Robust watermarking using diffusion of logo into auto-encoder feature maps

Maedeh Jamali¹ · Nader Karimi¹ · Pejman Khadivi² · Shahram Shirani³ · Shadrokh Samavi¹,²,³

Received: 1 November 2021 / Revised: 15 September 2022 / Accepted: 15 April 2023 / Published online: 28 April 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023, corrected publication 2023

Abstract

Digital content has grown dramatically in recent years, leading to increased attention to copyright. Image watermarking has been considered one of the most popular methods for copyright protection. With the recent advancements in the application of deep neural networks in image processing, these networks have also been used in image watermarking. Robustness and imperceptibility are two challenging features of watermarking methods that the trade-off between them should be satisfied. In this paper, we propose to use an end-to-end network for watermarking. We use a convolutional neural network (CNN) to control the embedding strength based on the images’ content. Dynamic embedding helps the network to have the lowest effect on the visual quality of the watermarked image. Different image processing attacks are simulated as a network layer to improve the robustness of the model. Our method is a blind watermarking approach that replicates the watermark string to create a matrix of the same size as the input image. This helps the network to spread the watermark information in a wider space. Therefore, by removing some parts of the image by attacks, the watermark information can be retrieved from other parts. Using dilation layers is another innovation of the proposed method that helps the watermark spread in the wider neighborhood. This will improve the robustness of the method against image processing attacks during extraction. Instead of diffusing the watermark data into the input image, we inject the data into the feature space and force the network to do this in regions that increase the robustness against various attacks. Having embedding, extraction, and attack layer in an end-to-end network helps increase robustness and imperceptibility. Experimental results show the proposed method’s superiority in imperceptibility and robustness compared to the state-of-the-art algorithms.

Keywords Blind watermarking · Convolutional auto-encoder · Dilation layer · Robustness · Data expanding
1 Introduction

Due to the dramatic increase in access to the Internet during the past decades, sharing digital content has become a prevalent practice. Contrary to analog versions, digital media can be copied with the same quality as the original one. Image editing tools can threaten digital media’s integrity. Digital tampering is now an easy task. Unauthorized copying and distribution of images are on the rise. Hence, the need for copyright protection has been emphasized in the literature [43]. One of the most reliable and popular solutions to address these security issues is digital watermarking. Image encryption [15, 57, 58], steganography [32], and digital watermarking can be considered information-hiding methods [31]. Digital watermarking was introduced in 1979 for anti-faked purposes [51], defined as recognizing original documents from fake ones. In watermarking methods, the ownership of copyrighted media is identified by embedding invisible information into an image extracted later [35]. Besides ownership authentication, watermarking has other applications, such as concealing patients’ information in their medical images for medical procedure matching and identification [48], broadcast monitoring [6], device control [12], and copy control purposes [5].

Usually, the hidden information on watermarked images is sensitive and vulnerable to image processing attacks like image compression, cropping, or undesired artifacts such as transmission noises during transmission. Hence, three key attributes are considered for evaluating the watermarking algorithms: imperceptibility, robustness, and capacity [42]. These features act contrary to each other, and the efficiency of watermarking methods highly depends on the trade-off between them. Robustness means the preservation of hidden information against image processing attacks. On the other hand, imperceptibility aims to make the watermarked image indistinguishable from the original image. The amount of hidden information (capacity) is also essential in some applications, such as medical procedures.

Different classifications can be considered for watermarking methods based on the embedding domain, the embedding method, and the extraction procedure. Concerning the extraction process, watermark methods will be classified into three classes, blind, semi-blind, and non-blind [4, 47]. Blind methods do not need the cover image during the watermark extraction process, while the other groups require extra side information alongside the extraction phase. Therefore, blind methods are practically preferred to semi or non-blind methods, although blindness will increase their complexity and may affect their imperceptibility or robustness.

The type of embedding domain is another categorization for watermarking methods. Two categories of spatial domain and frequency domains are usually considered. In spatial domain algorithms [52, 55, 59, 61], data are directly hidden in the image pixels. However, these methods are generally not robust against typical image processing attacks. In contrast to the spatial methods, frequency-domain algorithms [3, 8, 9, 13, 14, 16, 17, 19, 20, 23–25, 33, 36, 37, 39, 40, 44–46, 50, 53, 54] are more robust against different attacks [42].

Recently, machine learning algorithms have been used in watermarking applications, as they provide efficient solutions for embedding and extraction of the watermarks [1, 18, 28, 49]. These methods use learning tools for a specific part of watermarking, such as attack detection [18], predicting neighbor pixels relationship for watermark extraction [49], parameter optimization [1], and frequency domain coefficients prediction [28]. One of the most popular machine learning tools is Convolutional Neural Networks (CNN), which have attained considerable attention in various computer vision applications. Using CNN in watermarking
is new, and recently, some works have used deep networks in watermarking [2, 11, 21, 26, 27, 38, 41, 60].

In this paper, we introduce end-to-end trained blind watermarking using CNN encoder-decoder framework. The proposed framework can prepare the watermark data based on the input image’s content. Hence, it has a considerable capability of maintaining imperceptibility and robustness in comparison to state-of-the-art methods. The proposed framework consists of two parallel networks: preparing a watermark alongside the main network for embedding and making a watermarked image. It also has an attack layer that tries to simulate image processing attacks. Using the attack layer as a part of the learning system helps the framework increase the robustness of the watermarking method. This improved robustness is achieved since attacks are considered during the training phase.

Traditional watermarking methods use fixed algorithms in the transform domain, such as changing or swapping coefficients. However, in the proposed method, we embed the data in feature maps. Based on the input image and the type of attacks, the proposed method learns to use different approaches such that it has had the highest efficiency based on the definition of the loss function. The rationale for selecting CNN for watermarking is the ability of the network to represent hidden data such as weight kernels, which makes it almost impossible to retrieve the information due to unknown hyper-parameters of the network. Therefore, the watermark will only be known to the sender and receiver. This feature makes our method highly robust against hackers who want to change the data. Furthermore, the proposed CNN-based network accepts primary watermark data and prepares it based on the input image to maximize imperceptibility. Hence, the watermark remains robust against various attacks. The proposed framework is a multi-objective network where the loss function tries to create a trade-off between the objectives with opposite aims. Moreover, in our method, the network diffuses the watermark over the entire image. The watermark data will be diffused inside the whole image by replicating the input watermark (the same size as the input image). Hence, different parts of the image will have the information of the watermark, and the watermarked image attains its robustness against various kinds of heavy attacks. Although significant parts of the image are deleted or corrupted in some attacks, the network could still retrieve the hidden data.

To summarize, the contributions of the proposed method are as follows:

1. Using an end-to-end network trained for embedding, manipulating, and extracting the watermark.
2. Using a parallel network to prepare the watermark based on the input image’s content to maximize imperceptibility.
3. Injecting the watermark into feature space to have better robustness against attacks
4. Considering global information besides local information to have robust watermarking using a dilation layer that contributes more pixel information in embedding.

The rest of the paper is organized as follows: Sect. 2 is dedicated to the literature review. Section 3 details the proposed method. Experimental results are discussed in Sect. 4. The paper is concluded in Sect. 5.
2 Literature Review

Since the advent of watermarking, there has been a significant amount of research to improve the main characteristics of the technique, such as fidelity, secrecy, capacity, imperceptibility, and robustness. At first, the spatial domain is used for embedding, and the information is embedded in the cover image by manipulating the pixel image [52, 55, 59, 61]. Substitution of LSB and the randomized process is used in [55], where the authors developed a genetic algorithm to find the rightmost LSBs of the cover image. For improving imperceptibility, the entropy of blocks and histogram are used in [61]. First, they divide the cover image into blocks, and some blocks are collected based on their entropy. Then embedding is performed using the histogram shape methods. In [59], a Gaussian low-pass filter is applied as a preprocessing in the embedding phase. Several gray levels are randomly removed using a secret key, then a histogram of the filtered image is made using the selected gray levels. Choosing pixel groups with the highest number of pixels based on the built histogram is the novelty of this method. Finally, they consider a safe band between selected and not selected pixels, and the chosen pixels are candidates for embedding. The proposed algorithm in [52] is a hash-based method that is robust against alternation such as rotation and cropping. The keynote of this method is that if the tampering is sufficiently sparse, it can be localized using convex optimization problem-solving.

Another group of watermarking methods uses the transform domain that embeds information in the frequency domain by manipulating frequency coefficients. Transform domain methods perform better than the previous group and effectively preserve the visual quality of the host image [50]. Many different transform domains have been used such as discrete Fourier Transform (DFT) [50], Discrete Cosine Transform (DCT) [19, 23, 24, 40, 53, 54], Discrete Wavelet Transform (DWT) [16, 25, 36, 37, 39, 44, 45], Contourlet Transform (CT) [3, 8, 13, 14, 17, 33, 46] and Hadamard transform [9, 20]. The authors of [50] have used DFT to manage global geometric distortion. In [53], the image is divided into 8×8 blocks, and the DCT of blocks is calculated before modifying the DC coefficients. They use Arnold transform in addition to chaotic encryption to add double-layer security to the watermark. Parah et al. [40] introduce a DCT-based algorithm by partitioning the input image into 8×8 blocks, then transforming each block into DCT and selecting some blocks using a Gaussian network classifier. Eventually, the DCT coefficients are changed based on watermark bits. In [19], a DCT-SVD-based algorithm is introduced. A luminance mask of the cover image is made in the first stage. In the embedding phase, they modify the singular values of the DCT of the original image with the singular values of the produced mask. They also use a genetic algorithm for finding the control parameters. Tian et al. [54] use the combination of CT, DCT, and SVD for embedding. First, the input image is decomposed by one-level CT, and its low-frequency sub-band is divided into eight-by-eight non-overlapping blocks. DCT transforms each block, and some middle-frequency DCT coefficients are selected to make the carrier matrix. Eventually, the watermark is embedded by modifying the largest singular values of two carrier matrices.

Some methods are proposed in the DCT domain using fuzzy systems [23, 24]. The authors of [24] use a neuro-fuzzy model for embedding and extraction. They inject the human visual system (HVS) parameters into the fuzzy system and utilize the output of the fuzzy-neural framework as the strength factor for embedding. Jagdeesh et al. [23] propose fuzzy inference models for calculating the strength factors using HVS.

An enormous number of algorithms have used DWT as a transform domain for watermarking, and the middle and high-frequency coefficients are generally considered for
embedding [42]. A DWT-based algorithm is used in [44]. They add pseudo-random codes to large coefficients in a middle and high-frequency band. A quantized-based algorithm is proposed in [39]. The angel quantization of gradient vectors in different wavelet scales is considered for embedding. A geometrical model is proposed in [16] for embedding. They extracted eight samples of wavelet approximation coefficients from each block of the image and built two-line segments in a two-dimensional space. The authors in [25] propose a combination of DCT, DWT, and fuzzy systems for embedding. They first transform the image into DWT for two levels and then divide the transformed image into $8 \times 8$ blocks and calculate the DCT of them. Three attributes related to HVS are fed to a fuzzy system to calculate each DCT block’s strength factor. Then, they manipulate the DCT coefficients based on these adaptive strength factors and watermark bits. Makbol et al. [36] used a combination of SVD and DWT for embedding, and singular values are used as secret keys. Authors in [37] use image partitioning for watermarking. They first scramble the watermark using Arnold transform and then embed the watermark in the low-frequency sub-band of the wavelet transform. Rezaee et al. [45] use the wavelet domain for embedding in medical images. First, they use DWT to estimate the related coefficients to singular values. Then the optimum embedding coefficients are selected using the Whale Optimization algorithm cost function. Finally, the watermark is embedded using SVD and ISVD.

Other types of transform domains that researchers use are CT and Hadamard transform. An adaptive blind method is proposed in [13] in which the DCT coefficients of CT are considered for embedding. They determine an adaptive strength factor for each block using entropy and other attributes. First, they apply a two-level CT on the input image. They divide the approximate image into blocks in the first level. Using their proposed edge detection method, they extract the important edges. Then, part of the image with high edge concentration is considered as a candidate region for embedding. In [8], a CT-based domain is proposed. They modeled the contourlet coefficients based on a “t-location-scale” distribution. They demonstrate that the “t-location-scale” distribution has high efficiency in modeling the coefficients. The authors of [3] choose the CT domain for embedding. For more robustness, they manipulate the DCT coefficients of CT blocks. In [46], the maximum likelihood method based on normal inverse Gaussian (NIG) distribution is used for extraction. Li et al. [33] proposed a watermarking scheme for color images using quaternion Hadamard transform and Schur decomposition. A combination of canny edge detection, CT, and SVD is proposed in [14]. First, they decomposed the cover image using CT. Then the obtained low-frequency sub-band from CT will be partitioned into $4 \times 4$ non-overlap blocks. Then the SVD coefficients of selected blocks using canny edge detection are used for embedding. The proposed method in [17] combines CT and SVD. First, the cover image is divided into non-overlapping blocks, and by considering the edge map of each block, the suitable blocks are selected and transferred to CT. The SVD is performed on the selected frequency CT coefficients. In [20], non-subsampled contourlet transform (NSCT) and Schur decomposition are used. NSCT is applied to find low-frequency coefficients of the host image. Then $4 \times 4$ non-overlapping coefficient blocks from layered low-frequency coefficients are selected using the Hash pseudo-random algorithm. Finally, quantization index modulation embeds the encrypted watermark into the maximum eigenvalue. In [9], a Hadamard transform is used for embedding. They analyze Hadamard’s coefficients and find a bit-plane that satisfies transparency and robustness, and then the watermark is redundantly embedded in the selected bit-plane.

Due to the high performance of machine learning methods in watermarking, they have recently become prevalent. A learning-based watermarking algorithm is proposed in [18]. They learn the influence of different attacks on the image. This effect manipulates different
parts of the frequency spectrum that have higher resistance against these attacks. They redundantly embed the watermark in a different part of the frequency domain. Using their attack classification method, they extract the watermark from the part of the frequency domain that is less damaged by the detected attack. A combination of extreme learning machine (ELM), online sequential extreme learning machine (OSELM), and weighted extreme learning machine (WELM) have been used in [49]. They utilize machine learning techniques to learn the relationship between neighbors’ pixels to extract watermarks. An optimization-based algorithm using machine learning methods for finding the best embedding strength factor is introduced in [1]. First, a DCT-based algorithm is proposed. Then an artificial bee colony is applied to watermarked images as a training process to find the optimum strength factor. A blind learning-based algorithm for classifying the embedded bits is proposed in [28]. This method tries to adaptively modify the decoding strategy to increase robustness against anticipated attacks. The watermark extraction is performed as a binary classification problem.

Heretofore, previous works have used deep networks for one phase of detection, such as determining the strength factor for encoding [26], embedding [27], or in the extraction phase [38]. However, in recent works, they modeled an end-to-end network that used the network for whole watermarking [2, 11, 41, 60]. Kandi et al. [27] proposed a non-blind watermarking using a CNN autoencoder to hide data in feature maps using a predefined embedding algorithm. An end-to-end watermarking method is proposed in [38] that embeds data in image blocks, similar to traditional methods with uniform local embedding. The authors of [60] present an end-to-end network for steganography and watermarking using the Generative Adversarial Networks (GAN) and CNN. They try to spread the watermark data in all input image pixels by replicating the data. Each pixel has all the information about the entire watermark, which can produce some artifacts. However, the GAN helps the framework prevent these artifacts from watermarked images.

In [2], an end-to-end autoencoder is proposed that simulates a DCT layer in their deep network. After concatenating the input image and the watermark, this combination is used as input to the DCT layer, and after transforming, the embedding is performed. Due to using the DCT layer, the proposed structure has good robustness against the JPEG attack. In contrast to the mentioned works, in our proposed method, we try to prepare the watermark data based on the input image to have the lowest effect on transparency and be robust against image processing attacks. In [11], an end-to-end network has been proposed that tries to improve the imperceptibility by using a GAN structure similar to [60]. They used medical images and tried to have the lowest effect on the watermarked images, so the doctor could not detect distortion. The proposed method in [41] is another end-to-end embedding network that spreads the watermark in the spatial domain by reducing the local bit per pixel capacity and increasing the robustness. Authors in [21] combine over-complete dictionaries and create redundancy in the signal representation in the transform domain. They proposed a random matching pursuit algorithm that facilitates watermark embedding and extraction.

3 Proposed Watermarking Framework

In this section, we explain our proposed method in detail. Our method is an end-to-end CNN that tries to embed, attack, and extract in one framework as a blind watermarking method for grayscale images. It is trained as an end-to-end network to have blind secure
watermarking. This network contains two consecutive convolutional neural networks for embedding and extraction. We also have a distinct CNN alongside the main framework for preparing the watermark data, which helps the network improve the extracted image’s transparency. Implementing an attack layer to simulate image processing attacks as a network layer guarantees robust watermarking and makes end-to-end training easy. We first explain the whole deep network structure and the details of each part (embedding, attack layer, and extraction) in subsection 3.1. The evaluation metrics and training of the network will be discussed in subsection 3.2.

### 3.1 Network Structure

Figure 1 indicates the block diagram of the proposed method, which includes three main modules: embedding module using CNN, attack layer for simulating image processing attacks, and extraction module for extracting the hidden data. The embedding module accepts two inputs and embeds the watermark in the input image. After image processing attacks, the extraction module must extract the watermark data from the reconstructed image. In the following, each of these three modules will be explained.

#### 3.1.1 Embedding module

The first part of the proposed network is responsible for hiding the watermark information into the cover image, but in contrast to traditional methods, which embed information in a transform domain, we have used a CNN to conceal information in feature map space. This would be more robust than typical methods and can be resistant to change or unauthorized access because no one knows the structure of the training network.

The embedding module includes two separate networks, one for preparing the input image and the other one for making the watermark ready based on the cover image’s content that adopts adaptive embedding to the cover image.

The main part of the embedding module is an auto-encoder that receives the host image as an input. The input will pass from convolutional layers but the same size as the input. These convolution layers help us extract the input image’s main features in each layer, such as edges and other meaningful attributes. Figure 2 indicates the embedding network that is fed by the input image.

As shown in Fig. 2, the encoder contains two distinct networks that link by concatenating the networks’ output. The lower network in Fig. 2 is a CNN whose primary responsibility is to prepare the input image for embedding. The network’s input is a grayscale image...
with size $H \times W$ ($H = W = 32$) that is fed to consecutive convolution layers. The input image can have different sizes (dividable by $32 \times 32$) and should be divided into $32 \times 32$ blocks. Hence, we need a reshape part of the first stage of our network. A batch of $32 \times 32$ input image blocks is fed into the network. Then, all the input images are normalized into $[0, 1]$. We do not use pooling layers because we want to keep all information of the layer’s input and the reconstructed image’s transparency. We have a ternary block [60] in our network’s layer that is a combination of the convolution layer, followed by the activation layer and batch normalization (BN) [22]. Each input to a layer will pass from this block. BNs help the network to learn faster, and it also improves accuracy. We use dilated convolution during learning. Dilated convolution helps the network aggregate multi-scale contextual information without missing resolution [56]. This attribute helps our network to have a global view of information and expand the hidden data in a larger area. Hence more parts of the feature map will contain the information of the watermark data. Wide distribution is helpful, especially against attacks when some part of the data will lose. So other parts of the image that have the information can participate in rebuilding the data.

After suitable numbers of convolution layers, the encoded image (feature map) will be produced by the encoder part in Fig. 2, which is the same size as the input. The watermark data will be concatenated with the output feature map in this part. This has several advantages. First, the network may memorize the input image due to removing the pooling layer in our network structure, and the accuracy will not be fair. But in this phase, the watermark data is concatenated with the feature map before sending it to the decoder. This works as a noise (because a different random watermark is used for each training input during learning) that prevents the network from memorizing and forces the network to learn. Second, the hidden data will be fused with feature map data that helps the robustness of the watermarking. Because no one knows about the embedding process and the network structure, thus cannot manipulate it. Third, the proposed framework is an end-to-end network that tries to improve robustness and imperceptibility together, and the loss function controls the trade-off between them. Hence, the network has to change the network weights and feature maps to maintain transparency and be robust against attacks. Therefore, some parts
of weights will have larger values to preserve the information in different conditions. This helps the whole network to automatically control all desired features.

After concatenating the encoded image with a watermark, the produced feature maps will be fed to the decoder part that tries to reconstruct the input image. This part has a similar structure to the encoder part but with one more layer to reconstruct the watermarked image with visual similarity to the input image.

The upper part of Fig. 2 shows a CNN that is fed by watermark data. We name it the Watermark Preparing Network (WPN). This network works in parallel with the lower auto-encoder. It is responsible for preparing the watermark such that it can be extracted better and have lower effects on image visual quality. During the training phase, the network’s input is a random watermark string with a size $4 \times 4$ that is smaller than the cover image. The structure of the WPN is similar to the encoder part of the auto-encoder network by ternary blocks and without pooling. The network’s output has the same size as the hidden input data. The network has learned to emphasize some parts of the data more than other parts during learning, so we can see that the network’s output has a different value than the main hidden data. We see that having a network in addition to the main embedding network and preparing the watermark before embedding increases the final image’s robustness and visual quality. A good idea for a better watermarking algorithm is to force the whole host image to have hidden information. Therefore, we change the size of the WPN’s output to $H \times W$ by tiling. By replicating the processed watermark along width and length, we form a $H \times W$ tiled watermark. This can be considered redundancy too. Next, the tiled watermark map is concatenated with the feature map and fed to the decoder part of the auto-encoder network. During the decoding, all the feature maps and the watermark will be fused into a feature map. Finally, the decoder’s output is the watermarked image that contains the watermark information. In this part, we need another reshaping to put the image blocks beside each other and make the new image similar to the input image with the same size. After embedding the watermark, we should add image processing attacks and then extract the watermark. For this aim, we add an attack layer as a part of the network to simulate digital attacks during learning and help the network develop robustness against them. The attack layer is explained in the next section.

### 3.1.2 Attack layer

In this section, we explain the details of the attack layer that is shown in Fig. 1. Considering the attack layer as a part of the network helps the network learn more complicated watermarking patterns, which are more robust against various attacks. Another advantage of using the attack layer during learning is mitigating overfitting and preventing the network from memorizing. We consider a combination of attacks to have a multi-attack network that is robust against different kinds of attacks. A random selection method is used to select and send one attack during training. Due to its randomization, each attack has the same chance to select. This strategy helps the network see various attacks during learning, so the network tries to find suitable parts of the feature space robust against all attacks. Eventually, we will have a network with robustness against different attacks. For example, some of them manipulate high-frequency coefficients, and some of them change the low-frequency parts. Hence, we will have a general multi-attack network by selecting the combination of attacks with both features. We briefly explain each attack in the following.
a) **Noise attacks**

Gaussian noise, salt and paper, and uniform noise can be considered noise attacks in which a random noise will be added to the input image in each training iteration. However, for salt and paper, we should randomly select a ratio of pixels and change their value to white or black. We used a binomial distribution for selecting and changing the values of pixels using two masks (one for producing white pixels and one for producing black pixels). The same noise patterns are used for all images in a batch, but they would be different from images in another batch.

b) **Gaussian blurring attack**

We have used a depth-wise convolution layer that applied a filter on each input channel independently for the blurring attack. We have fixed weights for this convolution layer, and these weights can be produced for various kinds of impulse response filters like Gaussian or sharpening filters.

iii) **JPEG attack**

Most of the mentioned attacks are differentiable and can be easily implemented as a network layer. However, JPEG compression has a quantization procedure that is not differentiable. Hence, we need to estimate the quantization to implement it as a network layer. JPEG uses a DCT transform and quantization for compression. First, it divides the input image into $8 \times 8$ blocks, and then each block is transferred into DCT. Then the DCT coefficients will be divided by a quantization matrix. This quantization process is not differentiable and so cannot be used in gradient descent optimization. During quantization, the high-frequency coefficients usually become lower, and the theoretical concept of quantization also means limiting the amount of information that can be transmitted. Figure 3 shows a method we used for implementing a JPEG attack. Figure 3 (a) is an $8 \times 8$ DCT block with different values. Figure 3b indicates a quantization block sample. The quantization values become higher from left to right and up to down in a quantization block, as the color shows in the image. By dividing the DCT coefficient by these quantization values, the high frequencies of the DCT block become lower (near-zero), and we can ignore them. Instead of using this quantization table which is not derivative, we use the third block in Fig. 3 that

![Fig. 3](image)

**Fig. 3** DCT, quantization coefficients values and simulated quantization coefficients for using in a deep network, (a) DCT block, (b) Pseudo-matrix of quantization values that high frequencies are divided with higher values, (c) simulated quantization values instead of real quantization table
keeps some high coefficients from the low frequency in the DCT block. Hence, we can simulate the JPEG compression by ignoring some high-frequency coefficients. We implement DCT transformation using a convolution layer with stride 8 and kernel size 8, and the kernels are the basis vector in DCT transformation for implementing this attack layer. Then we keep a few DCT coefficients from the low-frequency part and make the high-frequency coefficients zero in a zigzag manner. Because we drop many coefficients, our layer can simulate low-quality JPEG attacks during learning.

### 3.1.3 Extraction module

The last part of the proposed network is the extraction module. The watermarked image will be fed into this network, and the network is responsible for extracting the watermark data. It is worth mentioning that the watermarked image may have been attacked before this step. The watermarked image will pass through consecutive convolutional layers without padding. Therefore, in each step, the data size will decrease until it achieves the same size as the main watermark size. The number of filters and the kernels’ sizes are the same as in the first part of the network in Fig. 2. However, we do not use dilation in the extraction module. The extraction structure is illustrated in Fig. 4, which has a simpler form than the embedding module. Like the embedding structure, we have a reshape phase for images larger than the network structure. After dividing the attacked image into the extraction network’s acceptable size, a batch of its patches will be injected into the network. Finally, the reconstructed watermark will be produced as the network’s output.

### 3.2 Network training and evaluating metrics

The whole structure is an end-to-end network that contains embedding and extraction modules. The encoder part consists of two networks: one with the cover image as input and the other one with watermark data as input. The two networks work together. The extraction module accepts an attacked watermark image and extracts the hidden data. Initially, it seems that the two parts of the network (embedding and extraction

---

**Fig. 4** The extraction module of the proposed method. Attacked image is the input to this network and extracted watermark is the output.
network) work separately and have opposite aims. Since we have end-to-end network training and the attacks are a part of the network, the extraction’s input is an attacked image. However, during the test phase, the attack layer will be removed, and we have a watermarked image as the input to this part. The embedding module is responsible for embedding the watermark data into the cover image with the lowest effect on the visual quality. High image quality requires weak embedding. On the other hand, the second part of the network should extract a watermark with the lowest difference compared to the original watermark. The extraction network is responsible for minimizing the bit error rate (BER) that is computed as (1):

\[
\text{BER}(W, W') = \frac{1}{L_w} \sum_{l=1}^{L_w} \text{XOR}(W(l), W'(l))
\]

(1)

where \(W\) and \(W'\) are the original and extracted binary watermarks, respectively. Parameter \(L_w\) indicates the length of the watermark. Based on (1), if the original and extracted watermarks are similar, the BER value should be near zero.

The similarity between the embedded and the extracted watermark requires a robust embedding that will not affect image quality. Since all parts work together, the loss function resolves the trade-off between transparency and robustness. It is a multi-objective network that the loss function is the combination of loss functions for the embedding and extraction modules:

\[
L = \lambda_1 L_1 + \lambda_2 L_2
\]

(2)

where \(\lambda_1\) and \(\lambda_2\) are coefficients that control the trade-off between the two contrasting characteristics of the watermarked image. The image’s visual quality and the extracted watermark act conversely. \(L_1\) is the loss function that represents the difference between the cover image and the watermarked image. We used mean square error (MSE) as a comparing metric between the input image and the watermarked image:

\[
L_1 = \frac{1}{n} \sum_{i=1}^{n} (\text{img}_i - \text{img}'_i)^2
\]

(3)

where \(\text{img}\) is the cover image and \(\text{img}'\) is the watermarked image. \(L_1\) will be calculated for each training sample in a batch, and the average of them will be considered as the MSE value for a batch.

In (2), \(L_2\) is the loss function that considers the difference between the input and the extracted watermark. We use binary-cross entropy and consider the extraction as a two-class classification that should label each pixel of the extracted watermark as zero or one. In (4), we compute the loss for each pixel, and it is performed for all extracted watermark bits.

\[
L_2 = - \sum_{\text{all pixels}} t_i \log(p_i) + (1 - t_i) \log(1 - p_i)
\]

(4)

where \(p_i\) is the probability of the \(i^{th}\) extracted watermark bit. \(t_i\) is the known target value of the logo bit at the \(i^{th}\) position. Hence \(t_i\) is 0 or 1. The details of implementation and chosen parameters are in Sect. 4.
4 Experimental results

The proposed method has been implemented on NVIDIA GeForce GTX 1080 Ti using Keras. We considered several experiments to show the performance of our method in terms of robustness and imperceptibility. The training phase is done using Pascal V.O.C. [10] and CIFAR10 [30]. Next, we use the Granada dataset [7], which contains 49 grayscale images, and the COCO dataset [34] to compare our results with other methods. Finally, our comparisons are made with well-known and concurrent works [2, 21, 60] to indicate its superiority.

The network structure and learning setup are explained in subsections 4.1 and 4.2. Subsections 4.3 reports the imperceptibility and robustness of the proposed method based on image processing metrics. We compare our approach with state-of-the-art techniques in subsection 4.4.

4.1 Network configuration

Based on Fig. 2, we have two networks in the embedding module. One network is for preparing the watermark data, and the second one is for embedding. The input image to WPN and embedding network have sizes 4×4 and 32×32, respectively. Hence, our cover image and watermark should be divided into blocks of these sizes and fed to networks as a batch of blocks. The WPN has four convolution layers. Each layer contains 64 filters with kernel size 2×2 and the same padding (each feature map has the same size as the input into the convolution layer), except the last layer that has one filter to concatenate to encoder output. Rectified Linear Unit (ReLU) is used as an activation function. The dilation convolution with size 2×2 is also used in each layer. The second network has the same network setting but with four convolution layers, and in the last layer, we only have one filter and ReLU activation. The strides of all filters in embedding and extraction modules are set to one.

The decoder part in Fig. 2 has five convolution layers with the same setting as the encoder part. In the last layer, the filter size is 1×1 with one filter and a sigmoid activation function.

Each attack, such as salt and pepper, Gaussian noise, Gaussian blurring, and JPEG, are implemented as a layer, and the corresponding setting is performed based on the attacks. However, for implementing the JPEG attack, we have to divide the input image into 8×8 blocks. We implemented this using a convolution layer by 64 filters with size 8×8 and stride eight. These filters make the DCT basis in JPEG compression.

The extraction network has nine convolution layers. Each layer contains 64 filters with kernel size 5×5 without padding, except the last layer, which has one filter and a sigmoid activation function to make the extracted watermark.

We use a 4×4 random binary string as the watermark data for embedding in a 32×32 block. However, based on the input image size, we replicate it to make the watermark the same size as the input image. For example, if the input images had a size of 512×512, we would need to divide them into 32×32 blocks. With this policy, the capacity will be N×16, where N is the number of blocks. With this image size N = 256, the overall capacity for embedding is 4096 bits. We can divide the watermark data and embed it into several blocks. If we have a watermark string with a lower length, we can use redundancy in embedding and employ voting to extract the watermark. Consequently, the accuracy will improve.
4.2 Learning setup

We use a combination of CIFAR10 and Pascal V.O.C. for training. The first dataset has 32×32 color images, and the second one includes high-resolution color images. Hence, we randomly extract 32×32 patches with smooth to high-frequency patterns from each image of Pascal V.O.C. After converting to gray, the combination of CIFAR10 and Pascal V.O.C. has about 334 K images. Ten percent of these images are used for the test phase. All images are normalized between [0, 1] before training.

Kernel size and learning rate are hyper-parameters of a network that affect on network’s performance. We can consider different scenarios for finding appropriate parameters. Here we first try to find the suitable kernel size. This helps us to have fewer searches by considering a specific optimizer. We consider stochastic gradient descent (SGD) and Adam [29] optimizers during training with their default learning rate and change the kernel size. SGD and Adam are standard optimizers in deep network learning, but Adam usually has a higher speed in converging to global optimal [29]. The accuracy of learned weights is evaluated on one of the standard watermark images [7]. We also consider equal coefficients for each part of the loss function (λ1 = λ2 = 1). With these settings, we try to find a suitable kernel size. Kernel size has an essential effect on network performance and parameter complexity. Increasing the kernel size will drastically increase the number of learning parameters. Table 1 shows the results of the proposed network with different kernel sizes. The PSNR and BER values are calculated as the average of five repetitions. Table 1 indicates that the kernel sizes (2,2) and (5,5) by the Adam optimizer have better results than others.

After selecting kernel size and optimizer, we try to find a suitable learning rate. Table 2 shows the results of the network with different settings. When the learning rate is decreased, the network has better results on training data, but our result will not be acceptable on test data due to overfitting. If the learning rate is high, our network cannot learn,
and we will see a low performance. As we can observe, the network has better imperceptibility and robustness by Adam optimizer with the default value of learning rate (0.001).

Loss function coefficients are another hyper-parameter in the network. Increasing each coefficient will improve the related output, negatively affecting the other results. Hence, selecting suitable coefficients that resolve the trade-off between two outputs is vital. Since the image size is larger than the watermark, assigning a larger coefficient to the watermark image output is reasonable. Based on Tables 1 and 2, the kernel size and optimizer are (2,2), (5,5), and Adam ($lr = 0.001$), respectively, in this experiment. We calculate $\lambda_1$ and $\lambda_2$ experimentally and they are 0.95 and 0.05, respectively.

Table 3 shows the final settings which are used in our network. In the test phase, the end-to-end trained model is split into two parts without the attack layer, and the embedding and extraction networks are considered two distinct networks, as shown in Figs. 2 and 4. We embed random 1024-bit (32×32) watermarks in 512×512 grayscale images to evaluate the trained networks. The cover image has a capacity of 4096 bits so that the watermark can be embedded with four times redundancy. Each watermark bit is repeated four times in a regular pattern in this plane to make a 64×64 bit redundant plane. The cover image and watermark plane are partitioned into 32×32 sub-images and 4×4 sub-watermarks for feeding to the embedding network based on network input size. The embedding network’s output is watermarked sub-images that are tiled with each other to form the watermarked image. Then the watermarked image is passed through several attacks to evaluate the robustness of the proposed method. The extraction phase has the same procedure for preparing the attacked image. First, the attacked watermarked image is divided into 32×32 sub-images and is fed into the extraction network, which extracts 4×4 patches of the watermark plane. Afterward, these partitions are tiled to make a 64×64 redundant watermark plane. Eventually, a voting procedure is applied to the extracted watermark plane to produce the 1024-bit watermark data.

### 4.3 Visual quality and robustness

Examples of the watermarked images that have been produced with the proposed method are shown in Fig. 5. We used 1024-bit watermark data for embedding, and PSNR and SSIM values are indicated for each image. As we can see, the reported values are significant and demonstrate that our network has successfully performed the embedding with the lowest artifacts.

| Description                          | Parameters | value  |
|--------------------------------------|------------|--------|
| Training image patch size            | (W,H)      | (32,32) |
| Watermark size                       | (w,h)      | (4,4)  |
| Embedding-Extraction kernel size     | (m,n)      | (5,5)  |
| WPN kernel size                      | (m,n)      | (2,2)  |
| Learning rate                        | lr         | 0.001  |
| Optimizer                            | -          | Adam   |
| Iteration number                     | Itr        | $10^6$ |
| Loss function weights                | $\lambda_1$, $\lambda_2$ | 0.95,0.05 |
To evaluate the robustness of the proposed framework, we consider common image processing attacks such as Gaussian noise ($\sigma = \{5, 15, 25\}$), median filter ($s = \{3, 5\}$), sharpening ($\rho = \{1, 5, 10\}$), Gaussian blurring ($\rho = \{1, 1.6, 2\}$), salt-pepper ($\pi = \{2, 6, 10\}$), and geometrical attacks such as crop ($\pi = \{10, 20, 30\}$), grid-crop ($\pi = \{10, 20, 30\}$), dropout, and resizing ($0.5, 0.75, 1.5$). The value of $\sigma$ in the Gaussian noise attack indicates the standard deviation of the noise. The value of $s$ in the median filter attack is the size of the filter. The value of $\pi$ in crop, grid-crop, and salt-pepper attacks is the percentage of changed pixels. Grid crop is a new attack that randomly selects $8 \times 8$ blocks of image and changes them to zero [49]. It can evaluate the method’s performance for watermark scattering and data sharing. The mentioned scale in the resize attack resizes the image for resizing attack and then resizes back to the original size using bilinear interpolation. BER is used as an evaluation measure. To have more accurate results, we consider 20 random watermark strings and repeat our experiments for each watermark, and then the final output is the average value of all tests. We calculate the average value for ten repetitions for attacks such as Gaussian noise, salt-pepper, and grid-crop. The results can be different each time.

To show the performance of our method, we report the BER values against different attacks for some standard images such as Barbara, Lena, Baboon, Living room, and Bridge. The average value of BER for all images in the Granada database [7] is also indicated in

![Fig. 5 The visual quality of the watermarked image, (a) original image, (b) watermarked image](image-url)
our reports. The watermark has 1024 bits in length, and the input images’ size is 512×512. Based on the network structure, the image will be divided into 32×32 blocks, and finally, we have 256 blocks that each have 16 bits capacity. Hence, in total, we will have 4096 bits capacity for embedding. We use this extra space for redundant embedding. The redundancy can be different based on the input image size and the watermark size.

The extracted watermark can be achieved using voting regardless of the input image and watermark size. First, we aggregate the watermark string as (5):

\[
V = \sum_{i=1}^{rn} W(i)
\]  

(5)

After aggregating to achieve the final watermark, we use voting as (6):

\[
W' = \begin{cases} 
1 & \text{if } V_i > \frac{rn}{2} \\
0 & \text{o.w}
\end{cases}
\]  

(6)

where \( rn \) is the number of redundancy. For input image of size \( \text{img}_H \times \text{img}_W \), the number of blocks will be \( N = (\text{img}_H \times \text{img}_W)/(32 \times 32) \) And the total capacity is \( C = 16 \times N \). Assuming the length of a watermark is \( l \), so \( rn = C/l \). Our proposed method does not have a dependency on input image or watermark size, and it is enough that its size is a multiple of 32. Tables 4 and 5 show the percentage of BER values against image processing attacks.

Based on the results, especially against crop and grid-crop attacks, we can conclude that our network has a good ability to distribute the watermark in all parts of the cover image. The proposed method could extract a considerable percentage of the watermark data by deleting random parts of the input image. Figure 6 shows image processing attacks on the Bridge image to show our method’s performance. The BER values are also reported.

### 4.4 Comparison with state-of-the-art

Several new watermarking methods based on deep networks have been investigated to evaluate the proposed method’s performance [2, 21, 60]. Each technique has used a different dataset and watermark length in their results; therefore, we use suitable watermark lengths based on their setting. Nevertheless, the proposed and comparable methods should have the same or near visual perceptuality for fair comparisons.

The method presented in [60] is an end-to-end network that uses a GAN network to improve the reconstructed image quality. In this method, the watermark is a vector of length L. The input image also has dimensions of \( M \times M \), and the watermark is repeated for each image pixel and is combined with the input image as a tensor with dimensions of \( M \times M \times L \). In this case, each image pixel has the information of all watermark bits, which reduces the image quality. They used the COCO database in their results, which has dimensions of 128×128, and the watermark’s length is equal to 30. The images are color images in the YCbCr space.

To compare our method with [60], we used similar cover images and a random 32-bit watermark. After converting the cover image to YCbCr space, each color channel is input to our proposed network. Finally, the final watermark is calculated by voting between extracted watermarks from each color channel.

The main idea in [2] is to use the DCT layer as a preprocessing phase in their structure. They change the input image into a vector with the size \( MN \times 1 \). Then after vectorizing the
Table 4  Imperceptibility and robustness of proposed method based on BER, PNSR, and SSIM

|                      | Imperceptibility | Robustness- BER (%) |                      |                      |
|----------------------|------------------|---------------------|---------------------|---------------------|
|                      | PSNR                  | SSIM                | Sharpening (radius) | JPEG  |
|                      |                     |                     | 1  | 5  | 10  | 90 | 70 | 50  | 3  | 5  | 2  | 6  | 10  |
| Barbara              | 37.14                | 0.980               | 0.006  | 0.009  | 0.013  | 0.148  | 0.310  | 0.364  | 0.372  | 0.379  | 0.070  | 0.144  | 0.237  |
| Living room          | 38.55                | 0.980               | 0.004  | 0.010  | 0.012  | 0.137  | 0.301  | 0.379  | 0.365  | 0.402  | 0.061  | 0.174  | 0.240  |
| Bridge               | 36.77                | 0.985               | 0.004  | 0.026  | 0.040  | 0.056  | 0.252  | 0.338  | 0.386  | 0.412  | 0.058  | 0.132  | 0.215  |
| Baboon               | 35.96                | 0.982               | 0.003  | 0.023  | 0.022  | 0.042  | 0.219  | 0.325  | 0.428  | 0.450  | 0.071  | 0.156  | 0.245  |
| Lena                 | 39.34                | 0.976               | 0.006  | 0.007  | 0.012  | 0.186  | 0.353  | 0.372  | 0.393  | 0.409  | 0.076  | 0.196  | 0.296  |
| Granada [7]          | 40.34                | 0.991               | 0.001  | 0.008  | 0.014  | 0.062  | 0.293  | 0.379  | 0.373  | 0.491  | 0.048  | 0.185  | 0.341  |
Table 5: Imperceptibility and robustness of proposed method based on BER, PNSR, and SSIM

|                | Imperceptibility | Robustness- BER (%) |
|----------------|------------------|---------------------|
|                | PSNR      | SSIM    | Gaussian noise | Gaussian blurring (radius) | Crop (%) | Grid-crop (%) | No-attack |
|                |           |         | 5   | 15  | 25  | 1   | 1.6 | 2  | 10 | 20 | 30 | 10 | 20 | 30 |
| Barbara        | 37.14     | 0.980   | 0.059 | 0.246 | 0.256 | 0.357 | 0.371 | 0.442 | 0.037 | 0.050 | 0.133 | 0.034 | 0.116 | 0.169 | 0.004 |
| Living room    | 38.55     | 0.980   | 0.067 | 0.277 | 0.276 | 0.378 | 0.376 | 0.385 | 0.023 | 0.053 | 0.125 | 0.047 | 0.110 | 0.210 | 0.006 |
| Bridge         | 36.77     | 0.984   | 0.045 | 0.216 | 0.220 | 0.350 | 0.387 | 0.386 | 0.018 | 0.079 | 0.103 | 0.018 | 0.097 | 0.177 | 0.005 |
| Baboon         | 35.96     | 0.982   | 0.032 | 0.208 | 0.228 | 0.316 | 0.371 | 0.394 | 0.035 | 0.084 | 0.127 | 0.043 | 0.135 | 0.216 | 0.013 |
| Lena           | 39.34     | 0.976   | 0.077 | 0.316 | 0.343 | 0.334 | 0.371 | 0.372 | 0.029 | 0.050 | 0.114 | 0.061 | 0.120 | 0.229 | 0.002 |
| Granada [7]    | 40.34     | 0.991   | 0.051 | 0.327 | 0.405 | 0.281 | 0.416 | 0.473 | 0.025 | 0.066 | 0.110 | 0.001 | 0.004 | 0.005 | 0.002 |
watermark, they concatenate the watermark vector with the image vector. The final vector is fed into the DCT layer, and the DCT coefficients are used in their deep network. Adding the input image to the end of the encoder module as a residual helps their method to have better imperceptibility. They also used a strength factor in the test phase to control the strength of watermark embedding. Table 6 shows the proposed method’s output against different attacks compared to other methods using the COCO database. SSIM and PSNR values of our approach are nearly the same as the methods in Table 6 for a fair comparison. In a crop-out attack, part of an image is randomly selected and replaced with part of the noise image. In a dropout attack, some pixels of the image are randomly selected and replaced with a pixel of noise image.

As we can see, our method has better results against crop, crop-out, and JPEG attacks than [60]. Compared to [2], which uses accurate quantized coefficients in their method, our

| BER       | JPEG (50%) | Crop-out (30%) | Dropout (30%) | Crop (3.5%) | Gaussian Filter (σ = 2) | Method |
|-----------|------------|----------------|---------------|-------------|------------------------|--------|
| 0.370     | 0.060      | 0.070          | 0.120         | 0.040       | [60]                   |        |
| 0.254     | 0.075      | 0.080          | 0.000         | 0.500       | [2]                    |        |
| 0.263     | **0.017**  | 0.093          | **0.000**     | 0.490       | Proposed method        |        |
method has a considerable result against the crop-out attack and comparable results against other attacks. However, the Gaussian filter [60] has a better output, which can be due to having a separate network for different attacks. Because our proposed network is multi-attack during learning, it cannot have better results against all attacks.

Table 7 shows another comparison with [21] and [2] that have near PSNR (ours = 40.34, [2] = 40.24, [21] = 32.5) and SSIM (ours = 0.991, [2] = 0.987, [21] = 0.973). They used the Granada dataset. The watermark length is 1024 bits, and the input image size is 512 × 512. The method in [21] has the indicated value for PSNR 32.5, but our PSNR is 40.3, so our method with higher PSNR has considerable BER values for the JPEG attack. Our method performs better in some attacks, such as grid-crop, which randomly deletes some parts of the image. This shows that the watermark is distributed inside the image; by removing some parts, other parts of the image would still have the information. Our network is just trained on JPEG, Gaussian noise, and salt-pepper attacks, but based on Tables 6 and 7, our method has considerable results against other attacks such as crop, dropout, crop-out, and grid-crop. All these attacks try to remove part of the image and replace it with different data. This means part of

| Attack            | model | Ours | [2]  | [21] |
|-------------------|-------|------|------|------|
| Gaussian noise (%)| 5     | 0.051| 0.417|-     |
|                   | 15    | 0.327| 0.456|-     |
|                   | 25    | 0.405| 0.469|-     |
| Crop (%)          | 10    | 0.025| 0.077|-     |
|                   | 20    | 0.066| 0.131|-     |
|                   | 30    | 0.111| 0.188|-     |
| Salt-pepper (%)   | 2     | 0.028| 0.029|-     |
|                   | 6     | 0.185| 0.045|-     |
|                   | 10    | 0.341| 0.091|-     |
| Grid-crop (%)     | 20    | 0.001| 0.086|-     |
|                   | 30    | 0.004| 0.130|-     |
|                   | 40    | 0.005| 0.180| 0.170 |
| JPEG (%)          | 90    | 0.062| 0.093| 0.243 |
|                   | 70    | 0.293| 0.164| 0.289 |
|                   | 50    | 0.379| 0.256|-     |
| Gaussian bluring (%)| 1  | 0.281| 0.086|-     |
|                   | 1.6   | 0.416| 0.390|-     |
|                   | 2     | 0.473| 0.500|-     |
| MED (%)           | 3     | 0.373| 0.134|-     |
|                   | 5     | 0.491| 0.500|-     |
| Sharpening (%)    | 1     | 0.001| 0.009|-     |
|                   | 5     | 0.008| 0.024|-     |
|                   | 10    | 0.014| 0.032|-     |
| Resize (%)        | 0.5   | 0.390| 0.212|-     |
|                   | 0.75  | 0.250| 0.038|-     |
|                   | 1.5   | 0.048| 0.026|-     |
the hidden information will remove. However, due to expanding the watermark into all parts of the image, other parts of the image can help retrieve the watermark data by deleting some parts of the watermark. Our method does not have better results against resizing in comparison to [2]. We do not consider resizing during training, and other used attacks are not similar to or near this attack. Therefore, our network is not very good in this attack.

The compared methods in Tables 6 and 7 are deep network-based [2, 21, 60]. Tables 8 and 9 show a comparison of the proposed method and frequency domain methods presented in [13, 16, 25], and [3]. In all subsequent comparisons, the string length is 128 bits, and the image size is 512×512. In [25], the DCT-DWT domain and in [3], the contourlet domain are used for embedding. As demonstrated in Table 8, the performance of the proposed method in the salt-pepper attack has been much better and with less error than the other two methods.

The proposed method in [13] also used the contourlet domain and embedding in the crowded places of the image. As can be seen in Table 9, the proposed method performed better in all attacks than the [13]. Moreover, in comparison with [16], which used the Wavelet domain, the performance of our method has been better in most attacks. But in the JPEG attack, because the deep method uses estimation for the quantization part, compared to [16], it has obtained higher BER values, which is expected.

### 4.5 Time complexity

We consider the time complexity for the test phase. We only load the learned weights into the encoder and decoder network during the test phase. The proposed method can be implemented in real-time for images with size 512×512. The embedding and extraction times with GPU GTX 1080 Ti are respectively 0.0051 s and 0.0032 s, that is, in total, 0.0083 s. This timing will increase to 0.1166 s using a Core i5 1.8 GHz system.

To have a fair comparison with other methods, we have to use the same hardware or a weaker system. Hence, we implement our method on a system with lower specs, and the considerable difference between our proposed and other methods demonstrates the superiority of our method. Table 10 indicates the execution time of some watermarking methods [2, 13, 25]. The timing results show that the proposed embedding and extraction networks are comparable to or faster than state-of-the-art methods. All of the mentioned methods embed in 512×512 grayscale images.

| Table 8 | Robustness of the proposed method based on BER in [3, 25], and the proposed method |
|---------|---------------------------------|
| image   | Salt and pepper attack |
|         | 1%   | 3%   | 5%   |
|         | Ours | [25] | [3]  | Ours | [25] | [3]  | Ours | [25] | [3]  |
| Barbara | 0.00 | 0.04 | 3.28 | **0.09** | 0.32 | 4.18 | **0.13** | 1.29 | 4.80 |
| Baboon  | 0.02 | **0.00** | 1.91 | **0.09** | 0.38 | 2.81 | **0.13** | 0.71 | 3.79 |
| Couple  | 0.00 | 0.15 | 6.72 | **0.06** | 0.62 | 8.55 | **0.16** | 5.28 | 10.35 |
| Bridge  | 0.01 | **0.00** | 5.43 | **0.05** | 0.46 | 7.27 | **0.15** | 1.38 | 9.10 |
| Map     | 0.02 | **0.00** | 7.7  | 0.05 | **0.00** | 9.26 | **0.14** | 0.26 | 10.98 |
Table 9  Robustness of the proposed method based on BER in [13, 16], and the proposed method

| Image   | Method   | JPEG | S&P 1% | Crop 20% | sharpening | Rotation 0.5° | MED (3,3) |
|---------|----------|------|--------|----------|------------|---------------|-----------|
|         |          | 10   | 20     | 30       | 40         | 50            | 70        | 80     | 90     | 10     | 20     | 30     | 40     | 50     | 70     | 80     | 90     | 10    | 20    | 30    | 40    | 50    | 70    | 80    | 90    | 10    | 20    | 30    | 40    |
| Bridge  | ours     | 0.32 | 0.32   | 0.35     | 0.27       | 0.27          | 0.28        | 0.19   | 0.11   | 0.09   | 0.00   | 0.00   | 0.00   | 0.00   | 0.34  | 0.32  |        |       |       |       |       |       |       |       |       |       |       |       |
|         | [16]     | 0.23 | 0.08   | 0.03     | 0.01       | 0.00          | 0.00         | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.53  | 0.20  |       |       |       |       |       |       |       |       |       |       |       |
|         | [13]     | 0.42 | 0.38   | 0.36     | 0.34       | 0.36          | 0.36         | 0.36   | 0.35   | 0.35   | 0.36   | 0.39   | 0.36   | 0.36   | 0.47  | 0.38  |       |       |       |       |       |       |       |       |       |       |       |
| Barbara | ours     | 0.32 | 0.34   | 0.32     | 0.26       | 0.27          | 0.23         | 0.20   | 0.19   | 0.07   | 0.00   | 0.00   | 0.00   | 0.00   | 0.38  | 0.38  |       |       |       |       |       |       |       |       |       |       |       |
|         | [16]     | 0.33 | 0.19   | 0.07     | 0.01       | 0.00          | 0.00         | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.61  | 0.13  |       |       |       |       |       |       |       |       |       |       |       |
|         | [13]     | 0.43 | 0.41   | 0.38     | 0.37       | 0.38          | 0.36         | 0.36   | 0.35   | 0.35   | 0.40   | 0.40   | 0.36   | 0.36   | 0.49  | 0.38  |       |       |       |       |       |       |       |       |       |       |       |
| Lena    | ours     | 0.38 | 0.34   | 0.31     | 0.29       | 0.34          | 0.34         | 0.31   | 0.39   | 0.18   | 0.00   | 0.01   | 0.00   | 0.00   | 0.40  | 0.40  |       |       |       |       |       |       |       |       |       |       |       |
|         | [16]     | 0.35 | 0.30   | 0.14     | 0.00       | 0.00          | 0.00         | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.60  | 0.13  |       |       |       |       |       |       |       |       |       |       |       |
|         | [13]     | 0.47 | 0.42   | 0.40     | 0.38       | 0.38          | 0.38         | 0.36   | 0.36   | 0.35   | 0.40   | 0.41   | 0.37   | 0.37   | 0.48  | 0.38  |       |       |       |       |       |       |       |       |       |       |       |
| Couple  | ours     | 0.31 | 0.32   | 0.31     | 0.35       | 0.26          | 0.30         | 0.24   | 0.20   | 0.06   | 0.00   | 0.00   | 0.00   | 0.00   | 0.36  | 0.36  |       |       |       |       |       |       |       |       |       |       |       |
|         | [16]     | 0.33 | 0.15   | 0.04     | 0.00       | 0.00          | 0.00         | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.52  | 0.11  |       |       |       |       |       |       |       |       |       |       |       |
|         | [13]     | 0.41 | 0.40   | 0.40     | 0.39       | 0.36          | 0.36         | 0.36   | 0.35   | 0.35   | 0.37   | 0.39   | 0.35   | 0.35   | 0.48  | 0.39  |       |       |       |       |       |       |       |       |       |       |       |
| Baboon  | ours     | 0.30 | 0.30   | 0.32     | 0.34       | 0.25          | 0.20         | 0.20   | 0.06   | 0.00   | 0.02   | 0.00   | 0.00   | 0.00   | 0.32  | 0.38  |       |       |       |       |       |       |       |       |       |       |       |
|         | [16]     | 0.19 | 0.04   | 0.01     | 0.00       | 0.00          | 0.00         | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.06   | 0.48  | 0.09  |       |       |       |       |       |       |       |       |       |       |       |
|         | [13]     | 0.40 | 0.38   | 0.36     | 0.35       | 0.35          | 0.35         | 0.36   | 0.36   | 0.36   | 0.36   | 0.00   | 0.00   | 0.00   | 0.34  | 0.32  |       |       |       |       |       |       |       |       |       |       |       |
| Map     | ours     | 0.35 | 0.23   | 0.23     | 0.16       | 0.12          | 0.08         | 0.02   | 0.02   | 0.01   | 0.02   | 0.00   | 0.00   | 0.02   | 0.34  | 0.38  |       |       |       |       |       |       |       |       |       |       |       |
|         | [16]     | 0.09 | 0.05   | 0.02     | 0.00       | 0.00          | 0.00         | 0.00   | 0.00   | 0.00   | 0.00   | 0.07   | 0.07   | 0.00   | 0.56  | 0.25  |       |       |       |       |       |       |       |       |       |       |       |
|         | [13]     | 0.38 | 0.36   | 0.36     | 0.34       | 0.34          | 0.35         | 0.35   | 0.35   | 0.35   | 0.40   | 0.36   | 0.47   | 0.36   |       |       |       |       |       |       |       |       |       |       |       |       |
Conclusion

This paper proposes an end-to-end trained deep network for blind watermarking. The network contains two main modules: embedding and extraction. The embedding’s responsibility is to make the input image ready for watermarking. The encoder part contains two networks that operate in parallel. One network accepts the input image, and after convolution layers, the high-level features of the image are extracted and ready for combination with a watermark.

The second network accepts a watermark string and prepares the watermark based on the input image content. Finally, the watermark network’s output is replicated to generate a matrix of the same size as the input image. The network’s outputs are concatenated and sent to the decoder network. Adding attack as a differentiable layer into the proposed structure helps the network to perform better against attacks and prevents the network from memorizing. Due to having a multi-object network with conflicting goals, the loss function tries to resolve the trade-off between robustness and imperceptibility of the outputs. The proposed network is a multi-attack network, which means we consider a combination of different attacks. Multi-attack design helps the network to perform better against attacks during the test phase. Comparative results against the state-of-the-art methods demonstrate the superiority of the proposed method in robustness and visual perceptuality.

Data Availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

References

1. Abdelhakim AM, Abdelhakim M (2018) A time-efficient optimization for robust image watermarking using machine learning. Expert Syst Appl 100:197–210
2. Ahmadi M, Norouzi A, Soroushmehr SM, Karimi N, Najarian K, Samavi S, Emami A (2020) ReD-Mark: Framework for residual diffusion watermarking based on deep networks. Expert Syst Appl 146:113–157
3. Akhaee A, Sahraeian SM, Marvasti F (2010) Contourlet based image watermarking using optimum detector in noisy environment. IEEE Trans Image Process 19(4):967–980
4. Anbarjafari G, Ozcinar C (2018) Imperceptible non-blind watermarking and robustness against tone mapping operation attacks for high dynamic range images. Multimed Tools Appl 77:24521–24535
5. Broughton RS, Laumeister WC (1989) “Interactive video method and apparatus,” United States Patent 4,807,031
6. Cheng Y (2001) “Music database retrieval based on spectral similarity,” 2nd Int. Symposium on Music Information Retrieval (IS-MIR)
7. Dataset of standard 512x512 grayscale test images. Available: https://ccia.ugr.es/cvg/CIG/base.htm. Accessed Oct 2021
8. Etemad S, Amirmazlaghani M (2018) A new multiplicative watermark detector in the contourlet domain using t location-scale distribution. Pattern Recognition 77:99–112
9. Etemad E, Samavi S, Soroushmehr SR, Karimi N, Etemad M, Shirani S, Najarian K (2017) Robust image watermarking scheme using bit-plane of Hadamard coefficients. Multimed Tools Appl 77:1–23
10. Everingham M, Eslami SA, Van Gooy L, Williams CK, Winn J, Zisserman A (2015) The pascal visual object classes challenge: A retrospective. Int J Comput Vision 111:98–136
11. Fan B, Li Z, Gao J (2021) DwiMark: a multiscale robust deep watermarking framework for diffusion-weighted imaging images. Multimed Syst 28:1–16
12. Faundez-Zanuy M, Hagemüller M, Kubin G (2007) Speaker identification security improvement by means of speech watermarking. Pattern Recogn 40(11):3027–3034
13. Fazlali HR, Samavi S, Karimi N, Shirani S (2017) Adaptive blind image watermarking using edge pixel concentration. Multimed Tools Appl 76:3105–3120
14. Gong LH, Tian C, Zou WP, Zhou NR (2021) Robust and imperceptible watermarking scheme based on Canny edge detection and SVD in the contourlet domain. Multimed Tools Appl 80(1):439–461
15. Gupta BB (2020) An efficient KP design framework of attribute-based searchable encryption for user level revocation in cloud. Concurr Comput Pract Exp 32(18):5291
16. Hamghalam M, Mirzakuchaki S, Akhaee MA (2017) Geometric modeling of the wavelet coefficients for image watermarking using optimum detector. IET Image Proc 8(3):162–172
17. Hatami E, RashidyKanan H, Layeghi K, Harounabadi A (2022) An optimized robust and invisible digital image watermarking scheme in Contourlet domain for protecting rightful ownership. Multimed Tools Appl 70:1–31
18. Heidari M, Samavi S, Soroushmehr SMR, Shirani S, Karimi N, Najarian K (2017) Framework for robust blind image watermarking based on classification of attacks. Multimed Tools Appl 76:23459–23479
19. Horng SJ, Rosiyadi D, Fan P, Wang X, Khan MK (2014) An adaptive watermarking scheme for e-government document images. Multimed Tools Appl 72:3085–3103
20. Hu F, Cao H, Chen S, Sun Y, Su Q (2022) “A robust and secure blind color image watermarking scheme based on contourlet transform and Schur decomposition”. Vis Comput, pp.1–20
21. Hua G, Zhao L, Zhang H, Bi G, Xiang Y (2018) Random matching pursuit for image watermarking. IEEE Trans Circuits Syst Video Technol 29(3):625–639
22. Ioffe S, Szegedy C (2015) “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” In International conference on machine learning, pp. 448–456
23. Jagadeesh B, Kumar PR, Reddy PC (2015) Fuzzy inference system based robust digital image watermarking technique using discrete cosine transform. Proced Comput Sci 46:1618–1625
24. Jagadeesh B, Rajesh Kumar P, Chenna Reddy P (2016) Robust digital image watermarking based on fuzzy inference system and backpropagation neural networks using DCT. Soft Comput 20:3679–3686
25. Jamalil M, Rafei S, Soroushmehr SM, Karimi N, Shirani S, Najarian K, Samavi S (2018) Adaptive image watermarking using human perception based fuzzy inference system. J Intell Fuzzy Syst 35(4):4589–4608
26. Jin C, Wang, S (2007) “Applications of a neural network to estimate watermark embedding strength,” In Eighth International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS’07), pp. 68–68
27. Kandi H, Mishra D, Gorthi SRS (2017) Exploring the learning capabilities of convolutional neural networks for robust image watermarking. Comput Secur 65:247–268
28. Khan A, Tahir SF, Majid A, Choi T-S (2008) Machine learning based adaptive watermark decoding in view of anticipated attack. Pattern Recogn 41(8):2594–2610
29. Kingma DP, Ba J (2014) “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980
30. Krizhevsky A, Nair V, Hinton G, The CIFAR-10 dataset, Available: http://www.cs.toronto.edu/kriz/cifar.html
31. Kumar D, Chahar V (2011) Digital Image Watermarking: A Review of SVD, DCT, and DWT Based Approaches. INFOCOMP J Comput Sci 10(3):25–35
32. Kumar V, Kumar D (2019) Performance evaluation of modified color image steganography using discrete wavelet transform. J Intell Syst 28(5):749–758
33. Li J, Yu C, Gupta BB, Ren X (2018) Color image watermarking scheme based on quaternion Hadamard transform and Schur decomposition. Multimed Tools Appl 77:4545–4561
34. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) “Microsoft coco: Common objects in context,” In European conference on computer vision, pp. 740–755
35. Liu J, He X (2005) “A review study on digital watermarking,” In Proc. IEEE International Conference on Information and Communication Technology, pp. 337–34
36. Makbol NM, Khoo BE, Rassem TH, Loukhaoukha K (2017) A new reliable optimized image watermarking scheme based on the integer wavelet transform and singular value decomposition for copyright protection. In Form Sci 417:381–400
37. Moad MS, Kafi MR, Khaldi A (2022) A wavelet based medical image watermarking scheme for secure transmission in telemedicine applications. Microprocess Microsyst 90:104490
38. Mun SM, Nam SH, Jang HU, Kim D, Lee HK (2017) “A robust blind watermarking using convolutional neural network,” arXiv preprint arXiv:1704.03248
39. Nezhadarya E, Wang ZJ, Ward RK (2011) Robust image watermarking based on multiscale gradient direction quantization. IEEE Trans Inf Forensics Secur 6(4):1200–1213
40. Parah SA, Sheikh JA, Loan NA, Bhat GM (2016) Robust and blind watermarking technique in DCT domain using inter-block coefficient differencing. Digit Signal Process 53:11–24
41. Plata M, Syga P (2020) “Robust Spatial-spread Deep Neural Image Watermarking,” In 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 62–70
42. Potdar VM, Han S, Chang E (2005) “A survey of digital image watermarking techniques,” In Proc. IEEE International Conference on Industrial Informatics, pp. 709–716
43. Rashid A (2016) Digital watermarking applications and techniques: A brief review. Int J Comput Appl Technol Res 5(3):147–150
44. Rasti P, Anbarjafari G, Demirel H (2017) “Colour image watermarking based on wavelet and QR decomposition,” In Proc. IEEE Signal Processing and Communications Applications Conference, pp. 1–4
45. Rezaee K, SaberiAnari M, Khosravi MR (2022) “A Wavelet-Based Robust Medical Image Watermarking Technique Using Whale Optimization Algorithm for Data Exchange Through Internet of Medical Things,” In Springer Intelligent Healthcare, pp. 373–394
46. Sadreazami H, Ahmad MO, Swamy MN (2016) Multiplicative watermark decoder in contourlet domain using the normal inverse Gaussian distribution. IEEE Trans Multimed 18(2):196–207
47. Savakar DG, Ghuli A (2017) Non-blind digital watermarking with enhanced image embedding capacity using Dmeyer wavelet decomposition, SVD and DFT. Pattern Recognit Image Anal 27:511–517
48. Shehab A, Elhoseny M, Muhammad K, Sangaiah AK, Yang P, Huang H, Hou G (2018) Secure and robust fragile watermarking scheme for medical images. IEEE Access 6:10269–10278
49. Singh RP, Dabas N, Chaudhary V (2016) Online sequential extreme learning machine for watermarking in DWT domain. Neurocomputing 174:238–249
50. Solachidis V, Ioannis P (2001) Circularly symmetric watermark embedding in 2-D DFT domain. IEEE Trans Image Process 10(11):1741–1753
51. Szepanski W (1979) “A signal theoretic method for creating forgery-proof documents for automatic verification.” In Proc. Carnahan Conference on Crime Countermeasures, pp. 101–109
52. Tagliasacchi M, Valenzise G, Tubaro S (2009) Hash-based identification of sparse image tampering. IEEE Trans Image Process 18(11):2491–2504
53. Taherinia AH, Jamzad M (2009) A robust spread spectrum watermarking method using two levels DCT. Int J Electron Secur Digit Forensics 2(3):280–305
54. Tian C, Wen RH, Zou WP, Gong LH (2019) Robust and blind watermarking algorithm based on DCT and SVD in the contourlet domain. Multimed Tools Appl 79:1–27
55. Wang R-Z, Lin C-F, Lin J-C (2001) Image hiding by optimal LSB substitution and genetic algorithm. Pattern Recogn 34(3):671–683
56. Yu F, Koltun V (2016) “Multiscale context aggregation by dilated convolutions,” Int Conf Learn Represent (ICLR)
57. Yu C, Li J, Li X, Ren X, Gupta BB (2018) Four-image encryption scheme based on quaternion Fresnel transform, chaos and computer generated hologram. Multimed Tools Appl 77(4):4585–4608
58. Yu C, Li X, Xu S, Li J (2021) Computer generated hologram-based image cryptosystem with multi-chaotic systems. Wirel Netw 27(5):3507–3521
59. Zhang H, Shu H, Coatireux G, Zhu J, Wu QJ, Zhang Y, Zhu H, Luo L (2011) “Affine Legendre moment invariants for image watermarking robust to geometric distortions,” IEEE Transactions on Image Processing, pp. 2189–2199
60. Zhu J, Kaplan R, Johnson J, Fei-Fei L (2018) “Hidden: Hiding data with deep networks,” In Proceedings of the European conference on computer vision (ECCV), pp. 657–672

61. Zong T, Xiang Y, Natgunanathan I, Guo S, Zhou W, Beliakov G (2014) Robust histogram shape-based method for image watermarking. IEEE Trans Circ Syst Vid Technol 25(5):717–729

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

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.