BitMark for Infinity: Watermarking Bitwise Autoregressive Image Generative Models

Louis Kerner, Michel Meintz, Bihe Zhao, Franziska Boenisch, Adam Dziedzic
{louis.kerner, michel.meintz, bihe.zhao, boenisch, adam.dziedzic}@cispa.de
CISPA Helmholtz Center for Information Security

Abstract

State-of-the-art text-to-image models like *Infinity* generate photorealistic images at an unprecedented speed. These models operate in a bitwise autoregressive manner over a discrete set of tokens that is practically infinite in size. However, their impressive generative power comes with a growing risk: as their outputs increasingly populate the Internet, they are likely to be scraped and reused as training data—potentially by the very same models. This phenomenon has been shown to lead to model collapse, where repeated training on generated content, especially from the models' own previous versions, causes a gradual degradation in performance. A promising mitigation strategy is watermarking, which embeds human-imperceptible yet detectable signals into generated images—enabling the identification of generated content. In this work, we introduce BitMark, a robust bitwise watermarking framework for Infinity. Our method embeds a watermark directly at the bit level of the token stream across multiple scales (also referred to as resolutions) during Infinity's image generation process. Our bitwise watermark subtly influences the bits to preserve visual fidelity and generation speed while remaining robust against a spectrum of removal techniques. Furthermore, it exhibits high radioactivity, i.e., when watermarked generated images are used to train another image generative model, this second model's outputs will also carry the watermark. The radioactive traces remain detectable even when only *fine-tuning* diffusion or image autoregressive models on images watermarked with our BitMark. Overall, our approach provides a principled step toward preventing model collapse in image generative models by enabling reliable detection of generated outputs.

1 Introduction

The image generation capabilities of large scale models [7, 10, 21, 24] have improved tremendously. These models can now generate at an unprecedented speed (of less than a second) high-quality photorealistic outputs, which are often indistinguishable from real images [10]. One of the current state-of-the-art image generative models is Infinity [10], which operates in a bitwise autoregressive manner [32, 37] over an almost infinite number of tokens and yields high-resolution images. This performance, however, comes at the expense of a training process that is highly time-, cost- and most of all data-intensive [23, 27]. The required training data are often scraped from the Internet [11, 22, 30]. This practice carries risks: with the ever-growing amount of generated data on the internet, the models get increasingly trained on *generated* data rather than *natural* data. The iterative training of models on their own generated data has been shown to degrade performance [1, 31], which is referred to as *model collapse*. The process also leads to encoding the current models' mistakes, biases, and unfairnesses into its future versions [36]. From the perspective of model owners, especially big companies such as OpenAI, Google, or ByteDance, who invest in large scale proprietary models that are exposed via public APIs, this poses a significant threat. To prevent the possible degradation of model quality, it is crucial for them to identify *their own* generated content.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

A promising way to address the above problems is watermarking [4, 8, 9, 12, 20, 34, 43]. By embedding human-imperceptible, yet detectable signals into generated images, a model owner can automatically identify and filter out their model's previously generated content when collecting training data for future versions. To prevent model collapse, watermarks should be radioactive [6, 26, 29]—that is, they should persist across model lineages, so that models trained on watermarked outputs also reproduce the watermark. This is crucial when third parties fine-tune models on such outputs and use them to generate new content, which may later be scraped and reused by the original model—causing it to (unknowingly) train on data derived from its own outputs. While watermarks have been extensively studied in diffusion models (DMs) [8, 9, 34, 43], no watermarking methods have been proposed for image autoregressive models such as Infinity, so far. Additionally, existing DM watermarks have been shown to lack radioactiveness [6], motivating the need for a new approach.

To close this gap and provide a robust way of detecting images generated by state-of-the-art models like Infinity [10], we propose the BitMark watermark that aligns with Infinity's bitwise image generation process and operates directly on a *bit-level*. We show that this bit-level design is crucial in image autoregressive models, because watermarks at the token level—a common practice in autoregressive language models [14, 41]—are suboptimal. This is because unlike language models, which operate on discrete tokens and yield stable token representations, image models must decode to and re-encode from a continuous image space, introducing discrepancies between the original and recovered tokens. These discrepancies can decrease the watermark signal. Yet, we find that the discrepancies are substantially smaller at the bit level (Figure 1), motivating our bitwise approach to obtain a robust watermark. To embed the watermark, BitMark first divides all possible *n-bit sequences* into *green* and *red* lists, where the green list is supposed to contain sequences indicating the presence of the watermark. The embedding process of the watermark aims at completing one of the green-list bit sequences for every next bit by slightly increasing each logit of the preferred bit that would complete the sequence. This eventually causes the generated image to contain more sequences from the green list than natural images, which can be used for detection. Concretely, the detection of the watermark is performed on re-encoded images and based on the count of bit sequences from the green list.

Our watermark exhibits many desirable properties. Most importantly, the generated images are of the same quality as images generated without the embedded watermark. Moreover, the watermark embedding has a negligible impact on the speed of the image generation. Also, the corresponding watermark detection method is fast and reliable. As in natural images all bit sequences are equally likely to occur while our watermarked images have the bias towards sequences from the green list, natural images are extremely unlikely to be detected as watermarked. Furthermore, working on the *bit-level* makes BitMark robust against a wide range of attacks, such as adding Gaussian noise or blur to the image, applying color jitter, random crop, or JPEG compression. Our watermark is also highly resilient against dedicated watermark-removal techniques, for example, the watermark in the sand attack [38] or CtrlRegen [18]. Finally, we show that BitMark is radioactive. This means that when we use the watermarked images to train or fine-tune other image generative models, our watermark is still present in the outputs from these other models. We demonstrate all these properties through our extensive experimental evaluation.

In summary we make the following contributions:

- We propose BitMark—the first watermarking scheme for Infinity. BitMark leverages Infinity's unique structure and bitwise image generation process to embed the watermark on the level of bits, which preserves the high quality of the models' outputs and its inference speed. Our approach can be applied at inference time and does not require additional retraining.

- We thoroughly evaluate BitMark against a wide variety of standard and dedicated watermark removal attacks and show its robustness to all of them. Specifically, BitMark is also resilient to the watermark in the sand attack [38], which is targeted at removing the watermarks from the generated content. Finally, BitMark is also robust against a Bit-Flipper attack carefully crafted against it.

- We also assess BitMark's *radioactivity* and show that when other image-generative models are trained or fine-tuned on images watermarked with our method, these new models' outputs will also contain our watermark. This makes BitMark highly suitable to identify generated content and contributes to preventing model collapse due to iterative training a model on its generated images.

2 Background

We consider the image autoregressive models, with the focus on the state-of-the-art text-to-image Infinity model that operates in a autoregressive manner, and the techniques used to robustly watermark image generative models.

Visual AutoRegressive Modeling (VAR) [32]. VAR leverages discrete quantization to predict tokens as residual maps, scaling from an initial low scale (coarse-grain) to a final scale (fine-grain) resolution. VAR further leverages the transformer architecture [33] across all tokens to significantly improve generation quality. VAR was further extended from class-to-image to the **Infinity** model [10], which is a text-to-image autoregressive model for high-resolution image synthesis, using next-scale prediction similar to VAR [32]. Instead of leveraging index-wise tokens for quantization like VAR [32], Infinity uses bitwise token prediction with an implicit codebook and applies binary spherical quantization [42] to select an entry from the codebook. This results in more efficient computation and further enables the usage of large codebook sizes, *e.g.,* 2^{64}. Random bit flipping further empowers the model to self-correct on each scale, leveling out errors stemming from earlier tokens. The visual autoregressive paradigm instantiated by Infinity achieves the state-of-the-art (SOTA) in image generation.

Watermarking Images. Watermarking entails embedding a message, pattern, or information into an image, that can be detected. The embedded information should be hard to remove and can be also used to verify the provenance of the image. In general, the watermarking methods can be either *static* or *learning-based*. In static watermark embedding, a transform is directly applied to an image [20]. Learning-based methods are generally based on three key components: a watermark (*w*), an encoder (*E*) and a decoder (*D*). The encoder is trained to embed the watermark into the image, while not impacting its quality and the decoder is trained to reconstruct the watermark given a watermarked image to perform detection. An important aspect of watermarks is the trade-off between their detectability and their impact on the quality of the data, which was thoroughly analyzed in [35].

Watermarking Diffusion Models. Learning-based watermarks represent the SOTA for Diffusion Models [8, 9, 34, 43]. They target different points of the image generation process, such as the initial data used to train the model [43], noise prediction [9, 34], or the decoding process [8]. The Recipe method [43] embeds a binary signature into training data using a pretrained encoder-decoder and trains an explicit decoder to extract this watermark from outputs. In contrast, Tree-Ring [34] embeds a structured pattern in the initial noise vector, designed in the Fourier space, achieving robustness to transformations by enabling detection through inversion of the diffusion process. On the other hand, the PRC Watermarking [9] generates the initial noise vector from a pseudorandom error-correcting code, allowing deterministic reconstruction of the noise for watermark retrieval. Finally, Stable Signature [8] fine-tunes the latent decoder of latent diffusion models (LDMs) to insert a recoverable watermark using a pretrained extractor, modifying the generation process at the decoding stage. However, the watermarking techniques for diffusion models are not radioactive [6], hence, do not prevent the model collapse.

Watermarking Autoregressive Models. As of now, all the efforts on watermarking autoregressive models have been directed towards creating watermarks for Large Language Models (LLMs) [14]. The KGW watermark [14] uses the concept of a green and red token lists, where generation is softly nudged to predict tokens from the green list rather than from the red list. The lists are generated dynamically, based on the hash value of the previously predicted token(s). For detection, the LLM used to generate the text is not needed. Instead, a suspect text is assessed based on how often tokens from the green list appear, compared to tokens from the red list. At the end of detection a statistical test is performed. KGW was extended to the Unigram-Watermark [41], which simplifies the design by introducing a fixed global green and red lists. The simplified approached allowed for guaranteed generation quality, provable correctness in watermark detection, and robustness against text editing and paraphrasing. In this work, we we build on these watermarking schemes originally developed for large language models (LLMs) to address the lack of watermarking techniques for image autoregressive models, with focus on the state-of-the-art Infinity model.

Private Watermarking. In our watermark, we follow the *Private Watermarking* from KGW [14]. For this end, the generative model is kept behind a secure API with a secret key $\mathcal{K}$. This secret key $\mathcal{K}$ is used as input to a pseudorandom function (PRF) F, with the h previously generated tokens, to compute the red-green list split for the next predicted token. Here the context window h is a hyperparameter, which trades robustness with privacy. For small context windows h (*e.g.,* 1 or 2) the

watermark is robust against removal attacks but an attacker could tabulate the frequency of token pairs and brute-force knowledge about the green list, reducing the privacy of the method. For larger values of h (*e.g.,* 5), changing a single token has an impact on the next h downstream tokens, potentially flipping them to the red set. The privacy of this approach can also be boosted by using k different private keys and choosing one randomly during generation.

Watermark Robustness. To reliably perform image provenance, a watermark must be robust to removal attacks. There is a wide range of these kinds of attacks that have been proposed in the literature [2, 18, 25]. For example, the scrubbing attacks aim at removing the watermarks from the outputs of a generative model, which can be done by producing a valid *paraphrase* that is detected as non-watermarked text [13] or image [3]. Zhang et al. [38] claim that under the assumption that both a quality oracle (that can evaluate whether a candidate output is a high-quality response to a prompt) and a perturbation oracle (which can modify an output with a nontrivial probability of maintaining quality) exist, it is feasible to remove existing watermarks in generative models without quality degradation of the watermarked content. We evaluate our BitMark against a wide spectrum of attacks and show that our method is robust against them.

3 BitMark Method

The Infinity architecture [10] follows an autoregressive generation paradigm, but instead of directly predicting tokens like other autoregressive models, such as VAR [32] or RAR [37], it predicts bits. We leverage this prediction scheme to design a dedicated watermark. Our *BitMark* watermark is embedded across all scales (resolutions) of Infinity, which subtly shift the bitwise representation towards a higher frequency of detectable patterns to resist watermark removal and manipulation.

Problem Formulation and Desiderata. Our goal is to embed a watermark during Infinity's image generation process and subsequently detect it by analyzing the generated image. BitMark targets the prevention of model collapse. In this setup, only the model owner is concerned with the watermark and wants to ensure that their own data are not used to train their next models. Thus, we aim at four essential properties for our watermarking scheme to prevent model collapse: (1) ***Negligible Impact on the Generation.*** To maintain the utility of the generative model, the watermark should not degrade the quality of the generated images, while also maintaining a fast inference speed. (2) ***Reliable Verification.*** The model owner should be able to detect the watermarked images and verify the confidence of the extracted watermark. The model owner encodes the image to their own space and it is only relevant to the model owner if the watermark is present there or not. The encoder part of the model is also kept private so only the model owner can verify the watermark. The natural images should not be detected as containing the watermark. At the scale of the large models trained on trillions of samples, a small number of false positives is acceptable as long as it does not exceeds a certain threshold (*e.g.,* 1%), which can also prevent model poisoning [40]. (3) ***Robustness.*** The watermark should be robust not only against conventional attacks, such as Gaussian noise or JPEG compression, but also against strong dedicated watermark removal attacks such as CtrlRegen [18] or watermarks in the sand [38]. (4) ***Radioactivity.*** If other generative models are trained on watermarked images, images generated by these new models should also contain our watermarks [6, 26, 29].

Threat Model. We assume two parties: a *model owner* providing an image generation API where our BitMark is deployed, and an *attacker* who aims to fool the model owner to train on their own generated images and to prevent the use of large-scale natural image datasets for model training. We assume that the ***model owner*** follows the *Private Watermarking* from [14] and keeps the generative model behind a secure API. The model owner wants to watermark their own images in order not to train on them. The watermark should also be radioactive so that if any other party fine-tunes their own models on the watermarked data, the fine-tuned model's outputs are watermarked, as well. Further, we assume that the watermarking technique is private and verifiable by the model owner. On the other hand, the ***attacker*** might attempt to remove the embedded watermark. For instance, competitors might still want to remove the watermark to harm the model owner. We assume that the attacker has black-box access to the model, *i.e.,* they can provide inputs and observe the corresponding outputs. The attacker is allowed to make arbitrary modifications to the generated image, including but not limited to: applying image transformations (*e.g.,* noising, blurring, cropping, etc.) or specific watermark removal techniques [18, 38]. We define adversarial behavior as any attempt to remove the watermark in the generated image while preserving high perceptual quality and semantic content. Attacks that destroy the usefulness or visual coherence of the image (*e.g.,* replacing it with noise) are not considered meaningful threats.

3.1 Motivation for the Bitwise Watermark

We identify an **inherent challenge** in watermarking the images generated by autoregressive models like Infinity, stemming from imperfections in image encoding and decoding. Specifically, the model first generates a sequence of tokens, which are decoded into an image. However, when the exact same image is re-encoded and tokenized, it produces a different set of tokens. This also holds partly true for bits that underlie the tokens. The observed behavior does not occur in the autoregressive LLMs, where the tokens produced by decoding and then re-encoding a generated text match *exactly*. We quantify the bit and token discrepancies in Figure 1. In both cases, bits and tokens are generated by Infinity and then reconstructed by its auto-encoder. For Infinity, the highest token overlap occurs at the initial and final resolutions (scales). This is likely because the early stages capture coarse structural features, while the final resolution refines the image, leading to fewer changes in the token sequence. We observe a consistent average token overlap of approximately 2.385% and the bit overlap of 77.432%. Given the significantly large codebook size of 2^n (where $n \in \{32, 48, 64\}$) in Infinity, we observe that the bit-wise information loss is much smaller than the token-wise loss. This motivates our work to design a watermark that biases the generated sequences of tokens on the *bit-level* instead of on the *token-level*.

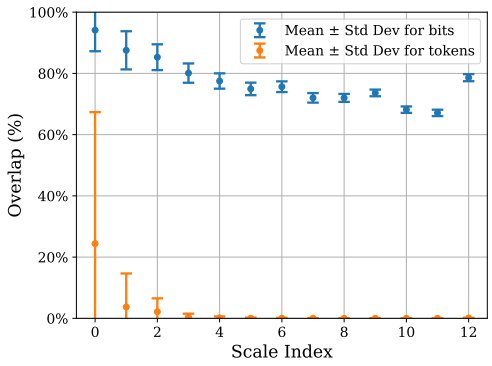

Figure 1: **Bit and token overlaps in Infinity.**

3.2 Embedding our BitMark Watermark

Next, we detail how BitMark is embedded. The embedding protocol is directly applied on the *bit-level* in the autoregressive generation process of the Infinity model. We present the full procedure in Algorithm 1. First, we separate the list of all bit sequences of length n: $S = \{0, 1\}^n$ into a red list R and a green list G. Intuitively, we softly nudge the model to generate bit sequences of length n that are part of the green list G. For each next bit prediction b_j, if the bit can complete one of the bit sequences from G, we add a small bias δ to the logit $l_j^{(b_j)}$ of the bit b_j:

$$p_j = \begin{cases} \frac{\exp(l_j^{(b_j)}+\delta)}{\exp(l_j^{(\neg b_j)})+\exp(l_j^{(b_j)}+\delta)}, & pre + b_j \in G \\ \frac{\exp(l_j^{(\neg b_j)})}{\exp(l_j^{(\neg b_j)})+\exp(l_j^{(b_j)}+\delta)}, & pre + \neg b_j \in R, \end{cases} \tag{1}$$

where $pre = (b_{j-(n-1)}, \ldots, b_{j-1})$ is the prefix bit sequence of length $(n-1)$, *i.e.,* the sequence of bits preceding b_j, $+$ is the bit concatenation operation, and l_j is the vector of logits for the bit b_j. We apply this procedure to all bits and across all resolutions (scales) that are generated by the model. Thus, as many bit sequences as possible are biased towards G and the final green fraction, *i.e.,* the number of sequences part of the green set, of the generated bits is significantly above the threshold of 50%, the standard fraction for clean images (see Appendix D). By biasing the logit for the preferred bit, BitMark has negligible impact on image quality, as mostly high entropy bits are flipped (see Table 1). Since high entropy bits have sufficiently small difference in the probability of assigning them to value 0 or 1, the relatively small added bias influences the final value. Additionally, biasing only the logits makes the watermarking process difficult to revert, as it is difficult to know for an attacker which bits are of high entropy and can be flipped to the red list R without degrading the final image quality.

3.3 Watermark Detection

Watermark detection is performed directly on the bits of encoded images. All that is required is knowledge about the green and red lists, as well as access to the tokenizer. Intuitively, we count how often each sequence of bits in the encoded image follows the green list and compare how the frequency differs from non-watermarked images.

Detecting the Watermark. The detection protocol follows the standard encoding from Infinity and is detailed in Algorithm 2. First we obtain the bit representations for each resolution. We then use our knowledge of G and R to test how often the bit sequences from the red and green lists appear and count them. This results in the knowledge of how many green sequences appear in the encoded image, which is random for every non-watermarked image, *i.e.,* 50% (see Appendix D).

Robust Statistical Testing. Given the final number of green and red sequences detected in the encoded image, we perform a statistical test based on the null hypothesis $\mathcal{H}_0$: *The sequence of bits is generated with no knowledge about the green and red lists.* Intuitively, this assumes that a given image is not watermarked. A watermarked image consists of γT green and $(1-\gamma)T$ red bit sequences, where γ determines the size of the green list and T is the total number of bits in the encoded image that our watermark can influence. Per a given token t with m bits, BitMark influences $k = m - (n-1)$ bits, since we start watermarking from the first n bit pattern. Thus, overall we have $T = \sum_{i=1}^{K} r_i \cdot k$, where r_i is number of tokens per resolution i. A natural image is expected to violate the red list rule with the half of its bits, while a watermarked image violates this rule less frequently. If the null hypothesis is true, then the number of green list bit sequences denoted C has the expected value of γT and variance $\gamma(1-\gamma)T$. To ensure statistical validity, we follow [14] and use a z-test. We reject the null hypothesis and detect the watermark if z is below or above a chosen threshold:

$$z = (C - \gamma T)/\sqrt{T\gamma(1-\gamma)}. \tag{2}$$

Algorithm 1 Watermark Embedding

Inputs: autoregressive model f_θ with parameters θ, green set G, red list R, image decoder $\mathcal{D}$.
Hyperparameters: steps K (number of resolutions), resolutions $(h_i, w_i)_{i=1}^{K}$, the number of tokens for resolution i is r_i, constant δ added to the logits of the $green$ list bits, n - the length of the bit sequences in the lists (G and R), m the number of bits per token.

$e = \text{init}()$
for $i = 1, \ldots, K$ **do**
  $(l_1, \ldots, l_{r_i \cdot m}) = f_\theta(e, h_i, w_i)$
  for $t = 1, \ldots, r_i$ **do**
    for $j = n, \ldots, m$ **do**
      $c = (t-1) \cdot m + j$ //counter
      $p_c = \text{Bias}(G, l_c, \delta)$
      $s_c = \text{Softmax}(p_c)$
      $b_c = \text{Sample}(s_c)$
  $u_i = (b_1, \ldots, b_{r_i \cdot m})$
  $z_i = \text{lookup}(u_i)$
  $z_i = \text{interpolate}(z_i, h_K, w_K)$
  $e = e + \phi_i(z_i)$
$im = \mathcal{D}(e)$
Return: watermarked image im

Algorithm 2 Watermark Detection

Inputs: raw image im, green list G, red list R, image encoder $\mathcal{E}$, quantizer $\mathcal{Q}$
Hyperparameters: steps K (number of resolutions), resolutions $(h_i, w_i)_{i=1}^{K}$, the number of bits for resolution i is r_i, n - the length of the bit vector.

$e = \mathcal{E}(im)$
$C = 0$
for $i = 1, \ldots, K$ **do**
  $u_i = \mathcal{Q}(\text{interpolate}(e, h_i, w_i))$
  $u_i = (b_1, \ldots, b_{r_i})$
  $C = \text{Count}((b_1, \ldots, b_{r_i}), G)$
  $z_i = \text{lookup}(u_i)$
  $z_i = \text{interpolate}(z_i, h_K, w_K)$
  $e = e - \phi_i(z_i)$
Return: StatisticalTest($\mathcal{H}_0, (C)$)

4 Empirical Evaluation

Experimental Setup. We employ the Infinity-2B checkpoints and code [10] to evaluate BitMark across different watermarking strengths and against a wide variety of attacks. [1] The Infinity-2B model generates 1024×1024 pixel RGB-images and we choose the visual-tokenizer with a vocabulary size of 2^{32} per token. We follow [10] in the choice of hyper-parameters and use a classifier-free-guidance of 4, with a generation over 13 scales, following the scale schedule for an aspect ratio of 1:1. Hence, we utilize the best performing model, tokenizer and hyperparameters reported in [10]. We ablate the choice of the green, red list split for our experiments in Appendix B.

[1] Infinity-2B was released six months ago, at the time of submission, allowing sufficient time for experimentation. Note that the most powerful Infinity-20B model (that we are aware of) is kept private (while the Infinity-8B was released less than 3 months ago before the submission, limiting our ability to test it).

4.1 BitMark Introduces Detectable Watermarks while Maintaining Generation Performance

We demonstrate that our BitMark preserves the utility of the Infinity model with negligible impact on the image quality and the inference speed.

High Image Quality. Table 1 presents a quantitative evaluation of image generation quality for different watermarking strengths dependent on the logit bias parameter δ. We generated 10,000 images based on prompts of the MS-COCO 2014 [16] validation dataset. We report the FID, KID, measuring the similarity of generated images and natural images in feature space, and the CLIP Score, which measures the alignment between prompt and image, for the varying watermark strengths δ. The results show that, for $\delta \leq 3$, BitMark has a negligible impact on image quality (as measured by FID and KID) and minimally affects the prompt-following capabilities of the underlying model, as indicated by the CLIP Score. We also present a qualitative analysis of image quality in Appendix E.2, which shows images generated for different δ. While there is an inherent trade-off in terms of watermark strength an image quality, we choose a small $\delta = 2$ for our watermarks to obtain minimal impact on the generation while having a significantly strong watermark.

Table 1: **Image quality mean (std) vs δ.**

δ	FID↓	KID ($\times 10^{-2}$)↓	CLIP Score↑
0	33.36	1.42 (0.12)	**31.16 (0.28)**
1	32.61	1.38 (0.13)	**31.16 (0.28)**
2	31.05	1.26 (0.12)	31.15 (0.28)
3	**29.61**	**1.03 (0.08)**	31.03 (0.27)
4	42.98	1.78 (0.08)	29.78 (0.32)
5	127.44	11.16 (0.34)	26.13 (0.40)

Fast Watermark Generation and Verification. For a watermark to be reasonably applicable in a real world setting, the watermark should have negligible impact on image generation speed [8, 34, 43]. We thus analyze the practicality of applying BitMark in a standard setting and its impact on generation speed of the Infinity model. Next to the generation performance, the speed of watermark detection has a strong impact on the possible applications, especially in a setting where we want to quickly detect data generated by Infinity. The quantitative analysis of BitMark in Appendix E.3 shows that generating a watermarked image requires at most one-tenth the time longer than the standard image generation. Verification, *i.e.,* encoding an image into its bit representation and computing the z-score, takes less than 0.5 seconds per image (usually around 5X faster than the image generation).

4.2 BitMark is Robust Against a Wide Range of Attacks

Our results demonstrate that BitMark is extremely robust against a wide range of attacks, including conventional attacks, reconstruction attacks, watermarks in the sand attack, and even our own adaptive *bit-flipping* attack, which we carefully design with the aim of exploiting knowledge of our watermark method to remove BitMark.

Robustness against Conventional Attacks. We perform a wide range of conventional attacks, namely: 1) **Noise:** Adds Gaussian noise with a std of 0.1 to the image, 2) **Blur** Noise attack + application of a Gaussian blur with a kernel size of 7, 3) **Color**: Color jitter with a random hue of 0.3, saturation scaling of 3.0 and contrast of 3.0, 4) **Crop:** Random crop to 70% 5) **Rotation:** Random rotation between different degrees in Table 2 and 0° and 180° in Table 3. 5) **JPEG:** 25% JPEG compression. Table 3 shows that our approach achieves a high TPR@1%FPR for most conventional attacks. Furthermore, BitMark achieves a high TPR@1%FPR against most attacks even for a low watermarking strength $\delta = 1$. Although BitMark has slightly lower performance towards arbitrary rotation angles, the watermark is still highly detectable and it still obtains high TPR@1%FPR within a given rotation threshold. We report TPR@1%FPR and mean (std) of z for different degrees of rotations for 1,000 images in Appendix E.5. Notably, it achieves as high as 99.9% TPR@1%FPR when the rotation range is between -20° and 20° Table 2. Given the strength of off-the-shelf tools for rotation estimation [19], it is highly achievable to rotate the image back into a tolerable range of rotations during the pre-processing stage in the model training process, thus ensuring that the watermark is robust against image rotations.

Table 2: **Rotation attack.**

Degrees	TPR@1%FPR	z (std)
-20, 20	99.9	14.87 (9.77)
-30, 30	87.4	11.78 (9.55)
-40, 40	70.0	9.66 (9.38)
-50, 50	56.8	8.21 (8.99)

Robustness against Reconstruction Attacks. We also evaluate our proposed method against two state-of-the-art reconstruction attacks (shown as the last two columns in Table 3): 1) **SD2.1-VAE:** Encoding and decoding with the Stable Diffusion 2.1 autoencoder [24], 2) **CtrlRegen+:** Application of controllable regeneration from noise in the latent space following [18]. For the reconstruction

Table 3: **BitMark is robust against watermark removal attacks.** We report the TPR@1%FPR (%) for the different attacks.

δ	Conventional Attacks							Reconstruction Attacks	
	None	Noise	Blur	Color	Crop	Rotation	JPEG	SD2.1-VAE	CtrlRegen+
1	100.0	97.8	99.2	99.3	68.5	14.5	100.0	100.0	61.9
2	100.0	99.6	99.9	99.8	98.8	20.1	100.0	100.0	91.6
3	100.0	99.7	99.9	99.9	99.8	23.8	100.0	100.0	97.0
4	100.0	99.9	99.9	100.0	99.9	29.2	100.0	100.0	99.0
5	100.0	100.0	100.0	99.9	100.0	34.3	100.0	100.0	99.8

Table 4: **BitMark is robust against the watermarks in the sand attack** [38]. We present the detection results of our watermarking scheme ($\delta = 2$) before and after the attack. We use 200 iterations, and generate 100 candidates during each iteration.

Setting	Accuracy (%)	AUC (%)	TPR@1%FPR (%)	Green Fraction (%)	z	z Threshold @1%FPR
Original Image	-	-	-	49.7	0.2	-
Watermarked Image, before the attack	100.0	100.0	100.0	57.6	87.8	59.6
Watermarked Image, after the attack	98.5	99.8	89.0	50.9	10.8	6.8

attacks, we find that the strength $\delta = 2$ can withstand the CtrlRegen+ reconstruction attack, which removes watermarks such as TreeRing [34] or StableSignature [8].

Robustness against Watermarks in the Sand Attack. We also evaluate the robustness of our watermarking scheme against the attack proposed in [38], namely, the watermarks in the sand attack. The attack leverages the CLIP Score as a quality oracle, the stable-diffusion-2-base [21] as a perturbation oracle, and iteratively modifies the watermarked image to remove the watermark. During each iteration, the attack produces potential candidates by inpainting randomly masked regions in the image. The attack successfully replaces most visual details in the original watermarked image while selecting the candidate with the best CLIP Score. As a result, many watermarks for diffusion models (*e.g.,* [8]) can be removed through this attack. However, results in Table 4 show that our proposed method achieves a significant detection rate with TPR@1%FPR of 89.0%. Although a large percentage of the green-list patterns are removed, the average z-score in the attacked images is still as high as 10.8, which remains significant when compared with a mean z-score of 0.2 for non-watermarked images. Moreover, the attack causes a quality loss of the image when compared to the original image generated by Infinity. With 200 iterations and 100 candidates during each iteration, the method further requires a high computational cost, where more than 12 minutes are needed to attack a single image in the environment specified in Appendix C.2. We note that, the results do not conclude the failure of the theoretical framework in watermarks in the sand, but rather that there are currently no successful instantiations of the perturbation oracles and quality oracle for Infinity that can remove our proposed watermark. More detailed analysis about this attack can be found in Appendix F.

Robustness Against our Novel Bit-Flipper Attack. We further demonstrate that, even giving the attacker much stronger knowledge than our threat model formulated in Section 3, BitMark cannot be erased with severe image degradation. Concretely, we assume the attacker is given the knowledge of the auto-encoder of the generative model, the watermarking embedding and detection algorithms, and even our red and green lists. We propose a novel Bit-Flipper attack tailored for this broader attack surface, aiming at removing our watermarks. During this attack, the attacker first computes the total number of bit sequences $|s|$ and calculate the green fraction $C/|s|$. Then, the attacker randomly flips bits to convert green bit sequences to red ones. The goal of the bit flipping is to reduce the green fraction to the level of non-watermarked images, which is 50%, as shown in Appendix D. Specifically, the attacker

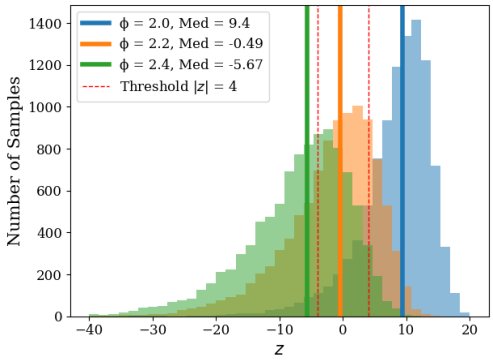

Figure 2: **Bit-Flipper attack** for $\delta = 2$

computes the probability of flipping a bit via a flip factor ϕ, such that $p = (C/|s|) - 0.5) * \phi$, whereas ϕ determines the bit sequence removal severity. A detailed algorithm of our attack is described in Algorithm 3. As shown in Appendix G, $\phi = 2.2$ has the most impact on detectability. However, randomly flipping bits comes with very high degradation in image quality, as shown in Appendix E.2. Due to the symmetric detection threshold $|z|$, if we destroy too many bit sequences in G, this further leads to the detectability of the watermark (see Appendix G).

4.3 Our BitMark Watermark Exhibits High Radioactivity

We demonstrate that our BitMark exhibits high radioactivity, *i.e.,* other generative models fine-tuned on our watermarked images also produce our bit patterns that are reliably detectable.

Formalizing Radioactivity. We denote by model M_1 our original model that generates watermarked outputs with a watermarking method, *e.g.,* our BitMark. These watermarked outputs might be used as a training set for a second model M_2. In the context of language models it has been shown, that there are watermarks that persist this process and remain detectable in the outputs of M_2, a property referred to as *radioactivity* [29]. This property enables the detection of generated content, even when it has been used to train subsequent models. However, recent findings indicate that this property does not hold for watermarks in image generative models, such as diffusion models [6].

Training Setup. Given the extensive compute costs of training high-performant image-autoregressive models from scratch, we simulate full training by full fine-tuning the models. Therefore, we first generate 1,000 images with prompts of the MS-COCO2014 [16] validation dataset with our BitMark embedded under a watermarking strength of $\delta = 2$. The watermarked outputs of M_1 are then used to train (*i.e.,* in our approximative setup full fine-tune) the M_2 model. We analyze two different settings, with the same M_1 of Infinity-2B, namely: (1) the same modality setting, transfer to Infinity-2B and (2) the cross-modality setting, transfer to the Latent Diffusion Model (LDM) Stable Diffusion 2.1 [24]. To test the radioactivity, we compute the TPR@1%FPR of output images of M_2 and report them in Table 5.

Table 5: **BitMark exhibits strong radioactivity.** We report the TPR@1%FPR (%).

Type of M_1	Type of M_2	Output of M_1	Output of M_2
Infinity-2B	Infinity-2B	100.0	99.7
Infinity-2B	Stable Diffusion 2.1	100.0	98.9

BitMark is Highly Radioactive. Table 5 highlights the strong radioactivity of our BitMark. When fine-tuning a model M_2 of the same architecture, the watermark is still detectable with a 99.7% TPR@1%FPR. Even in the cross-modality setting, BitMark is still detectable with a 98.9%TPR@1%FPR. Dubiński et al. [6] attribute the lack of radioactivity of watermarks in LDMs to the encoding from the pixel space into the latent space and the *noising-denoising* process while training M_2. However, our BitMark remains detectable despite the encoding and the noising-denoising process. The robustness of our watermark arises from slightly changing all parts of a given generated image, including low-level and high-level features, which are then also partly adapted by M_2. The observed results are in line with our robustness against CtrlRegen+ (see Table 3), an attack which simulates noising and denoising in the latent space. BitMark's radioactivity is a very useful property that allows model owners to track provenance of their data and data generated based on it, even when this new data does not originate *directly* from their original model.

5 Conclusions

We present BitMark, the first bitwise watermarking framework for image autoregressive models, such as Infinity. We show that our watermark can be efficiently embedded and detected, imposes negligible overhead on image quality and generation speed, while exhibiting a strong robustness against a wide range of watermark removal attacks, including the strong watermarks in the sand and our own adaptive Bit-Flipper attack. Moreover, BitMark exhibits strong radioactivity, *i.e.,* downstream models fine-tuned on our watermarked outputs consistently inherit and reproduce our bit patterns, which makes our watermark reliably detectable. This is important for model owners to identify images derived from their own generated images, and exclude them from training of future model versions. Overall, BitMark not only sets a new standard for watermarking in autoregressive image generation, but also provides a critical defense against model collapse, contributing to further advancements in image generative models.

References

[1] Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, and Richard Baraniuk. Self-consuming generative models go MAD. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ShjMHfmPs0.

[2] Bang An, Mucong Ding, Tahseen Rabbani, Aakriti Agrawal, Yuancheng Xu, Chenghao Deng, Sicheng Zhu, Abdirisak Mohamed, Yuxin Wen, Tom Goldstein, et al. Waves: Benchmarking the robustness of image watermarks. In *International Conference on Machine Learning*, pages 1456–1492. PMLR, 2024.

[3] Niyar R Barman, Krish Sharma, Ashhar Aziz, Shashwat Bajpai, Shwetangshu Biswas, Vasu Sharma, Vinija Jain, Aman Chadha, Amit Sheth, and Amitava Das. The brittleness of ai-generated image watermarking techniques: Examining their robustness against visual paraphrasing attacks. *arXiv preprint arXiv:2408.10446*, 2024.

[4] Franziska Boenisch. A systematic review on model watermarking for neural networks. *Frontiers in big Data*, 4:729663, 2021.

[5] Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, 2024.

[6] Jan Dubiński, Michel Meintz, Franziska Boenisch, and Adam Dziedzic. Are watermarks for diffusion models radioactive? In *The 1st Workshop on GenAI Watermarking*, 2025. URL https://openreview.net/forum?id=gtXbVRMwQh.

[7] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[8] 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 (ICCV)*, pages 22466–22477, October 2023.

[9] Sam Gunn, Xuandong Zhao, and Dawn Song. An undetectable watermark for generative image models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=jlhBFm7T2J.

[10] Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[11] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR, 2021.

[12] Hengrui Jia, Christopher A Choquette-Choo, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. *arXiv preprint arXiv:2002.12200*, 2020.

[13] Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. In *International Conference on Machine Learning*, pages 22570–22593. PMLR, 2024.

[14] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR, 2023.

[15] Linyang Li, Botian Jiang, Pengyu Wang, Ke Ren, Hang Yan, and Xipeng Qiu. Watermarking llms with weight quantization. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3368–3378, 2023.

[16] 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.

[17] Aiwei Liu, Sheng Guan, Yiming Liu, Leyi Pan, Yifei Zhang, Liancheng Fang, Lijie Wen, Philip S. Yu, and Xuming Hu. Can watermarked LLMs be identified by users via crafted prompts? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ujpAYpFDEA.

[18] Yepeng Liu, Yiren Song, Hai Ci, Yu Zhang, Haofan Wang, Mike Zheng Shou, and Yuheng Bu. Image watermarks are removable using controllable regeneration from clean noise. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mDKxlfraAn.

[19] Subhadip Maji and Smarajit Bose. Deep image orientation angle detection. *arXiv preprint arXiv:2007.06709*, 2020.

[20] K.A. Navas, Mathews Ajay, M. Lekshmi, Tampy Archana, and M. Sasikumar. Dwt-dct-svd based watermarking. pages 271 – 274, 02 2008. ISBN 978-1-4244-1796-4. doi: 10.1109/COMSWA.2008.4554423.

[21] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[22] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[23] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[24] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[25] Mehrdad Saberi, Vinu Sankar Sadasivan, Keivan Rezaei, Aounon Kumar, Atoosa Chegini, Wenxiao Wang, and Soheil Feizi. Robustness of AI-image detectors: Fundamental limits and practical attacks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dLoAdIKENc.

[26] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Radioactive data: tracing through training. In *International Conference on Machine Learning*, pages 8326–8335. PMLR, 2020.

[27] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[28] Tom Sander, Pierre Fernandez, Alain Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. *Advances in Neural Information Processing Systems*, 37: 21079–21113, 2024.

[29] Tom Sander, Pierre Fernandez, Alain Oliviero Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=qGiZQb1Khm.

[30] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022.

[31] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022): 755–759, 2024.

[32] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[34] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-rings watermarks: Invisible fingerprints for diffusion images. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 58047–58063. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/b54d1757c190ba20dbc4f9e4a2f54149-Paper-Conference.pdf.

[35] Bram Wouters. Optimizing watermarks for large language models. In *Forty-first International Conference on Machine Learning*, 2024.

[36] Sierra Wyllie, Ilia Shumailov, and Nicolas Papernot. Fairness feedback loops: training on synthetic data amplifies bias. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, pages 2113–2147, 2024.

[37] Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation, 2024. URL https://arxiv.org/abs/2411.00776.

[38] Hanlin Zhang, Benjamin L Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. Watermarks in the sand: impossibility of strong watermarking for language models. In *Forty-first International Conference on Machine Learning*, 2024.

[39] Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. {REMARK-LLM}: A robust and efficient watermarking framework for generative large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1813–1830, 2024.

[40] Yiming Zhang, Javier Rando, Ivan Evtimov, Jianfeng Chi, Eric Michael Smith, Nicholas Carlini, Florian Tramèr, and Daphne Ippolito. Persistent pre-training poisoning of LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=eiqrnVaeIw.

[41] Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=SsmT8aO45L.

[42] Yue Zhao, Yuanjun Xiong, and Philipp Kraehenbuehl. Image and video tokenization with binary spherical quantization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yGnsH3gQ6U.

[43] Yunqing Zhao, Tianyu Pang, Chao Du, Xiao Yang, Ngai-Man Cheung, and Min Lin. A recipe for watermarking diffusion models, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claimed to developed a watermark which is robust towards most state of the art attacks. It sets a new benchline for well established attacks, as well as outperforms current benchmarks regarding more advanced attacks like Watermark in the Sand or CtrlRegen. We showed exhaustive empirical results as well as ablations in the appendix to support those claims.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, we pointed towards the limitation to bitwise domains as well as limitaitons towards robustness regarding rotation. Further, limitations have been stated in Appendix H.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: There are no theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We stated our method in Section 3.2 and Section 3.3 and displayed all core algorithms (see Algorithm 1, Algorithm 2, Algorithm 3) as well as settings and hyperparameters Appendix C.1 to reproduce these results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide access to the code in the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Appendix C.1, Section 4 and the provided supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All plots and tables contain the mean over the specified number of samples.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix C.2 for specification of the compute resources and environment.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research is conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix H for a broader impact of the work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper does not provide new models or datasets.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Model owners and used code are clear and properly cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code for our method will be submitted as supplementary material and published upon acceptance of the paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Additional Algorithms

We detail the functions used in Algorithm 1, Algorithm 2 and Algorithm 3 here:

- Bias: Addition of the bias δ to the current logit l_c based on the green list G.
- Sample: Use the probabilities in p_c to sample the returned bit, either 0 or 1.
- lookup: Perform the BSQ lookup [42] to return the quantized states.
- interpolate: Scaling to target resolution.
- Count: Action performed for the statistical test, which counts the number of bits which are part of the green list.

We present the pseudo code of the novel Bit-Flipper attack in Algorithm 3. To apply the attack, we have to first encode the image in to bits (line 1). During deconstructing the old watermarked image, we construct a new attacked image. We recursively iterate through all scales K, retrieve the bit representations of the respective scale (line 4) and count how many bits are part of the green list and the red list (line 5). Then we define the probability of flipping a bit based on the green fraction and the flip factor ϕ. Next, we change the bits which are part of the green list based on the random probability p (line 8, 9). We follow the standard encoding and decoding process, to iterate through all resolutions of the image (line 10-13). Finally, we return the decoded version of the attacked image.

Algorithm 3 Bit-Flipper Attack

Inputs: Image im, green list G, red list R, image decoder $\mathcal{D}$.
Hyperparameters: Flip factor ϕ, steps K (number of resolutions), resolutions $(h_i, w_i)_{i=1}^{K}$, the number of bits for resolution i is t_i, n - the length of the bit vector, $|s|$ the total number of bit sequences.
$e = \mathcal{E}(im)$
$img =$ init()
for $i = 1, \ldots, K$ **do**
  $u_i = \mathcal{Q}(\text{interpolate}(e, h_i, w_i))$
  C_i = Count($(b_1, \ldots, b_{t_i})$, G)
  $p = (\frac{C_i}{|s_i|} - 0.5) * \phi$
  for $j = n, \ldots, t_i$ **do**
    if $(b_{j-n}, \ldots, b_j) \in G \wedge$ *random*() $< p$ **then**
      $b_j = \neg b_j$ {Flip bit with probability p if sequence is in green list}
  $z_i = \text{lookup}(u_i)$
  $z_i = \text{interpolate}(z_i, h_K, w_K)$
  $e = e - \phi_i(z_i)$
  $img = img + \phi_i(z_i)$
Return: attacked image $\mathcal{D}(img)$

B Green and Red List Selection

Selection of Red and Green Lists. The re-encoding loss presented in Section 3.1 also motivates the choice of a small length of the green list n, as larger consecutive bit sequences are more likely to be interrupted. Additionally, this auto-encoder has the property of re-encoding images towards an equal frequency of 0's and 1's, so it is not efficiently possible to bias each bit towards either 0 or 1, making the choice of $n = 1$ unpractical (see Table 6). Thus we leverage the second smallest possible choice, $n = 2$, resulting in 6 possible choices to separate G and R by keeping $|G| = |R|$.

As we have the constraint to not bias towards 0 or 1, as this type of pattern is more likely removed by the auto-encoder, and given the constraint shown that the same prefix must not be in the same list twice as discussed in Section 3.2, the most effective G for BitMark are $G = \{01, 10\}$ and $G = \{00, 11\}$. We choose $G = \{01, 10\}$ throughout all experiments.

Changing the Length of the Bit Sequence. We additionally ablate the size of the bit sequence in Table 6. For each sequence size n, if both lists should have the same size, we choose $\frac{2^n}{2}$ out of 2^n possibilities without replacement, so there are $\binom{2^n}{2^{n-1}}$ possibilities of constructing the G and R. For the 2-bit sequences, flipping the green and red list has only limited impact on the final z-score, thus

for the 3-bit sequences we only analyze all unique choices, *i.e.,* we do not interchange green and red list, leaving us with 35 distinct possible green and red lists. For $n = 3$, two G have a similar z compared to the best 2-bit sequences. These choices have the same properties as discussed above, namely (1) they equally often bias towards 0 and 1 and (2) none of the sequences share the same prefix, *i.e.,* the biasing is always effective.

Table 6: **Green and Red list selection.** We analyze all possible green and red lists for $n = 1$, $n = 2$ and $n = 3$ with $\delta = 2$ and 1,000 images. We also report the fraction of 1's that are in the bit representation during generation of the image (**1s Gen**) and after re-encoding the image using the image encoder (**1s Enc**).

Green List G	Red List R	Average Z-Score	1s Gen	1s Enc
$\{0\}$	$\{1\}$	6.18	31.50%	49.47%
$\{1\}$	$\{0\}$	2.39	64.52%	50.21%
$\{11, 00\}$	$\{01, 10\}$	91.35	49.74%	49.98%
$\{01, 10\}$	$\{00, 11\}$	90.33	49.97%	49.99%
$\{00, 10\}$	$\{01, 11\}$	6.52	31.85%	49.44%
$\{11, 01\}$	$\{00, 10\}$	2.56	63.70%	50.22%
$\{00, 01\}$	$\{10, 11\}$	0.25	49.92%	49.98%
$\{11, 10\}$	$\{00, 01\}$	-0.25	49.92%	49.98%
$\{000, 011, 100, 111\}$	$\{001, 010, 101, 110\}$	88.00	49.75%	49.98%
$\{000, 011, 110, 111\}$	$\{001, 010, 100, 101\}$	87.91	49.75%	49.98%
$\{000, 011, 101, 111\}$	$\{001, 010, 100, 110\}$	67.90	49.75%	49.98%
$\{000, 100, 110, 111\}$	$\{001, 010, 011, 101\}$	44.95	40.92%	49.52%
$\{000, 001, 011, 111\}$	$\{010, 100, 101, 110\}$	32.91	58.19%	50.24%
$\{000, 001, 010, 101\}$	$\{011, 100, 110, 111\}$	32.85	44.09%	49.68%
$\{000, 010, 011, 111\}$	$\{001, 100, 101, 110\}$	29.04	40.92%	49.52%
$\{000, 100, 101, 111\}$	$\{001, 010, 011, 110\}$	29.04	40.92%	49.52%
$\{000, 101, 110, 111\}$	$\{001, 010, 011, 100\}$	26.43	40.92%	49.52%
$\{000, 011, 100, 110\}$	$\{001, 010, 101, 111\}$	20.11	49.75%	49.98%
$\{000, 001, 011, 110\}$	$\{010, 100, 101, 111\}$	10.03	58.19%	50.24%
$\{000, 001, 011, 100\}$	$\{010, 101, 110, 111\}$	8.72	58.19%	50.24%
$\{000, 001, 010, 100\}$	$\{011, 101, 110, 111\}$	6.48	44.09%	49.68%
$\{000, 010, 100, 101\}$	$\{001, 011, 110, 111\}$	6.20	33.07%	49.48%
$\{000, 010, 100, 110\}$	$\{001, 011, 101, 111\}$	5.99	33.07%	49.48%
$\{000, 010, 011, 100\}$	$\{001, 101, 110, 111\}$	5.55	40.92%	49.52%
$\{000, 001, 010, 110\}$	$\{011, 100, 101, 111\}$	3.74	44.09%	49.68%
$\{000, 010, 101, 110\}$	$\{001, 011, 100, 111\}$	3.21	33.07%	49.48%
$\{000, 010, 011, 110\}$	$\{001, 100, 101, 111\}$	2.93	40.92%	49.52%
$\{000, 100, 101, 110\}$	$\{001, 010, 011, 111\}$	2.93	40.92%	49.52%
$\{000, 010, 100, 111\}$	$\{001, 011, 101, 110\}$	2.14	33.07%	49.48%
$\{000, 001, 100, 101\}$	$\{010, 011, 110, 111\}$	0.25	49.92%	49.98%
$\{000, 001, 010, 011\}$	$\{100, 101, 110, 111\}$	0.25	49.92%	49.98%
$\{000, 011, 100, 101\}$	$\{001, 010, 110, 111\}$	0.11	49.75%	49.98%
$\{000, 001, 101, 110\}$	$\{010, 011, 100, 111\}$	0.07	49.92%	49.98%
$\{000, 011, 101, 110\}$	$\{001, 010, 100, 111\}$	0.02	49.75%	49.98%
$\{000, 001, 100, 110\}$	$\{010, 011, 101, 111\}$	-0.05	49.92%	49.98%
$\{000, 001, 101, 111\}$	$\{010, 011, 100, 110\}$	-0.26	49.92%	49.98%
$\{000, 001, 100, 111\}$	$\{010, 011, 101, 110\}$	-0.37	49.92%	49.98%
$\{000, 001, 110, 111\}$	$\{010, 011, 100, 101\}$	-0.55	49.92%	49.98%
$\{000, 010, 101, 111\}$	$\{001, 011, 100, 110\}$	-0.65	33.07%	49.48%
$\{000, 010, 110, 111\}$	$\{001, 011, 100, 101\}$	-0.85	33.07%	49.48%
$\{000, 001, 011, 101\}$	$\{010, 100, 110, 111\}$	-1.45	58.19%	50.24%
$\{000, 010, 011, 101\}$	$\{001, 100, 110, 111\}$	-12.98	40.92%	49.52%
$\{000, 001, 010, 111\}$	$\{011, 100, 101, 110\}$	-13.39	44.09%	49.68%

Consistency in Longer Sequences. We analyze the overlap of bit sequences with growing size in Figure 3 which shows that with increased length of the sequence, the consistency of the pattern

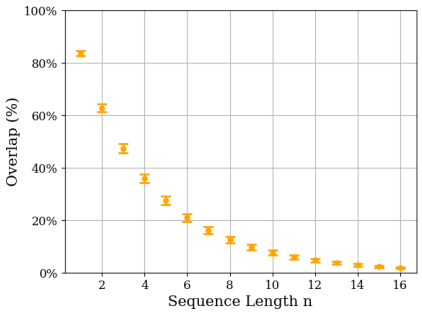

Figure 3: **Longer sequences are less consistent after re-encoding.** We analyze the consistency of sequences of length n after re-encoding over the whole image, following the setup of Figure 1.

decreases. This has a direct impact on the choice of n, as it shows that larger n lead to less overlap, decreasing the impact of BitMark which is already noticeable for $n = 3$ (see Table 6).

C Experimental Setup

C.1 Hyperparameters

For the experiments on radioactivity, we chose 1,000 prompts from the MS-COCO 2014 validation dataset, generated the respective number of images with the M_1 Infinity model and then fine-tuned the second model M_2 for 5 epochs with a learning rate of 10^{-4} and a batch-size of 4 (batch-size of 1 for Infinity-2B).

Since VAR [32] and RAR [37] are class-conditional models for the ImageNet1k dataset, we first generated ImageNet fakes using Infinity-2B with the ImagenNet class name and "A photo of {class name}" as prompt. We used the same hyperparameters such as batch-size, number of epochs and learning rate consistent with the experiments on Infinity-2B and Stable Diffusion 2.1.

C.2 Environment

Our experiments are performed on Ubuntu 22.04, with Intel(R) Xeon(R) Gold 6330 CPU and NVIDIA A40 Graphics Card with 40 GB of memory.

D Natural Images

Table 7: **Detection on non-watermarked samples.** Mean (std) of z and green fractions for 1,000 non-watermarked samples.

Dataset	z	Green Fraction
Imagenet1k	0.6311 (1.966)	0.5005 (0.0017)
MS-COCO2014	0.6773 (1.735)	0.5006 (0.0015)
MS-COCO2014 (Infinity-2B generated)	0.3620 (1.641)	0.5003 (0.0014)

To ensure the validity of our hypothesis we compute the average number of members of the green list and z for natural images from different datasets, as well as images generated for the non-watermarked Infinity model. Table 7 shows that indeed, the green fraction for non-watermarked images is 50%.

E Additional Experiments

E.1 Detailed Analysis of BitMark

Next, we analyze the low-level behavior of BitMark during image generation. Table 9 depicts information about the generation of images with Infinity, averaged over 10,000 images. It shows that

the entropy is low in earlier scales, meaning the bits are chosen with high certainty as these depend more the input prompt, but large on later scales, which means sampling either of the possible values for the bit can lead to different, but both high-quality content. This insight aligns with the re-encoding loss, which keeps more bits consistent on earlier scales.

Table 10, depicts the low-level information of applying BitMark with $\delta = 2$. Applying BitMark changes up to 37% of bits per scale, yet the re-encoding loss only increases by 5% compared to the non watermarked image. Table 8 further shows the correlation between different analyzed factors. As expected, there is a high correlation between changed bits and entropy, as due to the soft-biasing we mostly change bits which have low entropy.

Table 8: **Correlations of different factors when applying BitMark on all scales within the Infinity generation process.** Entropy, Green Fraction, Changed Bits & Re-Encoding Loss are relative values. The Re-Encoding Loss displays the number of bits changed during re-encoding, whereas Changed Bits display the number of bits changed during the generation process. 10,000 images, $\delta = 2$.

	Entropy	Changed Bits	z	Green Fraction
Changed Bits	0.999	-	-	-
z	0.706	0.689	-	-
Green Fraction	0.918	0.917	0.642	-
Re-Encoding Loss	0.926	0.931	0.475	0.751

Table 9: **Statistics regarding the infinity generation process for 10,000 images.** Entropy, Green Fraction& Re-Encoding Loss are relative values. The Re-Encoding Loss displays the number of bits changed during re-encoding. We report the mean (std) for all values.

Scale	Entropy	z	Green Fraction	Re-Encoding Loss	Number of Tokens (Bits)
1	0.051 (0.037)	0.01 (0.876)	0.501 (0.079)	0.068 (0.063)	1 (32)
2	0.108 (0.044)	-0.195 (1.114)	0.491 (0.049)	0.132 (0.063)	4 (128)
3	0.133 (0.034)	-0.463 (1.226)	0.49 (0.027)	0.145 (0.041)	16 (512)
4	0.178 (0.029)	-0.522 (1.217)	0.492 (0.018)	0.2 (0.031)	36 (1,152)
5	0.202 (0.024)	-0.558 (1.199)	0.494 (0.013)	0.226 (0.024)	64 (2,048)
6	0.215 (0.021)	-0.663 (1.229)	0.495 (0.009)	0.247 (0.02)	144 (4,608)
7	0.223 (0.019)	-0.5 (1.266)	0.497 (0.007)	0.239 (0.017)	256 (8,192)
8	0.223 (0.015)	-0.349 (1.187)	0.498 (0.005)	0.277 (0.016)	400 (12,800)
9	0.226 (0.014)	-0.122 (1.173)	0.5 (0.004)	0.279 (0.014)	576 (18,432)
10	0.243 (0.014)	0.157 (1.165)	0.5 (0.003)	0.264 (0.012)	1,024 (32,768)
11	0.237 (0.012)	0.227 (1.161)	0.501 (0.003)	0.319 (0.011)	1,600 (51,200)
12	0.233 (0.012)	0.329 (1.198)	0.501 (0.002)	0.328 (0.01)	2,304 (73,728)
13	0.234 (0.014)	0.462 (1.173)	0.501 (0.002)	0.212 (0.012)	4,096 (131,072)
1-13	0.193 (0.012)	0.232 (1.644)	0.497 (0.008)	0.266 (0.009)	10,521 (336,372)

Watermarking in Early vs Late Scales. We provide both qualitative and quantitative analysis for the scales in the Infinity architecture. We observe that the model produces high-level information in the image during the initial scales, forming a coarse shape of the main visual components, such as the object and the background (see Figure 4. In larger scales, the model predicts the details of the image (see Figure 5). As shown in Table 11, earlier and late scales contribute differently to the robustness of BitMark. Earlier scales are more robust against like CtrlRegen, as CtrlRegen does not change the content of the image itself.

E.2 Image Quality

We analyze the impact of applying BitMark with different δ on the image quality. Figure 6 shows that applying BitMark with $\delta \leq 3$ keeps the semantic structure and quality of the image the same, while changing small features, *e.g.,* the color of the train in the first row. For larger watermarking strengths, the image quality starts to deteriorate, where artifacts become visible in the final image.

E.3 Timing

We present the average elapsed time (in seconds) for the generation of an image with Infinity, when the watermark is applied to all scales (resolutions) and when using the standard generation process

Table 10: **Statistics regarding the infinity generation process with BitMark applied on all scales for 10,000 images,** $\delta = 2$**.** Entropy, Green Fraction, Re-Encoding Loss & Changed Bits are relative values. The Re-Encoding Loss displays the number of bits changed during re-encoding, whereas Changed Bits display the number of bits changed during the generation process. We report the mean (std) for all values.

Scale	Entropy	z	Green Fraction	Re-Encoding Loss	Changed Bits
1	0.051 (0.037)	0.218 (0.897)	0.52 (0.081)	0.088 (0.076)	0.09 (0.077)
2	0.108 (0.043)	0.854 (1.105)	0.538 (0.049)	0.181 (0.079)	0.179 (0.075)
3	0.132 (0.032)	2.37 (1.452)	0.552 (0.032)	0.2 (0.053)	0.208 (0.05)
4	0.175 (0.028)	3.965 (1.613)	0.558 (0.024)	0.26 (0.037)	0.279 (0.042)
5	0.2 (0.024)	6.252 (1.812)	0.569 (0.02)	0.286 (0.029)	0.313 (0.035)
6	0.213 (0.021)	10.118 (2.304)	0.575 (0.017)	0.305 (0.022)	0.329 (0.029)
7	0.224 (0.019)	15.701 (2.832)	0.587 (0.016)	0.3 (0.02)	0.346 (0.027)
8	0.224 (0.016)	16.247 (2.788)	0.572 (0.012)	0.329 (0.016)	0.345 (0.022)
9	0.228 (0.015)	20.876 (3.341)	0.577 (0.012)	0.333 (0.013)	0.351 (0.02)
10	0.244 (0.015)	32.837 (5.104)	0.591 (0.014)	0.319 (0.011)	0.373 (0.019)
11	0.237 (0.013)	29.535 (5.06)	0.565 (0.011)	0.361 (0.009)	0.363 (0.017)
12	0.242 (0.013)	34.973 (6.681)	0.564 (0.012)	0.37 (0.008)	0.368 (0.017)
13	0.247 (0.014)	64.333 (11.823)	0.589 (0.016)	0.256 (0.015)	0.374 (0.019)
1-13	0.194 (0.012)	90.784 (14.852)	0.566 (0.013)	0.312 (0.009)	0.366 (0.017)

Table 11: **Robustness of BitMark applied to different scales with** $\delta = 2$**.** We report the TPR@1%FPR (%) for the different attacks on 5,000 watermarked images.

Scales	Conventional Attacks							Reconstruction Attacks	
	None	Noise	Blur	Color	Crop	Rotation	JPEG	SD2.1-VAE	CtrlRegen+
1-10	100.0	97.9	98.7	97.6	61.6	15.0	100.0	100.0	82.1
11-13	100.0	96.7	98.6	99.5	20.0	9.8	100.0	100.0	15.5
1-13	100.0	99.6	99.9	99.8	98.8	20.1	100.0	100.0	91.6

(without a watermark). For hardware information see Appendix C.2. Table 12 shows that applying BitMark results in negligible overhead to image generation and allows for a fast detection in under 0.5 seconds.

Table 12: **Timing results for image processing and detection per image.** We report the mean (std) for 1,000 images.

Setup	Seconds/Image
BitMark	2.5817 (0.0063)
No watermark	2.3196 (0.0043)
Detection	0.4491 (0.0046)

E.4 Radioactivity

Table 13: **We analyze the radioactivity of BitMark for class-conditional autoregressive models.** We report the TPR@1%FPR (%).

Type of M_1	Type of M_2	Output of M_1	Output of M_2
Infinity-2B	VAR-16	100.0	24.2
Infinity-2B	VAR-20	100.0	25.8
Infinity-2B	VAR-24	100.0	25.7
Infinity-2B	VAR-30	100.0	25.6
Infinity-2B	RAR-B	100.0	4.3
Infinity-2B	RAR-L	100.0	3.3
Infinity-2B	RAR-XL	100.0	3.9
Infinity-2B	RAR-XXL	100.0	4.1

Table 13 reports the radioactivity of BitMark for the class-conditional autoregressive models, namely, **VAR** [32], which operates in the multi-scale manner similarly to Infinity, and **RAR** [37], which generates tokens only on a single scale in an autoregressive fashion. We find that BitMark exhibits detectable traces of radioactivity for the class-to-image autoregressive models models with

TPR>>1%@FPR=1%, with slightly lower traces for RAR (probably due to its single scale generation) than VAR. A potential cause of the overall relatively lower radioactivity for VAR and RAR is stemming from the class-conditional setting, instead of the text-to-image as in Infinity.

E.5 Rotation

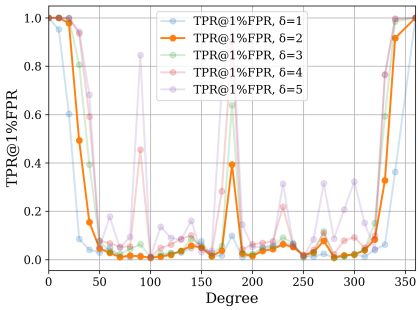

Figure 7: **BitMark is robust against rotation within ± 30 degrees.** The TPR@1%FPR for different δ and rotation degrees.

We ablate the robustness of BitMark against different degrees of rotation and different watermarking strengths δ. We find that with $\delta = 2$ BitMark is robust for rotation degrees ± 30. Additionally we find an increase in TPR@1%FPR for a rotation of 180°.

F Analysis for the Watermarks in the Sand Attack

As shown by the qualitative analysis in Figure 8, the watermarks in the sand attack causes a significant loss of details and produces many perceptible artifacts, limiting the applicability of this attack.

G Analysis for the Bit-Flipper Attack

Figure 9 shows different flipping strengths ϕ for $\delta = 2$ on 1,000 watermarked samples. The most effective ϕ is $\phi = 2.2$ with a TPR@1%FPR below 50% (see Table 14). The decrease in image quality for Bit-Flipper for different ϕ is shown in Figure 9, as well a more detailed analysis of $\phi = 2.2$ on BitMark with a $\delta = 2$ is displayed in Figure 10.

Table 14: **Applying Bit-Flipper with different factors ϕ on watermarked images with $\delta = 2$.** We report the TPR@1%FPR for 1,000 Images.

ϕ	TPR@1%FPR (%)	z
1.8	0.878	10.06 (4.24)
2.0	0.845	9.17 (4.16)
2.2	0.472	5.52 (5.07)
2.4	0.595	8.06 (7.28)
2.6	0.52	6.97 (6.96)

H Limitations and Broader Impact

Limitations. While BitMark are robust against moderate rotations (99.9% TPR@1%FPR within rotation angles -20° to 20°), the detection accuracy degrades with larger rotation angles, dropping to 56.8% TPR@1%FPR for ±50° rotations. Although open-source methods have a strong capability for rotation correction, this can be a limitation for real-world scenarios. This rotation sensitivity, however, is an inherent issue in Infinity's data augmentation pipeline, which makes the autoencoder less robust to large rotations. Moreover, the soft biasing parameter δ requires tuning, as values exceeding $\delta = 3$

lead to noticeable quality degradation (FID increases from 29.61 to 127.44 when δ increases from 3 to 5), constraining the maximum achievable watermark strength. Although we evaluate the detection BitMark with a commonly adopted metric, *i.e.,* true positive rate at very low false positive rates (1%), it could potentially become problematic in extreme-scale applications involving trillions of samples, where even small false positive rates might impact training data quality.

Broader Impact. Image generative models have the capabilities of generating photorealistic images, which are near imperceptible to human generated content. BitMark provides a method to watermark the content, without harming image generation capabilities of the model and allows for provenance tracking of generated images for models hosted under a private API. It supports the defense against model collapse, when training a model on data generated by itself.

I Additional Related Work

LLM Watermarking. Here, we review more related work regarding the LLM watermarking, which we build upon in our method. Beyond the soft biasing approach proposed by KGW [14], SynthID-Text [5] introduces tournament sampling during the model inference to enhance watermark detectability. This sampling strategy modifies the token sampling process to embed watermark signals. However, tournament sampling is only applicable to autoregressive models with large vocabulary sizes, making it incompatible with the binary bit sampling process used in Infinity. There are also works that incorporate watermarks by training or finetuning the models. For example, REMARK-LLM trains a learning-based encoder for watermark insertion and a decoder for watermark extraction, while using a reparameterization module to bridge the gap between the token distribution and the watermark message [39]. Moreover, potential malicious transformations are incorporated into the training phase to enhance the robustness of the watermarking. Although effective, these training-based methods require substantial computational resources and time. Therefore, we focus on inference-time watermarking methods that can be applied to Infinity without any retraining.

Sander et al. [28] investigate the *radioactivity* of LLM watermarking, which is the ability of watermarks to transfer across different model architectures. They design a specific protocol for amplifying the watermark signals after transferring across models. The radioactivity of watermarks also falls into the scope of our work, as our proposed watermark exhibits significant radioactivity across various model architectures. Regarding the undetectability of the watermark, recent studies show that the biases between watermarked and non-watermarked outputs from the LLM are identifiable with carefully designed prompts [17]. As a specialized application of LLM watermarking, Li et al. [15] design a watermarking scheme tailored for quantized LLMs. The proposed watermarks are only detectable on quantized models while remaining invisible for full-precision models.

Model Collapse. With the fraction of artificial generated content within the World Wide Web increasing, Shumailov et al. [31] investigate the phenomenon of model collapsing, which is defined as the decrease of generation quality when models are recursively trained on (partly) generated data. This decrease in generation quality is mainly characterized in overestimating probable events and underestimating low-probability events, as well as recursively adding noise (*i.e.,* errors) which is generated during earlier model generations. Hence, this generated data is polluting future datasets, decreasing the potential quality of successor models and therefore highlighting the necessity to distinguish human- from artificial-generated data. BitMark mitigates model collapsing by robustly watermarking generated images to exclude said marked images from future datasets.

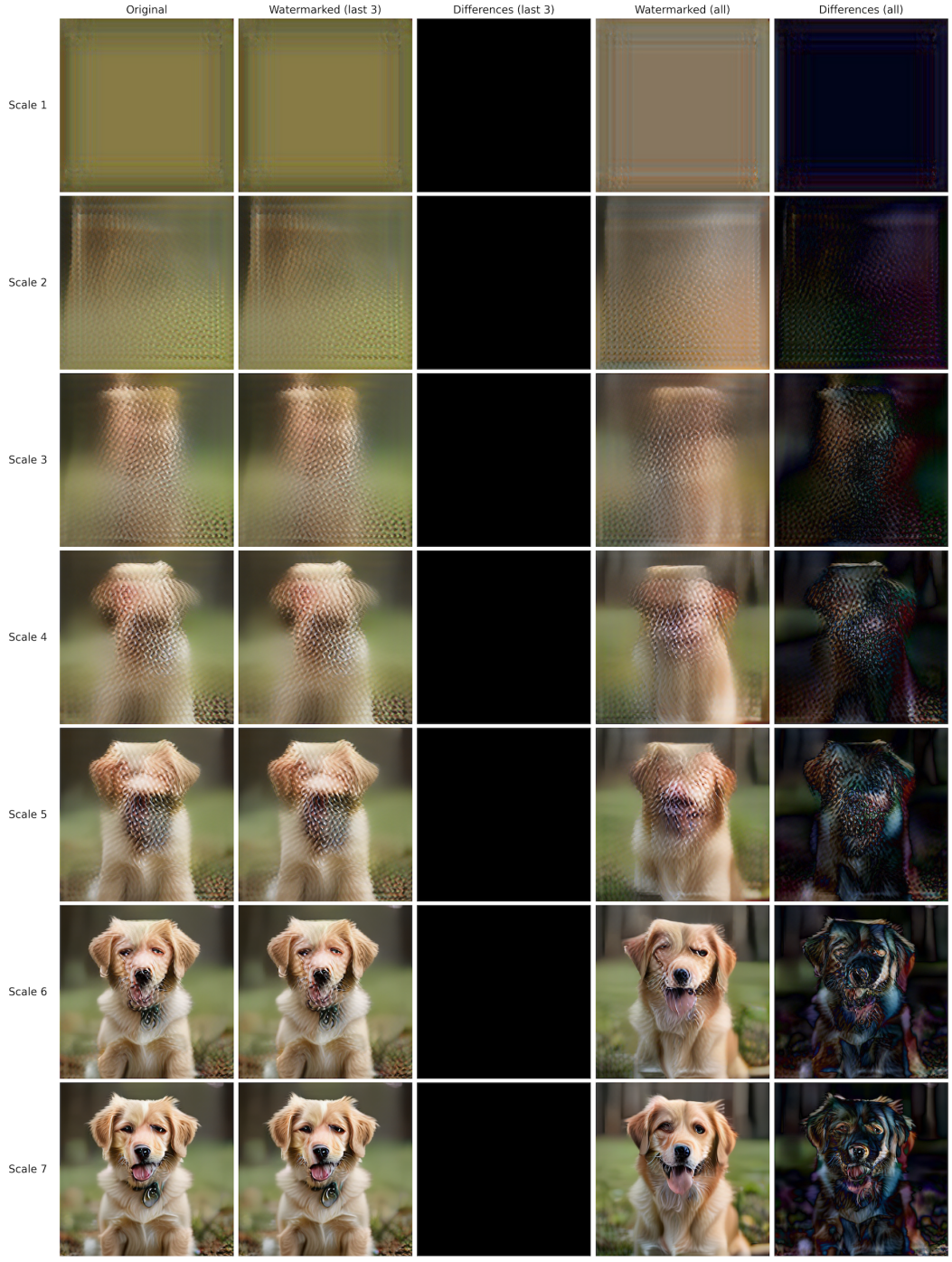

Figure 4: **Visualization of different images on different scales.** Here, the image on scale i refers to the cumulative image obtained by adding up 1 to i scales. We visualize five sets of images: ***1) Original*** images without watermarking, ***2) Watermarked (last 3)***, where the watermarks are generated on scale 11 to 13, ***3) Differences (last 3)***, which is the difference between the original and watermarked images (last 3), ***4) Watermarked (all)***, where the watermarks are generated on all scales, ***5) Differences (all)***, which is the difference between the original and watermarked images (all). Here, we visualize scales 1-7.

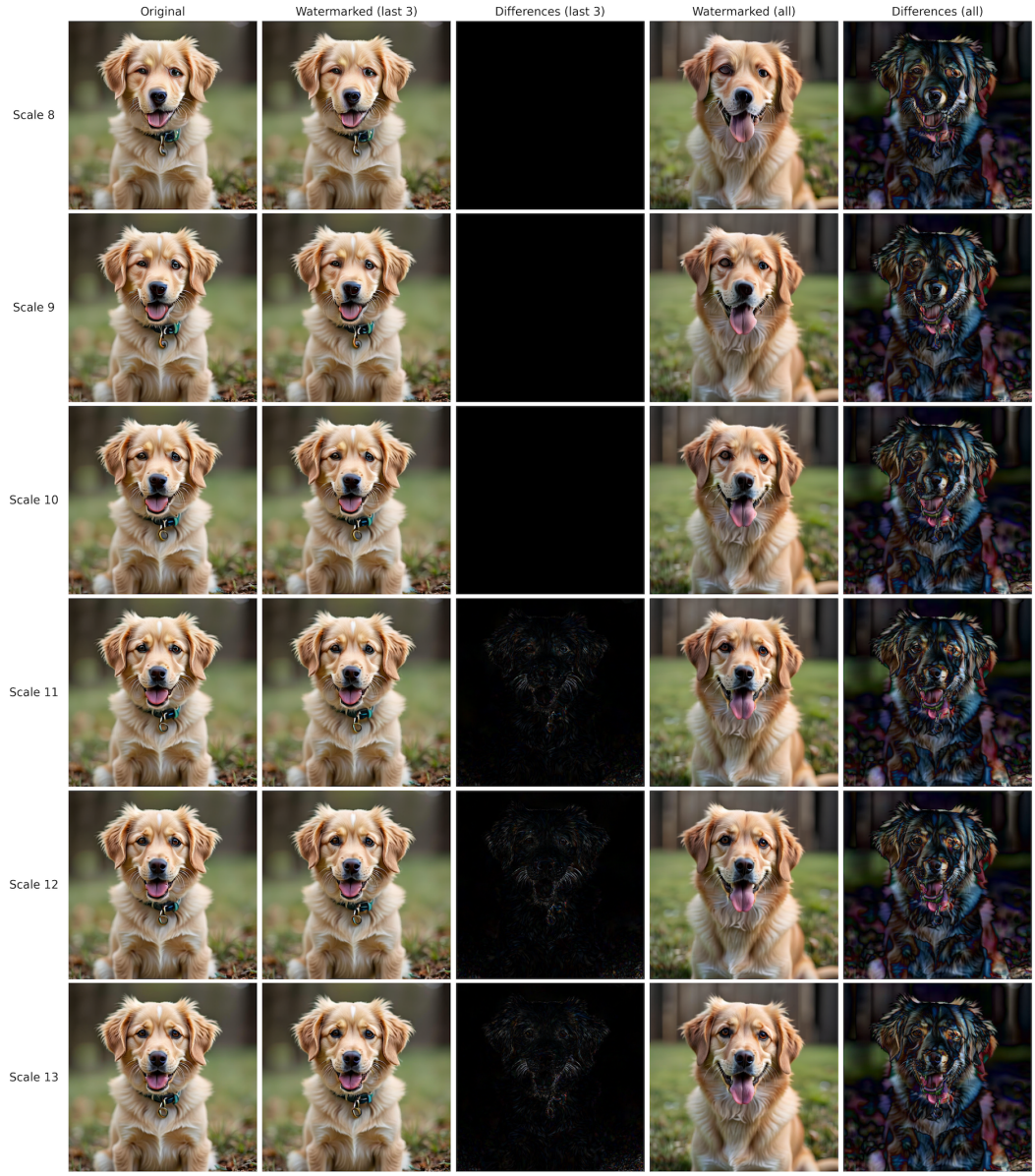

Figure 5: **Visualization of different images on different scales.** Here, the image on scale i refers to the cumulative image obtained by adding up 1 to i scales. We visualize five sets of images: ***1) Original*** images without watermarking, ***2) Watermarked (last 3)***, where the watermarks are generated on scale 11 to 13, ***3) Differences (last 3)***, which is the difference between the original and watermarked images (last 3), ***4) Watermarked (all)***, where the watermarks are generated on all scales, ***5) Differences (all)***, which is the difference between the original and watermarked images (all). Here, we visualize scales 8-13.

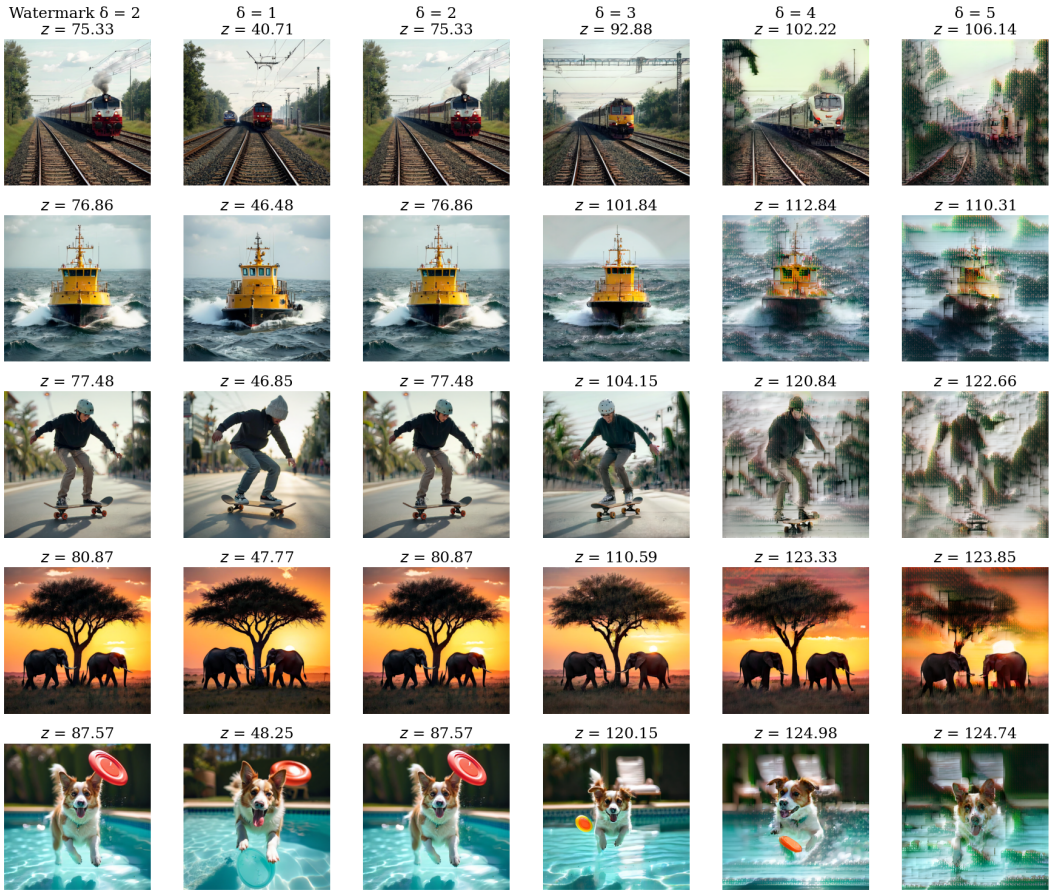

Figure 6: **Artifacts start to form in high detailed images (third row) already with $\delta = 3$.** Qualitative analysis of the impact of the choice of watermarking strength (δ) on the perceived image quality.

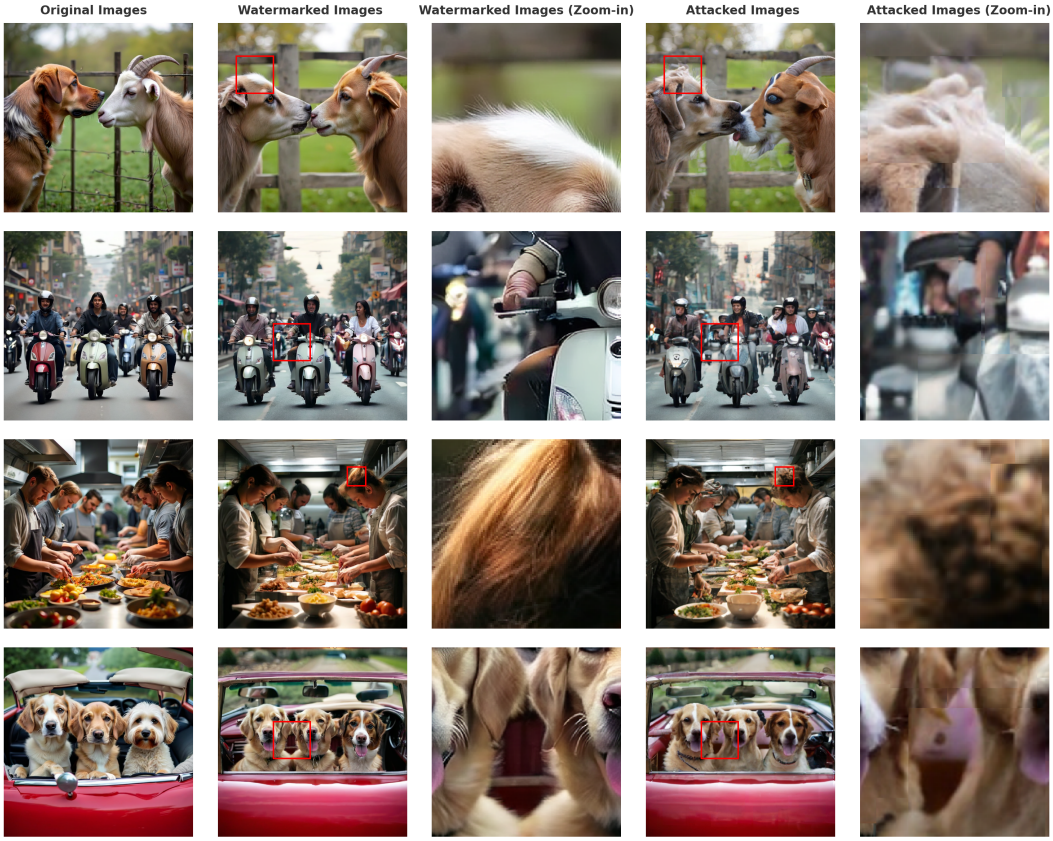

Figure 8: **Watermarks in the Sand has non-negligible impact on image quality.** Qualitative analysis of the impact of the watermarks in the sand. Here, we zoom in on some regions in the image to demonstrate the quality loss in details.

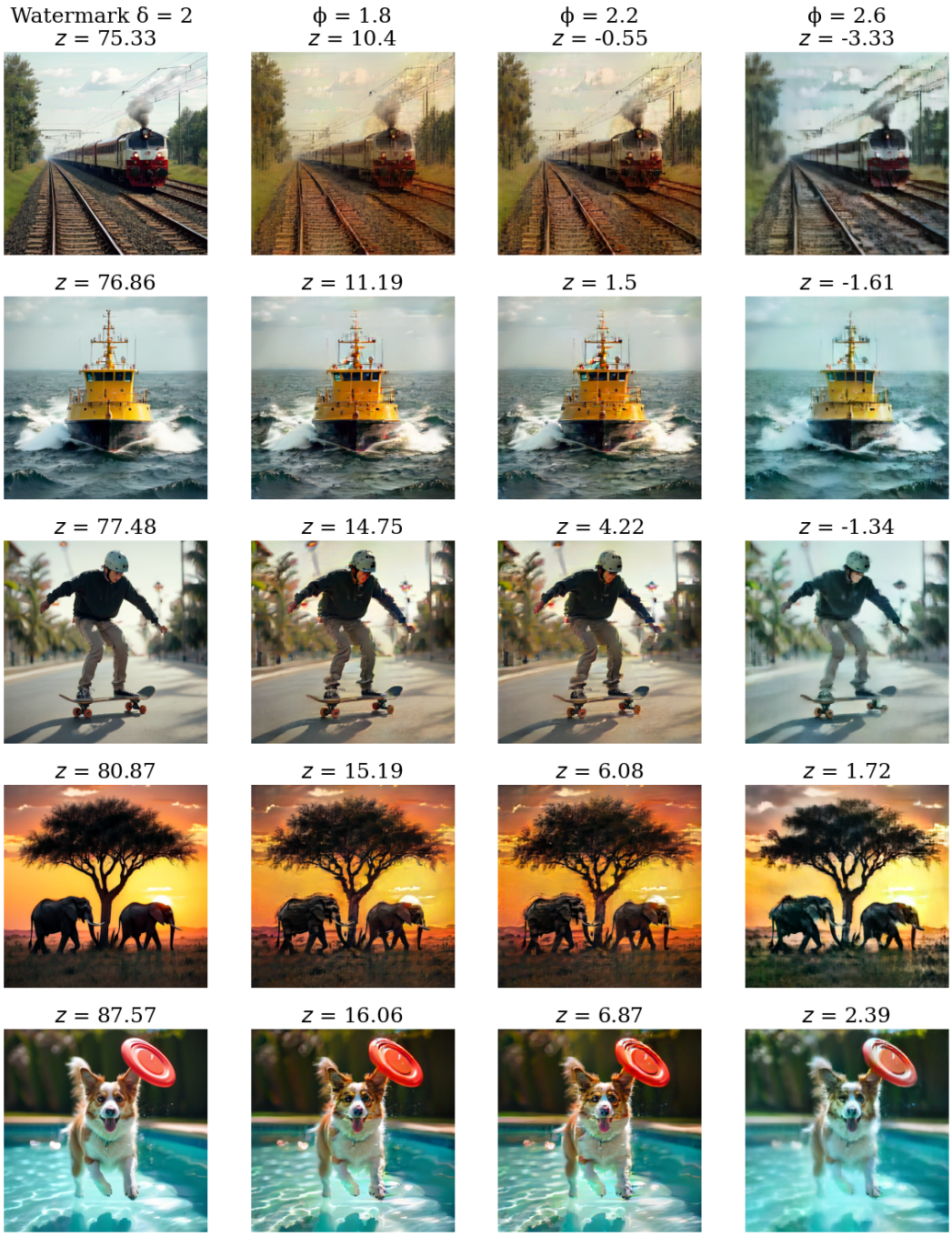

Figure 9: **A higher flip factor leads to worse image quality but better watermark removal.** Qualitative analysis of the impact of the Bit-Flipper with different ϕ.

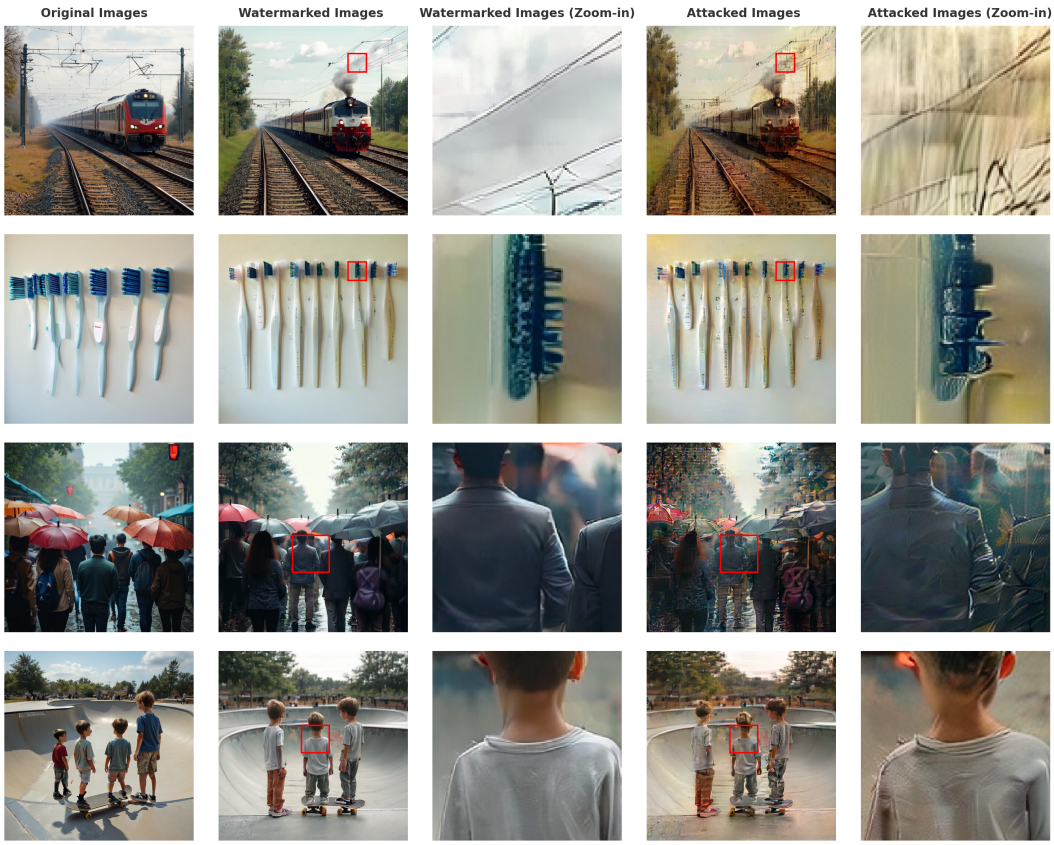

Figure 10: **The Bit-Flipper has strong impact on the image quality.** Here, we zoom in on some regions in the image to demonstrate the quality loss in details. $\phi = 2.2$, $\delta = 2$.