Journal of *Imaging*

Article

A Neural-Network-Based Watermarking Method Approximating JPEG Quantization

Shingo Yamauchi and Masaki Kawamura *

Graduate School of Sciences and Technology for Innovation, Yamaguchi University, Yamaguchi 753-8512, Japan; c028vbw@yamaguchi-u.ac.jp
* Correspondence: kawamura@sci.yamaguchi-u.ac.jp; Tel.: +81-83-933-5701

Abstract: We propose a neural-network-based watermarking method that introduces the quantized activation function that approximates the quantization of JPEG compression. Many neural-network-based watermarking methods have been proposed. Conventional methods have acquired robustness against various attacks by introducing an attack simulation layer between the embedding network and the extraction network. The quantization process of JPEG compression is replaced by the noise addition process in the attack layer of conventional methods. In this paper, we propose a quantized activation function that can simulate the JPEG quantization standard as it is in order to improve the robustness against the JPEG compression. Our quantized activation function consists of several hyperbolic tangent functions and is applied as an activation function for neural networks. Our network was introduced in the attack layer of ReDMark proposed by Ahmadi et al. to compare it with their method. That is, the embedding and extraction networks had the same structure. We compared the usual JPEG compressed images and the images applying the quantized activation function. The results showed that a network with quantized activation functions can approximate JPEG compression with high accuracy. We also compared the bit error rate (BER) of estimated watermarks generated by our network with those generated by ReDMark. We found that our network was able to produce estimated watermarks with lower BERs than those of ReDMark. Therefore, our network outperformed the conventional method with respect to image quality and BER.

Keywords: watermarking method; neural network; activation function; JPEG compression

check for updates

Citation: Yamauchi, S.; Kawamura, M. A Neural-Network-Based Watermarking Method Approximating JPEG Quantization. *J. Imaging* **2024**, *10*, 138. https://doi.org/10.3390/jimaging10060138

Academic Editor: Alessandro Piva

Received: 14 May 2024
Revised: 30 May 2024
Accepted: 4 June 2024
Published: 6 June 2024

1. Introduction

People are now easily able to upload photos and illustrations to the Internet, owing to smartphones and personal computers. To protect content creators, we need to prevent unauthorized copying and other abuses because digital content is not degraded by copying or transmissions. Digital watermarking is effective against such unauthorized use.

In digital watermarking, secret information is embedded in digital content by making slight changes to the content. In the case of an image, the image in which the information is embedded is called a stego-image, and the embedded information is called a digital watermark. There are two types of digital watermarking: blind and non-blind. The blind method does not require the original image to extract the watermark from the stego-image. However, the non-blind method requires the original image when extracting a watermark from a stego-image. Therefore, the blind method is more practical. In addition, because stego-images may be attacked by various kinds of image processing, watermarking methods must have the ability to extract watermarks from degraded stego-images. Two types of attacks on stego-images can occur: geometric attacks such as rotation, scaling, and cropping, and non-geometric attacks such as noise addition and JPEG compression [1].

Neural-network-based methods have been proposed. In single-stage training, where the embedding and extraction are performed in a single network, the network has been trained to output a watermark from an input image [2,3]. The overall performance of the network is low because the relationship between the image and the watermark is trained individually. To improve performance, watermarking methods using autoencoders (AE) have been proposed [4–6]. The input layer to the middle layer is called the embedding network, and the middle to the output layer is called the extraction network. Both the original image and the watermark are input into the input layer of the AE, and the identity mapping is learned to retrieve them in the output layer. The stego-image is extracted from the middle layer [6]. Since the original image is unnecessary during extraction, it is often omitted to output only the watermark. Furthermore, AE with convolutional neural networks has been proposed [7]. An adversarial network has also been added to improve image quality [8]. DARI-Mark [9] is a DNN-based watermarking method using attention to determine the embedding regions. It can find non-significant regions that are insensitive to the human eye and increases robustness by embedding the watermark with larger intensities. Thus, end-to-end models were proposed [6–11]. However, a huge training dataset was needed to train the connections as the network became more complex. Although data augmentation was sometimes introduced, a model with internal networks mimicking attacks was proposed in order to train on a relatively small training dataset [8,10,11].

HiDDeN [8], proposed by Zhu et al., has an attack layer that simulates attacks such as Gaussian blur, per-pixel dropout, cropping, and JPEG compression attacks on images during training. Here, the implementation of JPEG compression is approximated by JPEG-Mask, which sets the high-frequency components of the discrete cosine transform (DCT) coefficients to zero, and JPEG-Drop, which uses progressive dropout to eliminate the high-frequency components of the DCT coefficients. Therefore, this implementation does not meet the standard for quantization in the JPEG compression process. It has also been noted that the JPEG-Mask and JPEG-Drop layers of HiDDeN do not provide sufficient performance for the robustness of the JPEG compression [12,13]. JPEGdiff is a method of approximating around the quantized values in JPEG compression by a cubic function. Hamamoto and Kawamura's method [10] also introduces a layer of additive white Gaussian noise as an attack layer to improve robustness against JPEG compression. Moreover, ReDMark proposed by Ahmadi et al. [11] has attack layers implementing salt-and-pepper noise, Gaussian noise, JPEG compression, and mean smoothing filters. The quantization of the JPEG compression is approximated by adding uniform noise. As described, the quantization process has been replaced by the process of adding noise, and the quantization process as per the JPEG standard has not been introduced.

Adversarial samples are a problem in the field of pattern recognition. They are generated by adding distortions to images to misclassify them. To avoid misclassification, a pattern recognition method using JPEG compressed images has been proposed [14]. JPEG compression is expected to effectively reduce noise while preserving the information needed for pattern recognition. However, JPEGdiff has been proposed as a way to break this technique [12]. By approximating the JPEG quantization with a differentiable function, a JPEG-resistant adversarial image can be generated. Therefore, approximating the JPEG quantization with a smooth function may affect the performance of the model.

In our previous work [15], we proposed a quantization activation function (QAF) that can simulate the quantization of JPEG compression according to a standard. That model consists of a network that introduces the QAF into the AE-based model proposed by Hamamoto and Kawamura [6]. Better performance was obtained in terms of JPEG compression robustness than the AE-based model [6]. The effectiveness of the QAF has been demonstrated in our previous work. However, in that model, a QAF with a constant quantization width was used instead of the quantization table. In this paper, we apply the QAF to the attack layer of the ReDMark [11], which is a CNN-based model, rather than an AE-based model. Furthermore, the proposed method uses the quantization table-based QAF. The robustness against the JPEG compression is expected to be improved using the QAF. The effectiveness of our method is evaluated by comparing JPEG-compressed images with QAF-applied images. The image quality of the stego-image is also evaluated.

The rest of the paper is organized as follows. In Section 2, the process of JPEG quantization is explained. In Section 3, we describe the ReDMark and, in addition, we address our previous work. In Section 4, we define the quantized activation function and describe the structure of the proposed network. In Section 5, we show the effectiveness of the function and demonstrate the performance of our network in computer simulations. The last section concludes the paper.

2. Preliminary: JPEG Quantization

JPEG compression is a lossy compression that reduces the amount of information in an image to reduce the file size. In this kind of compression, an image is divided into 8×8-pixel blocks. Then, in each block, the processes of the DCT, quantization, and entropy coding are sequentially performed. In JPEG compression, the process of reducing the amount of information is the quantization process of the DCT coefficients. We focus on the quantization of the DCT coefficients of the luminance component in an image because the watermark is embedded in these coefficients of the image. Figure 1 shows the quantization process in JPEG compression for DCT coefficients [16]. The process consists of three steps: (1) the creation of the quantization table $\boldsymbol{T}_Q$, (2) the quantization process, and (3) the dequantization process.

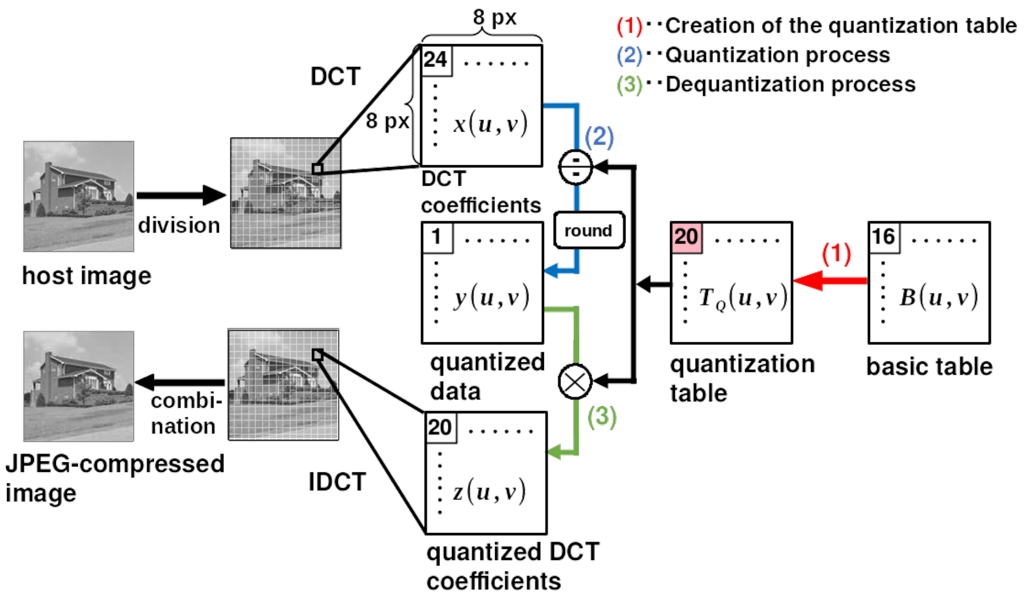


Figure 1. JPEG compression quantization for luminance components. (1) Creation of the quantization table $\boldsymbol{T}_Q$, (2) the quantization process, and (3) the dequantization process.

During the quantization process, the DCT coefficients are quantized based on a default basic table or a self-defined basic table. The default basic table $\boldsymbol{B}$ is defined as

$$\boldsymbol{B} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}. \tag{1}$$

The quantization table is then determined using the quality factor (Q) and the basic table $\boldsymbol{B}$. The quantization table $T_Q(u,v)$ for the quantization level Q at coordinates $(u,v), u = 0, 1 \cdots, 7, v = 0, 1 \cdots, 7$ is defined as

$$T_Q(u,v) = \left\lfloor \frac{B(u,v)s(Q)+50}{100} \right\rfloor, \tag{2}$$

where $\lfloor \cdot \rfloor$ is the floor function and where $B(u,v)$ is the (u,v) component of the basic table $\boldsymbol{B}$. Also, the scaling factor $s(Q)$ is given by

$$s(Q) = \begin{cases} \frac{5000}{Q} & (Q < 50) \\ 200 - 2Q & (Q \geq 50) \end{cases}. \tag{3}$$

The quantization process is performed using the quantization table $\boldsymbol{T}_Q$. Let $y(u,v)$ be the quantized data, and let $x(u,v)$ be the DCT coefficients in an 8×8-pixel block. The quantization process is performed as

$$y(u,v) = \mathrm{round}\left(\frac{x(u,v)}{T_Q(u,v)}\right), \tag{4}$$

where

$$\mathrm{round}(a) = \begin{cases} \lfloor a + 0.5 \rfloor & (a \geq 0) \\ -\lfloor -a + 0.5 \rfloor & (a < 0) \end{cases}. \tag{5}$$

Let $z(u,v)$ be the quantized DCT coefficients; then, the dequantization process is performed as

$$z(u,v) = y(u,v)T_Q(u,v). \tag{6}$$

3. Related Works

3.1. ReDMark

Figure 2 shows the overall structure of ReDMark [11], which consists of an embedding network, an extraction network, and an attack layer. The h and w are the height and width of the 2D watermark, and H and W are the height and width of the original and stego-images. The images are divided into $M \times N$-pixel blocks, where $M = N = 8$ as it is in ReDMark. The process flow during training is illustrated by the red arrows in Figure 2. The host image and watermark are fed to the embedding network, and the attack layer degrades the generated stego-image. By feeding the degraded image to the extraction network, the extraction network learns to extract the watermark from the degraded image. After training, the embedding and extraction networks are used individually. The process flow during testing is illustrated by the blue arrows in Figure 2. A stego-image is generated by the embedding network. This image is published and attacked. Let us assume that the attack is to compress the image by some JPEG tool. When the attacked image is obtained, the watermark is extracted from the image in the extraction network.

In ReDMark [11], normalization and reshaping are performed on the input image in preprocessing. For the input image $I_{\mathrm{in}}(i,j), i = 0,1,\cdots,H-1, j = 0,1,\cdots,W-1$, the normalized image is given by

$$I(i,j) = \frac{I_{\mathrm{in}}(i,j) - 128}{255}. \tag{7}$$

Reshape is an operation that divides an image into $M \times N$-pixel blocks and transforms them into a 3D tensor representation. The image size $H \times W$ is assumed to satisfy $H = hM, W = wN$. The reshaped image is represented by a three-dimensional tensor of $h \times w \times MN$. This image is called the image tensor of size (h, w, MN). If necessary, the tensor is inverse transformed back to its original dimension.

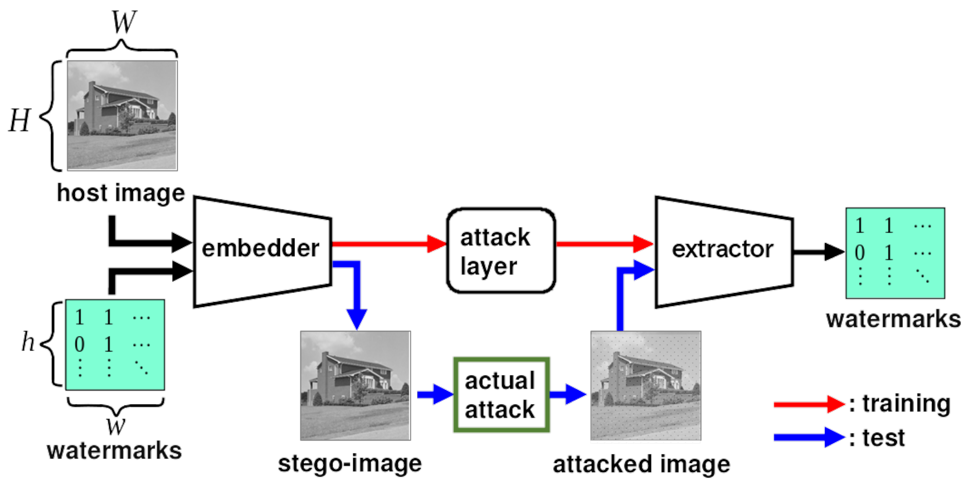


Figure 2. Overall structure of ReDMark [11]. The red and blue arrows represent the process flow during training and testing, respectively.

3.1.1. Embedding Network

The embedding network, as shown in Figure 3, consists of three layers: convolution, circular convolution, and transform. The transform layer can perform lossless linear transforms using 1×1 convolutional layers, e.g., the DCT, wavelet transform, and Hadamard transform. In our method, the DCT is selected as the transform layer, as it is in ReDMark. The circular convolution layer extends the input to make it cyclic before the convolution is performed. Figure 4 shows an example of applying a circular convolution layer with a 2×2 filter when the input is 3×3 pixels. When a circular convolution layer is used, the dimension of the output after convolution is the same as the dimension of the input.

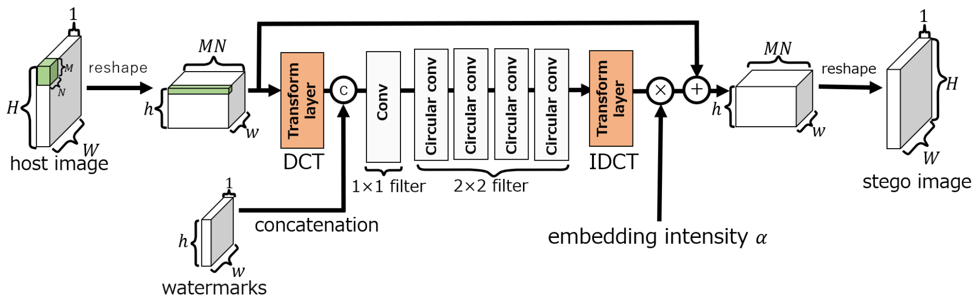


Figure 3. Embedding network of ReDMark [11].

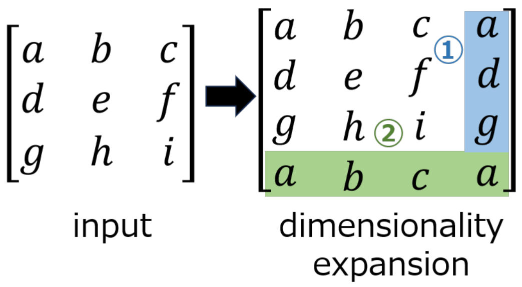


Figure 4. Extended input in a circular convolution layer. ① Extended in the column direction. ② Extended in the row direction.

In the embedding network, the convolution and circular convolution layers use 1×1 and 2×2 filters, respectively, and both have 64 filters. In each layer, an exponential linear unit (ELU) [17] activation function is used. The output of the embedding network is obtained by summing the output of the transform layer performing the inverse DCT (IDCT) (for the IDCT layer) with the input image tensor of size (h, w, MN) and then by performing the inverse process of reshaping. Here, the output of the IDCT layer can be adjusted by the embedding intensity α. The intensity is fixed as $\alpha = 1$ during training and can be changed during an evaluation.

3.1.2. Extraction Network

The extraction network consists of a convolution layer, a circular convolution layer, and a transform layer as shown in Figure 5. The transform layer of the extraction network also performs the DCT. The filter sizes of the convolutional and circular convolutional layers are 1×1 and 2×2, respectively. The number of filters is 64 for the fourth layer and 1 for the fifth layer. The activation function up to the fourth layer is ELU [17], and a sigmoid function is used in the fifth layer. Let $p_o(i, j)$ be the output of the extraction network, and the estimated watermark $p_e(i, j)$ is given by

$$p_e(i,j) = \begin{cases} 1, & p_o(i,j) > 0.5 \\ 0, & p_o(i,j) \le 0.5 \end{cases}. \tag{8}$$

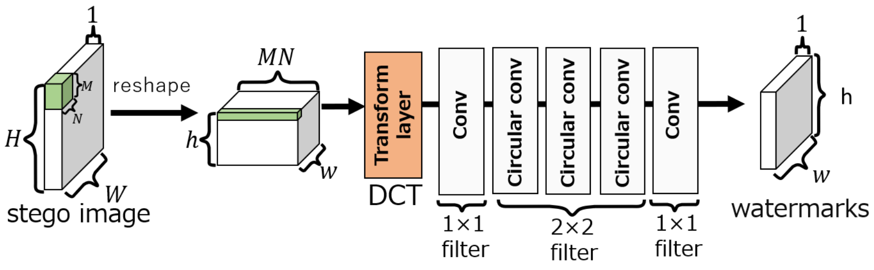


Figure 5. Extraction network of ReDMark [11].

3.1.3. Attack Layer

The attack layer lies between the embedding and the extraction networks and operates when ReDMark is trained. The attack layer itself is not trained. By simulating possible attacks on the stego-image and feeding the attacked image to the extraction network, the network can be trained to extract the watermark from the degraded image. Various attacks can be simulated in the attack layer. In the attack layer of ReDMark [11], three networks were implemented according to the type of attacks: a GT-Net (Gaussian-trained network), a JT-Net (JPEG-trained network), and a MT-Net (multi-attack-trained network).

In ReDMark, the quantization process is approximated using the quantization table $T_Q(u, v)$ and uniform noise ϵ. Let $x(u, v)$ represent the DCT coefficients of an 8×8-pixel block of a stego-image, and the quantized DCT coefficients $z(u, v)$ are given as

$$\begin{aligned} z(u,v) &= \left(\frac{x(u,v)}{T_Q(u,v)} + \epsilon \right) T_Q(u,v) \quad (9) \\ &= x(u,v) + T_Q(u,v)\epsilon, \quad (10) \end{aligned}$$

where ϵ represents noise subject to a uniform distribution in the interval $[-0.5, 0.5]$. In other words, the quantization process is equivalent to adding a uniform noise ϵ proportional to the quantization table $T_Q(u, v)$.

3.2. JPEGdiff

In the quantization of JPEG compression, the DCT coefficient values are converted to integers. This causes a problem that the activation function cannot be differentiated when training a neural network. For example, the JPEG-Mask and JPEG-Drop layers of HiDDeN do not provide sufficient performance against robustness for JPEG compression. Therefore, the JPEGdiff, which approximates the activation function to a cubic function around the quantized value, has been proposed [12,13]. This approximation can reduce the number of non-differentiable points and reduce the number of regions with a zero gradient. As a result, performance is improved by training. The JPEGdiff is given by

$$\text{JPEGdiff}(x) = \text{round}(x) + (x - \text{round}(x))^3. \tag{11}$$

Note that this function still has non-differentiable points on the boundaries of the intervals.

3.3. Previous Work

In our previous work [15], we proposed a quantized activation function (QAF). This function was a functional representation of the quantization process of JPEG compression. Specifically, the function $\text{QAF}(x)$ consists of several hyperbolic tangent functions and returns the quantized value of the argument x. The proposed QAF was applied to the attack layer of the AE-based model proposed by Hamamoto and Kawamura [6]. Moreover, all DCT coefficients $x(u,v)$ were quantized with the same intensity. In other words, the value of the quantization table $T_Q(u,v)$ was constant as given by

$$T_Q(u,v) \quad = \quad \delta, \tag{12}$$

where δ is a constant. Even with a constant quantization table, the previous method had a certain level of tolerance to JPEG compression. However, if the original values of the quantization table could be applied, the tolerance could be enhanced.

4. Proposed Method

We propose a quantization table-based QAF, and apply it to the attack layer of the ReDMark [11], which is a CNN-based model rather than an AE-based model. To demonstrate the effectiveness of the QAF, we compare the proposed layer using the QAF with the JT-Net of ReDMark [11]. The embedding and extraction networks of the proposed method are the same as those of ReDMark.

4.1. Quantized Activation Function

We propose a quantization table-based QAF for neural networks to implement the quantization process of the JPEG compression according to the standard. The QAF consists of n hyperbolic tangent (tanh) functions and is defined as

$$\text{QAF}(x|t_Q) = \sum_{i=0}^{n} \frac{t_Q}{2} \tanh\left\{\beta\left(x \pm t_Q\left(\frac{1}{2} + i\right)\right)\right\}, \tag{13}$$

where t_Q is the value of the quantization table $T_Q(u,v)$ when the quantization level is Q. The parameters n and β denote the number and slope of the tanh functions, respectively. The red dashed line in Figure 6 represents $\text{QAF}(x|16)$ when the slope $\beta = 1000$ and the value of the quantization table $T_Q(u,v) = 16$. In addition, the values of the DCT coefficients after JPEG quantization are plotted with a black line. We can see that they are almost the same.

Quantization is essentially a rounding operation to integer values. It should therefore be represented by a discontinuous, step-like function. In other words, a sign function should be used for the representation of JPEG quantization (13). However, the tanh function was used in the proposed method. When training a neural network, the differentiable function works better for training. Therefore, we chose the tanh function, which is a continuous function. Figure 7a shows the tanh functions for different slopes $\beta = 1, 10, 100, 1000$. We can see that the slope becomes steeper and asymptotically closer to the sign function as β increases. When the slope of tanh is set to $\beta \to \infty$, it asymptotically approaches the sign function. For practical use, a large value of β can approximate the quantization with sufficient accuracy.

Let us see how the quantization with QAF differs from the quantization with JPEGdiff. Figure 7b shows a comparison of JPEGdiff (11) with QAF functions. The green curve represents the JPEGdiff, and the blue and orange curves represent the QAF at slopes $\beta = 10, 1000$. We can see that the JPEGdiff has discontinuities, while the QAF is smooth. Since the QAF has no discontinuities, the network may be better trained.

Finally, we consider the number n of tanh functions. The number n depends on the value of $T_Q(u,v)$. If the minimum value of $T_Q(u,v) = 1$, because the maximum value of the DCT coefficients is 2040, then at most, 2039 tanh functions are required. A sufficiently large constant n is chosen because it does not matter if the possible values of the QAF exceed the maximum value of the DCT coefficients.

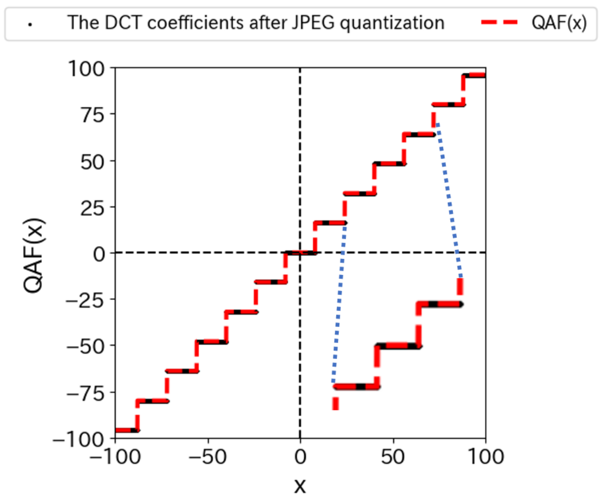
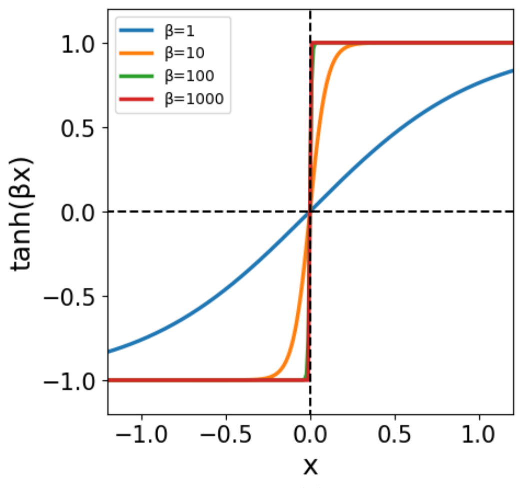
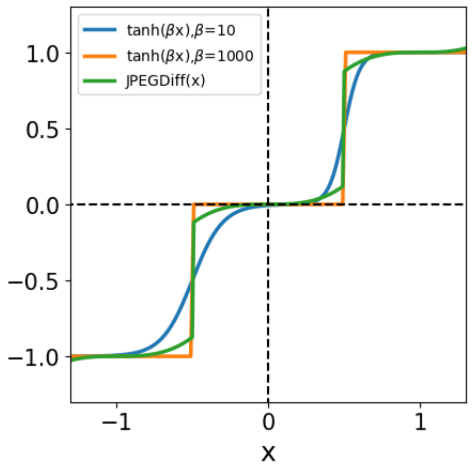


Figure 6. Overview of the quantized activation function: QAF$(x|16)$ with slope $\beta = 1000$ and number of hyperbolic tangent functions, $n = 500$. The blue dotted line indicates that the function has been expanded in this range.



(**a**)



(**b**)

Figure 7. Activation functions: (**a**) the hyperbolic tangent functions for slopes $\beta = 1, 10, 100, 1000$. (**b**) JPEGdiff vs. QAF.

4.2. Proposed Attack Layer

The proposed attack layer consists of three sublayers: a DCT layer, a layer introducing QAF (QAF layer), and an IDCT layer. Note that in our network, the DCT coefficients are quantized as in the JPEG compression. Figure 8 shows the structure of the network. The embedding and extraction networks have the same structure as that of ReDMark [11]. The output $O(i,j,k), i = 0,1,\cdots,h-1,\ j = 0,1,\cdots,w-1,\ k = 0,1,\cdots,MN-1$ of the DCT layer is processed by the QAF (13) at the QAF layer. The output $z(i,j,k)$ of this layer is calculated by

$$z(i,j,k) = \mathrm{QAF}\left(O(i,j,k)\middle|\frac{T_Q(u,v)}{255}\right). \tag{14}$$

Note that the quantization table $T_Q(u,v)$ is divided by 255 because it is normalized by (7). The quantization table values in (14) are determined according to $T_Q(u,v)$ because the quantization table values are different for each of the coordinates (u,v) of the DCT coefficients. The coordinates (u,v) are defined by

$$u = \left\lfloor \frac{k}{8} \right\rfloor,\ v = k - 8\left\lfloor \frac{k}{8} \right\rfloor. \tag{15}$$

The attack layer [11] in ReDMark performs noise addition to the coefficients, while the one in our network performs the quantization with the QAF.

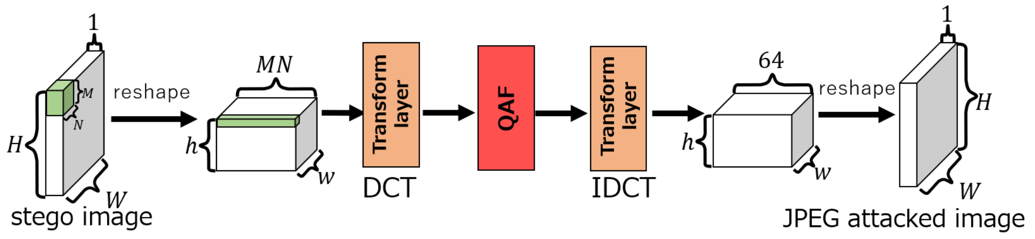


Figure 8. Proposed attack layer.

4.3. Training Method

The embedding and extraction networks are trained in the same way as they were in ReDMark [11], respectively. The loss function L_1 of the embedding network is defined as

$$L_1 = 1 - \mathrm{SSIM}(\boldsymbol{I}, \boldsymbol{I}_o), \tag{16}$$

where SSIM represents the structural similarity function (SSIM) [18], which measures the structural similarity between two images. The closer it is to 1.0, the larger the similarity between the two images [18]. It is defined in

$$\mathrm{SSIM}(\boldsymbol{I}, \boldsymbol{I}_o) = \frac{(2\mu\mu_o + c_1)(2\mathrm{Cov}(\boldsymbol{I}, \boldsymbol{I}_o) + c_2)}{(\mu^2\mu_o^2 + c_1)(\sigma^2\sigma_o^2 + c_2)}, \tag{17}$$

where $\boldsymbol{I}$ is the original image given as a teacher and where $\boldsymbol{I}_o$ is the output of the embedding network. μ and μ_o are the means of $\boldsymbol{I}$ and $\boldsymbol{I}_o$, respectively. σ and σ_o are the variances of each image, and $\mathrm{Cov}(\boldsymbol{I}, \boldsymbol{I}_o)$ represents the covariance between the two images. Let c_1 and c_2 be constants, and let $c_1 = 10^{-4}, c_2 = 9 \times 10^{-4}$. Because the output of the embedding network takes a real number, the stego-image $\boldsymbol{I}_{\mathrm{st}}$ is obtained by converting it back to 256 levels of the pixel value. That is, it is given by

$$I_{\mathrm{st}}(i,j) = \begin{cases} 255, & I_{\mathrm{o}}(i,j) > 0.5 \\ \lfloor 255\ I_{\mathrm{o}}(i,j) + 128 \rfloor. & -0.5 \leq I_{\mathrm{o}}(i,j) \leq 0.5\,. \\ 0, & I_{\mathrm{o}}(i,j) < -0.5 \end{cases} \tag{18}$$

The loss function L_2 of the extraction network is defined as

$$L_2 \quad = \quad -\sum_{i=0}^{h-1}\sum_{j=0}^{w-1}\{p(i,j)\log(p_o(i,j)) + (1-p(i,j))\log(1-p_o(i,j))\}, \tag{19}$$

where $\boldsymbol{p}$ is the watermark used as a teacher for the extraction network. $\boldsymbol{p}_o$ is the output of the sigmoid function of the extraction network. That is, the value of the element takes a value between 0 and 1. The total loss function L of our network is defined as

$$L = \gamma L_1 + (1-\gamma)L_2, \tag{20}$$

where the parameter γ determines the balance between the two loss functions. The embedding and extraction network are trained by the back propagation [19] using stochastic gradient descent (SGD) as the optimization method.

5. Computer Simulations

5.1. Evaluation of the QAF

First, the ability that the QAF function can approximate the quantization process of JPEG compression was assessed, using computer simulations. Because the DCT and quantization were applied to 8×8-pixel blocks in the JPEG compression, the QAF was also applied to 8×8-pixel blocks. The evaluation images were taken from a dataset provided by the University of Granada [20]. They consist of 49 images of 512×512 pixels. Each image was normalized by (7). The 512×512-pixel image is divided into blocks of 8×8 pixels, resulting in 64×64 blocks. These blocks were indexed in raster scan order as $\mu = 1, 2, 3, \cdots, 4096$. The DCT was performed on each block. Let $I_b^\mu(u,v), \mu = 1, 2, 3, \cdots, 4096$ be the DCT coefficients of the μ-th block. The QAF was applied to the μ-th block as

$$I_{\mathrm{QAF}}^\mu(i,j) = \mathrm{QAF}\left(I_b^\mu(i,j) \,\middle|\, \frac{T_{70}(i,j)}{255} \right), \tag{21}$$

where the quantization level of the JPEG compression was set to $Q = 70$. The parameters of the QAF in (13) were set as gradient $\beta = 1000$, and the number of the hyperbolic tangent functions $n = 500$. For all the QAF-applied blocks, an IDCT was performed, and the luminance values were inversely normalized using (18). Next, all blocks were combined. The combined image was converted back to the original image size. Then, the QAF-applied image was given by

$$\boldsymbol{I}_{\mathrm{QAF}} = \left(\boldsymbol{I}_{\mathrm{QAF}}^1, \boldsymbol{I}_{\mathrm{QAF}}^2, \cdots, \boldsymbol{I}_{\mathrm{QAF}}^{4096} \right). \tag{22}$$

In general, the peak signal-to-noise ratio (PSNR) of an evaluated image $\boldsymbol{I}'$ against a reference image $\boldsymbol{I}$ is defined by

$$\mathrm{PSNR}(\boldsymbol{I}' | \boldsymbol{I}) = 10 \log_{10} \left(\frac{255^2}{\mathrm{MSE}(\boldsymbol{I}', \boldsymbol{I})} \right) [\mathrm{dB}], \tag{23}$$

where

$$\mathrm{MSE}(\boldsymbol{I}', \boldsymbol{I}) = \frac{1}{HW} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \left\{ I'(i,j) - I(i,j) \right\}^2. \tag{24}$$

To see the difference between the QAF and JPEG-compressed images shown in Figure 6, the difference could be evaluated by PSNR rather than MSE. The accuracy of the QAF could be measured by the PSNR of a QAF-applied image against a JPEG-compressed image, that is, $\mathrm{PSNR}(\boldsymbol{I}_{\mathrm{QAF}} | \boldsymbol{I}_{\mathrm{jpeg}})$, where $\boldsymbol{I}_{\mathrm{jpeg}}$ is the JPEG-compressed image. Note that the PSNR was measured against the JPEG-compressed image, not the original image. Similarly, we evaluated the approximation ability of JT-Net using PSNR and compared it with QAF. Figure 9 shows a histogram of PSNRs for JT-Net-applied images by using (9) and QAF-applied images given by (21) and (22), where the quantization level is $Q = 70$. The PSNRs for QAF-applied images are clearly greater than those for JT-Net-applied images. Figure 10 shows three examples of QAF-applied images and their PSNRs. Thus, we found that the QAF more adequately represents the quantization of the JPEG compression.

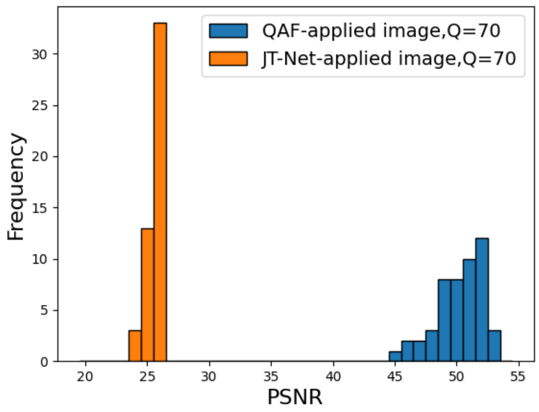
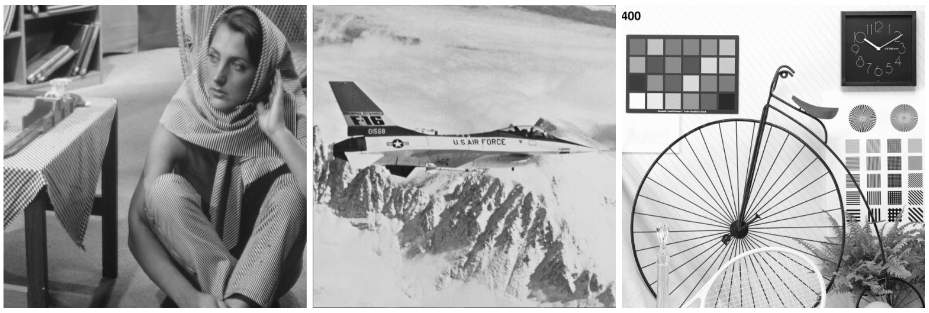


Figure 9. Histogram of PSNRs for JT-Net-applied images and QAF-applied images.



(**a**) PSNR = 51.52 dB. (**b**) PSNR = 53.12 dB. (**c**) PSNR = 49.55 dB.

Figure 10. QAF-applied images and their PSNRs.

5.2. Evaluation of the Proposed Attack Layer

We compared the JT-Net in the ReDMark, our previous model, and the proposed quantization table-based QAF network (QT-QAF-Net) based on the image quality of stego-images and the BER of watermarks extracted from stego-images after the JPEG compression. Note that our previous model [15] is the AE-based model with the quantization table quantized by constant intensity. For comparison, we used an improved model of the CNN-based QAF network with a constant intensity quantization table (constant QAF-Net). The only difference between the QT-QAF-Net and the constant QAF-Net is the values of the quantization table.

5.2.1. Experimental Conditions

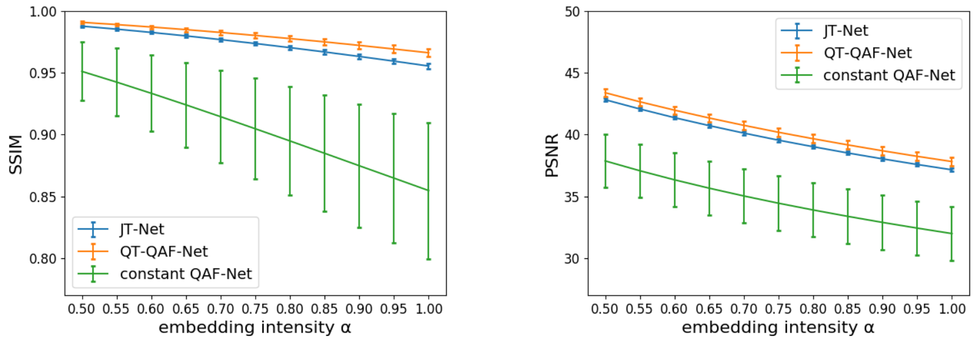
The training and test images were selected as they were in ReDMark [11]. The training images were 50,000 images of 32×32 pixels from CIFAR10 [21] ($H = 32, W = 32$). An $h \times w$-bit watermark was embedded, where $h = 4, w = 4$. The watermark was randomly generated. Therefore, the amount of watermark embedded per pixel was approximately 0.0156 bits per pixel. In the parameters used for training, the block height and width sizes were set to $M = 8$ and $N = 8$, respectively. The parameter of the loss function (20) was set to $\gamma = 0.75$, the number of learning epochs was set to 100, and the mini-batch size was set to 32. For the parameters of SGD, the training rate was set to 10^{-4}, and the moment was set to 0.98. For training the proposed attack layer, the gradient of the QAF (13) was set to $\beta = 1000$, and the number of the hyperbolic tangent functions was set to $n = 500$. In the attack layer of JT-Net and the proposed method, the quantization level was set to $Q = 70$, and the quantization table T_{70} was used. The embedding, attack, and extraction networks were all connected, and the network was trained using the training images and watermarks. Here, the embedding intensity was fixed at $\alpha = 1$.

For testing, 49 images of 512×512 pixels from the University of Granada were used. These images were divided into 32×32 pixel subimages, and the embedding process was performed on each of them. The 256 subimages were given to the network as test images. Meanwhile, a 32×32-bit watermark was randomly generated. One watermark was embedded four times in one image. That is, the watermark was divided into 4×4-bit subwatermarks, and finally each subwatermark was embedded in one subimage. An estimated watermark was determined by bit-by-bit majority voting because the same watermark was embedded four times in one image.

As stated in Section 3.1, the attack layer was not used during testing. The test images and the watermarks were used to output stego-images in the embedding network. Here, the embedding intensity was set to values from $\alpha = 0.5$ to 1.0. The stego-image is published. Subsequently, we assumed that it was JPEG-compressed by some JPEG tool with quantization levels $Q = 10, 20, \cdots, 90$. The compressed stego-images were input to the extraction network, and the estimated watermarks were output. These networks were trained 10 times with different initial weights for the comparison of our network with the JT-Net on image quality and BER. The mean and standard deviation of structural similarity index measures (SSIMs), PSNRs, and BERs were calculated.

5.2.2. Evaluation of the Image Quality

The image quality of the stego-images obtained from the JT-Net, the constant QAF-Net and QT-QAF-Net was evaluated using the SSIMs and PSNRs. The image quality of the stego-image $\boldsymbol{I}_{st}$ against the original image $\boldsymbol{I}$ can be expressed as $\mathrm{SSIM}(\boldsymbol{I}_{st}, \boldsymbol{I})$ by (17) and $\mathrm{PSNR}(\boldsymbol{I}_{st}|\boldsymbol{I})$ by (23). Figure 11 shows the SSIM and PSNR. The horizontal and vertical axes represent the embedding intensity α and SSIM or PSNR, respectively. The error bars represent the standard deviation of the SSIMs and PSNRs. Embedding a watermark strongly causes degradation in image quality. Therefore, the SSIM and PSNR decreased as the intensity α increased. The image quality of the proposed QT-QAF-Net was higher than that of the other two networks. In other words, our network can reduce the degradation of image quality even with the same embedding intensity.



(**a**) SSIM (**b**) PSNR

Figure 11. Image quality of stego-images (tained by $Q = 70$).

As the images were processed block by block, they were visually checked for block artifacts. Three images selected from the dataset were cropped to 128×128-pixel size as shown in Figure 12. These images were generated from the proposed network trained with embedding intensity $\alpha = 1.0$. The images were not reduced in size when displayed. Few noticeable artifacts were observed.

Figure 12. Images cropped to 128 × 128 pixels.

5.2.3. Evaluation of the BER

The robustness of our network against the JPEG compression was evaluated. The estimated watermark obtained by (8) was evaluated by using the BER. The BER of the estimated watermark $\boldsymbol{p}_e$ can be defined by

$$\mathrm{BER} = \frac{1}{hw}\sum_{i=0}^{h-1}\sum_{j=0}^{w-1} p(i,j) \oplus p_e(i,j), \tag{25}$$

where $\boldsymbol{p}$ is the original watermark, and $\oplus$ represents the exclusive OR.

First, we compared the robustness of our network with that of the JT-Net and constant QAF-Net using the same embedding intensity α. Figure 13 shows the BER of the estimated watermark for the embedding intensity α. The horizontal and vertical axes represent the intensity α and BER, respectively. For different compression levels Q, the dashed lines with circles represent the BER for the QT-QAF-Net, the solid lines with squares represent the BER for the JT-Net, and the dotted lines with triangles represent the BER for the constant QAF-Net. When the watermark was strongly embedded, it was extracted correctly. Therefore, the BER decreased as the intensity α increased. The lowest BER was obtained when $\alpha = 1.0$. At compression level $Q \leq 70$, the BER of the proposed network was lower than that of the JT-Net. Also, at $Q = 80$, they had almost the same BER. Furthermore, at $Q = 90$, the BER of our network was larger than that of the JT-Net. The BER of the constant QAF-Net was always larger than that of the other two networks.

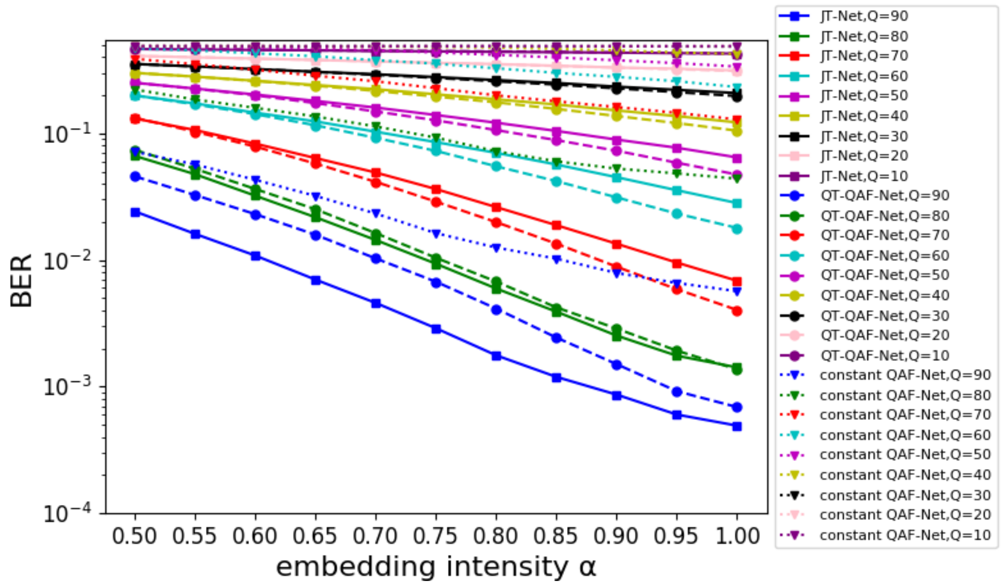


Figure 13. BER for embedding intensity α.

Even with the same embedding intensity, the image quality of our network differs from that of the JT-Net and the constant QAF-Net. Therefore, we next adjusted the embedding intensity so that the PSNRs of these three networks were approximately the same, and we compared the BERs of the networks under this condition. Figure 14 shows the histograms of PSNRs for the QT-QAF-Net with embedding intensity $\alpha = 1.0$. Figure 14a shows the histograms for the embedding intensity $\alpha = 1.0$ for the JT-Net and $\alpha = 0.55$ for the constant QAF-Net, respectively. Figure 14b shows histograms for these embedding intensities $\alpha = 0.95$ and $\alpha = 0.50$, respectively. The intensities of the three networks were chosen so that the histograms shown look similar. To measure the robustness against JPEG compression, we set the intensity α to ensure that the PSNR obtained from these networks is approximately the same. Specifically, we set the intensity for the JT-Net and QT-QAF-Net to $\alpha = 0.95$ (average PSNR = 37.58 dB) and $\alpha = 1.0$ (average PSNR = 37.81 dB), respectively. Figure 15 is the BER for the compression level Q. The error bars are the standard deviation of the BERs. The BER of the estimated watermark for QT-QAF-Net is lower than that for the JT-Net and constant QAF-Net. Thus, we can say that our network can generate watermarks with fewer errors under the given PSNR.

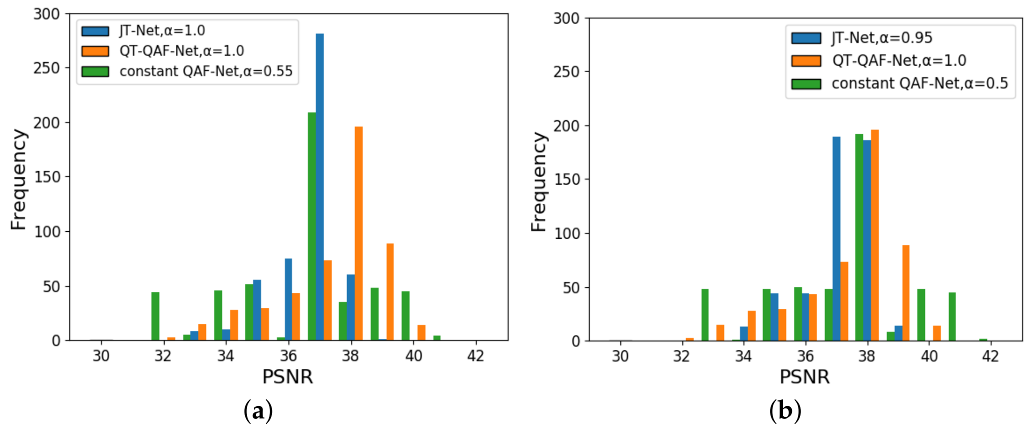


Figure 14. Histogram of PSNR for embedding intensity $\alpha = 1.0$ for QT-QAF-Net: (**a**) intensity $\alpha = 1.0$ for JT-Net and $\alpha = 0.55$ for constant QAF-Net and (**b**) intensity $\alpha = 0.95$ for JT-Net and intensity for constant QAF-Net $\alpha = 0.50$.

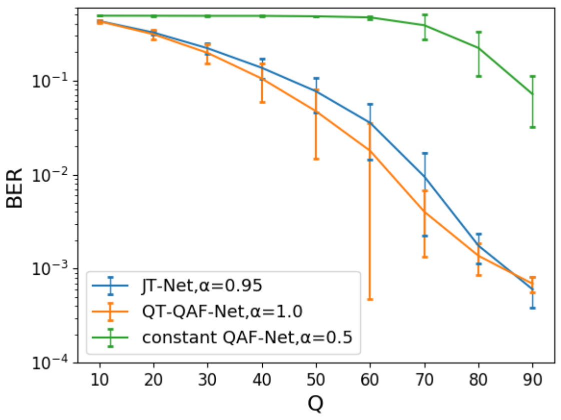


Figure 15. BER for the JPEG compression level Q.

6. Conclusions

The JT-Net in ReDMark [11] is a network that simulates JPEG compression. This network substitutes the quantization process with a process that adds noise proportional to the value of the quantization table. In previous work [15], a quantized activation function (QAF) quantized by constant intensity (constant QAF-Net) was proposed. In this paper, we proposed the quantization table-based QAF network (QT-QAF-Net), which can approximate the quantization process of the JPEG compression according to the standard. By approximating the quantization of the JPEG compression using the QAF, we expected to improve the robustness against the JPEG compression. The results of computer simulations showed that the QAF represented quantization with sufficient accuracy. Also, we found that the network trained with the QAF was more robust against the JPEG compression than those trained with the JT-Net. Because the embedding and extraction networks were more robust against the JPEG compression when trained with the QAF, we conclude that our method is more suitable for simulating JPEG compression than conventional methods applying additive noise.

Further studies with QAF are expected. For example, since QAF is differentiable over the whole interval, it may produce better adversarial images compared to JPEGdiff [12]. Furthermore, there is a study on the estimation of the sign bit of DCT coefficients [22]. The non-linearity of the quantized DCT coefficients makes estimation difficult. The solution could be simplified by using QAF.

Author Contributions: Conceptualization, M.K.; methodology, M.K. and S.Y.; software, S.Y.; investigation, S.Y.; resources, M.K.; data curation, M.K. and S.Y.; writing—original draft preparation, S.Y.; writing—review and editing, M.K.; visualization, S.Y.; supervision, M.K.; project administration, M.K.; funding acquisition, M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by JSPS KAKENHI Grant Number 20K11973 and Support Center for Advanced Telecommunications Technology (SCAT) research grant.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. These data can be found here: CIFAR-10 dataset https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 3 June 2024) and Computer Vision Group. University of Granada https://ccia.ugr.es/cvg/CG/base.htm (accessed on 3 June 2024).

Acknowledgments: The computations were performed using the supercomputer facilities at the Research Institute for Information Technology of Kyushu University.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Wan, W.; Wang, J.; Zhang, Y.; Li, J.; Yu, H.; Sun, J. A comprehensive survey on robust image watermarking. *Neurocomputing* **2022**, *488*, 226–247. [CrossRef]
2. Vafaei, M.; Mahdavi-Nasab, H.; Pourghassem, H. A new robust blind watermarking method based on neural networks in wavelet transform domain. *World Appl. Sci. J.* **2013**, *22*, 1572–1580.
3. Sy, N.C.; Kha, H.H.; Hoang, N.M. An efficient robust blind watermarking method based on convolution neural networks in wavelet transform domain. *Int. J. Mach. Learn. Comput.* **2020**, *10*, 675–684. [CrossRef]
4. He, D.; Xu, K.; Wang, D. Design of multi-scale receptive field convolutional neural network for surface inspection of hot rolled steels. *Image Vis. Comput.* **2019**, *89*, 12–20. [CrossRef]
5. Singh, H.K.; Singh, A.K. Digital image watermarking using deep learning. *Multimed. Tools Appl.* **2024**, *83*, 2979–2994. [CrossRef]
6. Hamamoto, I.; Kawamura, M. Image watermarking technique using embedder and extractor neural networks. *IEICE Trans. Inf. Syst.* **2019**, *E102-D*, 19–30. [CrossRef]
7. Jamali, M.; Karimi, N.; Khadivi, P.; Shirani, S.; Samavi, S. Robust watermarking using diffusion of logo into auto-encoder feature maps. *Multimed. Tools Appl.* **2023**, *82*, 45175–45201. [CrossRef]
8. Zhu, J.; Kaplan, R.; Johnson, J.; Fei-Fei, L. HiDDeN: Hiding data with deep networks. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; Part XV, pp. 682–697.
9. Zhao, Y.; Wang, C.; Zhou, X.; Qin, Z. DARI-Mark: Deep learning and attention network for robust image watermarking. *Mathematics* **2023**, *11*, 209. [CrossRef]
10. Hamamoto, I.; Kawamura, M. Neural watermarking method including an attack simulator against rotation and compression attacks. *IEICE Trans. Inf. Syst.* **2020**, *E103-D*, 33–41. [CrossRef]

11. Ahmadi, M.; Norouzi, A.; Soroushmehr, S.M.R.; Karimi, N.; Najarian, K.; Samavi, S.; Emami, A. Redmark: Framework for residual diffusion watermarking based on deep networks. *Expert Syst. Appl.* **2020**, *146*, 113157. [CrossRef]
12. Shin, R.; Song, D. JPEG-resistant adversarial images. In Proceedings of the NIPS 2017 Workshop on Machine Learning and Computer Security, Long Beach, CA, USA, 4–9 December 2017; Volume 1, pp. 8–13.
13. Mareen, H.; Antchougov, L.; Wallendael, G.V.; Lambert, P. Blind deep-learning-based image watermarking robust against geometric transformations. In Proceedings of the 2024 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 5–8 January 2024; pp. 1–2.
14. Guo, C.; Rana, M.; Cisse, M.; Maaten, L.v. Countering adversarial images using input transformations. *arXiv* **2018**, arXiv:1711.00117.
15. Yamauchi, S.; Kawamura, M. Neural network based watermarking trained with quantized activation function. In Proceedings of the Asia Pacific Signal and Information Processing Association Annual Summit and Conference, Chiang Mai, Thailand, 7–10 November 2022; pp. 1608–1613.
16. Independent JPEG Group. Available online: http://www.ijg.org/ (accessed on 21 May 2023).
17. Clevert, D.; Unterthiner, T.; Hochreiter, S. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv* **2015**, arXiv:1511.07289.
18. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef]
19. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
20. Computer Vision Group at the University of Granada, Dataset of Standard 512 × 512 Grayscale Test Images. Available online: http://decsai.ugr.es/cvg/CG/base.htm (accessed on 9 June 2023).
21. Krizhevsky, A.; Nair, V.; Hinton, G. The CIFAR-10 Dataset. Available online: https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 9 June 2023).
22. Lin, R.; Liu, S.; Jiang, J.; Li, S.; Li, C.; Kuo, C.-C.J. Recovering sign bits of DCT coefficients in digital images as an optimization problem. *J. Vis. Commun. Image Represent.* **2024**, *98*, 104045. [CrossRef]