Learning Robust Image Watermarking with Lossless Cover Recovery

Supplementary Material

Table 3. Comparison of SSIM and PSNR (mean ± standard deviation) for the proposed methods with watermark lengths of 144 bits and 64 bits. Results are reported for non-recoverable (†) and cover-recoverable (†) variants. The best SSIM and PSNR values are highlighted in bold.

Method	SSIM	PSNR
144†	0.9989 ± 0.0008	37.03 ± 0.86
144-R†	0.9988 ± 0.0008	36.97 ± 0.88
64†	**0.9995 ± 0.0005**	**40.95 ± 1.24**
64-R†	**0.9995 ± 0.0005**	40.90 ± 1.23

A. Up/Down-Sampling Network Architecture

In the CRMark framework, due to the significant dimensional differences between images and watermarks, the network architectures for $\mathcal{U}_i(\cdot)$, $\mathcal{Q}_i(\cdot)$, and $\mathcal{S}_i(\cdot)$ within the integer coupling layer are different, specifically, $\mathcal{U}_i(\cdot)$ is designed as an up-sampling network, while $\mathcal{Q}_i(\cdot)$ and $\mathcal{S}_i(\cdot)$ serve as down-sampling networks, with their architectures illustrated in Fig. 9, we symmetrically design the symmetric up/down-sampling networks. Each network comprises an initial convolutional layer, multiple sampling blocks, and a final convolutional layer. Taking the down-sampling network as an example, the input features first undergo initial convolution, followed by processing through multiple sampling blocks, and conclude with a final convolutional layer to produce the extracted watermark features. Each sampling block incorporates a spatial attention module, reducing the spatial resolution of the features by half while increasing the number of feature channels by N_f. Conversely, the up-sampling network reconstructs the image from the watermark feature map. Through a combination of transposed convolutions, spatial attention, and convolutional layers, each block doubles the spatial resolution and symmetrically reduces the number of channels, with the final convolutional layer restoring the image.

B. Embedding More Watermark Bits

We evaluate CRMark with 64-bit and 144-bit embeddings in terms of visual quality, robustness, and auxiliary bitstream length. Fig. 10 shows the visual comparison. According to Tab. 3, 64-bit CRMark achieves an SSIM of 0.9995 ± 0.0005 and a PSNR of 40.95 ± 1.24 dB. CRMark-R performs similarly. In contrast, 144-bit CRMark and CRMark-R reach SSIMs of 0.9989 ± 0.0008 and 0.9988 ± 0.0008, and PSNRs of 37.03 ± 0.86 and 36.97 ± 0.88 dB, respectively. The 144-bit variant drops by 3.92 dB in PSNR and 0.0006 in SSIM. Experiments reveal that increasing capacity degrades image quality.

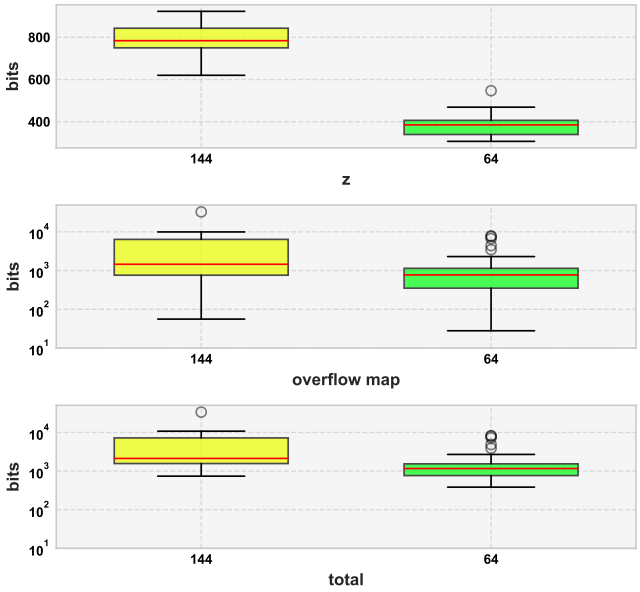

Figure 8. Comparisons of auxiliary bitstream lengths for 64-bit and 144-bit CRMark demonstrate that the auxiliary bitstream length increases with the embedding capacity.

Moreover, we evaluate the robustness of 64-bit CRMark and 144-bit CRMark under distortions shown in Tab. 4. Under JPEG compression ($Q_f = 50$), 64-bit CRMark achieves 99.27%, 64-bit CRMark-R reaches 99.31%, 144-bit CRMark hits 98.99%, and 144-bit CRMark-R scores 99.04%. For Gaussian blur ($\sigma = 7.0$), 64-bit CRMark obtains 93.87%, 64-bit CRMark-R gets 93.83%, 144-bit CRMark drops to 76.03%, and 144-bit CRMark-R reaches 76.12%. Under S&P noise ($p = 0.7$), 64-bit CRMark achieves 95.53%, 64-bit CRMark-R scores 95.88%, 144-bit CRMark records 85.73%, and 144-bit CRMark-R hits 85.80%. We note that 64-bit variants outperform 144-bit ones. Although capacity rises to 144 bits, there is only a slight decline compared to 64 bits.

We examine the auxiliary bitstream lengths for 64-bit and 144-bit watermarks in CRMark. Fig. 8 shows the results. We observe that the latent variable $\mathbf{z}$ of the 64-bit CRMark watermark averages 382.61 bits. Its overflow map averages 1558.58 bits. The total length reaches 1962.19 bits. We found that the latent variable z of the 144-bit CRMark watermark averages 782.84 bits. Its overflow map averages 4199.10 bits. The total length amounts to 5002.94 bits. The 144-bit CRMark bitstream surpasses the 64-bit CRMark by 2.5 times. We conclude that a higher watermark capacity increases auxiliary bitstream length. Nevertheless, all test images are successfully embedded by 144-bit CRMark.

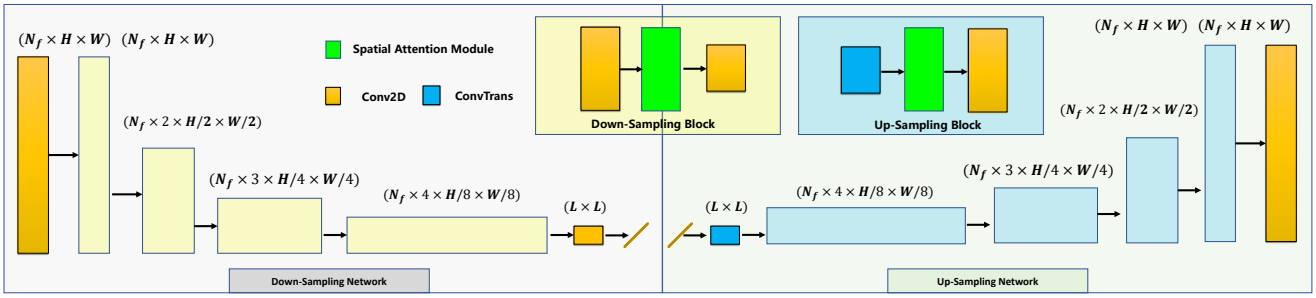

Figure 9. The architecture of the up-sampling and down-sampling networks in the proposed CRMark. The up-sampling and down-sampling networks are symmetric, each consisting of an initial convolution, multiple sampling blocks, and a final convolution.

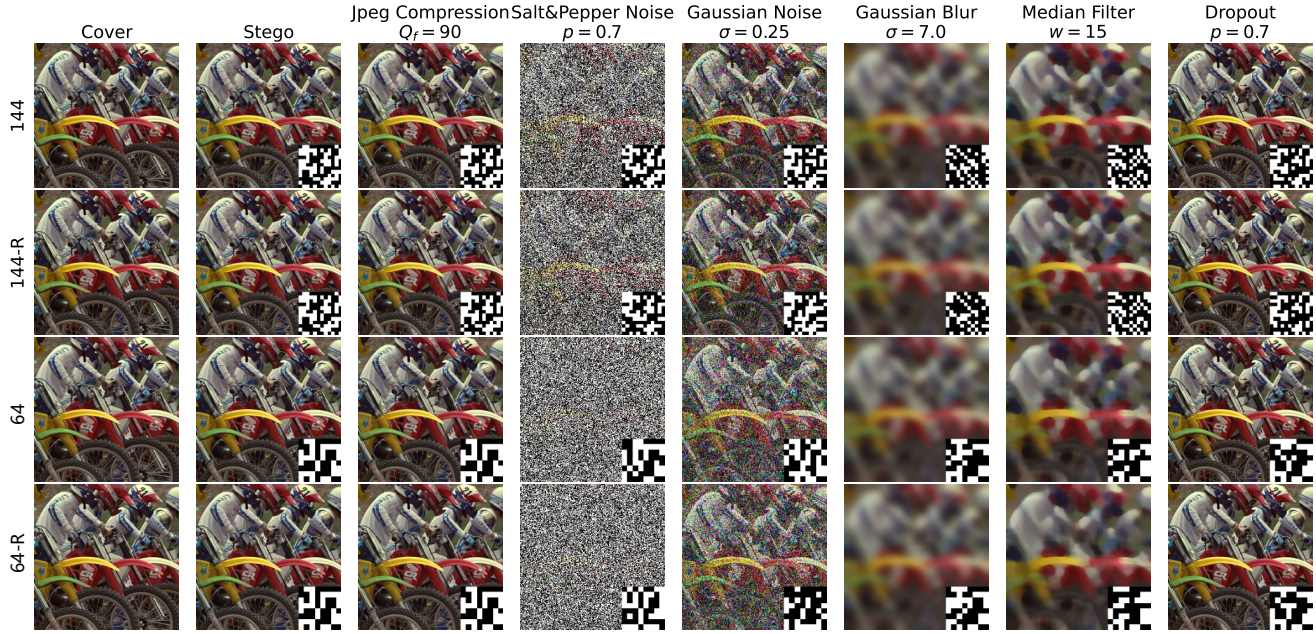

Figure 10. Visual comparison of stego images generated by 64-bit and 144-bit CRMark.

Table 4. Extraction accuracy (%) of the proposed CRMark variants embedding watermarks of lengths 64 and 144 (denoted as 64, 144) under various distortions. Here, † indicates cover-recoverable watermark methods, while † denotes non-recoverable ones.

Methods	JPEG(Q_f)			Gaussian Blur(σ)			Gaussian Noise(σ)		
	50	70	90	5.0	6.0	7.0	0.05	0.15	0.25
144†	98.99	99.26	99.31	99.01	94.91	76.03	99.37	98.52	92.92
144-R†	99.04	99.15	99.44	99.26	94.96	76.12	99.28	98.30	93.50
64†	99.27	100.0	100.0	98.70	97.11	93.87	100.0	97.59	88.87
64-R†	99.31	100.0	100.0	98.77	97.00	93.83	100.0	97.56	89.16
Methods	S&P(p)			Dropout(p)			Median Filter(w)		
	0.3	0.5	0.7	0.3	0.5	0.7	11	13	15
144†	97.94	95.77	85.73	98.72	97.02	93.21	98.81	98.21	95.81
144-R†	98.01	95.70	85.80	98.66	96.98	93.37	98.84	97.89	96.03
64†	99.86	99.06	95.53	99.98	97.75	92.35	99.41	98.83	97.52
64-R†	99.85	99.28	95.88	99.99	97.68	92.40	99.39	98.77	97.61

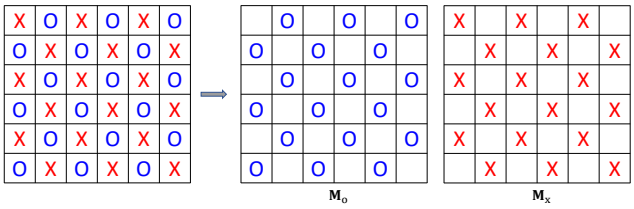

Figure 11. Illustration of the checkerboard masks $\mathbf{M}_o$ and $\mathbf{M}_x$ for pixel grouping in PEE-based embedding.

C. Attack Detection

CRMark leverages the SHA-256 cryptographic hash function to provide a reliable authentication mechanism. During the second embedding stage, the 256-bit hash of the original cover image is encoded and embedded into the clipped stego image along with other auxiliary information using RDH. Upon recovery, the hash of the reconstructed image is recomputed and compared to the embedded one. Any mismatch between the two indicates that the image has been altered or attacked. Thanks to the avalanche effect of SHA-256, where even a single-pixel change results in a completely different hash, CRMark can detect both seen and unseen attacks with high sensitivity. This makes it a robust solution for integrity verification in real-world applications. Moreover, the length of the 256-bit hash is small compared to the overall size of the merged auxiliary bitstream, making it negligible in terms of RDH capacity consumption. Therefore, no significant performance trade-off is introduced.

D. Reversible Data Hiding

CRMark employs a prediction-error expansion (PEE) method [6] in the second stage to achieve reversible embedding of auxiliary bitstreams. Compared to traditional histogram shifting techniques, this method offers a larger embedding capacity. The detailed procedure is as follows.

D.1. Embedding

Let $\mathbf{I} \in \{0, 1, \ldots, 255\}^{H \times W \times C}$ denote the original image to be embedded. The image is divided into two disjoint pixel sets, indicated by binary masks $\mathbf{M}_o, \mathbf{M}_x \in \{0, 1\}^{H \times W \times C}$. A checkerboard-like pattern is typically used for these masks, as illustrated in Fig. 11.

First, watermark bits are embedded into the pixel positions marked by $\mathbf{M}_x$. We extract $\mathbf{I}_o = \mathbf{I} \odot \mathbf{M}_o$ and use it as input to a predictor $\mathcal{P}(\cdot)$ to estimate the pixel values at $\mathbf{M}_x$, resulting in $\hat{\mathbf{I}}_x = \mathcal{P}(\mathbf{I}_o)$. Then, we extract $\mathbf{I}_x = \mathbf{I} \odot \mathbf{M}_x$ and compute the prediction error as $\mathbf{I}_x^{\text{error}} = \hat{\mathbf{I}}_x - \mathbf{I}_x$. We collect the prediction errors at $\mathbf{M}_x$ and construct a histogram, as shown in Fig. 12. Two peak values e_1 and e_2 are selected from the histogram, where $e_1 < e_2$. The pixel values are then shifted as follows: pixels with errors less than e_1 are

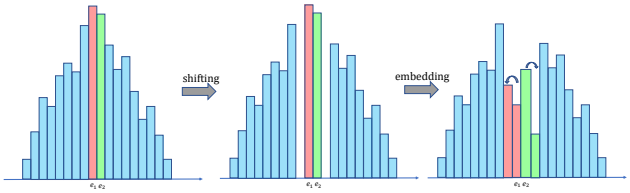

Figure 12. Histogram shifting based reversible data hiding.

decremented by one, while those with errors greater than e_2 are incremented by one. This creates two empty bins between e_1 and e_2, as illustrated in Fig. 12. For pixels with prediction errors equal to e_1 or e_2, a single watermark bit $w \in \{0, 1\}$ is embedded by updating their pixel values as: $p_{e_1} = p_{e_1} - w$ and $p_{e_2} = p_{e_2} + w$, where p_{e_1} and p_{e_2} represent the pixel values corresponding to prediction errors e_1 and e_2, respectively. After embedding, the stego image $\mathbf{I}_x^{\text{stego}}$ corresponding to $\mathbf{I}_x$ is obtained.

Next, we embed watermark bits into $\mathbf{I}_o$ using $\mathbf{I}_x^{\text{stego}}$ as input to the predictor, yielding $\hat{\mathbf{I}}_o = \mathcal{P}(\mathbf{I}_x^{\text{stego}})$. The prediction error is computed as $\mathbf{I}_o^{\text{error}} = \hat{\mathbf{I}}_o - \mathbf{I}_o$, and the same histogram-shifting-based embedding process is applied to $\mathbf{I}_o$. To ensure reversibility, auxiliary information such as the peak values e_1, e_2, and overflow locations must be embedded. This information is compressed using arithmetic coding [5] and subsequently embedded into the least significant bits (LSBs) of the first few rows of the original image $\mathbf{I}$. The original LSBs of $\mathbf{I}$ are preserved using the PEE method described above, enabling lossless recovery.

D.2. Extraction

To extract the embedded watermark and recover the original image $\mathbf{I}$, we first retrieve the compressed auxiliary information from the least significant bits (LSBs) of the first few rows of the stego image $\mathbf{I}^{\text{stego}}$. This data includes the histogram peak values e_1 and e_2, overflow locations, and other metadata necessary for reversibility.

Then, the stego image $\mathbf{I}^{\text{stego}}$ is divided into two disjoint pixel sets using checkerboard masks $\mathbf{M}_o$ and $\mathbf{M}_x$, resulting in $\mathbf{I}_x^{\text{stego}} = \mathbf{I}^{\text{stego}} \odot \mathbf{M}_x$ and $\mathbf{I}_o = \mathbf{I}^{\text{stego}} \odot \mathbf{M}_o$. We use $\mathbf{I}_x^{\text{stego}}$ as input to the predictor $\mathcal{P}(\cdot)$ to estimate the pixel values at positions marked by $\mathbf{M}_o$, yielding $\hat{\mathbf{I}}_o = \mathcal{P}(\mathbf{I}_x^{\text{stego}})$, and compute the prediction error $\mathbf{I}_o^{\text{error}} = \hat{\mathbf{I}}_o - \mathbf{I}_o$. Based on the known peak values e_1 and e_2, we analyze the prediction errors to extract the embedded watermark bit w. When a pixel's error falls near e_1 or e_2, it indicates that a watermark bit was embedded. Specifically, by identifying whether the error corresponds to e_1, $e_1 - 1$, e_2, or $e_2 + 1$, we determine the value of w as either 0 or 1. After extracting the watermark, we restore the histogram to its original state to ensure lossless reconstruction. Pixels with an error of $e_1 - 1$ are incremented by one, and those with an error of $e_2 + 1$ are decremented by one, returning all modified pixels to their

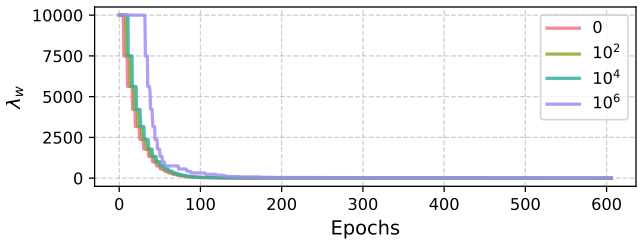

Figure 13. Curves of λ_w over training epochs for different λ_p, demonstrating its automatic adjustment and convergence.

pre-embedding values. Next, we reconstruct the pixel values at positions marked by $\mathbf{M}_{\mathrm{x}}$. Using the recovered $\mathbf{I}_{\mathrm{o}}$ as input, we generate $\hat{\mathbf{I}}_{\mathrm{x}} = \mathcal{P}(\mathbf{I}_{\mathrm{o}})$ and compute the prediction error $\mathbf{I}_{\mathrm{x}}^{\mathrm{error}} = \hat{\mathbf{I}}_{\mathrm{x}} - \mathbf{I}_{\mathrm{x}}^{\mathrm{stego}}$. Based on the known e_1 and e_2, we apply the inverse of the histogram-shifting operation to revert $\mathbf{I}_{\mathrm{x}}^{\mathrm{stego}}$ back to its original form.

Finally, to achieve full lossless recovery, we also restore the LSBs of the original image that were used to embed the auxiliary information. During embedding, the original LSB values were preserved using the PEE method and encoded into the auxiliary bitstream. In extraction, these values are retrieved and used to restore the LSB layer to its original state. This completes the reversible extraction process, allowing both the watermark and the original image to be perfectly recovered without any distortion.

Based on the above processes, we can observe that the embedding capacity depends on the distribution of prediction errors. More accurate prediction leads to a more concentrated error distribution, resulting in higher histogram peaks and thus increased embedding capacity. Therefore, employing a better predictor $\mathcal{P}(\cdot)$, such as a CNN-based model [1, 2], can improve overall performance.

E. How to train CRMark?

During training, different loss terms usually require careful weighting to stabilize the optimization process. However, in training CRMark, we found that the overflow penalty term often requires a relatively large loss of weight to constrain pixel overflow effectively. This leads to a problem: if the weight of the watermark loss is too small, the watermark may fail to embed; if it is too large, excessive distortion in the stego image can occur. To address this issue, we propose a dynamic loss weight adjustment strategy.

In watermarking tasks, the primary goal is to embed the watermark and accurately extract it, while preserving image quality. Importantly, watermark extraction accuracy is often prioritized over image imperceptibility during training. Therefore, *can we adjust the watermark loss weight to gradually balance the watermark extraction accuracy and imperceptibility by fixing the weights of other losses?* We consider this idea is feasible. Specifically, we initialize the watermark loss weight with a large value and adaptively adjust it based on extraction accuracy over recent epochs.

Let $\mathbf{A} = [a_{t-n}, \ldots, a_t]$ denote the extraction accuracies from epoch $t-n$ to t. The average accuracy is computed as:

$$\mathrm{acc} = \frac{1}{n} \sum_{i=t-n}^{t} \mathbf{A}(i),$$

where n is the average window size, when acc exceeds a target threshold or the extraction error falls below a preset tolerance δ, it indicates stable watermark embedding under the current weight λ_w^t. We then reduce the weight to enhance image quality:

$$\lambda_{\mathrm{w}}^{t+1} = \lambda_{\mathrm{w}}^{t} \times v, \quad \text{if acc} > 1 - \delta,$$

where $\delta \in (0, 1)$ is the tolerable error, and $v \in (0, 1)$ is a discount factor. This adaptive adjustment stabilizes training and facilitates a progressive balance between watermark robustness and image quality. Fig. 13 shows the variation of the watermark loss weight under different penalty loss weights. It can be observed that as the number of epochs increases, the watermark loss weight gradually decreases and eventually converges.

F. Limitations and Discussions

While CRMark achieves better performance than STDM [3] and PZM [4] in terms of computational efficiency, auxiliary bitstream length, visual quality, and general robustness, it still has limitations. One major drawback is its lack of robustness to geometric distortion, especially image rotation. In contrast, STDM [3] and PZM [4] are more resilient to such attacks. Therefore, a promising direction for future work is to improve CRMark's resistance to geometric transformations. In addition, although CRMark supports embedding more watermark bits, an interesting question remains: *Can the embedding capacity be further increased by integrating integer flow-based lossless compression techniques?* For example, achieving *reversible image-in-image hiding* could open up an interesting research direction.

References

[1] Ruohan Hu and Shijun Xiang. Reversible data hiding by using cnn prediction and adaptive embedding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(12):10196–10208, 2021. 4

[2] Ruohan Hu and Shijun Xiang. Cnn prediction based reversible data hiding. *IEEE Signal Process. Lett.*, 28:464–468, 2021. 4

[3] Yichao Tang, Chuntao Wang, Shijun Xiang, and Yiu-ming Cheung. A robust reversible watermarking scheme using attack-simulation-based adaptive normalization and embedding. *IEEE Trans. Inf. Forensics Secur.*, 33(4):1593–1609, 2024. 4

[4] Yichao Tang, Shuai Wang, Chuntao Wang, Shijun Xiang, and Yiu-ming Cheung. A highly robust reversible watermarking scheme using embedding optimization and rounded error compensation. *IEEE Trans. Circuits Syst. Video Technol.*, 33(4):1593–1609, 2022. 4

[5] Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, 1987. 3

[6] Vasiliy Sachnev, Hyoung Joong Kim, Jeho Nam, Sundaram Suresh, and Yun Qing Shi. Reversible watermarking algorithm using sorting and prediction. *IEEE Trans. Circuits Syst. Video Technol.*, 19(7):989–999, 2009. 3