Hindawi
Security and Communication Networks
Volume 2023, Article ID 3813977, 12 pages
https://doi.org/10.1155/2023/3813977

Research Article

Toward High Capacity and Robust JPEG Steganography Based on Adversarial Training

Jianhua Yang,[1,2] **Fei Shang**,[2] **Yi Liao**,[3] **and Yifang Chen**[1,2]

[1]*Guangdong Polytechnic Normal University, Guangzhou 510635, Guangdong, China*
[2]*Guangdong Key Laboratory of Information Security, Sun Yat-sen University, Guangzhou 510006, Guangdong, China*
[3]*South China University of Technology, Guangzhou 510006, Guangdong, China*

Correspondence should be addressed to Fei Shang; shangf5@mail2.sysu.edu.cn

Received 4 August 2022; Revised 7 October 2022; Accepted 24 November 2022; Published 21 February 2023

Academic Editor: Hanzhou Wu

JPEG steganography has become a research hotspot in the field of information hiding. However, the capacity of conventional JPEG steganography methods is hard to meet the requirements in high-capacity application scenarios and also can not extract secret messages accurately after JPEG compression. To mitigate these problems, we propose a high-capacity and robust JPEG steganography based on adversarial training called HRJS, which implements an end-to-end framework in the JPEG domain for the first time. The encoder is responsible for embedding the secret message while the decoder can reconstruct the original secret message. To enhance robustness, an attack module forces the neural network to automatically learn how to correctly recover the secret message after an attack. Experimental results show that our method achieves near 100% decoding accuracy against JPEG_50 compression at 1/3 bits per channel (bpc) payload while preserving the imperceptibility of the stego image. Compared with conventional JPEG steganography methods, the proposed method is feasible with high capacity (e.g., 1 bpc) and has an obvious advantage in terms of robustness against JPEG compression at the same time.

1. Introduction

Modern steganography is a technology used to realize secret communication, which embeds a secret message into the cover image and ensures the imperceptibility. Nowadays, image steganography [1] has played a significant role in the fields of covert communication, medical systems [2], information certification [3], digital communication [4], and so on.

Image steganography makes use of visual redundancy to embed secret information so that the naked eye and Steg-Analyzer can not detect the suspicious visual changes of the image. In order to achieve this goal, the traditional adaptive steganography methods hide the secret message in complex texture regions, while avoiding the smooth regions. Based on the above-given perception, researchers proposed many spatial adaptive steganography algorithms, such as S-UNIWARD [5], HILL [6], and MiPOD [7]. For the JPEG domain, J-UNIWARD [5] and UERD [8] are proposed. However, the design of distortion functions is heuristic, which excessively lies on the experience of designers. At the same time, the hand-crafted distortion functions based on heuristic principles do not fully consider the statistical undetectability, which leads to traditional adaptive steganography methods can not effectively resist the detection of many advanced steganalysis methods [9, 10].

With the development of deep learning, some steganography methods combined with CNNs (convolutional neural networks) have been proposed, which can effectively alleviate the disadvantages of heuristic design as well as improve the performance to resist advanced steganalysis [11]. The deep steganography methods can be mainly divided into two different categories: automatic embedding cost learning-based image steganography methods and end-to-end image steganography methods. In the first category, neural networks are used a similar traditional algorithm to identify the locations suitable for embedding data, such as ASDL-GAN [12], UT-GAN [13], and JS-GAN [14]. These methods can automatically find the embedding locations and generate the embedding cost. However, their embedding and extracting message processes are completely dependent on syndrome trellis code (STC) [15]. It should be noted that the cost design-based steganography can not against attack; e.g., the message can not be extracted when the stego is under JPEG compression attack, and the robustness should be improved for practical application.

In order to improve the robustness of steganography, researchers have proposed end-to-end steganography methods, such as deep steganography [16], HiDDeN [17], SteganoGAN [18], and IS-GAN [19]. As shown in Figure 1, the end-to-end steganography framework takes the cover image and secret message as input and finally outputs the decoded secret message. In particular, embedding and extracting secret messages are accomplished by the hiding network and revealing network, respectively. By inserting an attack module to force networks to learn how to recover messages after being attacked, the robustness can be improved. Nevertheless, all the above-given end-to-end steganography methods are concentrated in the spatial domain. Taking account of the image on the Internet will inevitably be compressed in the transmission process; so JPEG image steganography [20] has higher practical value. However, there are the following challenges when applied to the JPEG domain. Firstly, due to the complex statistical characteristics of discrete cosine transform (DCT) coefficients, the difficulty and complexity of steganography in the JPEG domain is usually higher than the spatial domain. Secondly, the lossy quantization in JPEG compression leads to more difficult to recover the secret message. Therefore, all of the previous works have not implemented the end-to-end steganography methods in the JPEG domain. So most existing JPEG image steganography methods are not robust to JPEG compression and have a low embedding capacity. Moreover, the existing image steganography methods are difficult to effectively allocate the secret message between RGB channels. In order to address these limitations, we propose a novel end-to-end JPEG steganography framework based on adversarial training [21]. The advantages of the proposed method are as follows:

(1) We propose a high-capacity and robust JPEG steganography framework called HRJS, which embeds and extracts secret message by a neural network. It is worth mentioning that our method implements the end-to-end steganography methods in the JPEG domain for the first time.

(2) The robustness is greatly improved. We insert an attack module between the encoder and decoder to simulate practical application scenarios. It can extract meaningful information from the network disturbance through adversarial training, which forces the neural network to automatically learn how to correctly recover the secret message from the attacked stego image.

(3) The capacity is greatly improved. Our method utilizes a neural network to adaptively embed a secret message into RGB channels. It realizes the embedding of 1 bpc secret message while preserving the imperceptibility of the stego image and robustness. By comparison, the conventional works only effective up to a payload of around 0.5 bpnzAC (bit per nonzero AC DCT coefficient) or even much lower.

The rest of this paper is organized as follows. We review the traditional JPEG image steganography and introduce end-to-end image steganography in Section 2. Then, in Section 3, we introduce the architecture and loss function of HRJS in detail. Next, we show the experimental results and comprehensive analysis in Section 4. Finally, in Section 5, we present conclusions and avenues for future work.

2. Related Works

2.1. Traditional JPEG Image Steganography. The embedding modification object of JPEG image steganography [22, 23] is the quantized DCT coefficients. Many adaptive JPEG image steganography methods are committed to designing a more appropriate distortion cost function for the embedding modification of DCT coefficients. UED [24] and UERD [8] proposed to uniformly extend the embedding modifications to the quantized DCT coefficients which have similar magnitudes. J-UNIWARD [5] took into account the statistical characteristics of the spatial domain when designed the cost function and obtained high security. BET [25] transformed the distortion cost function for spatial images into JPEG images and utilized the statistics of the DCT domain and spatial domain. It showed excellent performance in resisting advanced JPEG steganalysis. GUED [26] proposed new distortion measures that can keep the statistical characteristics of the cover unchanged on DCT block and AC (alternating current) mode. Moreover, a general and effective empirical rule is proposed to select the parameters of the exponential function. Work [27] creatively put forward a block boundary maintenance (BBM) principle, and the nonadditive cost function of JPEG steganography is defined by the coefficient correlation of intrablocks within the DCT domain.

2.2. End-to-End Image Steganography. The previous cost learning-based steganography methods using deep learning still rely on other traditional algorithms like STC to embed or extract the secret message. Different from these methods, the embedding and extracting processes of end-to-end steganography methods are both accomplished by networks.

The end-to-end steganography methods usually contain encoder and decoder networks. The encoder produces the visually indistinguishable stego image by inputting the secret message and cover image, from which the decoder could reconstruct the original secret message. Work [16] proposed an end-to-end steganography structure and embedded a full-size color image within another image of the same size, which significantly increased the payload. Due to the large payload, it could be easily detected by steganalysis tools. Work [17] can extract useful information under the attack of

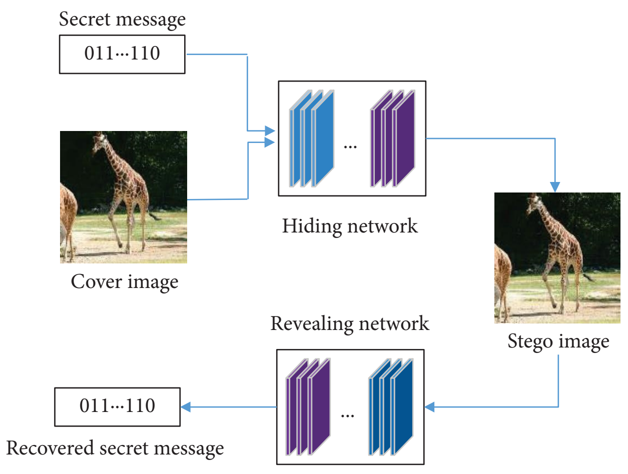

Figure 1: Existing end-to-end architecture in spatial domain.

adversarial perturbations and greatly improve the robustness by inserting an optional noise layer. It is worth mentioning that it has achieved good performance in steganography and watermarking. Although this work can effectively resist many kinds of attacks, it applied upsampling to cover the image before it was fed into the encoder and reduced the quality of the image significantly. SteganoGAN [18] explored the encoder architecture of three connectivity patterns, which contributed to optimize the perceptual quality of stego image.

Different from the above-given methods which took cover and secret message/image as the input of the encoder, UDH [28] disentangled the encoding of a secret image from the cover, which can analyze the embedding mechanism of the secret image conveniently. Benefiting from this universal deep hiding (UDH) framework, it found that frequency discrepancy between encoded secret image and cover is the key to the success of deep steganography. Work [29] utilized the forward and backward propagation of an invertible steganography network to handle the embedding and extracting message processes. In addition, with the increase in the number of hidden image channels, the steganography capacity increases. Although end-to-end steganography has been developed in the spatial domain, it is still in the initial stage in the JPEG domain.

3. Proposed Methods

In this section, we will define the basic notations and introduce the overall architecture of our proposed HRJS. Moreover, the details of encoder, inverse discrete cosine transform (IDCT), attack module, decoder, and discriminator will be given. After that, we will describe our loss function in detail. Finally, we will introduce the training steps of HRJS.

3.1. Notation. The capital letter $\mathbf{C} = (c_{i,j})^{3\times W\times H}$ and $\mathbf{S} = (s_{i,j})^{3\times W\times H}$, respectively, represent the cover and stego images, where W and H mean the width and height of the images, respectively. Specifically, both $\mathbf{C}$ and $\mathbf{S}$ are RGB color images. We use $\mathbf{M} \in \{0,1\}^{D\times W\times H}$ and $\mathbf{M}' \in \{0,1\}^{D\times W\times H}$ to represent the binary secret message and decoded secret message, respectively, where D means the depth of the secret message.

3.2. Architecture. The architecture of our proposed HRJS is shown in Figure 2, which consists of the following five modules. Encoder is responsible for embedding the secret message, which takes the DCT coefficients of the cover image and secret message as input and generates the DCT coefficients of the stego image. IDCT module is similar to JPEG decompression operation, which is the reverse process of JPEG compression. It converts the JPEG domain image into the corresponding spatial domain, while the DCT module is just the opposite. The attack module simulates JPEG compression. It receives a stego image from the IDCT module and produces the attacked stego image. In particular, it can efficiently force the network to learn how to recover secret messages correctly from the attacked stego image. Decoder receives the DCT coefficients of attacked stego image and attempts to recover the original secret message. Discriminator is responsible for distinguishing the stego image from the cover image. The principle and implementation of these five modules in detail will be introduced in the next section.

3.2.1. Encoder. The encoder receives the DCT coefficients of the cover image as well as a binary secret message and then produces the DCT coefficients of the stego image, which is also an RGB color image and has the same shape with the cover image. It should be noted that the secret message is binary data and has the same width and height with a cover image. It is worth mentioning that hiding binary messages with neural networks in the vast majority of JPEG image steganography methods has a low embedding capacity, which does not satisfy the needs of large-scale data hiding. To mitigate this problem, we set an adjustable parameter D, which denotes the depth of the secret message. By adjusting the value of D, the payload of our HRJS can reach up to 1 bpc, which is much larger than the cost design-based steganography methods.

The network structure of the encoder is shown in Figure 3. We choose a residual variant, which is inspired by the ResNet [30]. Firstly, we apply a convolutional block to the cover image to produce a high-dimensional representation of shape $(32, W, H)$ and then concatenate it with secret message. The combined tensor is through three convolutional blocks and then add input cover image tensor to form an output. Specifically, each convolutional layer has a kernel size of 3×3, with stride and padding 1. The first three convolutional blocks utilize the LeakyReLU activation function while the last one utilizes the Tanh activation function.

3.2.2. IDCT Module. This module is used to obtain the spatial cover and stego images. It firstly segments the DCT coefficients of the image and then performs dequantization

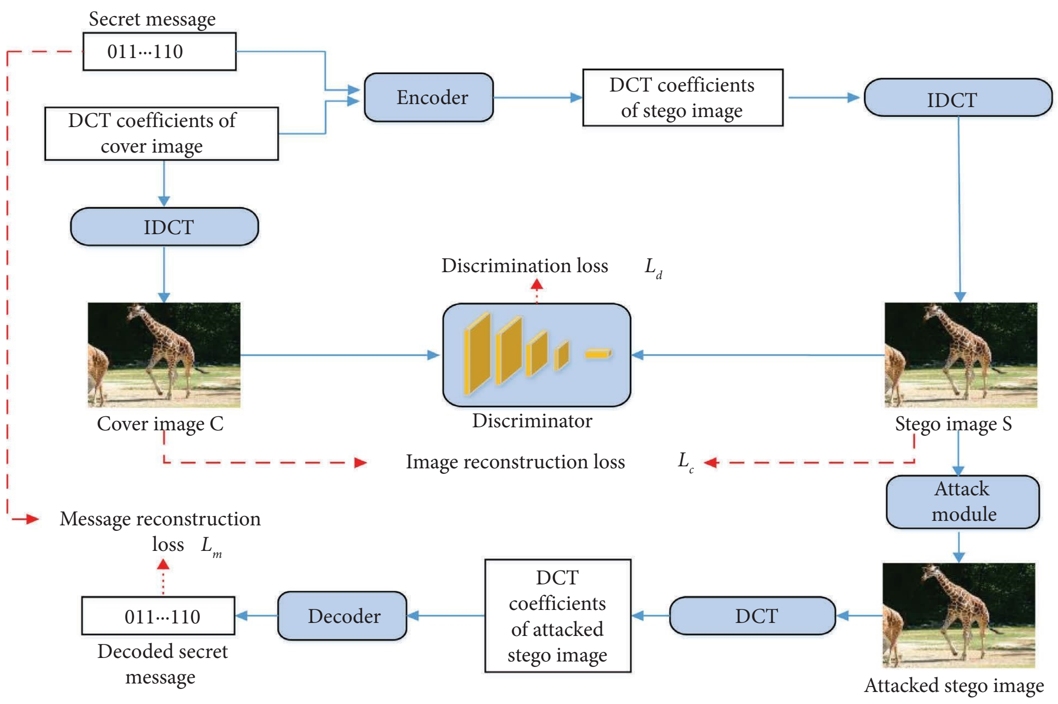
Secret message
011···110
DCT coefficients of cover image
Encoder
DCT coefficients of stego image
IDCT
IDCT
Discrimination loss L_d
Cover image C
Discriminator
Stego image S
Image reconstruction loss L_c
Attack module
Message reconstruction loss L_m
011···110
Decoded secret message
Decoder
DCT coefficients of attacked stego image
DCT
Attacked stego image

Figure 2: The architecture of the HRJS. The encoder is used to generate the DCT coefficients of the stego image, while the decoder is used to reconstruct the original secret message. The attack module simulates JPEG compression. The IDCT module and DCT module convert the image into spatial domain and JPEG domain, respectively. The discriminator is used to identify whether a given image conceals the secret message.

and inverse discrete cosine transformation on the coefficients in each block and finally concatenates the subblocks to transform them into the spatial image. Due to the color image dataset used in our method, we need to add the color space transformation in the above-given process. The core of this module is the following IDCT function:

$$f(a,b) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \alpha(u)\alpha(v)F(u,v), \left[\cos\frac{\pi u(2a+1)}{2N}\right]\left[\cos\frac{\pi v(2b+1)}{2N}\right], \tag{1}$$

where $a, b = 0, 1, 2, \ldots, N-1$. $f(a,b)$ and $F(u,v)$ represent the spatial pixel value and DCT coefficient, respectively. The coefficients $\alpha(u)$ and $\alpha(v)$ can be expressed as

$$\alpha(u) = \begin{cases} \sqrt{\dfrac{1}{N}}, & \text{if}\, u = 0, \\ \sqrt{\dfrac{2}{N}}, & \text{if}\, u \neq 0, \end{cases}$$
$$\alpha(v) = \begin{cases} \sqrt{\dfrac{1}{N}}, & \text{if}\, v = 0, \\ \sqrt{\dfrac{2}{N}}, & \text{if}\, v \neq 0. \end{cases} \tag{2}$$

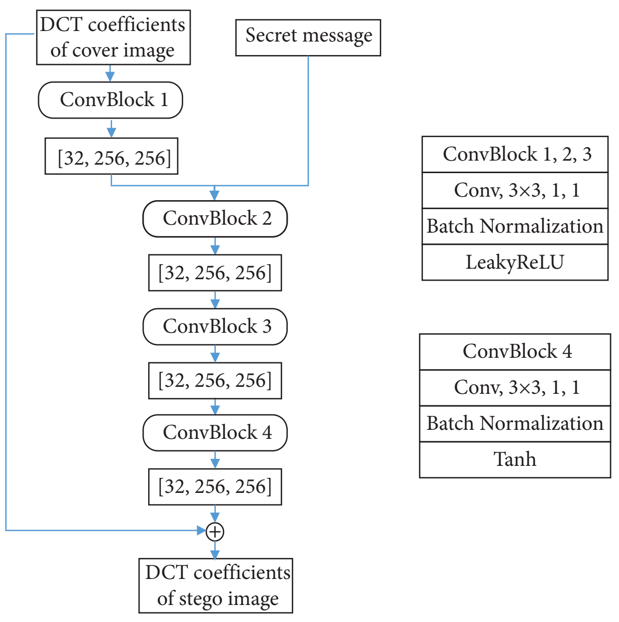

Figure 3: The architecture of the encoder. "Conv, 3 × 3, 1, 1" means the convolutional layer using a kernel size of 3 × 3, with stride and padding 1.

This IDCT module can keep the back-propagation while performing the IDCT process and ensure the gradient does not disappear. In addition, we utilize matrix multiplication to realize the IDCT module, which ensures the high efficiency.

3.2.3. Attack Module. Digital images are always accompanied by various attacks in the process of real Internet transmission, such as JPEG compression in the social Internet. However, most existing image steganography networks are vulnerable to these attacks, which can not satisfy the steganographic requirement in the practical application scenario. Moreover, if the stego image obtained by traditional steganography suffers these attacks in the process of transmission, it is difficult to recover the secret message correctly. To mitigate this problem, we insert an attack module after the IDCT module during the training stage to simulate JPEG compression.

Generally, JPEG compression consists of two steps, which are DCT transformation and quantification. The quantification process is followed by rounding. Because the rounding function is a piecewise step function and cannot be differentiated, the transfer of gradient will be truncated after rounding. Therefore, we utilize the following rounding operation to simulate rounding [31]:

$$r(x_q) = [x_q] + (x_q - [x_q])^3, \tag{3}$$

where $[x_q]$ stands for rounding x_q. Formula (3) has a nonzero derivative almost everywhere, which can simulate the round operation and at the same time keep gradient propagation.

Thanks to the careful design of the JPEG compression attack module, our method can effectively resist relative attacks by simulating the attack in the training step and greatly improve the robustness of the JPEG steganography method. It is worth mentioning that the attack module can also simulate other malicious attacks, such as Gaussian noise and dropout and can also achieve excellent performance.

3.2.4. Decoder. The decoder takes the DCT coefficients of the attacked stego image as input and produces the decoded secret message, which has the same shape as the original secret message. The network structure of the decoder is shown in Figure 4. We apply six convolutional blocks to obtain the decoded secret message. Each convolutional layer has a kernel size of 3×3, with stride and padding 1. In particular, we add the DCT coefficients of the attacked stego image to the output of the fifth convolution block and then carry out the last convolution. This operation contains more steganographic weak signals, which is conducive to the recovery of secret messages.

3.2.5. Discriminator. To provide feedback on the performance of the network and produce more realistic images, we introduce an adversarial discriminator as shown in Figure 5. It takes a spatial cover image and a spatial stego image as input. We use four convolutional blocks, each convolutional layer has a kernel size of 3×3, with stride 2 and padding 1. The first three blocks take LeakyReLU as the activation function and the last block uses ReLU. At the end of the last convolutional block, adaptive average pooling is performed. After that, the discriminator performs squeeze and full connection operations to output the result of binary classification.

3.3. Loss Function. The objective loss function of HRJS contains reconstruction loss L_{rec} and adversarial loss L_{adv}. The training objective is to minimize:

$$L_{\text{total}} = L_{\text{rec}} + \lambda_a \times L_{\text{adv}}, \tag{4}$$

where λ_a is used to adjust the weight of the above two losses.

The reconstruction loss L_{rec} encourages the stego image and the decoded secret message closer to the cover image and the original secret message, respectively. Therefore,

$$L_{\text{rec}} = \lambda_c \times L_c + \lambda_m \times L_m, \tag{5}$$

where λ_c and λ_m are hyper-parameters which balance the fidelity of the stego image and the degree of secret message recovery. L_c and L_m represent the image reconstruction loss and message reconstruction loss, respectively. In the next section, we will present the effect of the different weights of L_m in detail.

In order to ensure the visual quality of the stego image, we use mean square error (MSE) and structural similarity (SSIM) to measure the similarity between the cover image $\mathbf{C}$ and stego image $\mathbf{S}$. So we define the following image reconstruction loss L_c:

$$L_c = \text{MSE}(\mathbf{C}, \mathbf{S}) + \beta \times (1 - \text{SSIM}(\mathbf{C}, \mathbf{S})), \tag{6}$$

where parameter β is used to adjust the importance of MSE and SSIM, and we will explain the determination of β in detail in the next section. $\text{MSE}(\mathbf{C}, \mathbf{S})$ denotes the mean square error between cover image $\mathbf{C}$ and the corresponding stego image $\mathbf{S}$. The smaller the value of MSE, the better the image quality. Similarly, $\text{SSIM}(\mathbf{C}, \mathbf{S})$ denotes the structural similarity between $\mathbf{C}$ and $\mathbf{S}$. The range of SSIM is [0, 1]. Closer to 1 means that the stego image is more similar to the cover image. Given the current image $\mathbf{X} = (x_{i,j})^{W\times H}$ and the reference image $\mathbf{Y} = (y_{i,j})^{W\times H}$, where W and H represent the width and height of the images, respectively. MSE and SSIM are calculated by

$$\text{MSE}(x, y) = \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}\left(x_{i,j} - y_{i,j}\right)^2, \text{SSIM}(x, y) = [L(x, y)]^l \times [C(x, y)]^m \times [S(x, y)]^n, \tag{7}$$

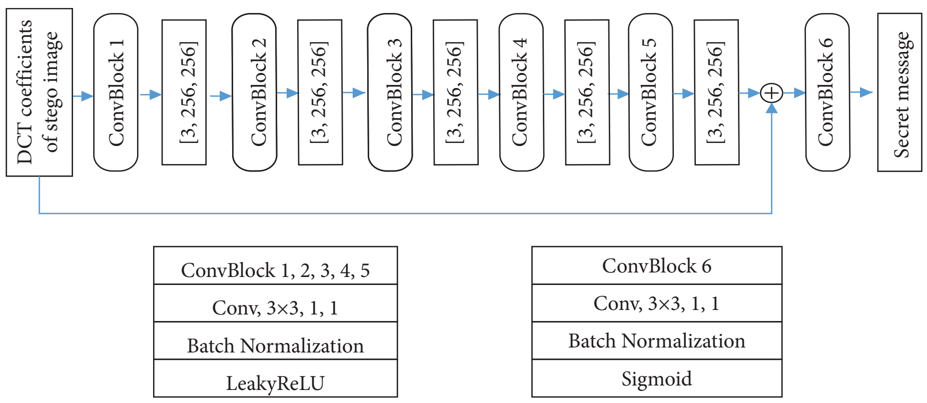

Figure 4: The architecture of the decoder. The first five convolutional blocks utilize LeakyReLU activation function while the last one utilizes sigmoid activation function.

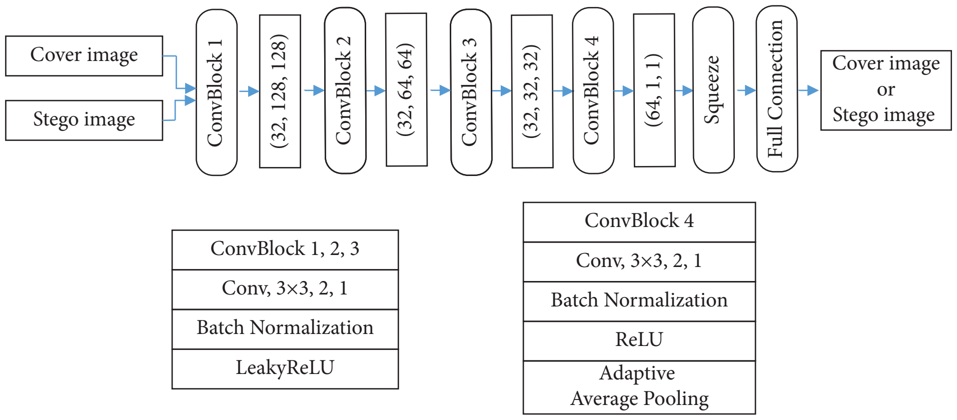

Figure 5: The architecture of the discriminator. “Squeeze” is to compress the dimension of tensor.

where $L(x,y)$, $C(x,y)$, $S(x,y)$ represent brightness comparison, contrast comparison, and structural comparison, respectively, while l, m, and n are used to adjust the weight of SSIM. Usually, we set $l=m=n=1$. $L(x,y)$, $C(x,y)$, and $S(x,y)$ are calculated by

$$L(x,y)=\frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1},$$
$$C(x,y)=\frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2+\sigma_y^2+C_2}, \tag{8}$$
$$S(x,y)=\frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3},$$

where μ_x and μ_y represent average pixel value, σ_x and σ_y represent standard deviation, σ_x^2 and σ_y^2 represent variance, σ_{xy} is the covariance of x and y, and C_1, C_2, C_3 are constants.

At the same time, in order to ensure the decoding accuracy of the message, we use MSE to measure the difference between the secret message $\mathbf{M}$ and the decoded secret message $\mathbf{M}'$. We define the following message reconstruction loss L_m:

$$L_m=\mathrm{MSE}\left(\mathbf{M},\mathbf{M}'\right). \tag{9}$$

Finally, the adversarial loss L_{adv} is defined by

$$L_{\mathrm{adv}}=-L_d, \tag{10}$$

where L_d represents the discrimination loss, which aims to train the discriminator to distinguish the cover image and the corresponding stego image, we define this loss by the following cross-entropy loss:

$$L_d=-\sum_{i=1}^{2} z_i'\times\log(z_i), \tag{11}$$

where z_1 and z_2 are the softmax outputs of the discriminator while z_1' and z_2' stand for the ground truth labels.

The training steps of HRJS is shown in Algorithm 1.

4. Experimental Results

In this section, we perform lots of experiments to verify the imperceptibility and high robustness of the proposed HRJS. We begin with introducing the implemental details and

```
Input: Secret message;
DCT coefficients of the cover image.
Step 1: Connect the DCT coefficients of the cover image with a binary secret message and input them into the encoder to obtain the DCT coefficients of the stego image;
Step 2: Convert the DCT coefficients of cover and stego images into corresponding spatial images by utilizing the IDCT module;
Step 3: Input the spatial stego image into the attack module to obtain the attacked stego image;
Step 4: Convert the attacked stego image into DCT coefficients;
Step 5: Input the DCT coefficients of attacked stego image into the decoder to obtain the decoded secret message;
Step 6: Feed spatial cover and stego image into the discriminator;
Step 7: Update network parameters alternately: $\theta_d \leftarrow \theta_d - \eta \times \partial L_d / \partial \theta_d$, $\theta_{total} \leftarrow \theta_{total} - \eta \times \partial (\lambda_c \times L_c + \lambda_m \times L_m + \lambda_a \times L_{adv}) / \partial \theta_{total}$.
```

Algorithm 1: Training steps of HRJS.

evaluation metrics, and then we show a comprehensive performance analysis of HRJS when suffers from various attacks. Finally, we provide many parameter selection and ablation tests.

4.1. Implemental Details. We use the MS COCO dataset [32] to train and evaluate our model. MS COCO is an RGB color image dataset and we randomly select 40000 images as our training set, 1000 images as the validation set, and 5000 images as the testing set. The images were uniformed to the size of 256×256, then compressed with *QF* (quality factor) = 75 (JPEG_75) by Matlab.

The model has been optimized by Adam optimizer. We set the learning rate as 0.0001, and batch size as 16 to adapt our devices. Each epoch includes 2500 iterations and the whole training process has 100 epochs. At the end of the training, the model has already converged sufficiently.

4.2. Evaluation Metrics. We evaluate our method along with three metrics, which are commonly used to evaluate deep steganography methods: capacity, de codi ng accuracy, and image.quality. We utilize bits per channel (bpc) to evaluate the capacity. For a steganography algorithm, we have to make sure that the decoded secret message and secret message are as similar as possible, consequently, the decoding accuracy is undoubtedly a fundamental metric. Moreover, we utilize peak signal-to-noise ratio (PSNR) to evaluate the distortion of the stego image, which is a metric commonly used to measure image quality. PSNR can be calculated by:

$$\mathrm{PSNR} = 10 \times \log_{10} \frac{\left(2^n - 1\right)^2}{\mathrm{MSE}}, \tag{12}$$

where n is the number of bits per pixel, generally be set as 8. The larger the value, the smaller the distortion.

Due to most errors, sensitivity-based quality assessment methods (such as MSE and PSNR) utilize linear transform to decompose image signals, which can not well reflect human visual characteristics. So we also utilize SSIM, which is more relevant to the perception of the human eyes to measure image quality. MSE and SSIM have been described previously.

4.3. Performances Analysis

4.3.1. Imperceptibility of the Image Steganography. As we all know, the most fundamental and intuitive indicator of image steganography is imperceptibility. Therefore, we randomly select a cover image from the MS COCO dataset and obtain corresponding stego images by embedding secret messages with different payloads in Figure 6. In particular, to better visually observe secret messages, we choose three clear images from the MNIST dataset [33] and process them into binary images to replace random secret messages. The differences between the cover image and corresponding stego images are boosted three times for illustration. We can obviously observe that the differences between the cover image and corresponding stego images is very small in magnitude. This shows that HRJS has a well imperceptibility, and its process of steganography cannot be recognized by human eyes. Even when 1 bpc secret message is embedded, the quality of the stego image is better preserved. In addition, Table 1 shows the image quality and decoding accuracy of embedding secret messages with different payloads in the case of no attack; we can clearly observe that when the embedding capacity is 1/3 bpc, and the values of SSIM, PSNR, and decoding accuracy are as high as 0.9908, 40.39 and 0.9999, respectively. Even if 1 bpc secret message is embedded, our method can still maintain high image quality and achieve almost 100% decoding accuracy for extracting secret messages. The above-given visual and quantitative results verify that our HRJS has high imperceptibility.

4.3.2. Robustness against JPEG Compression. We train several different models and the results of image quality and decoding accuracy are presented in Table 2. We set $\beta = 1$, $\lambda_c = 1$, $\lambda_m = 3$, and $\lambda_a = 1$ in this experiment. The determination of those parameters would be discussed in the next subsection. We can observe that the image quality is well maintained under JPEG compression, especially under JPEG compression with QF = 95, the values of SSIM and PSNR can reach up to 0.9602 and 34.16, respectively. Even under a heavy JPEG compression with QF = 50, the values of SSIM and PSNR can be up to 0.8791 and 28.85, respectively. It is worth mentioning that the overall decoding accuracy is close to 100%. The above-given analysis reflects our method is robust to JPEG compression.

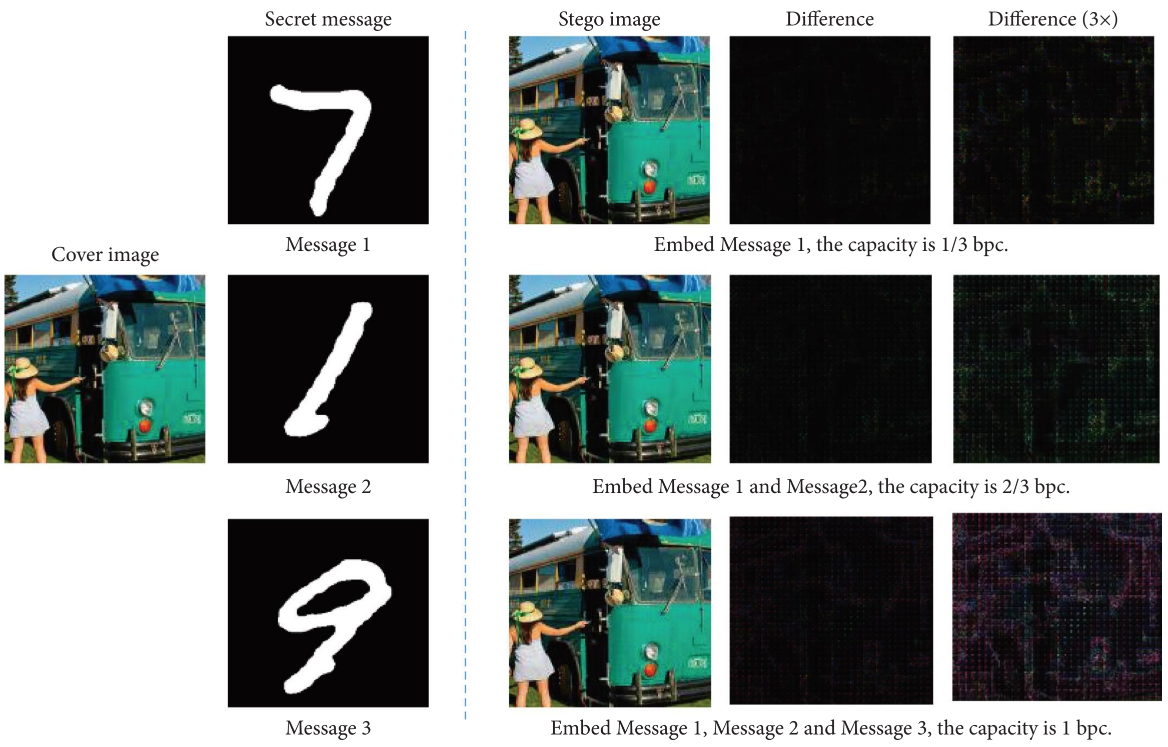
Secret message
Stego image
Difference
Difference (3×)
Cover image
Message 1
Embed Message 1, the capacity is 1/3 bpc.
Message 2
Embed Message 1 and Message2, the capacity is 2/3 bpc.
Message 3
Embed Message 1, Message 2 and Message 3, the capacity is 1 bpc.

FIGURE 6: Example of embedding secret messages with different payloads. "Message 1," "message 2," and "message 3" are binary secret images selected from NMIST dataset, and when each of them is embedded, the embedding capacity is increased by 1/3 bpc.

TABLE 1: The image quality and decoding accuracy of embedding secret messages with different payloads in the case of no attack. The embedded secret messages come from MNIST dataset, and the size of them are adjusted to 256×256 by Matlab. "Decoding_acc" means the decoding accuracy.

Payload	SSIM	PSNR	MSE	Decoding_acc
1/3 bpc	0.9908	40.39	0.0001	0.9999
2/3 bpc	0.9485	32.68	0.0006	0.9991
1 bpc	0.8110	25.97	0.0027	0.9956

TABLE 2: The image quality and decoding accuracy at 1/3 bpc under JPEG compression with different quality factors. The embedded secret messages also come from MNIST dataset.

Attacks	SSIM	PSNR	MSE	Decoding_acc
QF = 95	0.9602	34.16	0.0004	0.9990
QF = 75	0.8846	28.80	0.0015	0.9962
QF = 50	0.8791	28.85	0.0015	0.9961

Figure 7 shows the decoded secret message, we can see that the secret messages can be recovered in high visual quality even if the stego images have gone through various heavy attacks. The numbers contained in the recovered secret messages are clearly recognizable.

Owning to JPEG compression discards a lot of trivial information of the cover image, it causes more disturbance to the stego image compared with other attacks. So when the stego image suffers from JPEG compression, it will cause larger image distortion, and the secret message is more difficult to recover. So we especially show the image quality and decoding accuracy of JPEG compression with different payloads in Table 3. In order to explore a more general situation, the embedded secret messages in Tables 3–5 are randomly generated binary messages.

As shown in Table 3, we can see that, at the same quality factor, the more information embedded, the lower the image quality and decoding accuracy. For example, under the JPEG_75 compression at 1/3 bpc payload, the model achieves 4.3% and 7.27% higher decoding accuracy than the two other payloads, respectively. Similarly, at the same payload, the smaller the quality factor, the greater the information loss, and the lower the image quality and decoding accuracy. In addition, we can observe that binary images from MNIST datasets are easier to embed and extract than randomly generated secret messages. For example, as shown in Table 2, the value of decoding accuracy can be up to 0.9990 under the JPEG_95 compression at 1/3 bpc payload, this accuracy drops from 0.9990 to 0.9778 in Table 3. The reason for this phenomenon is that the binary images from MNIST datasets have less information and strong texture regularity, which is easier to train the network. The experimental results in Table 3 also show that our HRJS has high robustness despite the presence of heavy JPEG compression.

Figure 8 presents the results of PSNR and SSIM of different payloads with different JPEG quality factors. The experimental results show that the image quality decreases with the increase of payload. We can draw a conclusion that no matter with a small payload or a relatively larger payload, the image quality is still pretty good. It proves that HRJS has good robustness against JPEG compression.

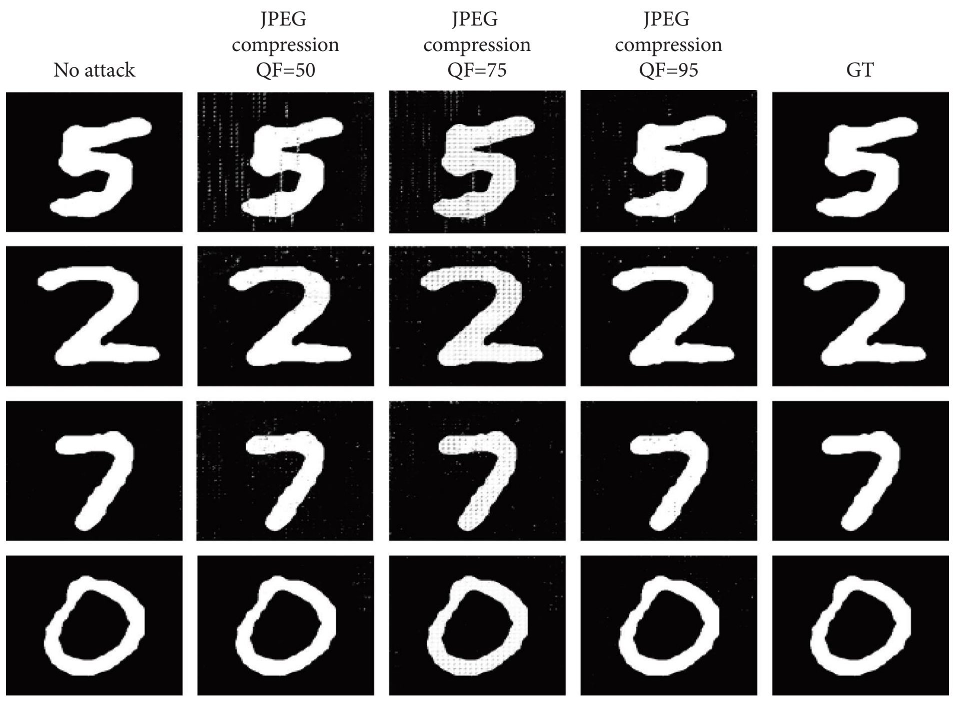

FIGURE 7: Secret messages recovery of the HRJS under various attacks. "GT" means the original image, the four rows represent four different examples.

TABLE 3: The image quality and decoding accuracy of JPEG compression with different payloads.

QF	Metrics	1/3 bpc	2/3 bpc	1 bpc
95	Decoding_acc	0.9778	0.9622	0.9599
	PSNR	29.91	28.49	28.12
	SSIM	0.9058	0.8690	0.8583
75	Decoding_acc	0.9440	0.9010	0.8713
	PSNR	28.26	25.62	25.43
	SSIM	0.8662	0.7665	0.7564
50	Decoding_acc	0.9214	0.9154	0.7951
	PSNR	27.26	24.46	23.99
	SSIM	0.8319	0.7130	0.6846

TABLE 4: The decoding accuracy under different JPEG compression attacks where HRJS is, respectively, implemented without and with attack module. "—" denotes no JPEG compression.

	—	QF = 95	QF = 75	QF = 50
Without	0.9827	0.6252	0.5367	0.5245
With	—	0.9778	0.9440	0.9214

TABLE 5: The effect of different λ_m. "—" denotes no JPEG compression.

QF	Metrics	$\lambda_m = 1$	$\lambda_m = 2$	$\lambda_m = 3$	$\lambda_m = 4$
—	Decoding_acc	0.9792	0.9813	0.9827	0.9872
	PSNR (dB)	32.44	32.30	32.41	32.18
	SSIM	0.9538	0.9521	0.9534	0.9503
75	Decoding_acc	0.5000	0.9076	0.9440	0.9463
	PSNR (dB)	33.27	28.63	28.26	28.09
	SSIM	0.9619	0.8739	0.8662	0.8562

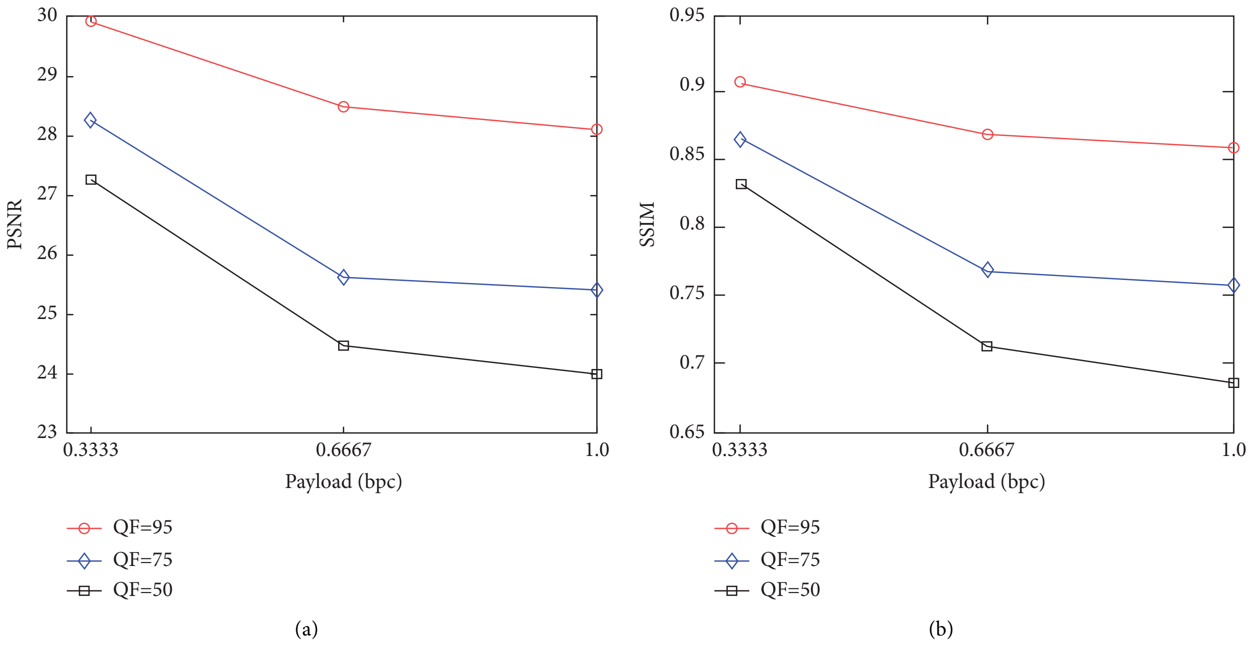

Figure 8: The PSNR and SSIM of different payloads with different JPEG quality factors. As the payload increases, the PSNR and SSIM are decreasing, and it shows that image quality would decrease with a bigger payload. (a) PSNR. (b) SSIM.

4.3.3. Security Analysis. We randomly select 500 cover and 500 stego images and train the common steganalysis tools StegExpose [34] to measure the security of our methods. As shown in Figure 9, we utilize the receiver operating characteristic (ROC) curves to show the detection results. These results show that StegExpose does not work well when QF = 95 or no attack, and our method has certain security when encountering slight JPEG compression.

4.4. Further Analysis. In order to further verify the security of our method, we utilize the discriminator trained by HRJS as a steganalyzer to distinguish the stego images from the cover images. The detection accuracy of this discriminator is 63.1% in the case of 1/3 bpc and without attack. This indicates that HRJS has certain security. Furthermore, we utilize two advanced deep steganalysis methods YangNet [35] and WISERNet [36] to detect the stego images. Unfortunately, these two methods can easily distinguish the stego images, and the detection accuracy is close to 100%. The reasons for this phenomenon are as follows: (1) the end-to-end steganography method has a larger embedding capacity, which will inevitably lead to the loss of security. (2) Incorporating the attacker model improved the robustness and sacrifice the security. (3) The security also depends on the weight of the discriminator of the loss function. We will focus on improving the security of end-to-end JPEG steganography with the same payload situation in the future.

4.4.1. The Effectiveness of Attack Module. To verify the effectiveness of the attack module, we train HRJS without and with the attack module then make experiments on the testing set with different JPEG compression attacks. According to the results shown in Table 4, the attack module can improve the decoding accuracy by 0.3526, 0.4073, and 0.3969 under the three quality factors, which proves the attack module is very essential in this architecture.

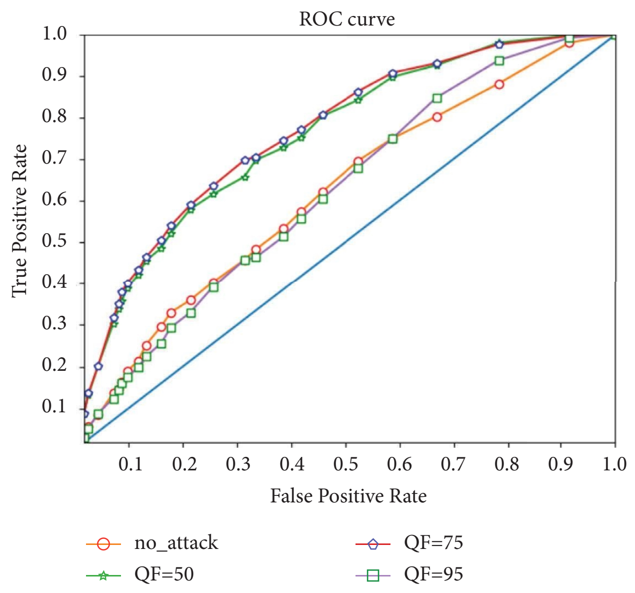

Figure 9: The ROC curve detected by steganalyzer StegExpose at 1/3 bpc.

4.4.2. Different Weights of the Message Reconstruction L_m. λ_m is used to adjust the importance of the message reconstruction L_m. Different λ_m would make an obvious effect

Table 6: The effect of different β.

Metrics	$\beta = 0$	$\beta = 0.01$	$\beta = 0.1$	$\beta = 1$	$\beta = 10$	$\beta = 100$
Decoding_acc	0.9869	0.9857	0.9618	0.9827	0.9766	0.9493
PSNR (dB)	31.86	32.24	32.69	32.41	32.24	32.04
SSIM	0.9462	0.9513	0.9557	0.9534	0.9508	0.9490

on decoding accuracy and image quality, we set $\lambda_c = 1, \lambda_a = 1$, and different λ_m in this experiment to explore its effect. Decoding accuracy and image quality conflict with each other. Image quality would be lower if decoding accuracy was higher, so we have to adopt a trade-off strategy. According to experimental results shown in Table 5, the secret message can not be extracted correctly with $QF = 75$ compression attack when $\lambda_m = 1$. We choose $\lambda_m = 3$ empirically to perform all the experiments. The parameter can be determined by application scenarios. For instance, if there is a higher requirement for image quality, choose $\lambda_m = 2$ will be better.

4.4.3. Different Weights of the Image Reconstruction Loss L_c. We set $\lambda_c = 1$, $\lambda_m = 3$, $\lambda_a = 1$ and different β to explore the effect of the different weights of the image reconstruction loss L_c. According to the result shown in Table 6, both of $\beta = 0$ and $\beta = 1$ perform well on decoding accuracy but $\beta = 1$ get a better performance on SSIM. We choose $\beta = 1$ to perform all the experiments. The determination of β depends on applications, $\beta = 0$ is a preferred taking account of the decoding accuracy.

5. Conclusion

In this paper, we propose an end-to-end JPEG steganography method based on adversarial training, whose embedding and extracting message processes are both realised by a neural network. Besides, to enhance robustness, we insert an attack module to force the neural network to automatically learn how to correctly recover the secret message after being attacked. The proposed method greatly improves the embedding capacity and decoding accuracy. Comprehensive experimental results demonstrate that the end-to-end method is feasible with high-capacity JPEG steganography and has an obvious advantage in terms of robustness against JPEG compression at the same time. Moreover, we explore the effects on the different parts of the loss function. In our future work, we would explore more advanced and complex network structures to propose a more robust model. We will also focus on improving the security of end-to-end JPEG steganography at a large payload.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the NSFC under Grant no. 62102462/62102100, the Guangdong Basic and Applied Basic Research Foundation under Grant no. 2022A1515010108, and Research Project of Guangdong Polytechnic Normal University under Grant no. 2022SDKYA027/2021SDKYA127.

References

[1] Y. Wang, W. Zhang, W. Li, X. Yu, and N. Yu, "Non-additive cost functions for color image steganography based on inter-channel correlations and differences," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2081–2095, 2020.

[2] P. Eze, U. Parampalli, R. Evans, and D. Liu, "Integrity verification in medical image retrieval systems using spread spectrum steganography," in *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pp. 53–57, Ottawa ON, Canada, June 2019.

[3] M. Dalal and M. Juneja, "Steganography and steganalysis (in digital forensics): a cybersecurity guide," *Multimedia Tools and Applications*, vol. 80, no. 4, pp. 5723–5771, 2021.

[4] J. Peng, Y. Jiang, S. Tang, and F. Meziane, "Security of streaming media communications with logistic map and self-adaptive detection-based steganography," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 4, pp. 1–1973, 2019.

[5] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," *EURASIP Journal on Information Security*, vol. 2014, no. 1, pp. 1–13, 2014.

[6] B. Li, M. Wang, J. Huang, and X. Li, "A new cost function for spatial image steganography," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pp. 4206–4210, Paris, France, October 2014.

[7] V. Sedighi, R. Cogranne, and J. Fridrich, "Content-adaptive steganography by minimizing statistical detectability," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 221–234, 2016.

[8] L. Guo, J. Ni, W. Su, C. Tang, and Y.-Q. Shi, "Using statistical image model for jpeg steganography: uniform embedding revisited," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2669–2680, 2015.

[9] W. You, H. Zhang, and X. Zhao, "A siamese cnn for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 291–306, 2021.

[10] S. Tan, W. Wu, Z. Shao, Q. Li, B. Li, and J. Huang, "Calpa-net: channel-pruning-assisted deep residual network for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 131–146, 2021.

[11] J. Butora and J. Fridrich, "Reverse jpeg compatibility attack," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1444–1454, 2020.

[12] W. Tang, S. Tan, B. Li, and J. Huang, “Automatic steganographic distortion learning using a generative adversarial network,” *IEEE Signal Processing Letters*, vol. 24, no. 10, pp. 1547–1551, 2017.

[13] J. Yang, D. Ruan, J. Huang, X. Kang, and Y. Q. Shi, “An embedding cost learning framework using gan,” *IEEE Transactions on Information Forensics and Security*, vol. 15, no. 99, pp. 839–851, 2020.

[14] J. Yang, D. Ruan, X. Kang, and Y.-Q. Shi, “Towards automatic embedding cost learning for jpeg steganography,” in *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, pp. 37–46, Paris France, July 2019.

[15] T. Filler, J. Judas, and J. Fridrich, “Minimizing additive distortion in steganography using syndrome-trellis codes,” *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 920–935, 2011.

[16] S. Baluja, “Hiding images in plain sight: deep steganography,” *Advances in Neural Information Processing Systems*, vol. 30, pp. 2069–2079, 2017.

[17] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, “Hidden: hiding data with deep networks,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 657–672, Munich, Germany, September 2018.

[18] K. A. Zhang, A. Cuesta-Infante, L. Xu, and K. Veeramachaneni, “Steganogan: high capacity image steganography with gans,” 2019, https://arxiv.org/abs/1901.03892.

[19] R. Zhang, S. Dong, and J. Liu, “Invisible steganography via generative adversarial networks,” *Multimedia Tools and Applications*, vol. 78, no. 7, pp. 8559–8575, 2019.

[20] J. Tao, S. Li, X. Zhang, and Z. Wang, “Towards robust image steganography,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 594–600, 2019.

[21] W. Jiang, D. Hu, C. Yu, M. Li, and Z.-Q. Zhao, “A new steganography without embedding based on adversarial training,” in *Proceedings of the ACM Turing Celebration Conference-China*, pp. 219–223, Hefei China, May 2020.

[22] W. Li, K. Chen, W. Zhang, H. Zhou, Y. Wang, and N. Yu, “Jpeg steganography with estimated side-information,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 2288–2294, 2020.

[23] Q. Giboulot, R. Cogranne, and P. Bas, “Detectability-based jpeg steganography modeling the processing pipeline: the noise-content trade-off,” *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2202–2217, 2021.

[24] L. Guo, J. Ni, and Y. Q. Shi, “Uniform embedding for efficient jpeg steganography,” *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 5, pp. 814–825, 2014.

[25] X. Hu, J. Ni, and Y.-Q. Shi, “Efficient jpeg steganography using domain transformation of embedding entropy,” *IEEE Signal Processing Letters*, vol. 25, no. 6, pp. 773–777, 2018.

[26] W. Su, J. Ni, X. Li, and Y.-Q. Shi, “A new distortion function design for jpeg steganography using the generalized uniform embedding strategy,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 12, pp. 3545–3549, 2018.

[27] Y. Wang, W. Zhang, W. Li, and N. Yu, “Non-additive cost functions for jpeg steganography based on block boundary maintenance,” *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1117–1130, 2021.

[28] C. Zhang, P. Benz, A. Karjauv, G. Sun, and I. S. Kweon, “Udh: universal deep hiding for steganography, watermarking, and light field messaging,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 10223–10234, 2020.

[29] S.-P. Lu, R. Wang, T. Zhong, and P. L. Rosin, “Large-capacity image steganography based on invertible neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10816–10825, Nashville, TN, USA, June 2021.

[30] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[31] M. Tancik, B. Mildenhall, and R. Ng, “Stegastamp: invisible hyperlinks in physical photographs,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2117–2126, Seattle, WA, USA, June 2020.

[32] T.-Y. Lin, M. Maire, S. Belongie et al., “Microsoft coco: common objects in context,” in *Proceedings of the European conference on computer vision*, pp. 740–755, Zurich, Switzerland, September, 2014.

[33] K. Cheng, R. Tahir, L. K. Eric, and M. Li, “An analysis of generative adversarial networks and variants for image synthesis on mnist dataset,” *Multimedia Tools and Applications*, vol. 79, pp. 13725–13752, 2020.

[34] B. Boehm, “Stegexpose-a tool for detecting lsb steganography,” 2014, https://arxiv.org/abs/1410.6656.

[35] J. Yang, X. Kang, E. K. Wong, and Y.-Q. Shi, “Jpeg steganalysis with combined dense connected cnns and sca-gfr,” *Multimedia Tools and Applications*, vol. 78, no. 7, pp. 8481–8495, 2019.

[36] J. Zeng, S. Tan, G. Liu, B. Li, and J. Huang, “Wisernet: wider separate-then-reunion network for steganalysis of color images,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2735–2748, 2019.