Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105
Volume **9**, Number **4**, November **2024**

A Novel Deep Learning Model Watermark Algorithm with Strong Anti-pruning Robustness

Xiuyan Sun

Public Education Department of Laiwu Vocational and Technical College
sunxiuyan0634@163.com

Chengling Gu

Teaching Affairs Office of Laiwu Vocational and Technical College
chengling_2119@163.com

Linlin Tang*

Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies
Harbin Institute of Technology (Shenzhen), Shenzhen 518067, P. R. China
hittang@126.com

Lingping Kong

Faulty of Computer Science
VSB-Technical University of Ostrava, Ostrava 708 00, Czech Republic
lingping.kong@vsb.cz

*Corresponding author: Linlin Tang

Received January 15, 2024, revised April 18, 2024, accepted August 1, 2024.

Abstract. *Deep learning has developed rapidly in recent years and has achieved surprising results in multiple fields. A trained model consumes a considerable amount of computational resources from the trainer and has certain value. Therefore, copyright protection for deep learning models is an issue that cannot be ignored. This article proposes a framework consisting of three neural networks: Encoder, Decoder and Adversary, and uses three steps: graphic watermark embedding, model watermark embedding, and extraction reinforcement to complete the training of neural network. In order to verify the effectiveness of the framework, this article sets up three experiments: effectiveness verification, uniqueness verification, and robustness verification. The experimental results indicate that the model embedded with watermarks in this framework has certain robustness against some attacks against watermarks.*
Keywords: Model Copyright Protection, Black Box Scene, Model Watermark

1. Introduction.

In recent years, field of artificial intelligence has developed rapidly and has been used in various fields such as image processing and speech recognition, providing excellent and concise solutions to solve various problems in these fields. In the realm of smart city traffic network prediction, the integration of quantum genetic optimization with Learning Vector Quantization (LVQ) neural networks [1] has been demonstrated to significantly enhance the accuracy and efficiency of predictions. Within the field of medical image analysis, the application of transfer learning models employing sparse coding and deep learning [2] has effectively reduced the issue of false positives, thereby increasing the accuracy of lymph node detection.

Notably, Deep Neural Networks (DNNs) [3, 4] have been increasingly used and commercialized in various fields due to their powerful performance. However, training DNN model is a labor-intensive and time-consuming process. It requires a large amount of data and computing resources, and requires the programmer's efforts to further adjust network topology structure and set parameters correctly. The production resources consumed by these behaviors have extremely high value. However, artificial intelligence models are very easy to steal [5, 6], and in the future, more and more data information will be saved in the form of artificial intelligence models for completing various tasks. How to ensure the security of these artificial intelligence models and safeguard the rights and interests of model owners has become a problem that must be considered.

The process of training a model is quite difficult, while stealing a model is very simple. The value contained in these models is sufficient to attract a large number of malicious attackers to attack. These attackers often steal their functions and undergo simple modifications to make these models provide their own paid services, which is undoubtedly a property infringement on model owner. These attackers have many ways to steal models, such as model extraction attack proposed by Tramer et al. in 2016. This approach involves attackers iteratively sending data and viewing the corresponding response results to infer the parameters or functions of the machine learning model, thereby replicating a machine learning model with similar or even identical functions.

It is precisely in order to confront these attackers that Uchida et al. first proposed the concept of artificial intelligence model watermarking [7]. Watermarking is a way to claim copyright by embedding identity information into some original data without affecting the use of data. Model watermarking [8, 9], on the other hand, applies traditional digital watermarking techniques and ideas used to protect multimedia content to the field of artificial intelligence. Before publishing a model, the watermark is embedded in the model. In the event of disputes over the copyright of the model, the watermark information can be extracted from the model in a predetermined manner. The proposal of model watermarking has provided model researchers with a powerful tool to combat model theft, which is of great significance.

This paper proposes a copyright protection algorithm for image processing models, which uses three steps: graphic watermark embedding, model watermark embedding, and extraction reinforcement to complete the training of neural networks.

In order to verify the effectiveness of the framework, this article sets up three experiments: effectiveness verification, uniqueness verification, and robustness verification. The experimental results are relatively acceptable, indicating that the model embedded with watermarks in this framework has a certain degree of robustness against some attacks against watermarks.

2. **Realted work.**

2.1. **Model Watermarking in Black Box Scenes.**

In a black box scenario, model owner or validator is unaware of the internal structure and weights of suspicious target model, and can only access the target model through the application program interface (API) to obtain specific output for copyright verification. Usually, in this case, black box watermarking method selects specific samples to form a trigger set. When model classifies these specific samples into preset incorrect labels, it can be inferred that the model has been stolen. For example, for an image that is clearly 'cow', set it to the wrong specific label 'cat', create a trigger set for samples like this, and train the model together with the trigger set and ordinary samples. After the training is

completed, the model will classify these specific images into specific error labels. If these trigger sets are not trained, the model will correctly classify them as 'cows'. When there is a dispute over model ownership, if the model categorizes a large number of trigger sets into preset error labels, it indicates that the model was obtained by stealing someone else's model.

Method proposed by Merrer et al. [10] embeds watermarks by fine-tuning the pre-trained model, so that the boundaries of the classification area present the desired shape. More specifically, by stitching it around a set of inputs that correspond to a set of adversarial examples calculated on a pre-trained model, the desired shape is obtained.

Another method proposed by Zhang et al. [11] closely tracks watermarks through backdoor mode. The key image is generated by overlaying visible patterns (watermark triggering patterns) unrelated to the host image onto some training images. Then, by changing the original true class of the image, the images with patterns are re labeled and used to train the watermark vector to output the selected label in the presence of watermark triggering patterns. Three different ways of generating trigger patterns were proposed. To verify the existence of the watermark, owner inputs the key image into DNN and verifies whether the response matches the required key label.

In Guo et al. [12], key images are also generated by adding trigger patterns, but these patterns are invisible and can be seen as a method of labeling image subsets in the training set. The labeled image is assigned a set of predefined (possibly random) labels. DNN first trains on non-labeled images, and then applies fine-tuning to guide the network in classifying labeled images as needed. Due to the invisibility of the mark, the non-watermark model will continue to classify the marked image as a pre-trained model, while marked model will recognize the presence of the watermark and exhibit consistent behavior.

Image classification model is a neural network that completes the classification of images by processing the input images and mapping them to a certain category label. In this classification model, trigger set can be constructed by selecting some special images, so model can correctly classify most of images, meet the requirements, and have a certain degree of specificity. This is implementation method of most image classification watermarking methods. For those image processing models, both the input and output are images, but similar ideas to image classification models can also be used to embed model watermarks.

At present, research on model copyright protection mainly focuses on image classification model watermarking, while there is relatively little research in the field of image processing model watermarking, and it is not possible to simply apply watermarking methods of image classification models directly to image processing models. This project aims to consider a method that can embed watermarks into the output during image processing. Method of embedding watermarks should not be easily removed by attackers by directly deleting some modules, and should be integrated with the entire model to prevent model thieves from directly removing the watermark code after finding it to obtain a model without embedded watermarks.

2.2. **Image Watermarking Algorithm Based on Deep Learning.**

Some deep neural networks are very sensitive to small and difficult to distinguish changes in the input image with the naked eye, so it is possible to try using partial models for watermark embedding and extraction. Jiren Zhu et al. attempted to embed watermarks using the HiDDeN model. HiDDeN is a data hiding framework that uses three convolutional networks for data hiding. The first convolutional network attempts to hide data into the image, the second convolutional network attempts to extract hidden data from

the image, and the third convolutional network is a supervisor that attempts to extract hidden data from the image. This network ensures that the data hidden by the first convolutional network is not too obvious.

In model training, for encoder E_e, it receives carrier image I_{co} and binary message M_{in} with length L, generating I_{en} similar to I_{co}; The noise layer N receives I_{co} and noise L_{en} as inputs, generating a noisy image I_{no}; Decoder D attempts to reply with a message of M_{out} from I_{no}; The discriminator trains on a mixed dataset of I_{co} and I_{en}, and predicts a probability of I_{en}, which is called $A\bar{I}$.

For image distortion loss λ_I, there are: (S is the image size) as shown in Formula (1).

$$\lambda_I\left(I_{co}, I_{en}\right) = \left|I_{co} - I_{en}\right|_2^2 / S \tag{1}$$

For adversarial losses, as shown in Formula (2).

$$\lambda_G\left(I_{en}\right) = \log\left(1 - A\left(I_{en}\right)\right) \tag{2}$$

The classification loss caused by the discriminator's prediction as shown in Formula (3).

$$\lambda_A\left(I_{en}, I_{en}\right) = \log\left(1 - A\left(I_{co}\right)\right) + \log\left(A\left(I_{en}\right)\right) \tag{3}$$

This method takes advantage of the strong learning ability of neural networks, without the need to design the embedding method and embedding position of watermarks. The embedding and extraction of watermarks are completed by training encoders and decoders, and noise layers are used in training to enhance the robustness of the neural network against various attack methods.

Merrer et al. [10] propose a zero-bit model watermarking algorithm by generating adversarial examples. To achieve this, they fine-tune model through carefully adjusting decision boundaries to fit the adversarial examples. Adi et al. [13] generate trigger keys with abstract images that are far from each other and also the training data. All methods mentioned above focus on watermarking image classification models. But watermarking the image processing models is seriously under-researched. Quan et al. [14] propose the first model watermarking method for image processing networks. They watermark the target model by fine-tuning it with elaborately generated trigger images and verification images. Specifically, trigger images are sampled from i.i.d. uniform distribution. So trigger images are far from both each other and often-used training data. As a result, this method is robust to fine-tuning attacks. Corresponding verification images are constructed by a task-specific operation on the trigger images. To guarantee fidelity of the watermarking method, operation should be similar to function of the original model. At the same time, to distinguish watermarked model from the original one, the operation should also maximize the difference between verification images and output of the original. After model fine-tuning is done, they update verification images by output images of the fine-tuned model on the trigger images. One big limitation of this method is that such a method of generating trigger images can fail for many different tasks. For example, such trigger images contain no bone to remove if this method is applied to the task of Chest X-ray image de-bone [15]. In addition, it is too difficult for model owner to find such a suitable operation. Because two requirements for the operation are contradictory to some extent. Therefore this method fails to generalize under many different image processing tasks. Distinct from [13], Zhang et al. [11] start paying attention to protecting output images of image processing models. They argue that attacker can replicate target model by training a surrogate model on output images of target model. So they append an embedding sub-network behind target model, which conceals an invisible watermark in

output images of target model. And another extractor sub-network is trained to extract watermark out from output images of surrogate model.

It is worthy of reference and reference for future image watermarking methods based on deep learning.

3. **Our Proposed Method.**

We propose a universal neural network framework to embed watermarks in image processing models.

Set $M(\theta;\cdot)$ represents a trained image processing model, where θ is the model parameter of the model. This parameter is trained on the training set (X, Y), where X is the set of input images in the training set, and Y is the expected output of the original training data X after passing through the model, which is the reference for the model output, expressed as Formula (4).

$$M(\theta;\mathrm{X}) \approx \mathrm{Y} \tag{4}$$

If an attacker Bob illegally steals model M and deploys it in his own product, only providing APIs for users to use, for such a suspicious model M', the owner of model M can only attempt to verify their ownership with the output of the suspicious model M. At this point, if the output of the suspicious model M' contains watermarks, it can be proven the ownership of the model. At present, there are also ways to embed invisible watermarks in images. Therefore, our article considers designing a model watermarking method can embed watermarks, so that the watermark can be extracted in a specific way from the output image of the model. At the same time, watermark will not cause image distortion and will not affect model performance as much as possible. The embedding process is shown in Figure 1.

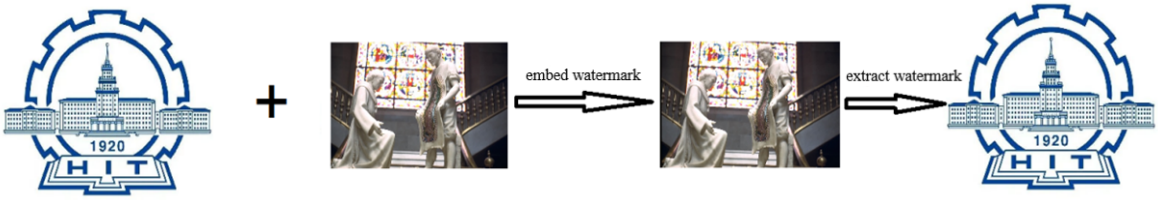

FIGURE 1. Image Watermark Embedding

In a black box scenario, we cannot obtain specific parameters within the model and can only verify model ownership based on the output of the target model. For image processing models, their output is an image, and if watermarks can be extracted from image, ownership verification can be completed. Relatively simple approach is to directly embed plaintext watermarks, i.e. unencrypted and visible simple watermarks, in the expected output image, and then use the expected output image and input image to construct a training set to complete the training of the model, as shown in Figure 2.

However, this type of plaintext watermark is easily removed by attackers using some simple graphic editing tools, such as Photoshop.

Therefore, this article needs to find a way to enable the trained model to embed invisible watermarks in the output image without causing a significant decrease in its performance on the original task.

For this, we firstly use an invisible watermark embedding method to embed watermark w into Y, and obtain Y'. Y' should maintain visual consistency with Y, as shown in Formula (5).

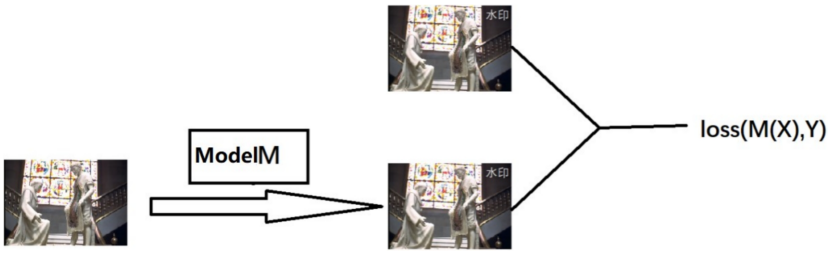

FIGURE 2. Plaintext Watermark Embedding

$$Y' \approx Y \tag{5}$$

Afterwards, we train the model based on the training data (X, Y') after embedding the watermark w, in order to embed the watermark into the model and obtain the model $M(;\cdot)$ after embedding the watermark w, where is the model parameter after embedding the watermark. Therefore, for the input X, the output of the model $M(;\cdot)$ should approach Y', as shown in Formula (6).

$$M\left(\theta'; X\right) \approx M(\theta; X) \tag{6}$$

In this way, the model $M(;\cdot)$ obtained by embedding watermark w can embed the watermark w in its own output image while ensuring its performance is not affected. In this way, if the model $M(;\cdot)$ is stolen by the attacker Bob, the owner of the model can extract the watermark from the output of the stolen model in a specific way to prove their ownership of the model.

3.1. **Framework Implementation.**

We need to solve two problems: 1. Embedding invisible watermarks w; 2. Extracting watermarks from output images.

In order to achieve the watermark embedding of the Black Box model mentioned above, this paper designs a model training strategy based on DNN, which is divided into three stages: image watermark embedding, model watermark embedding, and extraction reinforcement. To complete the training of the model and obtain an Encoder for embedding invisible watermarks into the model; A decoder for extracting watermarks from the output image, as shown in Figure 3.

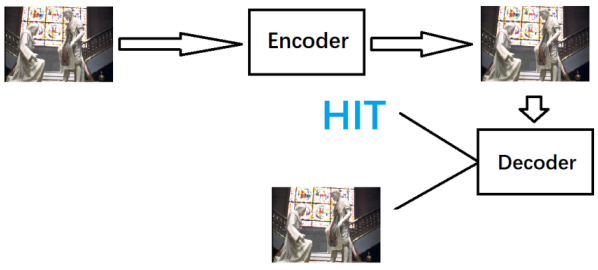

FIGURE 3. Framework of Embedding and Extracting Watermarks

At this stage, we need to train to obtain the watermark embedder Encoder. We process the watermark and training set X as inputs to the Encoder, then use another model Decoder as the watermark extractor and set up a supervisor, Adversary. Task of Encoder is to embed watermarks in the GT (Ground Truth) image, which refers to the model output reference Y mentioned earlier, while ensuring visual invisibility. Task of the Decoder is to

extract watermark information from the embedded image. Note that the Decoder should also ensure that watermark information can be extracted from the image embedded with the watermark, and the watermark cannot be mistakenly extracted from the image without the watermark. Task of Adversary is to supervise the correct completion of task by the Encoder, which can identify differences between watermarked images and the original image. Through continuous training, the encoder is ultimately equipped with the ability to embed invisible watermarks.

To ensure embedding of watermark does not affect the visual consistency between embedded watermark image and original image, it is necessary to minimize unnecessary information in the watermark. A normal RGB image contains values of three channels: red, green, and blue, while in a grayscale image, the values of the three channels are equal, so only one byte can store the pixel values stored in the original RGB image. Therefore, when embedding the watermark, we use a grayscale image containing only one channel as input of Encoder, and also use the three channel values of the original image as the input of the Encoder. In this way, the image volume of the watermark image is smaller, which can better avoid the impact of watermark embedding on the visual effect of the original image.

This article uses three loss functions to measure the visual similarity between the original image Y and embedded watermark image Y'. They are image loss, perceptual loss and adversarial loss. Their respective algorithms are shown in Formulas (7), (8), and (9):

$$L_{img} = \|Y' - Y\|^2 \tag{7}$$

$$L_{vgg} = \|VGG\,(Y') - VGG(Y)\|^2 \tag{8}$$

$$L_{adv} = \log\,(1 - A\,(Y')) \tag{9}$$

Therefore, the total embedded loss is shown in the Formula (10).

$$L_E = \lambda_{img} L_{img} + \lambda_{vgg} L_{vgg} + \lambda_{adv} L_{adv} \tag{10}$$

Similarly, when the extractor Decoder extracts watermark image Y' with watermark and watermark image Y without watermark, loss function defined in this article is shown in Formula (11).

$$L_D = \|D\,(Y') - w\|^2 + \|D(Y) - c\|^2 \tag{11}$$

Among them, the above c is a blank image as a control.

In summary, the loss function used for both Encoder and Decoder training in this article is shown in Formula (12).

$$L_E = \lambda_{img} L_{img} + \lambda_{vgg} L_{vgg} + \lambda_{adv} L_{adv} + \lambda_D L_D \tag{12}$$

3.2. **Model watermark embedding.**

At this stage, we obtain (X, Y') by embedding watermarks into the training data (X, Y) using the Encoder trained in the previous stage, and train the target model using the training data after embedding the watermarks. After training, the target model can perform image processing on the output image without affecting its original performance, while also completing the task of embedding watermarks, as shown in Figure 4.

We further train extractor Decoder to enhance its ability to extract watermarks. The training data is used as original model $M(\theta; X)$ and model after embedding the watermark $M(\theta'; X)$, respectively, to obtain output image without embedding watermark $\tilde{X}$ and the output image with embedding the watermark $\tilde{X}'$. We mix them with original GT image set Y and the watermarked GT image set Y' as inputs to the Decoder. We require the

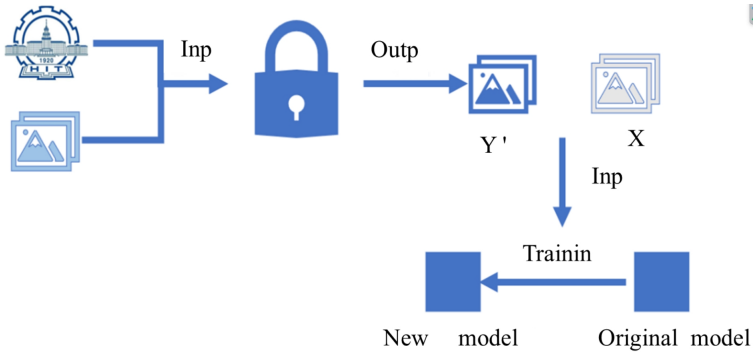

FIGURE 4. Model Watermark Embedding Process

Decoder to extract blank image c in Y and watermark image w in Y' to enhance its extraction ability. The loss functions set during training are as shown in the following four Formulas (13), (14), (15) and (16).

$$L_{RD} = \lambda_{wd} L_{wd} + \lambda_{bid} L_{bid} + \lambda_c L_c \tag{13}$$

$$L_{wd} = \|D(Y') - w\|^2 + \left\|D\left(\tilde{X}'\right) - w\right\|^2 \tag{14}$$

$$L_{bid} = \|D(Y) - c\|^2 + \|D(\tilde{X}) - c\|^2 \tag{15}$$

$$L_c = \left\|D(Y') - D\left(\tilde{X}'\right)\right\|^2 \tag{16}$$

Among them, L_{wd} is watermark distortion loss, representing $L2$ distance between w and the information extracted from watermark Y' and $\tilde{X}'$, L_{bid} is the blank distortion loss, representing $L2$ distance between the blank image and the information extracted from Y and $\tilde{X}$, L_c is watermark consistency loss, representing the $L2$ distance of the information extracted from Y' and $\tilde{X}'$.

This stage will not affect the already trained Encoder and target model, only to enhance the Dncoder's ability to correctly extract watermarks from the output image of the target model, as shown in Figure 5.

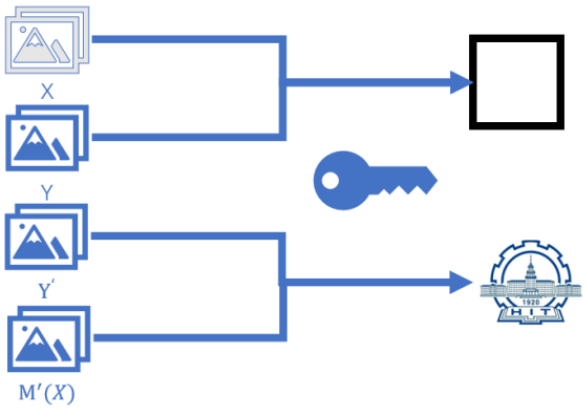

FIGURE 5. Strengthening Process of Watermark Extraction

Model ownership verification is to ensure the rights of the model owner. In reality, when a suspicious model M_s appears, if the model only provides an API interface and cannot obtain its internal parameters, we can only verify ownership through the output of model. When conducting ownership verification, we first randomly select images to

form the trigger set Z, and then input each image of the trigger set Z into the suspicious model's API interface to obtain the output $M_s(Z)$. From this output, we use Decoder to extract watermark information, and calculate the watermark extraction success rate SR based on the success of extraction, as shown in Formula (17).

$$SR = \frac{1}{k} \sum_{zi \in Z} \mathrm{f}(zi) \tag{17}$$

$$(zi) = \begin{cases} 1, NCC\left(D\left(M_s(zi)\right), w\right) > 0.95 \\ 0, \text{ otherwise} \end{cases} \tag{18}$$

where $NCC(D(M_s(zi)), w)$ is the normalization coefficient value between the extracted watermark and the original watermark. The closer the value is to 1, the smaller the difference between the two. Therefore, when value is greater than 0.95, we consider the watermark extraction successful. If the success rate of watermark extraction is high, we can assume that the suspicious model was stolen from the model owner, thus completing the copyright protection of the image processing model.

3.3. **Experimental Setup.**

This paper uses VDSR as the target model for training and embeds watermarks in it. VDSR proposed a model for completing super-resolution tasks by Kim et al. [10] in 2016. It uses a very deep network to complete super-resolution tasks, optimizes and accelerates network training using methods such as residual learning, and achieves good results. The structure of the network is shown in Figure 6.

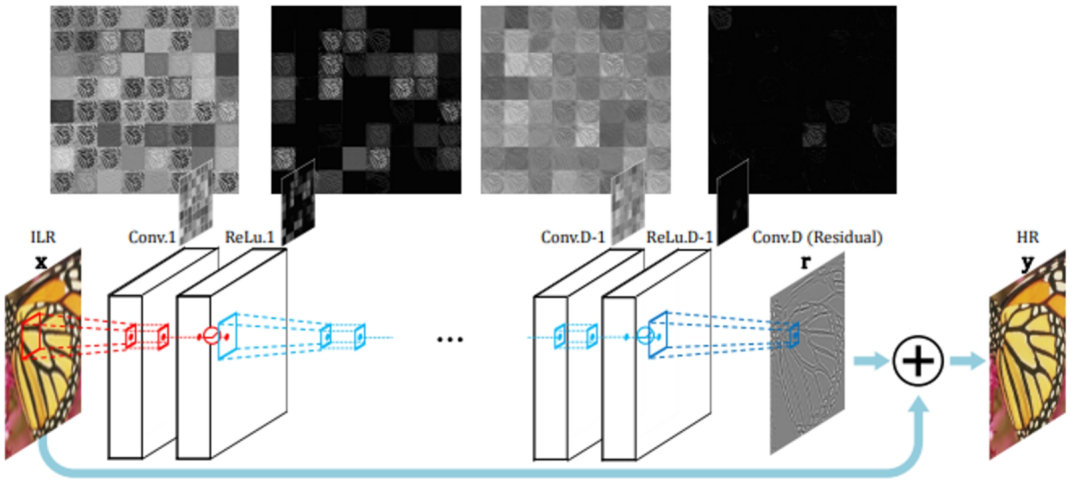

FIGURE 6. VDSR Model Structure

We use 291 images used in the original experiment, 200000 small images were cropped to obtain the real image Y. Then, each image was zoomed in and out (i.e. upsampling and downsampling) to artificially reduce the resolution of the images and obtain image X. During training, X is the processed image, while Y is the GT image.

In terms of model structure selection for Encoder and Decoder, this framework follows the strategy of Zhang et al., using Unet as the Encoder, CEILNet as the Decoder, and PatchGAN as the Adversary.

4. **Experimental Results.**

4.1. **Fidelity Experiment.**
Tested on three datasets, Set5, Set12, and BSD100, with 5, 14, and 100 images, commonly used in super-resolution task testing. Use Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) as evaluation indicators. For the real image I of $m \times n$ and the output image I' of model, PSNR is defined as Formula (19):

$$\text{PSNR} = 10 \log_{10} \frac{\left(2^d - 1\right)^2 * m * n}{\sum_{i=1}^{m} \sum_{j=1}^{n} \left(I(i,j) - I'(i,j)\right)^2} \tag{19}$$

Among them, $(2^d - 1)$ is the maximum pixel value of an image, which is the amplitude. SSIM can be defined as Formula (20).

$$\text{SSIM}\left(I, I'\right) = \frac{\left(2\mu_I \mu_{I'} + c_1\right)\left(2\sigma_{II'} + c_2\right)}{\left(\mu_I^2 + \mu_{I'}^2 + c_1\right)\left(\sigma_I^2 + \sigma_{I'}^2 + c_2\right)} \tag{20}$$

where μ_I and $\mu_{I'}$ represent the mean of I and I', σ_I and $\sigma_{I'}$ represent the standard deviation of I and I', $\sigma_{II'}$ is the covariance between I and I', c_1 is defined as $(k_1(2^d - 1))^2$, c_2 is defined as $(k_2(2^d - 1))^2$, and k_1, k_2 are far less than 1.

The final results are shown as in the following Table 1.

Table 1. Fidelity Test Results for VDSR

Model	PSNR/SSIM		
	Set5	Set14	BSD100
VDSR [8]	30.87/0.9381	27.45/0.8607	27.09/0.8303
VDSR$^{\text{w}}$	30.91/0.9388	27.55/0.8627	27.13/0.8302

Among them, VDSR is the model without embedding [16] the watermark, and VDSR$^{\text{w}}$ is the model with embedding watermark. It can be seen that before and after embedding the watermark, the peak signal-to-noise ratio and structural similarity are equivalent, and the difference in PSNR does not exceed 0.07dB. It can be judged that there are no changes that affect the performance of the model before and after embedding the watermark, and the fidelity of the model is guaranteed.

4.2. **Uniqueness Experiment.**
When verifying model ownership, we need to ensure that the suspicious model is the one we are looking for, embedded with watermarks, and not other models used to complete the same task. Therefore, by mixing the output of the embedded VDSR model, DPSR, IMDN, ESRGAN models that complete similar tasks, as well as the VDSR model without embedding the watermark, the experimental Decoder can correctly extract the watermark from them. We require it to be able to extract watermarks from the output images of VDSR and blank images from the output images of the other four models. Based on this, record the success rate of watermark extraction.

The final results can be shown in the following Table 2.

Table 2. VDSR Uniqueness Experiment Results

Decoder	Watermark Extraction Success Rate				
	VDSR$^{\text{w}}$	VDSR [16]	DPSR	IMDN	ESRGAN
D_{VDSR}	100%	0%	2.89%	3.02%	0%

It can be seen that the success rate of watermark extraction in Decoder fully meets the requirements. In fact, there are also examples of extracting fuzzy watermarks from images

that have never been embedded with watermarks. However, in this case, the watermarks are fuzzy and rare, which does not affect our ability to distinguish model ownership.

4.3. **Robustness Experiment.**

This experiment mainly focuses on the robustness of the experimental model to model pruning. Model Pruning refers to the technique of modifying and optimizing neural network models to achieve the goals of reducing the number of model parameters, reducing model size, improving model operation efficiency, and reducing model storage space. Model pruning technology can be used to reduce the computational complexity of the model, improve inference speed of the model, reduce the storage requirements of the model, etc., so that the model can run on smaller devices or perform inference in slower environments. Commonly used model pruning methods include structural pruning, channel pruning, weight pruning, etc.

Pruning method of experiment is to set parameter with a certain proportion of the minimum absolute value of each layer in the model to 0, and this proportion increases from 10% to 90%, which is called pruning rate. The results are shown in Table 3. The algorithm used for comparison comes from reference [17].

Table 3. VDSR Robustness Experimental Results

Model pruning rate (%)	Watermark extraction success rate D_{VDSR}	
	Ours	Reference [15]
10	100%	99%
20	100%	97%
30	100%	95%
40	99%	92%
50	97%	89%
60	87%	77%
70	39%	29%
80	24%	11%
90	5%	3%

It can be seen that as the pruning rate increases, the success rate of extraction continues to decrease. Fortunately, in actual situations, the pruning rate is greater than 40%, and the model is basically distorted, making it difficult to complete the model function. Therefore, within this range, the success rate of watermark extraction is still close to 100%, which is sufficient to complete the watermark task.

5. **Conclusion.**

This paper analyzes the scenario of copyright verification of image processing models in black box scenarios. Starting from the conditions required to complete model copyright verification in this scenario, it analyzes and summarizes a method that can complete watermark embedding in black box scenarios, which can complete ownership verification of the model. Starting from this, a framework was designed to accomplish this task, which includes three roles: Encoder, Decoder, and Adversary, each completing their respective tasks. This enables the target model to autonomously embed invisible watermark information in its output image without affecting its own performance. Use of fidelity, uniqueness, robustness as evaluation criteria has demonstrated the feasibility of this framework.

Acknowledgment.
This work is supported by school research of Laiwu Vocational and Technical College named Application of Wavelet Domain Features in Facial Recognition Algorithm Research(2021qnzx05). And it is also supported by Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies (2022B1212010005). This research is also supported by Key Basic Research Projects of Shenzhen with Grant No. JCYJ20220818102414030.

REFERENCES

[1] F. Zhang, T.-Y. Wu, Y. Wang, R. Xiong, G. Ding, P. Mei, L. Liu, "Application of Quantum Genetic Optimization of LVQ Neural Network in Smart City Traffic Network Prediction," *IEEE Access*, vol. 8, pp. 104555-104564, 2020.

[2] Y. Ma, Y. Peng, T.-Y. Wu, "Transfer Learning Model for False Positive Reduction in Lymph Node Detection via Sparse Coding and Deep Learning," *Journal of Intelligent & Fuzzy Systems*, vol. 43, no. 2, pp. 2121-2133, 2022.

[3] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770-778.

[4] K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks for Largescale Image Recognition," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 1409-1556.

[5] M. Juuti, S. Szyller, S. Marchal, N. Asokan, "PRADA:protecting against DNN model stealing attacks," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 512-527.

[6] F. Tramèr, F. Zhang, A. Juels, M.-K. Reiter, T. Ristenpart, "Stealing Machine Learning Models via Prediction Apis," in *25th USENIX Security Symposium (USENIX Security 16)*. USENIX, 2016, pp. 601-618.

[7] Y. Uchida, Y. Nagai, S. Sakazawa, S. Satoh, "Embedding Watermarks into Deep Neural Networks," in *2017 ACM on International Conference on Multimedia Retrieval*. ACM, 2017, pp. 269-277.

[8] J. Wang, H. Wu, X. Zhang, Y. Yao, "Watermarking in Deep Neural Net-works via Error Backpropagation," *Electronic Imaging*, vol. 32, pp. 1-9, 2020.

[9] T. Wang, F. Kerschbaum, "Attacks on Digital Watermarks for Deep Neural Networks," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2622-2626.

[10] E. Le Merrer, P. Perez, G. Trédan, "Adversarial frontier stitching for remote neural network watermarking," *Neural Computing and Applications*, vol. 32, pp. 9233-9244, 2019.

[11] J. Zhang, Z. Gu, J. Jang, H. Wu, M.-P. Stoecklin, H. Huang, I. Molloy, "Protecting Intellectual Property of Deep Neural Networks with Watermarking," in *ACM Asia Conference on Computer and Communications Security*. ACM, 2018, pp. 159-172.

[12] J. Guo, M. Potkonjak, "Watermarking Deep Neural Networks for Embedded Systems," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE/ACM, 2018, pp. 1-8.

[13] Y. Adi, C. Baum, M. Cisse, B. Pinkas, J. Keshet, "Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring," in *27th USENIX Security Symposium (USENIX Security 18)*. USENIX, 2018, pp. 1615-1631.

[14] Y. Quan, H. Teng, Y. Chen, H. Ji, "Watermarking Deep Neural Networks in Image Processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 5, pp. 1852-1865, 2020.

[15] W. Yang, Y. Chen, Y. Liu, L. Zhong, G. Qin, Z. Lu, W. Chen, "Cascade of Multi-Scale Convolutional Neu- ral Networks for Bone Suppression of Chest Radiographs in Gradient Domain," *Medical Image Analysis*, vol. 35, pp. 421-433, 2017.

[16] J. Kim, J.-K. Lee, K.-M. Lee, "Accurate Image Super Resolution Using Very Deep Convolutional Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 1646-1654.

[17] S. Kakikura, H. Kang, K. Iwamura, "Deep Learning Model Protection Using Negative Correlation-based Watermarking with Best Embedding Regions," in *2023 25th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2023, pp. 1438-1445.