

A. Further Background

A.1. Discrete Wavelet Transform

The Discrete Wavelet Transform (DWT) is a method that converts an image from its spatial representation to a frequency-based representation. When applied to an image, the DWT creates four frequency subbands: low-low (LL), low-high (LH), high-low (HL), and high-high (HH). To achieve additional levels of DWT, each subband can be further decomposed into four new subbands. The LL subband, containing the lowest frequency components, holds most of the image’s weight and important structural information. While it is resilient to subtle, image-wide changes such as noise and compression, it demands precise editing to preserve image quality, as alterations in this band are more likely to create visible artefacts [30]. Higher frequency subbands (LH, HL, and HH) capture fine details and edges of the image. These components are more easily affected by typical image modifications, such as JPEG compression. To embed a watermark, we carefully alter the coefficients in the LL subband. After embedding, we can reconstruct the original image using the inverse DWT.

A.2. Entropy of Visual Data

Image entropy, specifically Shannon entropy, quantifies the randomness or information content of an image [28]. It is calculated from the empirical probability distribution of pixel intensities. For a grayscale image, entropy H is calculated as

$$H = - \sum p(i) \log(p(i)), \quad (1)$$

where $p(i)$ represents the frequency of intensity level i in the image. Intuitively, regions with high entropy signify areas in the image where there is a lot of variation in pixel intensities, such as edges, textures, and details. These complex regions are often more suitable for embedding watermarks because the changes introduced by the watermark are less likely to be noticeable to the human eye [26].

B. Advanced Considerations and Variants of MELB

This section provides the core algorithmic components behind our watermarking method. We provide pseudocode and formal descriptions of the embedding and detection procedures and formulate the statistical hypothesis test used for detection. We also present an optional extension of our method that strengthens the system against brute-force attacks and elaborate on the details of two further refinements that were used in our experiments in Sec. 5.4.

B.1. Algorithms

In this section, we present additional algorithms for the detection process, as described in Sec. 4.1 and Sec. 4.2. Al-

gorithm 1 and Algorithm 4 represent the main detection steps, while Algorithm 2 summarises the evaluation procedure. The statistical hypothesis test refers to the procedure described in Sec. 4.2, where we determine whether a block is watermarked by assessing the number of its DWT coefficients that fall within the allowed domain. The optional post_process step, introduced in Sec. 4.2.1, applies a low-pass filter to the detection map to enhance its utility as a visual aid. This post-processing step does not influence the final image-level score.

Algorithm 1 Embedding Algorithm for MELB

Input: Image I , interval length l , block size k , sub-block size m , number of DWT-levels d , DWT coefficient limit r , key sk

```

1: procedure embed( $I, l, k, m, d, r$ )
2:   Decompose image into channels  $R, G, B \leftarrow I_{:, :, 0}, I_{:, :, 1}, I_{:, :, 2}$ 
3:   for each channel  $C \in \{R, G, B\}$  do
4:     Partition  $C$  into non-overlapping  $m \times m$  blocks
5:     for each block  $b$  do
6:       seed  $\leftarrow$  get_seed( $b$ )
7:        $b^* \leftarrow$  embed_block( $b$ , seed,  $k, d, sk$ )
8:       Replace block  $b$  in  $C$  with watermarked block  $b^*$ 
9:     end for
10:  end for
11:  return Image made of watermarked  $R, G, B$  channels
12: end procedure
13:
14: procedure embed_block( $b$ , seed,  $k, d, l, r, sk$ )
15:   Use seed to generate a binary sequence  $S \leftarrow F_{sk}(seed)$ 
16:   Split the DWT domain interval  $[-r, r]$  into sub-intervals of length  $l$ 
17:   Generate red list  $R$  and green list  $G$  by colouring the intervals based on the sequence  $S$ 
18:   Partition  $b$  into  $k \times k$  sub-blocks
19:   for each sub-block  $s \in b$  do
20:     Apply  $d$  levels of DWT to  $s$  and extract LL coefficients
21:     for each coefficient  $c$  in  $s$  do
22:       if  $c \in R$  then
23:          $c^* \leftarrow$  perturb( $c, G$ )
24:       end if
25:     end for
26:   end for
27:   return Watermarked block  $b^*$ 
28: end procedure
29:
30: procedure perturb( $c, G$ )
31:   Find green interval  $[g_{min}, g_{max}] \in G$  that is closest to  $c$ 
32:    $\tilde{c} \leftarrow (g_{min} + g_{max})/2$ 
33:   return  $\tilde{c}$ 
34: end procedure

```

Algorithm 2 Evaluation Procedure

```
1: procedure evaluate( $\mathcal{S}, \alpha$ )
2:   Initialise detection map  $\mathcal{M}$  with same shape as  $\mathcal{S}$ 
3:   Initialise green count  $n \leftarrow 0$ 
4:   for each position  $(i, j)$  in  $\mathcal{S}$  do
5:      $\triangleright \mathcal{S}[i, j]$  stores the number of watermarked
       coefficients
6:     Perform one-sided hypothesis test on  $\mathcal{S}[i, j]$  at
7:     significance level  $\alpha$ 
8:     if  $H_0$  is rejected then
9:        $\mathcal{M}[i, j] \leftarrow \text{green}$ 
10:       $n \leftarrow n + 1$ 
11:     else
12:       $\mathcal{M}[i, j] \leftarrow \text{red}$ 
13:     end if
14:   end for
15:   post_process( $\mathcal{M}$ )  $\triangleright$  optional low-pass filter
16:   Compute score  $\leftarrow \frac{n}{|\mathcal{S}|}$ 
17:   return score,  $\mathcal{M}$ 
18: end procedure
```

Algorithm 3 Calculating the Seed

```
1: procedure get_seed( $s$ , round_val  $\leftarrow 30$ )
2:   Compute the mean pixel intensity  $\mu$  of all pixels in  $s$ 
3:   Round  $\mu$  to the nearest multiple of round_val
4:   return rounded value
5: end procedure
```

B.2. Parameter Configuration and Trade-offs

The effectiveness of our watermarking method depends on multiple parameters. The most crucial parameter in our method is the **interval length** l , which controls the trade-off between watermark robustness and visual imperceptibility: shorter interval lengths result in less visible watermarks but decreased robustness, while longer intervals enhance robustness at the potential cost of visual perceptibility (see Fig. 6). This is explained by the fact that larger interval lengths provide greater tolerance for variations in the DWT coefficients, allowing them to undergo minor modifications while staying in the green domain. When choosing excessively large interval lengths, we suggest embedding the watermark only in high entropy areas to reduce the visual impact. It is practical to define a consistent **range** $[-r, r]$ that will be considered for the DWT coefficients that applies to all watermarked images. This ensures that the intervals are uniformly divided and always start from the same reference point, allowing the reliable reconstruction of the interval colouring during detection. While we can compute the theoretical bounds for DWT coefficients and work with the whole range, we note that most coefficients in natural images fall significantly below this bound. To reduce computational complexity and storage, we define the coefficient range as a parameter, chosen to cover the region

Algorithm 4 Detection Algorithm for MELB

Input: Image I , interval length l , block size k , sub-block size m , number of DWT-levels d , significance level α , key sk

```
1: procedure detect( $I, l, k, m, d, \alpha$ )
2:   Decompose image into channels  $R, G, B \leftarrow$ 
        $I_{::,0}, I_{::,1}, I_{::,2}$ 
3:   Obtain score maps  $\mathcal{S}_R, \mathcal{S}_G, \mathcal{S}_B$  by applying
       detect_channel
4:   Compute mean score map  $\mathcal{S}_{\text{mean}}$  over all channels
5:   score, map  $\leftarrow \text{evaluate}(\mathcal{S}_{\text{mean}}, \alpha)$ 
6:   return score, map
7: end procedure
8:
9: procedure detect_channel( $C, l, k, m, d, \alpha$ )
10:  Partition  $C$  into non-overlapping  $m \times m$  blocks
11:  Initialise score matrix  $\mathcal{S}$  with the same shape as block grid
12:  for each block  $b$  at position  $(i, j)$  do
13:    seed  $\leftarrow \text{get\_seed}(b)$ 
14:     $\mathcal{S}[i, j] \leftarrow \text{detect\_block}(b, \text{seed}, k, d, sk)$ 
15:  end for
16:  return  $\mathcal{S}$ 
17: end procedure
18:
19: procedure detect_block( $b, \text{seed}, k, d, sk$ )
20:  Initialise countG  $\leftarrow 0$ 
21:  Use seed to generate a binary sequence  $S$ 
22:  Generate red list  $R$  and green list  $G$  according to  $S$ 
23:  Partition  $b$  into  $k \times k$  sub-blocks
24:  for each sub-block  $s \in b$  do
25:    Apply  $d$  levels of DWT to  $s$  and extract LL coefficients  $\mathcal{L}$ 
26:    Increment countG by  $|\mathcal{L} \cap G|$ 
27:  end for
28:  return countG
29: end procedure
```

where most DWT coefficients are expected to lie, and consider each value outside this range as allowed.

We propose to define the get_seed function as the rounded mean colour of each image block. Generating distinct seeds per block is crucial, as it prevents uniform allowed value ranges across blocks, thereby strengthening security by avoiding repetitive patterns exploitable by attackers. Additionally, deriving each seed from the block's content enhances robustness to cropping, since the seed can be independently recovered even if parts of the image are removed. Using randomly generated seeds for each block would necessitate storing all seeds for detection, which is impractical at scale due to high storage overhead. Therefore, using the mean channel average strikes a practical balance between security, efficiency, and storage complexity. Exploring more sophisticated hash functions for seed generation is left as future work.

When the seed is derived from the rounded mean colour

of the block, the **rounding value** presents a trade-off between the robustness against image manipulations and forgability. In general, a smaller rounding value would create more distinct seed values across the image, making it harder for malicious actors to gather sufficient samples of allowed and disallowed modifications for any particular seed value. This is critical because if too many blocks share the same seed value, an attacker could analyse the pattern of allowed and disallowed modifications across these blocks to approximate the watermarking rules. For instance, if multiple blocks with the same mean colour exhibit consistent patterns of allowed colour modifications, an attacker could learn and exploit these patterns to forge or remove the watermark. However, the rounding value cannot be too small, as it must also provide sufficient robustness against simple mean-shifting attacks. In Sec. D.3.1 we demonstrate through experimental validation, that a rounding value of 30 achieves this balance effectively. Any attempt to shift block mean values enough to change their rounding behaviour results in severe degradation of visual quality.

When employing DWT in image watermarking, it is common practice to apply two or three levels of DWT to enhance robustness [7, 16]. On top of this, we observe that using a higher number of DWT transforms helps to make the watermark more imperceptible, as illustrated in Fig. 5, since the embedding is spread more subtly across the transformed coefficients. In Tab. 3, we report the WDR for different levels of DWT. We observe that the level does not affect the WDR, as all results are at $\approx 100\%$. For this reason, it is generally preferable to choose a larger d . The **number of levels of DWT** d that can be applied to each sub-block is constrained by the **sub-block size** k . Since each level of DWT decomposition reduces the dimensions of the image (or sub-band) by half in both directions, k must be greater than or equal to 2^d . Additionally, the **block size** m must be a multiple of k to ensure proper application of the DWT decomposition to the whole image area.

# DWT levels	FPR@1%		FPR@5%	
	COCO	DALL-E	COCO	DALL-E
$d = 1$	100.0%	100.0%	100.0%	100.0%
$d = 2$	100.0%	100.0%	100.0%	100.0%
$d = 3$	99.8%	100.0%	100.0%	100.0%

Table 3. We watermark 500 images from both MS-COCO and DALL-E datasets using three different settings for the number of DWT levels d . We observe that all settings yield a highly detectable watermark, with detection rates at approximately 100% for every d .

B.3. Hypothesis Test for Watermark Detection

To reduce false positives, we use a statistical test per block instead of a fixed 50% threshold, making the method more

robust to natural variation. Our one-sided hypothesis test for each block (conducted at a 5% significance level in our experiments, can be adjusted by the user) is formulated as follows:

H_0 : The block is not watermarked. It violates the red-green rule in at least 50% of its DWT coefficients.

H_1 : The block is watermarked.

The significance level of the hypothesis test can be used to control the false positive rate. Increasing the significance threshold reduces the likelihood of misclassifying unwatermarked blocks as watermarked, which comes at the potential cost of slightly reduced robustness against manipulations.

If the null hypothesis is true, the number of green list coefficients c_G has an expected value of $N/2$ and a variance of $N/4$, where N is the total number of DWT coefficients after d levels of DWT transformations to the block. We can calculate the threshold c_G^* for the number of green list coefficients for which we start rejecting the null hypothesis as:

$$c_G^* = \frac{z_{0.95}}{2} \cdot \sqrt{N} + \frac{N}{2}. \quad (2)$$

Example 1. To better understand the impact of scheme parameters on detection, consider example parameter settings. For a block size of $m = 96$ and sub-block size of $k = 8$ with $d = 3$ levels of DWT, this yields a threshold of $c_G^* \approx 82$. This means that for a block to reject the null hypothesis and consider this block watermarked, at least 82 out of 144 DWT coefficients ($> 56.94\%$) need to be from the green domain.

B.4. Strengthening against Brute-Force Attacks

In Sec. 4.1 we assume that the user generates a single secret key sk which we keep fixed for the whole image. Our system can be strengthened against brute-force attacks by employing $K > 1$ distinct secret keys for watermark embedding. Instead of using a single secret key for the entire image, the algorithm randomly selects a key from a pre-defined set for each $m \times m$ block. During detection, we evaluate the image using all K possible keys and correct for multiple hypotheses via Bonferroni correction. When partitioning DWT coefficients, half of the intervals are defined as allowed regions, meaning that each coefficient will be in the allowed domain for $K/2$ keys and in the disallowed domain for the other $K/2$ [17]. This approach helps prevent statistical attacks since no consistent patterns can be easily discovered when analysing large numbers of watermarked images, even when an attacker attempts to aggregate statistics across multiple images.



Figure 5. The same image watermarked with different levels of DWT (using $l = 8$). As the number of DWT decomposition levels increases, the embedded watermark becomes more imperceptible to the human eye.

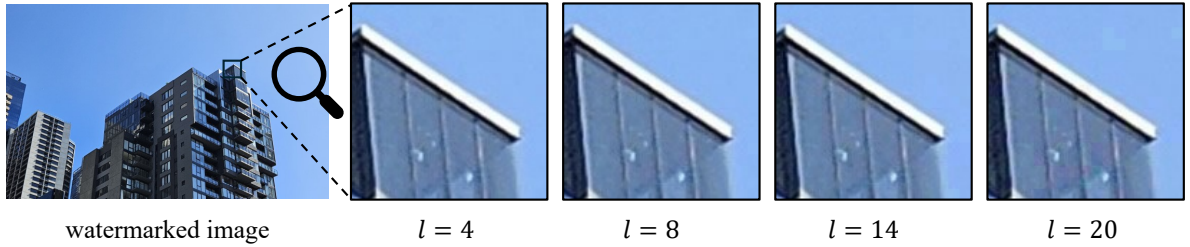


Figure 6. Four watermarks using different interval lengths l . We find that using $l = 8$ provides good image quality in all areas of the image. For larger interval lengths, we find that $l = 14$ is the largest length not introducing visual artefacts in non-monochromatic regions. Excessively high values of l lead to visible perturbations throughout the whole image.

B.5. Adaptive Embedding Based on Block Entropy

The interval length l of our watermark presents a trade-off between visibility and robustness. Higher values of l result in watermarks that are more robust to image manipulations but may become visible in low-entropy regions (see Fig. 6). To preserve this robustness while maintaining imperceptibility, we propose to embed the watermark only in high-entropy areas. For the experiments in Sec. 5.4 we implement the following procedure: Before executing line 5 of Algorithm 1 (and for the detection before line 12 of Algorithm 4), we calculate the entropy $H(b)$ of each block b according to Eq. (1). Subsequently, we determine the entropy threshold $\tau = \text{median}\{H(b_1), \dots, H(b_n)\}$ where n is the total number of blocks in the image. Only blocks with entropy above the threshold ($H(b) > \tau$) are processed for watermark embedding or detection. Note that while we use the median as our threshold, alternative percentile thresholds could be selected by the user depending on the desired embedding capacity and imperceptibility. Excluding low-entropy regions from watermarking is generally reasonable, as these areas typically correspond to monochromatic or background regions with minimal information rather than the actual content we seek to protect. This approach aligns with similar entropy-based filtering strategies employed in previous work [17].

B.6. Crop-Resilient Detection

Since we embed the watermark into blocks throughout the image, we can implement a detection method that identifies watermarks even in cropped images. As visualised in Fig. 7, this detection is based on a brute-force strategy to systematically search for the correct starting point of the embedded blocks.

To detect a watermark in a potentially cropped image, we iterate through the image in $m \times m$ pixel blocks. Since the original starting point of the watermark grid is unknown in a cropped image, we exhaustively test all possible starting positions for the first $m \times m$ block. For the detection process, we propose two different strategies:

1. **Score-based selection:** We collect the detection scores for all possible grid alignments and output the highest identified score.
2. **Early stopping via fixed threshold:** We choose a detection threshold calibrated to our desired FPR and stop the search if this threshold is met.
3. **Early stopping via hypothesis testing:** We evaluate the percentage of watermarked blocks for each iteration, performing a one-sided hypothesis test to determine if the proportion of watermarked blocks exceeds 50% at a specified significance level. If this threshold is met, we conclude that a watermark has been detected and stop

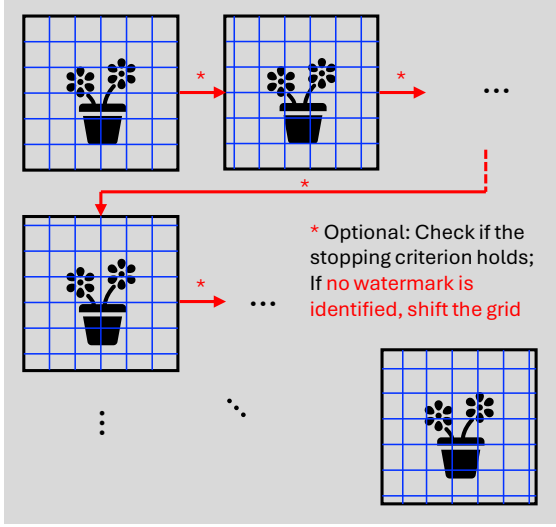


Figure 7. To make our watermark detection method resilient to cropping, we implement a brute-force approach to find the original position of the grid. An early stopping criterion can be used for computational efficiency.

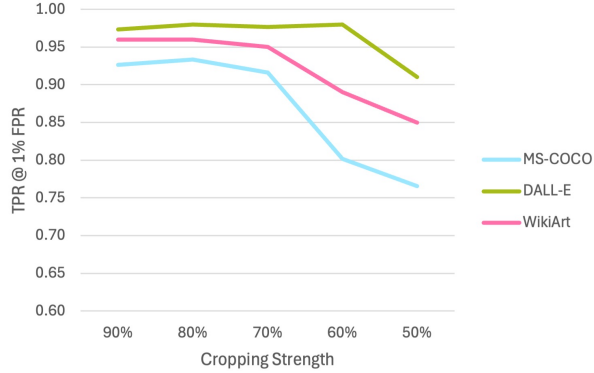


Figure 8. We evaluate the TPR at 1% FPR under varying cropping strengths across datasets. Despite increasing image loss, the watermark remains detectable, with performance remaining strong even when up to 50% of the image is cropped.

the search.

This approach requires a maximum of m^2 iterations to cover all possible starting positions, which is computationally practical for the empirically effective block sizes we consider ($m = 96$).

C. Further Experimental Details

C.1. Embedding Settings

We apply $d = 3$ levels of DWT, and accordingly set the block size to the smallest allowed value $k = 2^d = 8$. We partition the domain of the DWT coefficients into intervals of size l over the range $[-3000, 3000]$ and treat values out-

side of this range as allowed (green). To minimise the visible impact of the introduced perturbations after reversing the transform, we restrict the maximum perturbation to at most $\pm 3l$. If no green intervals are found within this range, the coefficient remains unchanged. This restriction is justified as the probability of all six intervals (left and right for each of the 3 channels) surrounding a coefficient being red is very low, at approximately $\frac{1}{2}^6 \approx 0.016$.

For the detectability, image forensics, and cropping experiments (Sections 5.2, 5.3, and 5.4.1), we set the interval length to $l = 8$. To assess the method’s behaviour under geometric transformations (Sec. 5.4.2), we increase the interval length to $l = 14$ and selectively embed the watermark in high-entropy regions. Specifically, we calculate the median entropy across all blocks and embed the watermark only in blocks with entropy values above this median threshold, thereby excluding regions with very low entropy values.

C.2. Choosing the Detection Threshold

For a direct comparison with the results presented in [43], we follow their methodology: For watermarking techniques that embed bitstrings, detection thresholds are defined to reject the null hypothesis (*i.e.*, the absence of a watermark) at $p < 0.01$. This requires the correct detection of at least 24 out of 32 bits for 32-bit methods, and 61 out of 96 bits for StegaStamp. For score-based watermarking methods, the threshold parameter p is calibrated to achieve reasonable FPRs. Following this approach, our score-based method is calibrated to match similar FPR values as in [43] across different datasets. Through empirical evaluation of various threshold settings (Tab. 4), optimal threshold values were determined: $p = 0.37$ for the COCO dataset, $p = 0.36$ for Diffusion DB, and $p = 0.46$ for WikiArt.

MS-COCO			DiffusionDB			WikiArt		
τ	FPR	WDR	τ	FPR	WDR	τ	FPR	WDR
0.37	0.062	1.000	0.35	0.052	1.000	0.45	0.067	1.000
0.40	0.046	1.000	0.36	0.047	1.000	0.46	0.060	1.000
0.50	0.014	1.000	0.40	0.028	1.000	0.50	0.039	1.000
0.60	0.005	0.998	0.50	0.009	1.000	0.65	0.006	1.000

Table 4. FPR and WDR for MS-COCO, DiffusionDB, and WikiArt at different detection thresholds τ .

D. Additional Figures and Analysis

In this section, we provide more details on our evaluations and additional figures.

D.1. Image Quality of Manipulated Images

We first illustrate two example images to demonstrate how the same set of modifications can have different visual effects depending on the input image. One of the images is

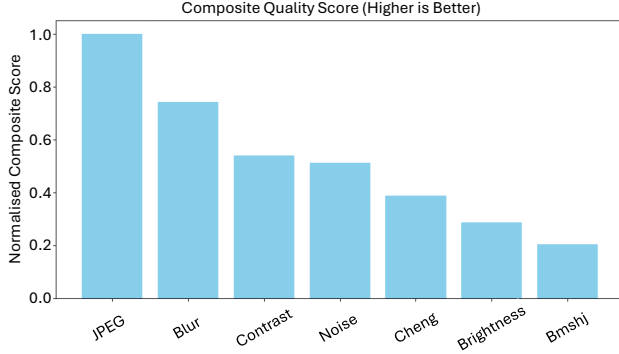


Figure 9. Combined similarity score for each attack, aggregating normalised SSIM, LPIPS, and PSNR scores. Higher scores indicate greater similarity to the original image and therefore less visible quality degradation.

more robust to the changes, while the other reacts more sensitively, as shown in Fig. 10.

Based on this observation, we further quantify the variability in modification effects by evaluating each manipulation using a set of similarity metrics. Fig. 12 presents the results for each similarity metric individually. It is evident that some manipulations, such as JPEG compression, are consistently less perceptible across all metrics, while others (e.g., Bmshj18) cause significant disruptions to image similarity. As each metric captures a different aspect of perceptual similarity, we cannot observe a consistent ranking of manipulations based on image quality degradation.

To provide a more comprehensive overview, we normalise all metric scores to the range $[0, 1]$ and combine them into a unified scale, shown in Figure 9. As the transformations Cheng20, Bmshj18 and Brightness failed to achieve 50% of the average quality metric, we deemed these to be insufficient to achieve the attacker’s goals and so did not consider them further.

D.2. Distribution of Image Sizes

To ensure diversity in our evaluation, we include datasets with varying image sizes. WikiArt images are the largest, with a median size over $9\times$ larger than MS-COCO and nearly $2.5\times$ larger than DALL-E. DALL-E and DiffusionDB represent mid-sized images, with DALL-E being notably consistent in resolution, while DiffusionDB shows broader variability. An overview of the pixel count distributions is shown in Figure 11.

When it comes to image forensics and analysing the detection map, our watermarking method performs best when applied to large images. In these cases, it becomes easier to analyse the patterns of red squares in the localised detection map. The larger scale helps us distinguish between actual modifications and random false positives, which can occur in small numbers even in unaltered images.

D.3. Additional Robustness Evaluations

In this section, we provide additional evaluation of our watermarking method’s robustness against various image manipulations. To complement the cropping analysis presented in Sec. 5.4.1, we include Fig. 8 for completeness, which visualises the robustness of our method to different levels of cropping. In addition to the results presented in Sec. 5.4.2, we assess the detection performance of our method under various image modifications:

- JPEG compression with quality factors 30, 50, 70 and 90.
- Gaussian blur with standard deviation 1 and kernel sizes of 3, 5 and 7.
- Adding Gaussian noise with standard deviation $\sigma \in \{0.02, 0.04, 0.05, 0.06, 0.08\}$.
- Contrast and brightness adjustment with factors $\in \{0.5, 0.8, 1.0, 1.2, 1.6, 2.0\}$.

We present ROC curves for each type of distortion in Fig. 13. We can observe that our watermarking scheme exhibits high robustness against JPEG compression. Detection performance is nearly perfect for quality factors 90, 70, and 50, with AUC values close to 1.00. Even under strong compression with a quality factor of 30, the method maintains good detectability, achieving an AUC of 0.83.

Similarly, the method shows excellent robustness to Gaussian blurring, as depicted in Fig. 13b. For a kernel size of 3, the AUC is 1.00, indicating perfect detection. Increasing the kernel size to results in only a marginal decrease in performance (AUC of 0.99).

The performance under additive Gaussian noise is evaluated in Fig. 13c. We can observe that the watermark detection remains high as the noise strength increases further to $\sigma = 0.06$ (AUC = 0.91) and $\sigma = 0.08$ (AUC = 0.81).

Contrast and brightness adjustments, however, pose a greater challenge (see Fig. 13d and Fig. 13e). The detection performance is weaker across all tested factors. These image modifications directly modify the pixel intensity values and the overall dynamic range of the image. Given that the DWT coefficients are computed from these pixel values, substantial alterations to the input data necessarily lead to significant changes in the calculated coefficient magnitudes and distributions.

Overall, this additional evaluation indicates that our watermarking method is highly robust to common distortions like JPEG compression, Gaussian blur, and additive Gaussian noise. While detection performance can be impacted by strong contrast and brightness adjustments, it is noteworthy that such significant modifications typically result in substantial and perceptible visual degradation of the image content. Consequently, the practical relevance of this sensitivity might be reduced in scenarios where maintaining the visual authenticity and quality of the content is important.

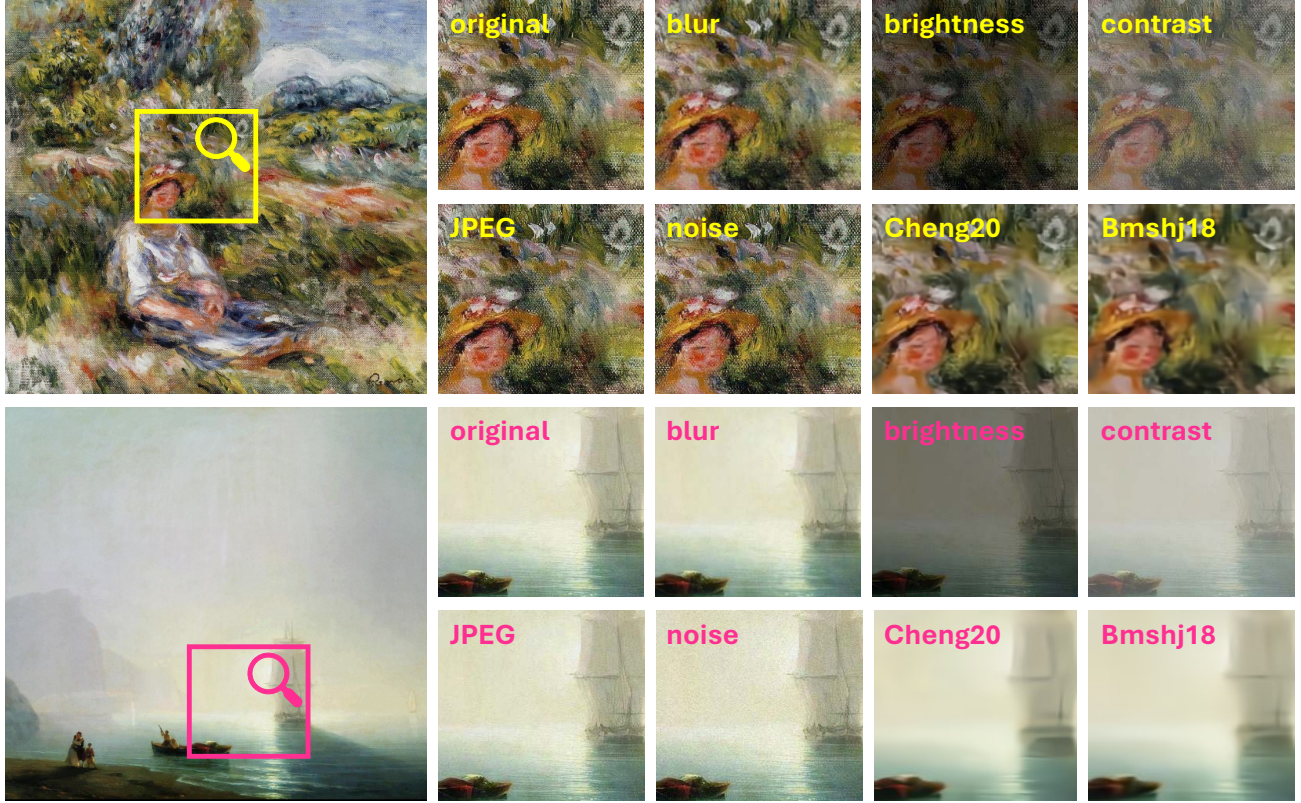


Figure 10. The effectiveness and visibility of attacks can vary across different images. We illustrate two examples - one highly resilient to visual changes, and one more sensitive. Each image is subjected to the full set of modifications described in Sec. 5.1.3. This figure compares the impact of different attacks on these images. The upper image contains rich visual detail and texture, which benefits the malicious actor by making perturbations less noticeable to the human eye. In contrast, the lower image is relatively smooth and monochromatic, offering fewer opportunities to conceal changes. As a result, even small amounts of added noise become clearly visible in the lower image, whereas they remain imperceptible in the upper one. Nonetheless, for both images, we observe that certain transformations, such as brightness, contrast, and the attacks Cheng20 and Bmshj18, significantly degrade visual quality.

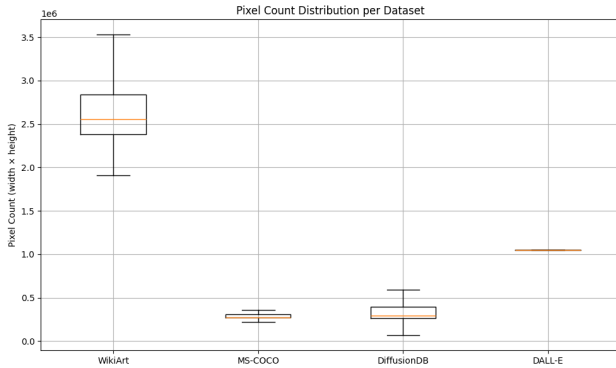


Figure 11. Pixel count distribution across all evaluation datasets used in our experiments.

D.3.1. Attacking the Seed

To evaluate the robustness of the detection process of our watermarking scheme, we design a straightforward attack targeting the sensitivity of the block seed (get_seed function in Algorithms 1 and 4, see Algorithm 3 for a definition). The attack aims to modify the mean colour of each block, which is crucial for watermark detection. Specifically, we explore three variations of this attack: 1) Adjusting the mean colour to the **nearest value** that would round to a different integer when using our rounding-to-30 scheme. 2) Increasing the mean colour to the **next higher value** that would result in a different rounded integer. 3) Decreasing the mean colour to the **next lower value** that would yield a different rounded integer.

In Fig. 14 we demonstrate example outputs from this attack, indicating that all three variations of this attack necessitated substantial alterations to the mean colour values. These changes resulted in visual artefacts, that significantly

compromise image quality. These results demonstrate the difficulty of watermark removal by manipulating the block seeds.

E. Societal Impact

Watermarking improves the traceability and authenticity of digital content, including AI-generated media. This capability offers important societal benefits, such as protecting intellectual property and helping detect misinformation or manipulated content. This traceability can also have unintended consequences, including privacy concerns and potential misuse, such as unauthorised watermarking of unwatermarked content to falsely claim ownership. Our approach mitigates forgery risks through secret-key-based pseudorandom embedding (see Remark 1), and the localised detection map provides transparency to aid forensic verification while recognising limitations such as possible false detection errors (Sec. 4.2.1). Overall, watermarking plays a crucial role in supporting digital content authenticity and traceability, making it a valuable tool. We believe the societal benefits of watermarking in maintaining trust and authenticity outweigh potential risks.

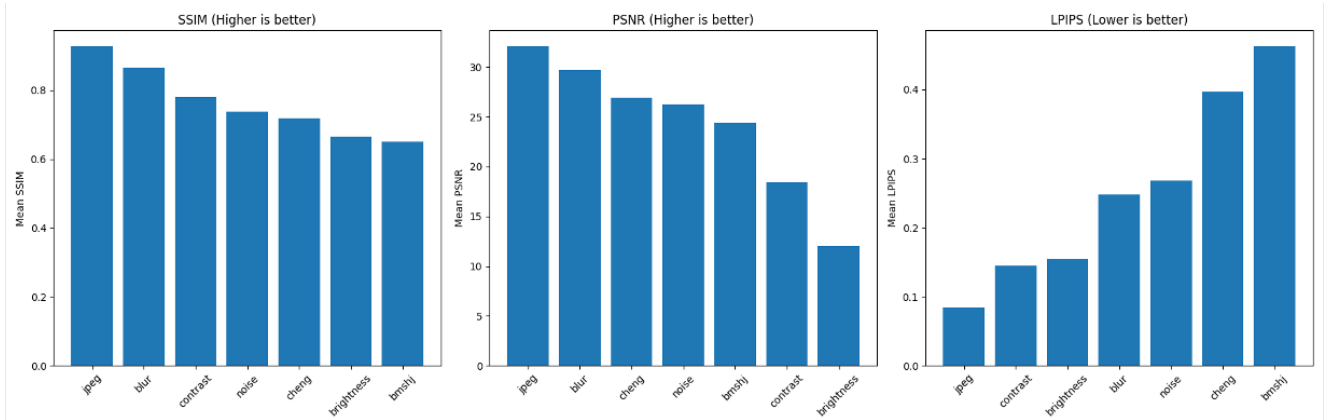


Figure 12. Similarity scores (SSIM, LPIPS, PSNR) for each attack, averaged over all datasets. Since these metrics capture different aspects of image similarity, we obtain varying rankings across attacks. Still, we can observe clear patterns, such as JPEG being consistently ranked first, meaning that it is causing the least degradation, while Bmjsh18 ranks among the most damaging.

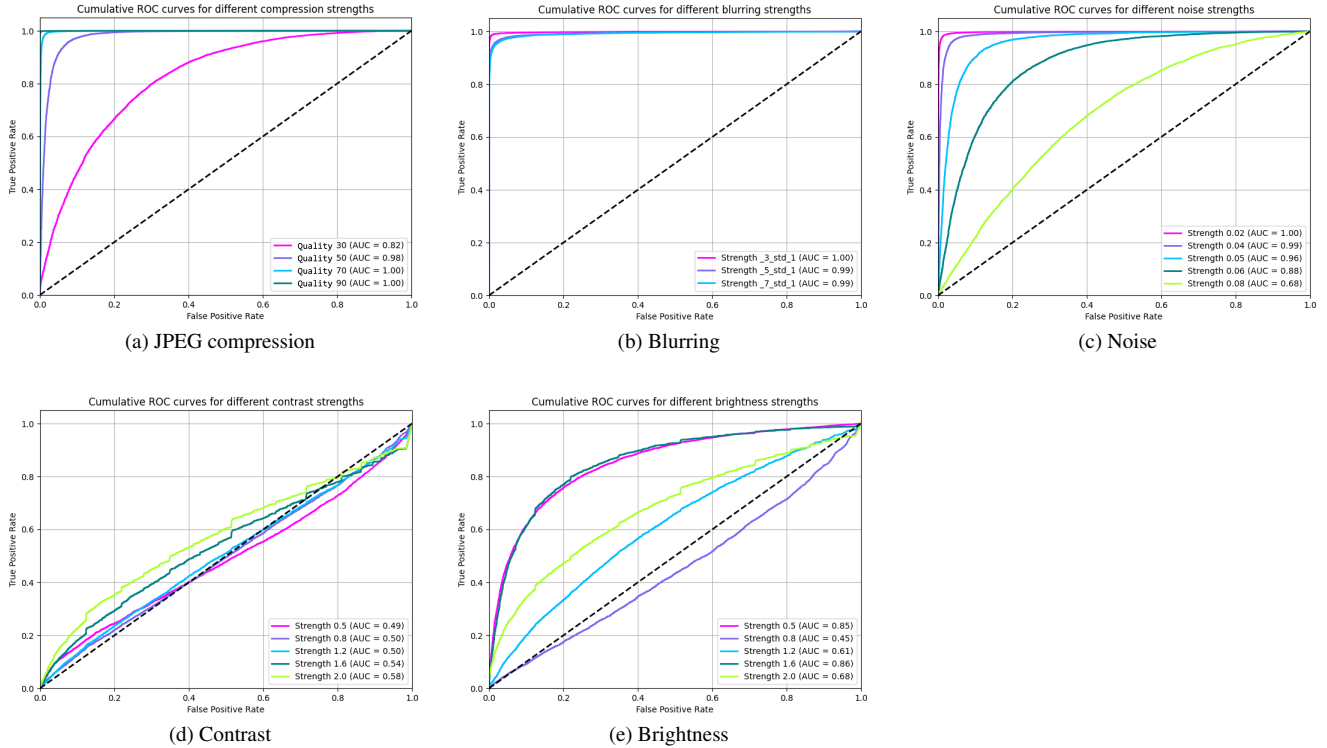
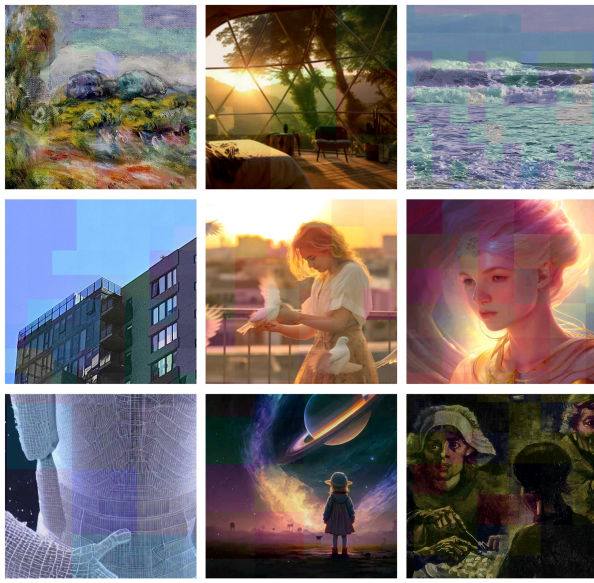


Figure 13. Cumulative ROC curves for MS-COCO, DiffusionDB and WikiArt.



(a) Attacked images



(b) Original images

Figure 14. Evaluation of seed robustness against mean colour alteration attacks (Sec. D.3.1). Fig. 14b shows the watermarked image, where the partitioning seed during embedding relies on the block mean colour rounded with $r = 30$. Fig. 14a shows the watermarked image after the attack aimed at modifying block mean colours to disrupt seed reconstruction. The visible degradation in Fig. 14a demonstrates that altering the seed sufficiently to hinder watermark detection requires introducing visually significant artifacts.