PAPER *Special Section on Enriched Multimedia—Making Multimedia More Convenient and Safer—*

# Image Watermarking Technique Using Embedder and Extractor Neural Networks

**Ippei HAMAMOTO**[†], *Nonmember and* **Masaki KAWAMURA**[†a)], ***Senior Member***

**SUMMARY** An autoencoder has the potential ability to compress and decompress information. In this work, we consider the process of generating a stego-image from an original image and watermarks as compression, and the process of recovering the original image and watermarks from the stego-image as decompression. We propose embedder and extractor neural networks based on the autoencoder. The embedder network learns mapping from the DCT coefficients of the original image and a watermark to those of the stego-image. The extractor network learns mapping from the DCT coefficients of the stego-image to the watermark. Once the proposed neural network has been trained, the network can embed and extract the watermark into unlearned test images. We investigated the relation between the number of neurons and network performance by computer simulations and found that the trained neural network could provide high-quality stego-images and watermarks with few errors. We also evaluated the robustness against JPEG compression and found that, when suitable parameters were used, the watermarks were extracted with an average BER lower than 0.01 and image quality over 35 dB when the quality factor $Q$ was over 50. We also investigated how to represent the watermarks in the stego-image by our neural network. There are two possibilities: distributed representation and sparse representation. From the results of investigation into the output of the stego layer (3rd layer), we found that the distributed representation emerged at an early learning step and then sparse representation came out at a later step.

***key words:*** *digital watermarking, neural network, autoencoder, distributed representation, sparse representation*

## 1. Introduction

Digital watermarking methods are commonly used for protecting digital content such as images, videos, and audio data. In order to maintain the quality of the content, watermarks need to be embedded in a way that is nearly imperceptible, and one of the best ways to do this is by considering human visual and auditory characteristics. A secret message to be embedded is usually encoded by error-correcting or spread spectrum techniques. The encoded message is called a watermark, and an image in which the watermark is embeded is called a stego-image. It needs to be possible for the watermark to be extracted from the stego-image, even if the image is attacked illegally or modified legally. Therefore, a robust watermarking method should be developed.

Most watermarking methods are complex techniques composed of embedding and extraction processes. The embedding process can be further broken down into three techniques: (1) selection of watermarking domain, (2) redundant representation of watermarks, and (3) embedding. For robust watermarking, suitable combinations of these techniques are selected in order to extract watermarks from attacked images correctly.

The most popular watermarking domain is the frequency domain, e.g., the discrete cosine transform (DCT) and discrete wavelet transform (DWT) domains [1]–[7]. Since JPEG and JPEG2000 compression respectively use the DCT and DWT, it is reasonable to embed the watermarks into the DCT and DWT domains. For example, the lower frequencies of the DCT domain have robustness against the JPEG compression. Another domain is the feature domain, e.g., the scale-invariant feature transform (SIFT) domain [8]–[10]. In methods using the feature domain, robust feature points are extracted and selected. These points have robustness against both clipping and geometric transform such as scaling and rotation of the image.

Even if a robust domain is selected, watermarks might be damaged by attacks involving compression, geometric transform, or clipping. Therefore, in order to decode messages correctly from damaged watermarks, the messages to be embedded are encoded by the error-correcting or spread spectrum techniques. The criteria for robustness of watermarks and image quality are defined by the committee of Information Hiding and its Criteria for Evaluation (IHC), IEICE [11].

In spread spectrum watermarking [1], [2], [12], [13], a message is spread by a spread code, and the spread message is then embedded as a watermark. Error-correcting codes, e.g., Bose-Chaudhuri-Hocquenghem (BCH) [14], [15] and low-density parity-check (LDPC) [16]–[18] codes, are also used to ensure robustness of the messages. In general, there is a trade-off relation between image quality and the bit error of the message. This relation can be controlled by the code length of the watermark (codeword). Since the LDPC code has a very high error correcting capability, a short watermark can be generated. This is desirable because shorter watermarks are better for the quality of the stego-image. Recently, a watermarking method using LDPC code [18] was able to meet the requirements of the IHC ver. 4.

The embedding technique is used for embedding a watermark into values of the transform domain of a host image. In order to keep the quality of the stego-image, degradation of the image should be kept to a minimum while embedding the watermark. In this work, we focus on the embedding technique. There are two types of embedding technique:

Manuscript received March 26, 2018.
Manuscript revised August 8, 2018.
Manuscript publicized October 19, 2018.

[†]The authors are with Graduate School of Sciences and Technology for Innovation, Yamaguchi University, Yamaguchi-shi, 753–8512 Japan.
a) E-mail: m.kawamura@m.ieice.org

DOI: 10.1587/transinf.2018MUP0006

blind and non-blind. While the blind type does not require any information about the original image while extracting and decoding from the stego-image, the non-blind one does require the original image. Therefore, the non-blind one may be impractical for commercial use.

Several embedding techniques have already been proposed. The literature [19] serves as a useful reference. Additive embedding has been used in many conventional methods due to its simplicity [4], [7], including most of the spread-spectrum watermarking methods [1]–[3], [12], [16]. The stego-image is generated by adding the watermark to the host image, so this technique is the non-blind type.

The least significant bit (LSB) modification is one of the simplest blind embedding techniques. It embeds the watermark by using it as a replacement for the LSB of a pixel value, a DCT coefficient, or a DWT coefficient in the host image [5], [20]. Since the LSB has little information, the degradation of the stego-image is small. However, the watermarks are fragile due to the compression. Another embedding technique is the patchwork algorithm [21], [22]. In this technique, the DCT coefficients are grouped according to watermarks, and a watermark is embedded into the corresponding coefficient by additive embedding.

The wet paper code [23], [24] and matrix embedding algorithms [25] use a system of linear equations to embed watermarks: specifically, the watermarks are scattered in binary data generated from the stego-image. Since the embedding matrix or parity check matrix are shared, they are blind types. However, the watermarks are fragile as a result of the compression when converting into binary.

Quantization index modulation (QIM) [6], [9], [10], [13], [17], [18], [26] is a blind embedding technique that is robust against compression. Specifically, since this technique quantizes coefficients of the host image according to a watermark, it has robustness against JPEG compression. Therefore, there have been many proposals for using it to meet the IHC requirements [11].

Recently, embedding techniques using layered neural networks have been proposed. In this technique, neural networks are applied to adjust the embedding strength [27]–[29]. The trained neural networks are used for determining the suitable strength, not for embedding watermarks. On the other hand, correlations among the coefficients of host images are learned by neural networks [30]–[32]. In this approach, since the neural networks can predict the value of a coefficient, watermarks can be embedded by additive embedding. That is, the value of the stego-image becomes the sum of the value of the host image and one bit of a watermark. While decoding, the watermark can be estimate by the difference of output value of the same neural network. Since the neural networks store side information about an original host image, these methods are non-blind types. In a similar manner, probabilistic neural networks [33]–[35] also store the relation between a stego-image and a watermark. Each image needs the respective probabilistic neural network to extract the watermark.

We propose a novel embedding technique using a layered neural network. In our view, the embedding process can be considered as a mapping from both a host image and watermarks to a stego-image. At the same time, the extraction process can be considered as a mapping from the stego-image to the watermarks. These mapping can be represented by a layered neural network. An autoencoder [36] is an hourglass-shaped neural network. It can map from high dimensional input to the lower dimensional output of an intermediate layer and then map from the lower output to the original input. The front side of the network can be an encoder and the back side of the network can be a decoder. By training a layered neural network as both encoder and decoder, the proposed method can achieve blind embedding.

The proposed neural network consists of both encoder and decoder neural networks. Lower-frequency coefficients in the DCT domain of an original image are given to the encoder neural network as teacher inputs. The decoder neural network receives the output of the encoder as inputs and learns a watermark as teacher output. After training the networks, the encoder can output the DCT coefficients of stego-images and the decoder can output the watermark from the stego-images.

In summary, the existing methods store the side information about original images in neural networks and then use this information to extract the watermarks. A dedicated network is required for each image. Strictly speaking, they are not blind types. In contrast, the proposed method needs only one neural network to embed and extract watermarks, and no original images are required while extracting.

A unique feature of our proposal is how and with what the proposed network is trained. The outputs of the encoder are slightly different from the original DCT coefficients due to watermarks. There are two possibilities to represent the watermarks in DCT coefficients. One is distributed representation, where a one-bit watermark is widely represented in all coefficients. In this case, the distributed representation can be robust against attacks. The other is sparse representation, where a one-bit watermark is sparsely represented in a few coefficients. In this case, a high-quality image can be generated by the sparse representation. These representations might be changed by training.

In Sect. 2 of this paper, we briefly discuss related work using a neural network. Section 3 presents our encoder and decoder neural networks. Computer simulations in Sect. 4 demonstrate that our encoder and decoder can learn watermarks and generate high-quality stego-images. In Sect. 5, we clarify which representations are learned in the neural network. We conclude in Sect. 6 with a brief summary.

## 2. Watermarking Method Using Neural Network

Hwang *et al.* [30] proposed a method that embeds watermarks into DCT coefficients by using a layered neural network. Since the neural network learns the correlation between the DCT coefficients of an original image, this is considered one of the blind type methods; that is, the watermarks can be extracted from a stego-image by using the

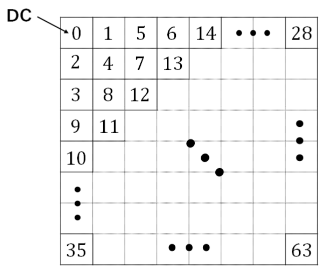


**Fig. 1** Indexes in a DCT block.

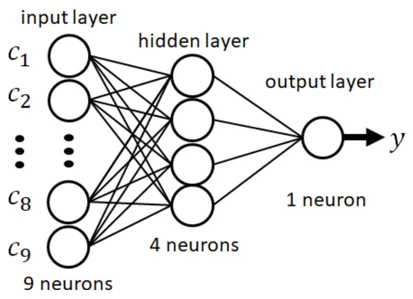


**Fig. 2** Structure of the neural network in the method of Hwang *et al.*

trained neural network. However, this method is not fully blind due to its use of side information. Since the network can learn only one original image, it stores the information related to the original image as side information.

### 2.1 Embedding Domain

In Hwang *et al.*'s method [30], several $8 \times 8$ pixel blocks are selected from an original image with allowing overlap. The selected blocks are transformed by DCT, and then AC elements are indexed from 1 to 63 in zigzag order, as shown in Fig. 1. The DC elements are indexed as 0. Let the $i$-th DCT coefficient of the $\mu$-th block be $C_i^\mu$, $i = 0, 1, \cdots, 63$. This method embeds one-bit of a watermark in the coefficient $C_{12}^\mu$ for each block. The coefficient $\widetilde{C}_{12}^\mu$ of the stego-image is given by the output of the trained neural network.

### 2.2 Structure of Neural Network

The method of Hwang *et al.* uses a fully connected neural network that consists of three layers: input, hidden, and output layers, as shown in Fig. 2. There are nine neurons in the input layer, four neurons in the hidden one, and one neuron in the output one. The activation function for each neuron is a sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{1}$$

The DCT coefficients $C_i^\mu$ in the $\mu$-th block are normalized to

$$c_i^\mu = N_0\left(C_i^\mu\right), \tag{2}$$

where $N_0(x)$ is the normalization function defined as

$$N_0(x) = \frac{x + 1000}{2000}. \tag{3}$$

The nine normalized coefficients $c_i^\mu, i = 1, 2, \cdots, 9$, are fed into the neural network, which is then trained so as to output the 12th coefficient $c_{12}^\mu$.

### 2.3 Embedding and Extracting Processes

First, we explain the embedding process. Let the output for the $\mu$-th block be $y^\mu$ when the nine coefficients $\boldsymbol{c}_{\text{in}}^\mu = \left(c_1^\mu, c_2^\mu, \ldots, c_9^\mu\right)^\top$ are fed into the trained neural network. A watermark $\omega^\mu \in \{0, 1\}$ is embedded into the 12th DCT coefficient $C_{12}^\mu$. The coefficient of the stego-image, $\widetilde{C}_{12}^\mu$, is given by

$$\widetilde{C}_{12}^\mu = \begin{cases} N_0^{-1}(y^\mu) - \delta, & \omega^\mu = 0 \\ N_0^{-1}(y^\mu) + \delta, & \omega^\mu = 1 \end{cases}, \tag{4}$$

where $\delta$ is an embedding strength and $N_0^{-1}(\cdot)$ is an inverse function of (3), given by

$$N_0^{-1}(x) = 2000x - 1000. \tag{5}$$

Next, we explain the extracting process. The normalized coefficients $\boldsymbol{c}_{\text{in}}^\mu$ are obtained in the same way as above. The output $y^\mu$ can be obtained by feeding to the coefficients $\boldsymbol{c}_{\text{in}}^\mu$, and also the 12th coefficient of the stego-image is $\widetilde{c}_{12}^\mu$. Therefore, extracted watermark $\hat{\omega}^\mu$ is given by

$$\hat{\omega}^\mu = \begin{cases} 0, & \widetilde{c}_{12}^\mu < y^\mu \\ 1, & \widetilde{c}_{12}^\mu > y^\mu \end{cases}. \tag{6}$$

### 2.4 Point at Issue

The neural network is trained by an original image, which means it cannot embed and extract watermarks into other images. That is, the network is image-dependent, and it stores the information related to the original image as side information. Therefore, this method is an incomplete blind type embedding technique.

## 3. Proposed Method

The autoencoder [36] has the potential ability to both compress and decompress information. We focus here on this ability. We consider the process of generating a stego-image from an original image and a watermark as compression, and the process of recovering the original image and the watermark from the stego-image as decompression.

In this section, we explain our proposed neural network in detail. Our network can embed watermarks into unlearned images; that is, it is blind type. We also explore how to represent the watermarks in a stego-image by our neural network.

### 3.1 Structure of Proposed Neural Network

The proposed neural network has five fully connected layers, as shown in Fig. 3. The 1st layer is an input layer, the

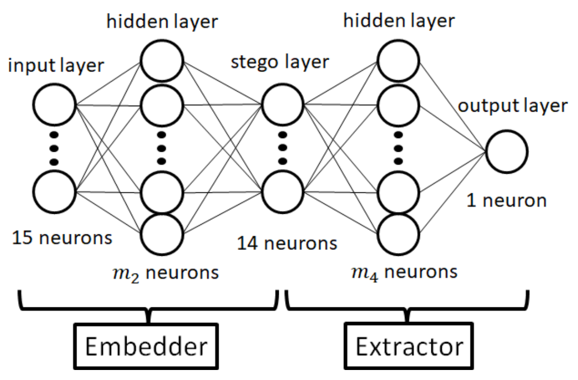


**Fig. 3** Structure of the proposed neural network.

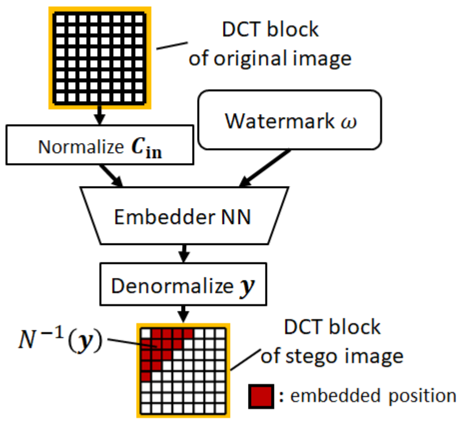


**Fig. 4** Embedding procedure in encoder network.

3rd one is a hidden layer called a stego layer, and the 5th one is an output layer. The network from the 1st to 3rd layers is called an embedder network, and the network from the 3rd to 5th layers is called an extractor network. The 2nd and 4th layers are hidden layers for the embedder and extractor networks, respectively. Let the number of neurons in $l$-th layer be $m_l$. We define $m_1 = 15, m_3 = 14$, and $m_5 = 1$, since 14 DCT coefficients of an original image and one-bit watermark are fed, 14 coefficients of a stego-image are output in the stego layer, and a one-bit watermark is output in the output layer. Note that the dimensions of the output layer are different from those of the autoencoder. Since the autoencoder learns identity mapping, the dimensions of the output layer are the same as those of the input layer. In the proposed network, since we do not use the original coefficients in the output layer, only one dimension exists.

The numbers $m_2$ and $m_4$ are selected in accordance with learning performance. The activation function for the 1st, 3rd, and 5th layers is an identity function, and the function for the 2nd and 4th layers is the rectified linear unit (ReLU) [37] defined by

$$\mathrm{ReLU}(x) = \max(0, x). \tag{7}$$

Let the output of the $j$-th neuron in the $l$-th layer be $V_j^l$, $j = 1, 2, \ldots, m_l$. The output of the $i$-th neuron in the $(l+1)$-th layer is given by

$$V_i^{l+1} = f^{l+1}\left(\sum_{j=1}^{m_l} w_{ij}^l V_j^l - \theta_i^l\right), \tag{8}$$

where $f^{l+1}(\cdot)$ is the activation function in the $(l+1)$-th layer, $w_{ij}^l$ is a connection from the $j$-th neuron in the $l$-th layer to the $i$-th neuron in the $(l+1)$-th layer, and $\theta_j^l$ is a threshold. Now, let the outputs in the stego layer $(l = 3)$ and output layer $(l = 5)$ be

$$\boldsymbol{y} = \left(V_1^3, V_2^3 \ldots, V_{m_3}^3\right)^\top, \tag{9}$$

$$z = V_1^5. \tag{10}$$

## 3.2 Preprocessing

The luminance of an original image, which is $W \times H$ pixel size, is divided into $8 \times 8$-pixel blocks without overlapping. Therefore, the number of blocks is $M = \left\lfloor \frac{W}{8} \right\rfloor \times \left\lfloor \frac{H}{8} \right\rfloor$. The AC elements of the DCT coefficients of each $8 \times 8$-pixel block are indexed from 1 to 63 in zigzag order. Let the $i$-th coefficient in the $\mu$-th block be $C_i^\mu$. The fourteen lower-frequency coefficients $\boldsymbol{C}_{\text{in}}^\mu = \left(C_1^\mu, C_2^\mu, \ldots, C_{14}^\mu\right)^\top$ are selected as an embedding domain. The coefficients are normalized by

$$\boldsymbol{c}_{\text{in}}^\mu = N\left(\boldsymbol{C}_{\text{in}}^\mu\right), \tag{11}$$

where $N(\cdot)$ is the normalization function defined as

$$N(\boldsymbol{x}) = \frac{1}{C_{\max}}\boldsymbol{x} \tag{12}$$

and $C_{\max}$ is the maximum value of the AC elements for any image, i.e., $C_{\max} = 1020$. The DCT coefficient $\boldsymbol{C}_{\text{in}}^\mu$ can also be obtained from the normalized coefficient $\boldsymbol{c}_{\text{in}}^\mu$ by the inverse function,

$$\boldsymbol{C}_{\text{in}}^\mu = N^{-1}\left(\boldsymbol{c}_{\text{in}}^\mu\right), \tag{13}$$

where $N^{-1}(\cdot)$ is defined by

$$N^{-1}(\boldsymbol{x}) = C_{\max}\boldsymbol{x}. \tag{14}$$

## 3.3 Embedder and Extractor Networks

Here, we explain the proposed embedder and extractor networks. As shown in Fig. 4, watermarks can be embedded into the DCT coefficients in an original image, $\boldsymbol{C}_{\text{in}}^\mu$, by replacing coefficients in a stego-image with the output of the embedder network, $N^{-1}(\boldsymbol{y})$. Moreover, as shown in Fig. 5, the watermark $\omega^\mu$ can be extracted by feeding the coefficients $\tilde{\boldsymbol{c}}_{\text{in}}^\mu$ of the stego-image into the extractor network.

### 3.3.1 Embedder Network

In the embedder network, a one-bit watermark is embedded into one $8 \times 8$-pixel block. As shown in Fig. 4, the original image is divided into blocks, and then these blocks are

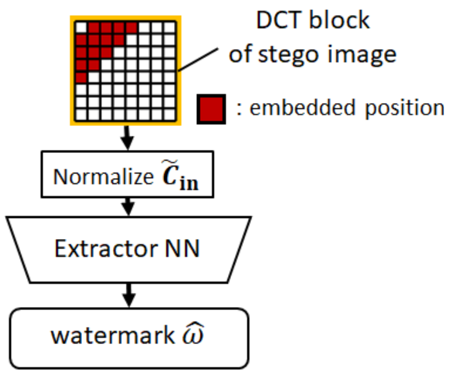


**Fig. 5** Extraction procedure in extractor network.

transformed to DCT domain. The DCT coefficients in the $\mu$-th block are normalized by (12). From the normalized coefficient $\boldsymbol{c}_{\text{in}}^{\mu}$ and one-bit watermark $\omega^{\mu} \in \{-1, +1\}$, input vector $\boldsymbol{x}^{\mu}$ is defined as

$$\boldsymbol{x}^{\mu} = \left( \left(\boldsymbol{c}_{\text{in}}^{\mu}\right)^{\top}, \quad \omega^{\mu} \right)^{\top}. \tag{15}$$

The input vector $\boldsymbol{x}^{\mu}$ is fed into the embedder network, that is, $V_i^1 = x_i^{\mu}$. Note that the dimension of the input vector is fifteen. The embedder network outputs the coefficient $\boldsymbol{y}^{\mu}$, whose dimension is fourteen. The DCT coefficient $\widetilde{\boldsymbol{C}}_{\text{in}}^{\mu}$ of the stego-image can be obtained from $\boldsymbol{y}^{\mu}$ by inverse function (14). Then, the luminance values can be given by inverse DCT (IDCT). By connecting all of the blocks, the whole stego-image can be obtained.

### 3.3.2 Extractor Network

In the extractor network, a one-bit watermark is extracted from each $8 \times 8$-pixel block by using the trained neural network. As shown in Fig. 5, the normalized coefficient in the $\mu$-th block $\tilde{\boldsymbol{c}}_{\text{in}}^{\mu}$ is fed into the extractor network, i.e., the stego layer. The extracted watermark can be obtained as the output $z^{\mu}$ in the output layer, that is,

$$\hat{\omega}^{\mu} = \operatorname{sgn}(z^{\mu}), \tag{16}$$

where $\operatorname{sgn}(z)$ is the signum function defined by

$$\operatorname{sgn}(z) = \begin{cases} +1, & z \geq 0 \\ -1, & z < 0 \end{cases}. \tag{17}$$

## 3.4 Training the Neural Network

In our method, the normalized coefficients $\boldsymbol{c}_{\text{in}}^{\mu}$ and one-bit watermark $\omega^{\mu} \in \{-1, +1\}$ are fed into the input layer. The encoder of the proposed neural network is trained so as to output the normalized coefficient $\boldsymbol{c}_{\text{in}}^{\mu}$ in the stego layer, and the extractor is trained so as to output the watermark $\omega^{\mu}$ in the output layer.

### 3.4.1 Input and Teacher Data

The original images can be classified into training and test images. A training image is used for training the neural network. The DCT coefficients from $M$ blocks in the training image are normalized by (11). The training dataset can be generated from the normalized coefficients $\boldsymbol{c}_{\text{in}}^{m}$ and $M$-bit watermarks $\omega^{m} \in \{-1, +1\}$, $m = 1, 2, \ldots, M$. The input dataset can be defined by

$$\mathcal{X} = \{\boldsymbol{\xi}^{p} \mid p = 1, 2, \cdots, 2M\}. \tag{18}$$

For $1 \leq p \leq M$, $\boldsymbol{\xi}^{p}$ is the input data corresponding to the watermark $\omega = +1$, i.e.,

$$\boldsymbol{\xi}^{p} = \left( \left(\boldsymbol{c}_{\text{in}}^{p}\right)^{\top}, \quad +1 \right)^{\top}, \tag{19}$$

and for $M < p \leq 2M$, it is the input data corresponding to the watermark $\omega = -1$, i.e.,

$$\boldsymbol{\xi}^{p} = \left( \left(\boldsymbol{c}_{\text{in}}^{p-M}\right)^{\top}, \quad -1 \right)^{\top}. \tag{20}$$

The encoder network outputs the normalized DCT coefficients of a stego-image as shown in Fig. 4. Therefore, it is trained so as to output the normalized coefficients $\boldsymbol{c}_{\text{in}}^{p} = \left(c_1^p, c_2^p, \cdots, c_{14}^p\right)^{\top}$. The teacher data for the stego layer, $\boldsymbol{t}_{\text{s}}^{p}$, is given by

$$\boldsymbol{t}_{\text{s}}^{p} = \begin{cases} \boldsymbol{c}_{\text{in}}^{p}, & p = 1, 2, \cdots, M \\ \boldsymbol{c}_{\text{in}}^{p-M}, & p = M+1, M+2, \cdots, 2M \end{cases}. \tag{21}$$

The output layer has to output watermark $\omega^{p}$. Therefore, the teacher data for the output layer, $t_{\omega}^{p}$, is given by

$$t_{\omega}^{p} = \begin{cases} +1, & p = 1, 2, \cdots, M \\ -1, & p = M+1, M+2, \cdots, 2M \end{cases}. \tag{22}$$

### 3.4.2 Error Function

Now, we define the learning error for $p$-th dataset $(\boldsymbol{\xi}^{p}, \boldsymbol{t}_{\text{s}}^{p}, t_{\omega}^{p})$ as $R\left(\boldsymbol{W}; \boldsymbol{\xi}^{p}, \boldsymbol{t}_{\text{s}}^{p}, t_{\omega}^{p}\right)$, where $\boldsymbol{W}$ represents all connections and thresholds in the neural network. Moreover, we define the error function for the whole dataset as

$$E(\boldsymbol{W}) = \frac{1}{2M} \sum_{p=1}^{2M} R\left(\boldsymbol{W}; \boldsymbol{\xi}^{p}, \boldsymbol{t}_{\text{s}}^{p}, t_{\omega}^{p}\right). \tag{23}$$

The optimal connection $\boldsymbol{W}$ can be obtained by back-propagation [38]. The error $R\left(\boldsymbol{W}; \boldsymbol{\xi}^{p}, \boldsymbol{t}_{\text{s}}^{p}, t_{\omega}^{p}\right)$ consists of mean square error (MSE) for the stego layer, $R_{\text{s}}$, and MSE for the output layer, $R_{\omega}$. That is, the error $R\left(\boldsymbol{W}; \boldsymbol{\xi}^{p}, \boldsymbol{t}_{\text{s}}^{p}, t_{\omega}^{p}\right)$ is defined by

$$R\left(\boldsymbol{W}; \boldsymbol{\xi}^{p}, \boldsymbol{t}_{\text{s}}^{p}, t_{\omega}^{p}\right) = (1 - \gamma) R_{\text{s}} + \gamma R_{\omega}, \tag{24}$$

where

$$R_{\text{s}} = \frac{1}{m_3} \|\boldsymbol{y}^{p} - \boldsymbol{t}_{\text{s}}^{p}\|^2, \tag{25}$$

$$R_{\omega} = \left(z^{p} - t_{\omega}^{p}\right)^2, \tag{26}$$

and $0 \leq \gamma \leq 1$ is the weight parameter. The value of $\gamma$

needs to be fixed by learning performance. $\|\cdot\|$ denotes 2-norm. (25) and (26) indicate the degree of deterioration of the image and that of the watermarks, respectively. That is, the proposed method optimizes both the image quality and the bit error rate.

## 4. Computer Simulations

Let us define the number of neurons in the embedder and extractor networks, $m_2$ and $m_4$, and the weight parameter $\gamma$ in (24). We will fix these parameters by computer simulations.

### 4.1 Experimental Conditions

#### 4.1.1 Training and Test Images

There are six IHC standard images defined by the IHC committee [11], as shown in Fig. 6. We select one of these images as a training image and use the others as test images. The width and height of each image are $W = 4608$ and $H = 3456$, respectively.

Since our method uses $8 \times 8$-pixel blocks to embed the watermark, an image is divided into $M = \left\lfloor \frac{4608}{8} \right\rfloor \times \left\lfloor \frac{3456}{8} \right\rfloor = 248,832$ blocks. Therefore, $M = 248,832$-bit watermarks can be embedded into an image. In terms of image quality, it is unnecessary to embed into the whole image. Since all blocks are used to train the neural network, the watermarks are embedded into all blocks to evaluate the performance. The watermarks $\omega^\mu \in \{-1, +1\}$ are generated at random.

(a) image 1 (b) image 2 (c) image 3 (d) image 4 (e) image 5 (f) image 6

**Fig. 6** IHC standard images.

#### 4.1.2 Parameters of Neural Network

The proposed neural network was implemented by Chainer [39]. The initial value of the connection $w^l_{ij}$ was given by the normal distribution of $\mathcal{N}\left(0, \frac{1}{m_l}\right)$, and the initial value of the threshold $\theta^l_j$ was zero. The neural network is trained by an adaptive moment estimation (Adam) optimizer [40]. The learning rate of Adam is 0.0001. The mini-batch size was 100. Values for other parameters were the defaults used in [40]. According to 3.4.1, $2M$ training data were generated from a training image. The computer simulations were executed 10 times by changing the initial values of $w^l_{ij}$. The performance was evaluated after 1000-epoch training.

#### 4.1.3 Performance Evaluation

The performance of watermarking methods is evaluated by both bit error rate (BER) and peak signal-to-noise ratio (PSNR). The correctness of a watermark is evaluated by the BER, which is defined as

$$\mathrm{BER} = \frac{1}{2}\left(1 - \frac{1}{M}\left|\sum_{\mu=1}^{M} \omega^\mu \hat{\omega}^\mu\right|\right), \tag{27}$$

where $\omega^\mu \in \{-1, +1\}$ and $\hat{\omega}^\mu \in \{-1, +1\}$ are the original and extracted watermarks, respectively. The image quality for a stego-image is evaluated by PSNR, which is defined as

$$\mathrm{PSNR} = 10 \log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right) \text{ [dB]}, \tag{28}$$

where MSE is mean squared error defined by

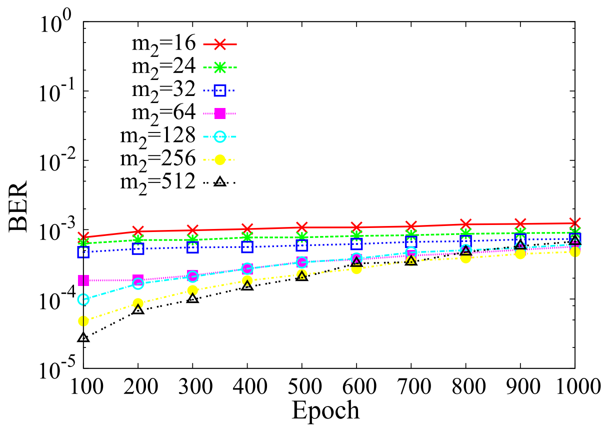
$$\mathrm{MSE} = \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}\left(I^S_{ij} - I^O_{ij}\right)^2, \tag{29}$$

where $I^O$ and $I^S$ are the original and stego-images, respectively.

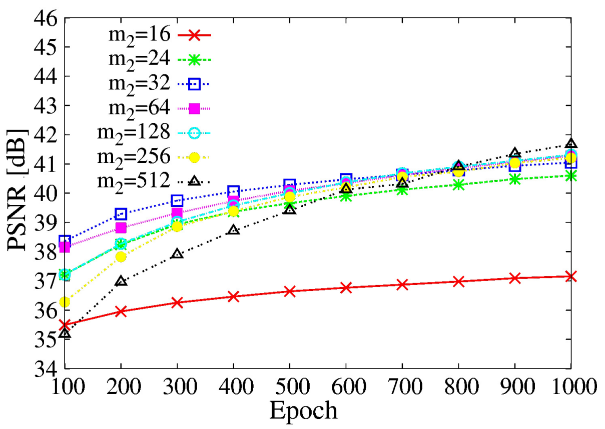
### 4.2 Number of Neurons, $m_2$, in Embedder

Let us fix the number of neurons, $m_2$, in the embedder network by computer simulations. In this case, the number of neurons in the extractor was $m_4 = 64$ and the weight parameter in (24) was $\gamma = 0.5$. The average BERs and PSNRs for $m_2 \in \{16, 24, 32, 64, 128, 256, 512\}$ are shown in Fig. 7. The abscissa axis represents epoch, which means one iteration over all of the training data. The ordinate axes represent (a) BER and (b) PSNR.

On the whole, at 100 epoch, both BERs and PSNRs were smallest, that is, errors in a watermark were the smallest but image quality was the worst. After 1000-epoch training, PSNRs were gradually increased, while BERs were slightly increased. That is, errors were larger and the image quality was better than those at initial epoch. Table 1



(a) epoch vs. BER

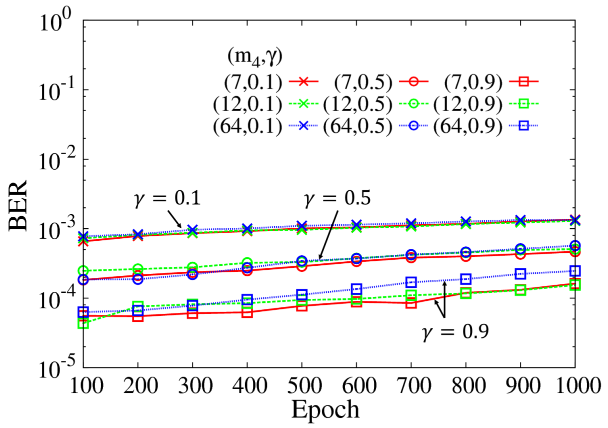


(b) epoch vs. PSNR

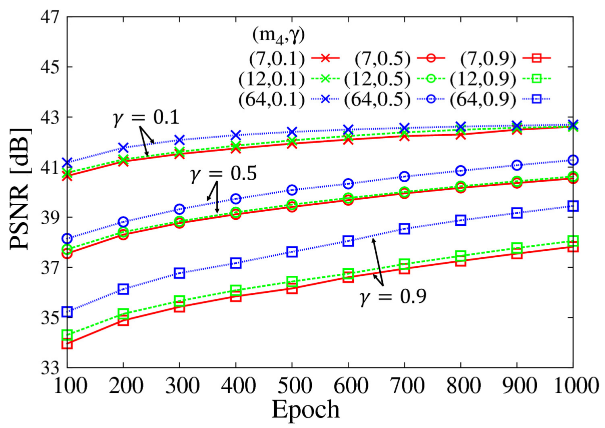
**Fig. 7** BER and PSNR for $m_2$.

**Table 1** BERs and PSNRs at 1000 epoch for $m_2$.

| $m_2$ | 16 | 24 | 32 | 64 | 128 | 256 | 512 |
|---|---|---|---|---|---|---|---|
| BER | 0.00124 | 0.000903 | 0.000735 | 0.000570 | 0.000613 | 0.000479 | 0.000675 |
| PSNR | 37.2 | 40.6 | 41.0 | 41.3 | 41.3 | 41.2 | 41.7 |



(a) epoch vs. BER



(b) epoch vs. PSNR

**Fig. 8** BER and PSNR for $m_4$ and $\gamma$.

lists the BERs and PSNRs at 1000 epoch. There are three groups for $m_2$. In the case of small $m_2 \in \{16, 24, 32\}$, BERs were not good due to high BER, and also image quality was not good. Therefore, the number of neurons, $m_2 \leq 32$, was not enough for embedding watermarks. In the case of middle $m_2 \in \{64, 128, 256\}$, BERs were converged to almost 0.0005 at 1000 epoch, and also PSNRs were converged to almost 41 dB. These numbers were sufficient for embedding. In the case of large $m_2 = 512$, BER was converged to almost 0.0005 at 1000 epoch, the same as in the middle case. Moreover, PSNR reached over 41 dB. In summary, at least $m_2 \geq 64$ neurons were required for the embedder network. Note that although the embedder with a large number of neurons could achieve high image quality, it required both a large training dataset and long computational time.

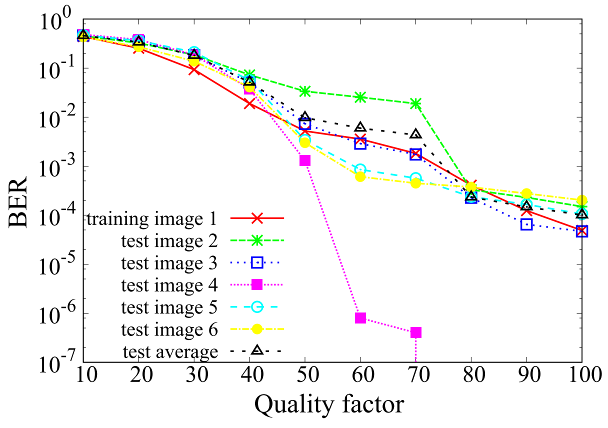
### 4.3 Number of Neurons, $m_4$, in Extractor and Weight Parameter $\gamma$

Next, let us fix the number of neurons, $m_4$, in the extractor network and the weight parameter $\gamma$ by computer simulations. The number $m_2$ in the embedder network was $m_2 = 64$. The average BERs and PSNRs for $m_4 \in \{7, 12, 64\}$ and $\gamma \in \{0.1, 0.5, 0.9\}$ are shown in Fig. 8. The abscissa axis represents epoch. The ordinate axes represent (a) BER and (b) PSNR.

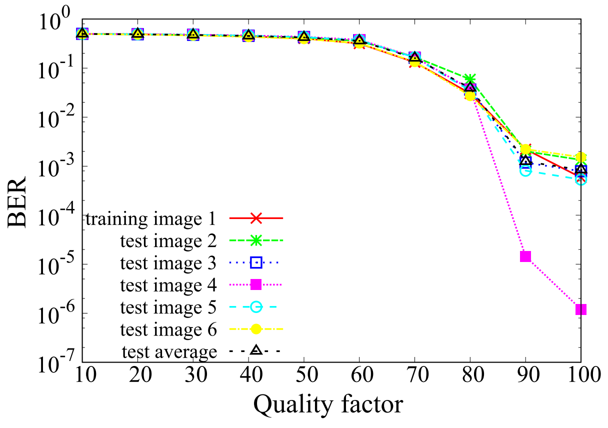
When the weight parameter $\gamma$ was close to 1, BERs and PSNRs were smaller than others at 1000 epoch. That is, watermarks had robustness but the image had low quality. In the case within the same $\gamma$, as the number of neurons $m_4$ was larger, BERs were larger and image quality was higher.

**Table 2** BERs and PSNRs at 1000 epoch for $m_4$ and $\gamma$.

| | $m_4 = 7$ | | | $m_4 = 12$ | | | $m_4 = 64$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\gamma = 0.1$ | $\gamma = 0.5$ | $\gamma = 0.9$ | $\gamma = 0.1$ | $\gamma = 0.5$ | $\gamma = 0.9$ | $\gamma = 0.1$ | $\gamma = 0.5$ | $\gamma = 0.9$ |
| BER | 0.00135 | 0.000466 | 0.000162 | 0.00129 | 0.000507 | 0.000155 | 0.00133 | 0.000570 | 0.000246 |
| PSNR | 42.6 | 40.5 | 37.8 | 42.6 | 40.6 | 38.1 | 42.7 | 41.3 | 39.4 |

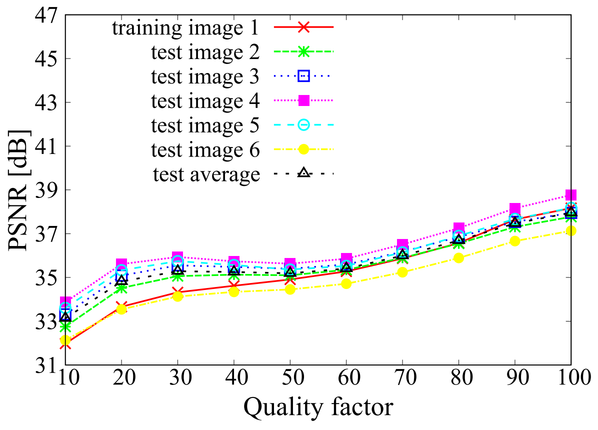


(a) low-error oriented model

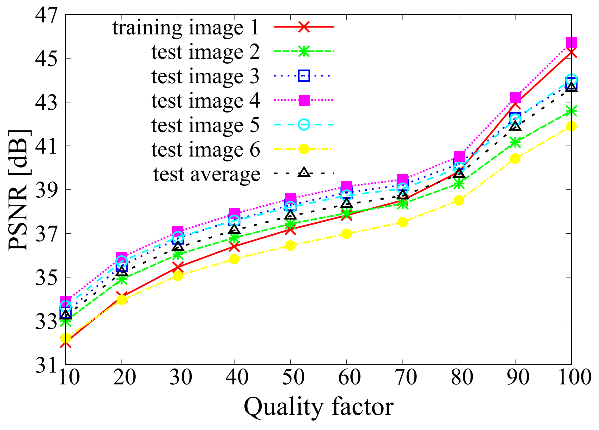


(b) image-quality oriented model

**Fig. 9** Quality factor vs. BER.



(a) low-error oriented model



(b) image-quality oriented model

**Fig. 10** Quality factor vs. PSNR.

Table 2 lists the BERs and PSNRs at 1000 epoch.

## 4.4 Robustness against JPEG Compression

From the findings in 4.2 and 4.3, we know that the embedder network needs at least $m_2 = 64$ neurons. The BER and image quality can be controlled by both the number of neurons, $m_4$, and the weight parameter $\gamma$. In this section, robustness against JPEG compression is evaluated in two models: a low-error oriented model (LE model) and an image-quality oriented model (IQ model). The former considers the lower BER and its parameters of the network given as $m_2 = 64, m_4 = 7$, and $\gamma = 0.9$. The latter considers image-quality and its parameters given as $m_2 = 64, m_4 = 64$, and $\gamma = 0.1$.

From six IHC standard images (Fig. 6), one image was used for training and the others for testing. After 1000-epoch learning with the training image, we could obtain stego-images by the trained embedder network. The stego-images were compressed with the quality factor $Q \in \{10, 20, \ldots, 100\}$. After that, watermarks were extracted from the compressed stego-images. The average BERs of the compressed images are shown in Fig. 9. The abscissa axis represents the quality factor $Q$, and the ordinate axis represents BER. PSNRs of the stego-images are shown in Fig. 10. The abscissa axis represents the quality factor $Q$, and the ordinate axis represents PSNR. In these figures, the line labeled 'test average' represents the average BER of the test images. The results for the LE and IQ models are shown in (a) and (b), respectively. Note that PSNR was measured using compressed stego-images in reference to uncompressed original images.

As shown in Fig. 9(a) and Fig. 10(a), the LE model could extract the watermarks with an average BER lower than 0.01 and could achieve at least 35 dB when $Q \geq 50$. Since all blocks were used to train the neural network, the watermarks were embedded into all blocks. These results show the worst cases for image quality, so it might seem that the quality is not good. From (b), the IQ model could achieve at least 40 dB when $Q \geq 85$.

Next, we checked the dependency on training images. Figure 11 shows the average BERs for the case where image 2 is used for training and the others are for testing. The conditions are the same as in Fig. 9(a). These results have the same tendency. We found that the BERs depended not on the training image but rather on the host images.

As described above, image quality can be improved by using fewer blocks. Moreover, bear in mind that watermarks are not encoded by any error-correcting codes. If an error-correcting code could be introduced, the BER could be improved. In this paper, we showed the performance without any error-correcting codes because we were investigating the ability of the neural network itself.

## 4.5 Secrecy of Watermark Extraction

Next, let us discuss the secrecy of watermark extraction. If the detailed procedure was open to the public, stego-images would be easily attacked, so in general, the details are hidden. Since the proposed method can generate a generic neural network, it is necessary to check its secrecy. We define three secrecy levels for our method:

Level 1 All parameters, i.e., initial values of connections $w_{ij}^l$, model parameters $m_2, m_4, \gamma$, and training images, are concealed.

Level 2 The model parameters $m_2, m_4$, and $\gamma$ are public, but the initial values of $w_{ij}^l$ and training images are concealed.

Level 3 The model parameters $m_2, m_4, \gamma$, and training images are public, but the initial values of $w_{ij}^l$ are concealed.

We evaluated the secrecy for these levels. The embedder network was trained over 1000 epochs by the IQ model using image 1 for the training image. The stego-image is generated from image 2 by using the embedder network.

In the case of level 1, the attacker trained a neural network by using different parameters, that is, the network was trained using the LE model and image 2 as a training image. Since there are two optimal models, the attacker might select the IQ model by chance. This case corresponds to level 2. In the case of level 3, the attacker could train by using the IQ model and image 1 as a training image, as with the embedder network, except for the initial values of the connections. Table 3 lists the average BER for these levels, which were 0.355, 0.216, and 0.107 for levels 1, 2, and 3.

We can assume level 1 for practical purposes. Even if the attacker could use the same model by chance, the average BER was 0.216, which is much larger than the BER of 0.00133 shown in Table 2. Therefore, the attacker is not able to extract the complete watermark.

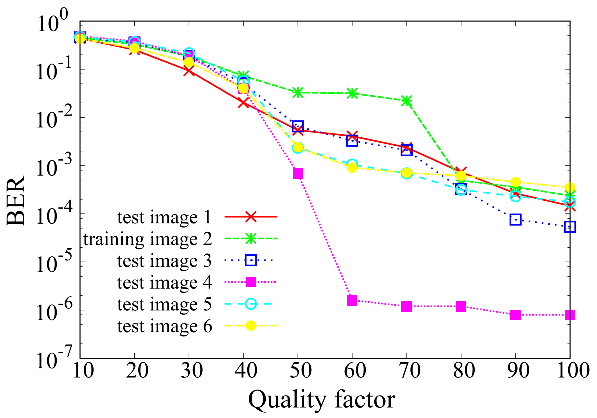


**Fig. 11** Quality factor vs. BER for low-error oriented model.

## 5. Distributed or Sparse Representation

We also investigated how to represent the watermarks in the stego-image by our neural network. There are two possibilities to represent the watermarks in DCT coefficients: distributed representation and sparse representation. In the former case, a one-bit watermark can be widely represented over all coefficients. In the latter case, the watermark can be sparsely represented in a few coefficients. We investigated the output of the stego layer.

### 5.1 Indicator

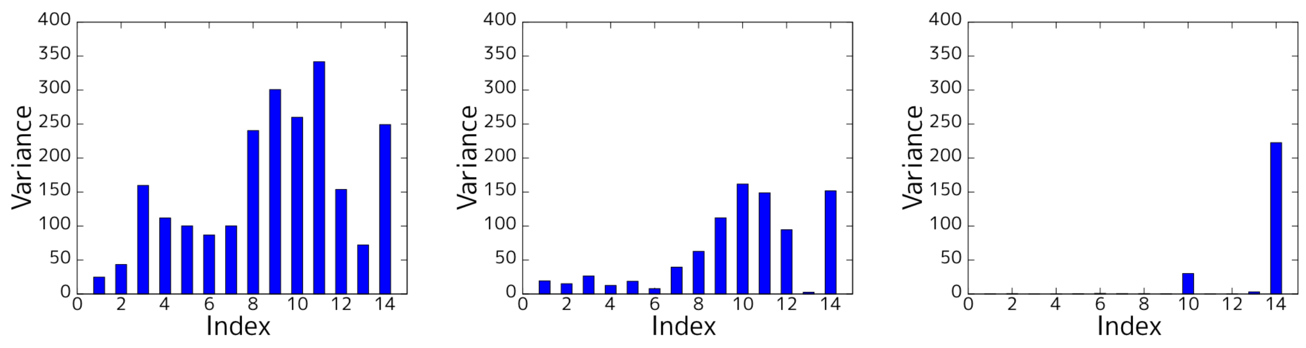
Using both the LE model and the IQ model, we investigated the output of the stego layer at $T \in \{100, 1000, 10000\}$ epoch. We defined the difference vector $\boldsymbol{d}^\mu = \left(d_1^\mu, d_2^\mu, \cdots, d_{14}^\mu\right)^\top$ as

$$\boldsymbol{d}^\mu = \widetilde{\boldsymbol{C}}_{\text{in}}^\mu - \boldsymbol{C}_{\text{in}}^\mu, \qquad (30)$$

where $\boldsymbol{C}_{\text{in}}^\mu$ and $\widetilde{\boldsymbol{C}}_{\text{in}}^\mu$ were the DCT coefficients of the original and stego-images, respectively. In addition, we defined the element-wise variance $v_i$ as

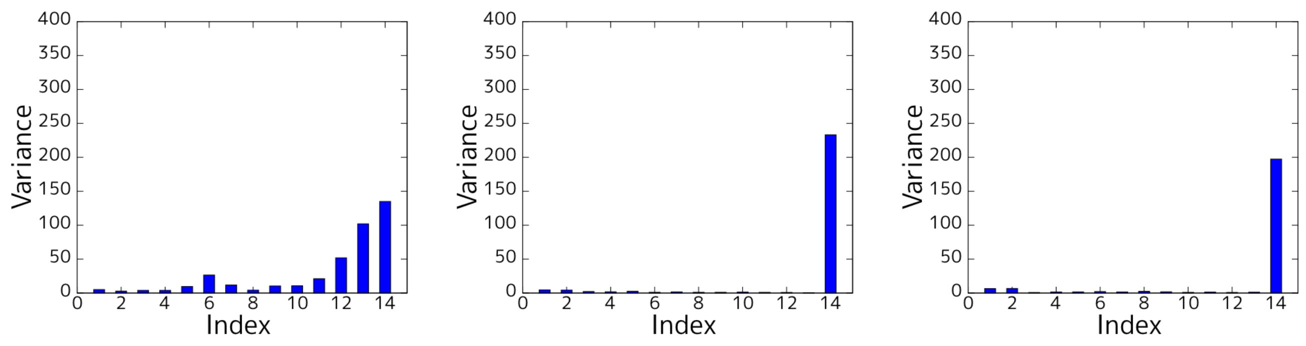
**Table 3** Conditions and BERs.

| | Level 1 | | Level 2 | | Level 3 | |
|---|---|---|---|---|---|---|
| Network | Embedder | Extractor | Embedder | Extractor | Embedder | Extractor |
| Training image | image 1 ≠ image 2 | | image 1 ≠ image 2 | | image 1 = image 1 | |
| Model | IQ model ≠ LE model | | IQ model = IQ model | | IQ model = IQ model | |
| Average BER | 0.355 | | 0.216 | | 0.107 | |



(a) $T = 100$ epoch (b) $T = 1000$ epoch (c) $T = 10000$ epoch

**Fig. 12** Index $i$ of DCT coefficients vs. variance $v_i$ for LE model.



(a) $T = 100$ epoch (b) $T = 1000$ epoch (c) $T = 10000$ epoch

**Fig. 13** Index $i$ of DCT coefficients vs. variance $v_i$ for IQ model.

**Table 4** BERs and PSNRs for LE model and IQ model.

| | Low-error oriented model | | | Image-quality oriented model | | |
|---|---|---|---|---|---|---|
| Epoch | $T = 100$ | $T = 1000$ | $T = 10000$ | $T = 100$ | $T = 1000$ | $T = 10000$ |
| BER | 0.000040 | 0.000076 | 0.000772 | 0.000478 | 0.001330 | 0.065791 |
| PSNR | 34.0 | 37.8 | 42.5 | 41.0 | 42.6 | 43.3 |

$$v_i = \frac{1}{M}\sum_{\mu=1}^{M}\left\|d_i^{\mu} - \bar{d}_i\right\|^2, \ i = 1, 2, \cdots, 14, \tag{31}$$

where $\bar{d}_i$ is the average of $d_i^{\mu}$, i.e.,

$$\bar{d}_i = \frac{1}{M}\sum_{\mu=1}^{M} d_i^{\mu}. \tag{32}$$

If the values of the DCT coefficients were changed by embedding, the variance $v_i$ would be large.

### 5.2 Analysis of Watermark Representation

Figures 12 and 13 show the variance $v_i$ for the LE model and the IQ model, respectively. The abscissa axis represents index $i$ of the DCT coefficients, and the ordinate axis represents the variance $v_i$. In both figures, (a) shows the variance at $T = 100$ epoch. Since many coefficients were modified by embedding, a watermark was widely represented in them. In both figures, (c) shows the variance at $T = 10000$ epoch. Since only a few coefficients were modified, the watermark was sparsely represented in them.

Table 4 lists the BERs and PSNRs for the LE model and IQ model. While the BERs were initially small and then gradually became large at 10,000 epoch, the image quality gradually became better. We found that the distributed representation could achieve robust watermarks and that the sparse representation could achieve a high-quality image. If the lowest BER rather than high image quality is required, training the network should be stopped at an early epoch.

## 6. Conclusion

We proposed embedder and extractor neural networks based on an autoencoder [36]. Once the proposed neural network is trained by using a training image, the network can embed and extract a watermark into any image. Therefore, the proposed neural network is a blind type embedding technique. We investigated the relation between number of neurons $m_2, m_4$ and BER and PSNR by computer simulations. Due to the trade-off between BER and image quality, we introduced two models: a low-error oriented model (LE model) and an image-quality oriented model (IQ model). The LE model is robust against compression, and the IQ model can achieve a high-quality image. By specifying the parameters $m_2, m_4$, and $\gamma$, the proposed neural network can control the quality of stego-images and bit error rate.

We also investigated the watermark representation in the stego-images and found that the distributed representation emerged at an early learning step and then sparse representation came out at a later learning step. This is because the error function (24) requires both a smaller difference between coefficients of the original image and those of the stego-image and a smaller difference between watermarks and the output of the network. It is possible to acquire both distributed and sparse representations by different learning steps.

## Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP16K00156. The computer simulations were carried out on PC clusters at Yamaguchi University.

## References

[1] I.J. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for images, audio and video," IEEE Int. Conf. Image Process., vol.3, pp.243–246, 1996.
[2] I.J. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," IEEE Trans. Image Process., vol.6, no.12, pp.1673–1687, Dec. 1997.
[3] J. Fridrich, "Combining low-frequency and spread-spectrum watermarking," Proc. SPIE, vol.3456, 1998.
[4] M. Barni, F. Bartolini, V. Cappellini, and A. Piva, "A DCT-domain system for robust image watermarking," Signal Processing, vol.66, no.3, pp.357–372, May 1998.
[5] S.D. Lin and C.F. Chen, "A robust DCT-based watermarking for copyright protection," IEEE Trans. Consum. Electron., vol.46, no.3, pp.415–421, Aug. 2000.
[6] D. Kundur and D. Hatzinakos, "Digital watermarking using multiresolution wavelet decomposition," Proc. IEEE Int. Conf. Acoustics, Speech Signal Process., 1998.
[7] M. Kuribayashi and H. Tanaka, "A new digital watermarking scheme applying locally the Wavelet transform," IEICE Trans. Fundamentals, vol.E84-A, no.10, pp.2500–2507, Oct. 2001.
[8] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vis., vol.60, no.2, pp.91–110, Nov. 2004.
[9] L. Verstrepen, T. Meesters, T. Dams, A. Dooms, and D. Bardyn, "Circular spatial improved watermark embedding using a new global SIFT synchronization scheme," 16th Int. Conf. Digital Signal Processing, pp.1–8, 2009.
[10] M. Kawamura and K. Uchida, "SIFT feature-based watermarking method aimed at achieving IHC ver.5," Advances in Intelligent Information Hiding and Multimedia Signal Processing, Computational Intelligence and Complexity, Springer, vol.81, pp.381–389, 2017.
[11] Information hiding and its criteria for evaluation, IEICE, http://www.ieice.org/iss/emm/ihc/en/ (accessed at Feb. 26, 2018).
[12] N. Teranishi and M. Kawamura, "Asynchronous stochastic decoder for spread spectrum digital watermarking," Proc. Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2011), 2011.
[13] T. Yamamoto and M. Kawamura, "Method of spread spectrum watermarking using quantization index modulation for cropped images," IEICE Trans. Inf. & Syst., vol.E98-D, no.7, pp.1306–1315, July 2015.
[14] V. Darmstaedter, J.F. Delaigle, J.J. Quisquater, and B. Macq, "Low cost spatial watermarking," Computers & Graphics, vol.22, no.4, pp.417–424, Aug. 1998.
[15] H.C. Huanga and W.C. Fang, "Metadata-based image watermarking for copyright protection," Simulation Modelling Practice and Theory, vol.18, no.4, pp.436–445, April 2010.
[16] A. Bastug and B. Sankur, "Improving the payload of watermarking channels via LDPC coding," IEEE Signal Process. Lett., vol.11, no.2, pp.90–92, Feb. 2004.
[17] N. Hirata and M. Kawamura, "Digital watermarking method using LDPC code for clipped image," Proc. 1st international workshop on Information hiding and its criteria for evaluation (IWIHC '14), pp.25–30, 2014.
[18] N. Hirata, T. Nozaki, and M. Kawamura, "Image Watermarking Method Satisfying IHC by Using PEG LDPC Code," IEICE Trans. Inf. & Syst., vol.E100-D, no.1, pp.13-23, Jan. 2017.
[19] F. Hartung and M. Kutter, "Multimedia watermarking techniques," Proc. IEEE, vol.87, no.7, pp.1079–1107, July 1999.
[20] R.G. van Schyndel, A.Z. Tirkel, and C.F. Osborne, "A digital watermark," Proc. IEEE Int. Conf. Image Process. (ICIP-94), 1994.
[21] H. Kii, J. Onishi, and S. Ozawa, "The digital watermarking method by using both patchwork and DCT," IEEE Int. Conf. Multimedia Computing Systems, vol.1, pp.895–899, 1999.
[22] I.K. Yeo and H.J. Kim, "Generalized patchwork algorithm for image watermarking," Multimedia Systems, vol.9, no.3, pp.261–265, Sept. 2003.
[23] J. Fridrich, M. Goljan, P. Lisoněk, and D. Soukal, "Writing on wet paper," IEEE Trans. Signal Process., vol.53, no.10, pp.3923–3935, Oct. 2005.
[24] J. Fridrich, M. Goljan, and D. Soukal, "Wet paper codes with improved embedding efficiency," IEEE Trans. Information Security and Forensics, vol.1, no.1, pp.102–110, March 2006.
[25] J. Fridrich and D. Soukal, "Matrix embedding for large payloads," IEEE Trans. Inf. Forensics Security, vol.1, no.3, pp.390–395, Sept. 2006.
[26] B. Chen and G.W. Wornell, "Quantization Index Modulation: A Class of Provably Good Methods for Digital Watermarking and Information Embedding," IEEE Trans. Inf. Theory, vol.47, no.4, pp.1423–1443, May 2001.
[27] L. Mao, Y.Y. Fan, H.Q. Wang, and G.Y. Lv, "Fractal and neural networks based watermark identification," Multimedia Tools and Applications, vol.52, no.1, pp.201–219, March 2011.
[28] M. Vafaei, H. Mahdavi-Nasab, and H. Pourghassem, "A new robust blind watermarking method based on neural networks in wavelet transform domain," World Applied Sciences Journal, vol.22, no.11, pp.1572–1580, 2013.
[29] B. Jagadeesh, P.R. Kumar, and P.C. Reddy, "Robust digital image watermarking based on fuzzy inference system and back propagation neural networks using DCT," Soft Computing, vol.20, no.9, pp.3679–3686, Sept. 2016.
[30] M.S. Hwang, C.C. Chang, and K.F. Hwang, "Digital watermarking of images using neural networks," J. Electronic Imaging, vol.9, no.4, pp.548–555, Oct. 2000.
[31] L.Y. Hsu and H.T. Hu, "Blind image watermarking via exploitation of inter-block prediction and visibility threshold in DCT domain," J. Visual Communication and Image Representation, vol.32, pp.130–143, Oct. 2015.
[32] J. Zhang, N.C. Wang, and F. Xiong, "A novel watermarking for images using neural networks," Proc. Int. Conf. Machine Learning and Cybernetics, vol.3, pp.1405–1408, 2002.
[33] X.B. Wen, H. Zhang, X. Xu, and J.J. Quan, "A new watermarking approach based on probabilistic neural network in wavelet domain," Soft Computing, vol.13, no.4, pp.355–360, Feb. 2009.
[34] A.N. Yahya, H.A. Jalab, A. Wahid, and R.M. Noor, "Robust watermarking algorithm for digital images using discrete wavelet and probabilistic neural network," J. King Saud University-Computer

and Information Sciences, vol.27, no.4, pp.393–401, Oct. 2015.
[35] G. Kulkarni and S. Kuri, "Robust digital image watermarking using DWT, DCT and probabilistic neural network," 2017 Int. Conf. Electrical, Electronics, Communication, Computer, and Optimization Techniques, pp.1–5, 2017.
[36] G. Cottrell, P. Munro, and D. Zipper, "Image Compression by Backpropagation: an example of extensional programming," Advances in Cognitive Science, vol.3, pp.208–240, 1988.
[37] V. Nair and G.E. Hinton, "Rectified linear units improve restricted Boltzmann machines," Proc. 27th Int. Conf. Mach. Learn., pp.807–814, 2015.
[38] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning representations by back-propagating errors," Nature, vol.323, no.6, pp.386–408, Oct. 1986.
[39] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a next-generation open source framework for deep learning," Proc. workshop on machine learning systems in the twenty-ninth annual conference on neural information processing systems, vol.5, 2015.
[40] D.P. Kingma and J.L. Ba, "Adam: A method for stochastic optimization," Proc. 3rd Int. Conf. Learning Representations, 2015.

**Ippei Hamamoto** received a B.S. degree from Yamaguchi University in 2017. Currently he is a master's degree student at Yamaguchi University's Graduate School of Sciences and Technology for Innovation. His research interests include neural networks and digital watermarking.

**Masaki Kawamura** received B.E., M.E., and Ph.D. degrees from the University of Tsukuba in 1994, 1996, and 1999. He joined Yamaguchi University as a research associate in 1999. Currently he is an associate professor there. His research interests include associative memory models and information hiding. He is a senior member of IEICE and a member of JNNS, JPS, and IEEE.