# Watermark Anything with Localized Messages

**Tom Sander**[1,2,*], **Pierre Fernandez**[1,3,*], **Alain Durmus**[2], **Teddy Furon**[3], **Matthijs Douze**[1]

[1]Meta FAIR, [2]École polytechnique, [3]Inria Rennes
[*]Equal contribution

Image watermarking methods are not tailored to handle small watermarked areas. This restricts applications in real-world scenarios where parts of the image may come from different sources or have been edited. We introduce a deep-learning model for localized image watermarking, dubbed the Watermark Anything Model (WAM). The WAM embedder imperceptibly modifies the input image, while the extractor segments the received image into watermarked and non-watermarked areas and recovers one or several hidden messages from the areas found to be watermarked. The models are jointly trained at low resolution and without perceptual constraints, then post-trained for imperceptibility and multiple watermarks. Experiments show that WAM is competitive with state-of-the art methods in terms of imperceptibility and robustness, especially against inpainting and splicing, even on high-resolution images. Moreover, it offers new capabilities: WAM can locate watermarked areas in spliced images and extract distinct 32-bit messages with less than 1 bit error from multiple small regions – no larger than 10% of the image surface – even for small $256 \times 256$ images.

**Correspondence:** {tomsander, pfz}@meta.com
**Code and models:** https://github.com/facebookresearch/watermark-anything

## 1 Introduction

Invisible image watermarking embeds information into image pixels in a way that is imperceptible to the human eye and yet robust. It was initially developed for intellectual property and copy protection, such as by Hollywood studios for DVDs. However, the applications of watermarking are evolving, particularly in light of the recent development of generative AI models (Kušen and Strembeck, 2018). Regulatory acts such as the White House executive order (USA, 2023), the Californian bill, the EU AI Act (Parliament and Council, 2024), and Chinese AI governance rules (of the People's Republic of China, 2023) require AI-generated content to be easily identifiable. They all cite watermarking as either compulsory or a recommended measure to detect and label AI-generated images.

Image splicing is one of the most common manipulations, whether applied for benign or malicious purposes (Christlein et al., 2012; Tralic et al., 2013). Splicing involves adding text or memes on a large portion of the image or extracting parts of images and overlaying them on others (Douze et al., 2021). It can bypass the state-of-the-art watermarking techniques, which take one global decision per image under scrutiny. Indeed, in traditional watermarking, the watermark signal fades away and is no longer detected as the surface of the watermarked area decreases. Besides, these techniques poorly answer the paradoxical question of deciding whether an image should be considered watermarked if only a small part carries the watermark. A positive decision triggered by a small area might be unfair to artists who use AI models for inpainting or outpainting. On the other hand, not being robust enough to splicing opens the door to easy removal.

To address these issues, this paper redefines watermarking as a segmentation task, giving birth to the Watermark Anything Models (WAM). Our motivation is to disentangle the strength of the watermark signal from its pixel surface, in contrast to traditional watermarking. More precisely, the WAM extractor detects if the watermark is present and extracts a binary string *for every pixel* rather than predicting a message for the whole image. These outputs are post-processed according to the final task. For global detection, the image is deemed watermarked if the proportion of watermarked pixels exceeds a user-defined threshold. For global decoding, a majority vote recovers the hidden message. A new application, out of the reach of traditional robust watermarking, is the localization of watermarked areas and the extraction of multiple hidden messages. For that purpose, we choose to apply the DBSCAN

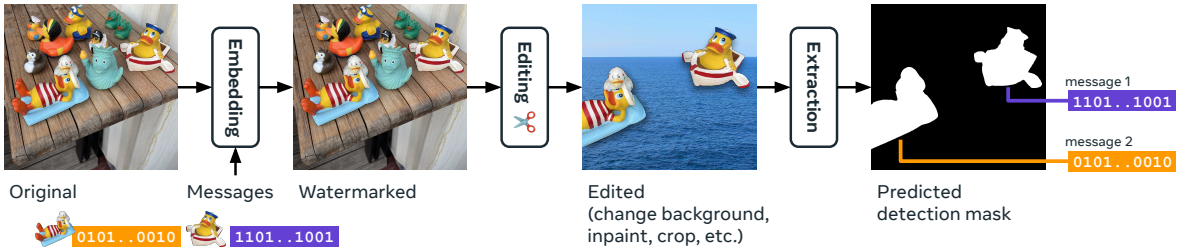



**Figure 1** Overview. (a) The embedder creates an imperceptible image modification. (b) Traditional transformations (cropping, JPEG compression, etc.) and/or advanced manipulations (mixing watermarked and non-watermarked images, inpainting, etc.) may be applied to the image. (c) The extraction creates a segmentation map of watermarked parts and retrieves one or several messages.

clustering algorithm over the pixel-level binary strings because it does not require any prior on the number of watermarks (or centroids). This is detailed in Sec. 3.

These new functionalities require a training with new objectives, that is split into two phases. The first phase pre-trains the embedder and extractor models for low-resolution images. It essentially targets the robustness criterion. The *embedder* encodes a $n_{\text{bits}}$-bit message into a watermark signal that is added to the original image. The *augmenter* randomly masks the watermark in parts of the image and augments the result with common processing techniques (e.g., cropping, resizing, compression). The *extractor* then outputs a $(1 + n_{\text{bits}})$-dimensional vector per pixel to predict the parts of the image that are watermarked and decode the corresponding messages. Detection and decoding losses are used as training objectives. The second training phase targets the following new objectives: (1) minimize the watermark's visibility in alignment with the human visual system, (2) allow for multiple messages within the same image. This two-stage training is less prone to instability, compared to previous use of adversarial networks and divergent objectives (Zhu et al., 2018). It also trains the extractor on both watermarked and non-watermarked images, for the first time in the literature. This increases the performance and the robustness of the detection.

We first compare WAM with state-of-the-art methods for regular tasks of watermark detection and decoding on low and high-resolution images. Our results show that WAM achieves competitive performance in terms of imperceptibility and robustness. To further highlight the advantages of WAM, we then evaluate its performance on tasks that are not considered in the literature. Namely, we evaluate the localization accuracy between the predicted watermarked areas and the original mask and assess the ability to detect and decode multiple watermarks in a single image. For instance, when hiding five 32-bit messages, each in a 10% area of the image, detection of watermarked areas achieves more than 85% mIoU, even after images are horizontally flipped and the contrast adjusted, and bit accuracy (for a total of 160 bits) achieves more than 95% under the same augmentation (Sec. 5.5).

In summary, our contributions are:

- the definition of watermarking as a segmentation task;
- a two-stage training able to strike a good trade-off between invisibility and robustness even for multiple watermarks and high-resolution images;
- WAM, an embedder/extractor model competitive with state-of-the-art methods;
- the highlight of new capabilities, localization of watermarks and extraction of multiple messages as depicted in Fig. 1, together with specially designed evaluations.

## 2 Related Work

**Semantic segmentation** aims to predict a category label for every pixel in an image. FCN (Long et al., 2015) employs a convolutional network predicting pixel-wise classification logits. More recent works (Zheng et al., 2021; Strudel et al., 2021) aggregate hierarchical context and are based on a ViT encoder (Dosovitskiy, 2020), followed by a decoder that generates the pixel-level predictions – similar

to our extractor's architecture. On the other hand, instance segmentation identifies individual objects within an image (He et al., 2017; Carion et al., 2020). Recent literature, such as MaskFormer (Cheng et al., 2021) and Mask2former (Cheng et al., 2022), unifies semantic and instance segmentation. Segment Anything (Kirillov et al., 2023) and follow-ups (Zhang et al., 2023; Zhao et al., 2023a; Ke et al., 2024; Ma et al., 2024; Ravi et al., 2024) go one step further. They now allow users to segment any object in images or videos using prompts, which are points, bounding boxes, or natural language.

**Robust watermarking** was first designed for copyright protection (Cox et al., 2007). Traditional methods operate either in the spatial domain (Van Schyndel et al., 1994; Nikolaidis and Pitas, 1998) or in the frequency domain modifying components through an invertible transform (e.g., Fourier, Wavelet) (Cox et al., 1997; Urvoy et al., 2014). These methods were progressively replaced by deep-learning ones which remove the expert need to crafting the transforms such that the embedding is imperceptible and robust. The first approach (Vukotić et al., 2018, 2020; Fernandez et al., 2022; Kishore et al., 2022; Chen et al., 2022) embeds the watermark into the representations of an off-the-shelf pre-trained model. Our method is inspired by the second approach, which jointly trains deep learning-based architectures to embed and extract watermarks while being robust to augmentations seen during training, such as HiDDeN (Zhu et al., 2018) and followers (Zhang et al., 2019b; Luo et al., 2020; Tancik et al., 2020; Ma et al., 2022; Bui et al., 2023b,a; Pan et al., 2024).

The two tasks of watermarking, detection and decoding, are rarely performed together. Detection of the watermark, a.k.a., zero-bit watermarking, distinguishes watermarked content from original images. Decoding of a hidden message, a.k.a., multi-bit watermarking, implicitly assumes that the content under scrutiny is watermarked. One possibility for combining the two tasks is to reserve $n_{\text{det}}$ bits of the message as a detection segment – that should match a fixed pattern – with the remaining bits carrying the actual payload. Assuming that the message decoded from a non-watermarked image is random, the False Positive Rate equals $2^{-n_{\text{det}}}$. However, a recent study (Fernandez et al., 2023a, App. B.5) shows that the decoded bits are neither equiprobable nor independent. In other words, it is difficult to decode a hidden message while being confident in detecting a watermark.

Very few papers in the literature deal with watermarking objects in picture (Bas et al., 2001; Barni et al., 2005). This trend was abandoned together with the object-oriented MPEG-4 video codec.

**Active tamper localization** relies on *semi-fragile* watermarking. Areas where the watermark is not recovered are deemed tampered. This idea appears early in the literature (Kundur and Hatzinakos, 1999; Lin et al., 2000), although deep-learning methods offer significant improvements (Asnani et al., 2022). There is a trade-off between the granularity of the localization of the tampered areas and the semi-fragility of the watermark. The biggest difficulty is to design a watermark robust to benign transformations (e.g., image compression) but fragile to malicious editing.

Our method combines detection and decoding together with a precise segmentation of the watermarked areas. This natively answers applications ranging from copyright protection, detection of AI-generated objects in images and tampering localization. EditGuard (Zhang et al., 2024) sequentially embeds a fragile watermark for localization and a robust watermark for copyright protection. Although the approach shares similarities with ours, it is not robust to geometric augmentations such as cropping or perspective changes. In contrast, our approach is conceptually simpler since a single embedding is used for detection and decoding. It offers better robustness, and enables the extraction of multiple watermarks. The approach most similar to ours is AudioSeal (San Roman et al., 2024), which introduces localized watermarking in the audio domain, but does not handle multiple messages.

An extended related work with more details on the methods mentioned above is presented in App. B.

## 3 Detection, Localization, and Message Extraction

Before introducing our method and its training, this section presents several applications, i.e., how to use WAM's extractor outputs for watermark localization, zero-bit detection, and decoding of multiple messages within the same image.

The key feature is the extractor $\text{ext}_{\theta^*}$ ($\theta^*$ refers to the weights of the model after training) which outputs a tensor $y = \text{ext}_{\theta^*}(x)$ of dimensions $(1+n_{\text{bits}}, h, w)$ for an input image $x \in \mathbb{R}^{3\times h\times w}$. This tensor consists of a watermark detection mask $y^{\text{det}} \in [0,1]^{1\times h\times w}$ and a decoding mask $y^{\text{dec}} \in [0,1]^{n_{\text{bits}}\times h\times w}$. We denote by $y_i^{\text{det}} \in [0,1]$ the predicted detection score for pixel $i$, and by $y_i^{\text{dec}} = [y_{i,1}^{\text{dec}} \dots y_{i,n_{\text{bits}}}^{\text{dec}}] \in [0,1]^{n_{\text{bits}}}$ its decoded message.

**Localization** (or pixel-level detection) spots watermarked pixels of the image. A pixel is deemed watermarked if its detection score $y_i^{\text{det}}$ exceeds a threshold $\tau$. Typically, we set $\tau$ empirically, by measuring the False Positive Rate (FPR) on all the pixels of a held-out training set (the FPR is the probability of a non-watermarked pixel being flagged as such).

**Detection** (or image-level detection) decides if an image is globally watermarked. From the pixel-level detection score, an image soft detection score can naturally be computed as:

$$s_{\text{det}} := \frac{1}{h\times w}\sum_{i=1}^{h\times w} \mathbb{1}\{y_i^{\text{det}} > \tau\} \in [0,1]. \tag{1}$$

The image is flagged if $s_{\text{det}}$ is higher than a threshold that is the proportion of watermarked pixels that the user considers enough to deem a content as watermarked.

**Decoding** of a single message within an image is done by the following weighted average of the pixel-level soft predictions of the hidden message:

$$\hat{m}_k = \begin{cases} 1 & \text{if } \left[\frac{1}{\sum_i \mathbb{1}\{y_i^{\text{det}}>\tau\}}\left(\sum_{i=1}^{h\times w} \mathbb{1}\{y_i^{\text{det}} > \tau\}\cdot y_{i,k}^{\text{dec}}\right)\right] > 0.5\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{2}$$

Decoding of multiple watermarks within one image uses a hard detection approach instead of the previous soft weighting. We isolate pixels detected as watermarked, i.e., pixels $i$ with $y_i^{\text{det}} > \tau$, and we compute the local decoded message $\tilde{m}_i$ such that for any $k \in \{1,\dots,n_{\text{bits}}\}$, $\tilde{m}_{i,k} = \mathbb{1}\{y_{i,k}^{\text{dec}} > 0.5\}$. The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm (Ester et al., 1996; Schubert et al., 2017) clusters the set of locally decoded messages. It outputs some centroids and an assignment for every pixel detected as watermarked. DBSCAN selects these centroids among the initial set, thus ensuring that the final decoded messages are binary words.

DBSCAN offers the advantage of not requiring a pre-defined number of clusters. Instead, we need to specify two parameters: (1) $\min_{\text{samples}}$, the minimum number of points for a cluster to be considered valid, and (2) $\varepsilon$, the maximum distance between two samples for one to be considered as in the neighborhood of the other. For further details on the DBSCAN algorithm, see App. C.1.

Handling multiple messages in a single image makes WAM robust against attacks that involve splicing several watermarked images together, which can compromise the decoding of traditional watermarking schemes in real-world scenarios. It also enables active object detection (Asnani et al., 2024), where the objective is to track watermarked objects. Additionally, it allows for the identification of the use of multiple watermarked AI tools.

# 4 Watermark Anything Models

## 4.1 The model

WAM considers two joint models: a watermark embedder $\text{emb}_\theta$ and a watermark extractor $\text{ext}_\theta$, where $\theta$ gathers the parameters of these functions. The embedder defines the watermark procedure for hiding a message into an image, while the extractor detects watermarking and decodes messages. This section discusses our choices for the different pieces that constitute WAM. App. D.1 gives the technical details of the network architectures.

**The watermark embedder** ($\text{emb}_\theta$) consists of an encoder, a binary message lookup table, and a decoder. The encoder represents an image in a latent space. The lookup table is used to translate the message into a tensor which is concatenated to the image representation. From that, the decoder outputs a watermark signal, which is added to the original image with adequate scaling.

The autoencoder that we consider is based on the architecture of the variational autoencoder of LDM (Rombach et al., 2022). Its encoder, $\text{enc}_\theta$, compresses a $h \times w$ input image $x$ into a latent $z \in \mathbb{R}^{d_z \times h' \times w'}$ (with downsampling factor $f = h/h' = w/w'$). The binary message lookup table $\mathcal{T}_\theta$ is of shape $(n_{\text{bits}}, 2, d_{\text{msg}})$. Each bit of the message $m$ is mapped to the embedding $\mathcal{T}_\theta(k, m_k, \cdot) \in \mathbb{R}^{d_{\text{msg}}}$, depending on its position $k \in \{1, \dots, n_{\text{bits}}\}$ and its value $m_k \in \{0, 1\}$. The $n_{\text{bits}}$ embeddings are averaged, resulting in a vector of size $d_{\text{msg}}$ which is repeated to form a tensor of shape $(d_{\text{msg}}, h', w')$. The concatenation to the image representation yields an activation of shape $(d_z + d_{\text{msg}}) \times h' \times w'$. The decoder, $\text{dec}_\theta$, maps back this activation to the watermark signal $\delta_\theta(x, m)$ of the same shape as the original image. The range of this signal is $[-1, 1]$ because the last layer of $\text{dec}_\theta$ is a hyperbolic tangent activation. It is finally added to the original image to produce the watermarked output $x_{\text{m}} = \text{emb}_\theta(x) = x + \alpha \cdot \delta_\theta(x, m)$. Parameter $\alpha \in \mathbb{R}_+$ is called the watermark strength as it controls the distortion applied to the original image.

**The watermark extractor** ($\text{ext}_\theta$) takes on the dual role of detecting watermarked pixels and decoding their embedded messages. We use an architecture similar to SETR (Zheng et al., 2021) and Segment Anything (Kirillov et al., 2023). It comprises a ViT encoder paired with a pixel decoder that upsamples the embeddings to the original image size. The watermark extractor outputs a vector of size $1 + n_{\text{bits}}$ for every pixel, as described in Sec. 3.

**High-resolution.** WAM operates at a fixed resolution of $h \times w$. To extend it to higher resolutions, an anisotropic scaling resizes the image to $h \times w$ and $\text{emb}_\theta$ computes the watermark signal $\delta$ from this resized image. A bilinear interpolation scales $\delta$ back to the size of the original image. The extraction also operates at $h \times w$, by resizing all images before feeding them to the network. Therefore the extraction process remains consistent with the pre-training conditions. We only use these interpolations at embedding time. WAM is thus only trained on low resolution images (Sec. 4.2) but can be used for high-resolution images too, which represents an important training compute gain. A similar approach also appears in the recent literature (Bui et al., 2023a).

## 4.2 Pre-training models for localized message embedding and extraction

The objective of this first training phase is to obtain a robust and localizable watermark hiding a $n_{\text{bits}}$-bit message inside parts of an image, without caring much about imperceptibility. This stage does not incorporate any perceptual loss: our goal is to achieve perfect localization and decoding even after severe augmentations. Figure 2 illustrates the detailed process.

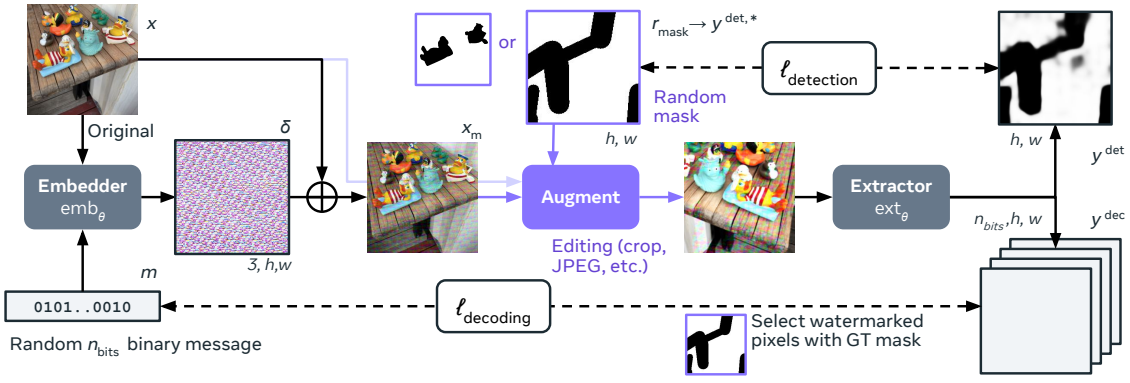



**Figure 2** The first training phase of WAM, as described in Sec. 4.2, jointly trains the *watermark embedder* to predict an additive watermark and the *watermark extractor* to detect watermarked pixels and decode the hidden message. In between, the augmenter 1) splices the watermarked and the original images based on a random mask and 2) applies classical image transformations.

**Augmentations** are two-step processes. They first splice images $x$ and $x_{\mathrm{m}}$ based on a binary mask and then apply usual image transformations to improve robustness.

The first step randomly samples a mask among full masks, rectangles, irregular shapes, or object segmentation maps (possibly provided by the training dataset). These masks are inverted with probability 0.5. It yields a mask $r_{\mathrm{mask}} \in [0,1]^{h \times w}$. Figure 2 shows examples of masks, with more samples displayed in Figure 6 of App. D.2. The first step then computes the spliced image as $x_{\mathrm{masked}} = r_{\mathrm{mask}} \odot x_{\mathrm{m}} + (1 - r_{\mathrm{mask}}) \odot x$, where $\odot$ denotes component-wise multiplication.

The second step of the augmentation applies a processing typical from real-world image editing including geometric transformations (identity, resize, crop, rotate, horizontal flip, perspective) and valuemetric adjustments (JPEG, Gaussian blur, median filter, brightness, contrast, saturation, hue). The geometric transformation alters the positions of watermarked pixels, so it is also applied to the mask to keep track of the watermarked areas. This adjusted mask is the ground truth for the localization and it is denoted $y^{\mathrm{det},\star}$ in the sequel. These augmentations are relatively strong to reflect the transformations found on media sharing websites, as opposed to professional photo publications. Additional details and discussions on augmentations are postponed to App. D.4.

When a given image goes through the embedder-augmentation-extractor sequence, the final output depends on the models' parameters $\theta$ (omitted so far for clarity):

$$y(\theta) = [y^{\mathrm{det}}(\theta), y^{\mathrm{dec}}(\theta)] = \mathrm{ext}_\theta \left( \text{ transformation } \left( \text{ masking } \left( \text{ emb}_\theta(x, m), \, x \right) \right) \right) . \tag{3}$$

**Objectives.** The training minimizes the objective function $\ell(\theta)$ which is a linear combination of the detection and decoding losses: $\ell(\theta) = \lambda_{\mathrm{det}} \cdot \ell_{\mathrm{det}}(\theta) + \lambda_{\mathrm{dec}} \cdot \ell_{\mathrm{dec}}(\theta)$.

The detection loss is the average of the pixel-wise cross-entropy between $y^{\mathrm{det}}(\theta) \in [0,1]^{h \times w}$ and the ground truth $y^{\mathrm{det},\star} \in \{0,1\}^{h \times w}$ (pixel watermarked or not). Similarly, the decoding loss is the average of the pixel-wise and bit-wise binary cross-entropy between $y^{\mathrm{dec}}_{i,k}(\theta)$ and $m_k$ only over the watermarked pixels, where $m$ is a random message originally embedded in that image. For a given image and message, $\ell_{\mathrm{det}}$ and $\ell_{\mathrm{dec}}$ are:

$$\ell_{\mathrm{det}}(\theta) = \frac{-1}{h \times w} \sum_{i=1}^{h \times w} \left[ y_i^{\mathrm{det},\star} \log(y_i^{\mathrm{det}}(\theta)) + (1 - y_i^{\mathrm{det},\star}) \log(1 - y_i^{\mathrm{det}}(\theta)) \right] , \tag{4}$$

$$\ell_{\mathrm{dec}}(\theta) = \frac{-1}{n_{\mathrm{bits}} \times \sum_{i=1}^{h \times w} y_i^{\mathrm{det},\star}} \sum_{i=1}^{h \times w} y_i^{\mathrm{det},\star} \sum_{k=1}^{n_{\mathrm{bits}}} \left[ m_k \log(y^{\mathrm{dec}}_{i,k}(\theta)) + (1 - m_k) \log(1 - y^{\mathrm{dec}}_{i,k}(\theta)) \right] . \tag{5}$$

### 4.3 Post-training for imperceptibility and multiple watermarks

The model trained in the first phase (Sec. 4.2) produces a too visible watermark. Moreover, it cannot deal within multiple watermarks in an image. The second training phase addresses these issues.


**(a)** Without JND **(b)** With JND

**Figure 3** Impact of the JND map on imperceptibility. (Left) After the first training phase, the watermark is highly perceptible. (Right) When applying the JND attenuation, it is hidden in areas where the eye is not sensitive to changes, which makes it less visible. Fine-tuning with the JND recovers the initial robustness. The difference is displayed as $10 \times \mathrm{abs}(x_{\mathrm{m}} - x)$.

**Perceptual heatmap.** The Just-Noticeable-Difference (JND) map is a hand-crafted model of the minimum artifact perceivable by a human at every pixel. For instance, artifacts on flat, uniform areas are more visible than on textured ones. The JND map was introduced by Chou and Li (1995) and later used by Wu et al. (2017). App. C.2 gives more details on its computation.

Given an image $x$, we have $\text{JND}(x) \in \mathbb{R}^{3\times h\times w}$. We modulate the intensity of the watermark per pixel at embedding time: the final watermarked image is computed as $x_\text{m} = x + \alpha_\text{JND} \cdot \text{JND}(x) \odot \delta_\theta(x, m)$. In practice, we found that applying this JND on the model trained in the first phase slightly degrades the detection and decoding performance. The fine-tuning cancels this degradation. Applying JND only in the second training phase makes training easier than previous methods relying on "perceptual" or "contradictory" losses (like StegaStamp (Tancik et al., 2020) or HiDDeN (Zhu et al., 2018)).

**Multiple watermarks.** Since the first training phase (Sec. 4.2) uses a single message, it tends to decode a constant message across pixels even on image regions with different messages. The second training phase introduces several masks to address this limitation. The masks are generated as before with distinct random messages in each of the masked areas. The number of disjoint masks goes from 1 to 3 with probabilities 0.6, 0.2, and 0.2 respectively (see App. D.2).

The detection loss $\ell_\text{det}$ takes the union of all masks as ground truth $y_i^{\text{det},\star}$. The decoding loss $\ell_\text{dec}$ is computed separately for each message and the losses are summed up. The scenario of multiple watermarks within a single image arises when watermarked images are combined. Unlike other methods producing a global message, WAM can now distinguish and decode each message separately.

# 5 Experiments & Results

## 5.1 Implementation details

We provide here further implementation details using the notations introduced in Sec. 4.1, and the rest of the architectures are described in App. D.1. All experiments are run with $n_\text{bits} = 32$. There is a total of 1.1M parameters for the embedder and 96M for the extractor (equivalent to a ViT-base). In our experiments, we found it necessary to increase the size of the extractor compared to the embedder. This is probably because the extractor has more tasks to handle, such as segmenting and extracting multiple messages, while the embedder only needs to embed one message into an image. Also, it is important for the embedding process to be quick since it happens on the user-side, so we keep it voluntarily small to be fast and usable at scale. We train our model on the MS-COCO training set with blurred faces (Lin et al., 2014), that contains 118,000 images, many of them with segmentation masks. We train at resolution $h \times w = 256 \times 256$, and with $f = 8$ (so $h' \times w' = 32 \times 32$). The first training phase (Sec. 4.2) is optimized with AdamW (Kingma, 2014; Loshchilov, 2017) with a linear warmup of the learning rate in 5 epochs from $1 \times 10^{-6}$ to $1 \times 10^{-4}$ and a cosine annealing to $1 \times 10^{-6}$. We set $\lambda_\text{dec} = 10$, $\lambda_\text{det} = 1$, $\alpha = 0.3$. We train with a batch size of 16 per GPU for 300 epochs using 8 V100 GPUs which takes roughly 2 days. The second training phase (Sec. 4.3) further trains the model with the JND attenuation for 200 epochs, hiding up to 3 messages per image using either randomly sampled rectangles or segmentation masks. During this phase, $\alpha_\text{JND} = 2$. App. D.2 provides details on the generation of masks used during both training phases. For the extraction of multiple messages, we use the Scikit-learn DBSCAN implementation (Pedregosa et al., 2011).

## 5.2 Quality

Table 1 evaluates quantitatively the difference between the watermarked and original images with PSNR, SSIM and LPIPS (Zhang et al., 2018) metrics. We adapt the methods so that they are comparable in terms of imperceptibility. We show qualitative examples for COCO and DIV2k images in App. F. The perceptual quality is good: the JND map successfully modulates the watermark signal in areas where the eye is not sensitive. Nevertheless, repetitive and regular patterns can be observed in some images when the bright or textured areas are big enough (furs of animals for instance).

**Table 1** Evaluation of the watermark imperceptibility. We report the PSNR, SSIM, and LPIPS between watermarked and original images of COCO (low/mid-resolution) and DIV2k (high-resolution).

| | | HiDDeN | DCTDWT | SSL | FNNS | TrustMark | WAM (ours) |
|---|---|---|---|---|---|---|---|
| COCO | PSNR (↑) | 38.2 | 37.0 | 37.8 | 37.7 | 40.3 | 38.3 |
| | SSIM (↑) | 0.98 | 0.98 | 0.98 | 0.98 | 0.99 | 0.99 |
| | LPIPS (↓) | 0.05 | 0.02 | 0.07 | 0.06 | 0.01 | 0.04 |
| DIV2K | PSNR (↑) | 38.4 | 38.7 | 38.2 | 39.0 | 39.1 | 38.8 |
| | SSIM (↑) | 0.98 | 0.99 | 0.98 | 0.99 | 0.99 | 0.99 |
| | LPIPS (↓) | 0.07 | 0.03 | 0.11 | 0.04 | 0.01 | 0.03 |

## 5.3 Detection and decoding

We benchmark WAM's performance against several watermarking methods: DCTDWT (Al-Haj, 2007), HiDDeN (Zhu et al., 2018), TrustMark (Bui et al., 2023a), SSL (Fernandez et al., 2022), FNNS (Kishore et al., 2022). We also compare WAM to generation-time watermarking methods (Fernandez et al., 2023a; Wen et al., 2023) in App. E.1. Although this list is not exhaustive, it covers the main types of state-of-the-art methods. For HiDDeN, we use a model which hides 48 bits. FNNS, SSL, and DCTDWT can hide messages or arbitrary length, we choose to hide 48 bits as well. We use the first 16 bits for detection and 32 bits for message decoding. For detection, an image is flagged as watermarked if at most 1 bit is wrongly decoded. This corresponds to a theoretical False Positive Rate (FPR) of $2.6 \times 10^{-4}$, i.e., 2.6 out of 10,000 images are falsely flagged as watermarked on average. TrustMark (Bui et al., 2023a) directly outputs a boolean (watermarked or not). We use the version that hides 40 bits, but only use the first 32 bits. For all evaluations, we apply the watermark embedding and extraction at the original image resolution as described in the high-resolution paragraph of Sec. 4.1. Finally, for WAM, an image is flagged if $s_{\text{det}}$ (Eq. 1) is higher than 0.07. This value empirically delivers an approximately similar FPR on the COCO validation set.

We evaluate the robustness against various transformations: *geometric* (flip, crop, perspective, and rotation), *valuemetric* (adjustments in brightness, hue, contrast or saturation, median or Gaussian filtering, and JPEG compression), *splicing*: scenarios where only 10% of the image is watermarked superimposed onto the original or an other background. We also evaluate the robustness against *inpainting*, using the LaMa (Suvorov et al., 2022) model to inpaint areas of the watermarked image specified by random or segmentation masks. For COCO, we use the union of all segmentation masks for each image which in average corresponds to 30% of the image, and for DIV2k we use a randomly generated mask (because there are no segmentation masks) that covers around 35% of the image. We give details on the compared baselines in App. D.3 and on the transformations in App. D.4.

Table 2a presents the detection and decoding results for low/mid-resolution images, averaged over the first 10k images of the COCO validation set – the detailed results for every augmentation used to form the different groups are in App. E.3. Table 2b does the same for high-resolution images from the DIV2k (Timofte et al., 2018) validation set. For both distributions, WAM is competitive and even shows improved robustness on classical *geometric*/*valuemetric* transformations. Most importantly, WAM performs better on *splicing* or *inpainting* where other methods fail to achieve more than 90% TPR and Bit acc. This is true even if WAM was not explicitly trained on high resolution images or to be robust against inpainting. We also show examples of WAM's detection masks after inpainting in Fig. 10 of App. E, where some modified parts of images are not detected as watermarked anymore.

## 5.4 Localization

WAM and EditGuard (Zhang et al., 2024) are the only methods that provide watermark localization. We therefore consider images at fixed resolution $512 \times 512$ (unlike in Sec. 5.3) to align with the setup of EditGuard. We nevertheless observe similar results at different resolutions for WAM. We focus on the COCO validation set and splice centered watermarked rectangles of various sizes within the images to ensure a well-controlled experiment. We then either apply no transformation or crop the upper left part of the image (25%), which is then resized to the original size. By doing so, the proportion of

**Table 2** Detection and decoding after image editing (detailed in Sec. D.4). We show the bit accuracy (Bit acc.) between the encoded and decoded messages, the proportion of images correctly deemed watermarked (TPR), and the proportion of non-watermarked images falsely detected as watermarked (FPR), in %. Since HiDDeN, DCTDWT, SSL and FNNS do not naturally provide a detection result, we hide 48 bits and reserve 16 bits for detection, and flag an image as watermarked if it has strictly less than two bits incorrectly decoded (these baselines are detailed in Sec. 5.3).

**(a)** On the first 10k validation images of COCO (low to mid resolution)

| | | Augmentations | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | None | | Geometric | | Valuemetric | | Inpainting | | Splicing | |
| Method | FPR | TPR | Bit acc. | TPR | Bit acc. | TPR | Bit acc. | TPR | Bit acc. | TPR | Bit acc. |
| HiDDeN | 0.08 | 76.9 | 95.5 | 31.2 | 80.1 | 48.4 | 87.2 | 44.7 | 88.7 | 0.9 | 68.0 |
| DWTDCT | 0.02 | 77.1 | 91.4 | 0.0 | 50.5 | 13.6 | 58.1 | 47.8 | 81.7 | 0.8 | 59.9 |
| SSL | 0.00 | 99.9 | 100.0 | 14.3 | 76.5 | 70.5 | 92.1 | 67.7 | 91.1 | 0.4 | 58.9 |
| FNNS | 0.10 | 99.6 | 99.9 | 62.4 | 86.6 | 82.1 | 93.9 | 89.4 | 97.3 | 38.7 | 88.5 |
| TrustMark | 2.88 | 99.8 | 99.9 | 36.3 | 71.4 | 90.1 | 98.2 | 39.4 | 83.2 | 83.2 | 57.1 |
| WAM (ours) | 0.04 | 100.0 | 100.0 | 99.3 | 91.8 | 100.0 | 99.9 | 97.9 | 99.2 | 100.0 | 95.3 |

**(b)** On the 100 validation images of DIV2k (high resolution)

| | | Augmentations | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | None | | Geometric | | Valuemetric | | Inpainting | | Splicing | |
| Method | FP | TPR | Bit acc. | TPR | Bit acc. | TPR | Bit acc. | TPR | Bit acc. | TPR | Bit acc. |
| HiDDeN | 0 | 72.0 | 95.8 | 33.5 | 81.3 | 53.3 | 88.1 | 53.3 | 88.1 | 1.5 | 70.2 |
| DWTDCT | 0 | 75.0 | 88.9 | 0.0 | 50.6 | 18.4 | 58.9 | 73.0 | 86.7 | 31.5 | 71.6 |
| SSL | 0 | 100.0 | 100.0 | 32.5 | 83.6 | 75.8 | 92.8 | 95.0 | 96.1 | 0.5 | 59.8 |
| FNNS | 0 | 97.0 | 99.9 | 63.5 | 87.2 | 79.8 | 93.9 | 94.0 | 99.2 | 48.5 | 89.5 |
| TrustMark | 5 | 100.0 | 100.0 | 33.1 | 70.5 | 87.4 | 97.9 | 0.0 | 72.1 | 1.5 | 57.3 |
| WAM (ours) | 0 | 100.0 | 99.9 | 96.1 | 89.0 | 100.0 | 99.9 | 100.0 | 99.8 | 99.5 | 94.2 |

watermarked pixels is still the same as in the spliced image before cropping, which allows us to evaluate the robustness of the localization. Figure 7 of App. D.5 illustrates this protocol.

Figure 4 reports the mean Intersection over Union (mIoU) to evaluate the localization, as commonly done in the segmentation literature. Figure 4 evaluates the bit accuracy – through localization – following Eq. 2. We observe that WAM accurately predicts both classes, even after cropping and resizing, except when the watermarked area covers 95% of the image, in which case the extractor tends to classify all pixels as watermarked. In terms of bit accuracy, WAM recovers in average 31 out of 32 bits even when only 10% of the $256 \times 256$ image is watermarked, and around 25 bits when only 10% of a 25% crop is watermarked (which corresponds to 2.5% of the overall number of pixels). For both evaluations, WAM outperforms EditGuard which, in particular, is not robust to cropping.

## 5.5 Multiple watermarks

We compare the detection and decoding of multiple watermarks before and after the second training phase of WAM (Sec. 4.3). We embed up to five distinct messages into five separate 10% areas of every image (first resized to 256 to ease the experiments). This is done by feeding the image several times to WAM's embedder, then pasting the different watermarked areas onto the original image. These areas are squares disposed in a checkerboard pattern, which ensures that the watermarked areas do not overlap and have the same size (see Fig. 8 of App. D.6). This is an arbitrary choice made to remove confounding factors during evaluation, although the training does not require the watermarked areas to be squared nor to have the same size. Following the methodology detailed in Sec. 3, we apply the DBSCAN algorithm to the vector outputs $\tilde{m}_i$ of the extractor corresponding to all pixels $i$ identified as watermarked ($y_i^{\text{det}} > \tau$). It identifies clusters corresponding to different watermarked regions without requiring the number of hidden messages to be known in advance. We use $\tau = 0.5$ to threshold the watermarked pixels, and we choose a rather strict setup with $\varepsilon = 1$ and $\min_{\text{samples}} = 1000$ pixels

(around 2% of the image) in our evaluations, resulting from an hyperparameter search detailed in Fig. 9 of App. E.2. This means that a cluster will be considered only if it has at least 2% of the image's pixels, and that the maximum distance between two predicted messages to be neighbors is 1.

Figure 5 presents the results. We compute the bit accuracies by comparing the centroid of each detected cluster (decoded message) with the ground truth message that has the largest overlapping area: the bit accuracy is thus computed only across the clusters that are discovered by DBSCAN. Without the second phase of training, the messages get mixed up, and WAM predicts the same (wrong) message for all watermarked pixels. After the training, WAM is able to accurately extract up to five different 32-bit messages from the areas covering 10% of the image each. Therefore, although the second phase of training embeds between 1 and 3 watermarks per image, WAM generalizes to more watermarks. It also shows that WAM's effective capacity is greater than 32 bits (since it can hide multiple messages) if we do not consider heavy crops as a transformation we want the method to be robust against, similarly as (Bui et al., 2023b; Tancik et al., 2020; Wen et al., 2023).

This method remains effective after some image manipulations; extended quantitative results are presented in Fig. 9 of App. E.2. For example, WAM achieves 85% mIoU even when the images have been horizontally flipped and the contrast adjusted. Furthermore, the bit accuracy, averaged across 5 messages totaling 160 bits, exceeds 95% under the same conditions. However, WAM fails when JPEG compression is also added on top of these two transformations. Illustrative examples of the identified clusters in various scenarios can be found in Fig. 8 of App. D.6.

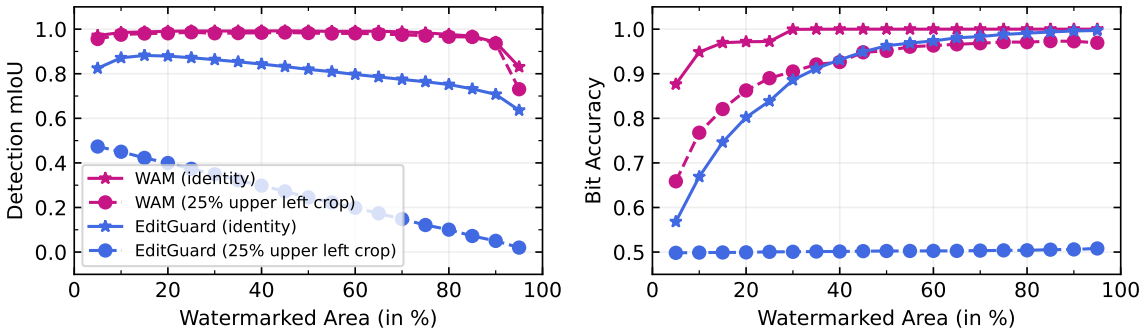
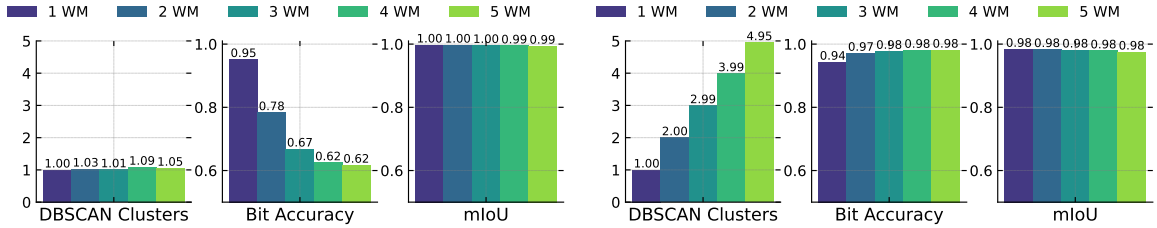



**Figure 4** Evaluation of the localization on the validation set of COCO, with or without cropping before extraction, following the setup described in Sec. 5.4. (Left) Localization accuracy using intersection over union between the predicted watermarked areas and the ground-truth mask. (Right) Bit accuracy between the ground truth message and the decoded message, computed from Eq. (2).




**(a)** After first training phase (PSNR ≈ 25 dB)

**(b)** After second training phase (PSNR ≈ 38 dB)

**Figure 5** Results on multiple watermarks extracted from a single image. We use non overlapping 10 % rectangular masks to watermark up to 5 parts of images from COCO, with different messages, and report the average number of clusters detected by DBSCAN, bit accuracy across found messages, as well as the mIoU of watermark detection on all objects. (Left) After the first training phase, the bit accuracy strongly decreases as the number of watermarks grows. (Right) After fine-tuning, it stays roughly constant no matter the number of watermarked parts. The mIoU stays stable in both cases.

## 6 Conclusion, Limitations, and Future Work

**Conclusion.** This work introduces the Watermark Anything Model, which approaches image watermarking as a segmentation task. WAM is able to predict whether an image is watermarked or not, as well as to localize its watermarked regions. It thus handles images with a small watermarked part, or where parts of the image have been removed or edited. Additionally, it is able to detect and extract multiple watermarks within the same image, for the first time in the literature. Our training which introduces localization under heavy augmentations also offers state-of-the-art robustness on classical settings considered by current watermark methods, where most of the image is watermarked.

We identify two main limitations that we address in the next paragraphs.

**Low payload.** In our experiments, WAM's capacity is limited to 32 bits, and training on larger messages is challenging. In contrast, other watermarking methods such as EditGuard or RoSteALS can successfully embed more than 100 bits per image, but are not robust to crops or outpainting. This is a trade-off between capacity and robustness. Note that in practice, other methods use decoding to match with the original message and conclude if an image is watermarked. In contrast, WAM's detection process is separate from the decoding, so 32 bits may be enough for most applications.

**Perceptual quality.** In spite of the JND weighting, we notice that the watermark can still be visible in some areas of the watermarked images, even at a relatively high PSNR (see examples in App. F). This could be due to the JND map focusing only on the cover image and not on the watermark signal itself. Improvements might be achieved by employing more sophisticated Human Visual System (HVS) models or by regularizing the watermark to eliminate repetitive patterns during training.

## Reproducibility statement

Section 5 provides the specifics of our training and evaluation setup. Further implementation details, including network architectures, mask designs, transformation parameters for robustness evaluation, and settings for baseline comparisons, are presented in App. D. The models and code for inference and training used in this paper will be made available upon publication. The code is based on PyTorch, the training dataset is publicly available, and the training requires half a week on 8 V100 GPUs, which makes the models reproducible at reasonable cost.

The training and inference code, as well as trained models are available at: https://github.com/facebookresearch/watermark-anything

## References


Mahdi Ahmadi, Alireza Norouzi, Nader Karimi, Shadrokh Samavi, and Ali Emami. Redmark: Framework for residual diffusion watermarking based on deep networks. *Expert Systems with Applications*, 2020.

Ali M. Al-Haj. Combined dwt-dct digital image watermarking. *Journal of Computer Science*, 3:740–746, 2007. https://api.semanticscholar.org/CorpusID:1454866.

Vishal Asnani, Xi Yin, Tal Hassner, Sijia Liu, and Xiaoming Liu. Proactive image manipulation detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15386–15395, 2022.

Vishal Asnani, Abhinav Kumar, Suya You, and Xiaoming Liu. Probed: proactive object detection wrapper. *Advances in Neural Information Processing Systems*, 36, 2024.

Shumeet Baluja. Hiding images in plain sight: Deep steganography. *NeurIPS*, 2017.

Mauro Barni, Franco Bartolini, Vito Cappellini, and Alessandro Piva. A dct-domain system for robust image watermarking. *Signal processing*, 66(3):357–372, 1998.

Mauro Barni, Franco Bartolini, and Alessandro Piva. Improved wavelet-based watermarking through pixel-wise masking. *IEEE transactions on image processing*, 10(5):783–791, 2001.

Mauro Barni, Franco Bartolini, and Nicola Checcacci. Watermarking of mpeg-4 video objects. *IEEE Transactions on Multimedia*, 7(1):23–32, 2005.

Patrick Bas, Nikolaos V Boulgouris, Filippos D Koravos, Jean-Marc Chassery, Michael G Strintzis, and Benoit MM Macq. Robust watermarking of video objects for mpeg-4 applications. In *Applications of Digital Image Processing XXIV*, volume 4472, pages 85–94. SPIE, 2001.

Patrick Bas, J-M Chassery, and Benoit Macq. Geometrically invariant watermarking using feature points. *IEEE transactions on image Processing*, 11(9):1014–1028, 2002.

Patrick Bas, Nicolas Le Bihan, and J-M Chassery. Color image watermarking using quaternion fourier transform. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 3, pages III–521. IEEE, 2003.

Adrian G Bors and Ioannis Pitas. Image watermarking using dct domain constraints. In *ICIP*, 1996.

Tu Bui, Shruti Agarwal, and John Collomosse. Trustmark: Universal watermarking for arbitrary resolution images. *arXiv preprint arXiv:2311.18297*, 2023a.

Tu Bui, Shruti Agarwal, Ning Yu, and John Collomosse. Rosteals: Robust steganography using autoencoder latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 933–942, 2023b.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *ICCV*, 2021.

Xiangyu Chen, Varsha Kishore, and Kilian Q Weinberger. Learning iterative neural optimizers for image steganography. In *The Eleventh International Conference on Learning Representations*, 2022.

Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in neural information processing systems*, 34:17864–17875, 2021.

Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1290–1299, 2022.

Chun-Hsien Chou and Yun-Chin Li. A perceptually tuned subband image coder based on the measure of just-noticeable-distortion profile. *IEEE Transactions on circuits and systems for video technology*, 5(6): 467–476, 1995.

Vincent Christlein, Christian Riess, Johannes Jordan, Corinna Riess, and Elli Angelopoulou. An evaluation of popular copy-move forgery detection approaches. *IEEE Transactions on information forensics and security*, 7(6):1841–1854, 2012.

Hai Ci, Yiren Song, Pei Yang, Jinheng Xie, and Mike Zheng Shou. Wmadapter: Adding watermark control to latent diffusion models. *arXiv preprint arXiv:2406.08337*, 2024a.

Hai Ci, Pei Yang, Yiren Song, and Mike Zheng Shou. Ringid: Rethinking tree-ring watermarking for enhanced multi-key identification. *arXiv preprint arXiv:2404.14055*, 2024b.

Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital watermarking and steganography*. Morgan kaufmann, 2007.

Ingemar J Cox, Joe Kilian, F Thomson Leighton, and Talal Shamoon. Secure spread spectrum watermarking for multimedia. *IEEE transactions on image processing*, 6(12):1673–1687, 1997.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Matthijs Douze, Giorgos Tolias, Ed Pizzi, Zoë Papakipos, Lowik Chanussot, Filip Radenovic, Tomas Jenicek, Maxim Maximov, Laura Leal-Taixé, Ismail Elezi, et al. The 2021 image similarity dataset and challenge. *arXiv preprint arXiv:2106.09672*, 2021.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, 1996.

Gautier Evennou, Vivien Chappelier, Ewa Kijak, and Teddy Furon. Swift: Semantic watermarking for image forgery thwarting. *arXiv preprint arXiv:2407.18995*, 2024.

Jianwei Fei, Zhihua Xia, Benedetta Tondi, and Mauro Barni. Supervised gan watermarking for intellectual property protection. In *2022 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2022.

Jianwei Fei, Zhihua Xia, Benedetta Tondi, and Mauro Barni. Robust retraining-free gan fingerprinting via personalized normalization. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2023.

Jianwei Fei, Zhihua Xia, Benedetta Tondi, and Mauro Barni. Wide flat minimum watermarking for robust ownership verification of gans. *IEEE Transactions on Information Forensics and Security*, 2024.

Liu Ping Feng, Liang Bin Zheng, and Peng Cao. A dwt-dct based blind watermarking algorithm for copyright protection. In *ICCSIT*. IEEE, 2010.

Weitao Feng, Wenbo Zhou, Jiyan He, Jie Zhang, Tianyi Wei, Guanlin Li, Tianwei Zhang, Weiming Zhang, and Nenghai Yu. Aqualora: Toward white-box protection for customized stable diffusion models via watermark lora. *arXiv preprint arXiv:2405.11135*, 2024.

Pierre Fernandez, Alexandre Sablayrolles, Teddy Furon, Hervé Jégou, and Matthijs Douze. Watermarking images in self-supervised latent spaces. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3054–3058. IEEE, 2022.

Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22466–22477, 2023a.

Pierre Fernandez, Matthijs Douze, Hervé Jégou, and Teddy Furon. Active image indexing. In *International Conference on Learning Representations (ICLR)*, 2023b.

Teddy Furon and Patrick Bas. Broken arrows. *EURASIP Journal on Information Security*, 2008:1–13, 2008.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Seongmin Hong, Kyeonghyun Lee, Suh Yoon Jeon, Hyewon Bae, and Se Young Chun. On exact inversion of dpm-solvers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7069–7078, 2024.

Jiangtao Huang, Ting Luo, Li Li, Gaobo Yang, Haiyong Xu, and Chin-Chen Chang. Arwgan: Attention-guided robust image watermarking model based on gan. *IEEE Transactions on Instrumentation and Measurement*, 72:1–17, 2023.

Zhaoyang Jia, Han Fang, and Weiming Zhang. Mbrs: Enhancing robustness of dnn-based watermarking by mini-batch of real and simulated jpeg compression. In *Proceedings of the 29th ACM international conference on multimedia*, pages 41–49, 2021.

Qiuping Jiang, Zhentao Liu, Shiqi Wang, Feng Shao, and Weisi Lin. Towards top-down just noticeable difference estimation of natural images. *IEEE Transactions on Image Processing*, 2022.

Junpeng Jing, Xin Deng, Mai Xu, Jianyi Wang, and Zhenyu Guan. Hinet: Deep image hiding by invertible network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4733–4742, 2021.

Lei Ke, Mingqiao Ye, Martin Danelljan, Yu-Wing Tai, Chi-Keung Tang, Fisher Yu, et al. Segment anything in high quality. *Advances in Neural Information Processing Systems*, 36, 2024.

Changhoon Kim, Kyle Min, Maitreya Patel, Sheng Cheng, and Yezhou Yang. Wouaf: Weight modulation for user attribution and fingerprinting in text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8974–8983, 2024.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

Varsha Kishore, Xiangyu Chen, Yan Wang, Boyi Li, and Kilian Q Weinberger. Fixed neural network steganography: Train the images, not the network. In *International Conference on Learning Representations*, 2022.

Lester E Krueger. Reconciling fechner and stevens: Toward a unified psychophysical law. *Behavioral and Brain Sciences*, 12(2):251–267, 1989.

D. Kundur and D. Hatzinakos. Digital watermarking for telltale tamper proofing and authentication. *Proceedings of the IEEE*, 87(7):1167–1180, 1999. doi: 10.1109/5.771070.

Ema Kušen and Mark Strembeck. Politics, sentiments, and misinformation: An analysis of the twitter discussion on the 2016 austrian presidential elections. *Online Social Networks and Media*, 5:37–50, 2018.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

Liangqi Lei, Keke Gai, Jing Yu, and Liehuang Zhu. Diffusetrace: A transparent and flexible watermarking scheme for latent diffusion model. *arXiv preprint arXiv:2405.02696*, 2024.

Zhen Li, Kim-Hui Yap, and Bai-Ying Lei. A new blind robust image watermarking scheme in svd-dct composite domain. In *ICIP*, 2011.

Eugene T. Lin, Christine I. Podilchuk, and Edward J. Delp III. Detection of image alterations using semifragile watermarks. In Ping Wah Wong and Edward J. Delp III, editors, *Security and Watermarking of Multimedia Contents II*, volume 3971, pages 152 – 163. International Society for Optics and Photonics, SPIE, 2000. doi: 10.1117/12.384969. https://doi.org/10.1117/12.384969.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Sasha Luccioni, Bruna Trevelin, and Margaret Mitchell. The environmental impacts of ai – primer. *Hugging Face Blog*, September 2024. https://huggingface.co/blog/community.

Xiyang Luo, Ruohan Zhan, Huiwen Chang, Feng Yang, and Peyman Milanfar. Distortion agnostic deep watermarking. In *CVPR*, 2020.

Jun Ma, Yuting He, Feifei Li, Lin Han, Chenyu You, and Bo Wang. Segment anything in medical images. *Nature Communications*, 15(1):654, 2024.

Rui Ma, Mengxi Guo, Yi Hou, Fan Yang, Yuan Li, Huizhu Jia, and Xiaodong Xie. Towards blind watermarking: Combining invertible and non-invertible mechanisms. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 1532–1542, 2022.

Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *International Conference on Multimedia*. ACM, 2010.

Zhicheng Ni, Yun-Qing Shi, Nirwan Ansari, and Wei Su. Reversible data hiding. *IEEE Transactions on circuits and systems for video technology*, 2006.

Nikos Nikolaidis and Ioannis Pitas. Robust image watermarking in the spatial domain. *Signal processing*, 1998.

Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10): e3, 2016.

State Council of the People's Republic of China. New generation artificial intelligence development plan, 2023. https://www.gov.cn/zhengce/content/202306/content_6884925.htm.

Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

Junlin Ouyang, Gouenou Coatrieux, Beijing Chen, and Huazhong Shu. Color image watermarking based on quaternion fourier transform and improved uniform log-polar mapping. *Computers & Electrical Engineering*, 2015.

Minzhou Pan, Yi Zeng, Xue Lin, Ning Yu, Cho-Jui Hsieh, Peter Henderson, and Ruoxi Jia. Jigmark: A black-box approach for enhancing image watermarks against diffusion model edits. *arXiv preprint arXiv:2406.03720*, 2024.

European Parliament and Council. Regulation (eu) 2024/1689 of the european parliament and of the council on artificial intelligence, 2024. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R1689.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Alessandro Piva, Mauro Barni, Franco Bartolini, and Vito Cappellini. Dct-based watermark recovering without resorting to the uncorrupted original image. In *Proceedings of international conference on image processing*, volume 1, pages 520–523. IEEE, 1997.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

Ahmad Rezaei, Mohammad Akbari, Saeed Ranjbar Alvar, Arezou Fatemi, and Yong Zhang. Lawa: Using latent space for in-generation image watermarking. *arXiv preprint arXiv:2408.05868*, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

Robin San Roman, Pierre Fernandez, Hady Elsahar, Alexandre Défossez, Teddy Furon, and Tuan Tran. Proactive detection of voice cloning with localized watermarking. In *International Conference on Machine Learning*, 2024.

Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.

Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021.

Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2149–2159, 2022.

Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *CVPR*, 2020.

Radu Timofte, Shuhang Gu, Jiqing Wu, Luc Van Gool, Lei Zhang, Ming-Hsuan Yang, Muhammad Haris, et al. Ntire 2018 challenge on single image super-resolution: Methods and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

Dijana Tralic, Ivan Zupancic, Sonja Grgic, and Mislav Grgic. Comofod—new database for copy-move forgery detection. In *Proceedings ELMAR-2013*, pages 49–54. IEEE, 2013.

Matthieu Urvoy, Dalila Goudia, and Florent Autrusseau. Perceptual dft watermarking with improved detection and robustness to geometrical distortions. *IEEE Transactions on Information Forensics and Security*, 2014.

USA. Ensuring safe, secure, and trustworthy ai. https://www.whitehouse.gov/wp-content/uploads/2023/07/Ensuring-Safe-Secure-and-Trustworthy-AI.pdf, July 2023. Accessed: [july 2023].

Ron G Van Schyndel, Andrew Z Tirkel, and Charles F Osborne. A digital watermark. In *Proceedings of 1st international conference on image processing*, volume 2, pages 86–90. IEEE, 1994.

Vedran Vukotić, Vivien Chappelier, and Teddy Furon. Are deep neural networks good for blind image watermarking? In *WIFS*, 2018.

Vedran Vukotić, Vivien Chappelier, and Teddy Furon. Are classification deep neural networks good for blind image watermarking? *Entropy*, 2020.

Andrew B Watson. Dct quantization matrices visually optimized for individual images. In *Human vision, visual processing, and digital display IV*, volume 1913, pages 202–216. SPIE, 1993.

Bingyang Wen and Sergul Aydore. Romark: A robust watermarking system using adversarial training. *arXiv preprint arXiv:1910.01221*, 2019.

Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *arXiv preprint arXiv:2305.20030*, 2023.

Eric Wengrowski and Kristin Dana. Light field messaging with deep photographic steganography. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1515–1524, 2019.

Hanzhou Wu, Gen Liu, Yuwei Yao, and Xinpeng Zhang. Watermarking neural networks with watermarked images. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(7):2591–2601, 2020.

Jinjian Wu, Leida Li, Weisheng Dong, Guangming Shi, Weisi Lin, and C-C Jay Kuo. Enhanced just noticeable difference model for images with pattern complexity. *IEEE Transactions on Image Processing*, 2017.

Xiang-Gen Xia, Charles G Boncelet, and Gonzalo R Arce. Wavelet transform based watermark for digital images. *Optics Express*, 1998.

XK Yang, WS Ling, ZK Lu, Ee Ping Ong, and SS Yao. Just noticeable distortion model and its applications in video coding. *Signal processing: Image communication*, 20(7):662–680, 2005.

Chong Yu. Attention based data hiding with generative adversarial networks. In *AAAI*, 2020.

Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 14448–14457, 2021.

Ning Yu, Vladislav Skripniuk, Dingfan Chen, Larry Davis, and Mario Fritz. Responsible disclosure of generative models using scalable fingerprinting. In *International Conference on Learning Representations (ICLR)*, 2022.

Aditi Zear, Amit Kumar Singh, and Pardeep Kumar. A proposed secure multiple watermarking technique based on dwt, dct and svd for application in medicine. *Multimedia tools and applications*, 77:4863–4882, 2018.

Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023.

Honglei Zhang, Hu Wang, Yuanzhouhan Cao, Chunhua Shen, and Yidong Li. Robust watermarking using inverse gradient attention. *arXiv preprint arXiv:2011.10850*, 2020.

Kevin Alex Zhang, Alfredo Cuesta-Infante, Lei Xu, and Kalyan Veeramachaneni. Steganogan: High capacity image steganography with gans. *arXiv preprint arXiv:1901.03892*, 2019a.

Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust invisible video watermarking with attention. *arXiv preprint arXiv:1909.01285*, 2019b.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Xiaohui Zhang, Weisi Lin, and Ping Xue. Just-noticeable difference estimation with pixels in images. *Journal of Visual Communication and Image Representation*, 19(1):30–41, 2008.

Xuanyu Zhang, Runyi Li, Jiwen Yu, Youmin Xu, Weiqi Li, and Jian Zhang. Editguard: Versatile image watermarking for tamper localization and copyright protection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11964–11974, 2024.

Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. Fast segment anything. *arXiv preprint arXiv:2306.12156*, 2023a.

Yunqing Zhao, Tianyu Pang, Chao Du, Xiao Yang, Ngai-Man Cheung, and Min Lin. A recipe for watermarking diffusion models. *arXiv preprint arXiv:2303.10137*, 2023b.

Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6881–6890, 2021.

Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 657–672, 2018.

# Appendix

## A Ethical Statement

### A.1 Societal Impact

Watermarking in general improves the traceability of content, be it AI-generated, or not. It can have positive consequences, for example when it is used to trace the origin of fake news or to protect intellectual property. This traceability can also have negative consequences, for example when it is used to trace political opponents in authoritarian regimes or whistleblowers in secretive companies. Besides, it is not clear how to disclose watermark detection results, which may foster a closed ecosystem of detection tools. It may also exacerbate misinformation by placing undue emphasis on content that is either not detected, generated by unknown models, or authentic but used out of context. We however believe that the benefits of watermarking outweigh the risks, and that the development of robust watermarking methods is a positive step for our society.

### A.2 Environmental impact

The cost of the experiments and of model training is high, though order of magnitude less than other computer vision fields (Oquab et al., 2023). One training with a schedule similar to the one reported in the paper represents $\approx$ 30 GPU-days. We also roughly estimate that the total GPU-days used for running all our experiments to 5000, or $\approx$ 120k GPU-hours. This amounts to total emissions in the order of 20 tons of $CO_2$eq. Estimations are conducted using the Machine Learning Impact calculator presented by Lacoste et al. (2019). We do not consider in this approximation: memory storage, CPU-hours, production cost of GPUs/ CPUs, etc.

We were careful to limit the environmental impact of our research by doing most of our experiments on smaller models, on fewer epochs (100) and at lower resolutions ($128 \times 128$), before scaling up to larger models, more epochs and higher resolutions. We also focus on small specialized model that are efficient at inference time and more environmental-friendly (Luccioni et al., 2024). At the end of the day, we believe that the environmental cost of this research is justified by the potential benefits of WAM.

## B Extended Related Work on Image Watermarking

### B.1 Traditional watermarking

We call traditional watermarking a technique embedding a digital watermark in a host content. As far as images are concerned, the first methods are usually classified into two categories depending on the space in which the watermark is embedded. In *spatial domain*, the watermark is encoded by directly modifying pixels, such as flipping low-order bits of selected pixels (Van Schyndel et al., 1994). For example, Nikolaidis and Pitas (1998) slightly modify the intensity of randomly selected image pixels while taking into account properties of the human visual system, robustly to JPEG compression and lowpass filtering. Bas et al. (2002) create content descriptors defined by salient points and embed the watermark by adding a pattern on triangles formed by the tessellation of these points. Ni et al. (2006) use the zero or the minimum points of the histogram of an image and slightly modifies the pixel grayscale values to embed data into the image. The second category *frequency domain* watermarking offers better robustness. It usually spreads a pseudorandom noise sequence across the entire frequency spectrum of the host signal (Cox et al., 1997). The first step is a transformation that computes the frequency coefficients. The watermark is then added to these coefficients, taking into account the human visual system. The coefficients are mapped back onto the original pixel space through the inverse transformation to generate the watermarked image. The transform domains include Discrete Fourier Transform (DFT) (Urvoy et al., 2014), Quaternion Fourier Transform (QFT) (Bas et al., 2003; Ouyang et al., 2015), Discrete Cosine Transform (DCT) (Bors and Pitas, 1996; Piva et al., 1997; Barni et al., 1998; Li et al., 2011), Discrete Wavelet Transform (DWT) (Xia et al., 1998; Barni et al., 2001; Furon and Bas, 2008), both DWT and DCT (Feng et al., 2010; Zear et al., 2018), etc.

Deep learning-based methods have recently emerged as alternatives for traditional watermarking. The first attempts use neural networks as a fixed transform into a latent space (Vukotić et al., 2018, 2020; Kishore et al., 2022; Fernandez et al., 2022) Since there is no inverse transform, the embedding is done iteratively by gradient descent over the pixels. The recent approaches are often built as embedder/extractor networks. They are trained end-to-end to invisibly encode information while being resilient to transformations applied during training. This makes it easier to build robust systems and avoids algorithms hand-crafted for specific transformations. HiDDeN (Zhu et al., 2018) is a famous representative of this approach and has been extended in several ways. Luo et al. (2020) add adversarial training in the attack simulation to bring robustness to unknown transformations. Zhang et al. (2019b, 2020); Yu (2020) use an attention filter further improving imperceptibility. Ahmadi et al. (2020) adds a circular convolutional layer that helps spreading the watermark signal over the image. Wen and Aydore (2019) use robust optimization with worst-case attack as if an adversary were trying to remove the mark. Many other approaches focused on improving robustness, imperceptibility, speed, etc. (Jia et al., 2021; Bui et al., 2023b,a; Huang et al., 2023; Evennou et al., 2024; Pan et al., 2024).

Another line of works focuses on steganography (Baluja, 2017; Wengrowski and Dana, 2019; Zhang et al., 2019a; Tancik et al., 2020; Jing et al., 2021; Ma et al., 2022) that pursues a different goal. Steganography hides a message in the image without leaving any statistical traces, but the robustness is null or limited.

### B.2 Watermarking for generative AI

The most trendy research direction in watermarking is arguably the detection of AI-generated images for transparency and for filtering such results when building new models. The *Post-generation* or *post-hoc* approach uses traditional watermarking: the content is first generated and then watermarked. On the contrary, the *generation-time* methods natively generate watermarked images. Watermarking no longer incurs additional runtime and is more robust and secure than post-hoc. We broadly categorize generation-time watermarking into the two following categories.

*In-model* methods modify the weights of the generative model. This allows open-sourcing the model without revealing the watermark. The earliest methods (Wu et al., 2020; Yu et al., 2021; Zhao et al., 2023b) watermark the images of the training set with the hope that the model learns what watermark is during its training. This is computationally expensive and not scalable. Alternatively, some proposals (Fei et al., 2022, 2024) train Generative Adversarial Networks (GAN) with additional watermarking losses such that generated images contain the watermark. Stable Signature (Fernandez et al., 2023a) focuses on Latent Diffusion Models (LDM) and fine-tunes the latent decoder to embed the watermark, while Feng et al. (2024) fine-tune the U-Net that predicts the latent diffusion noise instead. To eliminate the need to fine-tune the model for every user, some papers use a hyper-network predicting the modifications to be applied to the generative model, be it a GAN (Yu et al., 2022; Fei et al., 2023) or diffusion-based (Kim et al., 2024).

*Out-of-model* methods alter the generation process. This is easier to implement since it does not require training or fine-tuning the model. Ci et al. (2024a) and Rezaei et al. (2024) propose an adapter to the decoder that takes the secret message as input. A different class of out-of-model methods, specific to diffusion models, embed the watermark by adding patterns to the initial noise (seed). For example, Tree-Ring (Wen et al., 2023) adds circular patterns to the initial noise and inverts the diffusion process to extract the watermark. As follow-up works, Hong et al. (2024) improve the inversion of the diffusion process, Ci et al. (2024b) extend the method to multi-bit watermarking, and Lei et al. (2024) deploy the embedder/extractor framework over the noise.

## C Algorithms details

### C.1 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Ester et al., 1996; Schubert et al., 2017) is a clustering method that groups points in a dataset based on their density and proximity to each other. In our case, points are locally decoded messages (one per pixel deemed as watermarked). It works as follows:

1. Initialization: Set the parameters $\varepsilon$ (maximum distance between two samples for one to be considered as in the neighborhood of the other) and $\min_{\text{samples}}$ (minimum number of samples in a cluster).
2. Neighborhood search: For each point $p$ in the dataset, find all points within a distance of $\varepsilon$ from $p$. This forms the neighborhood of $p$.
3. Cluster formation: If the neighborhood of $p$ contains at least $\min_{\text{samples}}$ points, form a cluster around $p$. Otherwise, mark $p$ as noise.
4. Cluster expansion: For each point $q$ in the cluster, find all points within a distance of $\varepsilon$ from $q$. Add these points to the cluster if they are not already part of it.
5. Repeat steps 3-4: Continue expanding the cluster until no more points can be added.
6. Assign labels: Assign a label to each point in the dataset indicating which cluster it belongs to (if any).

It is important to note that $\varepsilon$ is neither a hard boundary nor a maximum bound on the distances of points within a cluster: points that are further apart than $\varepsilon$ but still within the neighborhood of a core point (a point with at least $\min_{\text{samples}}$ neighbors within $\varepsilon$ distance) can also be part of the same cluster.

One of the significant advantages of DBSCAN is its ability to cluster points without knowing the number of clusters beforehand, unlike some other clustering algorithms like K-Means. Additionally, it identifies arbitrarily shaped clusters (watermarked area in our case), including those that may be surrounded by a different cluster. This clustering is also robust to outliers. However, DBSCAN also has some limitations. It is sensitive to hyper-parameters $\varepsilon$ and $\min_{\text{samples}}$, and to the choice of the distance metric (bit difference in our case), which may affect the clustering results. DBSCAN may also be computationally expensive for a large number of data points. This is less of an issue in our case since it is lower than $256 \times 256$.

We use the Scikit-learn (Pedregosa et al., 2011) implementation of DBSCAN in our experiments.

### C.2 Just-Noticeable-Difference

The maximum change that the human visual system (HVS) cannot perceive is referred to as the Just-Noticeable-Difference (JND) (Krueger, 1989). It is used in image/video watermarking, compression, quality assessment, etc. JND models in the pixel domain directly calculate the JND at each pixel location (i.e., how much pixel difference is perceivable by the HVS).

This section describes in detail the JND map used in Sec. 4.3. It is based on the work of Chou and Li (1995). We use this model for its simplicity, its efficiency, and its good qualitative results. More complex HVS models could also be used if higher imperceptibility is needed (Watson, 1993; Yang et al., 2005; Zhang et al., 2008; Jiang et al., 2022). The JND map takes into account two characteristics of the HVS, namely the luminance adaptation (LA) and the contrast masking (CM) phenomena. We follow the same notations as Wu et al. (2017); Fernandez et al. (2023b), and consider images that are in the range $[0, 255]^{3\times256\times256}$.

*Luminance masking.* Luminance masking refers to the phenomenon where the HVS is less sensitive to distortions in bright regions of an image. We denote by $K_{lum}$ the luminance kernel, and $x$ the input image. The local background luminance of the image is:

$$B(x)(i,j) = \frac{1}{32} \sum_{k,l\in[-2,2]^2} K_{lum}(k,l) \cdot x(i+k, j+l), \text{ with } K_{lum} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 2 & 0 & 2 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{6}$$

These luminance values are then post-processed as follows to account for the non-linear response of the HVS:

$$LA(x)(i,j) = \begin{cases} 17 \cdot \left(1 - \sqrt{\frac{B(x)(i,j)}{127} + \epsilon}\right) + 3, & B(x)(i,j) \leq 127 \\ \frac{3}{128} \cdot (B(x)(i,j) - 127) + 3, & B(x)(i,j) > 127, \end{cases} \tag{7}$$

where $\epsilon$ is a small positive value to ensure differentiability during back-propagation.

*Contrast masking.* Contrast masking refers to the phenomenon where the HVS is less sensitive to distortions in regions of high contrast. The gradient magnitude is:

$$C(x)(i,j) = \sqrt{\left(\sum_{k,l} K_X(k,l) \cdot x(i+k, j+l)\right)^2 + \left(\sum_{k,l} K_Y(k,l) \cdot x(i+k, j+l)\right)^2}, \tag{8}$$

$$\text{with } K_X = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, K_Y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \tag{9}$$

where $K_X$ and $K_Y$ are the horizontal and vertical gradient kernels, and $x$ is the input image. To account for the non-linear response of the HVS, the contrast masking values $CM(x)$ at each pixel location:

$$CM(x) = \frac{16 \cdot C(x)^{2.4}}{C(x)^2 + 26^2}. \tag{10}$$

*Heatmap generation.* The heatmap generation component of the JND model combines the luminance masking and contrast masking values to produce the JND heatmap:

$$H(x) = LA(x) + CM(x) - \gamma \cdot \min(LA(x), CM(x)), \tag{11}$$

where $\gamma$ is a parameter that controls the trade-off between luminance masking and contrast masking.

For color images, we compute the heatmap from the image's luminance. Then we repeat it over the 3 color channels but with a scaling that differs for each channel: $H_{\text{JND}} = [\alpha_R H, \alpha_G H, \alpha_B H]$, where $(\alpha_R, \alpha_G, \alpha_B) = (1, 1, 2)$, because the human eye is more sensitive to red and green than blue color shifts. This produces slightly more distortion in the blue channel.

# D Implementation details & parameters

## D.1 Architectures of the watermark embedder/extractor

We hereby detail the modeling choices and architectures of Sec. 4.1. Our models, in particular the embedder, are kept voluntarily small to be fast and usable at scale. For instance the embedder and extractor described bellow have respectively 1.1M and 96M ($\approx$ ViT-Base) parameters. Further work could explore how to build larger models with same throughput to improve the results.

In the following we consider an image $x \in \mathbb{R}^{3 \times H \times W}$ and a message $m \in \{0,1\}^{n_{\text{bits}}}$ The embedder and extractor operate at resolution $h \times w = 256 \times 256$ (see Sec. 4.1).

*Embedder.* Our goal is to embed a message in an image in a way that is imperceptible to the human eye. This task is similar in many ways to image compression or image-to-image translation. The most used architectures for this arguably come from the works of Rombach et al. (2022); Esser et al. (2021), which strike a very good balance between image quality and efficiency.

The encoder and decoder are described in Tab. 3 (we refer to the VQGAN paper (Esser et al., 2021) for more details). They mainly consist of residual blocks optionally followed by upsampling or downsampling blocks. For the encoder, we use $m = 4$ residual blocks with output channels $d = 32, 32, 32, d' = 64$, downsampling factor of 2 for the 3 first blocks (leading to a division by $f = 8$ of the edge size of the latent map), and $d_z = 4$. The decoder mirrors the encoder, and we choose $d_{\text{msg}} = n_{\text{bits}} = 32$. The Up

**Table 3** High-level architecture of the encoder and decoder of the watermark embedder $\text{emb}_\theta$. The design of the networks follows the architecture presented by Ho et al. (2020); Esser et al. (2021); Rombach et al. (2022).

| Encoder | Decoder |
|---|---|
| $x \in \mathbb{R}^{3\times H\times W}, m \in \{0,1\}^{n_{\text{bits}}}$ | $(z, z_{\text{msg}}) \in \mathbb{R}^{(d_z+d_{\text{msg}})\times 32\times 32}$ |
| Interpolation, Conv2D $\to \mathbb{R}^{d\times 256\times 256}$ | Conv2D $\to \mathbb{R}^{d'\times 32\times 32}$ |
| $m\times$ { Residual Block, Down Block} $\to \mathbb{R}^{d'\times 32\times 32}$ | Residual Block $\to \mathbb{R}^{d'\times 32\times 32}$ |
| Residual Block $\to \mathbb{R}^{d'\times 32\times 32}$ | Non-Local Block $\to \mathbb{R}^{d'\times 32\times 32}$ |
| Non-Local Block $\to \mathbb{R}^{d'\times 32\times 32}$ | Residual Block $\to \mathbb{R}^{d'\times 32\times 32}$ |
| Residual Block $\to \mathbb{R}^{d'\times 32\times 32}$ | $m\times$ { Residual Block, Up Block} $\to \mathbb{R}^{d\times 256\times 256}$ |
| GroupNorm, Swish, Conv2D $\to \mathbb{R}^{d_z\times 32\times 32}$ | GroupNorm, Swish, Conv2D $\to \mathbb{R}^{3\times 256\times 256}$ |
| $\mathcal{T}_\theta$(m), Repeat $\to \mathbb{R}^{d_{\text{msg}}\times 32\times 32}$ | TanH, Interpolation $\to [-1,1]^{3\times H\times W}$ |

**Table 4** High-level architecture of the encoder and decoder of the watermark extractor $\text{ext}_\theta$. The design of the networks follows the architecture presented by Zheng et al. (2021); Kirillov et al. (2023).

| Image encoder (ViT) | Pixel decoder (CNN) |
|---|---|
| $x \in \mathbb{R}^{3\times H\times W}$ | $z \in \mathbb{R}^{d'\times 16\times 16}$ |
| Interpolation $\to \mathbb{R}^{3\times 256\times 256}$ | $m'\times$ { Residual Block, Up Block} $\to \mathbb{R}^{d''\times 256\times 256}$ |
| Patch Embed (Conv2D), Pos. Embed $\to \mathbb{R}^{d\times 16\times 16}$ | Linear $\to \mathbb{R}^{(1+n_{\text{bits}})\times 256\times 256}$ |
| $m\times$ { Transformer Block } $\to \mathbb{R}^{d\times 16\times 16}$ | Sigmoid (optional) $\to \mathbb{R}^{(1+n_{\text{bits}})\times 256\times 256}$ |
| LayerNorm, GELU, Conv2D $\to \mathbb{R}^{d'\times 16\times 16}$ | Interpolation $\to \mathbb{R}^{(1+n_{\text{bits}})\times H\times W}$ |

block interpolates the activation map to the new size, then applies a 2D convolution with kernel size 3, stride 1 and padding 1. In particular, we choose not to use deconvolution layers (ConvTranspose2D) because of the checkerboard patterns they introduce (Odena et al., 2016). The Down block average-pools the activation map with $2 \times 2$ kernels with stride 2.

Our task does not need a bottleneck, since we are not interested in learning compressed representations. Therefore the autoencoder predicts a signal $\delta \in [-1,1]^{3\times H\times W}$ – and not directly the final image – which is added to the image with a residual layer. Another difference is that the message is embedded in the latent space of the encoder, which means that the first layer of the decoder is bigger. As a reminder, the binary message lookup table is $\mathcal{T}_\theta \in \mathbb{R}^{n_{\text{bits}}\times 2\times d_{\text{msg}}}$. Each bit of the message $m$ is mapped to the embedding $\mathcal{T}_\theta(k, m_k, \cdot) \in \mathbb{R}^{d_{\text{msg}}}$, depending on its position $k \in \{1,\dots,n_{\text{bits}}\}$ and its value $m_k \in \{0,1\}$, then repeated, which yields $z_{\text{msg}} = \text{repeat}\,(1/n_{\text{bits}} \sum_{k=1}^{n_{\text{bits}}} \mathcal{T}_\theta(k, m_k, \cdot)) \in \mathbb{R}^{d_{\text{msg}}\times 32\times 32}$.

*Extractor.* Our goal is to detect and extract the message from an image, at the pixel level. This task is similar to image segmentation. Our extractor is based on a vision transformer (ViT) (Dosovitskiy, 2020) followed by a pixel decoder, as commonly done in the literature (Kirillov et al., 2023; Zheng et al., 2021; Oquab et al., 2023).

The architecture of the extractor is detailed in Tab. 4. The encoder is a ViT which consists of a series of attention blocks to process the image's patches into a high-dimensional feature space. We use the ViT-Base architecture (86M parameters), with patch size 16 (Dosovitskiy, 2020), with $d = d' = 768$. The patch embeddings are then upscaled by the pixel decoder to predict the segmentation masks and messages for every pixel. The latter uses $m' = 3$ Up blocks which upsample (bilinearly) the activation map by factors $4, 2, 2$ respectively (total upscale factor is 16, which is the patch size used in the ViT encoder). Each Up block is followed by a Conv2D with kernel size of 3 and stride of 1, a LayerNorm, and a GELU activation. The number of channels output by the convolution is its number of input channels divided by the upsampling factor of the Up block that precedes. We obtain a latent map of shape $(d'' = d/16, 256, 256)$, which is mapped to $(1 + n_{\text{bits}})$-dimensional pixel features by a linear layer. Finally, a Sigmoid layer scales the outputs to $[0, 1]$ (this is in fact only done at inference, since the training objective implicitly applies it in PyTorch).

## D.2 Mask generation

We follow the protocol of LaMa (Suvorov et al., 2022), and generate most of our masks used during the two phases of training by adapting the authors' code github.com/advimman/lama. Figure 6 shows

some qualitative examples. More specifically, a diverse set of random masks are used:

- Box-shaped masks are defined with a margin of 10 pixels and bounding box sizes width and height randomly chosen, ranging from 30 to 100 pixels. They can be generated multiple times (1 to 3 times) per image.
- Full-image masks cover the entire image.
- Segmentation-based masks are segments corresponding to object boundaries or specific areas within an image. We use the masks of the COCO dataset, and either select the union of all masks with probability 0.5, or the union of a random number of objects.
- Irregular masks are random brush strokes, characterized by parameters such as maximum and minimum angles (up to 4 degrees), lengths and widths (ranging from 20 to 50 pixels). The brush strokes can be applied between 1 to 5 times per image.

During the first training phase (described in Sec. 4.2), we use irregular, box-shaped, full-image, and segmentation-based masks, each with a probability of 0.25. Additionally, there is a 50% chance of inverting any of the masks.

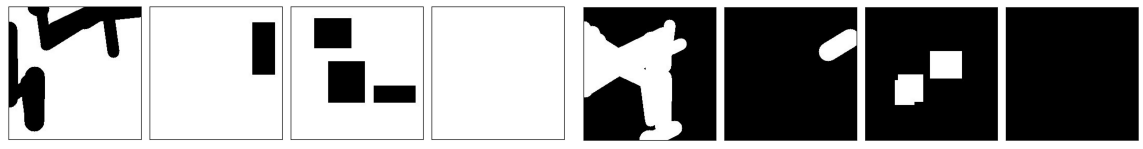
During the second training phase (described in Sec. 4.3), we use the same masks except that:

- for "box-shaped masks", with probability 0.5, instead of outputting a single mask as in the first phase, we output a random number from 1 to 3 of non overlapping rectangles;
- for "segmentation-based masks", with probability 0.5, instead of outputting a single mask as in the first phase, we output a random number from 1 to 3 of segmented objects.

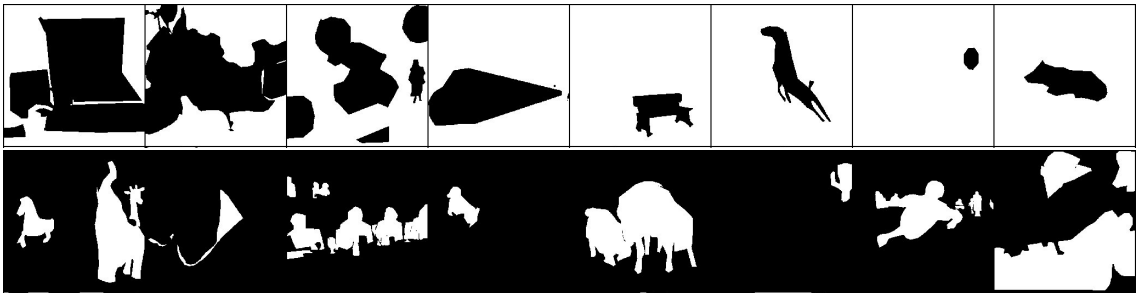
When randomly selected during this post-training phase, both types of multiple masks cannot be inverted and they are used to hide different watermarks within the same image.

### D.3 Watermarking methods

We follow the evaluation setup used by Fernandez et al. (2023a). For HiDDeN (Zhu et al., 2018), we use the model available in github.com/facebookresearch/stable_signature and modulate the output with the same JND mask. For DCTDWT, we use the implementation of github.com/ShieldMnt/invisible-watermark (the one used in Stable Diffusion). For SSL Watermark (Fernandez et al., 2022) and FNNS (Kishore et al., 2022) the watermark is embedded by optimizing the image, such that the output of a pre-trained model is close to the given key. SSL Watermark uses a model pre-trained with



**(a)** Random masks



**(b)** Segmentation masks

**Figure 6** Examples of masks used during training. Only the white areas of the image end up being watermarked. (a) Random masks (irregular, rectangles, inverted, full, null or inverted). (b) Segmentation masks created from the union of COCO's segmentation masks.

DINO (Caron et al., 2021), while FNNS uses the HiDDeN extractor used in all our experiments, and not SteganoGan (Zhang et al., 2019a) as in the original paper. We optimize the distortion image for 10 iterations, and modulate it with the same JND mask. This avoids visible artifacts and gives a PSNR comparable to our method ($\approx$ 38dB). For TrustMark (Bui et al., 2023a), we use the "C variant" with 40 bits in the official implementation github.com/adobe/trustmark, as it achieves a similar PSNR.

### D.4 Transformations

Transformations seen at training time and evaluated in Sec. 5 simulate common image processing steps. We categorize them into different groups: *valuemetric*, which change the pixel values; *geometric*, which modify the image's geometry; *splicing*, which add watermarked areas inside the image; and *inpainting*, which fill masked areas with plausible content. The geometric and valuemetric transformations are displayed in Tab. 5 and detailed in the following table:

| Transformation | Type | Parameter | Training | Evaluation |
|---|---|---|---|---|
| Brightness | Valuemetric | from torchvision | Random between 0.5 and 2.0 | 1.5 and 2.0 |
| Contrast | Valuemetric | from torchvision | Random between 0.5 and 2.0 | 1.5 and 2.0 |
| Hue | Valuemetric | from torchvision | Random between -0.1 and 0.1 | -0.1 and 0.1 |
| Saturation | Valuemetric | from torchvision | Random between 0.5 and 2.0 | 1.5 and 2.0 |
| Gaussian blur | Valuemetric | kernel size $k$ | Random odd between 3 and 17 | 3 and 17 |
| Median filter | Valuemetric | kernel size $k$ | Random odd between 3 and 7 | 3 and 7 |
| JPEG | Valuemetric | quality $Q$ | Random between 40 and 80 | 50 and 80 |
| Horizontal flip | Geometric | NA | | |
| Crop | Geometric | edge size ratio $r$ | Random between 0.33 and 1.0 | 0.33 and 0.5 |
| Resize | Geometric | edge size ratio $r$ | Random between 0.5 and 1.5 | 0.5 |
| Rotation | Geometric | angle $\alpha$ | Random between -10 and 10 | -10 and 10 |
| Perspective | Geometric | distortion scale $d$ | Random between 0.1 and 0.5 | 0.1 and 0.5 |

For crop and resize, each new edge size is selected independently, which means that the aspect ratio can change (because the extractor resizes the image). Moreover, an edge size ratio of 0.33 means that the new area of the image is $0.33^2 \approx 10\%$ times the original area. For brightness, contrast, saturation, and sharpness, the parameter is the default factor used in the PIL and Torchvision (Marcel and Rodriguez, 2010) libraries. For *splicing*, we crop a random area of the image and paste it back at the same location on the original image or on a different background image.

For evaluation (Sec. 5.3) we select some transformations from the list above and apply them with the parameters given in the table. We chose these parameters high enough to have pronounced effects on the robustness of the watermark. In practice, they are quite strong and would not not be encountered often in real-world scenarios. Additionally we evaluate the robustness against *inpainting* which is not seen at training time. We use LaMa (Suvorov et al., 2022) to modify masked areas in the image conditioned on the rest of the image.


**Figure 7** Experimental protocol for the evaluation of watermark localization as performed in Sec. 5.4.

**Table 5** Illustration of transformations evaluated in Sec. 5.

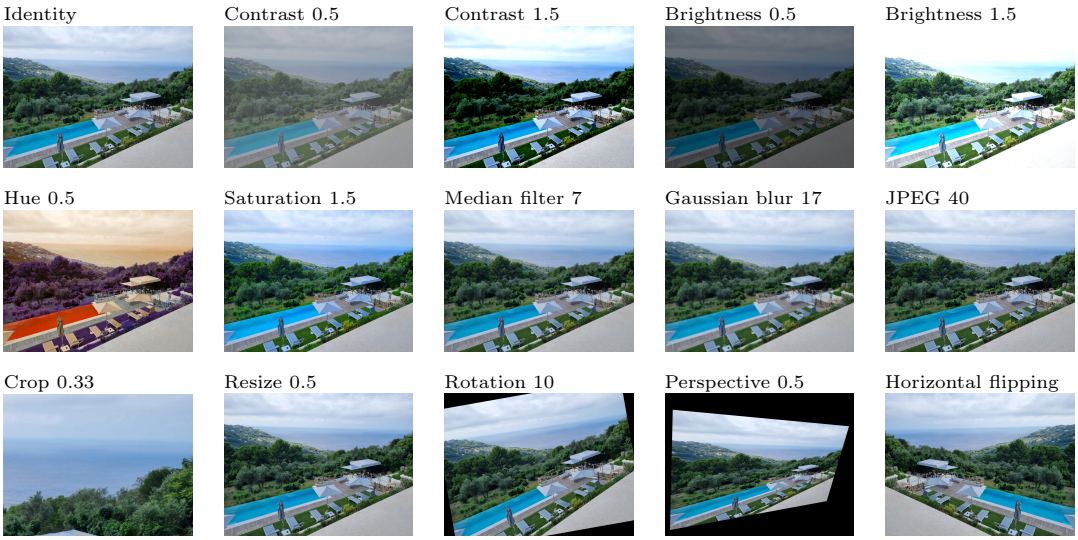



**Table 6** Detection results for WAM and generation-time watermarking methods, on 1k negative (non-watermarked), and 1k positive (watermarked), possibly edited, images. AUC refers to the Area Under the ROC curve, TPR@$10^{-2}$ is the TPR at FPR= $10^{-2}$. Stable Signature (Fernandez et al., 2023a) embeds a 48-bit message and uses the bit accuracy as score, while Tree-Ring (Wen et al., 2023) and WAM output a detection score. WAM is competitive with watermarking methods for generative models (although the latter offer noteworthy advantages).

| | AUC | | | TPR@$10^{-2}$ | | | TPR@$10^{-4}$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | Stable Sig. | Tree-Ring | WAM | Stable Sig. | Tree-Ring | WAM | Stable Sig. | Tree-Ring | WAM |
| None | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Valuemetric | 0.94 | 1.00 | 1.00 | 0.90 | 1.00 | 1.00 | 0.90 | 1.00 | 1.00 |
| Geometric | 1.00 | 0.91 | 1.00 | 0.97 | 0.56 | 1.00 | 0.87 | 0.43 | 1.00 |
| Comb. | 1.00 | 0.75 | 1.00 | 0.99 | 0.05 | 1.00 | 0.99 | 0.01 | 1.00 |
| Splicing | 0.97 | 0.72 | 1.00 | 0.90 | 0.00 | 1.00 | 0.81 | 0.00 | 0.00 |

### D.5 Localization experiments

Figure 7 gives an example of how the evaluation for watermark localization is performed in Sec. 5.4, for the "crop 0.25" augmentation. The mask used to place the watermark (second image) is the same for each image with an area covering from 10% to 100% of the image. We perform a 25% crop of the upper left corner. After that, we resize the image to its original size. Note that after this augmentation, the watermarked area still recovers the same proportion of the crop (by design, as the watermarked area was originally centered).

### D.6 Multiple Watermarks

Figure 8 shows an example of how the evaluation for multiple watermarks is performed in Sec. 5.5. After watermarking 5 areas of the image that are 10% each with different messages, the extractor detects several watermarked areas and outputs one message for each.

## E Additional Results

### E.1 Comparison with watermarking methods for generative models

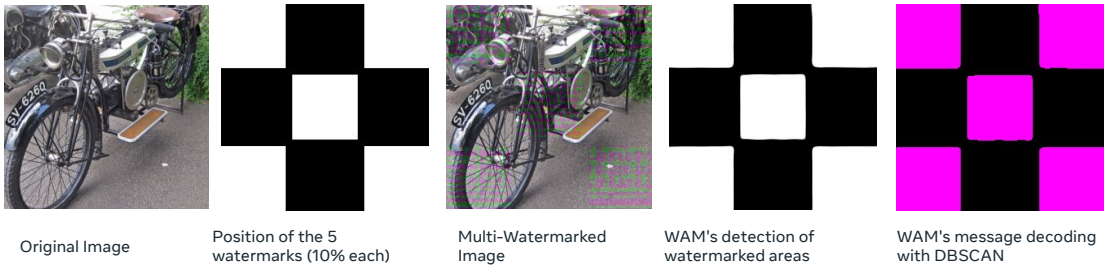
So far, we only considered general watermarking methods that apply the watermark on existing images. We now compare WAM to methods specific for LDM, which watermark at generation time. We generate 1k images from text prompts with Stable Signature (Fernandez et al., 2023a), Tree-Ring (Wen et al.,

2023) and without watermarking. We also generate a set of watermarked images with WAM from this last set. For each method, we retrieve a detection score for both the watermarked set of images and the non-watermarked images, that we use to compute the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) for each method. In the case of Stable Signature, we embed a 48-bit binary message and the score is the bit accuracy with this message, while Tree-Ring and WAM directly output a detection score.

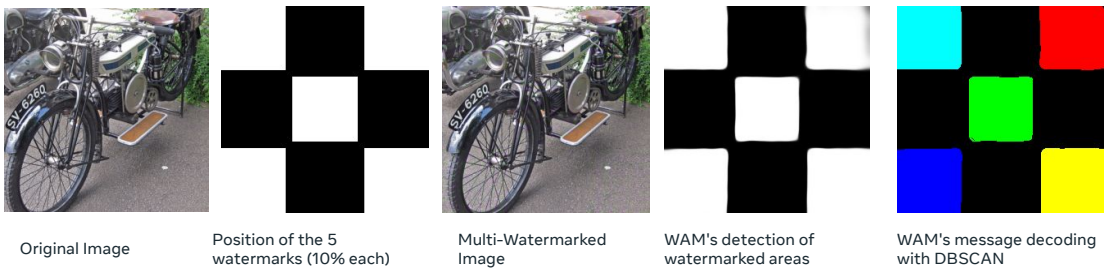
The results are shown in Tab. 6. We observe that WAM overall obtains better performance. However, the watermarking methods for generative models are still competitive and offer important advantages, namely, the possibility to open-source the model for Stable Signature, and the possibility to watermark with virtually no image degradation as well as very strong robustness against valuemetric transformations for Tree-Ring. Note that the results are not perfectly apple-to-apple since the efficiency of the extractor is different between methods: Tree-Ring requires to inverse the diffusion noise which is considerably heavier, while Stable Signature uses a very small extractor ($\approx$ 100k parameters).

## E.2 Hyper parameter search for DBSCAN

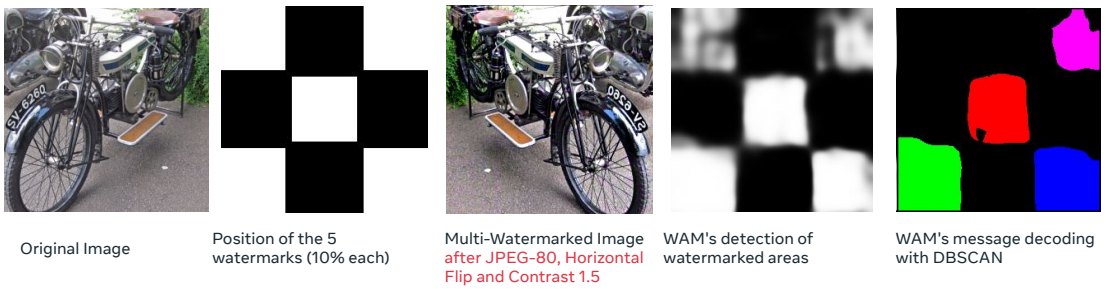
In Section 5.5, the DBSCAN algorithm is used to decode multiple messages from each image, with parameters set to $\varepsilon = 1$ and $\min_{\text{samples}} = 1000$. Under the same experimental setting, Figure 9 displays




**(a)** After the first phase of the training, evaluation without augmentation




**(b)** After the second phase of the training, evaluation without augmentation




**(c) Failure:** after the second phase of the training, with JPEG-80, horizontal flip and contrast 1.5

**Figure 8** Evaluation protocol for multiple watermark extraction as described in Sec. 5.5. The masks used to place the different watermarks (second image) are the same for each image. We then evaluate the bit accuracy across all discovered messages and their ground truth. Different colors represent different detected clusters for each image, but the color scheme is not coherent between images.

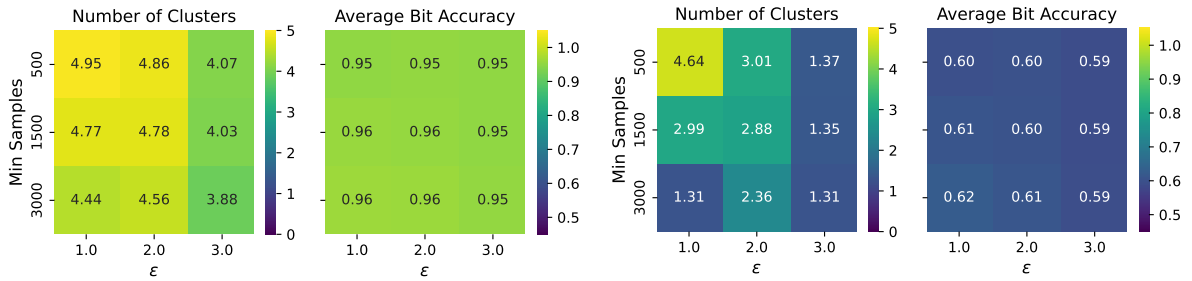
the results of a hyperparameter search conducted under two scenarios: images that underwent horizontal flipping and a contrast adjustment with a parameter of 1.5 on the left, and the same transformations followed by JPEG compression at a quality level of 80 on the right. In the absence of JPEG compression, WAM effectively recovers the correct number of messages with satisfactory bit accuracies. However, when JPEG compression is applied on top, WAM encounters difficulties. For both scenarios, smaller values of $\varepsilon$ and $\min_{\text{samples}}$ yielded the most favorable outcomes. These findings guided the selection of the aforementioned values for the primary evaluation in sec. 5.5.

### E.3 Detailed robustness results

Table 7 shows the detailed results for all the transformations. We observe that WAM handles many of them, although the performance decreases as they get stronger (e.g., for the combination). Figure 10 shows several examples of segmentation masks predicted after LaMa inpainting applied to both COCO and DIV2K datasets, consistent with the evaluation settings described in Sec. 5.3.

## F Qualitative Examples

We show examples of images from the COCO dataset in Fig. 11 and for DIV2k in Fig. 12. We observe that the watermark is imperceptible to the human eye, primarily due to the JND map used to mask the watermark in areas where the human visual system is less sensitive. With enough attention, it is however possible to see it in some images, especially in very white or very dark areas (e.g., the boat in the last row of Fig. 11) and in the fur of the animals (e.g., the wolf and the penguin of Fig. 12). We hypothesize that it comes from the fact that the JND map only accounts for the cover image where the watermark signal $\delta$ is added, and not for $\delta$ itself. However, repetitive and regular patterns are perceptible, and, when the bright or textured areas are big enough, this becomes noticeable. We believe that further work could improve the imperceptibility of the watermark, for instance by using more complex HVS models (Watson, 1993) or by using a regularization on the watermark signal to remove repetitive or structured patterns.




**(a)** After horizontal hlip and contrast adjustment of 1.5. The overall mIoU is at 85%.

**(b)** After horizontal hlip, contrast adjustment of 1.5and JPEG-80. The overall mIoU is at 59%.

**Figure 9** DBSCAN hyperparameter search in the same setting as in Sec. 5.5, under two different types of augmentations. Qualitative examples are shown in Fig. 8.

**Table 7** Full decoding results on the COCO validation set, used for the aggregated results presented in Tab. 2a of Sec 5.3. "Combination" corresponds to JPEG-80, Brightness 1.5 and Crop 50% all on the same image, and was not part of the evaluation of Sec. 5.3. TPR is for a threshold on the detection score $s_{\text{det}}$ such that FPR = 0.04%. The bit accuracy is computed as described in Eq. 2.

| | HiDDeN | | DCTDWT | | SSL | | FNNS | | TrustMark | | WAM | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TPR | Bit acc. | TPR | Bit acc. | TPR | Bit acc. | TPR | Bit acc. | TPR | Bit acc. | TPR | Bit acc. |
| None | 76.0 | 95.5 | 77.1 | 91.4 | 99.9 | 100.0 | 99.6 | 99.9 | 99.8 | 99.9 | 100.0 | 100.0 |
| Crop (20%) | 66.5 | 93.3 | 0.1 | 50.2 | 3.3 | 68.8 | 98.5 | 99.6 | 2.1 | 50.3 | 99.7 | 88.8 |
| Crop (50%) | 71.0 | 94.6 | 0.0 | 50.4 | 7.0 | 73.4 | 99.2 | 99.8 | 1.9 | 60.9 | 99.6 | 95.9 |
| Rot (10%) | 7.1 | 80.8 | 0.0 | 50.7 | 11.2 | 78.1 | 76.0 | 94.9 | 3.0 | 56.4 | 97.7 | 77.0 |
| Perspective (0.1) | 65.7 | 93.6 | 0.0 | 51.8 | 15.6 | 80.0 | 98.0 | 99.6 | 77.8 | 96.7 | 100.0 | 99.8 |
| Perspective (0.5) | 32.3 | 89.1 | 0.1 | 50.0 | 0.8 | 61.3 | 51.6 | 91.3 | 2.6 | 50.8 | 100.0 | 96.0 |
| Horizontal Flip | 0.1 | 54.3 | 0.0 | 50.2 | 32.7 | 86.2 | 0.1 | 56.2 | 99.8 | 99.9 | 100.0 | 100.0 |
| Brightness (1.5) | 85.6 | 96.9 | 16.1 | 60.9 | 90.9 | 97.7 | 99.1 | 99.7 | 83.2 | 97.1 | 100.0 | 100.0 |
| Brightness (2.0) | 83.2 | 96.2 | 0.1 | 49.4 | 67.8 | 91.9 | 97.2 | 99.0 | 58.8 | 92.2 | 100.0 | 99.9 |
| Contrast (1.5) | 74.1 | 95.4 | 20.4 | 63.4 | 92.9 | 98.2 | 98.9 | 99.7 | 77.1 | 96.2 | 100.0 | 100.0 |
| Contrast (2.0) | 67.3 | 94.3 | 0.6 | 49.4 | 66.0 | 92.4 | 97.0 | 99.4 | 52.1 | 90.8 | 100.0 | 99.9 |
| Hue (-0.1) | 9.0 | 77.4 | 29.5 | 66.7 | 98.0 | 99.4 | 96.7 | 99.2 | 99.3 | 99.9 | 100.0 | 100.0 |
| Hue (+0.1) | 19.3 | 84.0 | 59.4 | 81.2 | 97.7 | 99.3 | 98.5 | 99.5 | 99.5 | 99.9 | 100.0 | 100.0 |
| Saturation (1.5) | 80.6 | 96.2 | 21.3 | 61.7 | 99.7 | 99.9 | 99.5 | 99.9 | 99.0 | 99.8 | 100.0 | 100.0 |
| Saturation (2.0) | 81.7 | 96.5 | 0.3 | 47.4 | 98.3 | 99.5 | 99.4 | 99.8 | 96.8 | 99.5 | 100.0 | 100.0 |
| Median filter (3) | 74.6 | 94.9 | 0.9 | 53.2 | 82.8 | 96.5 | 99.4 | 99.9 | 99.7 | 99.9 | 100.0 | 100.0 |
| Median filter (7) | 23.4 | 83.0 | 0.4 | 53.0 | 20.4 | 80.7 | 61.1 | 90.2 | 99.4 | 99.9 | 100.0 | 100.0 |
| Gaussian Blur (3) | 47.1 | 88.0 | 41.7 | 77.0 | 97.2 | 99.1 | 87.3 | 95.5 | 99.8 | 99.9 | 100.0 | 100.0 |
| Gaussian Blur (17) | 0.1 | 51.2 | 0.0 | 49.8 | 3.8 | 69.6 | 0.0 | 50.5 | 98.7 | 99.8 | 100.0 | 99.8 |
| JPEG (50) | 11.2 | 77.3 | 0.0 | 49.8 | 5.1 | 72.5 | 34.5 | 86.9 | 99.1 | 99.8 | 99.9 | 99.0 |
| JPEG (80) | 32.8 | 86.8 | 0.1 | 50.5 | 66.1 | 92.6 | 80.5 | 95.4 | 99.6 | 99.9 | 100.0 | 99.9 |
| Proportion (10%) | 0.8 | 65.9 | 1.1 | 56.9 | 0.6 | 61.8 | 36.3 | 88.2 | 2.0 | 58.6 | 99.9 | 94.2 |
| Collage (10%) | 0.9 | 70.1 | 0.5 | 62.8 | 0.1 | 55.9 | 41.1 | 88.7 | 1.3 | 55.7 | 100.0 | 96.5 |
| Combination | 44.7 | 88.7 | 0.0 | 49.9 | 1.2 | 62.8 | 87.7 | 96.1 | 1.7 | 58.8 | 99.2 | 87.2 |

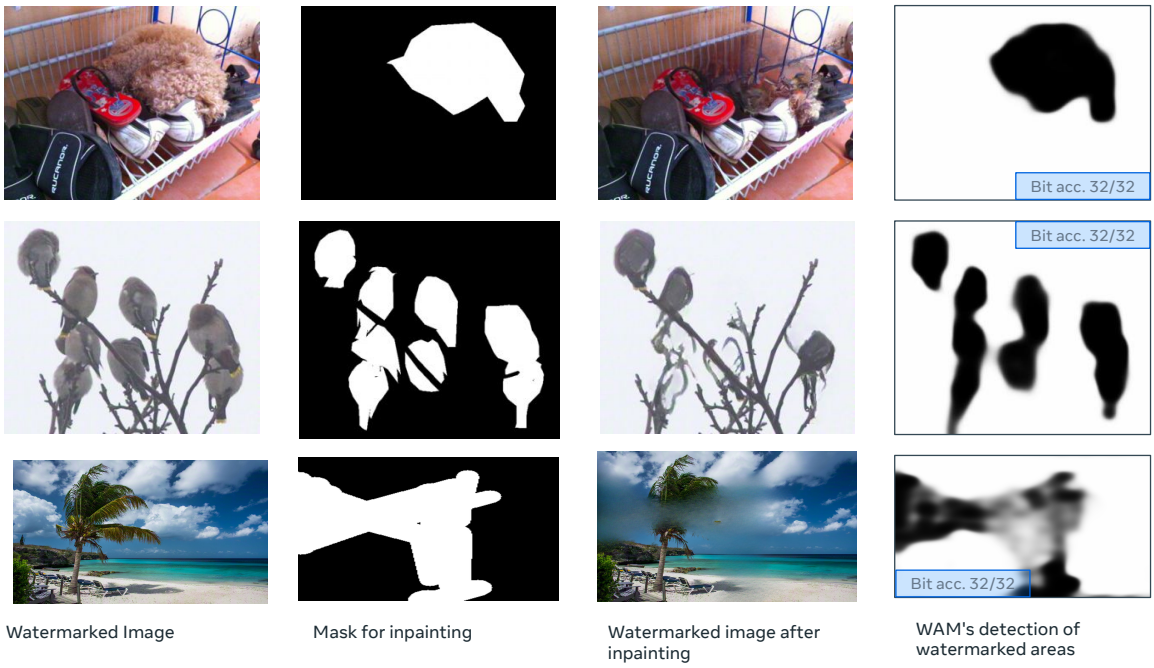



**Figure 10** Examples of WAM's detection after inpainting with LaMa (Suvorov et al., 2022), with the experimental set-up detailed in Sec. 5.3. The first two lines are with images from the validation set of COCO (using the union of the segmentation masks), while the third line is with an image from the DIV2k dataset, with a random mask.

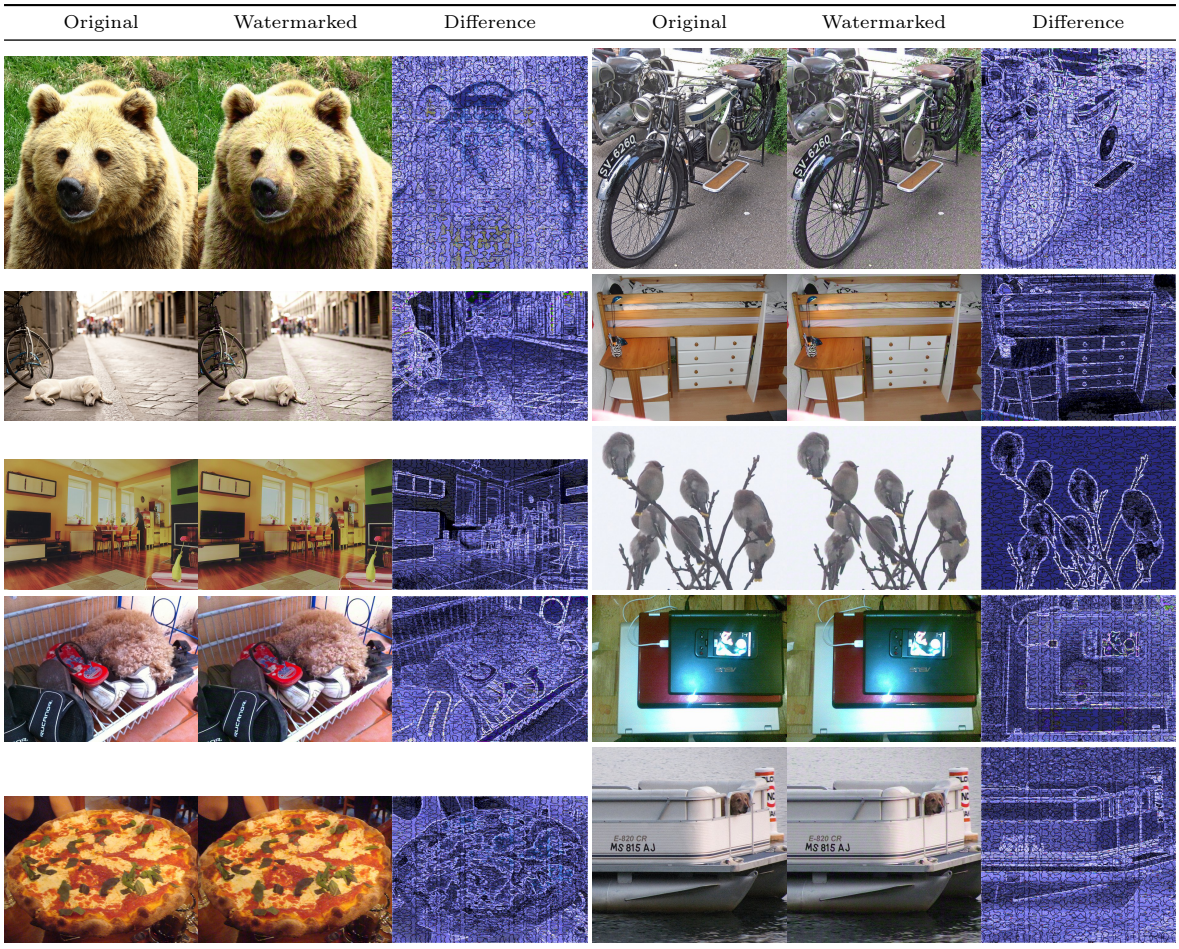



**Figure 11** Qualitative results on the validation set of MS-COCO, at various resolutions and for a 32-bits message. The difference image is displayed as $10 \times \text{abs}(x_\text{m} - x)$.

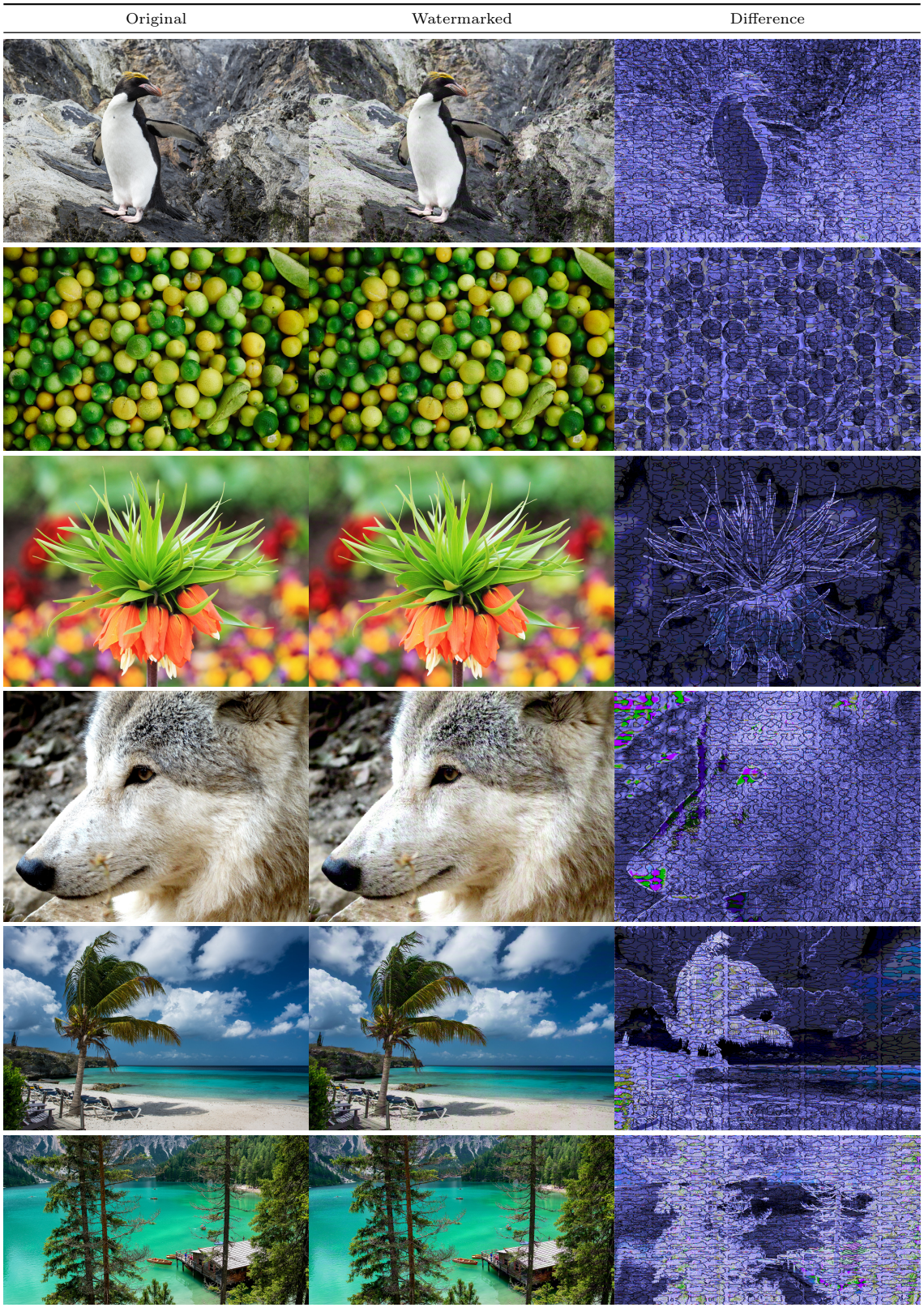



**Figure 12** Qualitative results on images from DIV2k, at higher resolution and for a 32-bits message. The difference image is displayed as $10 \times \text{abs}(x_\text{m} - x)$.