A Robust Blind Watermarking
Using Convolutional Neural Network

Seung-Min Mun, Seung-Hun Nam, Han-Ul Jang, Dongkyu Kim, and Heung-Kyu Lee

Abstract—**This paper introduces a blind watermarking based on a convolutional neural network (CNN). We propose an iterative learning framework to secure robustness of watermarking. One loop of learning process consists of the following three stages: Watermark embedding, attack simulation, and weight update. We have learned a network that can detect a 1-bit message from a image sub-block. Experimental results show that this learned network is an extension of the frequency domain that is widely used in existing watermarking scheme. The proposed scheme achieved robustness against geometric and signal processing attacks with a learning time of one day.**

Index Terms—**digital watermarking, color image watermarking, blind watermarking, convolutional neural network(CNN).**

I. INTRODUCTION

WATERMARKING is used to identify and protect ownership of copyrighted media content, by embedding invisible data into the original content. The most difficult requirements for watermarking schemes is robustness; the detector should be able to extract the watermark properly even if the content has little distortion. The second requirement is invisibility; the visual quality should not be damaged too much by the mark.

Over the last decade, most watermarking techniques have acquired robustness using the frequency domain. Generally, the first step is to use a transform such as DCT or DWT to create a set of values corresponding to the original pixel data. Then some of the values in the transformed domain are modified and become the pixel data by inverse transform [1], [2]. In recent years, quaternion discrete Fourier transform (QDFT) performs best for blind watermarking techniques for color images as shown in [3]–[5]. These techniques are robust against signal processing attacks by applying QDFT for small image blocks. For geometric attacks, the robustness was secured with the help of a template. However, since their templates are limited to compensating for RST attacks, they are vulnerable to general affine transformation.

In this paper, we optimize robustness while considering invisibility in depth. Our goal is to learn convolutional neural network (CNN) beyond QDFT and replace it. In fact, watermark detection using QDFT is representable by a very shallow network. According to [4], each component of QDFT can be obtained as a linear combination of DFT coefficients on each color channel. Most of recent techniques use quantization index modulation (QIM) [3]–[5], which can be represented by one ReLU and two fully connected layer back and forth. Here, adjacent linear layers can be replaced by one linear layer. Taken together, QDFT and QIM is like combining three layers in a linear, ReLU, and linear order as shown in Figure 1.
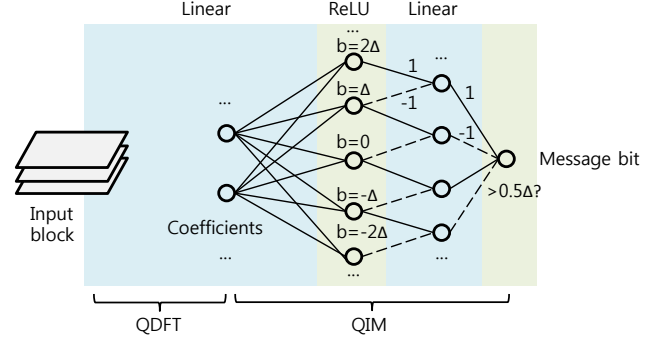

Fig. 1. A network representation of an existing watermarking domain. The same logic holds for a linear transform such as DFT or DCT as well as QDFT. Where $\triangle$ is the quantization step of QIM.

Therefore, CNN with a sufficiently large number of layers and filters can be seen as a generalization of QDFT and QIM. The proposed CNN model has 12 ReLU layers alternating with linear layers. In addition, the proposed model parameters are optimized for learning while QDFT is fixed. Unlike QDFT, the proposed scheme has domain specialized to watermarking.

As shown in Figure 2, we propose a light-weight and simple framework using a CNN. And this iterative process obtains a robust domain based on the given attack such as JPEG, resizing and noise addition. As a result, when new attacks are given, our framework can adaptively re-acquire robustness.

Signal processing attacks and geometric attacks were tested experimentally. The comparison with the QDFT-based watermarking paper [5] are shown. Experimental results show that the learning-based domain can surpass the existing frequency domain used for watermarking.

II. PROPOSED CNN BASED WATERMARKING

A. Main idea of learning framework

The goal of this iterative process is to construct a CNN model that takes an $R \times C$ size block as input and determines a message bit. This CNN is then used in a similar way to QDFT in traditional frequency domain watermarking techniques [3]. Training comprises three stages starting from a CNN with random weights. This three stages are repeated until the watermark that CNN embedded is correctly identified after the attack. In this subsection we provides details for each stage.

1) Watermark embedding using CNN: The first stage is the process of inserting a watermark into the image using the CNN of the previous loop. This stage accepts images and watermarks of $M \times N$ size and generates watermarked images. First, images and the corresponding watermarks are
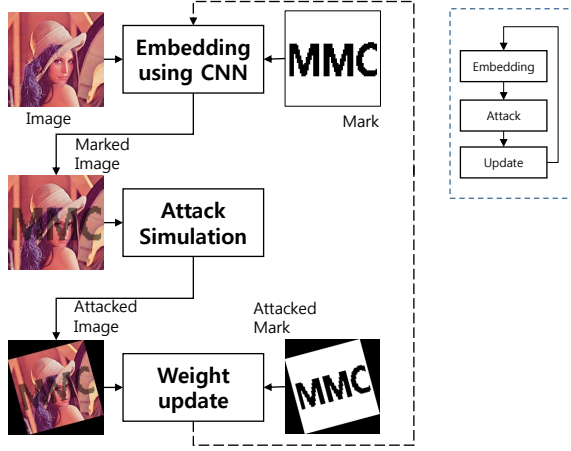
Fig. 2. Framework of learning CNN for watermarking. The embedded mark is shown as visible for better understanding. At the top right corner is a simplified illustration
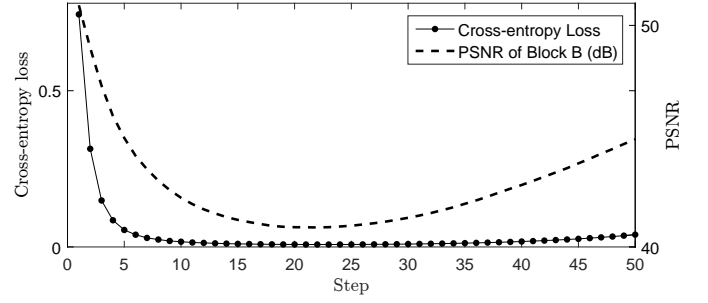

Fig. 3. Changes in Cross-entropy loss and invisibility according to the embedding step. The points are determined by the average of the 128 blocks. A description of the PSNR is given in Section III.
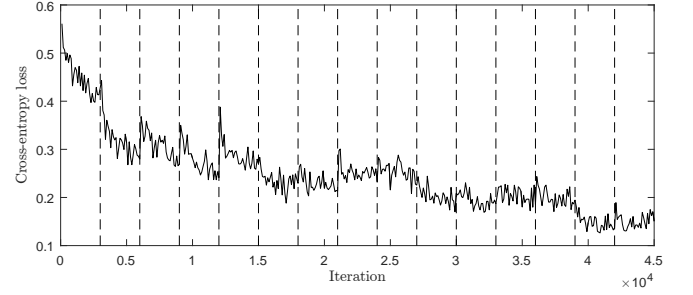

Fig. 4. Cross-entropy loss in the updating stage. The vertical lines divide each update stage and indicate that it has passed the embedding and attack stages.

divided into non-overlapping blocks of $R \times C$ size. The message bit m of each block $\mathbf{B}$ is determined by looking at the watermark image of the corresponding block position. We want to incrementally change the image so that CNN recognizes block $\mathbf{B}$ as m. We update the image by taking the gradient decent method (SGD), which is designed to update the weights:

$$\mathbf{B}^{(t+1)} = \mathbf{B}^{(t)} - \alpha \nabla_{\mathbf{B}} L(\mathbf{W}, \mathbf{B}^{(t)}, m) \qquad (1)$$

$$L(\mathbf{W}, \mathbf{B}, m) = -\log p(m|\mathbf{W}, \mathbf{B}) + \frac{1}{2}\lambda \|\mathbf{B} - \mathbf{B}_0\|_2^2 \qquad (2)$$

where $\mathbf{B}^{(t)}$ is the block to embedding at iteration t, α is the embedding rate, $\mathbf{W}$ is the weights of the CNN, L is the loss function and includes a cross-entropy loss term as in Equation 4. The loss function was devised to protect the invisibility while embedding the watermark. $p(m|\mathbf{W}, \mathbf{B})$ is the probability that message m is inserted into $\mathbf{B}$ predicted by CNN. We introduced the l^2 regularization term $\| \bullet \|_2$ to ensure invisibility.

Empirically, we have confirmed that if α is sufficiently small, the loss gradually decreases as shown in Figure 3 even in the case of random weights. A loss close to 0 means that the message m is properly inserted into block $\mathbf{B}$. The blocks are then combined to produce a watermarked image. Note that the weight $\mathbf{W}$ does not change in this stage.

2) Attack simulation: Attack simulation is necessary for CNN to adaptively capture invariant features for various attacks. This stage accepts the watermarked image to produce the attacked image. The important thing is that not only the watermarked cover image but also the watermark images are attacked. This attacked watermark is used as a true label for supervised learning in the next stage. The simulated attack sets include affine transform, cropping, JPEG compression, noising, Gaussian filtering, median filtering, rotation, rescaling. For each image and watermark, a pair of attacked images is created as many times as the number of attacks.

3) CNN weight update: Now we update the CNN weights so we can correctly extract the watermark from the given image. That is, CNN itself has to imitate the role of the detector. This stage begins by setting the size of all the attacked image to $M \times N$. As before, divide the image into non-overlapping blocks of $R \times C$ size. Again, The message bit m' of each distorted block $\mathbf{B}'$ is determined by looking at the watermark image of the corresponding block position. If the average value of the watermark image block is grater than 0.5, we set the message bit m' as 1 and set as 0 otherwise. We applied the gradient descent method so that when CNN takes $\mathbf{B}'$ as an input, it correctly predicts m'.

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \nabla_{\mathbf{W}} L(\mathbf{W}^{(t)}, \mathbf{B}', m'), \qquad (3)$$

where $\mathbf{W}^{(t)}$ is the weights at iteration t, η is the learning rate. The loss function, L, is identical to that of the embedding stage. At the end of this stage, all watermarked and attacked images created so far are deleted and the process returns to the embedding stage.

The lower the loss, the more CNN can correctly extract message bits regardless of attack. In other words, robustness against various attack is secured. We measured the loss every time the stage was repeated. Figure 4 shows empirical reduction of the loss by the proposed framework.

B. Use of trained CNN

If CNN's weights parameters converged sufficiently due to the learning framework, this subsection describes how CNN can be used for embedder and detector of applicable watermarking. The roles and functions of the embedder are described in detail in the previous subsection. The embedding

process for learning and the actual application are exactly the same. However, the CNN used in this case has a weight obtained at the end of learning. The CNN as a detector extracts a 1-bit message for each $R \times C$ block as before. The CNN predicts the message from the block of the attacked image. If the probability of 0 as a result of the softmax layer is larger than 0.5, the message of the block is decided to 0. Otherwise, the message bit is decided to 1. As a result, a watermark of $(M/R) \times (N/C)$ size is extracted for one image.

C. Implementation details

We propose a CNN model modified from a part of the most powerful model called Residual Network [6], which includes residual units and batch normalization. The proposed CNN contains 14 convolution layers and the kernel size of each convolution layer is 1×1 or 3×3. The number of filters in each convolution layer was 128. A soft-max layer is added at the end for classification. Generally, the more layers, the better the detection performance, but there is a trade-off where the feed-forwarding and the back-propagation time increase. The number of layers and filters was determined experimentally. Dropout was not used because it can be replaced by batch normalization. Pooling is aimed at ensuring spatial invariance, but it causes performance degradation to capture fine patterns. So we did not use max-pooling or average-pooling.

The image data used for training consisted of 4000 24-bit colour images from the BOSSBase dataset designed for data hiding studies [7]. Without loss of generality, we set $M = N = 512$ and $R = C = 8$. Initially, α and η were set to 0.01 and 0.001 respectively. We used 8 randomly sampled images per embedded stage and 15 types of attacks are used. For one image, we used a pair of watermarks, 0 and 1 being inverted from each other. Each image is divided into 4096 blocks so $8 \times 2 \times 15 \times 4096$ blocks are used in one updating stage. The update unit, batch, consists of 128 blocks and is sampled from the attacked images.

In practice, we used adaptive moment estimation (Adam) instead of SGD because Adam is more complicated but has better loss reduction. Also, the gradient in Equation 1 is normalized with mean and standard deviation for invisibility and fast convergence. Inspired by learning rate annealing, we reduced the α by 0.9 times per iteration. Iteration t for each blocks in the embedding phase is limited to 12.

III. EXPERIMENTAL RESULTS

The network was trained using a GPU, NVIDIA GTX 1070 for one day, and test images and test watermarks were not used for the training. Attack experiments were performed using StirMark benchmark [8] for JPEG, noising and affine transforms. For the remaining attacks, Python libraries scikit-image [9] and scipy [10] were used.

We conducted robustness testing for signal processing attacks and geometric attacks. The comparison technique is QDFT based watermarking. One bit is inserted into the QDFT mid-frequency coefficients of four components by QIM method. Existing techniques [3], [5] estimate the rotation angle, the moving distance, and the scaling factor at the

Fig. 5. Visual impact comparison before and after watermark embedding. The top row is the original test images and the bottom row is the watermarked test images.

detection step using the inserted template. We assumed that the RST correction through this estimation was performed without error to compare performance with existing QDFT, as [4] did. Table I shows the result of registering an attacked image first, and we implemented the comparison technique. To make the embedding capacity the same as the proposed technique, the comparison technique inserts one bit in one block as [5] does. For robustness comparison, both techniques omitted the scrambling of the watermark.

Like the related works [3]–[5], [11], standard test images Baboon, Lenna and Peppers were used as cover images. They are all 24 bit color images and are 512×512 in size as shown in Figure 5. MPEG-7 CE Shape-1 [12], a binary image data set, was processed for use as a watermark image.

The standard deviation of the kernel used in Gaussian filtering is 1. Median filtering was performed for each channel with a 3×3 kernel. We experimented with six types of affine transformations that move the pixel coordinates a bit as in Stirmark benchmark [8]:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = M_i \begin{bmatrix} x \\ y \end{bmatrix} \tag{4}$$

$$M_1 = \begin{bmatrix} 1 & 0 \\ 0.01 & 1 \end{bmatrix}, \qquad M_2 = \begin{bmatrix} 1 & 0.01 \\ 0 & 1 \end{bmatrix},$$

$$M_3 = \begin{bmatrix} 1 & 0.01 \\ 0.01 & 1 \end{bmatrix}, \qquad M_4 = \begin{bmatrix} 1.01 & 0.013 \\ 0.009 & 1.011 \end{bmatrix},$$

$$M_5 = \begin{bmatrix} 1.007 & 0.01 \\ 0.01 & 1.012 \end{bmatrix}, \qquad M_6 = \begin{bmatrix} 1.013 & 0.008 \\ 0.011 & 1.008 \end{bmatrix}$$

where M_i is a matrix corresponding to transformation number i. x and y are the pixel coordinates before the transformation, and x' and y' are the coordinates after the transformation.

Robustness and invisibility performance were measured by normalized correlation (NC) and peak signal-to-noise ratio (PSNR), respectively.

$$PSNR(\mathbf{I}, \mathbf{I}') = 10 \log_{10}\{\frac{255^2 \times 3MN}{\|\mathbf{I} - \mathbf{I}'\|_2^2}\} \tag{5}$$

$$NC(\mathbf{w}, \mathbf{w}') = \frac{\langle \mathbf{w}, \mathbf{w}' \rangle}{\|\mathbf{w}\|_2 \|\mathbf{w}'\|_2} \tag{6}$$

TABLE I
COMPARISON OF THE EXTRACTED WATERMARKS WITH REGISTRATION

	Attacks	QDFT [5]			Training A1			Training A1-A8			Training A1-A16		
		Baboon	Lenna	Peppers	Baboon	Lenna	Peppers	Baboon	Lenna	Peppers	Baboon	Lenna	Peppers
A1	No attack (PSNR)	35.7dB	37.2dB	37.4dB	38.9dB	38.9dB	39.0dB	35.1dB	38.3dB	38.8dB	35.9dB	39.2dB	38.9dB
A2	JPEG 80	0.5008	0.7673	0.8313	-0.0372	0.1326	0.2127	0.5305	0.5826	0.6578	0.6384	0.8963	0.8938
A3	JPEG 90	0.6880	0.8143	0.8850	-0.0252	0.1886	0.2253	0.7684	0.8571	0.9251	0.9129	0.9676	0.9614
A4	Median filtering	0.6596	0.9460	0.9760	0.1951	0.3221	0.3322	0.7189	0.8461	0.8751	0.8377	0.8987	0.8985
A5	Gaussian filtering	0.4746	0.7933	0.8947	0.4374	0.4631	0.6218	0.9076	0.9238	0.9454	0.9625	0.9343	0.9590
A6	Affine 1	0.3062	0.3148	0.3229	0.1955	0.2459	0.3128	0.9673	0.9831	0.9910	0.8971	0.9738	0.9451
A7	Affine 2	0.3036	0.3143	0.3201	0.2281	0.2330	0.2482	0.9857	0.9858	0.9933	0.8428	0.9417	0.9236
A8	Affine 3	0.1402	0.0477	0.0185	0.1016	0.3122	0.4088	0.8381	0.8918	0.9005	0.9532	0.9032	0.9090
A9	Affine 4	0.1212	0.0803	-0.0105	0.1019	0.3083	0.3735	0.7197	0.7952	0.8446	0.9207	0.8343	0.8794
A10	Affine 5	0.1324	0.0899	-0.0217	0.0239	0.2248	0.3587	0.8093	0.8654	0.8822	0.9450	0.9055	0.9125
A11	Affine 6	0.1494	0.0753	0.0057	-0.0257	0.1149	0.2585	0.7785	0.7996	0.8744	0.9337	0.8631	0.8661
A12	Noising	0.9220	0.8880	0.9077	0.9704	0.9863	0.9962	0.6594	0.7167	0.9076	0.9737	0.8784	0.9084
A13	Resizing 75%	0.8425	0.9539	0.9780	0.7081	0.8002	0.8360	0.9903	0.9934	0.9970	0.9891	0.9947	0.9961
A14	Rotation 10°	0.6721	0.8733	0.9252	0.5918	0.6425	0.7367	0.9857	0.9934	0.9970	0.9749	0.9781	0.9817
A15	Rotation 90°	1.0000	1.0000	1.0000	0.9967	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
A16	Cropping 80%	0.8362	0.9910	0.9745	0.8740	0.9868	0.9910	0.8746	0.9921	1.0000	0.8730	1.0000	0.9921

where $\mathbf{I}$ and $\mathbf{I}'$ denote the original cover image and the cover image after embedding, respectively. $\mathbf{w}$ and $\mathbf{w}'$ denote the bit sequences of the original watermark and the watermark detected after the attack, respectively. $\langle \cdot, \cdot \rangle$ means the inner product. NC and PSNR are the most widely used performance indicators as shown in previous watermarking papers [2], [5], [11].

Table I shows the test results according to the attack set to be trained. Two observations are derived from the third column. First, the attacks that are included in the training set show higher robustness than those that are not. Second, The robustness of the third column tends to be lower than the robustness of the rightmost column. We infer the cause of this anomaly as certain attacks will help to find robust features for other attacks when added to training. Based on two observations, it is obvious that we have to train enough of the various sets of attacks. In this paper, therefore, the rightmost column is proposed.

IV. DISCUSSION AND RELATED WORK

The proposed scheme assume that it is used with existing template matching methods. So we can not get robust correlation results against rotation and cropping without assuming registration. The extracted watermarks in the absence of registration are as shown in table II. The computational complexity of proposed method is higher than that of QIM after QDFT and the time required for detection is longer. Experimentally, the proposed technique took about 1.6s to detect watermark from one image and the comparison technique took 0.5s using a CPU i7-4770k.

The first CNN-based watermarking technique is proposed by [11]. Unlike the proposed technique, [11] uses CNN as an auto-encoder to present a new domain. However, [11] is a non-blind method. Therefore, although the robustness

TABLE II
COMPARISON OF THE EXTRACTED WATERMARKS WITHOUT REGISTRATION

Attacks	QDFT [5]			Proposed		
	Baboon	Lenna	Peppers	Baboon	Lenna	Peppers
Rotation 90° (NC)						
	0.0104	-0.0190	0.0062	0.1129	0.4309	0.2496
Rotation 10° (NC)						
	0.1059	0.1193	0.1109	0.4352	0.4710	0.5505
Cropping 80% (NC)						
	0.0198	0.0258	0.0638	0.5062	0.5363	0.5381

of these techniques is high, they are less useful in real scenarios. The proposed technique, on the other hand, is a blind watermarking.

V. CONCLUSION

We proposed a robust blind watermarking technique using CNN. Experiments have yielded higher PSNR and NC values than comparative methods for test images. The learned models outperformed QDFT with assuming registration. Traditional watermarking researches first design the transform, and then attack and measure performance. In contrast, the proposed scheme optimized the watermarking domain through attack functions. Above all, the scheme does not require detailed algorithms of each attack or expert knowledge to counter them. Our technique was adaptive to the attacks and required only one day of learning time.

REFERENCES

[1] X. Kang, J. Huang, Y. Q. Shi, and Y. Lin, "A dwt-dft composite watermarking scheme robust to both affine transform and jpeg compression," *IEEE transactions on circuits and systems for video technology*, vol. 13, no. 8, pp. 776–786, 2003.

[2] S. D. Lin and C.-F. Chen, "A robust dct-based watermarking for copyright protection," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 3, pp. 415–421, 2000.

[3] X.-y. Wang, C.-p. Wang, H.-y. Yang, and P.-p. Niu, "A robust blind color image watermarking in quaternion fourier transform domain," *Journal of Systems and Software*, vol. 86, no. 2, pp. 255–277, 2013.

[4] B. Chen, G. Coatrieux, G. Chen, X. Sun, J. L. Coatrieux, and H. Shu, "Full 4-d quaternion discrete fourier transform based watermarking for color images," *Digital Signal Processing*, vol. 28, pp. 106–119, 2014.

[5] J. Ouyang, G. Coatrieux, B. Chen, and H. Shu, "Color image watermarking based on quaternion fourier transform and improved uniform log-polar mapping," *Computers & Electrical Engineering*, vol. 46, pp. 419–432, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[7] P. Bas, T. Filler, and T. Pevnỳ, " break our steganographic system: The ins and outs of organizing boss," in *International Workshop on Information Hiding*. Springer, 2011, pp. 59–70.

[8] F. A. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Attacks on copyright marking systems," in *International workshop on information hiding*. Springer, 1998, pp. 218–238.

[9] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014.

[10] E. Jones, T. Oliphant, and P. Peterson, "{SciPy}: open source scientific tools for {Python}," 2014.

[11] H. Kandi, D. Mishra, and S. R. S. Gorthi, "Exploring the learning capabilities of convolutional neural networks for robust image watermarking," *Computers & Security*, vol. 65, pp. 247–268, 2017.

[12] L. Latecki, "Shape data for the mpeg-7 core experiment ce-shape-1," *Web Page: http://www. cis. temple. edu/~ latecki/TestData/mpeg7shapeB. tar. g z*, 2002.