Staged Adaptive Blind Watermarking Scheme

Baowei Wang* and Yufeng Wu

Nanjing University of Information Science and Technology, Nanjing 210044, China
wbw.first@163.com

Abstract. In traditional digital image watermarking methods, the strength factor is calculated from the content of the carrier image, which can find a balance between the robustness and imperceptibility of encoded images. However, traditional methods do not consider the feature of the message and it is also unrealistic to calculate the strength factor of each image separately when faced with a huge number of images. In recent years, digital image watermarking methods based on deep learning have also introduced the strength factor. They assign the strength factor of each image to a fixed value to better adjust the robustness and imperceptibility of the image. We hope that the network can choose the most appropriate strength factor for each image to achieve a better balance. Therefore, we propose a staged adaptive blind watermarking scheme. We designed a new component - the adaptor, and used two stages of training by training different components in different stages, and improved the robustness and imperceptibility of watermarked images. By comparing the experimental results, our algorithmic scheme shows better results compared to current advanced algorithms.

Keywords: Strength factor · Adaptor · Staged training.

1 Introduction

Traditional blind watermarking methods of digital watermarking is divided into two embedding methods: Spatial domain [1–3] and Frequency domain [4–7]. [1] was the first to embed digital image watermarking in the spatial domain. They proposed the basic concept of digital watermarking and encoded the message in the least significant bit (LSB) of the image pixel to realize the embedding of the message. This method is well hidden but the robustness is poor and the message is easy to be eliminated or detected by statistical measures. [4] chose to embed the message in the frequency domain, transform the carrier image into the frequency domain through DCT and modify the frequency domain components, so that the encoded image has higher robustness and confidentiality. Later studies found that embedding messages in DFT domain [5, 8], DCT domain [6, 9, 10], DWT domain [7, 11, 12] and SVD domain [13, 14] works better. [15, 16, 9] achieve a balance between the imperceptibility and robustness of image watermarking by controlling the strength factor of the watermark. The experimental results under different strength factors are calculated from different images. Finally, a suitable strength factor S is manually selected. These methods have poor applicability

and cannot be applied to large-scale image strength factor calculations. Besides, analyzing the carrier image will ignore the unique characteristics of the message.

In recent years, deep learning has become an indispensable key technology in the field of image research. Among them, DNN-based digital watermarking schemes have made good progress [17–19]. Deep learning-based image watermarking methods can be roughly divided into two types of embedding methods. A mainstream method is end-to-end watermarking, which integrates the entire process into an overall network, that is, the process of embedding and extracting watermarks is completely handed over to DNN. The end-to-end solution can make all components closely related and interlocked, improving the embedding and extraction efficiency of the message. [17] proposed a relatively complete end-to-end robust blind watermarking classical network architecture for the first time. Given a masked image and a binary message, the encoder produces a visually indistinguishable encoded image that contains the message, which can be recovered by a decoder with a high degree of accuracy. Later model architectures related to deep learning watermark hiding are based on this and continue to improve it. The improved framework of [18] introduced the concept of the strength factor, multiplied the watermark W by the strength factor S to obtain the watermark to be embedded, and then directly added the watermark and the carrier image I_c to obtain the final encoded image I_{en}. The process is shown in Eq. (1):

$$I_{en} = S \times W + Ic \tag{1}$$

In view of the poor practicability of the current traditional digital watermarking strength factors and the inability to carry out the mass evaluation, the digital image watermarking algorithm based on deep learning remains in the stage of manual selection of strength factors Our proposed staged adaptive blind watermarking scheme can effectively solve the problems. We designed an end-to-end phased network framework and added a new component to the classic HiDDeN network architecture, named the adaptor. The adaptor exists independently of the network's main body, which can be understood as an auxiliary network. The function of the adaptor is to extract the important features of the carrier image and the message. At the same time, the staged training can fine-tune the decoder, so that we can improve the performance of different components in different stages of training.

2 Related Work

2.1 Deep Learning-Based Digital Image Watermarking

The rise of deep learning began in 2012. In order to prove the potential of deep learning, Hinton's research group participated in the ImageNet image recognition competition for the first time. It won the championship through the built CNN network AlexNet, and crushed the second place (SVM) classification performance. It is precise because of this competition that CNN has attracted the attention of many researchers. Many scholars applied deep learning to digital

image watermarking until [20] first applied deep learning to digital image watermarking and proposed a new deep learning-based auto-encoder convolutional neural network (CNN) for embedding and extracting non-blind watermarking, and is far superior to traditional transform domain watermarking methods in imperceptibility and robustness. Subsequently, in 2018, [17] first proposed an end-to-end neural network-based blind watermark embedding scheme. The network architecture of HiDDeN is composed of four components: encoder, noise layer, discriminator, and decoder. The encoder can embed the message, the noise layer is used to improve the anti-attack ability of the watermark, and the discriminator is used to distinguish whether the input image has a watermark or not. The decoder is responsible for extracting the message in the encoded image. [18] proposed an adaptive diffusion watermarking framework composed of two residually connected fully convolutional neural networks, which took the image through DCT transformation as preprocessing and proposed a differentiable approximation of the JPEG attack. With the help of the confrontation network, the robustness of the watermark to JPEG attack is greatly improved. Not only that, but they also proposed for the first time to use a strength factor to control the strength of watermarking in the image. [21] hope to use a deep learning model to realize the application of digital image watermarking in real scenes. They simulated the distortion caused by real printing or display of images in the real world and subsequent image capture through an image perturbation module. The algorithm makes encoded images robust to image perturbations distorted by real printing and photography. [22] proposed a two-stage separable watermarking structure, which trains different components in the network in different stages, and finally achieves the optimal case of each component, which makes the watermarked image robust and imperceptible in terms of All achieved good results. [23] explored the research on the watermark based on the unknown distortion, and adopted the antagonistic training mode of the unknown distortion and the redundant channel coding mode to improve the robustness of the watermark against the unknown distortion. [19] proposed a new mini-batch method for simulated and real JPEG compression, which improved the noise layer of the attack simulation, after each batch training, from simulated JPEG, real JPEG, and no JPEG as one of the noise layers is randomly selected as the noise layer of the next mini-batch. The experimental results show that the end-to-end algorithm is very robust to JPEG and other attacks. According to the characteristics of [19] and [22]. This paper designs a staged and separable blind watermarking method based on an adaptive strength factor auxiliary network that can significantly improve the imperceptibility of encoded images.

2.2 Strength factor

The strength factor S first emerged in traditional digital watermarking. [15] made a specific description of the estimation of the watermarking strength of the watermark in the article. Calculating the optimal S based on the content of the cover image is aimed to achieve a balance between robustness and imperceptibility. In many subsequent traditional watermarking methods, the evaluation

method of the S has also been improved, but the robustness or imperceptibility will be reduced. At present, the digital watermarking methods based on deep learning have not conducted in-depth research on S [18, 19]. S is regarded as a tool for experiments and the artificially assigned fixed value is used to control the watermarking strength of the watermark to change the robustness and imperceptibility.

2.3 Staged separable training

In order to overcome the situation that traditional end-to-end training is sensitive to hyperparameters, it is necessary to jointly train multiple components and the encoder needs to take into account the balance between robustness and imperceptibility during training, which can easily cause S to be overvalued. Extreme cases of too high or too low, and the selection of strength factor S is extremely unstable. We employ a staged training approach to help the model train and tune better. In stage one, the adaptor does not participate in training and the fixed S is 1, and only the encoder, discriminator, and decoder are trained, where noise is added for attack simulation to enhance the robustness of the encoded image. Thus, what we get is the best encoder capable of redundantly embedding a message into images. In the second stage, the parameters of the encoder are frozen to keep the weights unchanged, and the adaptor is added for training. The purpose is to enable the adaptor to comprehensively evaluate the carrier image and message to obtain an optimal S. The S should be the maximum value that can improve the image quality without reducing the bit error rate. After the attack simulation is performed, the encoder is fine-tuned to obtain the best decoder under noise attack. Compared with the traditional methods that use manual determination of watermarking strength, our staged training scheme can adjust the overall network in time while S changes.

3 Proposed framework

3.1 Model Architecture

As shown in Fig. 1, the entire model architecture consists of five components: 1.Encoder, the encoder with parameters θ_{en} receives the message, carrier image, and S obtained by the adaptor, then outputs the encoded image with watermark. 2.Adaptor, the adaptor with parameters θ_{ad} outputs an S by receiving the carrier image and the reshaped message. 3.Noise layer, the noise layer receives the encoded image output by the encoder, performs attack simulation on it, and outputs the encoded image with noise. 4.Decoder, the decoder with parameters θ_{de} receives the noise image output by the noise layer and decodes it, and outputs the message contained in the watermark. 5.Discriminator, the discriminator with parameters θ_{di} receives the encoded image output by the encoder and discriminates whether the image is encoded. Next, each component will be described in detail.

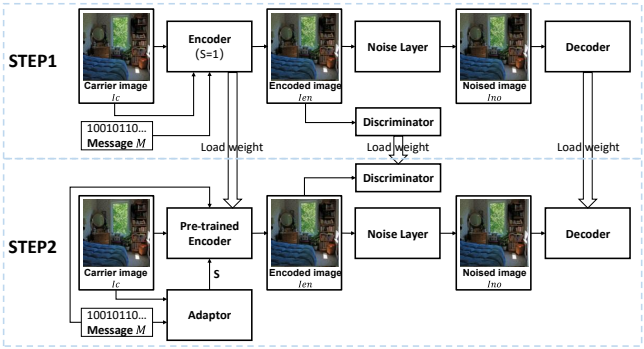

Fig. 1. The network model and training process of the network. The overall framework consists of an encoder, an adaptor, a noise layer, a decoder, and a discriminator. The training phase is divided into a no-adapter training phase and an adaptive factor overall fine-tuning phase.

Encoder The encoder is divided into two parts: message processing and carrier image processing. First, the one-dimensional message $M \in \{0,1\}^L$ needs to be preprocessed for calculation. According to the calculation formula obtained in MBRS[19], we can reshape M into $M' \in \{0,1\}^{1\times h\times w}$, after a 3×3 convolutional layer for preliminary feature extraction, n times of 2×2 upsampling operations with a stride of 2 are performed to make its size equal to the carrier image I_c, to make the message spread over the entire feature map. Finally, pass the diffused feature map through n SE blocks that do not change the shape of the feature map to obtain a more delicate message feature $I_m \in \mathbb{R}^{C\times H\times W}$. The message reshaping announcement is shown in Eq. (2):

$$L = h \times w = (H/2^n) \times (W/2^n) \quad I_m \in \mathbb{R}^{C\times H\times W} \tag{2}$$

where L is the message length (integer), h, w are the length and width of the reshaped message, and H, W are the length and width of the carrier image.

For the image processing part, I_c goes through a convolution layer and four SE blocks for feature extraction, and then the obtained image features $I_{cf} \in \mathbb{R}^{C\times H\times W}$ are concatenated with I_m to obtain the total feature I_g. Then, I_g are further extracted through a 3×3 dilated convolution to obtain features, and adding a receptive field is used to obtain more feature information. Finally, I_c combines S with I_m and encodes through a 3×3 dilated convolution. The encoded image is $I_{en} = I_c + S * I_m$. Since the encoder needs to make the encoded image I_{en} as similar to the carrier image I_c as possible, it chooses to use the mean square error between I_{en} and I_c as the loss function $\mathcal{L}_{en}$ to update the value of θ_{en}:

$$\mathcal{L}_{en} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (f(Ic) - f(Ien)) \tag{3}$$

where M, N is the image size.

Adaptor The model architecture of the adaptor is shown in the Fig. 9. The adaptor is essentially an auxiliary network, which is frozen in the first stage and does not participate in training. The function of the adaptor is to determine the optimal S that the encoder should choose according to I_c and M. Regarding the optimal S, we think that *best* refers to the state where the decoder can correctly decode the message M in the watermark and reduce the embedding strength of the watermark to the greatest extent. In layman's terms, it is to fully exploit the potential of the image to enhance the imperceptibility without increasing the Bit Error Rate(BER). Usually, researchers use Peak Signal to Noise Ratio(PSNR)([24]) and Structural Similarity (SSIM)([25]) to evaluate watermark imperceptibility in digital image watermarking, so we judge watermark imperceptibility according to PSNR and SSIM. The formulas of PSNR and SSIM are shown in Eq. (4) and Eq. (5, 6, 7, 8):

$$PSNR = 10 \times \log_{10}\left(\frac{MAX^2}{MSE}\right) \tag{4}$$

where MAX is the maximum pixel value of the image, and MSE is the mean square error.

$$SSIM = \left[a(x,y)^{\alpha} \times b(x,y)^{\beta} \times c(x,y)^{\gamma}\right] \tag{5}$$

where x, y are samples, α, β are constants, μ is the mean, σ^2 is the variance, σ_{xy} is the covariance, k is a constant, and Eq. (6,7,8) are the brightness, contrast, and structure of the image, respectively. Together they form SSIM.

$$a(x,y) = \frac{2\mu_x\mu_y + k_1}{\mu_x^2 + \mu_y^2 + k_1} \tag{6}$$

$$b(x,y) = \frac{2\sigma_x\sigma_y + k_2}{\sigma_x^2 + \sigma_y^2 + k_2} \tag{7}$$

$$c(x,y) = \frac{2\sigma_{xy} + k_3}{\sigma_x\sigma_y + k_3} \tag{8}$$

In the past, the effect of S was to achieve a balance between robustness and imperceptibility of the watermarked image. However, through our experiments, we found that each image has its unique S that can increase its imperceptibility. We redefine S as the embedding strength that maximizes the potential of image imperceptibility while maintaining robustness. The framework of the adaptor is shown in Fig. 9. The M of the extracted features is preprocessed by the same reshaping operation as in the encoder, and then through a convolutional layer, multiple upsampling and three SE blocks that do not change the feature size for feature extraction to obtain the feature $\mathrm{I_{am}} \in \mathbb{R}^{C\times H\times W}$. The carrier image passes through a 3×3 convolutional layer and a SE block to obtain the feature $I_{acf} \in \mathbb{R}^{C\times H\times W}$, and after concatenating I_{am} and I_{acf} into a convolutional layer and three SE blocks for feature extraction. Finally, the optimal S is obtained by downsampling by an average pooling layer and sent to the encoder. Since the

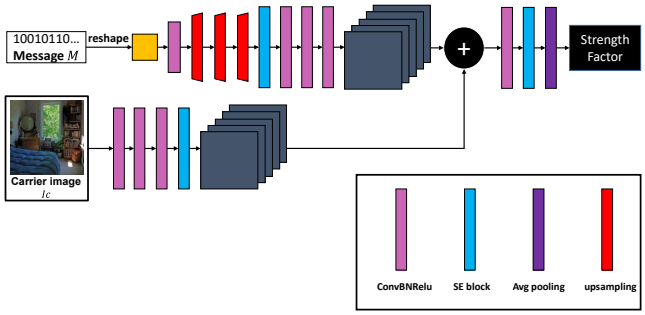

Fig. 2. Model architecture of the adaptor.

adaptor needs to be as imperceptible as possible while remaining robust, we will use a polynomial consisting of PSNR, SSIM, and BER as the loss function $\mathcal{L}_{ad}$ to update the value of θ_{ad}:

$$\mathcal{L}_{\text{ad}} = \lambda_{PSNR} \times (p_1 - PSNR) + \lambda_{SSIM} \times (1 - SSIM) + \lambda_{BER} \times BER \quad (9)$$

where $\lambda_{PSNR}, \lambda_{SSIM}, \lambda_{BER}, p_1$ is the weight factor.

Noise layer The encoded image produced by the encoder is not robust, but the noise layer can increase its resistance to attacks. I_{no} is obtained by attacking the encoded image I_{en}, so that encoder can robustly embed the message in a position that is not easily destroyed by the attack.

Decoder In the decoder, we interpret the message M' from the noisy encoded image I_{no} , and the features are extracted through a 3×3 convolutional layer and downsampling by n SE blocks. To extract messages more accurately, we still use dilated convolution in the penultimate layer to improve the receptive field. Finally, we use a 3×3 convolution layer to change the feature map into a single channel, which can be obtained after reshaping message M'. Since the decoder needs to make M and M' as similar as possible, it chooses to use the mean square error between M and M' as the loss function $\mathcal{L}_{de}$ to update the value of θ_{en}:

$$\mathcal{L}_{de} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(f(M) - f\left(M'\right) \right) \quad (10)$$

where M, N is the image size.

Discriminator Even though the discriminator is only composed of 3 layers of 3×3 convolutional layers and one pooling layer for classification, its presence can improve the imperceptibility of images in constant adversarial. In this paper, we use 1 to represent that the image contains a watermark, and 0 to represent the carrier image. The discriminator uses the loss function $\mathcal{L}_{di}$ to improve the accuracy of the binary classification results by updating θ_{di}:

$$\mathcal{L}_{di} = \mathcal{L}_{di2} + \log\left(D_i\left(\theta_{di}, I_{en}\right)\right) \quad (11)$$

At the same time, to make the encoded image similar to the carrier image, the loss function $\mathcal{L}_{di2}$ is used to improve the image quality of I_{en} by updating θ_{en}:

$$\mathcal{L}_{di2} = \log\left(1 - D_i\left(\theta_{di}, I_{en}\right)\right) \tag{12}$$

3.2 Staged training

Stage 1: The no-adapter training stage In the first stage of end-to-end encoder training, we uniformly set the adaptive factor to 1, to make the encoded image achieve higher robustness first, which is convenient for us to further fine-tune in the second stage. The encoded image is sent to the decoder for decoding after attack simulation, and the discriminator will also participate in the training. Use $\mathcal{L}_1$ as the total loss function to achieve the specified training goal:

$$\mathcal{L}_1 = \lambda_{en} \times \mathcal{L}_{en} + \lambda_{de} \times \mathcal{L}_{de} + \lambda_{di} \times \mathcal{L}_{di} \tag{13}$$

where $\lambda_{en}, \lambda_{de}, \lambda_{di}$ are weight factors.

Stage 2: The overall fine-tuning stage of the adaptive factor Through the first stage of training, we get a strong encoder responsible for watermark embedding, after which the model moves to the adaptive fine-tuning stage. First, freeze the parameters of the encoder obtained by training in stage 1. In this stage, the adaptor is added to the training, and the decoder is fine-tuned in a targeted manner to discover the most suitable S that the encoded image can accept. Loading the model weights obtained from the first stage as pre-trained weights can significantly speed up the training speed of the second stage. For stage two, we use $\mathcal{L}_2$ as the loss function to find the optimal S:

$$\mathcal{L}_2 = \lambda_{di} \times \mathcal{L}_{di1} + \lambda_{de} \times \mathcal{L}_{de} + \mathcal{L}_{ad} \tag{14}$$

where $\lambda_{di}, \lambda_{de}$ are weight factors.

3.3 Why use end-to-end training?

In our early experiments, we did try end-to-end training. However, the results were very bad. Specifically, because of the existence of strength factors, the model needs to modify too many super parameters. If we want to get a higher image quality, it will make the model set the strength factor directly to 0, that is, no watermark is embedded to ensure that the image is not damaged. On the contrary, if we pursue a lower bit error rate, the image will directly set the strength factor to a very high level, which will greatly damage the carrier image itself. To sum up, we hope to reduce the modification of super parameters, and finally adopt the phased training method of freezing parameters

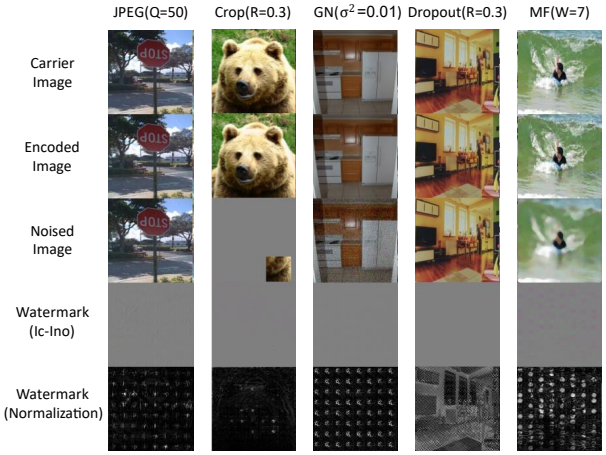

Fig. 3. Experimental results under various noise attacks. From top to bottom, the images are carrier image, encoded image, noise image, watermark, and normalized watermark. Since the pixel of the watermark itself is basically at a low value, The human eye is difficult to observe, and we normalize it to facilitate observation.

4 Experiment

Experimental details We randomly select 10,000 128×128 images from the COCO dataset as the training set, and also randomly select 5,000 128×128 images from the COCO validation set as the validation set, and 5,000 128×128 images as the test set. The model framework is implemented using PyTorch and trained and validated on NVIDIA RTX 4000. Each piece of message M is composed of random 0, 1 with a length of 64. For the real JPEG compression in the noise layer, we use the official JPEG compression API to call. After repeated testing, in one stage of training, we choose $\lambda_{en} = 1, \lambda_{de} = 10, \lambda_{di} = 0.0001$. In the second stage,$\alpha, \beta, \gamma = 1, \lambda_{PSNR} = 0.75, \lambda_{SSIM} = 6, \lambda_{BER} = 10000, p_1 = 100$, in order to get better training results, we also set the values of λ_{de}, the updated value is $\lambda_{de} = 15000$. To improve the convergence performance, the two-stage gradient descent method uses Adam optimizers instead of Stochastic Gradient Descent(SGD) optimizer, which has high computational efficiency and low memory usage, and sets its learning rate to 10^{-3}. At the same time, the size of the mini-batch training is 4, and each stage is trained for 100 epochs. For specific training, we train 18 specific decoders at different intensities using 5 traditional noises.

For combinatorial training, we train only one combinatorial decoder using a different traditional noise attack on each mini-batch.

Metrics We employ robustness and imperceptibility, which are now commonly used in digital image watermarking evaluation, to evaluate our model performance. Robustness We use the bit error rate (BER) to measure, imper-

Table 1. Comparison of experimental results of JPEG compression under different Q.

JPEG	Q=30				Q=50				Q=70			
Metric	BER	SSIM	PSNR	S	BER	SSIM	PSNR	S	BER	SSIM	PSNR	S
MBRS[19]	0.61%	0.937	35.14	1	0.031%	0.947	36.41	1	0.044%	0.947	38.14	1
Ours	**0.51%**	**0.953**	**37.63**	0.87	**0.016%**	**0.963**	**38.42**	0.76	**0.037%**	**0.970**	**40.96**	0.74

Table 2. Comparison of experimental results for different Crop ratios.

Crop	R=0.3				R=0.5				R=0.7			
Metric	BER	SSIM	PSNR	S	BER	SSIM	PSNR	S	BER	SSIM	PSNR	S
HiDDeN[17]	31.46%	0.921	34.24	1	24.5%	0.943	35.77	1	15.8%	0.950	37.56	1
MBRS[19]	25.6%	0.943	37.36	1	10.5%	0.946	37.69	1	0.90%	0.963	40.40	1
Ours	**24.7%**	**0.945**	**37.56**	1.08	**10.4%**	**0.968**	**40.86**	0.97	**0.86%**	**0.972**	**41.92**	0.91

ceptibility is the quality of the image, we use and values to measure. We chose to compare PSNR and SSIM with almost close BER to evaluate the performance of our model.

Baseline Our experiments refer to MBRS[19], so we will add it to the comparison, not only that, HiDDeN[17] will also be our comparison object. Due to the fact that [23] is biased towards agnostic distortion, and our model has a strong pertinence to the known distortion. This is unfair, so we did not include it in the experimental comparison.

4.1 Robustness and imperceptibility of encoded images to non-differentiable noise

JPEG compression is a commonly used lossy compression method for digital images. JPEG compression is implemented in five steps, color mode conversion, data sampling, DCT transformation, quantization frequency coefficients, and encoding. Since quantization makes the inverse gradient 0, JPEG is non-differentiable. The selection of the quality factor Q in the quantization operation is important. The larger the Q, the higher the compression degree, but the image quality will be reduced. In the experiment, we choose Q=10, 30, 50, 70, and 90 to test against JPEG compression attacks. Because [17], [18] and [22] did not conduct experiments on these quality factors, and the anti-JPEG compression The capability is also far inferior to MBRS[19], so we only compare with [19].

As can be seen from Table 1, the PSNR and SSIM values increase with increasing Q, while the BER decreases accordingly. As Q increases and image quality improves, S also tends to choose smaller values to reduce the strength of the embedded watermark. Our model improves PSNR and SSIM by about 2 and 0.02 on average, respectively, which fully demonstrates the potential of watermarked images to get better.

4.2 Robustness and imperceptibility of encoded images to Differentiable noise

Common differentiable noises include Crop, Dropout, Gaussian noise, and Median blur.

Cropping refers to randomly cropping an image from top/bottom and left/right and then complementing the areas where the image is lost with black pixels. We choose the cropping ratio as 0.3, 0.5, and 0.7 for testing.Cropping can do a lot of damage to the information embedded in the image. In the Table 2, Adapter chooses a value greater than 1 as S for more robust watermarking, which can improve SSIM and PSNR. This fully shows that after adjusting the watermark strength, the entire model still has room for adjustment.

Dropout removes random pixels from noisy images and replaces them with pixels from the cover image, we use dropout with ratios of 0.3, 0.5, and 0.7 for testing. As shown in the Table 3, since the damage of the replacement attack to the image is much smaller than that of the crop, the S selected by this scheme is all less than 1, and the watermark information can be hidden with a lower watermark strength to obtain higher imperceptibility. It can be seen that compared with the second best MBRS[19], the average SSIM is improved by about 0.01, and the PSNR is also improved by about 2, which greatly enhances the imperceptibility of the image.

Gaussian noise refers to a class of noise whose probability density function obeys a Gaussian distribution. The main source of Gaussian noise in digital images occurs during acquisition. Due to poor lighting or sensor noise caused by high temperature, we adopted values of 0.001, 0.005, and 0.01 four different variances to evaluate the anti-Gaussian noise performance of the model. Combining the Table 4 and Fif. 3 analysis, it is concluded that since the Gaussian noise damages the image more and presents a Gaussian distribution, the encoder is more inclined to regularly embed information, the embedded information presents a regular point-like distribution, and the changed position is more obvious. At the same time, due to the large damage to the image by Gaussian noise, the adaptor also selects a value greater than 1 for adjustment when selecting S.

The space for the model to be adjusted also becomes limited and the imperceptibility is only slightly improved. Median blurring means that for each pixel, in the window centered on it, the median pixel value of the neighboring pixel is taken to replace the pixel value of the position. For median blur, due to the smoothing attack method of median filtering sliding window, the encoder chooses the embedding method of block embedding and the size of the block is related to the size of the filter. The adaptor of our scheme chooses the watermarking strength with an average value of about 0.9, which makes the model performance more excellent. Compared with MBRS[19], SSIM and PSNR value are improved by 0.1 and 2 on average. Greatly improved imperceptibility in the event of a drop in BER. This method is median smoothing, also known as median filtering. We selected different window sizes, the filtering windows of 3×3, 5×5, 7×7. The specific results of the experiment are shown in Fig. 3 and Table 5.

Table 3. Comparison of experimental results of Dropout at different ratios.

Dropout	R=0.3				R=0.5				R=0.7			
Metric	BER	SSIM	PSNR	S	BER	SSIM	PSNR	S	BER	SSIM	PSNR	S
HiDDeN[17]	40.60%	0.937	34.82	1	26.1%	0.923	34.69	1	23.5%	0.902	32.58	1
MBRS[19]	0.0052%	0.961	41.39	1	0.0021%	0.977	44.45	1	0.00051%	0.983	46.64	1
Ours	0.0052%	**0.971**	**43.20**	0.87	**0.0018%**	**0.989**	**48.47**	0.79	**0.00050%**	**0.993**	**48.89**	0.76

Table 4. Comparison of experimental results of Gaussian noise under different σ^2.

Gaussian noise	σ^2=0.001				σ^2=0.005				σ^2=0.010			
Metric	BER	SSIM	PSNR	S	BER	SSIM	PSNR	S	BER	SSIM	PSNR	S
HiDDeN[17]	23.2%	0.910	34.60	1	28.1%	0.930	33.47	1	30.3%	0.908	32.86	1
MBRS[19]	0.031%	0.957	41.08	1	0.052%	0.923	38.03	1	0.10%	0.911	37.25	1
Ours	**0.016%**	**0.970**	**43.56**	0.77	**0.052%**	**0.937**	**39.30**	0.84	**0.084%**	**0.924**	**38.63**	0.91

4.3 Combined noise

In addition to specific encoders, we also experiment with a combination of various noises We add several differentiable and non-differentiable attacks used above to the noise layer, and perform random combined attacks on each image. Combined noises include: Crop (R=0.3), Dropout (R=0.3), Gaussian noise (σ^2=0.01), Median blur (W=30), JPEG compression (Q=50).The experimental results are shown in Table 6. The adaptor selects an average value of 0.9 as the best S for watermark embedding in the face of a combination of various noises. Although the effect is not as good as the 18 specific decoders described above, it also shows that compared with MBRS[19] for better results.

4.4 Ablation experiment

To further validate our idea, we will conduct ablation experiments. One option is training without stages. In the first 100 stages of training, we still choose 1 as the value of S. In the second stage, we leave the model unchanged to continue training for 100 stages and add the staged training of the adaptor, respectively. Another option is to train with a fixed strength factor in stages. We take the average value of each image strength factor output by the adaptor as a fixed strength factor and put it into the model without an adaptor for 100 stages of training.

The experimental results are shown in Table 7. We found that after continuing to use 1 for 100 training sessions, the performance of the model was not improved because the model had converged. When we use a fixed S for each image to perform fixed S training in stages, although the model has been improved to a certain extent, different images should choose different optimal S, and the fixed S will The watermarking strength of the actual watermark is slightly deviated from the optimal strength, making its final effect inferior to the experimental results of our staged and adaptive training scheme.

Table 5. Comparison of experimental results of median smoothing under different window sizes.

Median blur	W=3				W=5				W=7			
Metric	BER	SSIM	PSNR	S	BER	SSIM	PSNR	S	BER	SSIM	PSNR	S
HiDDeN[17]	23.00%	0.892	32.74	1	27.6%	0.900	33.66	1	28.57%	0.888	31.33	1
MBRS[19]	0.0015%	0.963	39.55	1	0.17%	0.952	37.76	1	0.20%	0.941	35.60	1
Ours	**0.0010%**	**0.977**	**42.03**	0.88	**0.14%**	**0.965**	**39.16**	0.90	**0.14%**	**0.952**	**37.76**	0.91

Table 6. Comparison of experimental results against each noise under combined noise. Among them, MBRS[19]: SSIM=0.905,PSNR=34.94,S=1.00, **Ours: SSIM=0.925, PSNR=38.55,** S=0.92

Noise	Model	BER
JPEG	MBRS[19]	0.42%
(Q=50)	**Ours**	**0.26%**
Crop	MBRS[19]	27.8%
(R=0.3)	**Ours**	**26.4%**
Dropout	MBRS[19]	0.60%
(R=0.3)	**Ours**	**0.090%**
Gaussian noise	MBRS[19]	0.87%
(σ^2=0.01)	**Ours**	**0.72%**
Median blur	MBRS[19]	0.096%
(W=3)	**Ours**	**0.068%**

4.5 Disadvantages of the model

Our model adopts a phased training mode, which will increase the training time. We compare the time complexity of the algorithm by training the average time spent(s) for an epoch of 10000 images on NVIDIA RTX 4000. Because the time spent in different noises is different, we only show the average training time in JPEG compression(Q=50). It can be seen from the Table 8 that our model takes longer than other models. And because the encoder parameters are frozen in the second stage of training, the training time in the second stage is lower than that in the first stage.

5 Conclusion

In this paper, we propose an adaptive watermarking strength factor embedding scheme with a two-stage training scheme. The training scheme consists of two stages: the no-adapter training stage and the adaptive factor overall fine-tuning stage. The first stage is responsible for training the overall network framework, adding the noise layer to the training to obtain images with higher robustness. In the second stage, the scheme introduces an adaptor into the component to adjust the model and improve the imperceptibility by selecting different S for each image. A large number of experiments show that the proposed watermark embedding scheme has excellent robustness and imperceptibility and stands out

Table 7. Comparison of experimental results of different training methods.

Noise	Metric	No staged training (S=1)	Staged Fixed Strength Factor Training	Staged and adaptive training
JPEG(Q=50)	BER	0.029%	0.019%	**0.016%**
	SSIM	0.944	0.954	**0.963**
	PSNR	36.51	37.52	**38.42**
Crop(R=0.7)	BER	0.90%	0.89%	**0.86%**
	SSIM	0.962	0.971	**0.972**
	PSNR	40.24	41.54	**41.92**
Dropout(R=0.7)	BER	0.00052%	0.00052%	**0.00050%**
	SSIM	0.982	0.992	**0.993**
	PSNR	46.60	48.19	**48.89**
Gaussian noise(σ^2=0.5)	BER	0.0048%	0.0048%	**0.0047%**
	SSIM	0.879	0.880	**0.880**
	PSNR	25.17	25.71	**25.74**
Median blur(W=5)	BER	0.20%	0.16%	**0.14%**
	SSIM	0.938	0.960	**0.965**
	PSNR	35.28	38.30	**39.16**

Table 8. Time complexity comparison of different models

Model	HiDDeN[17]	MBRS[19]	Ours(step1)	Ours(step2)	Ours(step1+step2)
Time spent	583	870	708	474	1182

among various current excellent watermark embedding algorithms. However, due to the staged training method, the cost of training has also increased.

References

1. Van Schyndel, R.G., Tirkel, A.Z., Osborne, C.F.: A digital watermark. In: Proceedings of 1st international conference on image processing. Volume 2., IEEE (1994) 86–90
2. Zong, T., Xiang, Y., Natgunanathan, I., Guo, S., Zhou, W., Beliakov, G.: Robust histogram shape-based method for image watermarking. IEEE Transactions on Circuits and Systems for Video Technology **25** (2014) 717–729
3. Hua, G., Xiang, Y., Zhang, L.Y.: Informed histogram-based watermarking. IEEE Signal Processing Letters **27** (2020) 236–240
4. Cox, I.J., Kilian, J., Leighton, T., Shamoon, T.: Secure spread spectrum watermarking for images, audio and video. In: Proceedings of 3rd IEEE international conference on image processing. Volume 3., IEEE (1996) 243–246
5. Ruanaidh, J., Dowling, W., Boland, F.M.: Phase watermarking of digital images. In: Proceedings of 3rd IEEE International Conference on Image Processing. Volume 3., IEEE (1996) 239–242
6. Hamidi, M., Haziti, M.E., Cherifi, H., Hassouni, M.E.: Hybrid blind robust image watermarking technique based on dft-dct and arnold transform. Multimedia Tools and Applications **77** (2018) 27181–27214

7. Guo, H., Georganas, N.D.: Digital image watermarking for joint ownership verification without a trusted dealer. In: 2003 International Conference on Multimedia and Expo. ICME'03. Proceedings (Cat. No. 03TH8698). Volume 2., IEEE (2003) II–497
8. Rawat, S., Raman, B.: A blind watermarking algorithm based on fractional fourier transform and visual cryptography. Signal Processing **92** (2012) 1480–1491
9. Mishra, A., Agarwal, C., Sharma, A., Bedi, P.: Optimized gray-scale image watermarking using dwt–svd and firefly algorithm. Expert Systems with Applications **41** (2014) 7858–7867
10. Meng, F., Peng, H., Pei, Z., Wang, J.: A novel blind image watermarking scheme based on support vector machine in dct domain. In: 2008 International Conference on Computational Intelligence and Security. Volume 2., IEEE (2008) 16–20
11. Lin, W.H., Wang, Y.R., Horng, S.J., Kao, T.W., Pan, Y.: A blind watermarking method using maximum wavelet coefficient quantization. Expert Systems with Applications **36** (2009) 11509–11516
12. Piao, C.R., Beack, S., Woo, D.M., Han, S.S.: A blind watermarking algorithm based on hvs and rbf neural network for digital image. In: International Conference on Natural Computation, Springer (2006) 493–496
13. Ali, M., Ahn, C.W., Pant, M.: A robust image watermarking technique using svd and differential evolution in dct domain. Optik **125** (2014) 428–434
14. Lai, C.C.: An improved svd-based watermarking scheme using human visual characteristics. Optics Communications **284** (2011) 938–944
15. Kang, X.b., Zhao, F., Lin, G.f., Chen, Y.j.: A novel hybrid of dct and svd in dwt domain for robust and invisible blind image watermarking with optimal embedding strength. Multimedia Tools and Applications **77** (2018) 13197–13224
16. Ariatmanto, D., Ernawan, F.: Adaptive scaling factors based on the impact of selected dct coefficients for image watermarking. Journal of King Saud University-Computer and Information Sciences (2020)
17. Zhu, J., Kaplan, R., Johnson, J., Fei-Fei, L.: Hidden: Hiding data with deep networks. In: Proceedings of the European conference on computer vision (ECCV). (2018) 657–672
18. Ahmadi, M., Norouzi, A., Karimi, N., Samavi, S., Emami, A.: Redmark: Framework for residual diffusion watermarking based on deep networks. Expert Systems with Applications **146** (2020) 113157
19. Jia, Z., Fang, H., Zhang, W.: Mbrs: Enhancing robustness of dnn-based watermarking by mini-batch of real and simulated jpeg compression. In: Proceedings of the 29th ACM International Conference on Multimedia. (2021) 41–49
20. Kandi, H., Mishra, D., Gorthi, S.R.S.: Exploring the learning capabilities of convolutional neural networks for robust image watermarking. Computers & Security **65** (2017) 247–268
21. Tancik, M., Mildenhall, B., Ng, R.: Stegastamp: Invisible hyperlinks in physical photographs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 2117–2126
22. Liu, Y., Guo, M., Zhang, J., Zhu, Y., Xie, X.: A novel two-stage separable deep learning framework for practical blind watermarking. In: Proceedings of the 27th ACM International Conference on Multimedia. (2019) 1509–1517
23. Luo, X., Zhan, R., Chang, H., Yang, F., Milanfar, P.: Distortion agnostic deep watermarking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 13548–13557

24. Almohammad, A., Ghinea, G.: Stego image quality and the reliability of psnr. In: 2010 2nd International Conference on Image Processing Theory, Tools and Applications, IEEE (2010) 215–220
25. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13** (2004) 600–612