Two-Stage Watermark Removal Framework for Spread Spectrum Watermarking

Jinkun You, *Graduate Student Member, IEEE*, and Yicong Zhou, *Senior Member, IEEE*

***Abstract*—Spread spectrum (SS) watermarking has gained significant attention as it prevents attackers from reading, tampering with, or removing watermarks. Secret key estimation can help with the first two unauthorized operations but cannot remove watermarks. Moreover, existing deep-learning watermark removal methods do not consider the characteristics of SS watermarking, thus leading to unsatisfactory results. In this paper, we design a secret key estimation method that treats secret key estimation as a binary classification problem and updates the estimated key via backpropagation and parameter optimization algorithms. We develop a watermark removal network using quaternion convolutional neural networks (QCNNs) to learn watermark features while capturing the relationship between channels to improve image quality. Based on our estimation method and QCNN-based network, we propose a two-stage watermark removal framework that utilizes information of the secret key to train the network. A loss function is introduced to directly prevent watermark extraction, thereby improving removal performance. Extensive experiments demonstrate the superiority of our methods over the state-of-the-art methods.**

***Index Terms*—Deep learning, secret key estimation, spread spectrum watermarking, watermark removal, watermarking security.**

I. Introduction

DIGITAL watermarks are messages embedded in multimedia files to protect the copyright of the owners [1], [2], [3]. Numerous watermarking methods have been proposed, each with unique advantages. For instance, object-based watermarking can embed messages into selected regions of a video sequence, providing robustness against geometrical attacks [4], [5]. On the other hand, SS watermarking embeds watermarks across the entire data. Specifically, a watermark is spread into a secret key and then added to the data. In this way, the watermark exhibits robustness against common signal processing operations and security of preventing unauthorized operations. Due to these advantages, SS watermarking has been widely applied to image, audio, and video watermarking [6], [7], [8]. Recently, researchers have focused on improving the security of SS watermarking [9], [10], [11]. A secure SS watermarking method can prevent malicious individuals from reading, tampering with, or removing the embedded watermarks [12]. Hence, it is crucial to study the security of SS watermarking to ensure the protection of intellectual property rights.

Security in SS watermarking is primarily studied from the attackers' perspective. Since SS watermarking uses the same secret key for both watermark embedding and extraction, attackers can read or tamper with embedded watermarks if they can accurately estimate the secret key. Hence, secret key estimation has emerged as an attack method based on Kerckhoffs' principle [13] and Diffie-Hellman methodology [14]. According to Kerckhoffs' principle, attackers should fully know the watermark embedding and extraction algorithms. As suggested by [14], attackers usually collect additional information from the owner to design estimation methods. This information consists of watermarked files and embedded watermarks in the known-message attack (KMA) scenario, which has gained significant attention recently [15], [16], [17]. However, only a few estimation methods have been proposed. The watermarked file exhibits a larger variance along the secret key direction. Cayer et al. [18] devised a FastICA-based method to determine the direction. This poses a threat to traditional SS watermarking methods, including additive SS watermarking [19] and improved SS (ISS) watermarking [20]. The distribution of watermarked files must be circular to conceal the direction information. This principle has inspired the development of more secure SS watermarking methods, such as robust-natural watermarking (RNW) [21], circular-extension of ISS (CW-ISS) [22], and transportation spherical watermarking (TSW) [23]. In [18] and [24], maximum likelihood estimation (MLE) was adopted to achieve higher estimation accuracy. Bas et al. [25] proposed an average operation (AVE) on watermarked files and embedded watermarks to estimate the secret key. Additionally, You et al. [17] introduced the equivalent key-based (EK)[1] method, which achieves the highest estimation accuracy at the cost of efficiency. Existing secret key estimation methods have contributed to security evaluation and improvement. Nevertheless, they have not yet been applied to watermark removal. This is because watermark extraction does not rely on certain parameters used in watermark embedding, making these parameters unknown to attackers. As a result, SS watermarking remains irreversible and prevents watermark removal. To solve

Manuscript received 11 September 2023; revised 28 January 2024; accepted 23 February 2024. Date of publication 27 February 2024; date of current version 24 April 2024. This work was supported in part by Science and Technology Development Fund, Macau SAR under Grant 0049/2022/A1, and in part by the University of Macau under Grant MYRG2022-00072-FST. The Associate Editor coordinating the review of this manuscript and approving it for publication was Prof. Xiaochun Cao. *(Corresponding author: Yicong Zhou.)*

The authors are with the Department of Computer and Information Science, University of Macau, Macau 999078, China (e-mail: youjinkun09@gmail.com; yicongzhou@um.edu.mo).

Digital Object Identifier 10.1109/TMM.2024.3370380

[1]Equivalent keys refer to unit vectors with expected decoding performance.

this problem, the deep learning techniques can be introduced to learn these unknown parameters.

In recent years, the deep learning techniques have been applied to remove watermarks from images while improving image quality [26], [27], [28], [29]. Specifically, a deep neural network (DNN) is trained on pairs of original and watermarked images. In [26], Geng et al. exploited the denoising convolutional neural network (DnCNN) [30] to remove watermarks. Multiple convolutional neural networks (CNNs) are stacked to learn the difference between the original and watermarked images. Inspired by U-Net [31], Hatoum et al. [27] designed a U-shaped network for watermark removal. Later, Li et al. [28] proposed a U-shaped watermark removal network based on generative adversarial training. Wang et al. [29] utilized the residual dense network (RDN) [32] to remove low-frequency and middle-frequency watermarks. Existing watermark removal networks can attack SS watermarking but do not consider its characteristics. First, SS watermarking embeds and extracts watermarks using a secret key that is unavailable to attackers in real-world applications. Hence, existing methods directly train networks on the image pairs created by a randomly generated key. Nevertheless, different keys embed watermarks in different subspaces, making it difficult to learn watermark features from these image pairs. A reasonable solution is to exploit more information about the secret key for network training. Second, the watermarks embedded in the frequency domain can distort various frequency coefficients. Existing methods use CNN-based networks to learn watermark features in the spatial domain but fail to explore the relationships between frequency coefficients. The capture of the relationships can improve the image quality. Third, existing methods remove watermarks only through image quality enhancement. The attacked image is expected to be visually similar to the original one. However, the watermarks are embedded imperceptibly and will cause the distortion that may be perceived as an inherent part of the original image. Consequently, it is difficult to remove watermarks via image quality enhancement. A more effective way is to directly increase the watermark extraction failure probability.

Motivated by the above observations, this paper proposes a two-stage watermark removal framework for SS watermarking and the KMA scenario. Since watermarked images and the embedded watermarks provide valuable information about the secret key, our framework utilizes them to learn watermark features. Our contributions are five-fold as follows:

- We propose a secret key estimation method by transforming secret key estimation into a binary classification problem. The estimated key corresponds to the predicted decision boundary and is updated by backpropagation and parameter optimization algorithms.
- We develop a QCNN-based watermark removal network to learn watermark features, capturing the relationships between different frequency information.
- Combining the proposed estimation method and QCNN-based network, we propose a two-stage watermark removal framework for SS watermarking. Our framework utilizes the information provided by secret key estimation to learn watermark features, filling the gap between secret key estimation and watermark removal.
- We introduce a loss function to directly decrease the success probability of watermark extraction, thus enhancing watermark removal performance.
- Extensive experiments and analysis demonstrate that our methods outperform the state-of-the-art methods in terms of estimation accuracy, removal rates, and image quality.

The rest of this paper is organized as follows. Section II briefly introduces SS watermarking and QCNNs. Section III presents the proposed methods in details. Section IV reports experimental results. Finally, Section V draws conclusions.

II. Background

In this section, SS watermarking is briefly reviewed. QCNNs are introduced. Finally, some measures are given to evaluate the performance of secret key estimation and watermark removal methods.

A. SS Watermarking

Notations: In this paper, $\mathbb{R}$ and $[N]$ denote the real space and integer set $\{1, 2, \ldots, N\}$, respectively; $\mathbf{m} \in \{0,1\}^{N_c \times 1}$ and $\hat{\mathbf{m}} \in \{0,1\}^{N_c \times 1}$ denote the original and extracted watermark messages, respectively; $\mathbf{I}_o \in \mathbb{R}^{H \times W}$ and $\mathbf{I}_w \in \mathbb{R}^{H \times W}$ denote the original and watermarked images, respectively; $\mathbf{x} \in \mathbb{R}^{N_v \times 1}$, $\mathbf{y} \in \mathbb{R}^{N_v \times 1}$, and $\mathbf{w} \in \mathbb{R}^{N_v \times 1}$ denote the host, watermarked, and watermark signals, respectively; the secret key is denoted as $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{N_c}]$, where $\mathbf{u}_i \in \mathbb{R}^{N_v \times 1}$ for $\forall i \in [N_c]$; $\hat{\mathbf{U}} = [\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \ldots, \hat{\mathbf{u}}_{N_c}] \in \mathbb{R}^{N_v \times N_c}$ represents the estimated key. Additionally, $\mathbf{U}$ has the following orthogonal property

$$\mathbf{u}_i^T \mathbf{u}_j = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{otherwise} \end{cases}. \tag{1}$$

Hence, $\mathbf{U}$ is a set of orthogonal bases with unit length in the space $\mathbb{R}^{N_c}$ and is in the subspace of $\mathbb{R}^{N_v}$ ($N_v \gg N_c$). Due to the orthogonal property, SS watermarking uses the same secret key to embed and extract watermark messages.

Embedding: To embed $\mathbf{m}$ into $\mathbf{I}_o$, $\mathbf{I}_o$ is first transformed into the frequency domain using techniques like the discrete wavelet transform (DWT). Afterwards, N_v coefficients are selected to form $\mathbf{x}$. Next, $\mathbf{w}$ is generated and added to $\mathbf{x}$ as follows:

$$\mathbf{y} = \mathbf{x} + \mathbf{w}. \tag{2}$$

$\mathbf{I}_w$ is finally obtained from $\mathbf{y}$ via the inverse transformation. The generation method of $\mathbf{w}$ determines the robustness and security of SS watermarking. In [19], Cox et al. proposed the first SS embedding method, namely additive SS. Additive SS generates $\mathbf{w}$ by

$$\mathbf{w} = \alpha \sum_{i=1}^{N_c} (-1)^{\mathbf{m}(i)} \mathbf{u}_i, \tag{3}$$

where $\alpha \in (0, +\infty)$ controls embedding strength. Additive SS suffers from the interference of $\mathbf{x}$, thus causing a decrease in robustness. To alleviate this problem, Malvar et al. [20] proposed

ISS as follows:

$$\mathbf{w} = \sum_{i=1}^{N_c} (\alpha(-1)^{\mathbf{m}(i)} - \lambda \mathbf{x}^T \mathbf{u}_i)\mathbf{u}_i, \tag{4}$$

where $\lambda \in [0, 1]$ tunes the attenuation of the host signal interference. In [22] and [21], Bas et al. designed two more secure SS embedding methods, i.e., CW-ISS and RNW. CW-ISS can be expressed as

$$\mathbf{w} = \sum_{i=1}^{N_c} (\alpha(-1)^{\mathbf{m}(i)} |\mathbf{v}(i)| - \lambda \mathbf{x}^T \mathbf{u}_i)\mathbf{u}_i, \tag{5}$$

where $\mathbf{v} \in \mathbb{R}^{N_c \times 1}$ is a randomly-generated vector with unit length. RNW can be implemented by

$$\mathbf{w} = \sum_{i=1}^{N_c} (\alpha(-1)^{\mathbf{m}(i)} |\mathbf{x}^T \mathbf{u}_i| - \mathbf{x}^T \mathbf{u}_i)\mathbf{u}_i. \tag{6}$$

Compared with additive SS and ISS, CW-ISS and RNW are more secure but less robust. To maintain robustness and security simultaneously, Wang et al. [23] proposed TSW. This method exploits the transportation theory to optimize the spherical embedding like

$$\mathbf{w} = \sum_{i=1}^{N_c} (\alpha(-1)^{\mathbf{m}(i)} |\mathbf{v}(i)| - \mathbf{x}^T \mathbf{u}_i)\mathbf{u}_i. \tag{7}$$

As a result, the robustness is significantly improved.

Extraction: $\mathbf{I}_w$ is transformed into the frequency domain to obtain $\mathbf{y}$ through the same process as in the watermark embedding. Subsequently, the watermark message is extracted from $\mathbf{y}$ by the following function

$$\hat{\mathbf{m}}(i) = \left(1 - \text{sgn}\left(\mathbf{y}^T \mathbf{u}_i\right)\right)/2, \tag{8}$$

where $\text{sgn}(\cdot)$ represents the sign function. As can be seen, the extracted message $\hat{\mathbf{m}}$ is determined by the sign of the product between $\mathbf{y}$ and $\mathbf{U}$.

B. Quaternion Convolutional Neural Network

QCNNs draw inspiration from quaternion algebra and CNNs. Quaternion algebra introduces the concept of quaternion numbers and defines a set of operations within the hypercomplex space [33]. A quaternion number is mathematically expressed as

$$\dot{q} = r + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}, \tag{9}$$

where r, x, y, and z are real numbers; $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the imaginary units satisfying the following properties: 1) $\mathbf{ijk} = \mathbf{ii} = \mathbf{jj} = \mathbf{kk} = -1$; 2) $\mathbf{ij} = -\mathbf{ji} = \mathbf{k}$; 3) $\mathbf{jk} = -\mathbf{kj} = \mathbf{i}$; 4) $\mathbf{ki} = -\mathbf{ik} = \mathbf{j}$. Given a scalar α, we have $\alpha\dot{q} = \alpha r + \alpha x\mathbf{i} + \alpha y\mathbf{j} + \alpha z\mathbf{k}$. Given any two quaternion numbers $\dot{q}_1 = r_1 + x_1\mathbf{i} + y_1\mathbf{j} + z_1\mathbf{k}$ and $\dot{q}_2 = r_2 + x_2\mathbf{i} + y_2\mathbf{j} + z_2\mathbf{k}$, the addition operation is defined as

$$\dot{q}_1 + \dot{q}_2 = (r_1 + r_2) + (x_1 + x_2)\mathbf{i} + (y_1 + y_2)\mathbf{j} + (z_1 + z_2)\mathbf{z}. \tag{10}$$

The rule for quaternion addition also applies to the subtraction operation. Besides, the quaternion algebra system defines the Hamilton product between $\dot{q}_1$ and $\dot{q}_2$ as

$$\begin{aligned} \dot{q}_1 \otimes \dot{q}_2 &= \begin{bmatrix} r_1 & -x_1 & -y_1 & -z_1 \\ x_1 & r_1 & -z_1 & y_1 \\ y_1 & z_1 & r_1 & -x_1 \\ z_1 & -y_1 & x_1 & r_1 \end{bmatrix} * \begin{bmatrix} r_2 \\ x_2 \\ y_2 \\ z_2 \end{bmatrix} \\ &= \dot{Q}_1 * \dot{q}_2 \\ &= \dot{q}_3 \\ &= \begin{bmatrix} r_3 & x_3 & y_3 & z_3 \end{bmatrix}^T, \end{aligned} \tag{11}$$

where "*" denotes the matrix multiplication; $\dot{q}_3 = r_3 + x_3\mathbf{i} + y_3\mathbf{j} + z_3\mathbf{k}$ is the output; $\dot{Q}_1$ is a matrix generated from $\dot{q}_1$. QCNNs follow the structure of CNNs but introduce several key differences. First, QCNNs use the Hamilton product to perform the convolution operation. Second, $\dot{Q}_1$ is considered the kernel in QCNNs. Third, r_2, x_2, y_2, and z_2 are treated as four distinct channels. Accordingly, the same weights are shared by every four channels, reducing the number of parameters. Finally, QCNNs require that the number of channels must be a multiple of three or four.

C. Evaluation Metrics

The normalized correlation (NC) is used to assess the estimation accuracy. It is calculated by

$$\text{NC} = \frac{\mathbf{u}^T \hat{\mathbf{u}}}{\|\mathbf{u}\| \cdot \|\hat{\mathbf{u}}\|}, \tag{12}$$

where $\mathbf{u}$ is the real secret key, $\hat{\mathbf{u}}$ is the estimated key, and $\|\cdot\|$ denotes the Euclidean norm. A larger NC value indicates higher accuracy. Watermark removal performance is evaluated by the removal rate (RR). It is calculated based on the bit error rate (BER) as follows:

$$\begin{aligned} \text{RR} &= 1 - 2\left|\text{BER} - 0.5\right|, \\ \text{BER} &= \frac{1}{N_c} \sum_{i=1}^{N_c} \mathbf{m}(i) \oplus \hat{\mathbf{m}}(i), \end{aligned} \tag{13}$$

where $\hat{\mathbf{m}}$ is the message extracted from the attacked image; "$\oplus$" and $|\cdot|$ denote the XOR and absolute value operations, respectively. A larger RR value indicates better watermark removal performance. Besides, image quality is quantitatively evaluated by the peak signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM). The PSNR is calculated by

$$\text{PSNR}_{[\text{dB}]} = 10\log_{10}\left(\frac{M^2 \cdot HW}{\sum_{i=1}^{H}\sum_{j=1}^{W} \|\mathbf{I}(i,j) - \mathbf{I}'(i,j)\|^2}\right), \tag{14}$$

where $\mathbf{I}$ and $\mathbf{I}'$ are images of size $H \times W$; M is the maximum possible pixel value of the image. The SSIM is calculated by

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{15}$$

where x and y are patches from two images; μ_x and μ_y denote the means of x and y, respectively; σ_x and σ_y represent the standard variances of x and y, respectively; σ_{xy} represents the

covariance of x and y; C_1 and C_2 are two constants. a larger value for both PSNR and SSIM indicates better image quality.

III. Proposed Method

Fig. 1 presents the proposed watermark removal framework for SS watermarking in the KMA scenario. As can be seen, the proposed framework has two stages with forward and backward branches. In the forward branch of Stage I, watermarked signals are extracted from the owner's watermarked images. These watermarked signals are normalized, multiplied with the estimated key, and then fed into the sigmoid-based decoder to extract the embedded messages. In the backward branch of Stage I, the extracted and original messages are used to calculate the MSE loss. Afterward, an optimizer updates the estimated key based on the loss value. In Stage II, new original images are sampled from a public dataset. These images and the estimated key from Stage I are used to generate new watermarked images, which are then used to train the proposed QCNN-based network. Besides, the estimated key is used to extract watermarks from the attacked images to calculate the RR loss assisted with learning watermark features.

A. Stage I: Secret Key Estimation

Secret key estimation aims to obtain an estimated key $\hat{\mathbf{U}}$ that closely approximates the real secret key $\mathbf{U}$. The watermark extraction function in (8) uses $\mathbf{U}$ as a decision boundary to classify each extracted watermark bit as "0" or "1". Therefore, the estimation problem can be transformed into a classification problem, where the estimated key $\hat{\mathbf{U}}$ corresponds to the predicted decision boundary. Better classification performance implies higher estimation accuracy. In this case, the watermark extracted by $\hat{\mathbf{U}}$ is more similar to the original one. Moreover, $\hat{\mathbf{U}}$ can be seen as trainable parameters and iteratively optimized by backpropagation and parameter optimization algorithms. Based on this analysis, we propose an estimation method, as illustrated in the upper part of Fig. 1. First, the watermarked signal $\mathbf{y}$ is extracted from the watermarked image $\mathbf{I}_w$. Then, a decoder extracts the watermark $\mathbf{m}$ from $\mathbf{y}$ by

$$\hat{\mathbf{m}} = \text{Decoder}(\mathbf{y}, \hat{\mathbf{U}}), \tag{16}$$

where Decoder$(\cdot)$ represents a watermark extraction function. Next, the backpropagation algorithm uses $\hat{\mathbf{m}}$ and $\mathbf{m}$ to calculate the gradients with respect to $\hat{\mathbf{U}}$

$$\nabla\hat{\mathbf{U}} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{U}}}, \tag{17}$$

where $\mathcal{L}$ is a loss function of $\mathbf{m}$ and $\hat{\mathbf{m}}$. Finally, an optimization algorithm updates $\hat{\mathbf{U}}$ via

$$\hat{\mathbf{U}} = \text{Optimizer}(\hat{\mathbf{U}}, \nabla\hat{\mathbf{U}}), \tag{18}$$

where Optimizer$(\cdot,\cdot)$ can be implemented by various optimization algorithms, including stochastic gradient descent and Adam algorithms [34]. The optimization algorithm minimizes the loss function $\mathcal{L}$ that is parameterized by $\hat{\mathbf{U}}$ and uses $\nabla\hat{\mathbf{U}}$ to update $\hat{\mathbf{U}}$ via a carefully-crafted strategy. Since each column of $\mathbf{U}$ is a unit vector, each column of $\hat{\mathbf{U}}$ is further normalized to have unit length.

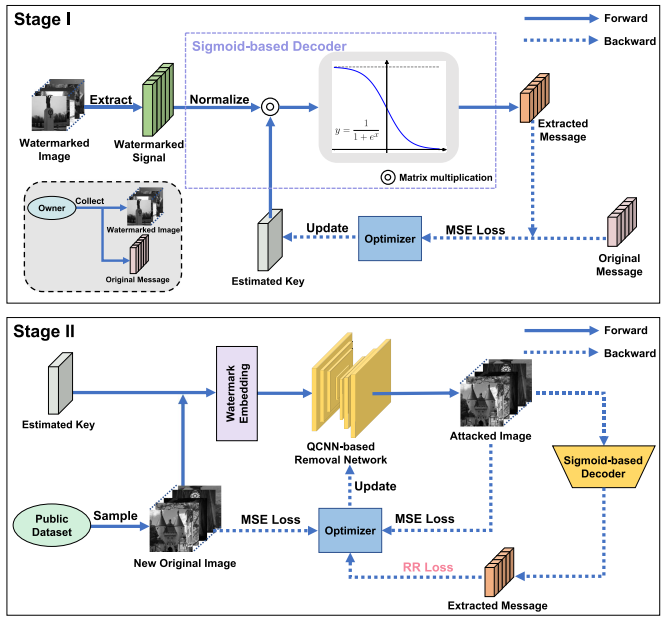

Fig. 1. Illustration of the proposed two-stage watermark removal framework for SS watermarking. The upper and lower parts show the flowcharts of Stage I and Stage II, respectively.

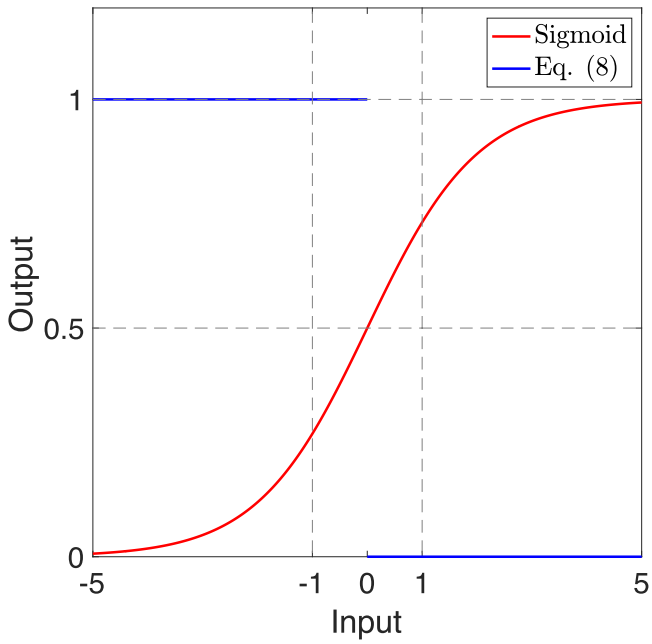

Fig. 2. Graph of different functions. The red line represents the sigmoid function, while the blue line represents the original watermark extraction function in (8).

Usually, the watermark extraction function in (8) is employed as the decoder in (16). Nevertheless, this function poses a challenge for the proposed estimation method due to its zero-gradient issue. Fig. 2 presents the graph of (8). As observed, the gradient of this function is always zero. Consequently, $\nabla\hat{\mathbf{U}}$ in (18)

becomes a matrix with all its elements equal to zero, preventing the update of $\hat{\mathbf{U}}$. To circumvent this problem, we propose an alternative watermark extraction function based on the sigmoid function:

$$\hat{\mathbf{m}}(i) = \text{Sigmoid}(-\mathbf{y}^T\hat{\mathbf{u}}_i), \tag{19}$$

where $\text{Sigmoid}(\cdot)$ represents the sigmoid function. Fig. 2 shows the graph of the sigmoid function. Its maximum and minimum possible values are close to 1 and 0, respectively, similar to the range of values in (8). Additionally, its gradient is not always zero, ensuring that $\nabla\hat{\mathbf{U}}$ contains non-zero values. These properties enable the sigmoid-based decoder to overcome the zero-gradient issue. From (8), $\hat{\mathbf{m}}(i)$ is 1 if $-\mathbf{y}^T\hat{\mathbf{u}}_i$ is positive and is 0 if $-\mathbf{y}^T\hat{\mathbf{u}}_i$ is negative. The sigmoid function of (19) maps $-\mathbf{y}^T\hat{\mathbf{u}}_i$ into the open interval $(0, 1)$. Its output is greater than 0.5 when $-\mathbf{y}^T\hat{\mathbf{u}}_i$ is positive, where $\hat{\mathbf{m}}(i)$ should be 1. On the other hand, $\hat{\mathbf{m}}(i)$ should be 0 when the output is smaller than 0.5. Moreover, since $\hat{\mathbf{u}}_i$ is a unit vector, $-\mathbf{y}^T\hat{\mathbf{u}}_i$ can be rewritten as

$$-\mathbf{y}^T\hat{\mathbf{u}}_i = -||\mathbf{y}|| \cdot \frac{\mathbf{y}^T\hat{\mathbf{u}}_i}{||\mathbf{y}|| \cdot ||\hat{\mathbf{u}}_i||} = -||\mathbf{y}|| \cos\theta_i, \tag{20}$$

where θ_i denotes the angle between $\mathbf{y}$ and $\hat{\mathbf{u}}_i$. Subsequently, $-\mathbf{y}^T\hat{\mathbf{u}}_i$ falls within the interval $[-||\mathbf{y}||, ||\mathbf{y}||]$. When $||\mathbf{y}||$ is very large, the gradient of (19) approaches zero. This can hinder the update of $\hat{\mathbf{U}}$ during secret key estimation. To address this issue, the sigmoid-based decoder is modified as

$$\hat{\mathbf{m}}(i) = \text{Sigmoid}(-\mathbf{y}^T\hat{\mathbf{u}}_i/||\mathbf{y}||). \tag{21}$$

Then, we have

$$-\frac{\mathbf{y}^T\hat{\mathbf{u}}_i}{||\mathbf{y}||} = -\frac{\mathbf{y}^T\hat{\mathbf{u}}_i}{||\mathbf{y}|| \cdot ||\hat{\mathbf{u}}_i||} = -\cos\theta_i. \tag{22}$$

Accordingly, $-\mathbf{y}^T\hat{\mathbf{u}}_i/||\mathbf{y}||$ falls within the interval $[-1, 1]$, ensuring that the gradient remains non-zero even for large values of $||\mathbf{y}||$.

The loss function $\mathcal{L}$ in (17) should measure the difference between $\mathbf{m}$ and $\hat{\mathbf{m}}$. In this work, we employ the mean squared error (MSE) as the loss function.

B. Stage II: Watermark Removal

Stage II aims to train a network capable of removing the watermark $\mathbf{m}$ embedded in the watermarked image $\mathbf{I}_w$ while preserving the visual quality of the original image $\mathbf{I}_o$. However, watermark removal is challenging due to several reasons. Firstly, watermarking is imperceptible. The visual difference between $\mathbf{I}_w$ and $\mathbf{I}_o$ is often subtle. This makes it difficult for the network to learn the visual cues that distinguish the two images. Secondly, $\mathbf{m}$ has good robustness. Even if $\mathbf{I}_w$ is distorted, $\mathbf{m}$ can still be successfully extracted. Last but not least, $\mathbf{m}$ is embedded in the subspace spanned by the real secret key $\mathbf{U}$. Without knowledge of $\mathbf{U}$, the network cannot easily identify and remove the watermark. To overcome these challenges, the key $\hat{\mathbf{U}}$ estimated in Stage I is used to provide the network with information about $\mathbf{U}$. The lower part of Fig. 1 illustrates the training process of the watermark removal network. Initially, a set of new original images $\mathbf{I}'_o$ are randomly sampled from a public dataset. $\hat{\mathbf{U}}$ embeds randomly generated watermarks $\mathbf{m}'$ into $\mathbf{I}'_o$ to obtain the watermarked images $\mathbf{I}'_w$. Subsequently, the watermark removal network is trained on the pairs of $\mathbf{I}'_w$ and $\mathbf{I}'_o$. Specifically, $\mathbf{I}'_w$ is fed into the network

$$\mathbf{I}'_r = \text{Net}(\mathbf{I}'_w), \tag{23}$$

where $\mathbf{I}'_r$ and $\text{Net}(\cdot)$ represent the attacked image and the watermark removal network, respectively. $\mathbf{I}'_r$ is visually similar to $\mathbf{I}'_o$ while containing minimal information about $\mathbf{m}$. Existing watermark removal methods often focus on minimizing the difference between $\mathbf{I}'_r$ and $\mathbf{I}'_o$. However, they often lead to unsatisfactory results. The network reconstruction ability determines image quality. It is important to design an effective network. Moreover, although $\mathbf{I}'_w$ provides the information about $\mathbf{U}$, the trained network cannot remove watermarks effectively. The minimization of $\mathbf{I}'_r$ and $\mathbf{I}'_o$ aims to achieve good image quality instead of removing watermarks. In this way, the distortion caused by watermark embedding may be seen as an inherent part of the original image. Hence, it is critical to design another objective function to guide watermark removal. Our solutions to these problems are given as follows:

1) *Image Quality:* To achieve high-quality image reconstruction, it is crucial to design the watermark removal network carefully. As described in Section II-A, SS watermarking embeds watermarks in the frequency domain. Reconstructing images in this domain can enhance image quality. Moreover, an image consists of various coefficients. Capturing their relationships is essential for image quality enhancement. To this end, we design a QCNN-based network to learn the relationships. Fig. 3 illustrates the structure of the proposed network. Our network comprises CNN and quaternion blocks. The quaternion block captures the relationships between frequency coefficients, while the CNN block learns frequency features individually. Let $\text{CBlock}(\cdot)$ and $\text{QBlock}(\cdot)$ denote the CNN and quaternion blocks, respectively. $\text{QConv}_{k\times k}(\cdot)$ denotes a QCNN of kernel size $k \times k$. $\text{Down}(\cdot)$ and $\text{Up}(\cdot)$ denote the downsampling and upsampling operations, respectively. For the proposed network, the input image is first transformed into the frequency domain using the 1-level DWT:

$$\mathbf{F} = \text{DWT}(\mathbf{I}'_w), \tag{24}$$

where $\mathbf{I}_w \in \mathbb{R}^{1\times H\times W}$ and $\mathbf{F} \in \mathbb{R}^{1\times H\times W}$. $\mathbf{F}$ is divided into four equal parts and reshaped into a four-channel feature $\mathbf{F}_1 \in \mathbb{R}^{4\times h\times w}$, where $h = H/2$ and $w = W/2$. Subsequently, a QCNN of kernel size 1×1 increases the number of channels of $\mathbf{F}_1$ to a specified integer N_{mid}:

$$\mathbf{F}_2 = \text{QConv}_{1\times1}(\mathbf{F}_1), \tag{25}$$

where $\mathbf{F}_2 \in \mathbb{R}^{N_{mid}\times h\times w}$. $\mathbf{F}_2$ is fed into a CNN block followed by a quaternion block and a downsampling module:

$$\begin{aligned}\mathbf{F}_3 &= \text{QBlock}_1(\text{CBlock}_1(\mathbf{F}_2)),\\ \mathbf{F}_4 &= \text{Down}_1(\mathbf{F}_3),\end{aligned} \tag{26}$$

where $\mathbf{F}_3 \in \mathbb{R}^{N_{mid}\times h\times w}$ and $\mathbf{F}_4 \in \mathbb{R}^{2N_{mid}\times h/2\times w/2}$. The same process is repeated once more:

$$\mathbf{F}_5 = \text{QBlock}_2(\text{CBlock}_2(\mathbf{F}_4)),$$

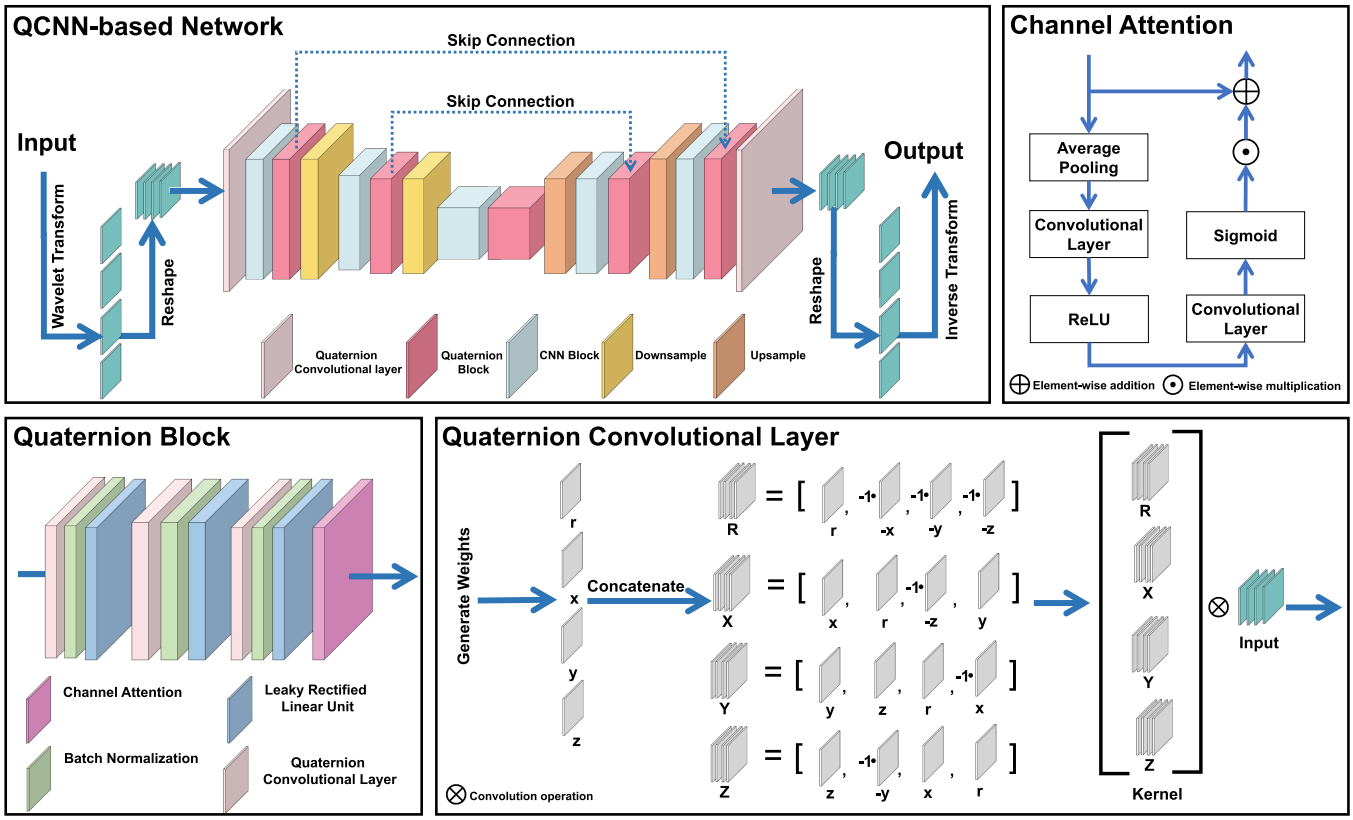

Fig. 3. Structure of our watermark removal network. The proposed network has one downsampling branch, one upsampling branch, and two skip connections. Both branches contain quaternion blocks and traditional CNN blocks.

$$\mathbf{F}_6 = \text{Down}_2(\mathbf{F}_5), \tag{27}$$

where $\mathbf{F}_5 \in \mathbb{R}^{2N_{mid} \times h/2 \times w/2}$ and $\mathbf{F}_6 \in \mathbb{R}^{4N_{mid} \times h/4 \times w/4}$. Next, $\mathbf{F}_6$ is processed by another CNN-quaternion block:

$$\mathbf{F}_7 = \text{QBlock}_3(\text{CBlock}_3(\mathbf{F}_6)), \tag{28}$$

where $\mathbf{F}_7 \in \mathbb{R}^{4N_{mid} \times h/4 \times w/4}$. $\mathbf{F}_7$ is upscaled and fed into a CNN-quaternion block. The process is repeated and formulated as follows:

$$\begin{aligned} \mathbf{F}_8 &= \text{QBlock}_4(\text{CBlock}_4(\text{Up}_1(\mathbf{F}_7))) + \mathbf{F}_5, \\ \mathbf{F}_9 &= \text{QBlock}_5(\text{CBlock}_5(\text{Up}_2(\mathbf{F}_8))) + \mathbf{F}_3, \end{aligned} \tag{29}$$

where $\mathbf{F}_8 \in \mathbb{R}^{2N_{mid} \times h/2 \times w/2}$ and $\mathbf{F}_9 \in \mathbb{R}^{N_{mid} \times h \times w}$. Both upsampling modules decrease the number of channels and increase the spatial size by a factor of two. After that, a 1×1 QCNN decreases the number of channels of $\mathbf{F}_9$ to four. The result is further reshaped into $\mathbf{F}_{10} \in \mathbb{R}^{1 \times H \times W}$. Finally, the inverse DWT is performed on $\mathbf{F}_{10}$ to obtain $\mathbf{I}'_r$. Within each quaternion block, each QCNN is followed by a batch normalization (BN) layer and a leaky rectified linear unit (LReLU). The first QCNN divides the input channels in half:

$$\mathbf{F}' = \text{LReLU}(\text{BN}(\text{QConv}_{1\times1}(\mathbf{F}_{in}))), \tag{30}$$

where $\mathbf{F}_{in} \in \mathbb{R}^{N_{mid} \times h' \times w'}$ is the input of the quaternion block and $\mathbf{F}' \in \mathbb{R}^{N_{mid}/2 \times h' \times w'}$ is the output of the first LReLU. Following that, the second LReLU layer outputs

$$\mathbf{F}'' = \text{LReLU}(\text{BN}(\text{QConv}_{3\times3}(\mathbf{F}'))), \tag{31}$$

where $\mathbf{F}'' \in \mathbb{R}^{N_{mid}/2 \times h' \times w'}$. The last LReLU layer outputs

$$\mathbf{F}''' = \text{LReLU}(\text{BN}(\text{QConv}_{1\times1}(\mathbf{F}''))), \tag{32}$$

where $\mathbf{F}''' \in \mathbb{R}^{N_{mid} \times h' \times w'}$. Based on the channel attention mechanism, a scale vector is calculated as follows:

$$\mathbf{S} = \text{Sigmoid}(\text{Conv}(\text{ReLU}(\text{Conv}(\text{AVGPool}(\mathbf{F}''')))))), \tag{33}$$

where $\mathbf{S} \in \mathbb{R}^{N_{mid} \times 1 \times 1}$; $\text{Conv}(\cdot)$ denotes the convolutional operation of kernel size 1×1; $\text{AVGPool}(\cdot)$, $\text{ReLU}(\cdot)$, and $\text{Sigmoid}(\cdot)$ represent the global average pooling, rectified linear unit, and sigmoid function, respectively. Finally, the output of the quaternion block is expressed as

$$\mathbf{F}_{\text{out}} = \mathbf{F}_{in} + \mathbf{F}''' \cdot \mathbf{S}, \tag{34}$$

where $\mathbf{F}_{\text{out}} \in \mathbb{R}^{N_{mid} \times h' \times w'}$ is the output and "$\cdot$" denotes the channel-wise multiplication operation. The CNN block has the same architecture as the quaternion block, except all QCNNs are replaced with CNNs. To train the network, $\mathbf{I}'_r$ and $\mathbf{I}'_o$ are used to calculate a loss value as follows:

$$\mathcal{L}_{\text{quality}} = \text{L}_1(\mathbf{I}'_r, \mathbf{I}'_o), \tag{35}$$

TABLE I
PERFORMANCE COMPARISON OF DIFFERENT METHODS UNDER DIFFERENT WATERMARK EMBEDDING METHODS

SS Method	N_o	FastICA	MLE	AVE	EK	Ours
Additive SS [19]	200	10.75	56.04	56.83	60.79	**61.36**
	600	23.07	76.21	75.96	78.49	**79.40**
	1000	29.95	82.28	81.95	83.67	**84.69**
	1400	34.10	85.25	84.95	86.13	**87.20**
	1800	37.57	87.09	86.82	87.68	**88.80**
ISS [20]	200	8.48	55.14	55.99	60.18	**60.82**
	600	16.74	75.58	75.35	78.08	**79.04**
	1000	22.62	81.84	81.53	83.38	**84.46**
	1400	29.59	84.88	84.58	85.89	**87.00**
	1800	30.94	86.77	86.52	87.48	**88.62**
CW-ISS [22]	200	8.78	47.80	48.69	53.12	**53.35**
	600	10.75	67.65	67.70	71.43	**71.91**
	1000	18.48	76.33	76.16	78.84	**79.50**
	1400	25.08	80.41	80.20	82.34	**83.12**
	1800	28.35	83.08	82.89	84.61	**85.43**
RNW [21]	200	15.86	50.93	51.97	53.72	**54.02**
	600	27.71	71.56	71.51	72.67	**73.23**
	1000	35.93	78.87	78.64	79.33	**80.06**
	1400	36.82	82.71	82.48	82.85	**83.65**
	1800	38.64	85.09	84.86	84.98	**85.85**
TSW [23]	200	8.63	51.68	52.49	56.63	**57.24**
	600	18.05	72.12	71.89	74.96	**75.90**
	1000	23.14	79.02	78.69	80.96	**82.00**
	1400	26.72	82.48	82.15	83.87	**84.97**
	1800	27.93	84.70	84.41	85.75	**86.88**

where $\mathrm{L}_1(\cdot,\cdot)$ denotes the L1 loss function. Under the guidance of $\mathcal{L}_{\text{quality}}$, the proposed network learns to enhance the visual quality of $\mathbf{I}'_r$.

2) *Removal:* If $\mathbf{I}'_r$ contains minimal information about the watermark $\mathbf{m}'$, the extracted watermark $\hat{\mathbf{m}}'$ from $\mathbf{I}'_r$ becomes significantly different from $\mathbf{m}'$. Commonly, BER in (13) is used to measure this difference. A larger BER value indicates a larger difference. However, BER is not suitable for evaluating watermark removal performance. To illustrate the issue, consider the following scenario: Let $\mathbf{m}' = [0,0,0]^T$ and $\hat{\mathbf{m}}' = [1,1,1]^T$. In this case, BER takes the maximum value of 1. However, if we flip each bit of $\hat{\mathbf{m}}'$, we obtain $\hat{\mathbf{m}}' = [0,0,0]^T$, leading to a BER value of 0. Hence, a large BER value does not necessarily indicate good watermark removal performance. Instead, removal performance should be evaluated by the minimum value in BER and $1-$ BER. This minimum value calculation can be expressed as the RR in (13). When the watermark is completely removed, the BER is equal to 0.5. That is to say, 50% of the bits in $\hat{\mathbf{m}}'$ are different from those in $\mathbf{m}'$. As discussed in Section III-A, the gradient of (8) is always zero, causing that the trainable parameters fail to be updated. To address this problem, Stage II employs the sigmoid-based decoder proposed in (21) to extract watermarks. If the decoder output is larger than 0.5, the extracted bit is considered as 1. Otherwise, the extracted bit is considered as 0. Thus, 0.5 is the decision boundary for the real secret key. Combining this observation with (13), we find that shifting the decision boundary can make watermark extraction failed for more bits. This motivates us to design an RR loss function to guide the watermark removal network and encourage the decoder output to approach a given boundary. Specifically, a watermark $\hat{\mathbf{m}}'$ is extracted from $\mathbf{I}'_r$ using the estimated key $\hat{\mathbf{U}}$ and the sigmoid-based decoder. Subsequently, $\hat{\mathbf{m}}'$ is used to calculate a loss value by

$$\mathcal{L}_{\text{RR}} = \frac{1}{N_c}\sum_{i=1}^{N_c} \|\mathbf{m}'(i) - \epsilon\|^2, \tag{36}$$

where ϵ is a given boundary. With the assistance of $\mathcal{L}_{\text{RR}}$, the decision boundary is shifted and the RR value is increased. The extracted watermark becomes more different from the original one.

Combining (35) and (36), the overall loss function for the watermark removal network can be written as

$$\mathcal{L}_{\text{II}} = \lambda_1 \mathcal{L}_{\text{quality}} + \lambda_2 \mathcal{L}_{\text{RR}}, \tag{37}$$

where λ_1 and λ_2 are two hyperparameters. Due to its generalization ability, the trained network can be applied to the watermarked images of the owner.

The proposed secret key estimation method may be applied only for some scenarios or watermarking techniques since it uses the unique characteristics of SS watermarking and the KMA scenario. Nevertheless, the proposed framework can be extended to other scenarios and watermarking techniques. Our framework consists of two stages. The first stage estimates the secret key and the second stage utilizes the estimated key to train a watermark removal network. A secret key estimation method can be designed for a specific watermarking technique and scenario. For example, the set-membership estimation method was developed for the quantization index modulation watermarking in the watermarked-only attack scenario, where attackers can access only watermarked data [35]. Then, such estimated key can be used to train the watermark removal network in our framework. In such a way, our framework can be generalized to other scenarios and watermarking methods. In addition, our framework can be applied to adaptive watermark embedding methods. In [29], a training strategy was designed to train a watermark removal network on different embedding strength factors. Specifically, the embedding strength decreases as the training epoch increases. As a result, the trained network learns the features of different embedding strength factors and can attack watermarked images with different visual qualities. Using this strategy, our framework can be adapted to different embedding parameters.

IV. EXPERIMENTAL RESULTS

In this section, the **BOWS2** dataset [36] is utilized to generate the additional information in the KMA scenario. For Stage II, the **BOSSBase** (v1.01) dataset [37] is selected as the public dataset. Both **BOWS2** and **BOSSBase** consist of 10,000 grayscale images of size 512×512. The original images $\mathbf{I}_o$ of the owner are randomly selected from **BOWS2** and then resized to a resolution of 256×256. The real secret key $\mathbf{U}$ embeds watermarks $\mathbf{m}$ into $\mathbf{I}_o$ to generate the watermarked images $\mathbf{I}_w$. In Stage II, new original images $\mathbf{I}'_o$ are randomly sampled from **BOSSBase** and embedded with new watermarks $\mathbf{m}'$ to generate the watermarked images $\mathbf{I}'_w$ of size 256×256. This is implemented using the estimated key $\hat{\mathbf{U}}$ of Stage I. The embedding strength is tuned to ensure that the average PSNR value between the original and

TABLE II
COMPARISON OF DIFFERENT DEEP WATERMARK REMOVAL METHODS

Attack Method	#Parameter	#Time	Additive SS	ISS	CW-ISS	RNW	TSW
			PSNR / SSIM / RR	PSNR / SSIM / RR	PSNR / SSIM / RR	PSNR / SSIM / RR	PSNR / SSIM / RR
Geng *et al.* [26]	0.6671 M	7.3	37.20 / 0.9854 / 0.15	36.58 / 0.9824 / 0.01	36.38 / 0.9819 / 0.16	36.09 / 0.9834 / 0.11	36.01 / 0.9801 / 0.01
Hatoum *et al.* [27]	54.4143 M	**0.3**	36.90 / 0.9840 / 0.11	36.30 / 0.9824 / 0.00	36.36 / 0.9828 / 0.15	35.28 / 0.9815 / 0.12	36.03 / 0.9812 / 0.00
Li *et al.* [28]	23.0894 M	2.4	35.61 / 0.9782 / 0.05	35.02 / 0.9744 / 0.00	34.88 / 0.9748 / 0.12	34.04 / 0.9715 / 0.10	34.67 / 0.9755 / 0.00
Wang *et al.* [29]	1.9736 M	12.6	38.01 / 0.9879 / 0.15	37.40 / 0.9861 / 0.03	37.35 / 0.9863 / 0.21	36.61 / 0.9868 / 0.13	37.16 / 0.9858 / 0.04
Baseline	0.2780 M	1.5	38.40 / 0.9883 / 0.13	37.61 / 0.9862 / 0.01	37.59 / 0.9869 / 0.16	36.83 / 0.9861 / 0.15	37.34 / 0.9854 / 0.01
Ours w/o RR	0.2780 M	1.5	38.43 / 0.9885 / 0.28	38.24 / 0.9880 / 0.16	38.17 / 0.9882 / 0.34	37.00 / 0.9873 / 0.23	38.24 / 0.9880 / 0.17
Ours	**0.2780 M**	1.5	**38.59 / 0.9886 / 0.32**	**38.54 / 0.9883 / 0.20**	**38.36 / 0.9885 / 0.38**	**37.10 / 0.9874 / 0.28**	**38.37 / 0.9887 / 0.20**

TABLE III
COMPARISON OF DIFFERENT REMOVAL METHODS

Attack Method	Additive SS	ISS	CW-ISS	RNW	TSW
	PSNR / SSIM / RR	PSNR / SSIM / RR	PSNR / SSIM / RR	PSNR / SSIM / RR	PSNR / SSIM / RR
AWGN (0.01)	33.65 / 0.9328 / 0.03	33.57 / 0.9327 / 0.00	33.53 / 0.9326 / 0.10	32.66 / 0.9317 / 0.09	33.62 / 0.9333 / 0.00
AWGN (0.02)	31.40 / 0.8382 / 0.03	31.35 / 0.8380 / 0.00	31.33 / 0.8379 / 0.10	30.78 / 0.8367 / 0.09	31.38 / 0.8384 / 0.00
AWGN (0.03)	29.17 / 0.7349 / 0.03	29.14 / 0.7347 / 0.00	29.13 / 0.7346 / 0.11	28.79 / 0.7331 / 0.10	29.16 / 0.7350 / 0.00
Salt-and-pepper noise (0.01)	24.56 / 0.7730 / 0.03	24.55 / 0.7730 / 0.00	24.55 / 0.7731 / 0.12	24.42 / 0.7717 / 0.12	24.56 / 0.7730 / 0.00
Salt-and-pepper noise (0.02)	21.76 / 0.6286 / 0.04	21.75 / 0.6279 / 0.00	21.76 / 0.6288 / 0.13	21.69 / 0.6269 / 0.14	21.76 / 0.6285 / 0.00
Salt-and-pepper noise (0.03)	20.07 / 0.5223 / 0.04	20.06 / 0.5219 / 0.00	20.07 / 0.5220 / 0.15	20.02 / 0.5208 / 0.15	20.07 / 0.5221 / 0.00
JPEG compression (60)	30.88 / 0.9064 / 0.03	30.83 / 0.9063 / 0.00	30.81 / 0.9061 / 0.10	30.33 / 0.9005 / 0.09	30.86 / 0.9068 / 0.00
JPEG compression (50)	30.37 / 0.8942 / 0.03	30.33 / 0.8941 / 0.00	30.31 / 0.8940 / 0.10	29.88 / 0.8934 / 0.09	30.35 / 0.8946 / 0.00
JPEG compression (40)	29.86 / 0.8802 / 0.03	29.82 / 0.8800 / 0.00	29.81 / 0.8799 / 0.11	29.42 / 0.8793 / 0.10	29.84 / 0.8805 / 0.00
Median filtering (3×3)	28.17 / 0.8612 / 0.04	28.13 / 0.8607 / 0.00	28.12 / 0.8606 / 0.11	27.89 / 0.8598 / 0.10	28.14 / 0.8611 / 0.00
Median filtering (5×5)	25.61 / 0.7647 / 0.06	25.58 / 0.7637 / 0.00	25.57 / 0.7635 / 0.15	25.48 / 0.7630 / 0.13	25.58 / 0.7638 / 0.00
Median filtering (7×7)	24.24 / 0.7042 / 0.13	24.21 / 0.7031 / 0.03	24.21 / 0.7030 / 0.24	24.16 / 0.7026 / 0.20	24.21 / 0.7031 / 0.04
Gaussian filtering (3×3)	28.75 / 0.8810 / 0.04	28.70 / 0.8806 / 0.00	28.69 / 0.8805 / 0.10	28.43 / 0.8797 / 0.09	28.71 / 0.8810 / 0.00
Gaussian filtering (5×5)	27.88 / 0.8542 / 0.04	27.83 / 0.8536 / 0.00	27.82 / 0.8535 / 0.11	27.63 / 0.8528 / 0.09	27.83 / 0.8539 / 0.00
Gaussian filtering (7×7)	27.82 / 0.8521 / 0.04	27.76 / 0.8515 / 0.00	27.75 / 0.8514 / 0.11	27.57 / 0.8507 / 0.09	27.76 / 0.8518 / 0.00
Rotation (1)	25.24 / 0.9705 / 0.04	25.23 / 0.9705 / 0.00	25.22 / 0.9704 / 0.14	25.08 / 0.9696 / 0.14	25.24 / 0.9711 / 0.00
Rotation (3)	20.87 / 0.9564 / 0.06	20.86 / 0.9564 / 0.01	20.86 / 0.9562 / 0.22	20.81 / 0.9555 / 0.22	20.87 / 0.9569 / 0.03
Rotation (5)	18.90 / 0.9402 / 0.08	18.89 / 0.9402 / 0.03	18.89 / 0.9401 / 0.22	18.86 / 0.9394 / 0.22	18.90 / 0.9408 / 0.06
Cropping (0.1)	25.13 / 0.9605 / 0.03	25.11 / 0.9605 / 0.00	25.11 / 0.9604 / 0.11	24.97 / 0.9596 / 0.12	25.12 / 0.9610 / 0.00
Cropping (0.2)	19.23 / 0.9250 / 0.05	19.22 / 0.9250 / 0.00	19.22 / 0.9248 / 0.18	19.19 / 0.9240 / 0.19	19.23 / 0.9255 / 0.02
Cropping (0.3)	15.77 / 0.8701 / 0.06	15.77 / 0.8701 / 0.01	15.77 / 0.8700 / 0.20	15.75 / 0.8691 / 0.20	15.77 / 0.8705 / 0.03
Resizing (1/2)	27.40 / 0.8381 / 0.04	27.35 / 0.8374 / 0.00	27.35 / 0.8373 / 0.11	27.18 / 0.8367 / 0.10	27.35 / 0.8377 / 0.00
Resizing (1/3)	25.46 / 0.7707 / 0.06	25.43 / 0.7699 / 0.00	25.42 / 0.7698 / 0.15	25.33 / 0.7694 / 0.14	25.42 / 0.7700 / 0.00
Resizing (1/4)	24.37 / 0.7187 / 0.16	24.34 / 0.7181 / 0.08	24.34 / 0.7180 / 0.29	24.30 / 0.7179 / 0.26	24.34 / 0.7181 / 0.11
Ours	**38.59 / 0.9886 / 0.32**	**38.54 / 0.9883 / 0.20**	**38.36 / 0.9885 / 0.38**	**37.10 / 0.9874 / 0.28**	**38.37 / 0.9887 / 0.20**

watermarked images is approximately 35.0dB. Besides, various embedding methods are considered, including additive SS, ISS, CW-ISS, RNW, and TSW. For these methods, the parameters N_v, N_c, and λ are set to 2,048, 16, and 0.8, respectively.

A. Secret Key Estimation

Experimental Settings: In this subsection, all experiments are conducted in Matlab (R2021b). our estimation method updates the estimated key using the Adam optimizer with decay rates of 0.9 and 0.99. The learning rate, batch size, and epoch are set to 0.001, 1, and 120, respectively. Our method is compared with FastICA, MLE, AVE, and EK.

Accuracy Comparison: First, watermarked signals are extracted from the watermarked images. Each estimation method then outputs an estimated key using the watermarked signals and the embedded watermarks. Next, the estimated and real secret keys are used to calculate the NC value via (12). A larger NC value indicates higher estimation accuracy. The comparison results are presented in Table I, where the best results are highlighted in bold and the second-best results are underlined. N_o represents the number of data used for estimation. As can be observed, the NC values of different estimation methods generally increase with the increase of N_o since more data are available for estimation. In all cases, our method outperforms the competing methods. Compared with the second-best method, our method achieves higher accuracy by approximately 0.57%-1.12%, 0.64%-1.14%, 0.23%-0.82%, 0.30%-0.87%, and 0.61%-1.13% on additive SS, ISS, CW-ISS, RNW, and TSW, respectively. This superiority is attributed to Adam's effective optimization strategy and the backpropagation algorithm.

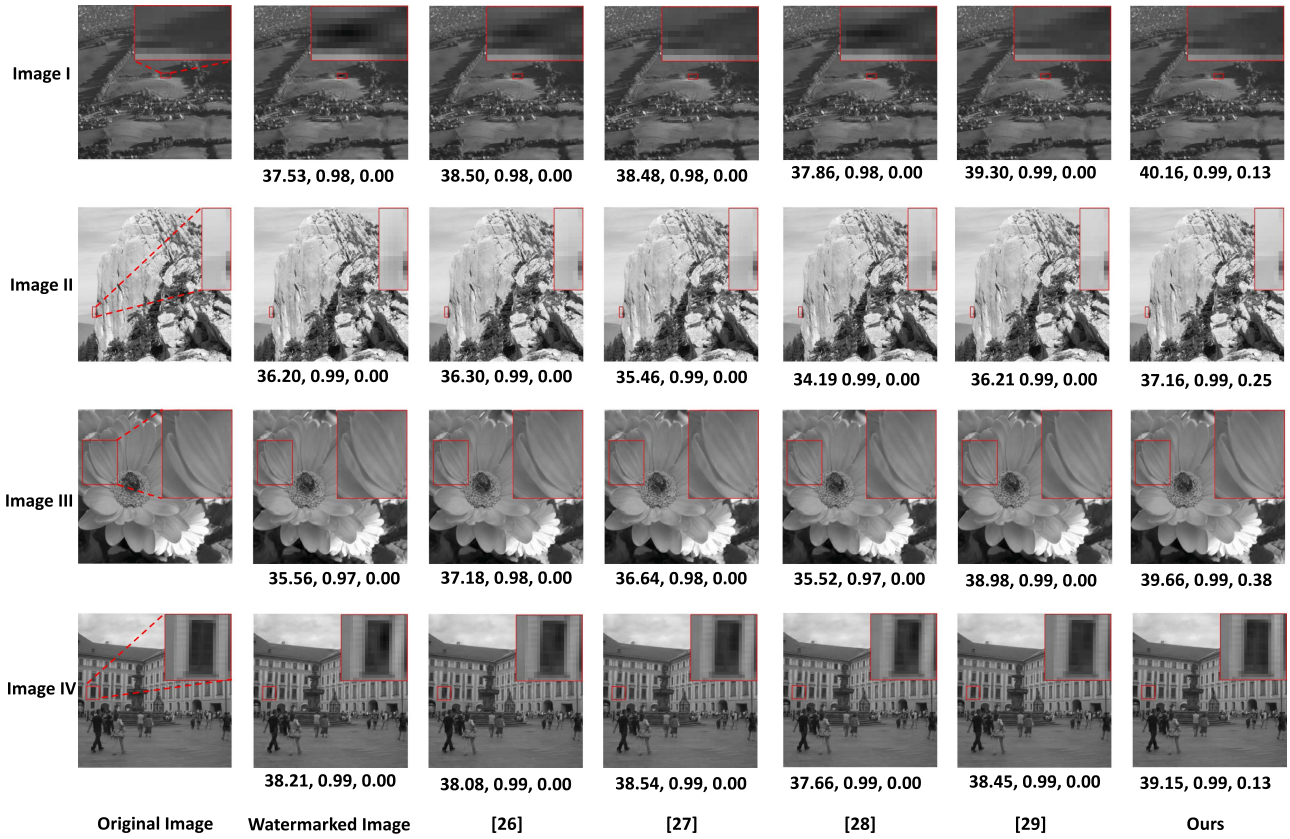

Fig. 4. Visual comparison of different deep watermark removal methods. The first and second columns are the original and watermarked images, respectively. The third to seventh columns are the watermark-removed images of [26], [27], [28], [29], and our method, respectively. Below each attacked and watermarked images, the PSNR, SSIM, and RR values are presented.

B. Watermark Removal

Experimental Settings: All experiments in this subsection are conducted using the PyTorch framework and a single NVIDIA RTX 3090Ti GPU. Training and test sets are generated from **BOWS2** using the real secret key $\mathbf{U}$. The training set is used to obtain the estimated key $\hat{\mathbf{U}}$ in Stage I while the test set is used to evaluate watermark removal performance. Besides, $\hat{\mathbf{U}}$ and **BOSSBase** are utilized to generate another training set to train the watermark removal network. All sets include 1,000 different data. The trainable parameters of our network are updated by AdamW [38]. The exponential decay rates of AdamW are set to 0.9. The learning rate, batch size, λ_1, λ_2, N_{mid}, and ϵ are set to 0.008, 8, 5, 0.5, 32, and 0.49, respectively. The watermark removal performance is evaluated using RR, PSNR, and SSIM.

Comparison with Deep Methods: Our method is compared with the state-of-the-art deep learning methods [26], [27], [28], [29]. Table II reports the experimental results. "w/o Estimation" and "w/o RR" indicate that the network is trained without using the secret key estimation and the RR loss, respectively. "Baseline" refers to our network trained using a randomly generated key. As can be observed, the baseline model outperforms the competing methods in terms of image quality. Utilizing secret key estimation further enhances the proposed model's performance in PSNR, SSIM, and RR. It provides more information about the real secret key, guiding the trained network to identify the subspace where the watermarks are embedded. Moreover, the RR loss further enhances performance by shifting the decision boundary of the original watermark extraction function. Consequently, our method surpasses existing deep learning methods with fewer parameters. Compared to the second-best method, our method achieves higher PSNR/SSIM/RR values by 0.58/0.0007/0.17, 1.14/0.0022/0.17, 1.01/0.0022/0.17, 0.49/0.0006/0.15, and 1.21/0.0029/0.16 on additive SS, ISS, CW-ISS, RNW, and TSW, respectively. The runtime of different watermark removal networks is compared on 1,000 images of size 256×256. The comparison results are shown in Table II. As can be observed, our method achieves a runtime of 1.5 seconds, which is the second best among the competing methods. This demonstrates the computation efficiency of our method, allowing it to handle more massive datasets within a reasonable computation cost.

Comparison with Traditional Methods: Experiments are also conducted to compare the proposed method with traditional methods. The competing methods include additive white Gaussian noise (AWGN), salt-and-pepper noise, JPEG compression, median filtering, Gaussian filtering, rotation, cropping, and resizing. These attacks are applied with different variances, densities, quality factors, kernel sizes, kernel sizes, angles, ratios, and scale factors, respectively. Table III reports the comparison

Fig. 5. Visual comparison of the traditional and our methods. Below each attacked image, the PSNR, SSIM, and RR values are presented.

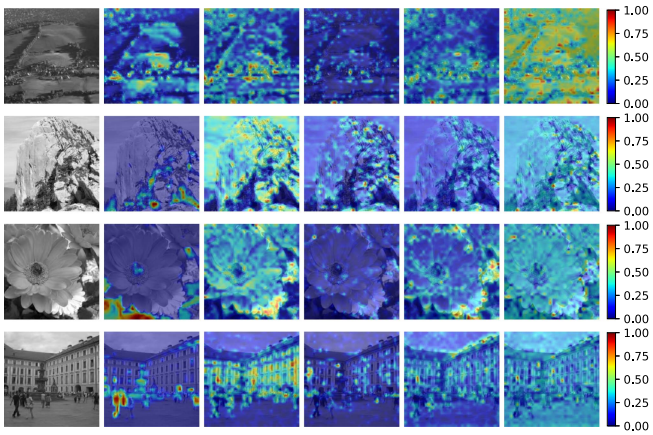

Fig. 6. Heat map visualization of the features learned by our method.

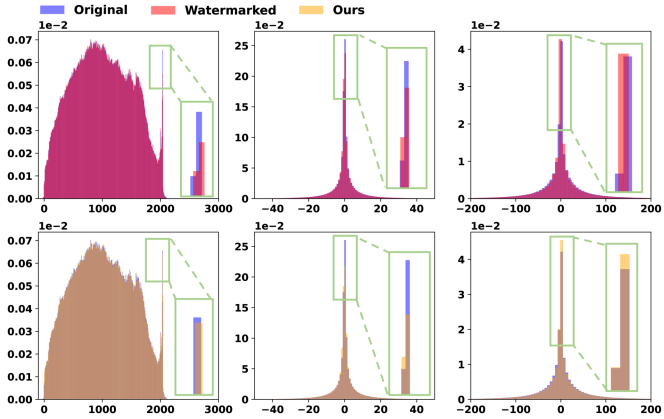

Fig. 7. Histogram comparison of the original image and the watermarked/attacked image.

results. As can be observed, the median filtering, resizing, and cropping methods perform better than other traditional methods. Nevertheless, our method outperforms all competing methods in terms of PSNR, SSIM, and RR. Existing SS watermarking methods are designed to be robust against traditional methods. Although watermarked images are distorted seriously, the embedded watermarks can be extracted successfully. Hence, traditional methods cannot remove the watermarks effectively. On the other hand, the proposed method not only removes watermarks but also improves image quality.

Visualization: Figs. 4 and 5 present visual comparisons of watermark removal results by different deep and traditional methods, respectively. These images contain diverse contents, including buildings, plants, mountains, and human subjects. As seen from the zoom-in regions at the upper right corner of each image, the deep methods in [26], [27], [28] cannot remove distortion and watermarks. The method in [29] can eliminate distortion but fails to remove the embedded watermarks. Our method can effectively remove both distortion and embedded watermarks. Traditional methods, on the other hand, remove watermarks while bringing distortions to the watermarked images. They often result in low removal rates and significantly degrade image quality. Therefore, our method achieves superior watermark removal performance compared to existing deep and traditional methods. Our method effectively restores

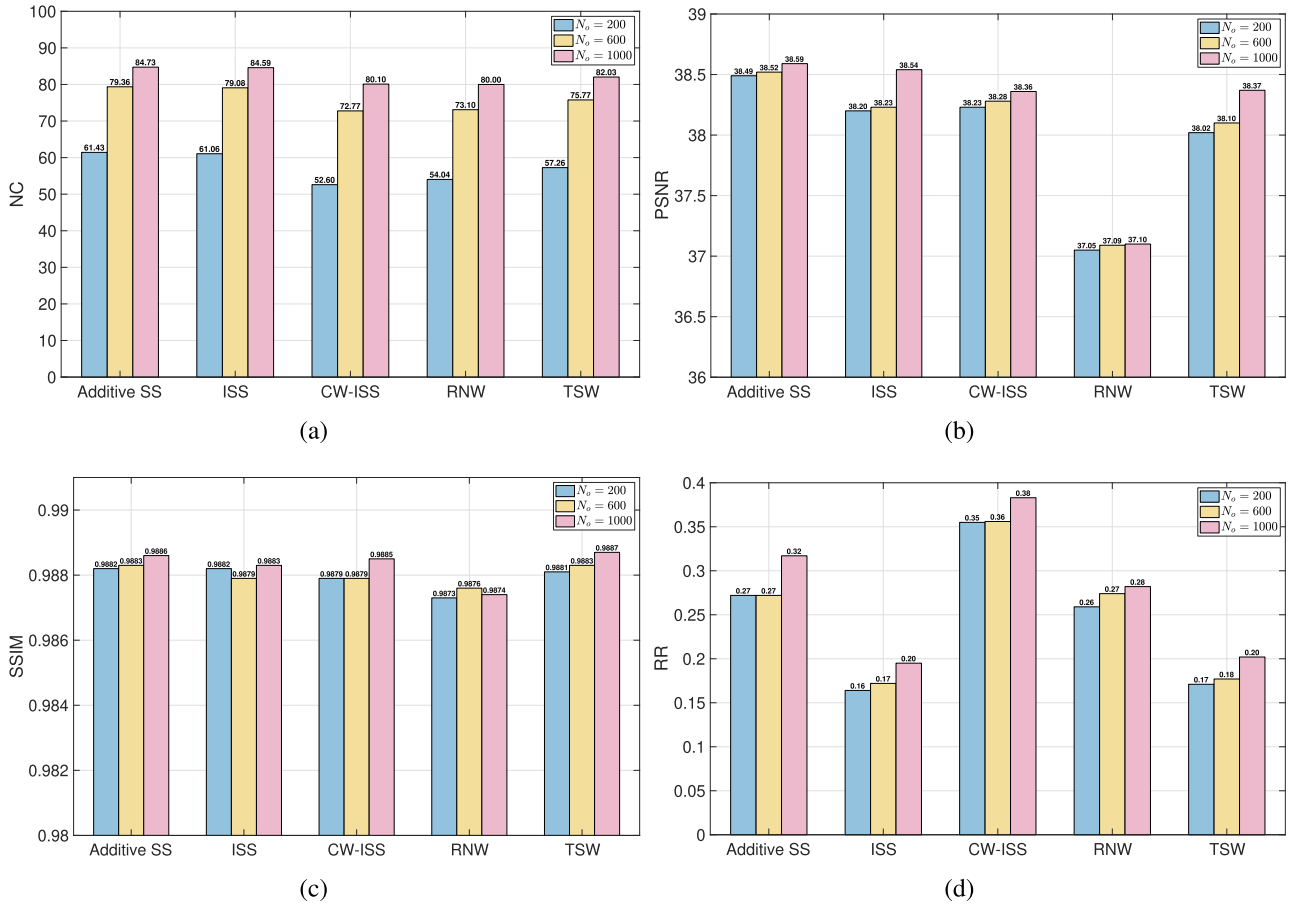

Fig. 8. Comparison of different numbers of training data (N_o) and SS embedding methods. (a) NC values in percentage (↑), (b) PSNR values (↑), (c) SSIM values (↑), and (d) RR values (↑).

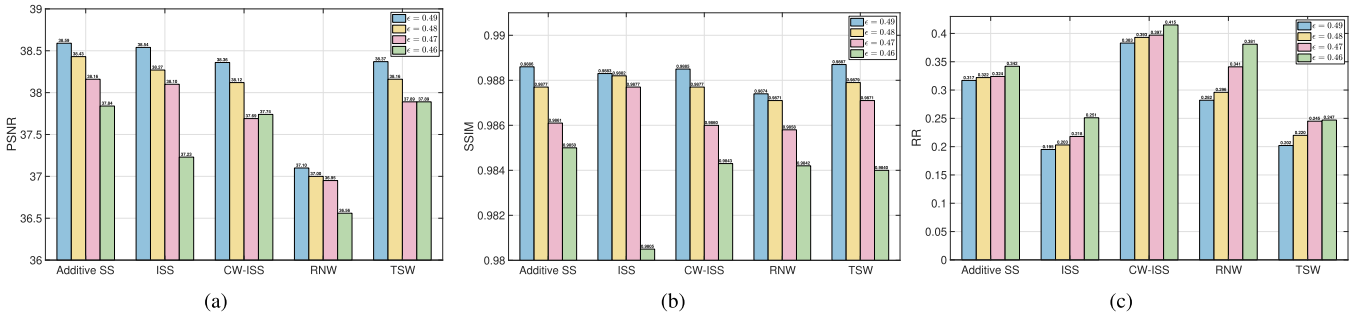

Fig. 9. Comparison of different ϵ values in (36) and SS embedding methods. (a) PSNR values (↑), (b) SSIM values (↑), and (c) RR values (↑).

low-frequency information, such as smooth flowers and ground. However, it is less effective at restoring high-frequency information, such as mountains and buildings with many textures. Additionally, watermarks are easier to remove if watermark embedding causes distortion mainly in smooth areas. To illustrate these findings, we show four images with different content types in Fig. 4. Image I and Image III contain more low-frequency information compared to Image II and Image IV. As a result, our method achieves higher image quality on Image I and Image III. Image III also contains less high-frequency information than Image I, making it easier to detect the distortion caused by watermark embedding. This leads to a higher removal rate for Image III. Image II and Image IV contain more high-frequency information than Image I and Image III. Hence, our method achieves lower visual quality. Moreover, the abundance of high-frequency information makes it challenging to perceive the distortion, resulting in lower removal rates. Notably, the removal rate is greater on Image II than on Image IV because the distortion of Image II appears in smooth areas. Fig. 6 presents a visualization of the features learned by different layers of our method. The first column shows the watermarked image, while other columns display the features obtained by

each layer. From left to right, the layer becomes progressively deeper. It can be observed that the shallow layers primarily capture contrast information, providing a basic understanding of the image structures. The middle layer begins to focus on specific regions with noticeable distortion. This indicates that the network learns to identify these distorted areas. In the deeper layers, the network learns a wider range of regions while still focusing on the distorted areas. The network learns to extract global features that indicate watermarks and distortion.

Frequency Analysis: Our method can recover the distributions of low-frequency coefficients and those embedded with watermarks, but the recovery of high-frequency coefficients is unsuccessful. To illustrate this, Fig. 7 presents the histograms of different spectral coefficients. The first and second columns represent low-frequency and high-frequency coefficients, respectively, while the third column represents coefficients embedded with watermarks. The blue, red, and yellow bars correspond to the original, watermarked, and attacked images, respectively. As can be observed, watermark embedding changes the distributions of various spectral coefficients, resulting in a degradation of image quality. To remove watermarks and improve image quality, our method learns the original distributions of these coefficients. Consequently, the distributions of low-frequency coefficients and those embedded with watermarks become more similar to the original ones. However, restoring high-frequency coefficients remains challenging as it is difficult to learn their original distributions.

Ablation Study 1: An ablation study is provided to further analyze the effect of secret key estimation on our method. A set of estimated keys are obtained using different sizes of training set for training watermark removal networks. Fig. 8 reports the performance of these networks. As can be observed, PSNR, SSIM, and RR increase with the increase of estimation accuracy. This is because an estimated key with higher accuracy provides more precise information about the actual secret key. This information aids in extracting the watermarks with higher success probability and effectively shifts the decision boundary of the watermark extraction function.

Ablation Study 2: Another ablation study is provided to demonstrate the effect of ϵ on our method. The networks are trained using different ϵ values smaller than 0.5. Fig. 9 shows their performance results. It is observed that the RR value increases as ϵ decreases. A smaller ϵ value means that the decision boundary becomes more different from the original one. More watermark bits cannot be extracted correctly. On the other hand, the PSNR and SSIM values decrease since this degrades image quality. Similarly, an ϵ value much larger than 0.5 also degrades image quality significantly. Hence, ϵ should balance the trade-off between removal rates and image quality.

V. Conclusion

In this paper, we have proposed a secret key estimation method to learn the information about the secret key effectively. It transforms the secret key estimation into a binary classification problem. The estimated key is considered as the predicted decision boundary and iteratively updated by the backpropagation algorithm and parameter optimizer. Unlike existing deep-learning methods, we have designed a QCNN-based network to learn watermark features in the frequency domain. The relationships between frequency coefficients are captured and the image quality is improved. Combining our estimation method and QCNN-based network, we have proposed a two-stage watermark removal framework for SS watermarking. This framework utilizes the information provided by secret key estimation to train the network. To further increase removal rates, we have introduced a loss function to prevent watermark extraction. Extensive experiments and analyses have been performed to validate the effectiveness of our methods. In future work, we will design an end-to-end framework to remove watermarks effectively.

References

[1] S. Yi and Y. Zhou, "Separable and reversible data hiding in encrypted images using parametric binary tree labeling," *IEEE Trans. Multimedia*, vol. 21, no. 1, pp. 51–64, Jan. 2019.

[2] J. Chang et al., "Reversible data hiding for color images based on adaptive 3D prediction-error expansion and double deep Q-network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 8, pp. 5055–5067, Aug. 2022.

[3] C. Qin et al., "Print-camera resistant image watermarking with deep noise simulation and constrained learning," *IEEE Trans. Multimedia*, vol. 26, pp. 2164–2177, 2024, doi: 10.1109/TMM.2023.3293272.

[4] K. Ntalianis, N. Doulamis, A. Doulamis, and S. Kollias, "Automatic stereoscopic video object-based watermarking using qualified significant wavelet trees," in *Proc. IEEEDig. Tech. Papers. Int. Conf. Consum. Electron.*, 2002, pp. 188–189.

[5] I. T. on Multimedia, "Watermarking of MPEG-4 video objects," *IEEE Trans. Multimedia*, vol. 7, no. 1, pp. 23–32, Jan. 2005.

[6] Y. Xiang, I. Natgunanathan, Y. Rong, and S. Guo, "Spread spectrum-based high embedding capacity watermarking method for audio signals," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 23, no. 12, pp. 2228–2237, Dec. 2015.

[7] Z. Suet al., "SNR-constrained heuristics for optimizing the scaling parameter of robust audio watermarking," *IEEE Trans. Multimedia*, vol. 20, no. 10, pp. 2631–2644, Oct. 2018.

[8] Y. Huang, B. Niu, H. Guan, and S. Zhang, "Enhancing image watermarking with adaptive embedding parameter and PSNR guarantee," *IEEE Trans. Multimedia*, vol. 21, no. 10, pp. 2447–2460, Oct. 2019.

[9] Y.-G. Wang, G. Zhu, S. Kwong, and Y.-Q. Shi, "A study on the security levels of spread-spectrum embedding schemes in the WOA framework," *IEEE Trans. Cybern.*, vol. 48, no. 8, pp. 2307–2320, Aug. 2018.

[10] G. Hua, "Over-complete-dictionary-based improved spread spectrum watermarking security," *IEEE Signal Process. Lett.*, vol. 27, pp. 770–774, 2020.

[11] J. You, Y.-G. Wang, G. Zhu, and S. Kwong, "Truncated robust natural watermarking with Hungarian optimization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 2, pp. 483–495, Feb. 2022.

[12] T. Kalker, "Considerations on watermarking security," in *Proc. IEEE 4th Workshop Multimedia Signal Process.*, 2001, pp. 201–206.

[13] A. Kerckhoffs, "La cryptographie militaire," *J. Des Sci. Militaires*, vol. 9, pp. 5–38, 1883.

[14] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[15] Y.-G. Wang, G. Zhu, J. Li, M. Conti, and J. Huang, "Defeating lattice-based data hiding code via decoding security hole," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 1, pp. 76–87, Jan. 2021.

[16] R. Xiao, W. Ren, T. Zhu, and K.-K. R. Choo, "A mixing scheme using a decentralized signature protocol for privacy protection in bitcoin blockchain," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1793–1803, Jul./Aug. 2019.

[17] J. You et al., "Estimating the secret key of spread spectrum watermarking based on equivalent keys," *IEEE Trans. Multimedia*, vol. 25, pp. 2459–2473, 2023.

[18] F. Cayre, C. Fontaine, and T. Furon, "Watermarking security: Theory and practice," *IEEE Trans. Signal Process.*, vol. 53, no. 10, pp. 3976–3987, Oct. 2005.

[19] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. Image Process.*, vol. 6, no. 12, pp. 1673–1687, Dec. 1997.
[20] H. S. Malvar and D.A. F. Florêncio, "Improved spread spectrum: A new modulation technique for robust watermarking," *IEEE Trans. Signal Process.*, vol. 51, no. 4, pp. 898–905, Apr. 2003.
[21] P. Bas and F. Cayre, "Natural watermarking: A secure spread spectrum technique for WOA," in *Information Hiding*. Berlin, Germany: Springer, 2007, pp. 1–14.
[22] P. Bas and F. Cayre, "Achieving subspace or key security for WOA using natural or circular watermarking," in *Proc. 8th Workshop Multimedia Secur.*, 2006, pp. 80–88.
[23] Y.-G. Wang, G. Zhu, and Y. Shi, "Transportation spherical watermarking," *IEEE Trans. Image Process.*, vol. 27, no. 4, pp. 2063–2077, Apr. 2018.
[24] L. Pérez-Freire, P. Moulin, and F. Pérez-González, "Security of spread-spectrum-based data hiding," *Proc. SPIE*, vol. 6505, pp. 157–168, 2007.
[25] P. Bas and T. Furon, "A new measure of watermarking security: The effective key length," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 8, pp. 1306–1317, Aug. 2013.
[26] L. Geng, W. Zhang, H. Chen, H. Fang, and N. Yu, "Real-time attacks on robust watermarking tools in the wild by CNN," *J. Real-Time Image Process.*, vol. 17, pp. 631–641, 2020.
[27] M. W. Hatoum, J.-F. Couchot, R. Couturier, and R. Darazi, "Using deep learning for image watermarking attack," *Signal Process. Image Commun.*, vol. 90, 2021, Art. no. 116019.
[28] Q. Li et al., "Concealed attack for robust watermarking based on generative model and perceptual loss," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 8, pp. 5695–5706, Aug. 2021.
[29] C. Wang et al., "RD-IWAN: Residual dense based imperceptible watermark attack network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 11, pp. 7460–7472, Nov. 2022.
[30] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.
[31] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. 18th Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2015, pp. 234–241.
[32] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2472–2481.
[33] T. Parcollet et al., "Quaternion recurrent neural networks," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–19.
[34] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3668–3681, Aug. 2020.
[35] L. Perez-Freire and F. Perez-Gonzalez, "Security of lattice-based data hiding against the watermarked-only attack," *IEEE Trans. Inf. Forensics Secur.*, vol. 3, no. 4, pp. 593–610, Nov. 2008.
[36] P. Bas and T. Furon, *Break Our Watermarking System*, 2nd ed., Washington, DC, USA: BOWS, 2007.
[37] P. Bas, T. Filler, and T. Pevńy, "Break our steganographic system": The ins and outs of organizing boss," in *Proc. Int. Workshop Inf. Hiding*, 2011, pp. 59–70.
[38] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–10.

Jinkun You (Graduate Student Member, IEEE) received the B.S. degree in computer science and technology from the Wuhan University of Technology, Wuhan, China, in 2019, and the M.E. degree in computer technology from the Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, Beijing, China, in 2022. He is currently working toward the Ph.D. degree with the University of Macau, Taipa, Macau. His research interests include multimedia security, super-resolution, and image processing.

Yicong Zhou (Senior Member, IEEE) received the B.S. degree in electrical engineering from Hunan University, Changsha, China, and the M.S. and Ph.D. degrees in electrical engineering from Tufts University, Medford, MA, USA. He is currently a Professor with the Department of Computer and Information Science, University of Macau, Macau, China. His research interests include image processing, computer vision, machine learning, and multimedia security. He is a Fellow of SPIE (the Society of Photo-Optical Instrumentation Engineers) and was recognized as one of "Highly Cited Researchers" in 2020, 2021, and 2023. He is an Associate Editor for IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING.