla, M. Nakano-Miyatake, M. Cedillo-Hernandez, L. Cleofas-Sanchez, and H. Perez-Meana, "A robust image zero-watermarking using convolutional neural networks," in *Proc. Int. Workshop Bio. Forensics*. IEEE, 2019, pp. 1–5. 3

[34] M. D. Swanson, B. Zhu, and A. H. Tewfik, "Multiresolution scene-based video watermarking using perceptual models," *IEEE J. Selected Areas Commun.*, vol. 16, no. 4, pp. 540–550, 1998. 3

[35] C.-C. Chang, J.-Y. Hsiao, and C.-S. Chan, "Finding optimal least-significant-bit substitution in image hiding by dynamic programming strategy," *Pattern Recognition*, vol. 36, no. 7, pp. 1583–1595, 2003. 3

[36] M. Barni, F. Bartolini, V. Cappellini, and A. Piva, “A dct-domain system for robust image watermarking,” *Signal Proc.*, vol. 66, no. 3, pp. 357–372, 1998. 3

[37] C. Chu, A. Zhmoginov, and M. Sandler, “Cyclegan, a master of steganography,” *arXiv preprint arXiv:1712.02950*, 2017. 3

[38] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, “Hidden: Hiding data with deep networks,” in *Proc. European Conf. Comput. Vision*, September 2018. 3

[39] “Magenta,” http://magenta.tensorflow.org/. 3

[40] Y. Nagai, Y. Uchida, S. Sakazawa, and S. Satoh, “Digital watermarking for deep neural networks,” *Int. J. Multimedia Info. Retrieval*, vol. 7, no. 1, pp. 3–16, 2018. 3

[41] “Tencent Youtu,” http://open.youtu.qq.com/. 3

[42] H. Chen, B. D. Rohani, and F. Koushanfar, “Deepmarks: A digital fingerprinting framework for deep neural networks,” *arXiv preprint arXiv:1804.03648*, 2018. 4

[43] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, “Semi-supervised knowledge transfer for deep learning from private training data,” in *Proc. Int. Conf. Learning Representation*, 2017. 4

[44] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, “Scalable private learning with pate,” 2018. 4

[45] J. Kim, J. Kwon Lee, and K. Mu Lee, “Accurate image super-resolution using very deep convolutional networks,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2016, pp. 1646–1654. 6, 9

[46] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proc. Int. Conf. Learning Representation*, 2015. 6

[47] J. Xu, H. Li, Z. Liang, D. Zhang, and L. Zhang, “Real-world noisy image denoising: A new benchmark,” *arXiv preprint arXiv:1804.02603*, 2018. 7

[48] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” in *Proc. Int. Conf. Learning Representation*, 2015. 9

[49] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in *Proc. Advances in Neural Inform. Process. Syst.*, 2015, pp. 1135–1143. 9

[50] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, “Eie: efficient inference engine on compressed deep neural network,” in *Ann. Int. Symp. Comput. Architecture*. IEEE, 2016, pp. 243–254. 9

[51] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, “Convolutional neural networks for medical image analysis: Full training or fine tuning?” *IEEE Trans. Medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016. 9

[52] M. Fritz, E. Hayman, B. Caputo, and J.-O. Eklundh, “The kth-tips database,” 2004. 9

Yuhui Quan received the doctoral degree in Computer Science from South China University of Technology in 2013. He worked as the postdoctoral research fellow in Mathematics at National University of Singapore from 2013 to 2016. He is currently the associate professor at School of Computer Science and Engineering in South China University of Technology. His research interests include computer vision, image processing, deep learning, and sparse representation.

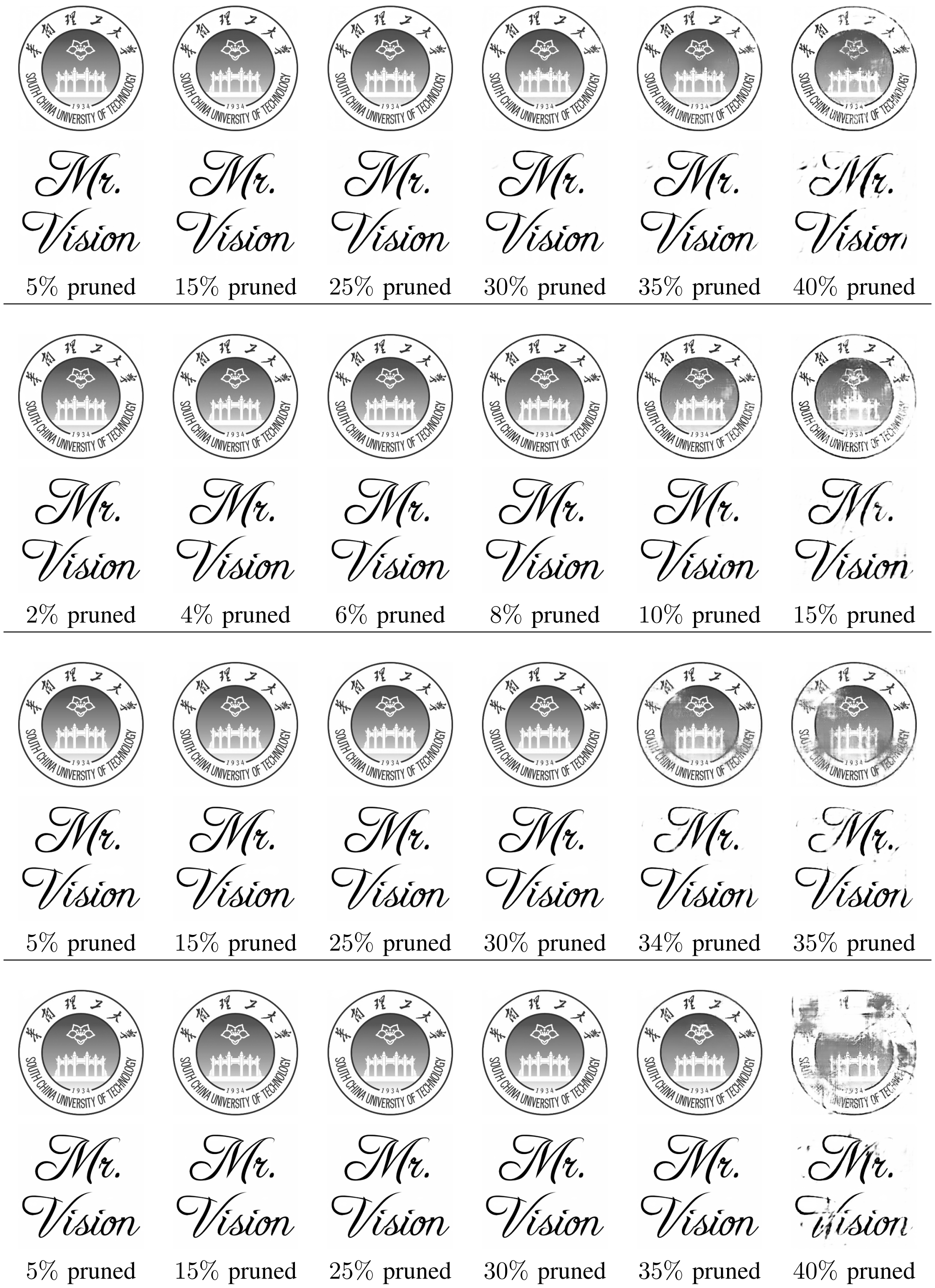
Huan Teng received the Bachelor degree in Computer Science from South China University of Technology in 2018. He currently is a master candidate at School of Computer Science and Engineering in South China University of Technology. His research interests include computer vision, image processing, deep learning, and sparse coding.

Yixin Chen received the Bachelor degree in Network Engineering from South China University of Technology in 2017. He currently is a master candidate at School of Computer Science and Engineering in South China University of Technology. His research interests include computer vision, image processing, and deep learning.

Hui Ji received the B.Sc. degree in Mathematics from Nanjing University in China, the M.Sc. degree in Mathematics from National University of Singapore and the Ph.D. degree in Computer Science from the University of Maryland, College Park. In 2006, he joined National University of Singapore as an assistant professor in Mathematics. Currently, he is an associate professor in Mathematics at National University of Singapore. His research interests include computational harmonic analysis, optimization, computational vision, image processing, and machine learning.

Supplementary Materials

Due to space limitation, we only show some visualization results in our paper. Here we show more visualization results in Fig. 1,2,3. It can be seen that our method is robust to different types of attacks.



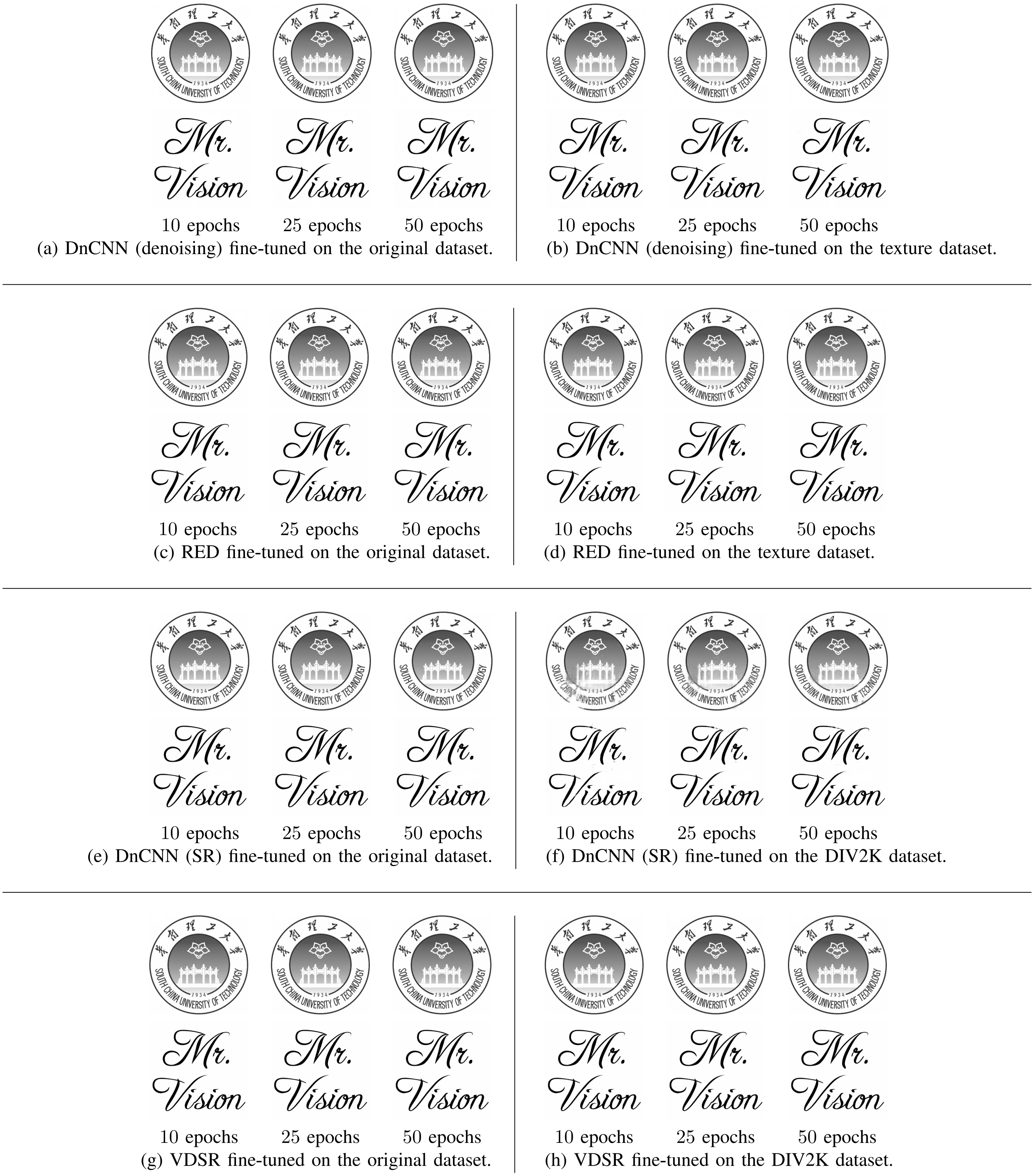
5% pruned 15% pruned 25% pruned 30% pruned 35% pruned 40% pruned

2% pruned 4% pruned 6% pruned 8% pruned 10% pruned 15% pruned

5% pruned 15% pruned 25% pruned 30% pruned 34% pruned 35% pruned

5% pruned 15% pruned 25% pruned 30% pruned 35% pruned 40% pruned

Fig. 1: Visualization of watermark information under model compression attacks. The columns from left to right correspond to different compression rates respectively, and the rows from top to bottom correspond to the DnCNN (denoising), RED, DnCNN (SR) and VDSR respectively.



(a) DnCNN (denoising) fine-tuned on the original dataset.

(b) DnCNN (denoising) fine-tuned on the texture dataset.

(c) RED fine-tuned on the original dataset.

(d) RED fine-tuned on the texture dataset.

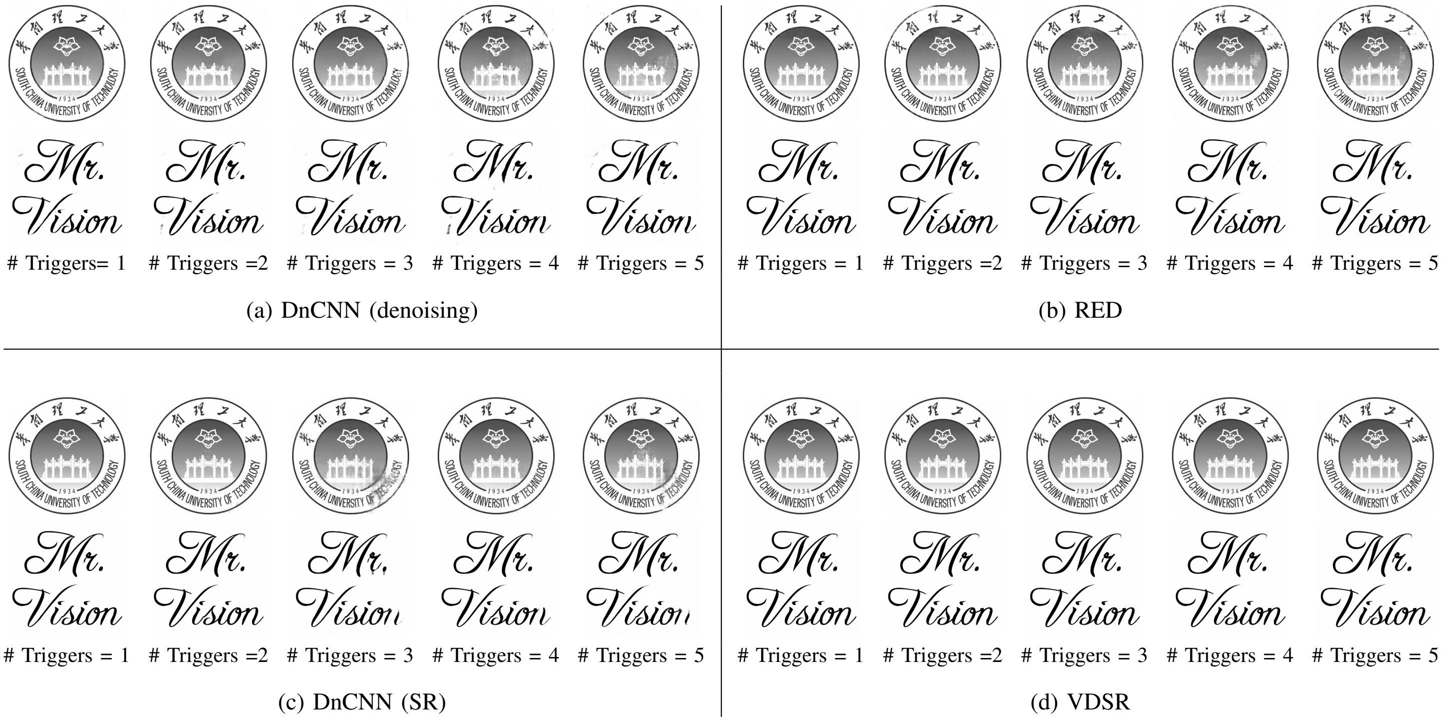
(e) DnCNN (SR) fine-tuned on the original dataset.

(f) DnCNN (SR) fine-tuned on the DIV2K dataset.

(g) VDSR fine-tuned on the original dataset.

(h) VDSR fine-tuned on the DIV2K dataset.

Fig. 2: Visualization of watermark information under model fine-tuning attacks.



(a) DnCNN (denoising)

(b) RED

(c) DnCNN (SR)

(d) VDSR

Fig. 3: Visualization of watermark information under watermark overwriting attacks.