Article

Not peer-reviewed version

A Novel Self-Recovery Fragile Watermarking Scheme Based on Convolutional Autoencoder

Chin-Feng Lee , Tong-Ming Li , Iuon-Chang Lin * , Anis Ur Rehman *

Posted Date: 13 August 2025

doi: 10.20944/preprints202508.1011.v1

Keywords: image authentication; self-recoverable fragile watermarking; convolutional autoencoder; blind watermarking

Article

A Novel Self-Recovery Fragile Watermarking Scheme Based on Convolutional Autoencoder

Chin-Feng Lee [1], Tong-Ming Li [2], Iuon-Chang Lin [2,*] and Anis Ur Rehman [3,*]

[1] Department of Information Engineering and Computer Science, Feng Chia University, Taichung City 407102, Taiwan (R.O.C.)
[2] Department of Management Information Systems, National Chung Hsing University, Taichung City 402202, Taiwan (R.O.C.)
[3] Department of Information Management, Chaoyang University of Technology, Taichung City 413310, Taiwan (R.O.C.)
* Correspondence: iclin@nchu.edu.tw (I.-C.L.); rehmananis771@ustb.edu.pk (A.U.R.)

Abstract

In the digital era where images are easily accessible, concerns about image authenticity and integrity are increasing. To address this, we propose a deep learning-based fragile watermarking method for secure image authentication and content recovery. The method utilizes bottleneck features extracted by the convolutional encoder to carry both authentication and recovery information, and employs deconvolution at the decoder to reconstruct image content. Additionally, the Arnold Transform is applied to scramble feature information, effectively enhancing resistance to collage attacks. At the detection stage, block voting and morphological closing operations improve tamper localization accuracy and robustness. Experiments tested various tampering ratios, with performance evaluated by PSNR, SSIM, Precision, Recall, and F1-score. Experiments under varying tampering ratios demonstrate that the proposed method maintains high visual quality and achieves reliable tamper detection and recovery, even at 75% tampering. Evaluation metrics including PSNR, SSIM, Precision, Recall, and F1-score confirm the effectiveness and practical applicability of the method.

Keywords: image authentication; self-recoverable fragile watermarking; convolutional autoencoder; blind watermarking

1. Introduction

1.1. Research Background

With rapid technological advancement, digital information has become deeply embedded in daily life. A smartphone enables us to capture special moments, preserving them as lasting digital records. The widespread use of digital imaging enables everyone to document and witness life. Meanwhile, advancements in information and communication technology (ICT) have fueled the growth of social media platforms like Facebook, Instagram, and Threads, connecting people worldwide and turning images into bridges for emotional exchange and interaction.

Beyond recording and sharing, images have become vital tools for justice. In judicial and administrative contexts, photos and videos—such as citizen recordings of crimes—serve as crucial evidence for investigation and trials. When individual rights are violated, images serve as factual evidence and offer reliable legal and social proof emphasizing their vital role in today's world.

However, the rise of deepfake technology increasingly threatens image authenticity, producing highly realistic forged images and facial videos. Meanwhile, accessible editing tools like Microsoft Paint, Adobe software, and Canva offer users easy ways to enhance images but can also be misused to alter content maliciously, fabricating falsehoods that mislead the public and judicial systems, damaging reputations and societal trust. Since images are often regarded as factual evidence, their credibility is now under serious threat, making image integrity verification a critical topic in current research and technology.

Previous research divides image authentication methods into active and passive approaches, as shown in Figure 1 [1]. Active methods generate and store features before transmission. Common techniques include digital signatures [2], encryption [3], and watermarking [4–8]. The receiver compares extracted features to verify integrity. Watermarking is categorized into Fragile watermarks, which detect minor changes and localize tampering, suitable for integrity verification [9–13], Robust watermarks resistant to noise and compression, used mainly for copyright protection [10], and Semi-fragile watermarks balancing tamper detection and resistance to common image processing [14].

Passive methods require no prior embedding and verify authenticity solely at the receiver by detecting traces left by image processing or tampering. Also called blind methods, passive approaches can be further divided based on processing domain into Internal consistency and irregularity analysis, tampering detection tailored to or independent of forgery types, and handling natural images versus AI-generated images.

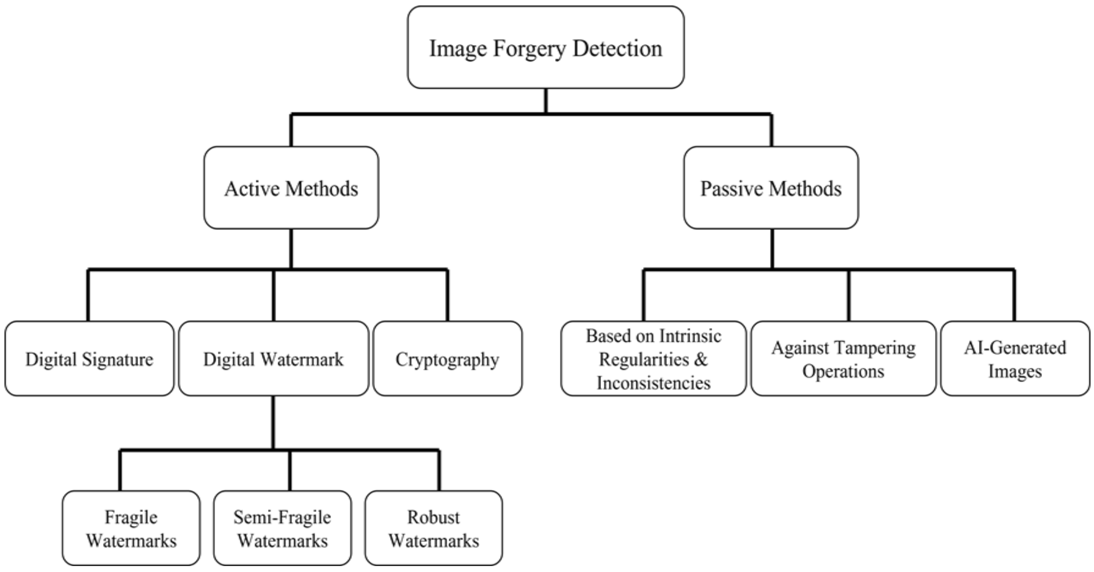


Figure 1. Image Authentication Methods.

Compared to passive authentication, active authentication methods offer superior tamper localization capabilities. Fragile watermarks can precisely mark altered image regions, making them ideal for sensitive fields like forensic and medical imaging. Active methods also enable real-time verification by allowing receivers to compare embedded features to check image integrity. In contrast, passive authentication relies on statistical and feature analysis, which is computationally intensive, slower, and less precise in locating tampering.

In the domains of digital signatures, watermarking, and cryptography, fragile watermarking often incorporates image self-recovery technology. This not only detects tampered regions but can restore damaged parts to closely approximate the original image [15]. Such recovery capabilities significantly enhance the utility of fragile watermarking in maintaining image authenticity and preserving content integrity.

1.2. Research Motivation and Objectives

In recent years, deep learning has rapidly advanced, with Convolutional Neural Networks (CNNs) [16] becoming a key technology in image processing. CNNs automatically learn and extract image features through hierarchical structures, offering greater adaptability and representational power than traditional handcrafted methods. Their core is the convolution operation, which uses local receptive fields and weight sharing to capture low-level features like edges and textures, and combines these into higher-level semantic information such as shapes and contours. Pooling reduces feature dimensionality, improving efficiency and translation invariance. CNNs outperform fully connected networks in generalization and classification tasks, making them central to modern deep learning architectures.

Convolutional Autoencoders (CAEs) are an extension of CNNs used for feature extraction and image compression/reconstruction [17]. Composed of an encoder and decoder, the encoder applies convolution and pooling to transform input images into low-dimensional latent representations that retain essential information. These latent features are analogous to principal components in PCA, enabling data compression and improved processing efficiency.

The decoder reconstructs the image using deconvolution or up sampling to restore the features to the original size, producing a reconstruction close to the input image.

The compressed feature representations produced by the encoder can be applied to various downstream tasks [18], such as anomaly detection, image denoising, and feature learning. When combined with a decoder, these features also support extensions like super-resolution, image inpainting, and Generative Adversarial Networks (GANs) to enhance image quality and realism.

The powerful feature extraction capabilities of CNNs, combined with the compression-reconstruction structure of CAEs, make them well-suited for fragile watermarking applications that require both authentication and self-recovery. Features generated by CAEs can represent both authentication and recovery information efficiently, enhancing the watermarking scheme's robustness and effectiveness.

This study proposes a fragile watermarking image authentication method based on a deep learning CAE architecture. The method not only detects and localizes tampering but also autonomously recovers the tampered regions.

Section 2 reviews related work, including the block-pixel authentication method by Lee et al. [13], the SVD-based approach by Shen et al. [19], the fundamentals of CNNs and CAEs, and Rezaei's CNN-based image recovery method [20]. Section 3 presents the proposed methodology and framework, Section 4 shows the experimental design and results, and Section 5 concludes with a summary and future directions.

2. Related Work

This Section explores existing methods and techniques proposed by researchers, including MSB, LSB, and SVD approaches. After reviewing these non-deep learning methods, we further investigate deep learning-based techniques. First, Convolutional Neural Networks (CNNs) are introduced, followed by Convolutional Autoencoders. Finally, Rezaei's CNN-based image watermark authentication method is discussed [20].

2.1. Symbol Definitions

To clearly explain the methods proposed by previous researchers, this subsection first provides complete definitions of the relevant symbols, as shown in Table 1.

Table 1. Symbol Definitions.

No.	Notation	Description
(1)	I_O	the original image
(2)	W	the weight of the original image
(3)	H	the height of the original image
(4)	$B_i, i = (1, 2, \ldots, N)$	every block in the original image
(5)	$m \times n$	the size of a block
(6)	N	the total number of blocks in an image
(7)	SK	the secret key (Generate block-mapping sequence)
(8)	V	bottleneck
(9)	$\|\|V\|\|$	the bottleneck length is represented in bytes

(10)	V_{2d}	reshape the bottleneck into a 2D format
(11)	v	take the square root of the length of the bottleneck (to convert it into the side length of a 2D shape) i.e., $\|\|V\|\| = v^2$
(12)	$BM_i\ , i = (1, 2, \dots , N)$	the mapping block
(13)	R_i	recovery code of each block
(14)	$M_{rm_i}\ , i = (1, 2, \dots , N)$	the mapping block recovery data
(15)	$A_i\ , i = (1, 2, \dots , N)$	the authentication code of each block
(16)	I_W	the watermarked image
(17)	I_T	the tampered image
(18)	$A'_i\ , i = (1, 2, \dots , N)$	the authentication message (receiver)
(19)	I_R	the recovered image
(20)	t	hyperparameter for designing the number of neurons
(21)	T	Number of arnold transform iterations

2.2. Fragile Watermarking

Rakhmawati et al. [15] described the structure of fragile watermarking, as shown in Figure 2. Fragile watermarking techniques are generally categorized into two types:

(1) Pixel-wise schemes, which extract features from individual pixels. These offer high localization accuracy but may reduce image quality due to the large amount of embedded data.

(2) Block-wise schemes, which divide the image into non-overlapping blocks. Although this method may falsely mark some untampered pixels, it improves image quality by reducing the amount of embedded data.

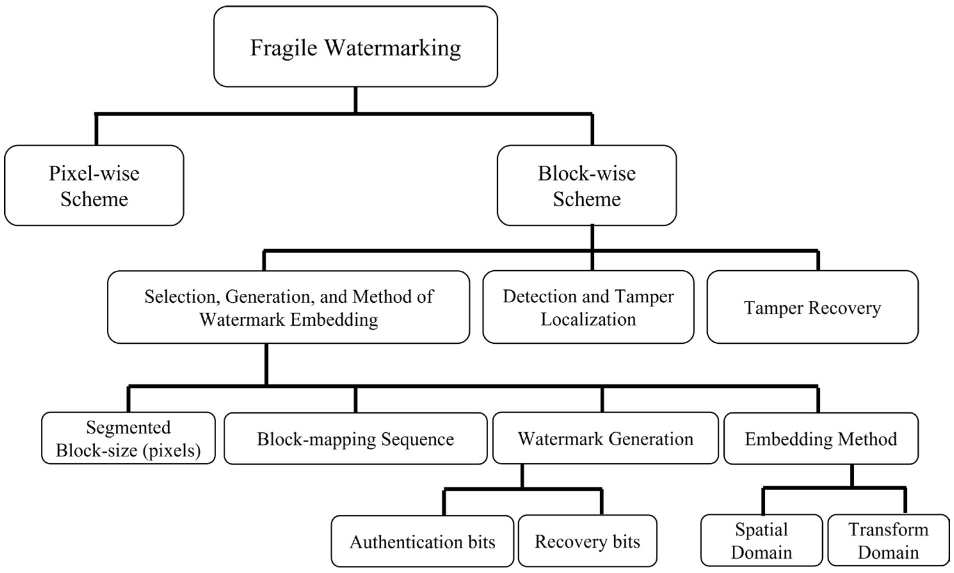


Figure 2. Fragile Watermarking Framework.

In block-based schemes, the process can be divided into three main stages: **Process 1:** Selection, Generation, and Method of Watermark Embedding. **Process 2:** Detection and Tamper Localization. **Process 3:** Tamper Recovery.

Among these, the first stage receives the most focus, as it directly influences the effectiveness of tamper detection and recovery.

When selecting the watermark, block size (e.g., 2×2, 3×3, 4×4, or 8×8) should be considered, as different sizes affect performance and can be evaluated experimentally. Another key factor is the block-mapping sequence; improper mapping may store both authentication and recovery data in the same block, compromising tamper recovery. Common mapping techniques include Linear Transform [21,22] and Arnold Transform [23].

For watermark generation, two types of information are typically created. 1. Authentication bits to verify the integrity of each block. **2**. Recovery bits to restore tampered content.

Authentication bits can be generated using hash functions or Singular Value Decomposition (SVD) [13,19], while recovery bits are often based on block averages [13,19].

Finally, watermark embedding can be performed in either the spatial domain (e.g., Least Significant Bit, LSB) or the transform domain, using techniques such as Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT).

2.3. Block-Pixel Wised Image Authentication (BP Wised) and Singular Value Decomposition (SVD Based) Image Authentication

In 2019, Lee et al. proposed the Block-pixel (BP) Wise Image Authentication technique [13]. First, the image is divided into non-overlapping blocks, and each block has two parts to be processed. The first part is to calculate the authentication information A_i for each block B_i, while the other part is to calculate the average value of each block B_i to be used as R_i. The following describes the detailed process of both parts during the A_i and R_i generation and embedding phases.

Step 1. Divide the original image I_O into N non-overlapping blocks B_i of size $m \times n$ (i $= 1,2,\ldots,N$).

Step 2. Use the 6 MSBs of each block B_i to perform a hash operation to generate the authentication information A_i.

Step 3. Use the average value M_i of each block B_i as R_i.

Step 4. Use a random seed SK to generate N unique pseudo-random numbers from 1 to N, to determine which block B_i corresponds to another block BM_i.

Step 5. Record R_i of B_i as the 8-bit average value M_{rm_i} of the corresponding block BM_i.

Step 6. The watermark I_W for each block B_i is composed of the block's A_i and R_i.

Step 7. Use LSB substitution to embed I_W into the LSBs of block B_i.

Upon receiving the tampered image I_T , the receiver generates authentication codes A'_i for each block B_i and compares them with the original A_i. If $A'_i = A_i$, the block is considered untampered; otherwise, $A'_i \neq A_i$ indicates tampering.

The extracted recovery data R_i, a downscaled version of the original block, is used to restore missing pixels via bicubic interpolation.

In 2020, Shen et al. proposed an SVD-based image authentication method [19], which differs from the BP-Wised approach by using finer block division. This improves tampering detection accuracy, reducing both false positive and false negative rates (FPR and FNR).

The method consists of two main parts: watermark generation and embedding, and tampering detection and self-recovery, as detailed above.

The original image I_O is divided into $\mathrm{N} = (W \times H)/(m \times n)$ non-overlapping blocks B_i of size $m \times n$. The average value of each block $m \times n$ is calculated to obtain the recovery data R_i.

For the authentication data A_i, the two LSBs of every pixel in B_i are first set to zero. Then, B_i is split into four sub-blocks and combined vertically into upper and lower halves, BU_i and BL_i, respectively, using Equations (1) and (2). These represent the upper and lower parts of B_i.

Next, singular value decomposition (SVD) is applied to BU_i and BL_i, yielding matrices EU_i and EL_i as in Equation (3). Each singular value is converted to binary, then merged via XOR operations according to Equations (4) and (5).

Finally, the binary sequences from the upper half a_U and lower half a_L are concatenated to form the complete authentication message A_i. Both A_i and R_i are embedded into the two LSBs of B_i blocks, similar to the BP-Wised approach.

$$BU_i = vertical\ concat\{(b_{i1} \,||\, abs(b_{i1} - b_{i2})), (round\left(\frac{b_{i1} - b_{i2}}{2}\right) || \, b_{i2})\} \tag{1}$$

$$BL_i = vertical\ concat\{(b_{i3} \,||\, abs(b_{i3} - b_{i4})), (round\left(\frac{b_{i3} - b_{i4}}{2}\right) || \, b_{i4})\} \tag{2}$$

$$EU_i = \begin{bmatrix} a_{U1} & 0 & 0 & 0 \\ 0 & a_{U2} & 0 & 0 \\ 0 & 0 & a_{U3} & 0 \\ 0 & 0 & 0 & a_{U4} \end{bmatrix} \; and \;\; EL_i = \begin{bmatrix} a_{L1} & 0 & 0 & 0 \\ 0 & a_{L2} & 0 & 0 \\ 0 & 0 & a_{L3} & 0 \\ 0 & 0 & 0 & a_{L4} \end{bmatrix} \tag{3}$$

$$a_U = XOR(a_{U1}, a_{U2}, a_{U3}, a_{U4}) \tag{4}$$

$$a_L = XOR(a_{L1}, a_{L2}, a_{L3}, a_{L4}) \tag{5}$$

The receiver regenerates the authentication code A'_i using the same method and compares it with the embedded A_i stored in the two LSBs of block B_i. If the first 12 bits match but the last 12 bits differ, it indicates tampering in the lower half of the block, while the upper half remains intact; the reverse applies if the first 12 bits differ and the last 12 match. If both halves differ, the entire block B_i is considered unusable. Upon detecting tampering, the embedded average value is extracted and enlarged using bicubic interpolation to restore the tampered pixels.

2.4. Rezaei's Method [20]

Rezaei et al. proposed a self-recovery framework using CNNs, where authentication and recovery bits are processed by different networks. Authentication uses a fine-tuned VGG-16 model, while recovery employs the CNN-based compression network by Feng et al. [24]. The following sections describe (1) watermark generation and embedding, and (2) image tampering detection and recovery.

The watermark A_i is generated by fine-tuning a VGG-16 model. The input to the model is a 16×16 block B_i, with its two least significant bits (LSBs) removed to reserve space for embedding A_i and recovery data R_i. The fine-tuned VGG-16 outputs a 16-element vector, with each value quantized to 8 bits.

The recovery data R_i generation follows the approach proposed by Feng et al., as illustrated in Figure 3.

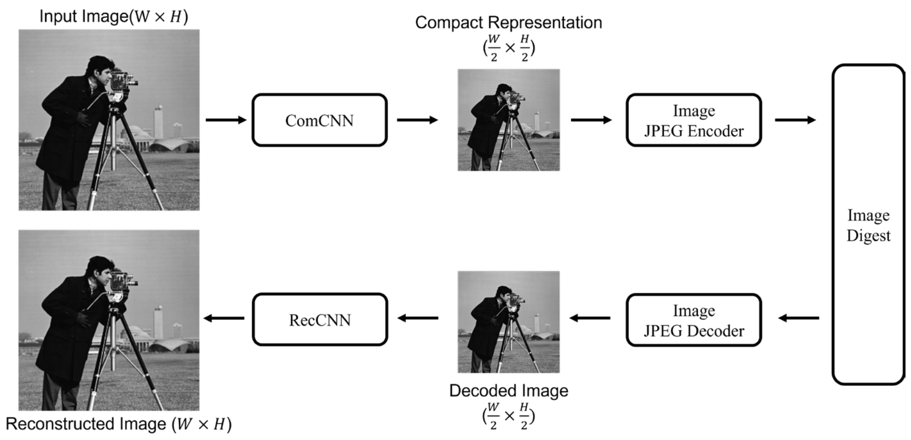


Figure 3. Recovery Data Generation using CNN-based ComCNN and RecCNN [20].

As ComCNN and RecCNN serve as the compression and decompression networks, respectively, both are designed based on convolutional neural networks (CNNs). ComCNN compresses the original image of size $(W \times H)$ into a reduced version of $(W/2 \times H/2)$ using convolutional operations. The image is then passed through a standard JPEG encoder and decoder, simulating traditional compression and decompression techniques.

The output from the JPEG encoder is organized in a zigzag scanning order, from which the first 10 coefficients are extracted. Each of these values is represented using 6 bits. These 10 coefficients are then encoded using Reed-Solomon (RS) codes for error correction, enabling recovery in case of tampering or data corruption.

To enhance security and resistance to collage attacks, the encoded data undergoes a structured scrambling process using the Arnold Transform. Finally, both the scrambled authentication data A_i and recovery data R_i are embedded into the two least significant bits (LSBs) of the output image I_O using a 2-LSB embedding technique.

In the second stage, tampering detection and recovery are performed. The receiver first uses a fine-tuned VGG-16 model to regenerate the authentication code A'_i. This regenerated code is then compared with the embedded A_i, which was stored in the two least significant bits (LSBs) of each block. If A' matches A_i, the corresponding block B_i is considered untampered. Otherwise, a mismatch indicates that B_i has been tampered with.

When tampering is detected, the embedded recovery data R_i , also stored in the 2 LSBs, is used to reconstruct the affected block. As shown in the lower part of Figure 3, the recovery process begins by applying inverse JPEG operations on R_i , generating a low-resolution image of size $(W/2 \times H/2)$. This image is then processed using RecCNN with transposed convolution to upscale and restore a high-resolution approximation of the original block B_i. The restored block is finally inserted back into the tampered region, completing the image recovery.

In summary, Section 2 reviewed the structure of fragile watermarking and key techniques including BP-Wise, SVD-based, and CNN-based methods. While Rezaei et al. achieved promising results with deep learning, their dual-network approach separates feature extraction and recovery, increasing design complexity.

In contrast, a Convolutional Autoencoder (CAE) allows shared feature representation for both tasks, simplifying model design and reducing embedding cost. The following section introduces our proposed CAE-based fragile watermarking and self-recovery scheme.

3. Proposed Method

Fragile watermarking is an effective technique for ensuring image integrity, supporting not only tamper detection and localization but also the recovery. To enhance restoration quality and leverage the strength of deep learning in feature extraction, this study proposes a fragile watermarking method based on a Convolutional Autoencoder (CAE).

The approach uses the CAE encoder to extract latent space features—high-level, compact representations of the image, referred to as bottleneck information. These features are embedded into non-overlapping image blocks. Before embedding, a random sequence generated from a secret key determines the number of Arnold Transform scrambling iterations per block, helping resist collage attacks.

At the receiver side, the embedded information is extracted and descrambled. A voting mechanism selects the most frequent bottleneck as the correct one, which is then decoded to reconstruct the image. Tampered blocks are replaced with restored versions, while untampered blocks remain unchanged, resulting in a fully recovered image.

The full process is detailed in Sections 3.1 and 3.2, with corresponding flowcharts provided in Figures 4 and 8.

3.1. Watermark Generation and Embedding

3.1.1. Encoder and Bottleneck

As shown in Figure 4, the watermark generation and embedding process begins by using an encoder to produce the bottleneck representation. The encoder is designed to align with the decoder used for tamper localization and self-recovery.

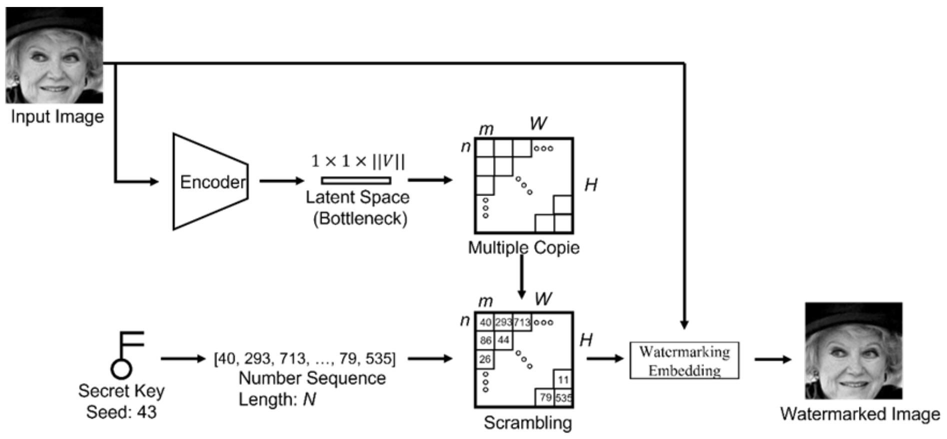


Figure 4. Watermark Generation and Embedding Flowchart.

The model architecture is inspired by the VGG11 network [25], but simplified to fit our specific task. Unlike the original VGG11, which is built for large-scale classification across 1,000 categories, our model does not require such complexity. The streamlined architecture of our encoder is illustrated in Figure 5.

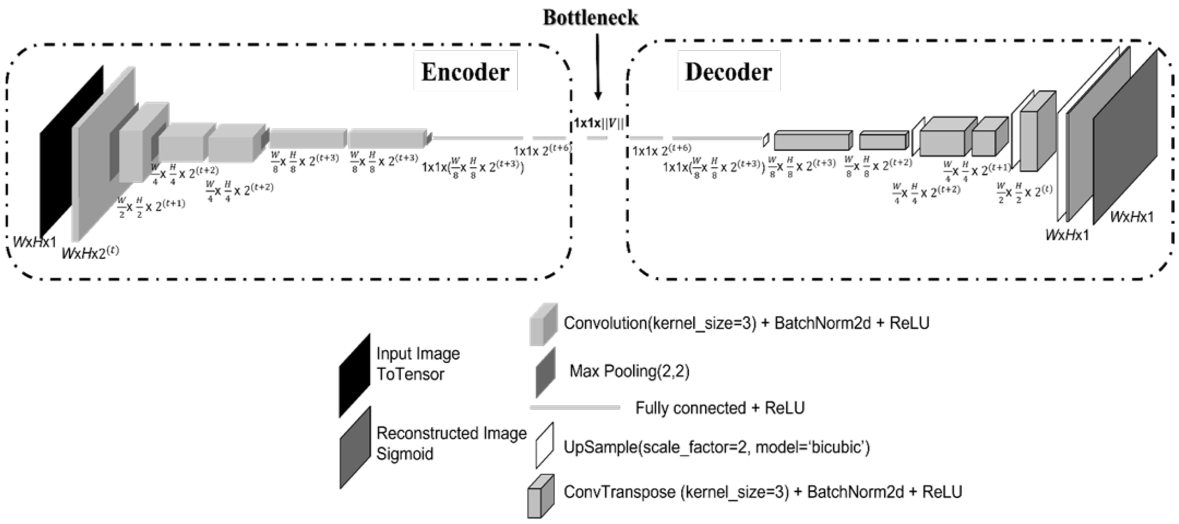

Bottleneck
Encoder
Decoder
1x1x||V||
WxHx1
WxHx2^(t)
Decoder
WxHx1
Input Image ToTensor
Reconstructed Image Sigmoid
Convolution(kernel_size=3) + BatchNorm2d + ReLU
Max Pooling(2,2)
Fully connected + ReLU
UpSample(scale_factor=2, model='bicubic')
ConvTranspose (kernel_size=3) + BatchNorm2d + ReLU

Figure 5. Simplified model architecture of the encoder and decoder, based on VGG11 structure.

The model takes a grayscale image of size W×H as input, which is first converted into a tensor for efficient GPU computation. The image then passes through multiple convolutional layers, each using a 3×3 kernel—consistent with the VGG11 architecture. The number of channels is controlled by a variable *t*, as detailed in Section 4. Each convolutional layer is followed by batch normalization and a Rectified Linear Unit (ReLU) activation to stabilize training and model non-linear relationships. Downsampling is performed using pooling layers after certain convolutions. The resulting feature maps are flattened into a 2048-dimensional vector and passed through a fully connected layer to generate a 64-dimensional bottleneck vector *V*. This vector is then quantized so that each element is represented using 8 bits.

3.1.2. Multiple Copies

In the watermark generation and embedding process, multiple copies of the Bottleneck are created to ensure image quality can still be restored even under heavy tampering. This redundancy not only improves recovery performance but also enables effective tampering localization.

Given an original image I_O of size $W \times H$, the Bottleneck vector V is duplicated N times, where $\mathrm{N} = (W \times H)/(m \times n)$. The vector is then reshaped into a 2D matrix, denoted as V_{2d}, to fill the entire image I_O, as illustrated in Figure 6.

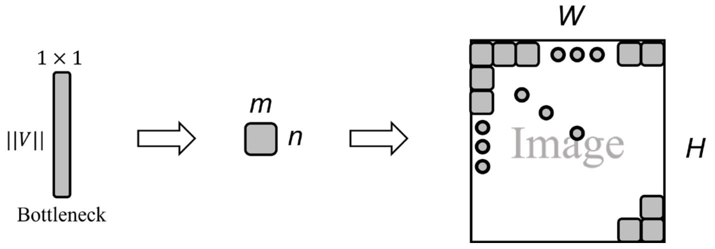


Figure 6. Illustration of generating multiple copies of the bottleneck vector for embedding.

3.1.3. Number Sequence and Scrambling

The generation of the pseudo-random sequence and scrambling process consists of two steps:

First, a random seed (Secret Key) is used to generate N pseudo-random numbers, where $N = (W \times H)/(m \times n)$. The numbers are produced using the Mersenne Twister algorithm, each a unique integer between 1 and 1000.

Next, the Bottleneck is scrambled using the Arnold Transform, commonly applied in image encryption by rearranging matrix positions to achieve encryption. The Arnold Transform matrix is shown in Equation 6, where (x, y) represents the original coordinates in the 2D Bottleneck matrix V_{2d}, (x', y') are the transformed coordinates after scrambling, and v is the matrix dimension. The modulo v ensures the transformed coordinates remain within matrix bounds.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^T \times \begin{bmatrix} x \\ y \end{bmatrix} \bmod v \tag{6}$$

The numbers in the Number Sequence generated by the Secret Key represent the number of times T the Arnold Transform is applied. This step ensures each embedded block is uniquely scrambled, preventing undetected collage attacks. The corresponding illustration is shown in Figure 7, where V_{2d} is a hypothetical value. We apply scrambling from 40 times at the top-left block up to 535 times at the bottom-right block.

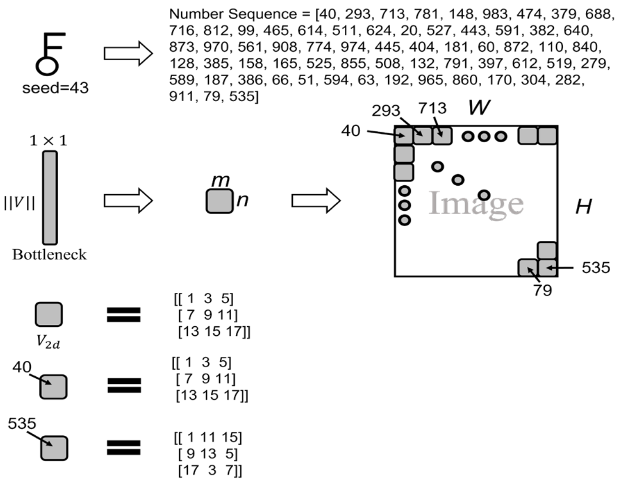

seed=43
Number Sequence = [40, 293, 713, 781, 148, 983, 474, 379, 688, 716, 812, 99, 465, 614, 511, 624, 20, 527, 443, 591, 382, 640, 873, 970, 561, 908, 774, 974, 445, 404, 181, 60, 872, 110, 840, 128, 385, 158, 165, 525, 855, 508, 132, 791, 397, 612, 519, 279, 589, 187, 386, 66, 51, 594, 63, 192, 965, 860, 170, 304, 282, 911, 79, 535]
1 × 1
||V||
Bottleneck
m
n
293
713
W
40
Image
H
535
79
V2d
[[1 3 5] [7 9 11] [13 15 17]]
40
[[1 3 5] [7 9 11] [13 15 17]]
535
[[1 11 15] [9 13 5] [17 3 7]]

Figure 7. Scrambling of Bottleneck Features using Number Sequence and Arnold Transform.

3.2. Tampering Localization and Self-Recovery

As illustrated in Figure 8, the tampering localization and self-recovery process begins by extracting the embedded scrambled bottleneck information from each block. Using the same random seed, a consistent number sequence is regenerated to descramble each block's bottleneck using the inverse Arnold matrix. The most frequently occurring bottleneck across all blocks is identified as the correct value; while differing values are marked as tampered blocks. Since the sequence is restored before statistical analysis, the tampering map corresponds to the original positions. Morphological closing is then applied to improve detection accuracy. Finally, the correct bottlenecks are decoded and merged with untampered blocks to produce the final recovered image.

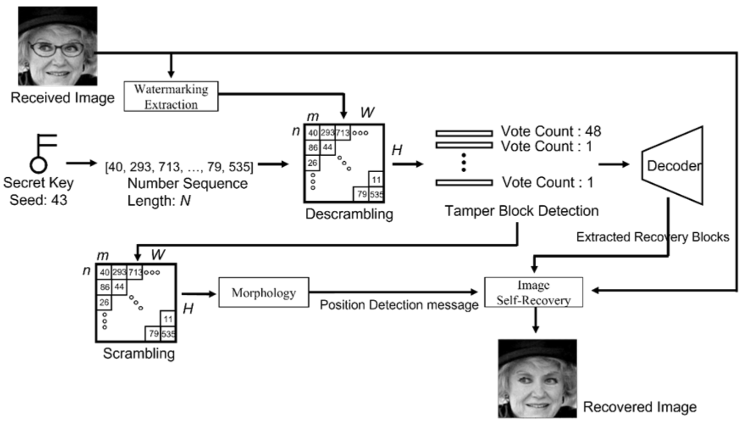


Figure 8. Flowchart of Tampering Localization and Self-Recovery.

3.2.1. Watermarking Extraction, Number Sequence and Descrambling

Watermark extraction involves retrieving information from each block using two LSBs, while generating a Number Sequence with the same key, as shown in Figure 9. The de-scrambling process restores the scrambled bottleneck values by applying the inverse Arnold Transform. The Number Sequence determines the number of inverse iterations, T. The inverse Arnold Transform is given in Equation (7), where (x', y') are the 2D coordinates of the extracted bottleneck block, and (x, y) are the original coordinates obtained by the inverse operation. Here, v represents the bottleneck's side length, and the modulo v ensures the coordinates remain within valid bounds.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}^T \times \begin{bmatrix} x' \\ y' \end{bmatrix} \, mod \, v \tag{7}$$

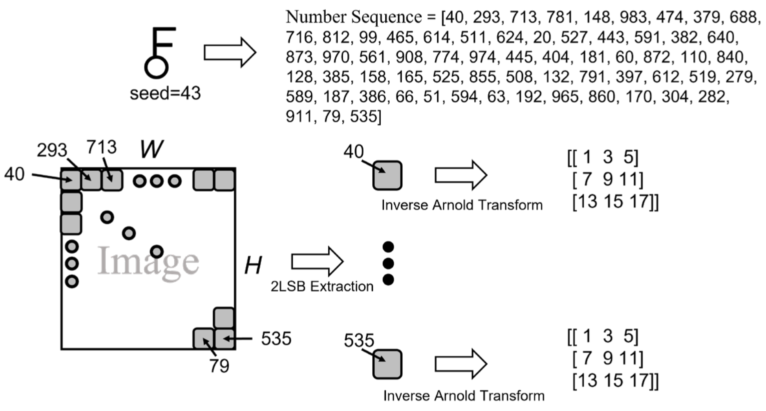

seed=43
Number Sequence = [40, 293, 713, 781, 148, 983, 474, 379, 688, 716, 812, 99, 465, 614, 511, 624, 20, 527, 443, 591, 382, 640, 873, 970, 561, 908, 774, 974, 445, 404, 181, 60, 872, 110, 840, 128, 385, 158, 165, 525, 855, 508, 132, 791, 397, 612, 519, 279, 589, 187, 386, 66, 51, 594, 63, 192, 965, 860, 170, 304, 282, 911, 79, 535]
40
293
713
W
Image
H
535
79
2LSB Extraction
40
Inverse Arnold Transform
[[1 3 5]
[7 9 11]
[13 15 17]]
535
Inverse Arnold Transform
[[1 3 5]
[7 9 11]
[13 15 17]]

Figure 9. Descrambling Bottleneck Blocks using Inverse Arnold Transform.

3.2.2. Tamper Block Detection (Vote), Scrambling and Morphology

After extracting and de-scrambling the bottleneck values of each block, we perform a statistical voting process to locate tampered regions, as illustrated in the tampering detection voting flow in Figure 10. However, to accurately pinpoint tampering locations, a second scrambling operation is necessary because the current voting is based on de-scrambled data.

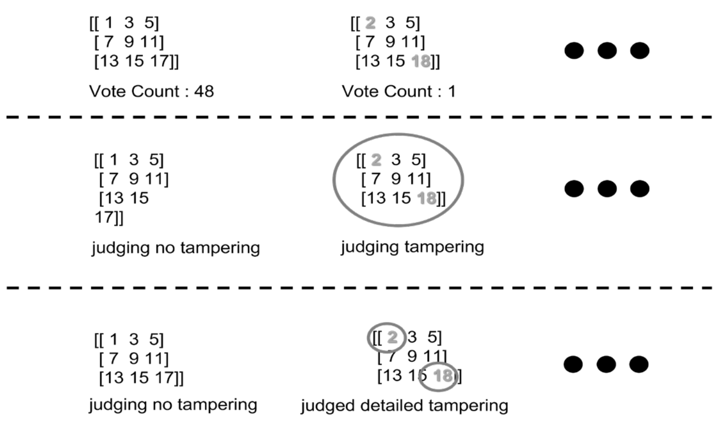


Figure 10. Tampering Detection Voting Process.

Next, morphological image processing is applied to address potential false negatives—cases where extracted LSBs are identical but the actual pixel values differ—allowing for correction of missed tampering detections.

3.2.3. Decoder and Image Self-Reocvery

After determining the most reliable bottleneck, it is fed into the decoder to reconstruct an image of the same size as the original. The decoder design Figure 5 mirrors the encoder structure: the bottleneck is first expanded to 2048 dimensions via a fully connected layer, reshaped into two dimensions, then processed through multiple upsampling and transposed convolution layers. Upsampling uses bilinear interpolation at twice the original size, while transposed convolutions incorporate batch normalization and ReLU activation to enhance feature learning and training stability. The final output layer employs a Sigmoid activation to ensure the output image values lie within [0,1], facilitating tensor operations and loss computation.

We designed a combined loss function integrating Structural Similarity Index (SSIM) and Root Mean Square Error (RMSE) to comprehensively evaluate the quality of images generated by the autoencoder. SSIM, a statistical metric assessing similarity in structure, luminance, and contrast [26], is formulated as in Equation (8). This loss balances structural features with pixel-level accuracy for improved reconstruction quality.

In this study, X and Y represent the input and target images, respectively. μ_X and μ_Y denote local means, ${\sigma_X}^2$ and ${\sigma_Y}^2$ are local variances, σ_{XY} is the covariance, and C_1 and C_2 are constants for numerical stability. We estimate these local statistics using a Gaussian blur with a 3×3 window. Since SSIM is a similarity metric ranging from [-1, 1] with an ideal value of 1, we convert it into a loss function by subtracting SSIM from 1, defining the SSIM loss as in Equation (9).

RMSE measures pixel-wise error between images, calculated as shown in Equation (10), where $W \times H$ is the number of pixels, and X_i, Y_i are the predicted and target pixel values. Lower RMSE indicates closer pixel-level similarity, making it suitable as a loss function defined in Equation (11).

To balance structural similarity and pixel-wise accuracy, we propose a combined SSIM-RMSE loss function weighted by α and β, as formulated in Equation (12).

$$SSIM(X,Y) = \frac{(2\mu_X\mu_Y + C_1)(2\sigma_{XY} + C_2)}{({\mu_X}^2 + {\mu_Y}^2 + C_1)({\sigma_X}^2 + {\sigma_Y}^2 + C_2)} \tag{8}$$

$$L_{SSIM} = 1 - SSIM(X,Y) \tag{9}$$

$$RMSE(X,Y) = \sqrt{\frac{1}{W \times H}\sum_{i=1}^{W \times H}(X_i - Y_i)^2} \tag{10}$$

$$L_{RMSE} = RMSE(X,Y) \tag{11}$$

$$L_{total} = \alpha L_{SSIM} + \beta L_{RMSE} \tag{12}$$

Finally, Image Self-Recovery performs the final processing by combining the tampering localization information, recovery data, and the received image. First, it identifies which blocks have been tampered with and which have not based on the localization results. For the tampered regions, the corresponding blocks from the recovery data are used for restoration, while untampered regions retain the blocks from the received image. This approach is adopted because, although untampered blocks contain watermarks, their image quality is generally better than that of the recovered blocks. By integrating these three sources of information, the method achieves the best possible image restoration.

4. Experimental Results

4.1. Experiment Environment and Dataset

The experiments were conducted using Python version 3.11, with the PyTorch 2.1.0 deep learning framework and GPU acceleration enabled via CUDA. The computing environment consisted of a 12th-generation Intel® Core™ i7-12700F CPU, an NVIDIA GeForce GTX 1650 GPU, 48 GB of RAM, and a 64-bit Windows 10 operating system.

This study utilized the CelebFaces Attributes Dataset (CelebA) [27], containing 202,599 celebrity face images across 10,177 identities with 40 facial attributes (e.g., gender, age, expression). Following CelebA's original split, images are divided into training (1–162,770), validation (162,771–182,637), and testing (182,638–202,599) sets to avoid identity overlap. We randomly selected 40,000 training images to train the convolutional autoencoder and 10,000 testing images for model evaluation and fragile watermark experiments.

During preprocessing, the original JPG images were first converted to PNG format to enable watermark embedding. Images were then converted to grayscale, and center-cropped to a fixed resolution of 128×128 pixels. If the original image dimensions were insufficient, black padding was applied to maintain uniformity. The grayscale conversion emphasized the green channel's luminance aligned with human visual perception—rather than computing a simple RGB average, as defined in Equation (13) [28].

$$Gray = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{13}$$

Figure 11 shows examples of our processed images, each sized 128×128 pixels, with their names displayed below. In the experimental design, these 128×128 images are divided into non-overlapping blocks of 16×16 pixels.

Figure 11. Sample Experimental Images.

4.2. Evaluation Metrics

To evaluate the quality of image processing, such as watermark embedding or tampering recovery, we use two common metrics: Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity

Index (SSIM). PSNR measures signal fidelity, while SSIM focuses on structural and perceptual quality [27].

For tampering recovery assessment, we apply two approaches: a full-image evaluation to quantify overall quality changes, and a focused analysis on tampered regions to detail recovery performance, with comprehensive results recorded.

PSNR is calculated as in Equation (14), where higher values indicate greater similarity. L represents the maximum pixel value (255 for 8-bit images). Mean Squared Error (MSE), defined in Equation (15), measures the average squared difference between pixels X_i and Y_i of the original and processed images, with image dimensions W×H. Lower MSE leads to higher PSNR, reflecting smaller differences between images.

$$PSNR = 10 \times log_{10}^{(\frac{L^2}{MSE})} \tag{14}$$

$$MSE = \frac{1}{\mathrm{W}\times H}\sum_{i=1}^{W\times H}(X_i - Y_i)^2 \tag{15}$$

The SSIM calculation is defined in Equation (16), where μ_X and μ_Y denote the mean luminance of the original image X and processed image Y, respectively. ${\sigma_X}^2$ and ${\sigma_Y}^2$ represent the variances of the images, measuring contrast. σ_{XY} is the covariance between the two images, indicating their similarity. C_1 and C_2 are stability constants.

$$SSIM(X,Y) = \frac{(2\mu_X\mu_Y + C_1)(2\sigma_{XY} + C_2)}{({\mu_X}^2 + {\mu_Y}^2 + C_1)({\sigma_X}^2 + {\sigma_Y}^2 + C_2)} \tag{16}$$

Following the image quality evaluation, we introduce recall, precision, and F1-score as metrics for tampering localization. These are calculated using True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN), with formulas given in Equations (17) and (18).

$$Recall = \frac{TP}{TP + FN} \tag{17}$$

$$Precision = \frac{TP}{TP + FP} \tag{18}$$

Recall and precision range from 0 to 1. Recall measures the proportion of actual tampered samples correctly identified, reflecting the model's detection capability. Precision indicates the proportion of predicted tampered samples that are truly tampered, reflecting prediction accuracy. To balance both, the F1-score is used as a comprehensive metric—defined as the harmonic mean of precision and recall Equation (19)—suitable for scenarios requiring both accuracy and completeness.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{19}$$

4.3. Comparison of Convolutional Autoencoders with Different Parameters

The convolutional autoencoder in this study is inspired by the VGG11 architecture. Since VGG11 was originally designed for ImageNet classification, its structure is relatively complex. However, this research focuses on image compression and reconstruction, which does not require such a large model. Therefore, VGG11 was simplified and adapted to better suit the needs of an autoencoder.

4.3.1. Need for Fully Connected Layer Design

This consideration stems from the question of whether flattening is necessary in convolutional networks. The network design comparison is illustrated in Figure 12.

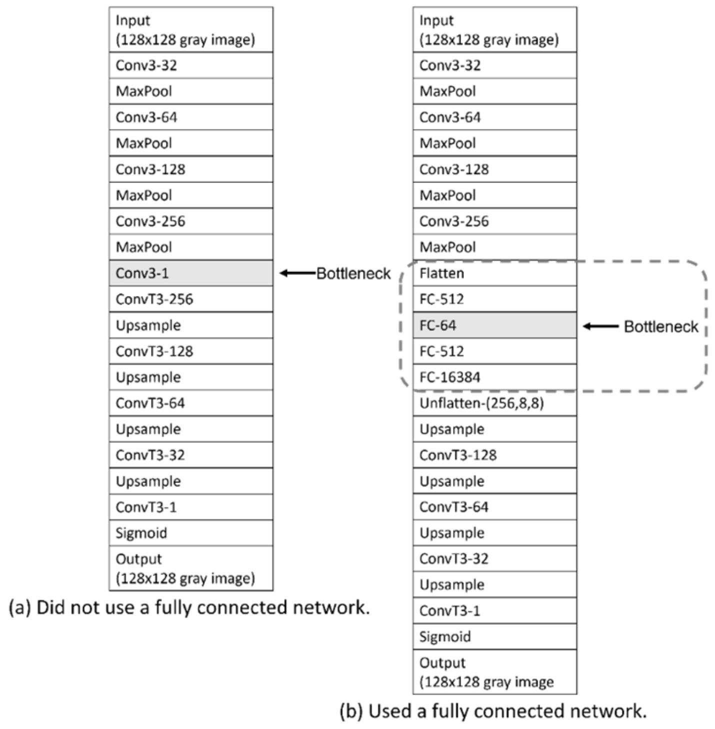


Figure 12. Whether to Use Fully Connected Network Architecture.

Figure 12(a) shows the architecture without fully connected layers, where the bottleneck retains a 2D shape of (8, 8). In contrast, Figure 12(b) introduces fully connected layers by flattening the feature map and reducing its dimensions sequentially to 512 and then 64 to form the bottleneck. Both architectures apply ReLU after convolutional and fully connected layers, and the bottleneck is quantized to 8-bit values within the range of 0–255.

This comparison aims to evaluate whether preserving a 2D structure benefits reconstruction. However, implementation results demonstrate that using fully connected layers significantly improves reconstruction quality. As shown in Figure 13, both models were trained on 40,000 images and tested on 10,000. Figure 13(a), without fully connected layers, shows inferior results, while Figure 13(b) illustrates noticeable improvement with them.

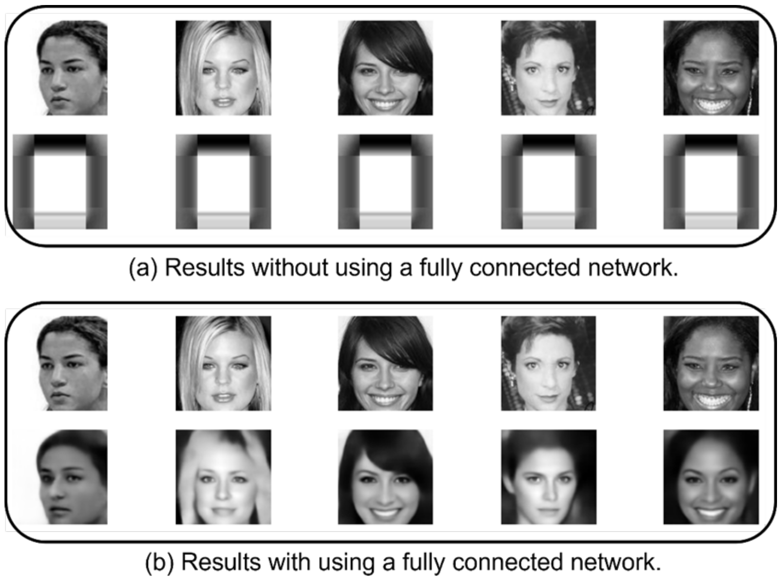

(a) Results without using a fully connected network.
(b) Results with using a fully connected network.

Figure 13. Results of Using Fully Connected Layers.

4.3.2. Adjustment of Network Scale

We evaluated the impact of varying network depths on image compression and reconstruction quality to determine the optimal architecture. The tested models followed a fully connected encoder–decoder design, as described in Section 4.3.1, using ReLU activation and 3×3 kernels for all convolutional and deconvolutional layers. Down sampling and up sampling were handled using MaxPooling and Up sampling layers, respectively. The results, summarized in Table 2, show how deeper networks improve performance up to a certain point before diminishing returns appear.

Table 2. Performance Comparison Across Different Depths (Average on 10,000 Test Images).

Model Architecture	PSNR	SSIM
(a)	28.417	0.803
(b)	28.581	0.849
(c)	**28.583**	**0.850**
(d)	28.328	0.796

The best performance was achieved using the configuration of model architecture (c), trained on 40,000 images and evaluated on a test set of 10,000 images. As shown in Table 2, this setup yielded the highest average PSNR of 28.583 dB and an SSIM of 0.850.

4.3.3. Whether to Use Batch Normalization

This subsection explores the impact of adding batch normalization to the model originally described in Section 4.3.1. In the modified architecture, batch normalization layers were introduced before each ReLU activation following the convolutional layers. This adjustment aimed to improve convergence and stability during training. The performance comparison between the original and modified models is summarized in the results, demonstrating improved reconstruction quality with the inclusion of batch normalization.

Table 3. Comparison of Models with and Without Batch Normalization (Averaged on 10,000 Test Images).

Model Architecture	PSNR	SSIM
(a)	28.583	0.850
(b)	**29.160**	**0.906**

As shown in the implementation results in Table 3, applying batch normalization yields better performance compared to models without it.

4.3.4. Effect of Dropout

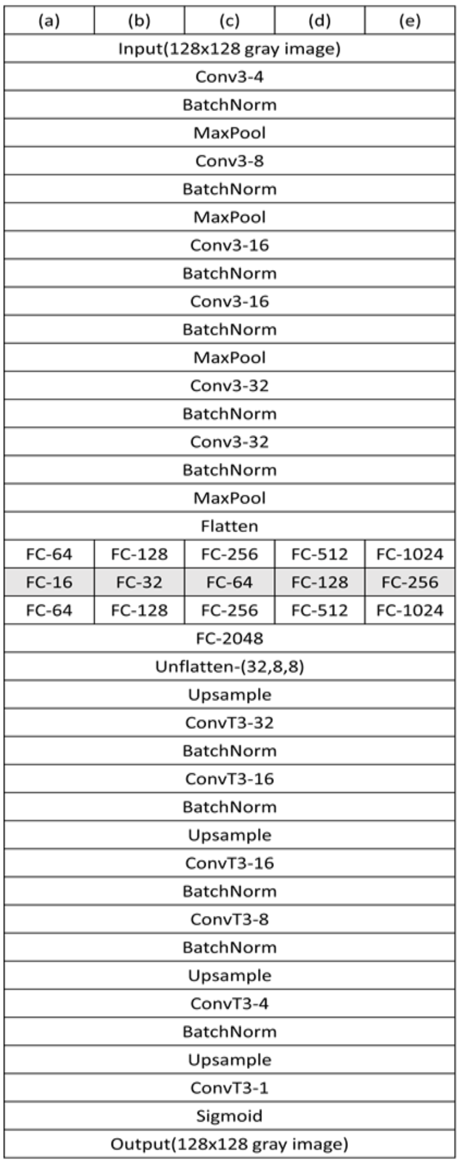
To investigate whether dropout could improve reconstruction performance and reduce overfitting, we experimented with dropout rates of 0%, 5%, 10%, and 20% in the fully connected layers. However, the results showed minimal or even slightly negative impact on image quality. Therefore, dropout was excluded from the final model configuration.

4.3.5. Loss Function Weight

We evaluated the effect of different weight combinations for the SSIM and RMSE components in our loss function. Among the tested settings, assigning a higher weight to SSIM (e.g., 0.8 SSIM, 0.2 RMSE) yielded the best balance between structural fidelity and pixel-wise accuracy. As a result, this weighting scheme was adopted for the final model.

4.3.6. Variation in the Number of Bottlenecks

This subsection investigates the impact of bottleneck vector size on image reconstruction quality. Experiments use the previously described architecture: four layers with fully connected layers, no dropout, batch normalization enabled, and a loss function weighted 0.8 for SSIM and 0.2 for RMSE. As shown in Figure 14, the number of bottlenecks increases from 16 to 256 (Figures 14(a)–(e)). Results summarized in Table 4, based on 40,000 training and 10,000 testing images, indicate that larger bottlenecks improve reconstruction quality but reduce tampering localization accuracy due to fewer voting blocks [13][19][20]. Therefore, a compromise of 64 bottlenecks is adopted.



(a)	(b)	(c)	(d)	(e)
Input(128x128 gray image)				
Conv3-4				
BatchNorm				
MaxPool				
Conv3-8				
BatchNorm				
MaxPool				
Conv3-16				
BatchNorm				
Conv3-16				
BatchNorm				
MaxPool				
Conv3-32				
BatchNorm				
Conv3-32				
BatchNorm				
MaxPool				
Flatten				
FC-64	FC-128	FC-256	FC-512	FC-1024
FC-16	FC-32	FC-64	FC-128	FC-256
FC-64	FC-128	FC-256	FC-512	FC-1024
FC-2048				
Unflatten-(32,8,8)				
Upsample				
ConvT3-32				
BatchNorm				
ConvT3-16				
BatchNorm				
Upsample				
ConvT3-16				
BatchNorm				
ConvT3-8				
BatchNorm				
Upsample				
ConvT3-4				
BatchNorm				
Upsample				
ConvT3-1				
Sigmoid				
Output(128x128 gray image)				

Figure 14. Comparison of Different Numbers of Bottlenecks.

Table 4. Results for Different Numbers of Bottlenecks (Average over 10,000 Test Images).

Model Architecture	PSNR	SSIM
(a) FC-16	28.664	0.857
(b) FC-32	28.846	0.879
(c) FC-64	**29.297**	**0.921**
(d) FC-128	29.299	0.927
(e) FC-256	29.351	0.939

Based on extensive experiments comparing factors such as the use of fully connected layers, network size, and batch normalization, we finalized the number of encoder bottlenecks at 64. When correctly extracting these bottlenecks for reconstruction, the restored images achieve an average minimum quality of PSNR 29.297 dB and SSIM 0.921.

4.4. Tampering Recovery Results under Different Scenarios

This section analyzes experimental results under watermark embedding and various tampering scenarios. Section 4.4.1 evaluates the impact of watermark embedding on image quality using PSNR and SSIM compared to the original images. Section 4.4.2 assesses tampering localization performance through recall, precision, and F1-score, alongside PSNR and SSIM metrics for both the overall and tampered regions in restored images. Section 4.4.3 further examines statistical results across different tampering ratios.

4.4.1. Watermarked Image Quality

Table 6 presents the results of embedding 64 compressed 8-bit messages—obtained via the encoder—into 128×128 images by replicating them across 16×16 blocks. The data represent averages over 10,000 test images and are comparable to prior studies employing dual LSB embedding methods [13][19][20]. Across the 10,000 test images, the average PSNR and SSIM after watermark embedding were 43.654 dB and 0.999, respectively, indicating negligible quality degradation.

Table 6. Image Quality After Watermark Embedding (Average over 10,000 Test Images).

	PSNR	**SSIM**
(a)	43.654	0.999

4.4.2. Tampering Methods in Different Scenarios

Figure 15 illustrates the complete processing pipeline of the proposed model in addressing eyeglass tampering attacks. Figure 15(a) presents the original input image, while 15(b) shows the watermarked version, which achieves a PSNR of 43.85 dB and an SSIM of 1.0. In (c), tampering is introduced using Photoshop by adding eyeglasses. Comparing (c) with (b) yields the tampering map in (d), revealing a tampered region covering 6.86% of the image. In the first stage of tamper detection, illustrated in 15(e), all bottleneck features are extracted and descrambled. A voting mechanism is then applied to detect suspicious 16×16 blocks with low vote frequency. Figure 15(f) further refines these regions by pinpointing tampered 2×2 sub-blocks. These refined tampering details are then scrambled back to their original positions in (g), followed by morphological closing in (h) to enhance localization. The final detection achieves a recall of 0.994, precision of 0.829, and F1-score of 0. 904. Figure 15(i) displays the reconstructed image generated from the decoded bottleneck features. In 15(j), the final restored image is created by combining information from 15(c), 15(h), and 15(i): untouched regions retain the original pixels from 15(c), while tampered regions are replaced by reconstructed content from 15(i). The overall restored image reaches a PSNR of 38.745 dB and SSIM of 0.994. Specifically, the restored tampered region achieves a PSNR of 29.281 dB and an SSIM of 0.743.

Figure 16 illustrates the processing of graffiti-like text tampering. (a) is the original image, and (b) is the watermarked version with PSNR 44.096 dB and SSIM 1. In (c), “NCHU” text is added on the face using MS Paint. (d) shows the ground truth tampering mask (3.229% tampered). (e) extracts bottleneck features, performs descrambling, and uses voting to detect anomalous blocks. (f) refines pixel-level anomalies, followed by scrambling to restore positions, shown in (g). (h) applies morphological closing for enhanced detection: Recall 1.0, Precision 0.722, F1-score 0.839. (i) is the decoder output using the most-voted bottleneck, and (j) is the final recovery—keeping (c)’s pixels for untampered regions and using (i) for tampered areas. Final image recovery: PSNR 40.748 dB, SSIM 0.999; tampered region: PSNR 29.707 dB, SSIM 0.921.

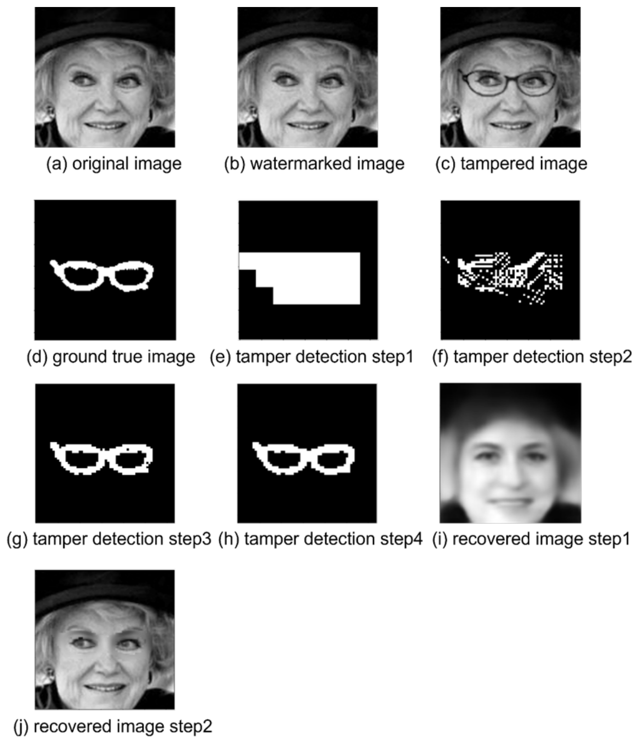

(a) original image
(b) watermarked image
(c) tampered image
(d) ground true image
(e) tamper detection step1
(f) tamper detection step2
(g) tamper detection step3
(h) tamper detection step4
(i) recovered image step1
(j) recovered image step2

Figure 15. Tampering Attack Simulation (Adding Glasses).

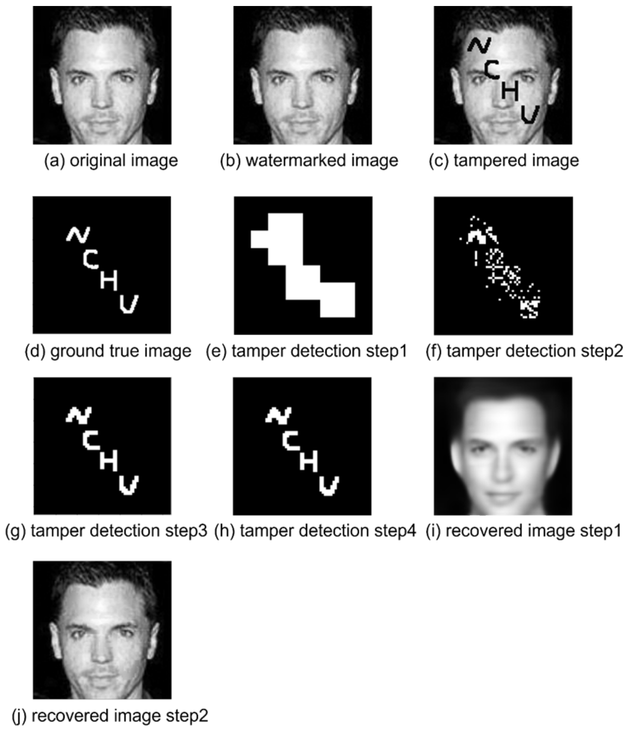

(a) original image
(b) watermarked image
(c) tampered image
(d) ground true image
(e) tamper detection step1
(f) tamper detection step2
(g) tamper detection step3
(h) tamper detection step4
(i) recovered image step1
(j) recovered image step2

Figure 16. Tampering Attack Simulation (Graffiti Text).

A more specific form of tampering is the collage attack, where one watermarked image is used to tamper with another. As shown in Figure 17, (a) and (b) are watermarked images with keys 42 and 43, respectively. In (c), the face from (b) is pasted onto (a) using Photoshop, creating a tampered image. (d) shows the ground truth tampering mask, with a tampering ratio of 31.104%.

The detection and recovery process (Figures 17(e)–(j)) follows the same steps as in standard tampering. The localization result (h) achieves Recall 0.995, Precision 0.929, and F1-score 0.961. The final recovered image (j) has a PSNR of 33.967 dB and SSIM of 0.991 overall; within the tampered region, PSNR is 29.52 dB and SSIM is 0.853.

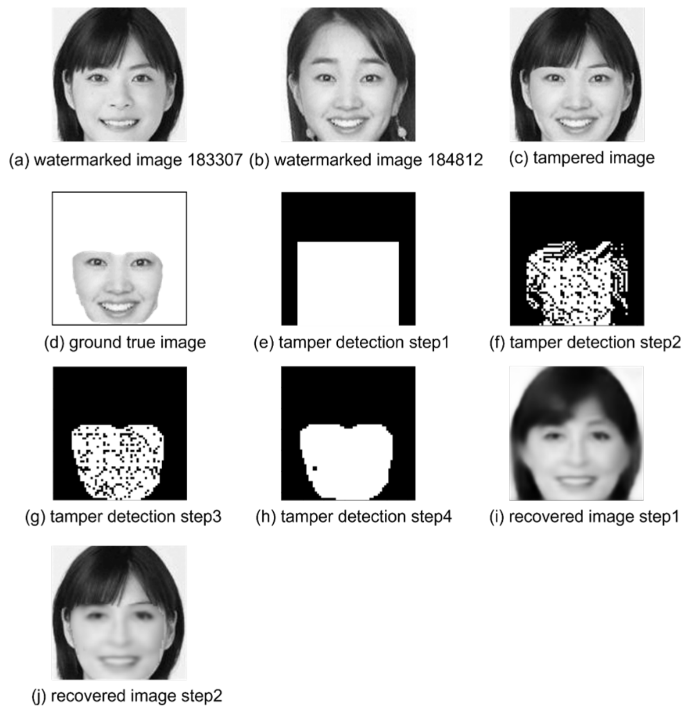

(a) watermarked image 183307
(b) watermarked image 184812
(c) tampered image
(d) ground true image
(e) tamper detection step1
(f) tamper detection step2
(g) tamper detection step3
(h) tamper detection step4
(i) recovered image step1
(j) recovered image step2

Figure 17. Collage Attack Simulation.

4.4.3. Analysis of Different Tampering Levels

To visually demonstrate the performance of the proposed watermarking method, we selected three representative tampering levels: 10%, 50%, and 75%.
As illustrated in Figure 18, the top row displays the tampered images, the middle row shows the tampering localization (binary masks), and the bottom row shows the corresponding recovered outputs.

At low tampering levels (10%), the model produces high-fidelity recovery with minimal visual artifacts. At moderate (50%) and severe (75%) tampering, the detection remains accurate, and the self-recovery performance maintains structural consistency. Despite some blurring at higher tampering rates, facial features and contextual information are largely preserved, proving the robustness of the proposed method.

Table 7 summarizes recovery statistics for entire images, averaged over 10,000 test images. Table 8 presents recovery statistics for tampered regions, also averaged over 10,000 images. Table 9 reports the average Recall, Precision, and F1-Score for different tampering levels on the test set.

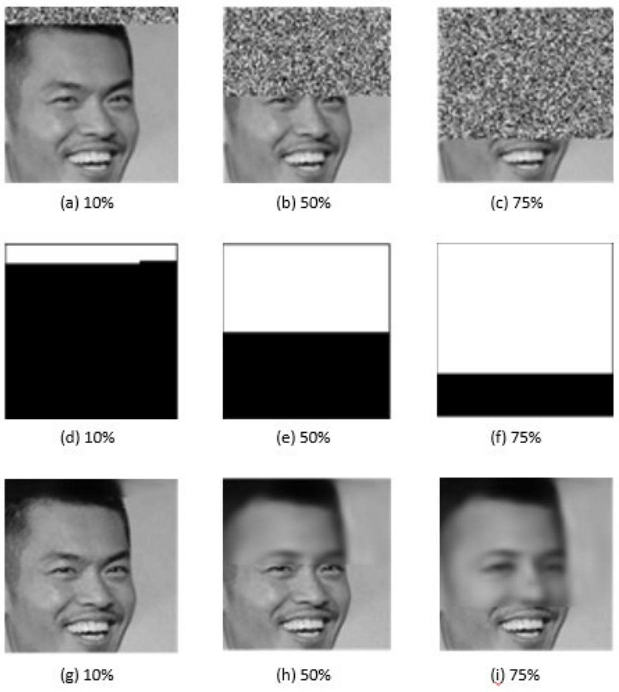


Figure 18. Tampering detection and recovery results under basic tampering at 10%, 50%, and 75% levels. (a-c) tampered images; (d-f) detected tampered regions; (g-i) recovered images.

Table 7. Recovered Image Quality for Various Tampering Levels.

Tampering Level	**0%**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**75%**
PSNR (dB)	43.654	37.964	35.742	34.282	33.162	32.238	31.383	30.787	30.518
SSIM	0.999	0.991	0.984	0.977	0.971	0.963	0.953	0.946	0.943

Table 8. Recovery Quality of Tampered Regions at Different Tampering Levels.

Tampering Level	**0%**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**75%**
PSNR (dB)	43.654	29.462	29.432	29.468	29.492	29.390	29.311	29.324	29.322
SSIM	0.999	0.812	0.872	0.908	0.922	0.921	0.920	0.924	0.924

Table 9. Effect of Different Tampering Levels on Tampering Detection and Localization (Average over 10,000 Test Images).

Tampering Level	**0%**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**75%**
Recall	-	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999
Precision	-	0.941	0.984	0.989	0.984	1	0.989	0.995	1
F1-Score	-	0.969	0.992	0.994	0.992	0.999	0.994	0.997	0.999

In addition to basic tampering, we evaluated the model's robustness under collage attacks, which simulate highly inconsistent and non-uniform tampering scenarios.

Figure 19 presents qualitative results at three tampering levels: 10%, 40%, and 75%.

The first row displays the tampered images with patches from foreign sources. The second row shows the tampering detection results, where the model successfully localizes irregular regions. The third row presents the recovered images, where even under severe 75% tampering, the method manages to reconstruct face structure with acceptable visual quality.

These results confirm the method's strong tampering localization and recovery capabilities in the presence of complex attacks.

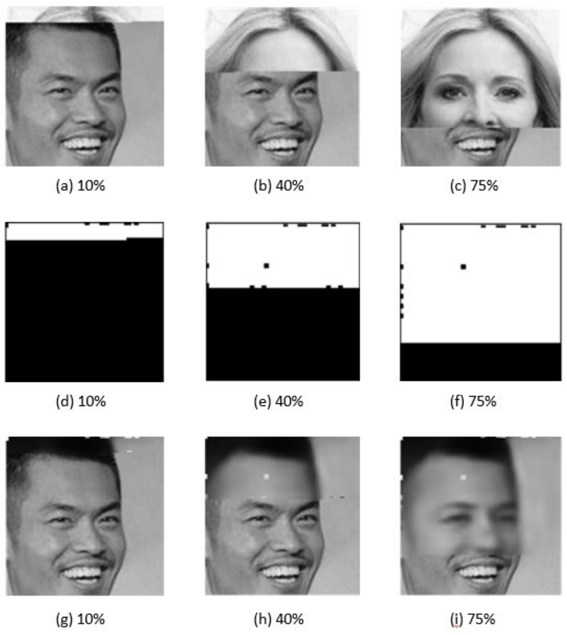


Figure 19. Tampering detection and recovery results under collage attacks at 10%, 40%, and 75% tampering levels. (a–c) Tampered collage images, (d–f) Detected tampered regions (binary masks), (g–i) Recovered outputs.

Table 10 presents the average overall recovery quality of 10,000 images after collage attacks at different tampering levels. Table 11 shows the average recovery quality within tampered regions. Table 12 summarizes the average detection metrics (Recall, Precision, F1-Score) across tampering levels.

Table 10. Recovered Image Quality under Different Collage Attack Levels.

竄改程度	0%	10%	20%	30%	40%	50%	60%	70%	75%
PSNR (dB)	43.654	37.964	35.742	34.282	33.140	32.218	31.364	30.771	30.502
SSIM	0.999	0.991	0.984	0.977	0.958	0.950	0.938	0.932	0.929

Table 11. Recovery Quality of Tampered Regions under Various Collage Attack Levels.

竄改程度	0%	10%	20%	30%	40%	50%	60%	70%	75%
PSNR (dB)	43.654	29.397	29.402	29.425	29.466	29.368	29.29	29.307	29.306
SSIM	0.999	0.664	0.79	0.844	0.887	0.894	0.896	0.904	0.906

Table 12. Tampering Localization Performance at Different Collage Attack Levels.

竄改程度	0%	10%	20%	30%	40%	50%	60%	70%	75%
Recall	-	0.965	0.982	0.978	0.985	0.988	0.987	0.989	0.99
Precision	-	0.965	0.984	0.99	0.984	1	0.99	0.995	1
F1-Score	-	0.965	0.983	0.984	0.985	0.994	0.989	0.992	0.995

4.5. Comparison of Our Method with Other Researchers' Methods

This section compares our method with BP-wised [13], SVD-based [19], and Rezaei et al. [20]. Table 13 shows watermark image quality results. Our method, BP-wised, and SVD-based averaged over 10,000 CelebA test images, while Rezaei et al. used the BOWS2 dataset, with data cited from [20].

Table 13. Image Quality After Watermark Embedding.

Method	PSNR	SSIM
BP-wised (2019)	44.163	0.999
SVD-based (2020)	44.013	0.999
Rezaei et al. (2022)	44.2	-
Proposed Method	43.654	0.999

Experiments show minimal difference between our method and others, as all embed watermarks in 2 LSBs. For PSNR evaluation of tampered region recovery under varying tampering levels, we, BP-wised, and SVD-based averaged results over 10,000 CelebA test images, while Rezaei et al. tested on the grayscale Cameraman image. Results are in Figure 20, with data from [20].

Our method excels at high tampering rates by relying on accurate bottleneck information for image restoration. BP-wised and SVD-based maintain decent PSNR under heavy tampering due to intact verification data and bicubic interpolation, but show relatively lower SSIM.

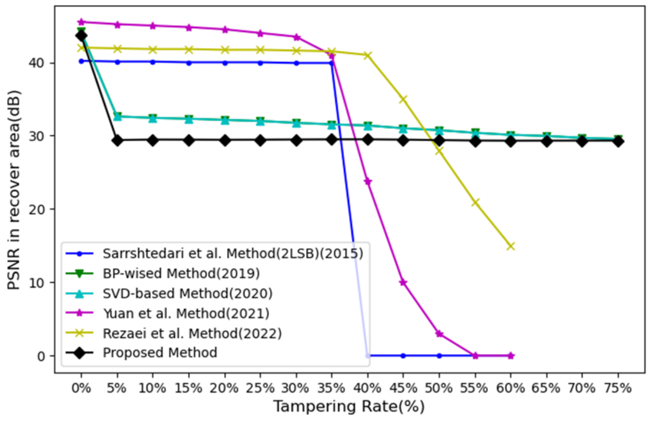


Figure 20. Comparison of Tampered Block Recovery Quality at Different Tampering Rates (PSNR (dB)).

SSIM was used to evaluate recovery in tampered regions at different tampering levels. Our method, BP-wised, and SVD-based used the CelebA test set, while Rezaei et al. used the BOWS2 dataset; results are shown in Figure 21 and referenced from [20].

Experiments show that above 60% tampering, our method outperforms others due to deep learning's ability to preserve image structure and improve SSIM. At low tampering levels, concentrated edge tampering and limited morphological effects increase noise sensitivity, resulting in lower SSIM.

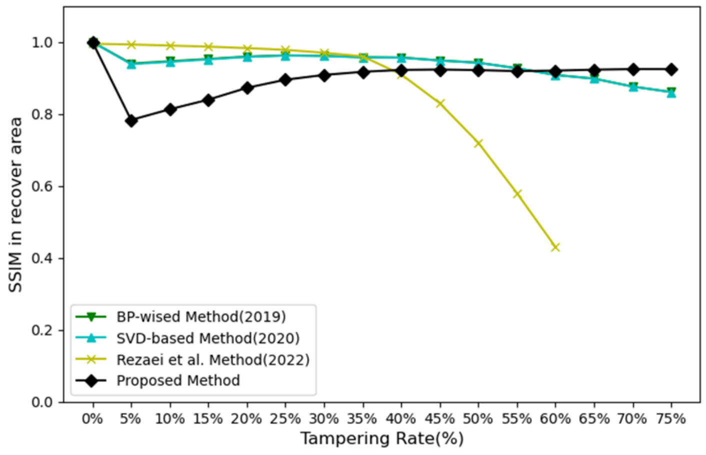


Figure 21. Comparison of Tampered Block Recovery Quality at Different Tampering Rates (SSIM).

Figure 22 compares image recovery under 75% basic tampering using our method, BP-wised, and SVD-based. It clearly shows our method's superior structural recovery under heavy tampering.

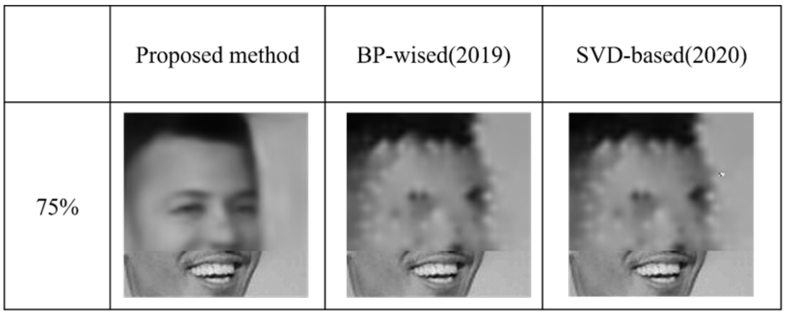


Figure 22. Comparison of 75% Basic Tampering Attack with BP-Wised and SVD-Based Methods.

Table 14 compares localization performance under basic tampering. Data for other methods is referenced from Rezaei et al. [20]. Our method achieves high Recall and F1-score, approaching 1, mainly due to fine-grained 2×2 block detection combined with morphological closing, which enhances localization accuracy.

Table 14. Detection and Localization Quality of Basic Tampering Attacks.

Schemes	**Tamper Rate**	**10%**	**20%**	**40%**
Sarreshtedari et al. [29] (2015)	Recall	1	1	1
	Precision	1	1	1
	F1-Score	1	1	1
BP-wised (2019)	Recall	0.999	0.999	1
	Precision	0.839	0.914	0.984
	F1-Score	0.912	0.955	0.992
SVD-based (2020)	Recall	0.999	0.999	0.999
	Precision	0.939	0.943	0.984
	F1-Score	0.968	0.971	0.992
Yuan et al. [30] (2021)	Recall	0.988	0.964	0.899
	Precision	0.956	0.935	0.817
	F1-Score	0.971	0.949	0.856
Rezaei et al. [20] (2022)	Recall	0.995	0.991	0.978
	Precision	1	1	1
	F1-Score	0.997	0.995	0.988
Proposed	Recall	0.999	0.999	0.999
	Precision	0.941	0.984	0.984
	F1-Score	0.969	0.992	0.992

The numerical performance metrics under collage attack are summarized in Table 10 (PSNR) and Table 11 (SSIM). The results confirm that the proposed method maintains high-quality reconstruction even under 75% tampering, with PSNR values above 32 dB and SSIM above 0.91. Experiments show that under heavy tampering, our method outperforms others. BP-wised and SVD-based methods, unable to effectively resist collage attacks, yield significantly lower performance, similar to results under basic tampering.

The proposed method outperforms previous approaches under high tampering rates. However, at lower tampering levels, detection accuracy is affected due to the limited effectiveness of morphological closing. As shown in Figure 23, under 75% collage attacks, our method demonstrates stronger resistance compared to BP-wised and SVD-based methods.

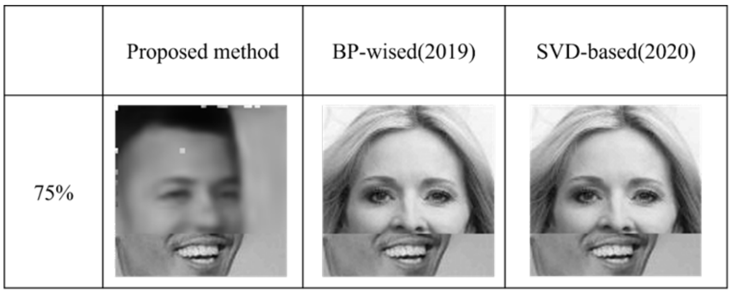


Figure 23. Recovery results under 75% collage attacks.

Table 15 shows the evaluation results for collage attack localization, with external data referenced from Rezaei et al. [20]. Overall, our method outperforms most existing approaches in F1-score, second only to Rezaei et al. [20].

Table 15. Collage Attack Tampering Detection and Localization Results.

Schemes	Tamper Rate	10%	20%	40%
Sarreshtedari et al. [29] (2015)	Recall	0.403	0.237	0.112
	Precision	**1**	**1**	**1**
	F1-Score	0.574	0.383	0.201
BP-wised (2019)	Recall	0.062	0.062	0.047
	Precision	0.245	0.398	0.752
	F1-Score	0.099	0.107	0.089
SVD-based (2020)	Recall	0.062	0.015	0.015
	Precision	0.490	0.231	0.504
	F1-Score	0.110	0.029	0.030
Yuan et al. [30] (2021)	Recall	0.982	0.969	0.897
	Precision	0.956	0.931	0.812
	F1-Score	0.968	0.949	0.852
Rezaei et al. [20] (2022)	Recall	**0.996**	**0.991**	0.977
	Precision	**1**	**1**	**1**
	F1-Score	**0.997**	**0.995**	**0.988**
Proposed	Recall	0.965	0.982	**0.985**
	Precision	0.965	0.984	0.984
	F1-Score	0.965	0.983	0.985

Overall, the proposed method shows clear advantages under high tampering rates, outperforming other approaches in SSIM for both general and collage attacks. Even at 75% tampering, it achieves an SSIM of 0.9, demonstrating strong structural recovery capability.

5. Conclusions and Future Work

5.1. Conclusions

This study proposes a novel fragile watermarking method based on a convolutional autoencoder, which differs from traditional approaches that embed authentication and recovery data separately. By leveraging compact bottleneck features, encoded compression, and a block-wise voting mechanism, the proposed method maintains high structural similarity (SSIM) even under extensive tampering scenarios.

To enhance robustness against collage attacks, we integrated secret key-based pseudo-random scrambling with the Arnold Transform, effectively resisting over 50% patch-level replacements.

Tamper localization is refined from 16×16 to 2×2 sub-blocks and further improved using morphological closing, thereby increasing localization granularity and pixel-level accuracy.

Overall, the proposed approach demonstrates robustness under high tampering ratios, effective resistance to collage attacks, and stable performance in both tamper localization and image self-recovery, making it a promising solution for image authentication tasks.

5.2. Future Work

Due to limited computational resources and dataset availability, this study currently focuses on facial image datasets and does not generalize to arbitrary images. In the future, this limitation could be addressed with increased computational power and access to diverse open-source image datasets, enabling the training of a more generalized restoration model.

In terms of security, while using a unique secret key per image enhances protection, it also introduces usability challenges. Future work will explore strategies to balance security with key management efficiency.

Additionally, adversarial attacks such as FGSM [31] have shown that small perturbations can mislead deep learning models. Since watermark embedding and image recovery also introduce perturbations, these may impact AI-based classifiers. As a countermeasure, we propose to utilize Principal Component Analysis (PCA) on bottleneck features extracted by the convolutional autoencoder. This allows classification tasks to rely directly on compact and meaningful embedded features, potentially improving model robustness even under tampering or restoration.

Acknowledgments: This work is supported by the National Science and Technology Council of Taiwan (NSTC 113-2221-E-324 -017-MY2).

References

1. Capasso, P.; Cattaneo, G.; De Marsico, M. A Comprehensive Survey on Methods for Image Integrity. *ACM Trans. Multimedia Comput. Commun. Appl.* **2024**, *20*, 1–34, https://doi.org/10.1145/3633203.
2. Lee, W.-B.; Chen, T.-H. A public verifiable copy protection technique for still images. *J. Syst. Softw.* **2002**, *62*, 195–204, https://doi.org/10.1016/s0164-1212(01)00142-x.
3. Li, X.; Guo, M.; Wang, Z.; Li, J.; Qin, C. Robust Image Hashing in Encrypted Domain. *IEEE Trans. Emerg. Top. Comput. Intell.* **2023**, *8*, 670–683, https://doi.org/10.1109/tetci.2023.3281843.
4. Cox, I.; Honsinger, C.; Miller, M.; Bloom, J. Digital Watermarking. *J. Electron. Imaging* **2002**, *11*, 414–414, https://doi.org/10.1117/1.1494075.
5. Ferrara, P.; Bianchi, T.; De Rosa, A.; Piva, A. Image Forgery Localization via Fine-Grained Analysis of CFA Artifacts. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 1566–1577, https://doi.org/10.1109/tifs.2012.2202227.
6. Jegou, H.; Douze, M.; Schmid, C. Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In Computer Vision – ECCV 2008; Forsyth, D., Torr, P., Zisserman, A., Eds.; Proceedings of the 10th European Conference on Computer Vision, Marseille, France, 12–18 October 2008; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5302, pp. 304–317.
7. Li, L.; Li, S.; Zhu, H.; Chu, S.-C.; Roddick, J.F.; Pan, J.-S. An Efficient Scheme for Detecting Copy-move Forged Images by Local Binary Patterns. *J. Inf. Hiding Multim. Signal Process.* **2013**, *4*, 46-56.
8. Mushtaq, S.; Mir, A.H. Digital Image Forgeries and Passive Image Authentication Techniques: A Survey. *Int. J. Adv. Sci. Technol.* **2014**, *73*, 15–32, https://doi.org/10.14257/ijast.2014.73.02.
9. Yang, C.-W.; Shen, J.-J. Recover the tampered image based on VQ indexing. *Signal Process.* **2010**, *90*, 331–343, https://doi.org/10.1016/j.sigpro.2009.07.007.
10. Di, Y.; Lee, C.; Wang, Z.; Chang, C.; Li, J. A Robust and Removable Watermarking Scheme Using Singular Value Decomposition. *KSII Trans. Internet Inf. Syst.* **2016**, *10*, 5831–5848, https://doi.org/10.3837/tiis.2016.12.008.
11. Singh, D.; Singh, S.K. Effective self-embedding watermarking scheme for image tampered detection and localization with recovery capability. *J. Vis. Commun. Image Represent.* **2016**, *38*, 775–789, https://doi.org/10.1016/j.jvcir.2016.04.023.

12. Qin, C.; Ji, P.; Zhang, X.; Dong, J.; Wang, J. Fragile image watermarking with pixel-wise recovery based on overlapping embedding strategy. *Signal Process.* **2017**, *138*, 280–293, https://doi.org/10.1016/j.sigpro.2017.03.033.
13. Lee, C.-F.; Shen, J.-J.; Chen, Z.-R.; Agrawal, S. Self-Embedding Authentication Watermarking with Effective Tampered Location Detection and High-Quality Image Recovery. *Sensors* **2019**, *19*, 2267, https://doi.org/10.3390/s19102267.
14. Lee, C.-F.; Shen, J.-J.; Hsu, F.-W. A Survey of Semi-Fragile Watermarking Authentication. In Recent Advances in Intelligent Information Hiding and Multimedia Signal Processing; Pan, J.-S., Ito, A., Tsai, P.-W., Jain, L., Eds.; Smart Innovation, Systems and Technologies; Springer: Cham, Switzerland, 2019; Volume 109. https://doi.org/10.1007/978-3-030-03745-1_33.
15. Rakhmawati, L.; Wirawan, W.; Suwadi, S. A recent survey of self-embedding fragile watermarking scheme for image authentication with recovery capability. *EURASIP J. Image Video Process.* **2019**, *2019*, 61, https://doi.org/10.1186/s13640-019-0462-3.
16. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998,** *86*, 2278–2324. doi:10.1109/5.726791
17. Theis, L.; Shi, W.; Cunningham, A.; Huszár, F. Lossy Image Compression with Compressive Autoencoders. arXiv 2017, arXiv:1703.00395.
18. Li, P.; Pei, Y.; Li, J. A comprehensive survey on design and application of autoencoder in deep learning. *Appl. Soft Comput.* **2023**, *138*, https://doi.org/10.1016/j.asoc.2023.110176.
19. Shen, J.-J.; Lee, C.-F.; Hsu, F.-W.; Agrawal, S. A self-embedding fragile image authentication based on singular value decomposition. *Multimedia Tools Appl.* **2020**, *79*, 25969–25988, https://doi.org/10.1007/s11042-020-09254-1.
20. Rezaei, M.; Taheri, H. Digital image self-recovery using CNN networks. *Optik* **2022**, *264*, https://doi.org/10.1016/j.ijleo.2022.169345.
21. Zhang, X.; Wang, S. Fragile Watermarking With Error-Free Restoration Capability. *IEEE Trans. Multimedia* **2008**, *10*, 1490–1499, https://doi.org/10.1109/tmm.2008.2007334.
22. Li, C.; Wang, Y.; Ma, B.; Zhang, Z. A novel self-recovery fragile watermarking scheme based on dual-redundant-ring structure. *Comput. Electr. Eng.* **2011**, *37*, 927–940, https://doi.org/10.1016/j.compeleceng.2011.09.007.
23. Chow, Y.-W.; Susilo, W.; Tonien, J.; Zong, W. A QR Code Watermarking Approach Based on the DWT-DCT Technique. In Information Security and Privacy – ACISP 2017; Lai, J., Ed.; Proceedings of the 22nd Australasian Conference on Information Security and Privacy, Auckland, New Zealand, 3–5 July 2017; Springer: Cham, Switzerland, 2017; Volume 10343, pp. 314–331.
24. Jiang, F.; Tao, W.; Liu, S.; Ren, J.; Guo, X.; Zhao, D. An End-to-End Compression Framework Based on Convolutional Neural Networks. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *28*, 3007–3018, https://doi.org/10.1109/tcsvt.2017.2734838.
25. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
26. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612, https://doi.org/10.1109/tip.2003.819861.
27. Liu, Z.; Luo, P.; Wang, X.; Tang, X. Deep Learning Face Attributes in the Wild. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 3730–3738.
28. OpenCV Team. Color Conversions. OpenCV Documentation. Available online: https://docs.opencv.org/4.x/de/d25/imgproc_color_conversions.html (accessed on 17 May 2025).
29. Sarreshtedari, S.; Akhaee, M.A. A Source-Channel Coding Approach to Digital Image Protection and Self-Recovery. *IEEE Trans. Image Process.* **2015**, *24*, 2266–2277, https://doi.org/10.1109/tip.2015.2414878.
30. Yuan, X.; Li, X.; Liu, T. Gauss–Jordan elimination-based image tampering detection and self-recovery. *Signal Process. Image Commun.* **2021**, *90*, https://doi.org/10.1016/j.image.2020.116038.
31. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv* **2014**, arXiv:1412.6572.