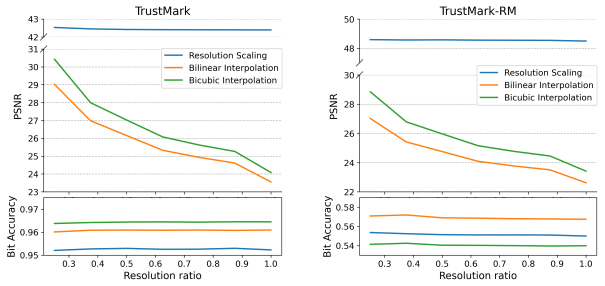
TrustMark: Robust Watermarking and Watermark Removal for Arbitrary Resolution Images

Supplementary Material



(a) TrustMark (b) ark

Figure 7. Resolution Scaling performance versus Bilinear and Bicubic interpolations for the watermarking task, evaluated with (a) TrustMark ($\alpha_{\max} = 27.5$) and (b) ReMark on DIV2K at different resolutions.

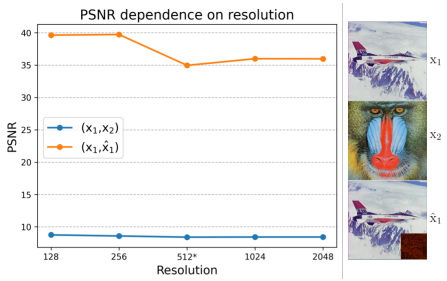


Figure 8. PSNR is dependent to the evaluating resolution and is more sensitive if the two images are similar (as in the case of watermarking). * refers to the original resolution of the evaluated images. $\hat{x}_1$ inset: pixel residual scaled 20 times for visualization purpose.

Size	PSNR (clean)	Acc. (clean)	Acc. (noised)
32	54.52	0.56	0.52
64	49.98	0.93	0.78
128	47.43	1.00	0.89
192	46.22	1.00	0.92
256	44.92	1.00	0.94
1024	45.42	1.00	0.93
4096	45.43	1.00	0.93

Table 5. Performance on very small and very large resolutions.

6. Resolution Scaling robustness

We examine the robustness of Resolution Scaling against resolution change in term of preserving image quality and watermark recovery. We first create multiple versions of the DIV2K benchmark by downsampling every image with a factor $0 < \rho < 1$, then evaluate our watermark embedding (TrustMark, Fig. 7a) and removal (ReMark, Fig. 7b) on these newly created benchmarks. We also compare with Bilinear and Bicubic interpolation methods *i.e.* output images at 256×256 resolution are scaled to the target resolution via Bilinear or Bicubic samplings instead of Resolution Scaling. Fig. 7 shows that all methods drop performance in PSNR when the target resolution increases beyond 256×256 towards the original image resolution, however the drop for Resolution Scaling is negligible (0.1dB for both TrustMark and ReMark) as compared with Bilinear and Bicubic interpolations. As a trade-off, the bit accuracy for Resolution Scaling is lower than Bicubic by 1% for TrustMark (higher is better) and ReMark (lower is better).

Robustness against very small and very large images. Here Trustmark is further tested on a wider range of resolutions. Using the MetFace [42] dataset, 7 different datasets are created by resizing all images to a fixed resolution between 32×32 to 4096×4096. Shown in Tab. 5, are the resolution of the images in the dataset, image quality (PSNR) after watermarking, average bit accuracy for clean images, and average bit accuracy for images with noise. On average, Trustmark can perfectly decode the clean watermark images at resolutions as small as 128×128 and as large as 4096×4096. Beyond the experiment reported in the table, we have tested adhoc examples up to around 10K pixels shortest side without degradation. Trustmark performance reduces to chance at 32×32. This is because the watermark is encoded at a resolution of 256, and we hypothesize that the watermark details are lost after ×8 severe downsampling of the residual. This is also evident by the higher PSNR value at lower resolutions (54.52 at 32). On the other hand, Trustmark generalizes perfectly on resolutions larger than 256 ($4096/256 = 16$), a more important use-case given the high-resolution cameras in today's world.

Note on PSNR's resolution dependence. We conducted an experiment to assess the dependence of the PSNR metric to image resolution (Fig. 8). Here, we compute PSNR on a pair of 2 totally different images (x_1, x_2) and a pair of cover and watermarked images $(x_1, \hat{x}_1)$. All images x_1, x_2 and $\hat{x}_1$ have the same resolution of 512×512 and the watermarking algorithm is dwtDctSvd [51]. PSNR is computed at the original resolution, and also at various downsampling and upsampling resolutions of the 2 pairs. While the PSNR scores for (x_1, x_2) vary slightly with 0.35dB maximum difference, the change in PSNR for $(x_1, \hat{x}_1)$ is significantly higher (4.7dB). This behavior does not depend on the interpolation (Bilinear, Bicubic) nor the watermarking methods. This indicates that PSNR is more sensitive to similar image pairs (as is the case of watermarking) when evaluated at different resolutions. It is because the subtle watermark degrades after up/down-sampling, increasing PSNR

Method	PSNR (clean)	Acc. (noised)
dwtDct [51]	39.40±1.87	0.51±0.06
OutGuess [58]	31.62±1.50	0.56±0.06
Jsteg [70]	31.67±1.50	0.56±0.06

Table 6. More classical comparisons on DIV2K.

in overall. In the broader context, if two watermark models work on two different image resolutions and are evaluated at its model-designed resolution, the resulting PSNR scores will not be directly comparable. This highlights the contribution of our Resolution Scaling algorithm in extending fixed-resolution watermarking methods to work on original resolution, therefore can be **directly compared with each other and with other arbitrary-resolution methods**.

7. Supporting quantitative experiments

7.1. Comparison with other traditional methods

While our main comparison baselines are the recent deep learning methods, we include dwtDctSvd [51], in Section Sec. 4, as a classical representative. This technique was chosen as it is receiving use via watermarking of Stable Diffusion generative images, and so is relevant to our content provenance use case and also has public code available.

Here, we include the results of 3 additional traditional methods commonly used in older literature, dwtDct [51], OutGuess [58] and Jsteg [70] in Tab. 6. Similar to dwtDctSvd, all these methods perform with lower bit accuracy given the complexity of modern noises introduced on the test images. They also underperform TrustMark on PSNR by a large margin (c.f. Tab. 1).

7.2. ReMark baselines

σ	0.08	0.09	0.10	0.20	0.3	0.4
PSNR	28.07	27.14	26.28	10.07	9.23	7.15
Bit acc.	0.99	0.99	0.98	0.73	0.66	0.51

Table 7. Watermark removal using Zhao *et al*. [79] method. σ is the noise strength.

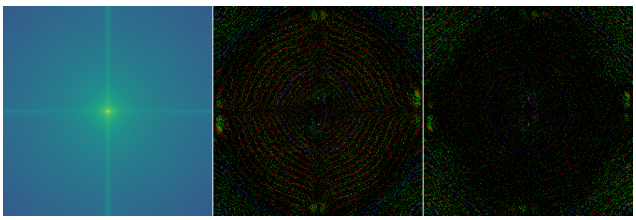


Figure 9. Effects of FFL in frequency domain: (left) average frequency spectrum of cover images in the MetFace benchmark, (middle) residual spectrum without FFL and (right) with FFL. The residual images are amplified 100x for visualization purpose.

We show in Tab. 3 our comparison of ReMark with Zhao *et al*. [79] – the only method specifically designed for invisible watermark removal to the best of our knowledge. We use the recommended Gaussian noise with standard deviation $\sigma = 0.3$ as the noise source, and KBNet [78] as the recovery model. Here, we provide the full details of our experiments with [79] at different noise strength levels in Tab. 7. It can be seen that the image quality reduces drastically as more noise is required to break the target model (TrustMark). We select $\sigma = 0.3$ as the optimal value for this baseline, as shown in the main paper.

8. Qualitative experiments

8.1. FFL effects

Fig. 9 shows the impact of FFL on the frequency spectrum of the watermarked images in the MetFace benchmark [42]. Compared with TrustMark without FFL (Fig. 9 middle), having this loss improves image quality across multiple frequency bands (note the brightness of the 4 regions at upper, lower, left and right of Fig. 9 right image).

8.2. Post-processing layers

We visualize several activation maps of the pre- and post-process layers in Fig. 10, along with the learned watermark and output image. The watermark is shown to located at the center of the image where salient objects are dominant and also probably because of random crop simulation during training. The watermark is shown visibly blended within the image content after pre-processing and perceptually invisible after the post-processing layers.

8.3. TrustMark on control images

We visualize the watermarking artifacts by applying TrustMark on a set of control images with a fixed watermark in Fig. 11. The learned watermark is shown to be adaptive to the image content (most visible when comparing the third column with the first and second columns). Additionally, the watermark complexity increases along with the content (left to right), also the diagonal object edges tend to cause more artifacts than the horizontal and vertical edges.

9. Training details

9.1. Training stages

Our training of TrustMark consists of 4 stages in an end-to-end manner to encourage convergence (Sec. 4.1). Subsequent stages are automatically executed when certain conditions are met (Tab. 9). At stage 0, the input watermarks are randomized but the mini-batch of cover images are fixed at every training iteration. Additionally, the noise model **N** and GAN loss $\mathcal{L}_{\text{GAN+GP}}$ are turned off; the trade-off parameter α is set at a low value of 0.05 to prioritize training the watermark extractor **X**. When the training bit accuracy reaches 90%, random image batches of the entire training dataset are fed into the network (stage 1). Noise simulation is activated when a training batch achieves 95% accuracy (stage 2). Finally, at bit accuracy of 98%, the GAN loss is turned on and the trade-off parameter α in Eq. (1) linearly increases to α_{max} for the next 10,000 iterations. We find

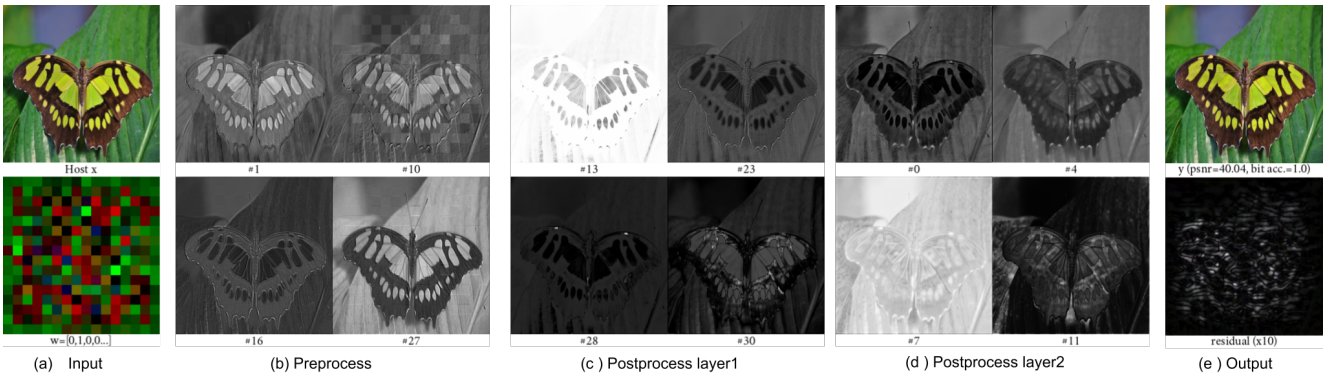


Figure 10. TrustMark pre- and post-process activation maps, best with zoom-in.

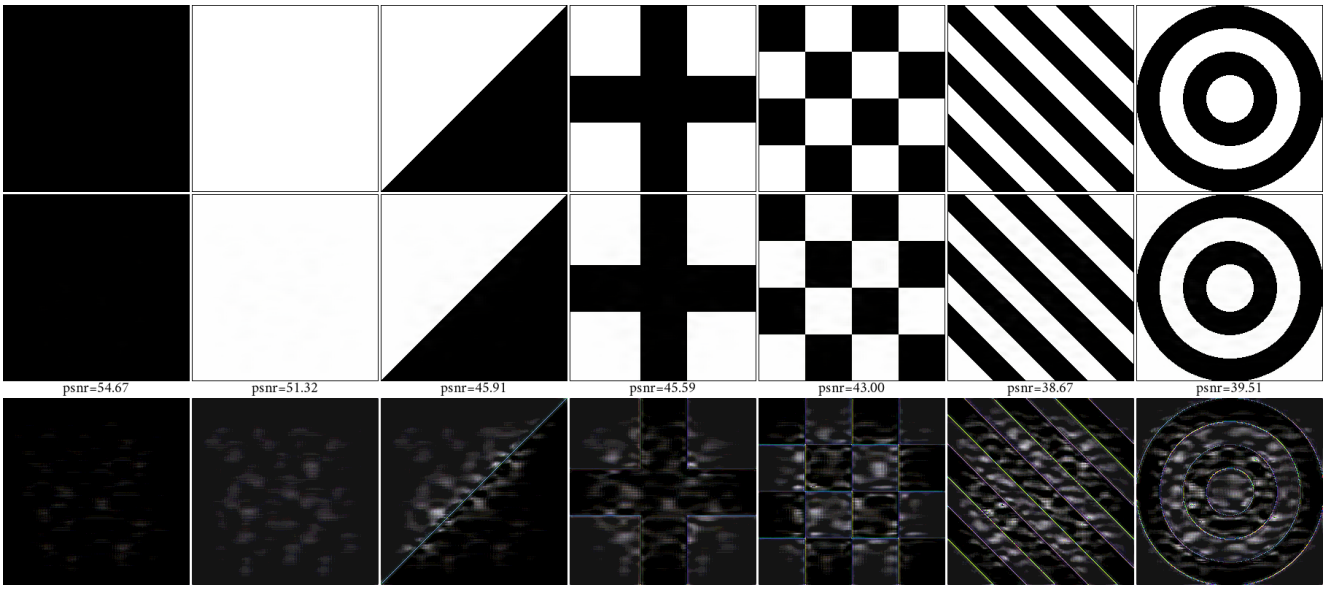


Figure 11. Watermarking on a set of control images. All achieves 100% bit accuracy. First row: host images , middle row: encoded images with a same random watermark; bottom row: residual images scaled up 20x.

	Jpeg Compress. (min q)	Brightness (bri.)	Contrast (con.)	Color Jiggle (bri., con., sat., hue)	Grayscale (p)	Gaussian Blur (k, σ)	Gaussian Noise (std.)
Low	70	0.9–1.1	0.9–1.1	0.05, 0.05, 0.05, 0.01	0.5	3, 0.1–1.0	0.02
Medium	50	0.75–1.25	0.75–1.25	0.1, 0.1, 0.1, 0.02	0.5	5, 0.1–1.5	0.04
High	40	0.5–1.5	0.5–1.5	0.1, 0.1, 0.1, 0.05	0.5	7, 0.1–2.0	0.08

	Hue (hue)	Posterize (bits)	RGB Shift (shift limit)	Saturation (sat.)	Sharpness (sha.)	Median blur (k)	Box blur (k)	Motion blur (k,β, t)
Low	0.01	5	0.02	0.9-1.1	0.5	3	3	3-5,-25–25,-0.25–0.25
Medium	0.02	4	0.05	0.75-1.25	1.0	3	5	3-7,-45–45,-0.5–0.5
High	0.05	3	0.1	0.5-1.5	2.5	3	7	3-9,-90–90,-1.0–1.0

Table 8. Noise simulation settings at low, medium and high levels. Notation *a–b* refers to the value range between *a* and *b* that a parameter is randomly drawn from. For Jpeg compression, q is the image quality factor. For blurring operations, k is the kernel size. Additionally, for motion blur, β and t are the angle and direction of the simulated motion. *bri., con., sat.* and *sha.* are the brightness, contrast, saturation and sharpness factors in Kornia [60].

Stage	Settings					Triggered bit acc.
	Fixed batch	Rnd. batch	Noise simulation	α ramping	GAN	
0	✓					-
1		✓				0.90
2		✓	✓			0.95
3		✓	✓	✓	✓	0.98

Table 9. Four training stages of TrustMark. The next stage is automatically triggered once a training batch bit accuracy exceed a threshold.

Figure 12. Examples of different noise sources applied during TrustMark training.

Method	emoji	meme	pilgram
RoSteALS[13]	0.98	0.70	0.97
RivaGAN[76]	0.93	0.94	0.88
dwtDctSvd[28]	**0.99**	0.50	0.49
TrustMark	**0.99**	**0.97**	**0.98**

Table 10. Bit Accuracies on untrained social noise and attacks.

these 4-stage settings aid convergence and are particularly useful when high-severity noise simulation, large $\alpha_{\max}$ or watermark size are required. A typical training of TrustMark takes 500, 600 and 800 iterations to reach stage 1, 2 and 3 respectively.

9.2. Noise severity

Tab. 8 shows details of our noise settings for our experiments in Fig. 4 and Sec. 4.3. We employ the Kornia library [60] for implementation of the differentiable transformations, except for Jpeg Compression in which the implementation algorithm comes from [63]. The geometrical transformations (not shown in Tab. 8) are kept fixed in all settings, specifically random flip with probability of 50%, random resize with scale range 0.8–1.0 and aspect ratio range 3/4–4/3 and crop size of 244 out of 256. Examples of the noises are shown in Fig. 12.

10. Social media robustness

Trustmark is deployed by several vendors for C2PA provenance recovery despite metadata stripping by social media platforms. TrustMark is known in practice to survive X, LinkedIn, Facebook and Instagram (IG) filters including IG cropping. Table 10 shows some social perturbations (emoji and meme addition, IG (pilgram) filters.

Additionally, we enclose a video demo demonstrating (i) TrustMark in embedding and decoding a watermark; and (ii) an use case for content provenance. Only in the scope of this demo, we use BCH [7] for error correction (we also report raw bit accuracy for completeness). For part (i), we demonstrate an application of Resolution Scaling that enables users to target a desired PSNR value at test time by optimizing λ in Algorithm 1. For part (ii), the watermark contains the hash key of the provenance manifest associated with the image. When uploaded to a social media platform (X.com), the provenance information can be recovered by using the extracted watermark to perform a key-value lookup in the provenance database. TrustMark works seamlessly with the C2PA open standards [16], and is formally accepted as an approved watermarking ('soft binding') technique, within C2PA version 2.1.