


RESEARCH

Open Access



# Robust JPEG steganography based on the robustness classifier

Jimin Zhang<sup>1,2</sup> , Xianfeng Zhao<sup>1,2</sup> and Xiaolei He<sup>1,2\*</sup>

## Abstract

Because the JPEG recompression in social networks changes the DCT coefficients of uploaded images, applying image steganography in popular image-sharing social networks requires robustness. Currently, most robust steganography algorithms rely on the resistance of embedding to the general JPEG recompression process. The operations in a specific compression channel are usually ignored, which reduces the robustness performance. Besides, to acquire the robust cover image, the state-of-the-art robust steganography needs to upload the cover image to social networks several times, which may be insecure regarding behavior security. In this paper, a robust steganography method based on the softmax outputs of a trained classifier and protocol message embedding is proposed. In the proposed method, a deep learning-based robustness classifier is trained to model the specific process of the JPEG recompression channel. The prediction result of the classifier is used to select the robust DCT blocks to form the embedding domain. The selection information is embedded as the protocol messages into the middle-frequency coefficients of DCT blocks. To further improve the recovery possibility of the protocol message, a robustness enhancement method is proposed. It decreases the predicted non-robust possibility of the robustness classifier by modifying low-frequency coefficients of DCT blocks. The experimental results show that the proposed method has better robustness performance compared with state-of-the-art robust steganography and does not have the disadvantage regarding behavior security. The method is universal and can be implemented in different JPEG compression channels after fine-tuning the classifier. Moreover, it has better security performance compared with the state-of-the-art method when embedding large-sized secret messages.

**Keywords** Steganography, Robust steganography, JPEG robust steganography

## 1 Introduction

Steganography is the technology of hiding secret messages in multimedia files, such as images, audio, or videos, without introducing the trace of modification. Its counterpart, steganalysis, is the technique of detecting whether the testing multimedia files have been modified to carry secret messages. Currently, the most advanced

steganography algorithm is adaptive steganography. The latest steganalysis algorithms are based on high-dimensional feature sets and the ensemble classifier [1–5] or deep learning methods [6–10], which are able to detect adaptive steganography. Adaptive steganography usually contains two parts: the cost function and the information embedding algorithm. The design of the cost function is related to the impact of embedding to its counterpart, steganalysis, so that the pixels or coefficients that are hard to detect by steganalysis after modification have low costs. The design of the cost function could be divided into two disciplines. One is defined empirically by assigning low-cost values to pixels or coefficients in the complex content areas without considering the specific steganalysis features [11–15]. Another method tries

\*Correspondence:

Xiaolei He  
[hexiaolei@iie.ac.cn](mailto:hexiaolei@iie.ac.cn)

<sup>1</sup> Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100085, China

<sup>2</sup> School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100085, China

to attack a specific steganalysis by setting the pixels or coefficients that weaken detection power as small costs [16–18]. After the cost function is calculated, the information embedding algorithm, e.g., the syndrome trellis codes (STC) [19], is used to realize information embedding with modifications introducing minimal costs.

Adaptive steganography can achieve information embedding with high-security performance, but it is based on the assumption that the transmission channel is error-free. In e-mail and lossless file sharing scenarios, adaptive steganography could be directly used. With the development of social networks, e.g., Facebook or Twitter, sharing photos on social networks becomes more convenient and popular. Unfortunately, the compression process in social networks changes the extraction domain of steganography. Adaptive steganography could not be directly applied for the failure of information extraction.

Researchers have recently proposed robust steganography to achieve information embedding in the lossy channels. In Zhang's methods [20–22], several techniques of watermarking technology are used to achieve steganography with robustness. In [23], Yu et al. introduced the generalized dither modulation-based robust steganography (GMAS), which uses ternary embedding and expands the embedding domain of the dither modulation-based method proposed in [22]. In [24], Kin-Cleaves et al. invented a dual STC method that reduces the channel error in stego images before information extraction. In [25], the authors studied that when the embedding processing is fully known to the embedder, the 100% correct information embedding and extraction can be achieved; however, the embedding process in the work does not include the lossy operation in the spatial domain, which is frequently used by social networks. In [26], the authors proposed a method that uses an auto-encoder to learn the reverse compression process. In [27], Zhao et al. proposed the transport channel matching (TCM)-based robust steganography, which uploads and downloads images from websites several times. Then, adaptive steganography is used to embed the message. However, because the stego image does not experience recompression iterations, the modified coefficients could still be changed by recompression. To overcome the blindness of the embedding process in the TCM robust steganography, Zhang et al. [28] proposed the robustness cost function, which reduces the modifications that break the robust domain formed by TCM. In [29], to minimize the error rate of channels, Zeng et al. proposed a method that uses the image after recompression as the embedding domain. In the method, the cost function is adjusted to avoid embedding in the non-robust areas and non-robust coefficients.

In the robust steganography algorithms that have been proposed, most methods require the cover images to be uploaded to the channel at least once to improve

the robustness. The supervisor may monitor the behavior of uploading images with the same contents multiple times, which increases the likelihood of the sender being detected. Designing a robust steganography that does not need to upload cover images into channels before uploading stego images is critical. One approach to solve this problem is to utilize deep learning to construct the compression process locally. Then the channel-related characteristics can be used without uploading. Another way is that the embedding is robust enough to withstand unknown compression processes, which is called the general robust steganography in [30]. However, it might perform poorly on a specific compression channel, for the specific compression process has not been considered.

The usage of deep learning in robust steganography to simulate the compression can be firstly found in [26]. The method utilizes an auto-encoder to learn the transformation relationship between the JPEG image after and before compression. The adaptive BCH encoding method that selects the coding parameter according to the content of cover images is proposed. In the method, the auto-encoder is employed to predict 64 DCT coefficients before compression in a DCT block. For embedding the secret information, the uploaded image is needed to generate robust images. For some, JPEG recompression channels may apply complicated processes, and the prediction error is large, which restricts the improvement in robustness of this method.

The main contribution of our method is that we utilize deep learning to develop a protocol message sharing-based robust steganography. To select the DCT blocks with good robustness, the deep learning-based classifier which learns the characteristics of the compression channel is used. The selected DCT blocks form the embedding domain of secret messages. The protocol message related to the selection information is generated and embedded using general robust steganography. To improve the possibility of correctly restoring important protocol messages, a robustness enhancement method is proposed. It performs modification in the low-frequency domain based on the prediction result of the learned classifier. By implementing the classifier that learns the specific processes of compression for embedding, the robustness of proposed robust steganography is better compared with general robust steganography. The proposed method does not need to upload cover images to compression channels, which eliminates the behavior security issues existing in most robust steganography.

The rest of this paper is organized as follows. In Section 2, the preliminaries are presented. In Section 3, the proposed method is introduced, which includes the overall process of proposed methods, the architecture of the learning network, the procedure of proposed robustness enhancement methods, and the process of protocol message embedding.

Section 4 is the experimental part of this paper, which gives experimental results and the discussions. The conclusion is given in Section 5.

## 2 Preliminaries

In this section, we give the preliminaries of this paper, which include the JPEG compression process, the STC embedding, and the error-correcting codes. The bold letters refer to matrices and vectors, and the non-bold letter with subscripts refers to an element of matrices or vectors. The symbols utilized in this paper are listed in Table 1. The side information representing the position of message embedding blocks is called the protocol message.

### 2.1 JPEG recompression

There usually are two processes in JPEG recompression: spatial image restoration and JPEG image compression. Suppose an  $8 \times 8$  DCT block before JPEG recompression is  $D_o$ , and the quality factor of the image is  $q_o$ , which corresponds to the quantization table  $Q_o$ . The process of restoring the spatial domain  $S_o$  can be denoted as

$$S_o = [\text{IDCT}(D_o \times Q_o) + 128], \quad (1)$$

where  $\text{IDCT}(\mathbf{x})$  is the two-dimensional inverse DCT transform and  $[\mathbf{x}]$  is the rounding operation. The multiplication here is the multiplication for the corresponding elements of two matrices. In this paper, we call the spatial values before the rounding operation as unquantized spatial values.

After restoring the rounded spatial values, a clipping operation is performed to generate the spatial image  $S'_o$ . The clipping operation sets the rounded spatial in the range of  $[0, 255]$  by clipping the outside value as the nearest integer boundary value. Suppose the quality factor of the JPEG compressor in the channel is  $q_c$ , which corresponds to the quantization table  $Q_c$ . The process of generating the quantized DCT coefficients  $D_c$  is

$$D_c = [\text{DCT}(S'_o - 128) \div Q_c], \quad (2)$$

where  $\text{DCT}(\mathbf{x})$  is the two-dimensional DCT transform, and the division here is the division for the corresponding elements of two matrices.

**Table 1** The list of symbols used and their meanings

Symbol	Meaning
$X$	Quantized DCT coefficients
$D$	Unquantized DCT coefficients
$Q$	Quantization table
$S$	Rounded spatial pixels restored from DCT coefficients

To increase the compression quality, some social networks or JPEG compression toolboxes employ other complicated compression processes. For example, the JPEG compression library mozjpeg<sup>1</sup> uses the trellis optimization method to decrease the file size of recompressed JPEG while maintaining the quality of spatial content. The operation causes more changed DCT coefficients after recompression and increases the errors of recompressed stego images, requiring further improved robustness in robust steganography.

### 2.2 The syndrome trellis codes

The syndrome-trellis codes (STC) [19] are convolution codes described in the dual domain. It solves the payload limited sender and the distortion limited sender embedding problems with a small coding loss. The parity-check matrix  $H \in \{0, 1\}^{m \times n}$  of a binary syndrome-trellis code of length  $n$  and co-dimension  $m$  is obtained by placing a small submatrix  $\hat{H}$  of size  $h_s \times w$  along the main diagonal as in Fig. 1. The height  $h_s$  of the submatrix is called the constraint height. The width of  $\hat{H}$  is dictated by  $m/n$ . If  $m/n$  equals to  $1/k$  for some  $k \leq n$ , the value of  $w$  equals  $k$ . The codeword of an STC is represented as a unique path through the syndrome trellis. The syndrome trellis can be parameterized by message  $m$  and can present members of the coset  $\mathcal{C}(m) = \{z \in \{0, 1\}^n, Hz = m\}$ . The syndrome trellis with the  $\hat{H}$  in Fig. 1 and a random  $m$  can be seen in Fig. 1. The binary embedding can be optimally solved by the Viterbi algorithm. The algorithm consists of two parts, the forward and the backward part. After the forward part, the shortest path through the entire trellis is perceived. In the backward part, the shortest path is traced back, and the parities of the closest stego object are recovered.

### 2.3 BCH codes and RS codes

The Bose, Chaudhuri, and Hocquenghem (BCH) codes are a class of random error-correcting cyclic codes. The code is a remarkable generalization of the Hamming codes for multiple-error correction. For any positive integers  $m$  ( $m \geq 3$ ) and  $t$  ( $t < 2^{m-1}$ ), there exists a binary BCH code with parameters as follows: the block length  $n = 2^m - 1$ , the number of parity-check digits  $n - k \leq mt$ . The code is capable of correcting any combination of  $t$  or fewer errors in a block of  $n = 2^m - 1$  digits. In this paper, the parameter  $(n, k)$  means that the length of the block digits is  $n$ , the number of bits of information is  $k$ , and the number of parity-check digits is  $n - k$ . The special subclass of  $q$ -ary BCH codes for which  $m = 1$  are called Reed-Solomon (RS) codes. They have been widely used in both digital communication and storage systems. For a Reed-Solomon code

<sup>1</sup> <https://github.com/mozilla/mozjpeg>

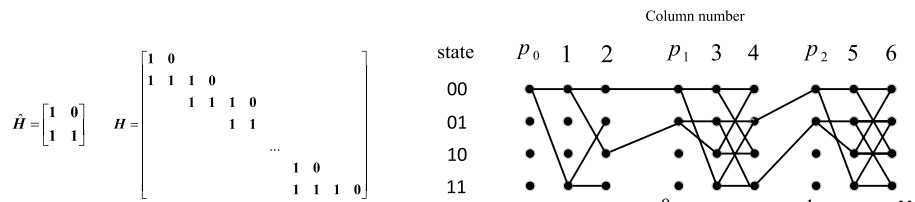


Fig. 1 The parity-check matrix of STC and its syndrome trellis

with symbols from  $GF(q)$ , the parameters are as follows: the maximum block length  $n = q - 1$ , the number of parity-check symbols  $n - k = 2t$ , and the code is capable of correcting  $t$  or fewer symbol errors. The length of symbols of RS encoding used in this paper is 8 bits. The parameter  $(n, k)$  means the length of symbols in a block of RS codes is  $n$ , and the number of parity-check symbols is  $n - k$ . More detailed information about BCH and RS error-encoding codes can be found in [31].

### 3 Proposed method

This section is divided into four parts. Firstly, the framework of the proposed robust steganography algorithm is presented in Section 3.1. Secondly, the network structure of the proposed robustness classifier and its training process are described in Section 3.2. Thirdly, the proposed robustness enhancement method is explained in Section 3.3. Finally, the process of protocol message embedding is given in Section 3.4.

#### 3.1 The framework of the proposed method

In this paper, with the output of the robustness classifier, a robust steganography based on the robust block selection

and the protocol message embedding is proposed. The framework of the proposed method is shown in Fig. 2.

The embedding process can be described as follows. First, using a secret key shared by the steganographer and the receiver, all the DCT blocks in cover images are randomly divided into two disjoint groups, which are named group1 and group2 separately. The length of the two groups is equal, which means each group contains half the number of all DCT blocks in cover images. Then, the DCT blocks in group1 are inputted into the trained classifier that can classify whether the input block is robust. The bit representation is 1 if the block is predicted as non-robust and 0 if it is predicted as robust. The bits result predicted by the classifier are concatenated into byte streams as protocol messages, which will be embedded into the DCT blocks in group2. To increase the success possibility of protocol message restoration, the DCT blocks in group2 experience the robustness enhancement operation that improves the robustness. After error-correcting code encoding, the protocol messages are embedded into robustness-enhanced DCT blocks in group2 with the general robust steganography. The DCT blocks in group1 will be further grouped by the prediction results of the classifier. The secret message

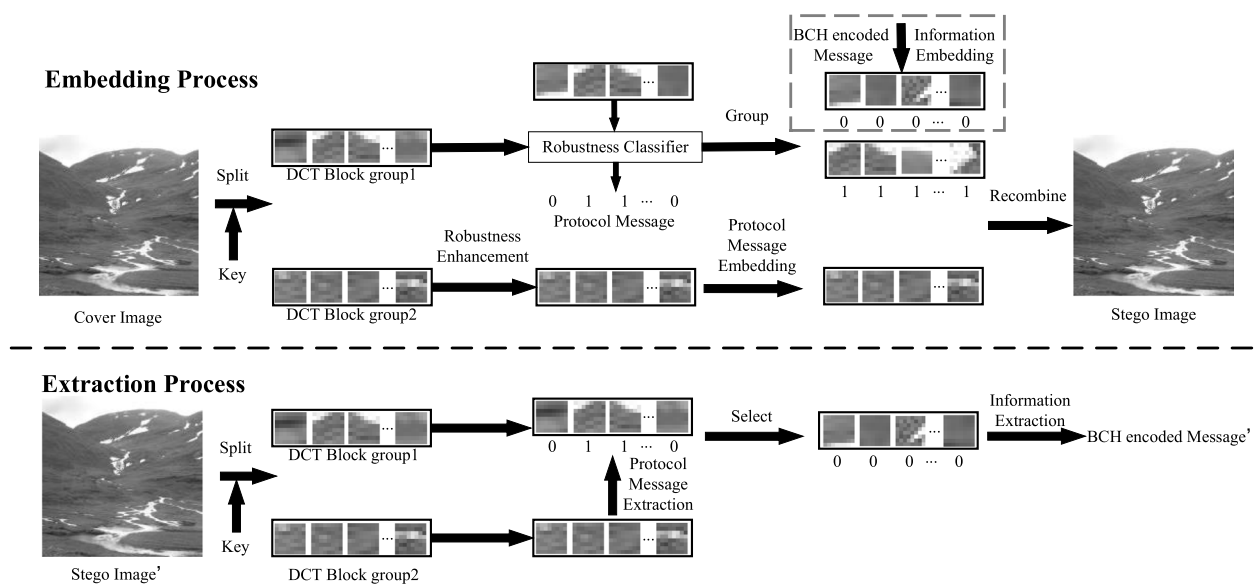


Fig. 2 The framework of the proposed method

is encoded by the error-correcting code encoder and randomly permuted. Then in group1, they are embedded into the DCT blocks predicted robust by the classifier with adaptive steganography. Finally, after embedding secret messages and protocol messages, the DCT blocks in group1 and group2 are combined to generate stego images.

The extraction process is described as follows. Firstly, the receiver receives the stego image recompressed by channels. Then, the DCT blocks of recompressed stego images are divided into two groups using the same key in the embedding process. The protocol message is extracted from DCT blocks in group2, and the prediction results of DCT blocks in group1 can be restored by the extracted protocol message. The DCT blocks in group1 with the bit representation as 0 are selected to form the extraction domain. After performing the STC extraction and inverse permutation, the error-correcting code encoded message is extracted. The message can be restored by performing error-correcting code decoding.

Aiming to improve the robustness, general robust steganography usually introduces more distortion compared with non-robust steganography. The reason that we divide DCT blocks into two groups for embedding secret messages and protocol messages separately is as follows. The length of the protocol message is fixed and is related to the size of an image. In our methods, when the payload is large, the adaptive steganography which is more secure is used to embed the message. The robustness is ensured by selecting the robust DCT blocks as the embedding domain by a trained classifier. The length of the protocol message remains the same, and the protocol message is embedded by general robust steganography. Compared with only using general robust steganography for embedding, the combination of two embedding processes can improve security performance when embedding large-sized secret messages. At the same time, through a trained classifier, we select robust DCT blocks as the embedding domain and perform a robust enhancement method. The robust performance is improved, and a higher success extraction rate can be achieved.

### 3.2 The robustness classifier

The first usage of deep learning in robust steganography appears in [26], in which the auto-encoder is utilized to train the network to predict the DCT coefficients before the compression operation. However, instead of obtaining the coefficients before compression, the results in [27] demonstrate that implementing modification without predicting coefficients before compression can also improve the robustness. Moreover, predicting 64 DCT coefficients before compression is more difficult than learning one output of the robustness possibility. The embedding process in [26] requires the image after the compression of channels. It increases the

chance of being detected by analysts analyzing this behavior. Thus to improve the efficiency and robustness of machine learning-based robust steganography and improve behavior security, studying the method that trains the classifier of non-robust possibility and embeds messages without uploading cover images is very meaningful.

From the result in [28], the spatial domain is more related to robustness. In mozjpeg, the trellis quantization is performed to realize the optimal distortion rate, which is related to the distortion function defined in the spatial domain. Thus the input of the deep learning architecture proposed is unquantized spatial values, which are acquired from IDCT operations performed on DCT coefficients in JPEG images. When designing the structure of deep learning, we find out that the structure with six layers of convolution can learn the robustness of the DCT block. Moreover, using residual learning structure [32] and increasing the depth of convolution layers can further increase the detection accuracy. The structure of the proposed deep learning network is shown in Fig. 3. From it, we can see that to capture the spatial feature of unrounded spatial values from a DCT block, the proposed classifier involves at most eight convolution layers. The Type A and Type B structures in Fig. 3 are typical residual learning structures. The ReLU [33] is used for learning the non-linear function determining whether the DCT block is robust. The last two fully connected layers and the softmax function convert the feature into the possibility that the DCT block is robust. The softmax function is defined as

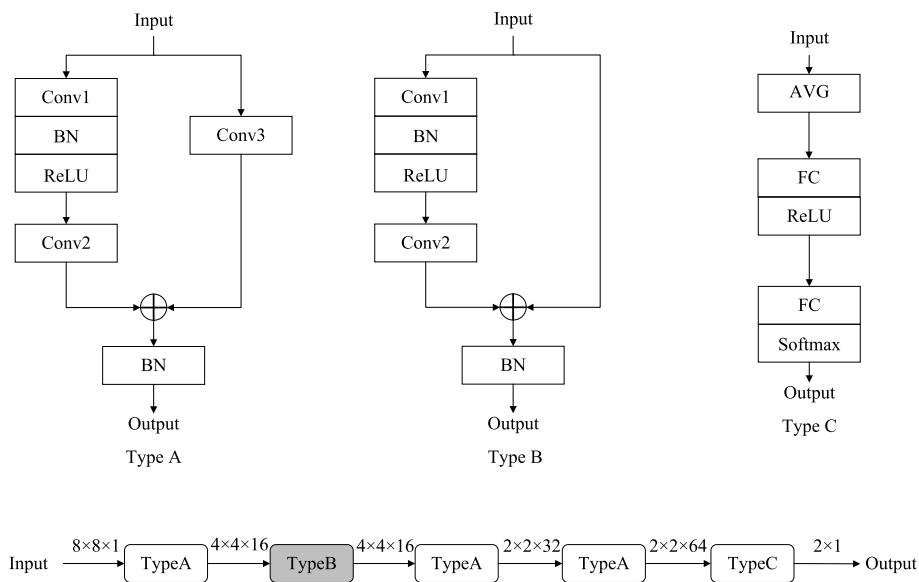
$$\text{Softmax}(x_i) = \frac{e^{x_i}}{e^{x_1} + e^{x_2}}, i \in \{1, 2\}, \quad (3)$$

where  $x_1$  and  $x_2$  are two outputs of the last fully connected layer. We define that if  $\text{Softmax}(x_1) > \text{Softmax}(x_2)$ , the classifier predicts the DCT block input as the robust block; otherwise, the DCT block is predicted as the non-robust block.

The loss function used by the classifier is the cross-entropy loss. Suppose the possibility of the input block is robust  $P_1 = \text{Softmax}(x_1)$ , and the possibility of the input is non-robust  $P_2 = \text{Softmax}(x_2)$ . The label equals zero means the input block is robust and one when the input is non-robust. The loss function can be calculated as

$$\text{loss} = -w * \text{label} * \log(P_2) - (1 - \text{label}) * \log(P_1). \quad (4)$$

The introduction of  $w$  is to add weight to the training items where  $\text{label} = 1$ . The reason is that the number of robust blocks is usually not equal to the number of non-robust blocks. When the number of non-robust blocks is far less than the number of robust blocks, the loss will be small even though all the non-robust blocks are misclassified. In that case, the gradients are very small to increase



**Fig. 3** The structure of the proposed deep learning network

the correction rate of detecting non-robust blocks. To overcome this problem, the class whose sample size is smaller than another class should have a larger weight. On the other hand, the importance of the false-positive rate and the miss detection rate are different in robust steganography. In order to adjust the balance between the false-positive rate and the miss detection rate, the weight parameter is needed.

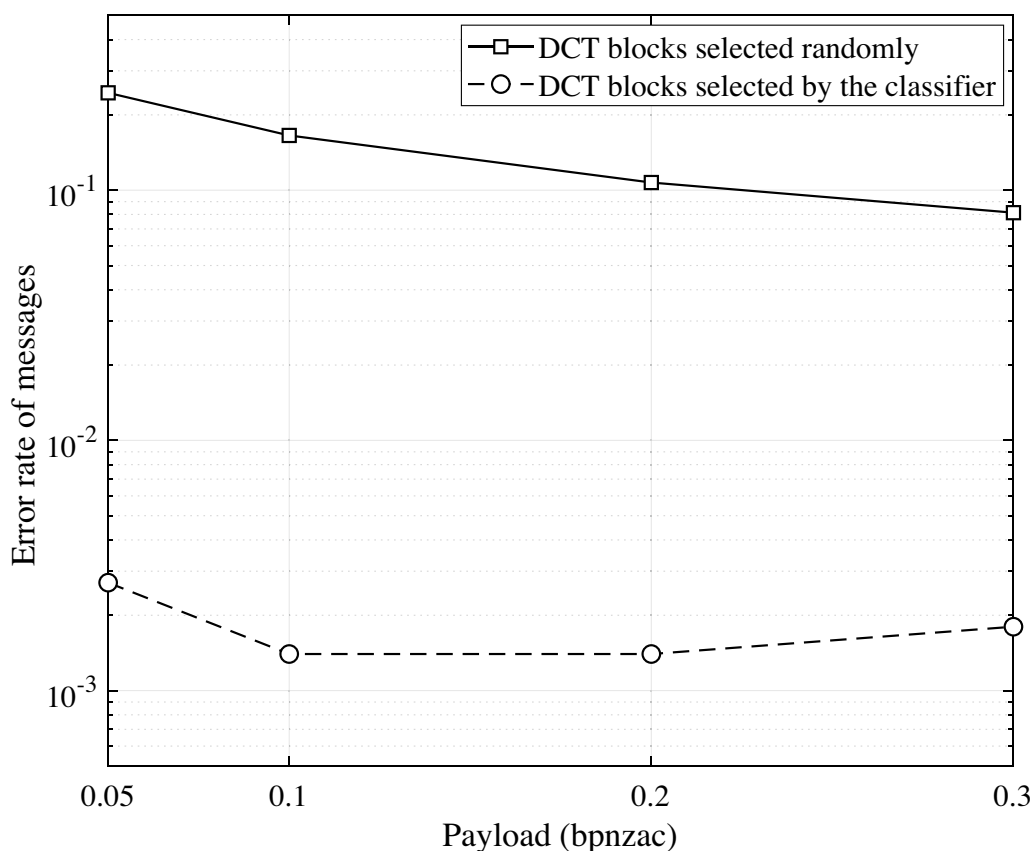
In this study, after the recompression, there are two kinds of situations. Suppose there is a DCT coefficient  $x_i$  which is in the DCT block  $X_{ab}$ . The  $x_i$  is a robust coefficient and  $X_{ab}$  is a robust DCT block if the DCT block  $X_{ab}$  is not changed after recompression. And  $x_i$  is a non-robust coefficient and  $X_{ab}$  is a non-robust DCT block if the DCT block  $X_{ab}$  is changed after recompression. The instance of  $x_i$  being a robust coefficient is represented as  $x_i = R$ . The instance of  $x_i$  being a non-robust coefficient is represented as  $x_i = NR$ . The instance of the classifier predicts the  $x_i$  as a robust coefficient is represented as  $x_i = CR$ . And the instance of the classifier predicts the  $x_i$  as a non-robust coefficient represented as  $x_i = CNR$ . The average possibility of a coefficient is predicted as non-robust by the classifier is

$$P\{x_i = CNR\} = P\{x_i = CNR|x_i = R\}P\{x_i = R\} + P\{x_i = CNR|x_i = NR\}P\{x_i = NR\}, \tag{5}$$

where  $P\{x_i = CNR|x_i = NR\}$  is the true-positive rate of the classifier, and  $P\{x_i = CNR|x_i = R\}$  is the false-positive rate of the classifier. If the  $x_i$  is a non-robust coefficient,  $P\{x_i = NR\}$  is 1; otherwise,  $P\{x_i = NR\}$  is 0. The possibility of non-robust coefficients being detected is equal to the true-positive rate  $P\{x_i = CNR|x_i = NR\}$  of the classifier. To ensure the non-robust coefficients are certainly detected, the true-positive rate of the classifier should be close to 1. To achieve this goal, we set  $w$  as a large value. The setting solves the problem of the sample imbalance and ensures that the miss detection rate, which equals  $1 - P\{x_i = CNR|x_i = NR\}$ , is lower than the false-positive rate. We count the ratio between the number of non-robust blocks and robust blocks in JPEG images from BOSSbase [34] after mozjpeg recompression. The result is that the ratios between the number of robust blocks and non-robust blocks are 16.3 and 10.89 when the quality factors are 85 and 90 respectively. When the quality factor is 85, the detection accuracy, miss detection rate, and false-positive rate of the trained classifier with different  $w$  values are shown in Table 2. The reason that the detection accuracy is closer to the false-positive rate than the miss detection rate is that there are more robust blocks than non-robust blocks. To increase the true-positive rate under the condition of increasing

**Table 2** The detection accuracy, miss detection rate, and false-positive rate with different  $w$

$w$	1	25	50	75
Detection accuracy (%)	98.76	94.30	92.98	92.11
Miss detection rate (%)	20.46	1.57	0.71	0.41
False-positive rate (%)	0.37	5.92	7.35	8.27



**Fig. 4** The error rate of messages when embedding is performed in DCT blocks randomly selected and selected by the classifier

the false-positive rate and solve the problem of sample imbalance, we set the  $w$  as 50.

The training process is as follows. Firstly, 10,000 images are randomly selected from ImageNet [35] datasets. The images are cropped as squared with the minimum side between the height and width. Then, images are reshaped to the size of  $512 \times 512$ , and the *imwrite* function in Matlab is used to generate JPEG images of specific quality factors. The reason that we train the classifier using a public dataset is to show the robustness of our classifier. It means that the performance is well even when the cover images come from a different dataset than the training dataset. In the training process, the images are randomly separated as the training set and the testing set with the numbers 9000 and 1000. The optimizer is the Adam optimizer. The learning rate decay method is polynomial decay<sup>2</sup> with the initial learning rate as 0.001. The decay step is set as 20 epochs. The final learning rate is 0.0001, and cycling is enabled. Because the input size of the network is  $8 \times 8$ ,

a JPEG image is divided into DCT blocks. Supposing the width and the height of the input image are  $W$  and  $H$ , both divisible by 8, DCT coefficients are divided into  $\frac{W}{8} \times \frac{H}{8}$  DCT blocks. The label of each block is acquired by recompressing the image with *mozjpeg*. The batch size of each iteration is 10, and the number of epochs is 40.

To illustrate the robustness improvement within the selected DCT blocks, we first generate JPEG images from BOSSbase using functions in Matlab. Then we use *mozjpeg* to recompress JPEG images. The quality factor used is 85. When embedding is performed separately in DCT blocks selected by the classifier and selected randomly, the error rate in STC extracted messages with different payloads is shown in Fig. 4. From the result, we can see that the recompression error in DCT blocks selected by the classifier is more robust than that in blocks selected randomly, which demonstrates the effectiveness of the proposed method. Besides, we can also observe that when embedding is performed in the DCT blocks that are randomly selected, the error rate of messages decreases as the payload increases. This property can be used when we embed protocol messages, for the embedding domain of protocol message embedding are the DCT blocks randomly selected.

<sup>2</sup> [https://www.tensorflow.org/api\\_docs/python/tf/compat/v1/train/polynomial\\_decay](https://www.tensorflow.org/api_docs/python/tf/compat/v1/train/polynomial_decay)

### 3.3 Robustness enhancement method

When TCM operation is performed, the process of uploading and downloading cover images from websites causes the behavior security problem. The process of TCM can be considered as follows: the method detects the non-robust coefficients first, and then the coefficients modification process is performed to improve the robustness of the JPEG image. Because we have the classifier which can predict whether the DCT block is robust, the TCM process can be simulated with the trained classifier.

To increase the robustness of cover images, we propose a robustness enhancement method named robustness embedding. It functions by modifying the DCT coefficients, which is similar to TCM. The process can be described as follows. Supposing  $\text{phase}_{\text{num}}$  of the DCT coefficient  $x_{ij}$  with phase  $(i, j)$  is  $i + j$ , we set the max  $\text{phase}_{\text{num}}$  value when performing robustness embedding as  $\text{phase}_{\text{max}}$ . Supposing there are  $n_p$  coefficients in which  $\text{phase}_{\text{num}}$  is less than or equal to  $\text{phase}_{\text{max}}$ , we separately perform  $+1, -1, +2, -2, +3, -3 \dots +h, -h$  operation to each DCT coefficient and keep other coefficients unchanged. The operations generate  $2hn_p$  changed DCT blocks. The robustness classifier is used to test the non-robust possibility of each changed block. The unchanged DCT block is also tested. The non-robust possibility of

the unchanged block is set as 0 if it is less than 0.5, for the unchanged DCT block is already robust, and the modifications are not needed. Subsequently, the possibility of the non-changed block is subtracted by the possibility of each changed block separately. The results with negative values are set to 0. Then, the results are divided by the  $x$ -th power of the J-UNIWARD cost  $\rho^x$ . The reason for the division is that we want the changes to be in the DCT coefficients with good security performance. Suppose the non-robust possibility of the DCT block before modification is  $z$ , and the non-robust possibility of the same DCT block after performing  $+m$  in the DCT phase  $(i, j)$  is  $z'_{ij}$ . The value of the increment of robustness (RI) after modification is

$$RI = \frac{z - z'_{ij}}{\rho^x}. \tag{6}$$

For each DCT block, we select  $\alpha$  modifications that have the highest non-zero RI values. When protocol messages are embedded, the influence of  $x$  on the robustness performance and distortion after robustness embedding is shown in Fig. 5. The embedding domain is 21 DCT coefficients in GMAS, and the embedding algorithm is the single-layered STC. The  $\text{phase}_{\text{max}}$  is set as 4,  $h$  is set as 3, and  $\alpha$  is set as 3. We can observe that the error rate of

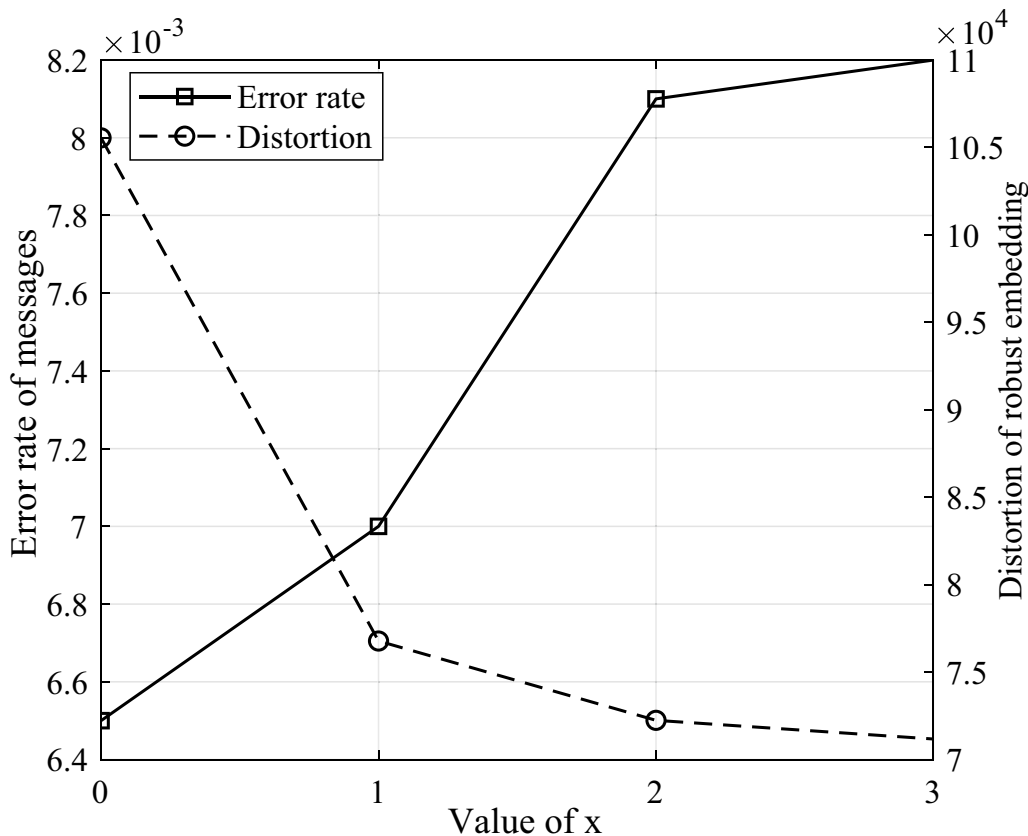


Fig. 5 The error rate of messages and distortion after robustness embedding when different values of  $x$  are used



messages is lowest when  $x = 0$ , where only robustness is considered. But the distortion is also the highest. When  $x = 1$ , the distortion value is reduced dramatically, and the error rate increases slightly. When  $x = 2$ , the distortion value reduces slowly, and the error rate increases significantly. Combining the above result, we set  $x = 1$ .

The algorithm of robustness embedding can be seen in Algorithm 1. To improve the effect of robustness embedding, we iterate the process for  $r$  times. The error rate of protocol messages and the total distortion with the condition of times  $r$  is shown in Fig. 6. From it, we can observe that the number of errors in the embedding domain of GMAS is decreased with the performance of robustness embedding. More times of robust embedding leads to more robust performance. So multiple times of robustness embedding can be used to increase the recovery possibility of protocol messages.

be robust enough. In GMAS, the embedding domain are the 21 middle-frequency coefficients. Work [24] shows that the higher the embedding ratio is, the better the robust performance. So the part of DCT blocks in DCT block group2 are selected to embed the protocol message. The selection satisfies the ratio between the length of protocol messages and the number of DCT coefficients in the embedding domain is 0.3. There comes the question of which blocks should be selected. There are two options we can choose from. One of them is that we randomly select part of the DCT blocks in DCT block group2 as the embedding domain. In the second method, for the blocks with higher frequency content are more secure to perform steganography, we want these blocks in group2 are selected. But the problem is how to make sure that the receiver can correctly recover the embedding

---

**Input:** DCT block  $D_i$ ; max phase  $\text{phase}_{\max}$ ; max change value  $h$ ; number of changed coefficients in a DCT block  $\alpha$ ; JUNIWARD distortion  $\rho_i$ ;

**Output:** DCT block  $O_i$  after robustness embedding;

- 1: Initialize  $O_i = D_i$ ;
  - 2: Suppose  $(p_x, p_y) \in \{(1, 1), (1, 2), \dots, (8, 8)\}$  is the phase index of DCT block. Then the phase set  $\mathbf{p}_{set} = \{(p_a, p_b), p_a + p_b \leq \text{phase}_{\max}\}$  are the domain needed to perform modifications. For every  $(p_a, p_b) \in \mathbf{p}_{set}$  and change value  $c \in \{\pm 1, \pm 2, \dots, \pm h\}$ , generate changed DCT block  $D_{i,p_a,p_b,c}$ , whose value
 
$$D_{i,p_a,p_b,c}(m, n) = \begin{cases} D_i(m, n) + c & \text{if } m = p_a, n = p_b; \\ D_i(m, n) & \text{otherwise.} \end{cases}$$
  - 3: Input  $D_{i,p_a,p_b,c}$  into the robustness classifier, where  $(p_a, p_b) \in \mathbf{p}_{set}$ ,  $c \in \{\pm 1, \pm 2, \dots, \pm h\}$ , which generates  $R_{i,p_a,p_b,c}$ . Generate  $G_{ic}$ , its value  $G_{ic}(p_a, p_b) = R_{i,p_a,p_b,c}$ ;
  - 4: Input  $D_i$  into the classifier, which outputs the result  $R_i$ , and generate  $G_i(p_a, p_b) = R_i$ , where  $(p_a, p_b) \in \mathbf{p}_{set}$ ;
  - 5: Generate  $\rho_{ic}$ , its value is  $\rho_{ic}(p_a, p_b) = \rho_i(p_a, p_b) * |c|$ , where  $(p_a, p_b) \in \mathbf{p}_{set}$ ,  $c \in \{\pm 1, \pm 2, \dots, \pm h\}$ . Then generate  $F_{ic} = \frac{G_i - G_{ic}}{\rho_{ic}}$ , and let  $F_{ic}(F_{ic} < 0) = 0$ ,  $F_{ic}(G_i < 0.5) = 0$ ;
  - 6: Find  $\alpha$  number of max value from  $F_{ic}$ , where  $c \in \{\pm 1, \pm 2, \dots, \pm h\}$ . Suppose  $F_{ic}(p_a, p_b)$  is found and  $F_{ic}(p_a, p_b) \neq 0$ , and then perform  $O_{p_a,p_b} = D_i(p_a, p_b) + c$ ;
  - 7: **return**  $O_i$ ;
- 

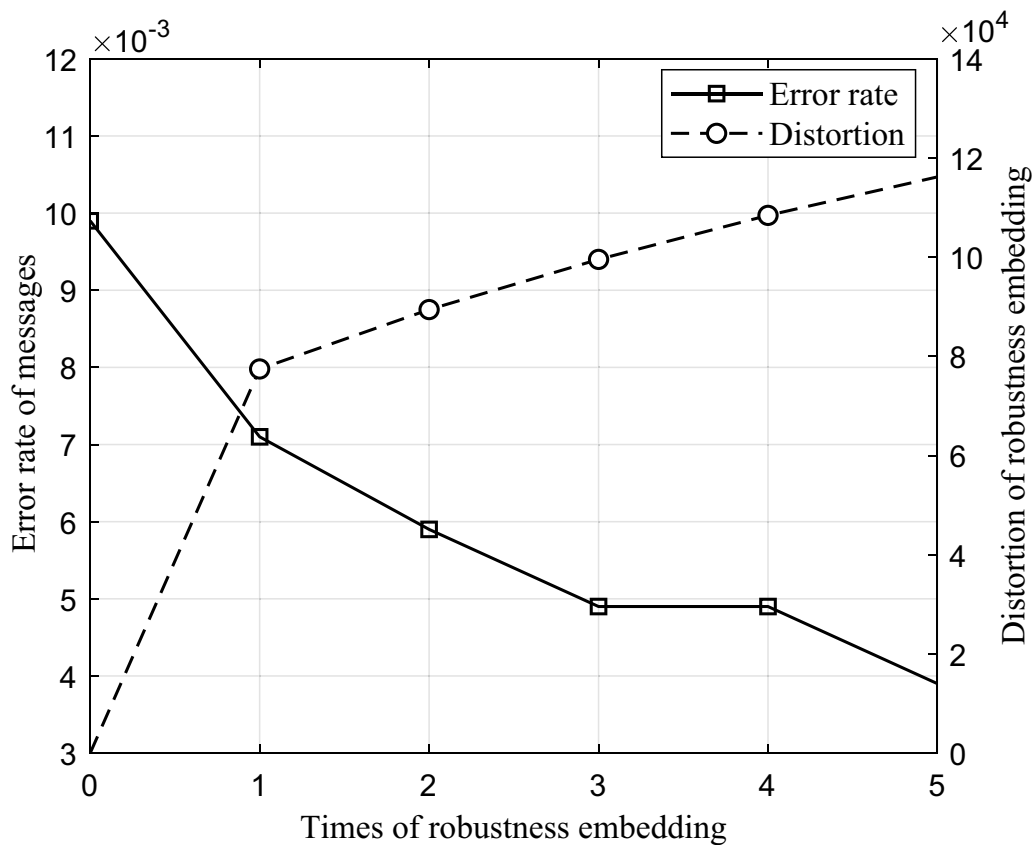
**Algorithm 1** Procedure of robustness embedding

### 3.4 The embedding process of protocol messages

In Section 3.1 the architecture of the proposed method has been described, and the embedding domain of secret message embedding is introduced in Section 3.2. The details of protocol message embedding are explained in this part. To restore the protocol message successfully in DCT block group2, we need the embedding algorithm to

domain. Based on the above two methods, we propose three methods to perform protocol message embedding.

To describe the motivation of the proposed methods, suppose the value of the DCT coefficient in the embedding domain of GMAS is  $\nu$ , and the change step of  $\nu$  after recompression with mozjpeg is  $c$ . Counted from testing images, the distribution of  $c$  in the condition of  $\nu$  is



**Fig. 6** The error rate of messages and distortion with different times of robustness embedding

shown in Table 3. We can see the most non-zero change steps after mozjpeg recompression are +1 and -1. And the changes are more easily happened in coefficients with  $|\nu| = 1$  and  $|\nu| = 2$ .

The abstract values of most DCT coefficients are 0 and 1, and DCT coefficients with the abstract value 1 are more easily changed after recompression. Thus, we use the number of DCT coefficients with the abstract value above 1 in the embedding domain of GMAS, called NAO

**Table 3** The conditional possibility (%) of change step  $c$  based on the quantized DCT coefficient value  $\nu$

$\nu$	$c$				
	-2	-1	0	1	2
-3	0	0.01	99.97	0.01	0
-2	0	0.02	99.75	0.21	0.018
-1	0	0.005	99.59	0.41	0
0	0	0	100	0	0
1	0	0.41	99.59	0.005	0
2	0.025	0.21	99.74	0.02	0
3	0	0.01	99.97	0.01	0

in the paper, as the condition to construct the embedding domain. The higher the NAO is, the more complex the DCT blocks are. Embedding the protocol message in DCT blocks with higher NAO can ensure security performance. Meanwhile, to increase the ability to restore the embedding domain, we need to increase the distance between NAO in the embedding domain and that in the non-embedding domain.

The first proposed protocol message embedding method is performed as follows. To form the embedding domain, we count NAO in every DCT block in group2, which forms an array of NAO  $n_a$ . Then, we calculate the histogram of  $n_a$  as  $n_h$ , and we use  $n_h(i)$  as the number of DCT blocks in  $n_h$  when NAO is  $i$ . Then supposing we need  $l$  blocks to embed the protocol message, we can find the max value  $n$  that satisfies

$$\sum_{i=n}^{21} n_h(i) \geq l. \tag{7}$$

After calculating  $n$ , we traverse every DCT block in DCT block group2. The blocks whose  $NAO \geq n$  are

selected as embedding blocks until we select  $l$  blocks. The last index when we choose the embedding block is  $index_{last}$ . To increase the possibility of restoring the embedding domain, we modified NAO of DCT blocks whose value is  $n$  and  $n + 1$  as  $n + 2$ . And we also set the DCT blocks before  $index_{last}$  whose NAO value is  $n - 1$  and  $n - 2$  as  $n - 3$ . After that, to reduce the possibility of failure to restore the embedding domain and decrease the error rate of protocol messages, two kinds of DCT blocks need to perform robustness embedding: the selected embedding domain for protocol messages, and the DCT blocks satisfying the NAO value equals  $n - 3$  and the possibility of non-robustness is higher than 0.5 before modification. After above operations, the smallest distance of NAO between DCT blocks in the embedding domain and the non-embedding domain becomes 4, and the maximal tolerable change of NAO becomes 2 in the same direction. In practical usage, the cyclic redundancy check (CRC) [36] code of the protocol message before error-correcting code encoding is generated. The generated CRC code is added to the end of the protocol message. The value of  $n$  does not need to be known on the receiver side. The receiver can try the value from large to small until the calculated CRC result of the extracted protocol message without the stored CRC value is equal to the stored value. After performing the robustness embedding, we embed the protocol message. To ensure that the receiver can successfully restore the embedding domain, when performing STC embedding with protocol messages, we chose the changing direction which does not generate more DCT coefficients with the abstract value as 1. It is achieved by limiting the modification direction when the abstract value of the coefficient before embedding is 2. We call the method as *methodI*.

To compare the performance of *methodI*, we proposed a simple method. We first randomly select  $l$  blocks from DCT block group2 as the embedding domain of protocol messages. Then, we perform robustness embedding and embed the protocol messages. We call this method as *methodII*. The case where the robustness embedding is not performed is named *methodIII*.

The third strategy is first randomly selecting  $l$  blocks from DCT block group2 as the embedding domain. Then, we also calculate  $n$  as (7), but we only perform robustness embedding with DCT blocks in the protocol message embedding domain whose  $NAO \geq n$ . The method is called *methodIII*. The reason that the *methodIII* also has good robustness can be seen in Tables 4 and 5. They show the error rate in DCT blocks with different NAO before and after three times of robustness embedding. We can observe from Table 4 that as NAO increases, the error rate becomes higher. After three times of robustness embedding, the error rate in the DCT blocks whose NAO is high decreases, which can be seen in Table 5. So compared with *methodI*, the method performs robustness embedding to the DCT blocks with the same NAO and replaces some DCT blocks in *methodI* with DCT blocks that have smaller NAO. Although the robustness embedding is not performed in these blocks, because their robustness is better, the overall robustness can be improved. Because the robustness embedding happens only in the complex blocks, the distortion caused by robustness embedding is less than *methodIII*.

#### 4 Experimental results and analyses

In this section, the experiments are implemented to test the performance of the proposed robust steganography, and comparisons are made with other robust steganography. In the first part, the settings of experiments are explained. In the second part, the influence of the settings of parameters in the proposed methods is tested. In the third part, the robustness performance is tested, and the comparison is made with GMAS method. In the fourth part, the security performance is tested and compared with GMAS. In the fifth part, the comparison of robustness with other state-of-the-art robust steganography is presented. At last, the robustness performance is tested on the Matlab recompression channel.

##### 4.1 Experimental settings

All the testing images come from BOSSbase1.01. The testing images are compressed to JPEG images with the *imread* and *imwrite* functions in Matlab with quality

**Table 4** The error rate in the DCT block with different NAO before robustness embedding

NAO	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Error rate ( $\times 10^{-2}$ )	0.17	1.19	2.18	2.65	2.68	2.98	3.19	3.64	4.09	4.20	4.72	5.00	5.71	6.35	6.68	6.68

**Table 5** The error rate in the DCT block with different NAO after three times of robustness embedding

NAO	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Error rate ( $\times 10^{-2}$ )	0.15	0.62	1.26	1.45	1.51	1.63	1.76	2.23	1.53	1.62	1.41	1.76	2.00	3.01	2.23	3.95

factors 75, 85, and 90 and named as BOSSbase75, BOSSbase85, and BOSSbase90 separately. To test the robust performance, we use the mozjpeg to recompress images generated from Matlab. When we embed protocol messages, the single-layered STC embedding is used, and the embedding domain is the same as GMAS. To ensure that the protocol messages can be successfully extracted, we calculate the value of payloads with the length of protocol messages after RS [37] encoding and the number of DCT coefficients in the embedding domain. The payload is set as 0.3. When we perform error-correcting code encoding on protocol messages, the parameter of RS codes is (200, 100) with 8-bit symbols. The submatrix height of binary STC for protocol message embedding is 3. When we embed secret messages, the ternary STC with the submatrix height of 7 is used. The parameter of the BCH [38] encoder for secret message encoding is (127, 64). The BCH encoded message is permuted before embedding to avoid the error diffusion problem of STC extraction.

In methodI, because the NAO in every DCT block of the protocol message embedding domain is subtracted by 3, the least NAO in images before embedding needs to be 3. So we first select the images satisfying the condition in the BOSSbase75, BOSSbase85, and BOSSbase90. The result is that 872, 2681, and 4422 images in BOSSbase75, BOSSbase85, and BOSSbase90 separately satisfy the condition. In our experiments, we select 100 images from quantified images in BOSSbase85 to test the influence of parameters. Then 872 images in qualified images in BOSSbase75, 1000 images randomly selected from quantified images in BOSSbase85, and 1000 images randomly selected from quantified images in BOSSbase90 are used to test the robustness and security performance.

### 4.2 The influence of the parameter settings

In this part, we aim to find the appropriate values for parameters  $phase_{max}$ ,  $\alpha$ , and  $h$ . The dataset consists of 100 JPEG images randomly selected from BOSSbase85 which satisfy the condition that the smallest NAO is at least 3. The embedding method is proposed methodIII. We use different settings of  $phase_{max}$ ,  $\alpha$ , and  $h$  to perform three times of robustness embedding. The results in terms of the error rate of extracted messages  $P_e$  and total distortion are shown in Tables 6, 7, and 8. From the

**Table 6**  $P_e$  and distortion between cover images and cover images after robustness embedding in methodIII with different  $\alpha$  when  $h = 3$  and  $phase_{max} = 4$

$\alpha$	1	2	3	4	5
$P_e$	0.0073	0.0073	0.0068	0.0068	0.0067
$Distortion(\times 10^5)$	1.1227	1.2595	1.6184	1.9863	2.4008

**Table 7**  $P_e$  and distortion between cover images and cover images after robustness embedding of methodIII with different  $h$  when  $\alpha = 3$  and  $phase_{max} = 4$

$h$	1	2	3	4	5
$P_e$	0.0077	0.0075	0.0068	0.0065	0.0063
$Distortion(\times 10^5)$	1.3750	1.3572	1.6184	1.9298	2.2211

**Table 8**  $P_e$  and distortion between cover images and cover images after robustness embedding of methodIII with different  $phase_{max}$  when  $h = 3$  and  $\alpha = 3$

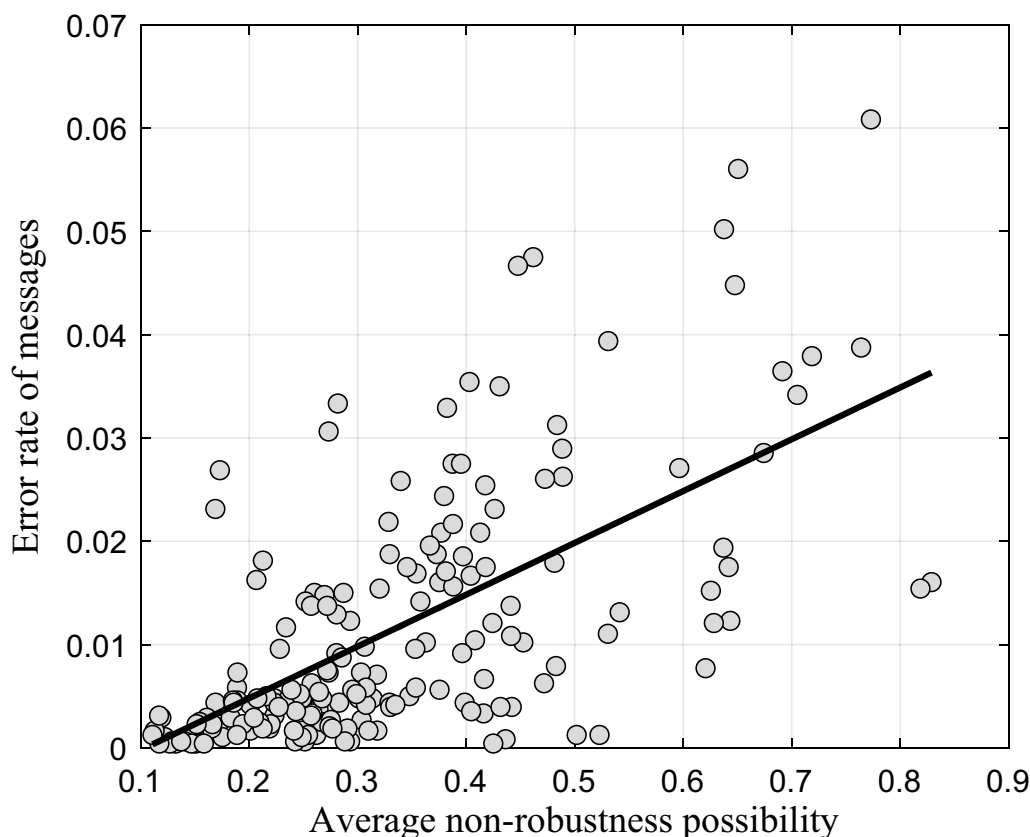
$phase_{max}$	2	3	4	5	6
$P_e$	0.0073	0.0068	0.0068	0.0068	0.0068
$Distortion(\times 10^5)$	1.4080	1.5202	1.6184	1.6888	1.7449

results, the number of changes is more important than the max number of phases, which can be seen that the bigger  $\alpha$  and  $h$  result in a lower error rate and more distortion. The parameter  $phase_{max}$  has less influence on the robustness, which can be seen in Table 8. To ensure robust and security performance, we set  $\alpha = 3$ ,  $h = 3$ , and  $phase_{max} = 4$  in our following experiments.

We then test the relation between the average non-robust possibility in the embedding domain of protocol messages and the error rate of extracted messages. Figure 7 shows the relationship between the error rate of messages and the average non-robust possibility in the embedding domain of cover images and the image after one time of robustness embedding. The line in Fig. 7 is the best fit of the polynomial with degree 1 in a least-square method. From the result, we can observe that the error rate of messages and the average non-robust possibility are clearly positively correlated. More specifically, when the average of non-robust possibility is in the area of [0.15, 0.25], the max value of the error rate is 0.026. When the average of non-robust possibility is in the area of [0.25, 0.35], the max value of the error rate is 0.031. To reduce the error rate of messages and the number of modifications when performing robustness embedding, we test the average of the non-robustness possibility in the embedding domain before starting robustness embedding. If the value is less than 0.25, we stop robustness embedding; otherwise, we continue robustness embedding, and the iteration begins. The max times of robustness embedding is 5 in the experiments. Because the robustness embedding is necessary for methodI to restore its embedding domain, the number of times of robustness embedding is at least three in methodI.

### 4.3 Robustness performance

As previously mentioned, the datasets used for training and testing the robustness classifier are images randomly



**Fig. 7** The scatter map of the average non-robust possibility versus the error rate of messages

picked from ImageNet. We test the robustness of the proposed robust steganography with BOSSbase. The robustness against recompression with mozjpeg with quality factors as 75, 85, and 90 are tested in this part. To test the performance of different quality factors, we first train the classifier in the quality factor as 85, which generates the classifier  $C_{85}$ . Then we fine-tune the classifier  $C_{85}$  with the images with quality factors as 75 and 90 generated from ImageNet with another 20 epochs respectively, which generates the classifier  $C_{75}$  and the classifier  $C_{90}$ . We set the error rate of STC extracted messages before error-correcting code decoding in stego images as  $p_e$ . The Success rate refers to the success rate of correctly restoring the secret messages. In the proposed methods, there is the error rate in protocol messages and secret messages; there is only the error rate in secret messages in GAMS. The experimental results of the robust performance of proposed methods with the classifier  $C_{85}$ ,  $C_{75}$ , and  $C_{90}$  testing on BOSSbase images with quality factors as 85, 75, and 90 are shown in Tables 10, 9, and 11, respectively. The *avg* means the average of results, and *var* means the variance of results.

From the results, we can observe that the robustness performance of proposed methods is better compared

with GMAS, which can be seen from the higher success rate. When the quality factors are 75 and 85, the error rate in the secret message embedding domain, which are coefficients in DCT blocks selected by the classifier, is lower than the embedding domain in middle-frequency DCT coefficients. It verifies that DCT blocks selected by the classifier are suitable for embedding secret messages. The reason is that a more efficient error-correcting code can be used, and the payload can be large. When the quality factor is 90, the error rate in DCT blocks selected by the classifier is comparable with the embedding domain of GMAS. It is because the error rate in middle-frequency DCT coefficients is less when the quality factor is 90, while the error rate in DCT blocks selected by the classifier remains almost the same in different quality factors. When we embed the protocol messages, the embedding domain is the same as the GMAS, but its error rate is lower. It is because compared with double-layered STC under the same error rate in stego images and the same message length, single-layered STC embedding has fewer errors when messages are extracted. And we calculate the value of embedding payloads using the number of all DCT coefficients in the embedding domain, so the value of payloads is usually higher than that calculated

**Table 9** Robustness performance of proposed methods and GMAS with the quality factor as 75

Algorithm	Payload	Protocol messages $avg(P_e) \times 10^{-2}$	Protocol messages $var(P_e) \times 10^{-4}$	Secret messages $avg(P_e) \times 10^{-2}$	Secret messages $var(P_e) \times 10^{-4}$	Success rate (%)
GMAS	0.05	-	-	19.21	2360.00	23.60
	0.1	-	-	12.82	12.80	28.60
	0.2	-	-	7.78	66.00	35.60
	0.3	-	-	5.93	45.00	39.30
methodI	0.05	1.35	4.55	0.30	0.13	96.79
	0.1	1.35	4.50	0.18	0.05	96.67
	0.2	1.35	4.54	0.21	0.12	96.56
	0.3	1.33	4.33	0.45	0.73	94.15
methodII0	0.05	1.73	6.44	<b>0.29</b>	0.13	95.53
	0.1	1.73	6.46	<b>0.17</b>	0.04	95.41
	0.2	1.73	6.44	<b>0.20</b>	0.12	95.53
	0.3	1.73	6.44	<b>0.43</b>	0.67	93.23
methodII	0.05	<b>1.19</b>	1.19	0.30	0.13	<b>99.08</b>
	0.1	<b>1.19</b>	1.96	0.18	0.06	<b>99.08</b>
	0.2	<b>1.20</b>	1.97	0.24	0.44	<b>98.85</b>
	0.3	<b>1.19</b>	1.96	0.52	0.73	<b>95.99</b>
methodIII	0.05	1.36	3.31	0.30	0.16	98.05
	0.1	1.36	3.33	0.18	0.05	98.05
	0.2	1.36	3.32	0.22	0.15	98.05
	0.3	1.36	3.31	0.47	0.09	95.41

Note: Boldface highlights the best performance in a column with the same payload

**Table 10** Robustness performance of proposed methods and GMAS with the quality factor as 85

Algorithm	Payload	Protocol messages $avg(P_e) \times 10^{-2}$	Protocol messages $var(P_e) \times 10^{-4}$	Secret messages $avg(P_e) \times 10^{-2}$	Secret messages $var(P_e) \times 10^{-4}$	Success rate (%)
GMAS	0.05	-	-	11.75	138.00	31.10
	0.1	-	-	7.35	73.00	38.20
	0.2	-	-	4.28	32.00	47.30
	0.3	-	-	3.22	20.00	53.50
methodI	0.05	0.81	3.16	<b>0.37</b>	0.21	98.20
	0.1	0.81	3.16	<b>0.20</b>	0.06	98.30
	0.2	0.81	3.16	<b>0.15</b>	0.05	98.20
	0.3	0.81	3.13	<b>0.25</b>	0.24	97.90
methodII0	0.05	0.99	2.66	<b>0.37</b>	0.22	98.70
	0.1	0.99	2.67	0.37	0.06	98.70
	0.2	0.99	2.66	<b>0.15</b>	0.05	98.60
	0.3	0.99	2.65	<b>0.25</b>	0.22	98.30
methodII	0.05	<b>0.72</b>	1.19	<b>0.37</b>	0.22	<b>99.60</b>
	0.1	<b>0.72</b>	1.18	0.21	0.06	<b>99.60</b>
	0.2	<b>0.71</b>	1.18	0.16	0.05	<b>99.60</b>
	0.3	<b>0.71</b>	1.18	0.26	0.27	<b>98.90</b>
methodIII	0.05	0.79	1.51	<b>0.37</b>	0.22	99.40
	0.1	0.79	1.51	0.21	0.06	99.40
	0.2	0.79	1.51	<b>0.15</b>	0.05	99.30
	0.3	0.79	1.50	0.26	0.28	98.60

Note: Boldface highlights the best performance in a column with the same payload

**Table 11** Robustness performance of proposed methods and GMAS with the quality factor as 90

Algorithm	Payload	Protocol messages $avg(P_e) \times 10^{-2}$	Protocol messages $var(P_e) \times 10^{-4}$	Secret messages $avg(P_e) \times 10^{-2}$	Secret messages $var(P_e) \times 10^{-4}$	Success rate (%)
GMAS	0.05	-	-	8.20	92.00	44.10
	0.1	-	-	4.92	42.00	51.50
	0.2	-	-	2.85	17.00	60.70
	0.3	-	-	2.12	10.00	63.70
methodI	0.05	0.44	1.53	<b>0.94</b>	0.25	98.10
	0.1	0.44	1.53	<b>0.51</b>	0.08	98.30
	0.2	0.44	1.54	<b>0.31</b>	0.10	98.10
	0.3	0.44	1.54	<b>0.32</b>	0.25	97.60
methodII0	0.05	0.69	1.60	<b>0.94</b>	0.25	99.00
	0.1	0.70	1.61	<b>0.51</b>	0.08	99.20
	0.2	0.69	1.61	<b>0.31</b>	0.09	99.00
	0.3	0.69	1.60	<b>0.32</b>	0.25	98.60
methodII	0.05	<b>0.30</b>	0.30	0.95	0.26	<b>99.70</b>
	0.1	<b>0.30</b>	0.29	0.52	0.08	<b>99.90</b>
	0.2	<b>0.30</b>	0.30	0.32	0.11	<b>99.70</b>
	0.3	<b>0.30</b>	0.30	0.33	0.26	<b>99.20</b>
methodIII	0.05	0.44	0.73	0.95	0.26	99.60
	0.1	0.44	0.74	0.52	0.08	99.80
	0.2	0.44	0.74	0.32	0.10	99.60
	0.3	0.44	0.74	<b>0.32</b>	0.20	99.10

Note: Boldface highlights the best performance in a column with the same payload

using non-zeros AC DCT coefficients. The error rate of protocol messages in methodII0 is higher compared with other proposed methods, showing that the robustness embedding can improve the robustness. The success rate of methodI is sometimes lower than methodII0. The reason is that the embedding domain of methodI may be corrupted if the change of NAO in DCT blocks is above 2, so that the protocol messages could not be extracted. Meanwhile, other methods do not need NAO to reconstruct the embedding domain in the receiver side.

#### 4.4 Security performance

Before testing the security performance of the proposed methods against the steganalysis, we first calculate the peak signal-to-noise ratio (PSNR) value of proposed methods and GMAS. The aim is to compare the influence of embedding to the image content. The result can be seen in Table 12. From it, we can first observe that methodI has the smallest PSNR value while methodII0 has the largest PSNR value. This is because there are more modifications to form the embedding domain in methodI, and there is no robustness embedding in methodII0. Secondly, the PSNR value of GMAS drops dramatically compared with proposed methods. It is

**Table 12** The mean PSNR (dB) value of testing stego images embedded with proposed methods and GMAS with different payloads

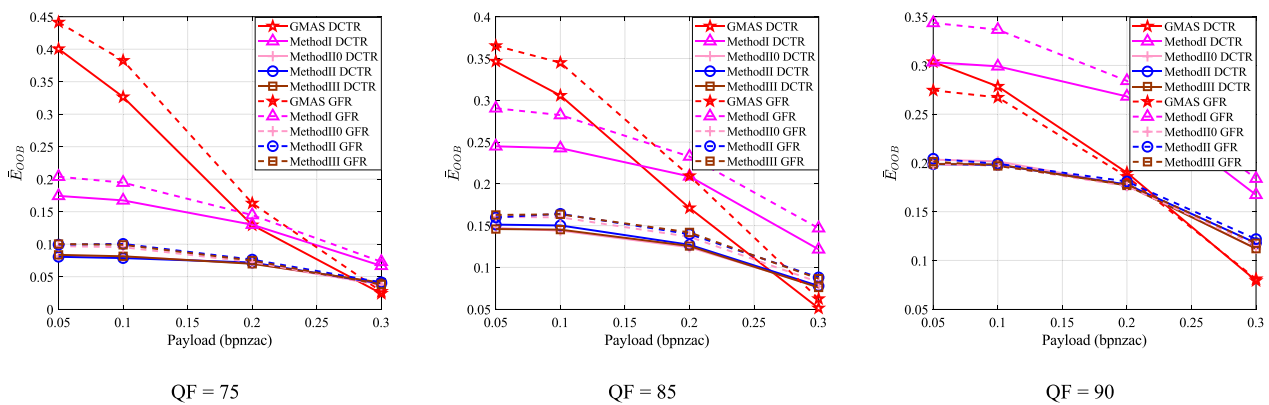
QF	Algorithm	Payload			
		0.05	0.1	0.2	0.3
75	GMAS	<b>49.77</b>	<b>45.94</b>	41.82	39.28
	methodI	38.29	38.17	37.79	37.16
	methodII0	43.57	43.19	<b>42.49</b>	<b>42.45</b>
	methodII	42.66	42.34	41.43	40.11
	methodIII	42.77	42.44	41.52	40.17
85	GMAS	<b>53.21</b>	<b>49.45</b>	45.44	42.97
	methodI	43.27	43.10	42.59	41.73
	methodII0	47.74	47.29	<b>46.06</b>	<b>44.37</b>
	methodII	47.09	46.71	45.63	44.09
	methodIII	47.02	46.64	45.58	44.06
90	GMAS	<b>56.13</b>	<b>52.46</b>	48.52	46.09
	methodI	46.97	46.76	46.15	45.17
	methodII0	51.12	50.60	<b>49.24</b>	<b>47.47</b>
	methodII	49.67	49.29	48.26	46.80
	methodIII	49.95	49.56	48.47	46.95

Note: Boldface highlights the best performance in a column with the same quality factor

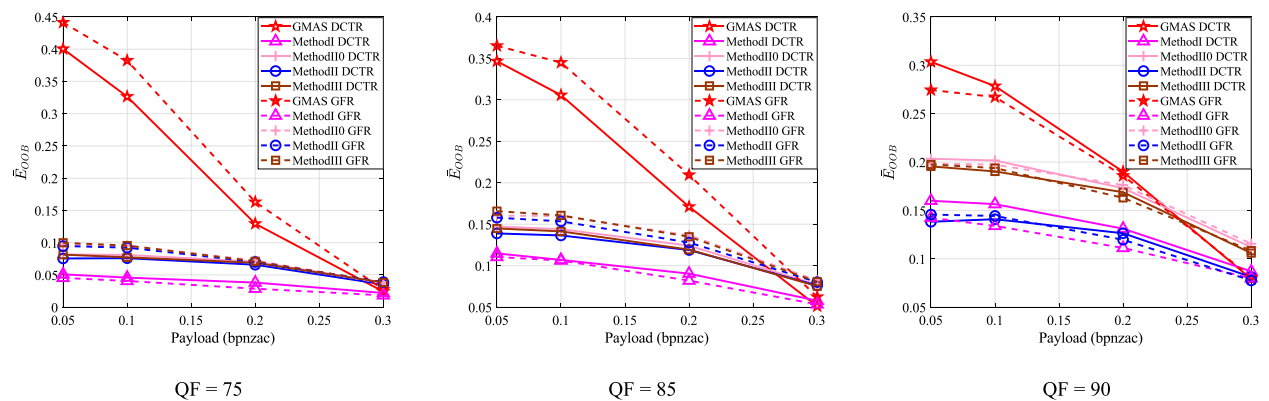
because although the embedding domain of GMAS and protocol message embedding are the middle-frequency DCT coefficients, the protocol messages of proposed methods keep the same length while secret messages of different sizes are embedded. Thus, the decreasing PSNR is only related to the secret message embedding in proposed methods. When embedding secret messages, the proposed methods embed in all DCT coefficients in DCT blocks selected by classifiers. The total distortion after embedding is smaller compared with that of embedding in the middle-frequency domain. When the payload is 0.3 bpnzac, the PSNR value of proposed methodIII is higher than that of GMAS with different quality factors. It shows that when large payloads are embedded, the influence of embedding to image content of methodIII is less compared with that of GMAS. We can also observe that when the quality factor is 90, the difference between methodII and methodIII is obviously larger. It indicates that the difference in security

performance between methodII and methodIII when the quality factor as 90 should be larger.

To test the security performance, we experiment on two situations. In the first situation, the cover images are the images before steganographic embedding that includes the embedding of protocol messages and secret messages, which are the images after robustness embedding. The result is shown in Fig. 8. In the second situation, the cover images are the images before robustness embedding. The result is shown in Fig. 9. Note that the cover images before robustness embedding in methodI refer to images without any modifications, which means the modifications to form the embedding domain are not included. The steganalysis algorithms used to test the security performance are DCTR [3] and GFR [4]. The images used for training and testing are images used for testing the robust performance. The  $\bar{E}_{OOB}$  is the average detection error rate for ten times of testing. In each test, the dataset is randomly split into the equal-sized training set and testing set.



**Fig. 8** The detection error rate using steganalysis algorithm DCTR and GFR with different quality factors when the cover images are images after robustness embedding



**Fig. 9** The detection error rate using steganalysis algorithm DCTR and GFR with different quality factors when the cover images are images before robustness embedding



From the results, we can observe that when the cover images are images after robustness embedding, the security performance of methodI is better compared with methodII0, methodII, and methodIII. It is because the protocol messages are embedded in the complex contents. When the cover images are images before robustness embedding, the security performance of methodI is lower compared with methodII0, methodII, and methodIII. The reason is that the modifications of DCT coefficients to construct the embedding domain in methodI introduce more distortion. The security performance of GMAS is better compared with the proposed methods when payloads as 0.05 bpnzac, 0.1 bpnzac, and 0.2 bpnzac, for the value of payloads is high when we embed the protocol messages in proposed methods. The security performance of proposed methods should be between the security performance of GMAS when the payload is 0.2 bpnzac and 0.3 bpnzac. It can be seen from the security performance of the proposed methods when the embedding payload as 0.05 bpnzac from Fig. 9. The security performance of GMAS is lower compared with proposed methodIII when the payload is 0.3 bpnzac. It means that the anti-detection ability of the combination of robustness embedding, protocol message embedding, and adaptive embedding with a larger payload is better compared with embedding the same payload using GMAS. The reason can also be found in Table 12, which shows the PSNR decreases faster compared with the proposed protocol message-based embedding methods. The security performance of methodIII is better compared with methodII in Fig. 9 in most cases, because the modifications of robustness embedding are performed in complex areas.

We further explore the reason for the low-security performance in methodI by differentiating cover images after robustness embedding from the cover images before robustness embedding. The result is shown in Table 13. It demonstrates the detection error rate of the detector with DCTR features. From Table 13, we can observe when the cover images are images before robustness embedding, the low-security performance in methodI is more related to the robustness embedding process. We can also observe that as the quality factor increases, the difference in security performance between methodII

**Table 13** The detection error rate (%) on images after robustness embedding and images before robustness embedding using steganalysis algorithm DCTR

QF	methodI	methodII	methodIII
75	8.13	44.27	<b>45.73</b>
85	15.80	41.88	<b>45.93</b>
90	20.39	23.91	<b>42.43</b>

Note: Boldface highlights the best performance in a row

and methodIII increases. It explains the lower security performance of methodII in Fig. 9 when the quality factor as 90.

#### 4.5 Robustness comparison with other methods

In this part, we compare the robustness performance with other state-of-the-art robust steganography, including JCRISBE [27] method and MINICER method [29]. Because in our implementation scenario, the embedder does not need to upload the cover image to the channel. So when performing JCRIBSE method and MINICER method, the Matlab's *imread* and *imwrite* functions are used to perform TCM process and generate channel processed cove images. The compression channel of stego image is mozjpeg. The quality factor is 85. The images are from the dataset testing the robustness performance in Section 4.3. The same BCH parameters in Section 4.3 are used in JCRIBSE and MINICER. The result of success rate in different payloads are shown in Table 14.

From the result, we can clearly see the advantage of the proposed method. In the method, we train a classifier that learns the robustness of the DCT block after recompression, and we use the classifier to improve the robustness without uploading the cover image to the compression channel. The JCRIBSE and the MINICER methods rely on acquiring the recompression process. The robust performance is unacceptable if the recompression cannot be acquired. Compared with state-of-the-art robust steganography, the proposed method has the benefit of satisfying robustness performance and can be implemented without knowing the exact result of channel compression. The method also removes the concern of behavior security on the sender's side.

#### 4.6 The generality of proposed methods on the Matlab recompression channel

In this part, to test the proposed method's generality, we examine the robustness performance of proposed methods in Matlab compression channel. The compression channel consists of *imread* and *imwrite* functions in Matlab. Firstly, the 10,000 images from ImageNet used for training and testing are recompressed with Matlab compression with

**Table 14** The success rate (%) of JCRISBE, MINICER, and proposed methodIII with different payloads. The quality factor is 85

Algorithm	Payload (bpnzac)			
	0.05	0.1	0.2	0.3
JCRISBE	0.20	0.80	4.60	9.00
MINICER	0.20	0.90	4.50	9.40
methodIII	<b>99.40</b>	<b>99.40</b>	<b>99.30</b>	<b>98.60</b>

Note: Boldface highlights the best performance in a column

**Table 15** Robustness performance of proposed methods and GMAS with the quality factor as 85 on the Matlab recompression channel

Algorithm	Payload	Protocol messages $avg(P_e) \times 10^{-2}$	Protocol messages $var(P_e) \times 10^{-4}$	Secret messages $avg(P_e) \times 10^{-2}$	Secret messages $var(P_e) \times 10^{-4}$	Success rate (%)
GMAS	0.05	-	-	<b>0.23</b>	0.25	98.00
	0.1	-	-	0.20	0.21	99.00
	0.2	-	-	0.18	0.17	98.00
	0.3	-	-	0.19	0.19	97.00
methodI	0.05	<b>0.02</b>	0.00	0.27	0.32	99.00
	0.1	<b>0.02</b>	0.00	<b>0.16</b>	0.11	99.00
	0.2	<b>0.02</b>	0.00	<b>0.14</b>	0.09	99.00
	0.3	<b>0.02</b>	0.00	<b>0.16</b>	0.12	99.00
methodII	0.05	0.18	0.18	0.27	0.31	<b>100.00</b>
	0.1	0.18	0.18	0.21	0.11	<b>100.00</b>
	0.2	0.18	0.18	<b>0.14</b>	0.09	<b>100.00</b>
	0.3	0.18	0.18	0.17	0.12	<b>100.00</b>
methodIII	0.05	0.12	0.07	0.27	0.31	<b>100.00</b>
	0.1	0.12	0.07	<b>0.16</b>	0.11	<b>100.00</b>
	0.2	0.12	0.07	<b>0.14</b>	0.09	<b>100.00</b>
	0.3	0.12	0.07	0.17	0.12	<b>100.00</b>
methodIII	0.05	0.14	0.09	0.27	0.31	<b>100.00</b>
	0.1	0.14	0.09	<b>0.16</b>	0.11	<b>100.00</b>
	0.2	0.14	0.09	<b>0.14</b>	0.09	<b>100.00</b>
	0.3	0.14	0.09	0.17	0.12	<b>100.00</b>

Note: Boldface highlights the best performance in a column with the same payload

quality factor 85. Then, the classifier  $C_{85}$  is fine-tuned with another 20 epochs with those images. The cover images are 100 images randomly picked from images satisfying the condition that the smallest NAO is 3 in BOSSbase85. After the stego images are recompressed by Matlab, the error rate and success rate are shown in Table 15.

From the results, we can see that the proposed methods still have the better robustness performance compared with GMAS. The success rate of methodI is lower compared with other proposed methods. It is caused by the possibility of failing to restore the protocol messages embedding domain. And the lower average error rate of protocol messages in methodI comes from at least three times of robustness embedding in the protocol messages embedding domain. From the comparison in the average error rate of protocol messages, the robustness embedding can also decrease the error rate when images are compressed by Matlab recompression. The above results clearly show the generality of proposed methods.

### 5 Conclusion

In this paper, a robust steganography method based on the robust block selection and protocol messages is proposed. The deep learning-based classifier is trained to predict the possibility of non-robustness in a specific recompression channel and is used to select the robust

embedding domain. To decrease the error rate of protocol messages indicating the selection information, we propose a robustness enhancement method called robustness embedding. The experimental result demonstrates that the proposed methods have a higher successful communication rate. When protocol messages are embedded, robustness is improved by the robustness embedding. The security performance and the robustness performance of proposed methods are better compared with GMAS when the embedding payload is large. The method does not need to upload the cover image to the compression channel, which increases the behavior security. The experimental results also show that the proposed methods are universal, which can be used on different recompression channels. In further study, we will implement generative models to generate robust cover images with higher robustness and improve the security performance of the proposed robust steganography methods.

### Abbreviations

STC	Syndrome trellis code
GMAS	Generalized dither modulation-based robust steganography
TCM	Transport channel matching
BCH	Bose-Chaudhuri-Hocquenghem
NAO	Number of DCT coefficients with the abstract value above one
CRC	Cyclic redundancy check
RS	Reed-Solomon codes

**Acknowledgements**

Not applicable.

**Authors' contributions**

All authors read and approved the final manuscript.

**Funding**

This work was supported by NSFC under 61972390 and 62272456, and National Key Technology Research and Development Program under 2022QY0101 and 2020AAA0140000.

**Availability of data and materials**

BOSSbase [34], ImageNet [35].

**Code availability**

The code associated with the current submission is available at <https://github.com/jiminzhang2016/RobustClassifier>.

**Declarations****Competing interests**

The authors declare that they have no competing interests.

Received: 3 July 2023 Accepted: 20 November 2023

Published online: 11 December 2023

**References**

- J. Fridrich, J. Kodovsky, Rich models for steganalysis of digital images. *IEEE Trans. Inf. Forensics Secur.* **7**, 868–882 (2012)
- J. Kodovsky, J. Fridrich, Steganalysis of JPEG images using rich models. *Proc. SPIE Media Watermark. Secur. Forensics* **8303**, 81–93 (2012)
- V. Holub, J. Fridrich, Low-complexity features for JPEG steganalysis using undecimated DCT. *IEEE Trans. Inf. Forensics Secur.* **10**(2), 219–228 (2014)
- X. Song, F. Liu, C. Yang, X. Luo, Y. Zhang, in *Proc. ACM Workshop Inf. Hiding Multimedia Secur. (IH&MMSec)*. Steganalysis of adaptive JPEG steganography using 2D Gabor filters. (ACM, New York, 2015), pp. 15–23
- V. Holub, J. Fridrich, Phase-aware projection model for steganalysis of JPEG images. *Proc. SPIE* **9409**, 94090T (2015)
- M. Chen, V. Sedighi, M. Boroumand, J. Fridrich, in *Proc. 5th ACM Workshop Inf. Hiding Multimedia Secur. (IH&MMSec)*. JPEG-phase-aware convolutional neural network for steganalysis of JPEG images. (ACM, New York, 2017), pp. 75–84
- G. Xu, in *Proc. 5th ACM Workshop Inf. Hiding Multimedia Secur. (IH&MMSec)*. Deep convolutional neural network to detect J-UNIWARD. (ACM, New York, 2017), pp. 67–73
- J. Ye, J. Ni, Y. Yi, Deep learning hierarchical representations for image steganalysis. *IEEE Trans. Inf. Forensics Secur.* **12**(11), 2545–2557 (2017)
- M. Yedroudj, F. Comby, M. Chaumont, in *Proc. Int. Conf. Acoust. Speech Signal Process.* Yedroudj-Net: An efficient CNN for spatial steganalysis (IEEE, Calgary, 2018), pp. 2092–2096
- M. Boroumand, M. Chen, J. Fridrich, Deep residual network for steganalysis of digital images. *IEEE Trans. Inf. Forensics Secur.* **14**, 1181–1193 (2018)
- V. Holub, J. Fridrich, T. Denemark, Universal distortion function for steganography in an arbitrary domain. *EURASIP J. Inf. Secur.* **2014**, 1–13 (2014)
- V. Holub, J. Fridrich, in *Proc. 4th IEEE Int. Workshop Inf. Forensics Security*. Designing steganographic distortion using directional filters (IEEE, Tenerife, 2012), pp. 234–239
- B. Li, M. Wang, J. Huang, X. Li, in *Proc. IEEE Int. Conf. Image Process. (ICIP)*. A new cost function for spatial image steganography (IEEE, Paris, 2014), pp. 4206–4210
- L. Guo, J. Ni, Y.Q. Shi, Uniform embedding for efficient JPEG steganography. *IEEE Trans. Inf. Forensics Secur.* **9**, 814–825 (2014)
- L. Guo, J. Ni, W. Su, C. Tang, Y.Q. Shi, Using statistical image model for JPEG steganography: Uniform embedding revisited. *IEEE Trans. Inf. Forensics Secur.* **10**, 2669–2680 (2015)
- J. Fridrich, J. Kodovsky, in *Proc. IEEE ICASSP*. Multivariate gaussian model for designing additive distortion for steganography (IEEE, Vancouver, 2013), pp. 2949–2953
- V. Sedighi, J. Fridrich, R. Cogranne, Content-adaptive pentary steganography using the multivariate generalized Gaussian cover model. *Proc. SPIE* **9409**, 94090H (2015)
- V. Sedighi, R. Cogranne, J. Fridrich, Content-adaptive steganography by minimizing statistical detectability. *IEEE Trans. Inf. Forensics Secur.* **11**, 221–234 (2015)
- T. Filler, J. Judas, J. Fridrich, Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Trans. Inf. Forensics Secur.* **6**, 920–935 (2011)
- Y. Zhang, X. Luo, C. Yang, D. Ye, F. Liu, in *Proc. IEEE 10th Int. Conf. Availability Rel. Secur.* A JPEG-compression resistant adaptive steganography based on relative relationship between DCT coefficients (IEEE, Toulouse, 2015), pp. 461–466
- Y. Zhang, X. Luo, C. Yang, F. Liu, Joint JPEG compression and detection resistant performance enhancement for adaptive steganography using feature regions selection. *Multimedia Tools Appl.* **76**, 3649–3668 (2017)
- Y. Zhang, X. Zhu, C. Qin, C. Yang, X. Luo, Dither modulation based adaptive steganography resisting JPEG compression and statistic detection. *Multimedia Tools Appl.* **77**, 17913–17935 (2017)
- X. Yu, K. Chen, Y. Wang, W. Li, W. Zhang, N. Yu, Robust adaptive steganography based on generalized dither modulation and expanded embedding domain. *Signal Process.* **168**, 107343 (2020)
- C. Kin-Cleaves, A.D. Ker, in *Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS)*. Adaptive steganography in the noisy channel with dual-syndrome trellis codes (Hong Kong, 2018), pp. 1–7
- J. Tao, S. Li, X. Zhang, Z. Wang, Towards robust image steganography. *IEEE Trans. Circuits Syst. Video Technol.* **29**, 594–600 (2019)
- W. Lu, J. Zhang, X. Zhao, W. Zhang, J. Huang, Secure robust jpeg steganography based on autoencoder with adaptive BCH encoding. *IEEE Trans. Circuits Syst. Video Technol.* **31**, 2909–2922 (2021)
- Z. Zhao, Q. Guan, H. Zhang, X. Zhao, Improving the robustness of adaptive steganographic algorithms based on transport channel matching. *IEEE Trans. Inf. Forensics Secur.* **14**, 1843–1856 (2018)
- J. Zhang, X. Zhao, X. He, H. Zhang, Improving the robustness of JPEG steganography with robustness cost. *IEEE Signal Process. Lett.* **29**, 164–168 (2021)
- K. Zeng, K. Chen, W. Zhang, Y. Wang, N. Yu, Improving robust adaptive steganography via minimizing channel errors. *Signal Process.* **195**, 108498 (2022)
- H. Fu, X. Zhao, X. He, Improving anticompression robustness of JPEG adaptive steganography based on robustness measurement and DCT block selection. *Security Commun. Netw.* **2021**, 1–15 (2021)
- S. Lin, D.J. Costello. Error control coding (Prentice-Hall, Inc., Bergen County, 2004)
- K. He, X. Zhang, S. Ren, J. Sun, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Deep residual learning for image recognition (IEEE, Las Vegas, 2016)
- V. Nair, G.E. Hinton, in *Proc. Int. Conf. Mach. Learn.* Rectified linear units improve restricted boltzmann machines. (ACM, Madison, 2010), pp. 807–814
- P. Bas, T. Filler, T. Pevný, in *Proc. Inf. Hiding*, ed. by T. Filler, T. Pevný, S. Craver, A. Ker. "Break Our Steganographic System": The ins and outs of organizing BOSS (Springer, Prague, 2011), pp. 59–70
- J. KDeng, W. Dong, R. Socher, L.J. Li, K. Li, L. Fei-Fei, in *Proc. Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* ImageNet: A large-scale hierarchical image database (IEEE, Miami, 2009), pp. 248–255
- T. Ramabadran, S. Gaitonde, A tutorial on CRC computations. *IEEE Micro* **8**(4), 62–75 (1988)
- I.S. Reed, X. Chen, *Reed-Solomon Codes* (Springer US, Boston, 1999), pp.233–284
- R. Bose, D. Ray-Chaudhuri, On a class of error correcting binary group codes. *Inf. Control.* **3**(1), 68–79 (1960)

**Publisher's Note**

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.