Computers, Materials & Continua
DOI: 10.32604/cmc.2023.033700
Article

Tech Science Press

Check for updates

A Dual Model Watermarking Framework for Copyright Protection in Image Processing Networks

Yuhang Meng[1], Xianyi Chen[1,*], Xingming Sun[1], Yu Liu[1] and Guo Wei[2]

[1]Engineering Research Center of Digital Forensics, Ministry of Education, Nanjing University of Information Science and Technology, Nanjing, 210044, China
[2]The University of North Carolina, Pembroke, 31321, USA
*Corresponding Author: Xianyi Chen. Email: 0204622@163.com
Received: 24 June 2022; Accepted: 04 August 2022

Abstract: Image processing networks have gained great success in many fields, and thus the issue of copyright protection for image processing networks has become a focus of attention. Model watermarking techniques are widely used in model copyright protection, but there are two challenges: (1) designing universal trigger sample watermarking for different network models is still a challenge; (2) existing methods of copyright protection based on trigger s watermarking are difficult to resist forgery attacks. In this work, we propose a dual model watermarking framework for copyright protection in image processing networks. The trigger sample watermark is embedded in the training process of the model, which can effectively verify the model copyright. And we design a common method for generating trigger sample watermarks based on generative adversarial networks, adaptively generating trigger sample watermarks according to different models. The spatial watermark is embedded into the model output. When an attacker steals model copyright using a forged trigger sample watermark, which can be correctly extracted to distinguish between the piratical and the protected model. The experiments show that the proposed framework has good performance in different image segmentation networks of UNET, UNET++, and FCN (fully convolutional network), and effectively resists forgery attacks.

Keywords: Image processing networks; copyright protection; model watermark

1 Introduction

In recent years, deep learning [1] has revolutionized the research of image processing fields, such as image classification [2–4], medical image segmentation [5,6], and object detection [7–9]. To fully excavate the learning capability of deep models, vast amounts of labeled data and high-efficiency computational resources are often required. However, model distribution on the Internet may suffer from attacks such as forgery attacks, model pruning [10], and watermark rewriting, so it is crucial to

protect the model copyright using the digital watermarking techniques [11–13], which is named model watermarking in a neural network.

The model watermarking methods could be divided into white-box and black-box cases according to whether the model details are needed to be known during the processing of watermark embedding. In the white-box case, the watermark is embedded into the to-be-protected network by modifying its structure or weights, so the full network information is needed to know, including the details of structure and weights. This case is generally used in neural networks with public model information. The advantage is that can withstand attacks such as model fine-tuning and pruning.

In the black-box case, a pair of input and output images are constructed as trigger sample watermarks, the input image is called the trigger key, and the preset output image is called the verification key. The advantage is that validation only needs to input trigger keys to verify the model copyright. Currently, there are two classes of methods for trigger key generation: (1) adding relevant images to the training set [14] and (2) modifying the input to obtain the trigger image by adjusting the network, such as embedding visible or invisible text, symbols, patterns or noise, etc. in the trigger key [15,16]. However, the above trigger key generation methods are often applied to the image classification networks and don't able to migrate to image processing models efficiently. Moreover, the trigger keys couldn't be generated adaptively according to different models.

Recently, some research work began to focus on copyright protection of image processing networks with commercial value. In real scenarios, the image processing models that are exposed are almost always served end-to-end to the user, who can only access the input and output sides of the model. In the previous approach [17,18], forgery attacks were not considered. Therefore, a trigger sample watermarking method that is resistant to forgery attacks is essential.

In this paper, we propose a new end-to-end dual model watermarking framework for the copyright protection of image processing networks. A pair of special input and output is embedded in the model training as a trigger sample watermark, while a spatial watermark is embedded into the output of the model. The trigger sample watermark is automatically generated using a method based on generative adversarial networks (GANs) [19], which adaptively is generated according to the protected models. In the validation process, enter a trigger key to the model to obtain the output, and compare the consistency between the output image and the preset validation key. When an attacker steals a model copyright using forged input and output, the spatial watermark can be correctly extracted from the output of the model to distinguish between the piratical and the protected model. Specifically, we design a location key to share in the process of embedding and extracting spatial watermarks. In the verification process, the location key is used to extract the watermark from the output of the model to prove the ownership of the model. On the contrary, the attacker does not know the location key, so the watermark cannot be extracted correctly. Experiments show that the proposed framework has good performance in image segmentation networks with UNET, UNET++, and FCN, while almost not affecting the overall segmentation accuracy and the number of parameters of the model. And proposed framework can effectively withstand forgery attacks.

The contributions of this paper are as follows:

- Due to the limitation of the application of the trigger sample watermark generated by the traditional method to the image processing network, we innovatively design a universal trigger sample watermark generation method, which can adaptively generate the trigger sample watermark according to different models.
- We embed a spatial watermark into the output of the model to better resist the forgery attack.

- Experiments show that the proposed framework has good performance in image processing networks, while almost not affecting the overall network accuracy and the number of parameters of the model.

The rest of the paper is structured as follows. We summarize the related work in Section 2. The proposed approach is described in detail in Section 3, followed by extensive experiments and analysis in Section 4. Finally, we conclude the paper and provide further discussion in Section 5.

2 Related Works

In the white-box case, where embeds and extracts the watermark by modifying the internal parameters or the structure of the model. Uchida et al. proposed adding regularizes to the loss function to control the distribution of weights, embedding the watermark into the weights of the model during its training [20]. The activation approximation of the intermediate layer of the neural network obeys a Gaussian mixture distribution. Based on it, Rouhani et al. embed a watermark in the activation distribution of the intermediate layer, which can effectively withstand model fine-tuning [21].

In the black-box case, where uses the specificity of the output results of the trigger key in the model to protect the copyright of the model. In recent works, a variety of methods for generating trigger keys have emerged. For example, Adi et al. first proposed using a neural network backdoor with an abstract image different from the training set as a trigger key to assign a label to it, constructing such a set of mapping relations that can then be used to protect the model copyright [14]. The scheme of Zhang et al. uses specific processing of some images from the training set so that they correspond to predefined labels [16]. Li et al. improved the trigger key generation method so that an invisible watermark is hidden on the images in the training set as a trigger key input to the model to obtain a predefined label that proves the copyright of the model [15]. The trigger keys are almost indistinguishable in appearance from the images in the original training set, which has the advantage of being more stealthy and less detectable by attackers than the previous method. However, there is a deficiency in that trigger keys depend on the manual design and cannot generate trigger keys adaptively according to different models in the existing methods. When an attacker forges a pair of inputs and outputs to confuse the model copyright, it is called a forgery attack. Xinpeng Zhang et al. [22] introduce one-way hash functions to solve the above problem and construct a matching relationship between trigger keys and labels. An attacker without network training rights would not be able to forge a set of trigger keys and labels, thereby not being able to obfuscate ownership.

In addition to the mentioned black-box and white-box cases, some research works have started to focus on the copyright protection of image processing networks. Jie Zhang proposed training a watermark embedding and extraction network outside of the original model such that each input image is embedded with a watermark while completing the image processing task [17]. Compared with the previous method, Wu et al. increased a key mechanism to ensure model security [18]. Adding a watermark embedding network increases the security and robustness of the model, but the increasing number of parameters makes the training image processing network much more complex and expensive than before. Moreover, a forgery attack is not considered in the copyright protection of an image processing network.

3 Proposed Method

In this section, the details will be elaborated on. We propose a new end-to-end dual model watermarking framework for copyright protection of image processing networks shown in Fig. 1. The

following will be divided into two parts: (1) the generation, embedding, and verification of trigger sample watermark and (2) the generation, embedding, and verification of spatial watermark.

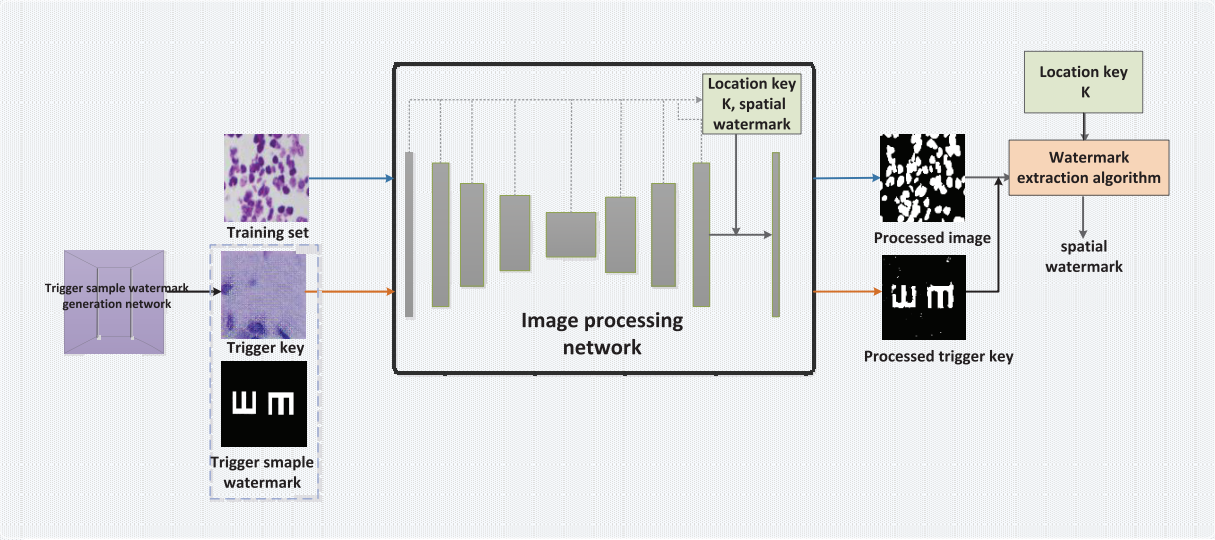

Figure 1: An end-to-end dual model watermarking framework

3.1 Trigger Sample Watermark

In the trigger sample watermark, the input image is called the trigger key, and the preset output image is called the verification key. Since the verification key is preset and does not need to be generated, the generation method of the trigger sample watermark is equivalent to the generation method of the trigger key in the following. Due to the limitation of the application of the trigger key generated by the traditional method to the image processing network, we innovatively design a universal trigger key generation method, which can adaptively generate trigger key watermark according to different models.

3.1.1 Trigger Key Generation

An adversarial example is made by intentionally adding subtle perturbations to the training example that are imperceptible to the human eye so that the model gives a high confidence error output. In traditional classification networks, there are two main methods to generate trigger keys by using adversarial examples: (1) Adding relevant images to the training set [14], as shown in Fig. 2b: and (2) directly modifying the images in the training set, such as embedding visible or invisible text, symbols, patterns, etc. [15] or adding noise [16], as shown in Figs. 2c–2f.

However, since the output of an image processing network is a processed image, the traditional trigger key generation methods in classification networks are not well migratory and rely on manual design, and cannot generate trigger keys adaptively according to different models. To this end, we propose a trigger key generation method using GAN for application to image processing networks, as shown in Fig. 3.

The network consists of three main components: the generator G, the discriminator D, and the image processing model M to be protected. Assume the original training set is X. The generator G takes X as input and generates the perturbation $G(X)$. And then sends $X' = G(X) + X$ into the

discriminator D to distinguish the original training set X from the trigger key X'. The two are iteratively trained to continuously optimize the generator and discriminator until the discriminator D is unable to distinguish between X and X', thus obtaining the optimal trigger key X'.

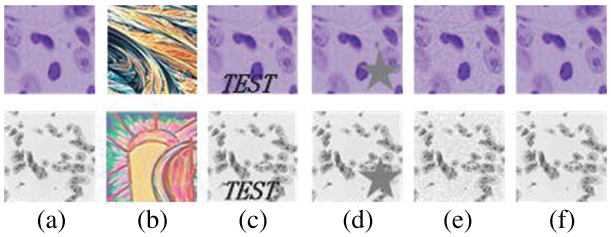

Figure 2: Traditional trigger key generation methods: (a) original image (b) abstract image (c) image with text (d) image with pattern (e) image with noise (f) hide an invisible logo

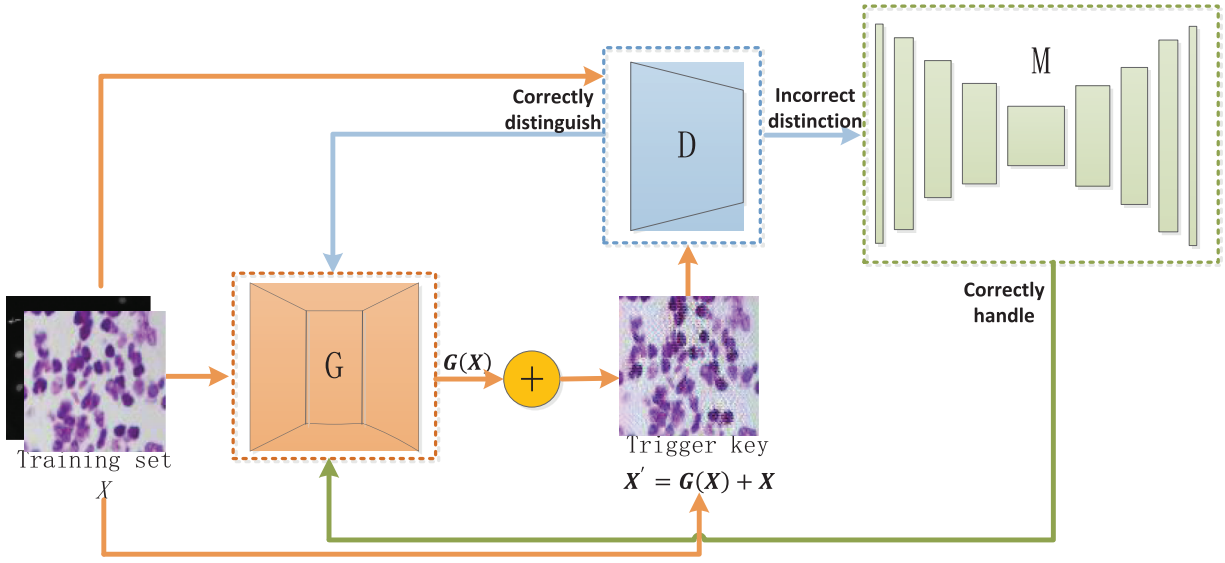

Figure 3: Trigger key generation network

It is not enough to train the generator G and the discriminator D to distinguish X and X' inputs into the image processing network. For this purpose, we add the image processing network M to the training of trigger key generation as well so that the trigger keys X' are generated adaptively according to the different image processing networks. To distinguish the output results of X and X' in M, the objective matrix Q is designed according to different image processing tasks such that the output results of X' in M are infinitely close to the objective matrix Q. The distance between the output obtained by inputting X' into M and the target matrix Q is fed back to the generator G to obtain the loss function as $\mathcal{L}^{adv}$:

$$\mathcal{L}^{adv} = \mathbb{E}_X \ell(M(X'), Q) \tag{1}$$

The design criterion for the target matrix is to make the difference between Q and the ground truth in the image processing task as large as possible. In Eq. (1), ℓ is the cross-entropy loss function.

In this paper, Q is designed as:

$$Q = \begin{bmatrix} 1 & \cdots & 1 & \cdots & 1 \\ 0 & \cdots & 0 & \cdots & 0 \\ 1 & \cdots & 1 & \cdots & 1 \\ 0 & \cdots & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}_{n \times n} \tag{2}$$

To improve the speed of network training and the generality of the proposed method, we refer to generative adversarial networks as AdvGAN [23]. Its adversarial loss function $\mathcal{L}^{GAN}$ [19] can be written as:

$$\mathcal{L}^{GAN} = \mathbb{E}_X log\ D(X) + \mathbb{E}_X \log(1 - D(X')) \tag{3}$$

To limit the magnitude of the perturbation, we add a hinge loss [24] as:

$$\mathcal{L}^{hinge} = \mathbb{E}_X max(0, \|G(X)\|_2 - c) \tag{4}$$

where c denotes a user-specified bound that also stabilizes the training of the GAN. [25] Finally our overall loss can be expressed as:

$$\mathcal{L} = \alpha \mathcal{L}^{GAN} + \beta \mathcal{L}^{adv} + \gamma \mathcal{L}^{hinge} \tag{5}$$

where α, β, and γ are the hyperparameters to balance these three loss terms. The goal of $\mathcal{L}^{GAN}$ is to generate trigger keys that are more similar to the original training set, while the goal of $\mathcal{L}^{adv}$ is to make the outputs obtained when the trigger keys and the original training set go through the image processing network significantly different. A comparison of the trigger keys generated using the method proposed in this paper with the images in the original training set is shown in Fig. 4.

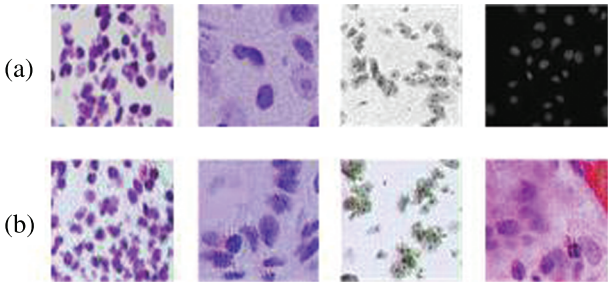

Figure 4: Comparison of the generated trigger keys with the images in the original training set: (a) images in the original training (b) the generated trigger keys

3.1.2 Trigger Sample Watermark Embedding

We use the method proposed in the previous section to obtain trigger keys, set the number of trigger keys X' to be 1% of the original training set X, and put them into the original training set to fine-tune the image processing network M, so that the trigger keys X' and the verification keys Y' form a set of mapping relationships. To make $M(X')$ close to the set verification key Y', fine-tune M according to the following loss function:

$$\mathcal{L}_m = \mathcal{L}_{org} + \lambda \mathcal{L}_w \tag{6}$$

where $\mathcal{L}_{org}$ is the loss of the original task of model M, $\mathcal{L}_w$ is the loss of the watermark, and λ is used as the adjustment coefficient to ensure that there is no impact on the accuracy of the original model M.

$$\mathcal{L}_{org} = \sum_{x_i \in X, y_i \in Y} \|M(x_i) - y_i\|^2 \tag{7}$$

where Y denotes the ground truth, and $\mathcal{L}_{org}$ aims to make $M(X)$ closer to Y.

$$\mathcal{L}_w = 1 - ssim\left(M(x_i') - y_i'\right) \tag{8}$$

The purpose of using the *ssim* function in Eq. (8) is to make the trigger key X' increasingly similar to the verification key Y' during the model training process to achieve watermark embedding. To ensure that $\mathcal{L}_w$ does not affect the accuracy of model training, the value of λ in $\mathcal{L}_m$ is taken as 0.4.

3.1.3 Trigger Sample Watermark Verification

For the verification process of the trigger sample watermark, as shown in Fig. 5, the trigger key X' is input into the model M, and the obtained $M(X')$ compares the similarity with the previously predefined verification key Y'. To better measure the similarity of $M(X')$ and Y', the *ssim* function is invoked for comparison, as shown in Eq. (9).

$$score = ssim(M(X'), Y') \tag{9}$$

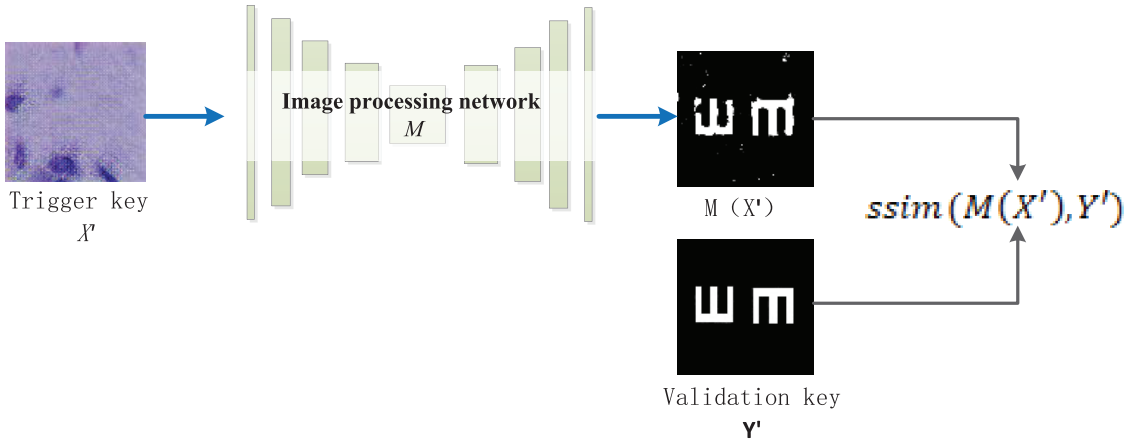

Figure 5: Trigger key verification process

The more similar the two images are, the closer the score value is to 1. Set the threshold value as η, prove that model M contains watermark when $score > \eta$, and obtain $\eta = 0.7$ by experimental verification.

3.2 Spatial Watermark

3.2.1 Spatial Watermark Generation and Embedding

A spatial watermark is embedded into the output of the model, which is generated by a random binary bitstream, defined as follows:

$$W = \{w_i | w_i \epsilon \{0, 1\}, i = 0, 1, 2, \ldots\} \tag{10}$$

We design a location key to select the location of the spatial watermark embedding. If location is fixed, it is vulnerable to attack. Ensure the randomness of the location key K so that it is closely related to the input of each image. Even if the attackers know that the output image contains a watermark, they cannot extract the watermark from it because they do not know the location key K. The spatial watermark embedding process is summarized in Algorithm 1.

Algorithm 1: Embedding of spatial watermark

Input: **Target model**(M); **Training data**(X); **watermark** (w_i)

Output: **Watermarked model**(M'); **Location key**(K)

l_i, l_L: ith and last layers of the model

①Watermark transformation:
$w_i' \leftarrow$ **trans_w** (w_i)

②Location key K generation:
for i in range($L-1$): **do**

$k_i \leftarrow$ **S_location** (X_i); $K \leftarrow k_i$

③Embedding of watermark
$X_L \leftarrow$ **embed_w** $\left(X_{L-1}, w_i', K\right)$

Summary of Algorithm 1: In the watermark transformation, we embed the watermark w_i into the model, and since the values 0 and 1 have a large impact on the image, it is transformed into a value suitable for embedding by the watermark transformation function (**trans_w**), which is **trans_w** (w_i):

$$w_i' = \textbf{trans\_w}\ (w_i) = \begin{cases} -0.95 & if \quad w_i = 1 \\ -1 & if \quad w_i = 0 \end{cases} \tag{11}$$

In the generation of the location key K, the location function **S_location** (X_i) is expressed as:

$$k_i = \frac{1}{m}\sum_{k=1}^{m}\sqrt{\frac{\sum_{j=1}^{n}\left(X_{ij}{}^{k} - \overline{X_i^k}\right)^2}{n-1}} \tag{12}$$

The location key K consists of k_i, and in Eq. (12) x_i represents the input of the image in the i-th layer of the model, and x_i is input into the location function (S_loaction) to obtain k_i. Where n denotes the image size, m denotes that X_i has m feature maps in i-th layer of the model and j denotes each element in X_i. Since k_i is the location of the watermark embedding, its size cannot exceed the image size n.

Finally, w_i' is embedded into X_{L-1} according to the location key K.

3.2.2 Spatial Watermark Verification

The algorithm for extracting spatial watermark is summarized in Algorithm 2. The output of model M is *Image*, and the spatial watermark is extracted using the extraction function **extract_w** (*Image*).

Algorithm 2: Watermark extraction

Input: *Image*, K

Output: w_i

l_i, l_L: ith and last layers of the model

①Watermark extraction:

Image $\leftarrow M(X)$ **or** $M(X')$; $w_i' \leftarrow$ **extract_w**$(Image, K)$

② Inverse watermark transform:

$w_i \leftarrow$ **trans_w**$'\left(w_i'\right)$

Summary of Algorithm 2: In watermark extraction, the location key K is shared in the embedding and extraction process of the spatial watermark. *Image* is the training set of images X input to model M to obtain $M(X)$ or trigger key X' input to model M to obtain $M(X')$. K and *Image* are used as inputs to the extraction function (**extract_w**) to obtain w_i'. Finally, the watermark inversion function (**trans_w**$'$) is used to convert w_i' to w_i, and the watermark inversion function is **trans_w**$'\left(w_i'\right)$:

$$w_i = \textbf{trans\_w}'(w_i') = \begin{cases} 1 & if \quad w_i > 0 \\ 0 & if \quad w_i = 0 \end{cases} \tag{13}$$

The spatial watermark is embedded in the last layer of the model, which in turn extracts the altered watermark in the output image of the model. The spatial watermark is independent of the structure of the model and can be embedded into different models to protect the copyright of the model together with the trigger sample watermark.

4 Experiments

In this paper, we take an image segmentation network as an example of an image processing network M. The segmentation networks used UNET++ [26], UNET [27], and FCN [28], whose task is to input an image of a cell to segment the cells and background. We use 670 cell images from the Cell nuclei dataset [26] as a training set.

We analyze the performance of the framework by the following metrics: the feasibility of the trigger key generation method (whether the proposed trigger key generation method is feasible), fidelity (the effect of watermarking on the accuracy of the model), watermark integrity (whether the watermark is extracted correctly), robustness (whether the model with the watermark embedded is withstood to attacks), and the number of parameters (the change in the number of parameters before and after the watermark is embedded).

4.1 Feasibility of the Trigger Key Generation Method

Different from the traditional trigger key generation methods, the proposed method can generate trigger keys adaptively according to different models, which solves the shortcomings of traditional methods that need to design trigger keys manually according to different models. As shown in Fig. 6, column (a) shows the image without any operations added and validation keys. Column (b) is used to generate trigger keys using the added text method and the output is obtained by inputting the trigger key in the model protected with this trigger key. Column (c) uses the abstract image as a trigger key and the output obtained by inputting the trigger key in the model protected with this trigger key. Column (d) is the method of adding noise to generate the trigger key and the output obtained by inputting the trigger key in the model protected with this trigger key. Column (e) is the method to generate trigger keys for hiding an invisible logo and the output is obtained by inputting the trigger key in the model

protected with this trigger key. Column (f) is our proposed method to generate the trigger key and the output obtained by inputting the trigger key in the model protected with this trigger key.

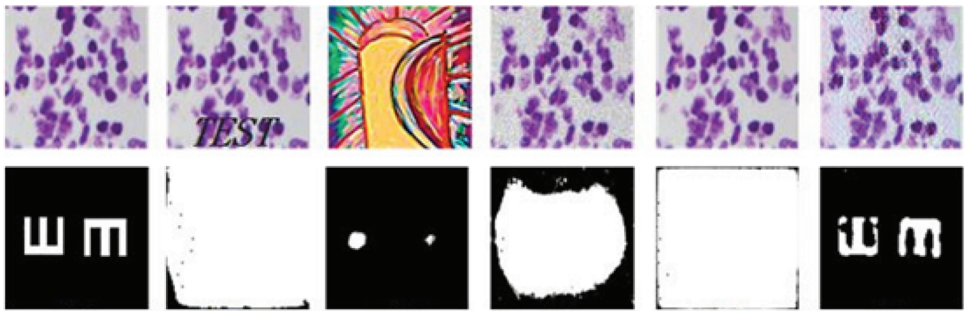

Figure 6: Trigger keys of different generation methods and their verification

The accuracy of embedding the trigger key watermark onto the UNET++ model using the five methods mentioned above. The intersection of unions (IOU) is a standard performance measure for segmentation problems, and a larger IOU value indicates higher segmentation accuracy. The IOU value between the output of the trigger key in the model and the validation key is shown in Table 1. We use the value of IOU between the output of the trigger key in the model and the validation key to measuring the performance of the trigger key in the model. Experiments show that the trigger key generation method proposed in this paper does not affect the accuracy of the model, and the IOU value of the verification key is 0.77, which can well realize the model verification. At the same time, the limitation of traditional methods in image processing networks is proved.

Table 1: Model ACC and validation IOU in trigger keys of different generation methods

	Text	Abstract	Noise	Hide logo	Ours
Model ACC	0.8	0.83	0.808	0.81	0.86
validation IOU	0.09	0.06	0.15	0.09	0.77

4.2 Model Fidelity

Model fidelity is that the model accuracy is not degraded by the embedded watermark. The model accuracy of different segmentation models without embedded watermarks and with embedded dual model watermarks is summarized in Table 2. The accuracy before embedding the watermark in the UNET model is 0.842, while the accuracy after embedding the watermark is 0.835. In the UNET++ and FCN models, the accuracy before watermark embedding is 0.845 and 0.741, and the accuracy after watermark embedding is increased to 0.86 and 0.84 due to the model fine-tuning.

The changes in model accuracy for different models with and without embedding the dual model watermarks at 200 epochs of training are shown in Fig. 7. After experimental comparison, our proposed protection framework for the image processing network has almost no effect on the model accuracy or even improves the accuracy after fine-tuning the model, which will not affect the use of the model.

Table 2: Accuracy of different models before and after embedding watermarks

	UNET++	UNET	FCN
No watermarked	0.845	0.842	0.741
Watermarked	0.86	0.835	0.84

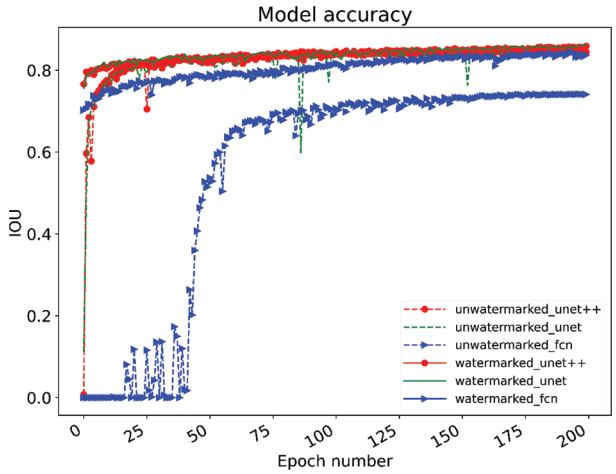

Figure 7: Changes in the accuracy of embedded and unembedded watermarking models during the training process

4.3 Watermark Integrity

Watermark integrity is whether the watermark embedded in the model can be extracted successfully. Trigger sample watermarks will not be triggered in models without watermarks. As shown in Table 3, the IOU value between the output of the trigger sample watermark in the UNET++ model and the validation key in the model without the watermark is only 0.1, however, the IOU value between the output and the validation key in the model with watermark is up to 0.77; the IOU value between the output of the trigger sample watermark in the UNET model and validation key in the model without the watermark is only 0.15, however, the IOU value between the output and the validation key in the model with watermark is up to 0.73; the IOU value between the output of the trigger sample watermark in the FCN model and validation key in the model without the watermark is only 0.149, however, the IOU value between the output and the validation key in the model with watermark is up to 0.691. Spatial watermark has a watermark extraction rate of 37.5% in the model without watermarks and 100% in the model with watermark protection, and it is possible to trace the image by extracting spatial watermark when obtaining an image. It is experimentally demonstrated that the trigger sample watermark and the spatial watermark can distinguish between the watermarked model and the unwatermarked model.

4.4 Robustness

Watermarking robustness is demonstrated by the ability to resist attacks. A forgery attack [28] is where an attacker forges a pair of inputs and outputs to obfuscate the trigger and verification keys, making the copyright of the model unverifiable. Embedding spatial watermark into the output of the

model, once the attacker wants to forge a pair of inputs and outputs to obfuscate the model copyright, we extract the watermark by output using location key K. However, the attacker's output cannot be extracted since location key K is not known. As shown in Table 4, assuming that the attacker extracts the watermark with a random position, the average match rate between what is obtained in this way and the correct watermark per bit is only 32.583%, which cannot verify the model copyright and proves that our model protection framework can effectively resist forgery attacks.

Table 3: Verification of watermark integrity under different models

	UNET++		UNET		FCN	
Watermarked or not	Not	Watermarked	Not	Watermarked	Not	Watermarked
Trigger sample watermark (IOU_value)	0.1	0.77	0.15	0.73	0.149	0.691
Spatial watermark (Extraction rate)	37.5%	100%	37.5%	100%	37.5%	100%

Table 4: Verification of resistance to forgery attacks in different models

	Fake model output	UNET++ output	UNET output	FCN output
Location key	random	K	K	K
Match rate	32.583%	100%	100%	100%
Verification	Fail	Success	Success	Success

4.5 Number of Model Parameters

In real scenarios, increasing the number of parameters makes training image processing networks much more complex and expensive than before. In the protection of the model, try not to increase the number of parameters of the model to ensure that the training of the model itself is not affected after the model is embedded in the watermark. As shown in Table 5, the parametric quantities of the model do not change before and after embedding the dual model watermarks.

Table 5: Comparison of the number of model parameters before and after embedding dual model watermark

	Unwatermarked	Watermarked
UNET++	9, 163, 329	9, 163, 329
UNET	7, 852, 545	7, 852, 545
FCN	16, 282, 881	16, 282, 881

5 Conclusion

We consider some drawbacks of using model watermarking in image processing networks. We propose a framework that includes two watermarks: trigger sample watermarking and spatial

watermarking. Due to the limitations of the trigger sample watermark generated by traditional methods for application in image processing networks, we innovatively design a general trigger sample watermark generation method, which can generate trigger sample watermarks adaptively according to different models. And the spatial watermark is embedded into the output of the model to better resist forgery attacks. Experiments show that the proposed framework has good performance in different image segmentation networks such as UNET, UNET++, and FCN, and can effectively resist forgery attacks. Two different watermarks are extracted 100% of the time in the protected model, effectively distinguishing the watermarked model from the unwatermarked model. The next work applies the framework to image classification networks to increase the generalizability of the method.

Acknowledgement: We thanks NUIST to give us the opportunity for this research work.

Funding Statement: This work is supported by the National Natural Science Foundation of China under grants U1836208, by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund, and by the Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET) fund, China.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Y. L. Cun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, San Diego, USA, pp. 1–14, 2015.
[3] L. Zhou, Z. Wang, Y. Luo and Z. Xiong, "Separability and compactness network for image recognition and superresolution," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3275–3286, 2019.
[4] Y. Xue, Y. H. Tang, X. Xu, J. Y. Liang and F. Neri, "Multi-objective feature selection with missing data in classification," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 2, pp. 355–364, 2022.
[5] S. Hong, M. Wu, Y. Zhou, Q. Wang and J. Xie, "Encase: An ensemble classifier for ECG classification using expert features and deep neural networks," in *Proc. CinC*, Rennes, France, pp. 1–4, 2017.
[6] S. Hong, Y. Zhou, M. Wu, J. Shang and J. Xie, "Combining deep neural networks and engineered features for cardiac arrhythmia detection from ECG recordings," *Physiol Meas*, vol. 40, no. 5, pp. 054009, 2019.
[7] Z. Q. Zhao, P. Zheng, S. T. Xu and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.
[8] L. Jiao, "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128837–128868, 2019.
[9] G. Anitha and S. Baghavathi Priya, "Vision based real time monitoring system for elderly fall event detection using deep learning," *Computer Systems Science and Engineering*, vol. 42, no. 1, pp. 87–103, 2022.
[10] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan *et al.,* "Learning efficient convolutional networks through network slimming," in *Proc. ICCV*, Venice, Italy, pp. 2755–2763, 2017.
[11] A. Gutub and M. Al-Ghamdi, "Image based steganography to facilitate improving counting-based secret sharing," *3D Research*, vol. 10, no. 1, pp. 1–36, 2019.
[12] A. Gutub, "Watermarking images via counting-based secret sharing for lightweight semi-complete authentication," *International Journal of Information Security and Privacy (IJISP)*, vol. 16, no. 1, pp. 1–18, 2022.
[13] X. R. Zhang, W. F. Zhang, W. Sun, X. M. Sun and S. K. Jha, "A robust 3-D medical watermarking based on wavelet transform for data protection," *Computer Systems Science & Engineering*, vol. 41, no. 3, pp. 1043–1056, 2022.

[14] Y. Adi, C. Baum, M. Cisse, B. Pinkas and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *Proc. USENIX Conf. on Security Symp.*, Baltimore, USA, pp. 1615–1631, 2018.
[15] Z. Li, C. Hu, Y. Zhang and S. Guo, "How to prove your model belongs to you: A blind-watermark based framework to protect the intellectual property of DN," in *Proc. ACSAC*, Austin, USA, pp. 126–137, 2019.
[16] J. Zhang, Z. Gu, J. Jang, H. Wu *et al.,* "Protecting intellectual property of deep neural networks with watermarking," in *Proc. ASIACCS*, Incheon, Korea, pp. 159–172, 2018.
[17] J. Zhang, D. Chen, J. Liao, W. Zhang, H. Feng *et al.,* "Deep model intellectual property protection via deep watermarking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 4005–4020, 2021.
[18] H. Wu, G. Liu, Y. Yao and X. Zhang, "Watermarking neural networks with watermarked images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 7, pp. 2591–2601, 2021.
[19] I. Goodfellow, J. P. Abadie, M. Mirza, B. Xu, D. Warde-Farley *et al.,* "Generative adversarial nets," in *Proc. NIPS*, Montreal, Canada, pp. 2672–2680, 2014.
[20] Y. Uchida, Y. Nagai, S. Sakazaw and S. Satoh, "Embedding watermarks into deep neural networks," in *Proc. ICMR*, Bucharest, Romania, pp. 269–277, 2017.
[21] B. D. Rouhani, H. Chen and F. Koushanfar, "DeepSigns: An end-to-end watermarking framework for ownership protection of deep neural network," in *Proc. ASPLOS*, Providence, USA, pp. 485–497, 2019.
[22] R. Zhu, X. Zhang and M. Shi, "Secure neural network watermarking protocol against forging attack," *EURASIP Journal on Image and Video Processing*, vol. 1, no. 2, pp. 1–12, 2020.
[23] C. Xiao, B. Li, J. Y. Zhu, W. He, M. Liu *et al.,* "Generating adversarial examples with adversarial networks," in *Proc. IJCAI*, Stockholm, Sweden, pp. 1801.02610, 2018.
[24] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. Symp. on Security and Privacy*, CA, USA, pp. 39–57, 2017.
[25] P. Isola, J. Zhu, T. Zhou and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. CVPR*, Hawaii, USA, pp. 5967–5976, 2017.
[26] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh and J. Liang, "UNet++: A nested U-Net architecture for medical image segmentation," *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, vol. 11045, pp. 3–11, 2018.
[27] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. MICCCAI*, Munich, Germany, Springer, Cham, pp. 234–241, 2015.
[28] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, USA, pp. 3431–3440, 2015.