

Watermarking Autoregressive Image Generation

Nikola Jovanović^{1,2,†}, Ismail Labiad^{1,3}, Tomáš Souček¹, Martin Vechev², Pierre Fernandez¹

¹Meta FAIR, ²ETH Zurich, ³Université Paris-Saclay

[†]Work done during a Meta internship.

Watermarking the outputs of generative models has emerged as a promising approach for tracking their provenance. Despite significant interest in autoregressive image generation models and their potential for misuse, no prior work has attempted to watermark their outputs at the token level. In this work, we present the first such approach by adapting language model watermarking techniques to this setting. We identify a key challenge: the lack of reverse cycle-consistency (RCC), wherein re-tokenizing generated image tokens significantly alters the token sequence, effectively erasing the watermark. To address this and to make our method robust to common image transformations, neural compression, and removal attacks, we introduce (i) a custom tokenizer-detokenizer finetuning procedure that improves RCC, and (ii) a complementary watermark synchronization layer. As our experiments demonstrate, our approach enables reliable and robust watermark detection with theoretically grounded p-values.

Correspondence: nikola.jovanovic@inf.ethz.ch, pfz@meta.com

Code: <https://github.com/facebookresearch/wmar>

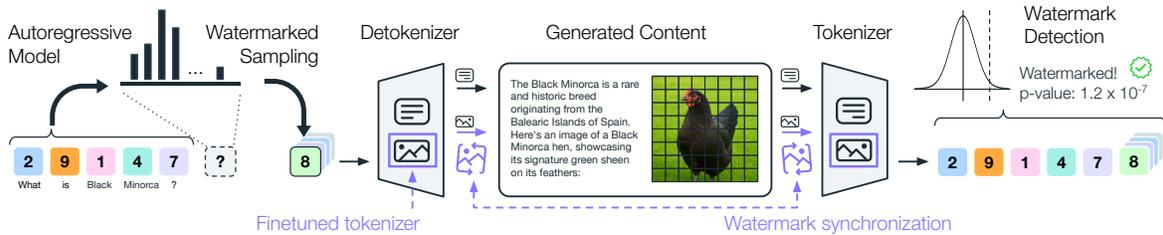


Figure 1 We watermark autoregressively generated images together with text in a theoretically principled way by adapting LLM watermarking. We identify and address the novel challenges present in this setting (Section 3) via a custom (de)tokenizer finetuning procedure (Section 3.1) and a watermark synchronization layer (Section 3.2).

1 Introduction

Autoregressive models are powerful frameworks for understanding and generating diverse content types. By converting multiple modalities into discrete representations via custom tokenizers [78, 97, 110], a single transformer is able to seamlessly process multiple domains, including text, images [11, 15, 93, 102], audio [8, 24], and even molecules [38]. Following patterns observed in large language models (LLMs), established scaling laws [2, 35, 86] demonstrate that the performance of these models improves predictably with size and computational resources, leading to increasing adoption across research and industry [21, 68, 72]. Most notably, in the image domain, autoregressive models are widely studied as an alternative to diffusion models for high-quality generation [76, 88, 94, 108].

Watermarking generative model outputs. Regardless of the specific method, the widespread deployment of high-quality generative models has made the detection of AI-generated content increasingly challenging. This has raised significant concerns about misuse, including deepfakes, harmful content generation, and intellectual property violations. One promising direction to help address these issues is *generative AI*

watermarking, in which the model provider proactively embeds imperceptible signals into generated content to verify its origin later, ideally even under common transformations and removal attacks.

Recent research in this area can be categorized into post-hoc methods, which modify already generated content in a model-agnostic way [9, 12, 22, 29, 83, 121], and modality-specific generation-time methods, which alter the generation process of a specific model [1, 20, 28, 46, 101, 105]. The latter are the standard in LLM watermarking, offering theoretically grounded watermark detection with provably low false positive rates. However, most image watermarking research focuses on diffusion models, and no prior work has attempted to adapt LLM watermarks to other token types, which could be a way to develop a robust and theoretically principled watermark that is tailored to autoregressive image generation models. This motivates our key question:

Can we robustly watermark autoregressive image generation models at the token level?

This work. To answer this question, we extend LLM watermarks to watermark autoregressively generated image tokens as illustrated in Figure 1. We identify and address a key technical challenge. Namely, while image tokenization is designed to be *forward cycle-consistent*, i.e., tokenizing and detokenizing an image does not significantly alter it, *reverse cycle-consistency* (RCC) is often violated. In particular, we show that decoding model-generated tokens and then re-tokenizing the resulting image leads to, on average, one-third of the tokens being different. The tokens differ even more if the images are transformed between generation and watermark detection (e.g., JPEG compressed or cropped), which is a common scenario in practice. While RCC may not be important for generative model performance, it is a crucial prerequisite to achieve strong and robust generation-time watermarking.

To mitigate this, we adopt two main strategies, shown in Figure 1. We introduce a lightweight finetuning procedure that optimizes the detokenizer and tokenizer to be more reverse cycle-consistent, improving watermark power and robustness to valuemetric transformations (e.g., JPEG) and attacks such as diffusion purification [71] and neural compression [5, 14, 25, 80]. To then improve robustness to geometric transformations (e.g., flips), we introduce a complementary post-hoc *watermark synchronization* layer, repurposing localized watermarking [83] to detect and revert geometric transformations and recover original tokens. As our experiments show, this results in a watermark that is quality-preserving, effective, and robust. To inspire future work, in Section 5 we take first steps to extend our approach to audio, another modality where autoregressive generation via tokenization is popular [8, 18, 24, 70].

Main contributions. We make the following key contributions:

- We conduct the first study of watermarking for outputs of autoregressive image generation models, adapting LLM watermarks to obtain theoretically principled p-values. We notably identify a key technical challenge, the lack of reverse cycle-consistency (RCC) (Section 3).
- We propose a finetuning procedure for image tokenizers and detokenizers that improves RCC and significantly increases watermark power and robustness (Section 3.1).
- We introduce a complementary post-hoc watermark synchronization layer, which achieves geometric robustness by leveraging off-the-shelf localized watermarking (Section 3.2).
- In our thorough experimental evaluation across several settings, we show that our watermark is strong, quality-preserving, and persistent under a wide range of valuemetric and geometric transformations and attacks such as diffusion purification and neural compression (Section 4).

2 Background and Related Work

Autoregressive image models. A long-studied approach to image generation, and the focus of our work, is to first learn an image tokenizer and then train a (conditioned) autoregressive model \mathcal{M} such as a transformer, to create images by generating corresponding token sequences [76, 88, 94, 96, 100, 108, 109]. Notable examples of such models include DALL-E [76], Parti [108], VAR [94], and RAR [109]. This approach is central to models for interleaved multimodal generation [11, 15, 31, 59, 60, 93, 102, 111], such

as Chameleon [11], AnyGPT [111], or Janus [102]. In this work, we do not consider other models that use continuous representations or combine diffusion with autoregressive mechanisms [26, 56, 66, 103, 120].

Tokenization. Formally, for a target modality m (in this work primarily text or image, but also audio in Section 5), a *tokenizer* \mathcal{T}_m maps each data sample x to a sequence of integer tokens $s = (s_1, \dots, s_T) \in V^T$, where V is the predefined vocabulary. The *detokenizer* \mathcal{D}_m attempts to reverse this process. Most text tokenizers are based on *byte-pair encoding* (BPE) [30]. While alternative approaches have been explored [56, 94], the tokenization of images overwhelmingly relies on vector quantization (VQ) [32, 56]. Most models use VQ-VAE [78, 97] or its variants VQGAN [25], ImprovedVQGAN [107], and FSQ [67]. VQ tokenizers generally consist of an encoder network E and a quantizer Q_C . E maps x to a sequence of *soft latents* $z = E(x) \in \mathbb{R}^{T \times d}$. Then, Q_C replaces each z_i with the index of the nearest entry in a *codebook* $C \in \mathbb{R}^{|V| \times d}$ to obtain discrete tokens $s \in V^T$:

$$s_i = Q_C(z_i) = \arg \min_{j \in \{1, \dots, k\}} \|z_i - C_j\|_2^2. \quad (1)$$

The detokenizer \mathcal{D}_m replaces each s_i with the corresponding $\hat{z}_i = C_{s_i}$ (*hard latents*), and then applies a decoder network D to obtain the detokenized sample $\hat{x} = D(\hat{z})$. All components (E , C , and D) are typically trained jointly, primarily with the reconstruction objective.

Watermarking AI-generated outputs. Methods for watermarking of AI-generated outputs broadly fall into two main categories based on the stage at which the watermark is introduced. *Generation-time watermarks* directly alter generations to embed statistically detectable patterns, and are the standard for LLM watermarking [1, 16, 46, 51]. Similar approaches also exist for diffusion models [28, 101, 105]. In contrast, *post-hoc watermarks* modify previously generated outputs in a modular model-agnostic way, by paraphrasing text [4, 12, 116] or altering image pixels [9, 10, 41, 63, 65, 90, 121].

For both of these categories, we say that a watermarking scheme is *zero-bit* if only the presence of the watermark can be detected, and *multi-bit* if it embeds a binary message. This message embedding is sometimes done in a *localized* way [83, 85], such that a different message can be extracted from different parts of the same data sample (e.g., one per pixel). While post-hoc watermarks have broad applicability, generation-time approaches that introduce semantic changes to the content often offer superior robustness to attacks such as diffusion purification [81] and provide provable, key-based guarantees on false positive rates (unlike neural extraction methods, where recovered bits may be biased or correlated [28, App. B.5]). To the best of our knowledge, no prior work attempts to develop a generation-time watermarking scheme for autoregressive image models.

LLM watermarking. In this work, we focus on the LLM watermark of Kirchenbauer et al. [46] (KGW, green/red scheme). At each step i of generation, this method uses a secret key ξ , and previous h tokens of *context* $s_{i-h:i}$ to pseudorandomly partition the vocabulary V of the tokenizer into $\gamma|V|$ *green* tokens \mathcal{G}_i and other *red* tokens \mathcal{R}_i . The logits corresponding to \mathcal{G}_i are then increased by δ , the *watermark strength*. The watermark detector computes the *score* $S = \sum_{i=h+1}^T \mathbb{1}(s_i \in \mathcal{G}_i)$ as the number of green tokens in the given sequence of T tokens. Under the null hypothesis \mathcal{H}_0 (no watermark), S follows a binomial distribution with parameters $T - h$ and γ . The p-value (often denoted as p) is calculated as:

$$\text{p-value}(S, T, h, \gamma) = \text{Prob}(X \geq S \mid X \sim \text{Binomial}(T - h, \gamma)). \quad (2)$$

A low p-value proves that the content was generated with \mathcal{M} . More details are provided in Appendix B.

3 Watermarking Autoregressive Image Generation

In this section, we present our approach to watermarking autoregressive image models. We identify and address the key challenge of low *reverse cycle-consistency* (RCC) via tokenization finetuning (Section 3.1) and watermark synchronization (Section 3.2). As our experiments in Section 4 demonstrate, this leads to a strong and robust watermark that does not affect generation quality.

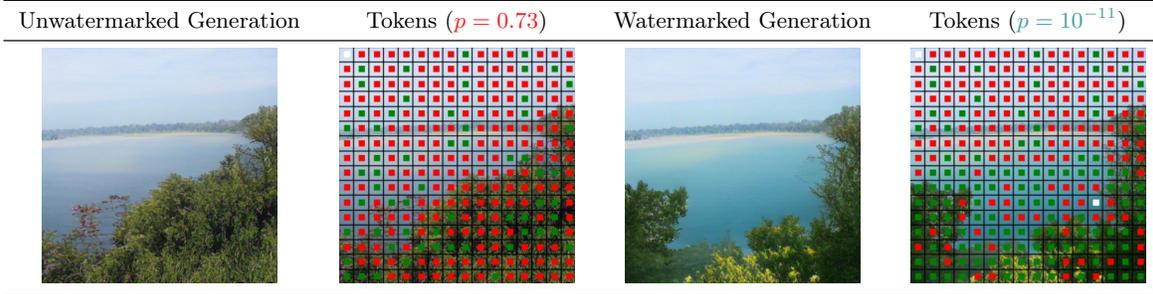


Figure 2 Example of our watermark on an autoregressively generated image. We generate the upper half of the image without the watermark. We then complete the bottom half in the same way (*left*) or with the watermark (*right*). The overlay indicates generated image tokens detected as green (■), red (■), or ignored as a duplicate (□). The watermark only alters semantics and could be detected even when applied only partially as in this case.

Setting. A model provider (Alice) deploys an autoregressive model \mathcal{M} that may generate arbitrarily interleaved text and images, using a tokenizer \mathcal{T}_m and detokenizer \mathcal{D}_m for each modality $m \in \{\text{text, image}\}$. In line with the most prominent choices, we assume BPE for text [30] and VQ for images [25, 97]. Alice’s goal is *out-of-model, generation-time, zero-bit watermarking* (see Section 2), i.e., embedding a later detectable watermark in *all* outputs of \mathcal{M} , without modifying the model’s weights. We assume that Bob has only black-box access to \mathcal{M} , and no access to any \mathcal{T}_m or \mathcal{D}_m .

Adapting LLM watermarking. When \mathcal{M} generates text, we directly apply KGW (Section 2) with $h = 1$. For images, using a fixed split ($h = 0$), known to make watermarks insecure for text, i.e., easy to reverse-engineer [42, 47, 118], may in our case be a more viable choice due to the opacity of the VQ tokenizers. We thus explore both $h \in \{0, 1\}$ in our experiments in Section 4. Another degree of freedom is the choice of watermark context—we did not find exploiting the 2D structure of images to be beneficial, despite the intuition that using tokens that correspond to spatially close regions as the watermark context (instead of those last generated by the autoregressive model) may benefit robustness. This idea and more values of watermark parameters are explored in Appendix F.2.

Detection. Given samples $x^{(i)}$ of varying modalities that Alice suspects were generated by \mathcal{M} (e.g., a post on a breed of hen as in Figure 1), she can apply Equation 2 in a unified way. We first tokenize each $x^{(i)}$ to $s^{(i)}$ of length $T^{(i)}$ tokens, and score it using the corresponding $h^{(i)}$ to obtain a score $S^{(i)}$. We next sum all $S^{(i)}$, $T^{(i)}$, and $h^{(i)}$, and deduplicate scored (context, token) pairs across all samples to preserve statistical soundness [27, 43, 46, 84]. Then, we apply Equation 2 to obtain a single p-value. Notably, the same γ must be used across all modalities. Alice may reject \mathcal{H}_0 (flag content as watermarked) if the p-value is below the desired false positive rate (FPR). In Section 4.3 we investigate the benefits of jointly watermarking multiple modalities, and discuss the involved tradeoffs.

In Figure 2, we visualize the watermark on an image generated with TAMING [25], by applying it only on the second half of generated tokens. As we later confirm in Section 4, the watermark imperceptibly modifies images by altering semantics, while achieving high watermark confidence (low p-value).

Challenge: reverse cycle-consistency (RCC). The tokens s shown in Figure 2 as input to the detector are those generated by the autoregressive model, which is not realistic. In practice, to apply the detector to a sample x' , Alice must first tokenize it as $s' = \mathcal{T}_m(x')$. If tokens s' significantly differ from s , the watermark may be lost. To quantify this, we define the *token match* as:

$$\text{TM}(s, s') = \frac{1}{T} \sum_{i=1}^T \mathbb{1}(s_i = s'_i), \quad (3)$$

where $s' = \mathcal{T}_m(\mathcal{D}_m(s))$. We say that *reverse cycle-consistency (RCC)* holds if $\text{TM}(s, s') \approx 1$. In Appendix C.3 we discuss similar concepts studied in prior work and relate them to RCC.

Table 1 Average token match between 1000 image token sequences generated with TAMING (see Section 4) and their re-tokenized versions, which may also undergo image transformations before re-tokenization.

Original	Blur $ksz = 9$	Noise $\sigma = 0.1$	JPEG $Q = 25$	Brighten $2\times$	Rotate 10°	Flip \leftrightarrow	Crop 0.75
0.66	0.26	0.17	0.31	0.11	0.02	0.01	0.01

Interestingly, RCC is not guaranteed to hold even in the text domain, despite BPE tokenizers ensuring *forward cycle-consistency (FCC)*, i.e., $\mathcal{D}_{\text{text}}(\mathcal{T}_{\text{text}}(x)) = x$ always holds. For example, if tokens A , B and their concatenation AB all exist in the vocabulary, then the token sequence $[A, B]$ will be re-tokenized as a single token AB due to the greedy tokenization algorithm of BPE, effectively violating RCC. In Appendix C.1 we discuss this in more detail and provide a real example. Still, RCC in text largely holds in practice, which is also evidenced by the fact that no prior LLM watermarking work has cited related challenges. We confirm this experimentally: across 1000 completions generated with LLAMA3.1-8B-INSTRUCT the average token match was 0.995.

RCC in image generation. We repeat this experiment on image models, presenting the results in Table 1 (full details in Appendix E.1). We also consider the case where images undergo transformations ($x \rightarrow a(x)$) before re-tokenization. Without transformations (*Original*), RCC is already weaker than expected with $\text{TM} = 0.66$. As Section 4 will show, this often suffices under ideal conditions (e.g., our example in Figure 2 has a p-value of 10^{-9} after re-tokenization). However, common valuemetric transformations (blur, noise, JPEG, brighten) lower TM (e.g., to 0.31 for JPEG with $Q = 25$), and geometric ones (rotate, flip, crop) cause a further drop to almost 0. Two key factors explain this behavior. First, neural image tokenizers are trained for FCC, not RCC. Their training data does not include detokenized samples, which often lie off-manifold. Second, spatial sensitivity of the tokenizer causes semantic-preserving edits to easily alter most tokens. We next show how to mitigate this.

3.1 Finetuning for Reverse Cycle-consistency

We propose a finetuning procedure (illustrated in Figure 3) that improves RCC in image tokenizers. Recall the VQ components (Section 2): encoder E , quantizer Q_C with codebook C , and decoder D . Let D_0, E_0 be the original weights of D and E . To avoid costly retraining of \mathcal{M} , we must keep (E, Q_C, C) fixed; otherwise, we risk modifying the codebook semantics (directly or by changing how images are encoded by E), which harms the autoregressive model. Thus, we propose to only finetune D and an encoder replica E' (initialized to E_0). E' is used only for watermark detection, while the original E may be used to condition \mathcal{M} on images. Unlike usual VQ training that promotes FCC, we optimize RCC: we aim to learn a decoder D whose outputs E' can reliably invert.

Finetuning objectives. We first precompute tokenizations s from a set of images, which we use as our training data. We encourage RCC by minimizing the following loss:

$$\mathcal{L}_{\text{RCC}}(s) = \mathbb{E}_{a \sim \mathcal{A}} \|\hat{z} - E'(a(D(\hat{z})))\|_2^2. \quad (4)$$

Its goal is to match the original *hard latents* $\hat{z} = C_s$ to *soft latents* obtained after detokenization and encoding using E' . To ensure RCC holds robustly even under transformations, we uniformly sample an augmentation $a \sim \mathcal{A}$ with preset probability p_{aug} in each training step, or set it to identity otherwise. Our augmentation set \mathcal{A} includes valuemetric (brighten, contrast, JPEG) and weak geometric transformations (e.g. $\pm 1^\circ$ rotation), with strength ramped up over training (see Appendix E.1).

To retain decoder quality we introduce a regularization that keeps the effect of D close to the effect of its initial weights D_0 via a mixture of MSE and LPIPS perceptual loss [115]:

$$\mathcal{L}_{\text{reg}}(s) = \|D(\hat{z}) - D_0(\hat{z})\|_2^2 + \mathcal{L}_{\text{LPIPS}}(D(\hat{z}), D_0(\hat{z})). \quad (5)$$

We found this sufficient as a quality constraint and easier to train compared to loss functions that compare reconstructions to the original images or the use of adversarial discriminators.

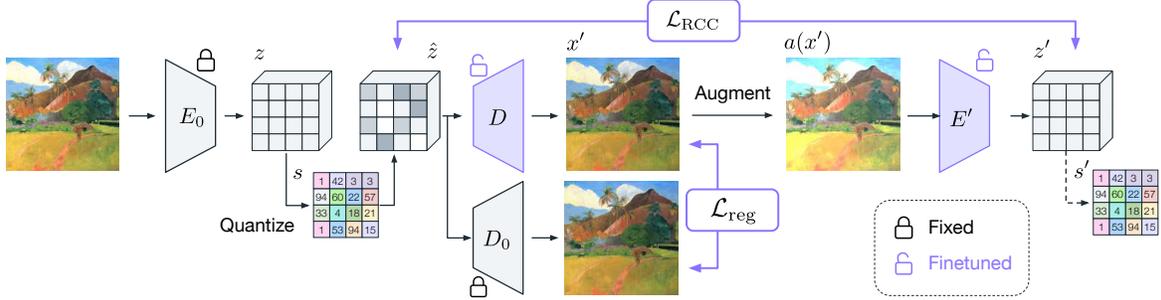


Figure 3 A replica E' of the encoder and the decoder D are jointly trained to improve reverse-cycle consistency, i.e., make $E'(D(\hat{z}))$ close to \hat{z} for most generations of the autoregressive model \mathcal{M} , even under transformations.

We jointly train D and E' to minimize:

$$\mathcal{L}(s) = \mathcal{L}_{\text{RCC}}(s) + \lambda \cdot \mathcal{L}_{\text{reg}}(s), \quad (6)$$

where λ is a tradeoff hyperparameter. In Section 4 we demonstrate that this efficiently boosts RCC and watermark robustness against valuemetric attacks, neural compression, and diffusion purification [71].

3.2 Post-hoc Watermark Synchronization

Semantic-preserving geometric transformations (e.g., flips) easily change image tokenization as each token loosely corresponds to a local image patch. Therefore, RCC finetuning alone cannot recover the watermark. One could run the watermark detector on multiple transformed image copies (rescaled, rotated, etc.), but this is costly and significantly inflates false positives as noted in prior work [46].

Localization as synchronization. To address this in a more practical way, we repurpose localized watermarks (see Section 2) as a synchronization signal. More precisely, we locally embed a fixed set of messages whose detection estimates the applied transform, which we then aim to invert before applying the original watermark detector. To not degrade original performance, our watermark should be robust to the addition of this signal, which we verify in Section 4. Detecting this signal could in principle be taken as evidence that the image is watermarked, as in some of the prior post-hoc watermarking schemes that explore synchronization [33, 64]. However, applying the original watermark detector is still necessary to obtain theoretically grounded p-values that can be combined with other samples across modalities as described above. Moreover, as we will see in Section 4, post-hoc watermarks are generally much more brittle to adversarial purification compared to the approach we propose.

Reverting transformations. In Figure 4, we show our instantiation of this idea, where we embed four 32-bit synchronization messages $\{0^{32}, 0^{16}1^{16}, 1^{16}0^{16}, 1^{32}\}$ via the method of Sander et al. [85] into the four image quadrants. We observe that the original watermark is well preserved (with p-value $p = 5 \cdot 10^{-38}$) after adding the synchronization signal. However, a horizontal flip shuffles tokens and breaks detection ($p = 0.66$). To identify this, we apply an algorithm that searches over a grid of rotation angles, and for each fits the best axis-aligned pair of orthogonal lines that separate the four messages. This is sufficient to identify and revert flips, rotations, and crops followed by upscaling to the respective model’s original generation size, which we revert by downscaling and padding appropriately. We note that our synchronization pattern (quadrants) assumes crops that preserve one corner—as we discuss in Section 6, the same idea can be directly extended to arbitrary crops by using a more elaborate synchronization pattern. In our example in Figure 4, our procedure detects that a flip was applied and restores $p = 5 \cdot 10^{-38}$. Full algorithm and more examples are deferred to Appendix D.1.

In the following, we empirically show that synchronization enhances geometric robustness, complementing RCC finetuning. This step is further aided by the use of small geometric augmentations during RCC finetuning, as they effectively compensate for minor errors in our transformation estimates.

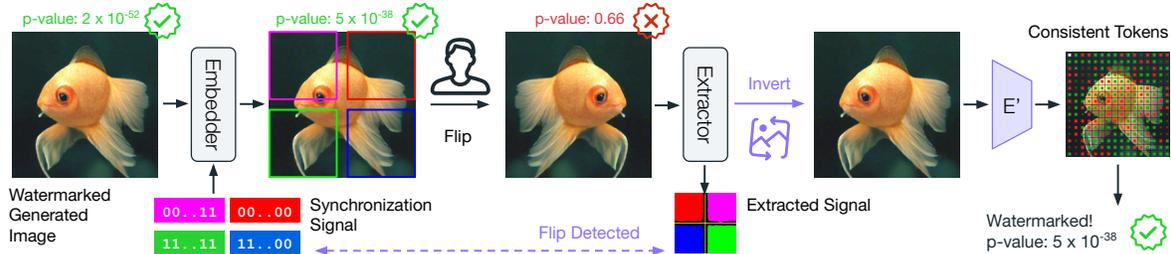


Figure 4 Watermark synchronization. Localized messages are embedded into a generated watermarked image and later used to discover the unknown transformation and revert it, which recovers the original watermark.

4 Experimental Evaluation

In [Section 4.1](#), we measure the effect of RCC finetuning ([Section 3.1](#)) and the synchronization layer ([Section 3.2](#)) on RCC, quality, and the power of our watermark. [Section 4.2](#) studies robustness under common transformations and attacks, while [Section 4.3](#) studies joint watermarking of text and images. Additional experimental details and results are given in [Appendix E](#) and [Appendix F](#), respectively.

Setup. We consider three autoregressive image generation models. First, the class-conditional ImageNet transformer from Esser et al. [25] that generates images at resolution 256×256 with a VQGAN tokenizer with $|V| = 16384$ and downsampling factor $f = 16$, denoted TAMING below. Next, the 7B variant of the mixed-modal CHAMELEON [11] that can generate interleaved text and 512×512 images, with $|V| = 8192$ and $f = 16$. Finally, RAR-XL, the 955M configuration of the state-of-the-art RAR model [109], generating 256×256 images conditioned on an ImageNet class, as TAMING, with $|V| = 1024$ and $f = 16$. In each experiment, we generate 1000 samples per model (100 samples per each of 10 ImageNet classes for TAMING and RAR-XL, and 1000 COCO prompts for CHAMELEON).

We evaluate 4 variants of our method: BASE, which uses original models and tokenizers, FT and FT+AUGS, which apply the same watermark after RCC finetuning ([Section 3.1](#)) without and with augmentations in training, respectively, and FT+AUGS+SYNC, which also uses our watermark synchronization ([Section 3.2](#)) on top of augmented-RCC finetuning. We use $\delta = 2$ and $\gamma = 0.25$ in all experiments, $h = 1$ for TAMING, RAR-XL, and CHAMELEON on text, and $h = 0$ for CHAMELEON on images. We finetune models on tokens derived from 50,000 ImageNet training samples for 10 epochs (2h on 16 V100 for TAMING, 2.5h on 8 H200 for CHAMELEON, and 0.5h on 8 H200 for RAR-XL).

4.1 Reverse Cycle-Consistency, Watermark Power, and Generation Quality

The key question raised in [Section 3](#) is if our proposed finetuning procedure can alleviate the lack of reverse cycle-consistency (RCC) in image tokenizers, and in turn improve watermark power. We also measure the effect of watermarking and finetuning on generation quality. In the following, we present results with TAMING. In [Appendix F.3](#), we repeat the same experiments on CHAMELEON and RAR-XL with the same finetuning hyperparameters (see [Appendix E.1](#)) and reach similar conclusions.

Finetuning improves RCC. We generate 1000 class-conditioned ImageNet samples using each of our 4 variants, and measure token match (TM, [Equation 3](#)) between the generated tokens and those obtained by re-tokenizing the image. In our results in [Figure 5](#) (left), we observe that TM is consistently below 0.8, as previously seen in [Table 1](#), while for all finetuned variants it is generally *above* 0.8. This demonstrates that finetuning is successful in improving RCC. AUGS and SYNC slightly reduce TM on unmodified images, but significantly increase robustness (see [Section 4.2](#)).

Finetuning improves watermark power. In [Figure 5](#) (right), we report the true positive rate (TPR) of the watermark detector for different false positive rates (FPR). The BASE variant already has practically viable power, achieving TPR of ≈ 1 at FPR of 10^{-2} (dashed line), the setting commonly considered in

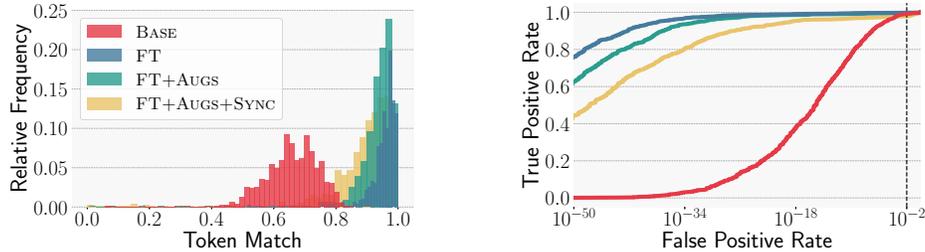


Figure 5 *Left*: Finetuning improves token match (Equation 3) between original and re-tokenized image tokens. *Right*: All variants achieve TPR ≈ 1 at FPR of 1%. Finetuning further boosts detection in low-FPR settings.

prior work [17, 20, 101, 119]. However, RCC gains directly translate to improvements in watermark power: for all 3 variants, the TPR at lower FPR settings is significantly higher.

Watermarking and finetuning do not harm generation quality. To measure the quality of the generated samples, we compute FID [36] on 50,000 generations (50 per ImageNet-1K class) for all variants. We find that none of BASE, FT, and FT+AUGS have FID above 16.7, which is the FID of an *unwatermarked* BASE model. This confirms that our watermark preserves generation quality even after finetuning. The FID of FT+AUGS+SYNC is 17.3, a minor increase inherited from the localized watermark used for synchronization. We complement this with qualitative samples on all models in Appendix G.

4.2 Watermark Robustness

An important requirement for a generative model watermark is robustness to common domain-specific transformations, as well as to removal attacks, which have shown to be effective against other watermarks [3, 28, 81]. To evaluate this, in Figure 6 we report the watermark TPR for a fixed FPR of 1% on a range of transformations of different strength, in the same setting as in Figure 5, using TAMING (the equivalent results for CHAMELEON and RAR-XL are deferred to Appendix F.3).

We summarize the results of this experiment for all three models in Table 2, where, as in prior work [101], we average TPR over a set of (transformation, parameter) pairs, detailed in Appendix E.2. Our main conclusions, discussed next, hold across all models, despite the variations in e.g., transformer or tokenizer codebook size, which demonstrates the transferability of our approach.

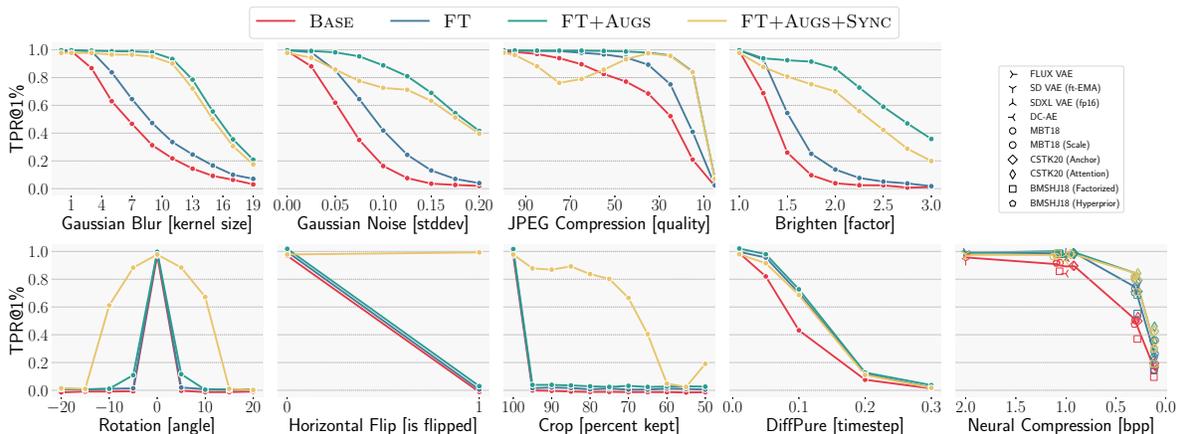


Figure 6 *Top*: RCC finetuning improves robustness to valuemetric transformations. *Bottom Left*: Watermark synchronization unlocks robustness to geometric transformations. *Bottom Right*: Our watermark is also fairly robust to realistic strengths of diffusion purification [71, 81] and neural compression [5, 13, 14, 53, 69, 75, 80].

Table 2 TPR at 1% FPR of our watermark and post-hoc baselines, under valuemetric (**Val.**) and geometric (**Geo.**) transformations, adversarial purification attacks (**Adv.**) and neural compression (**NC**). Scores (see [Appendix E.2](#)) below 0.6 are marked **red**. RCC finetuning and the synchronization layer lead to a strong and robust watermark.

		TAMING (256 × 256)					CHAMELEON (512 × 512)					RAR-XL (256 × 256)				
		None	Val.	Geo.	Adv.	NC	None	Val.	Geo.	Adv.	NC	None	Val.	Geo.	Adv.	NC
Ours	BASE	0.99	0.26	0.01	0.43	0.48	0.98	0.50	0.02	0.80	0.82	0.98	0.22	0.01	0.29	0.27
	FT	1.00	0.45	0.01	0.70	0.71	0.99	0.53	0.03	0.85	0.87	1.00	0.74	0.03	0.58	0.76
	FT+AUGS	1.00	0.92	0.01	0.70	0.79	0.99	0.89	0.02	0.82	0.88	1.00	0.98	0.03	0.79	0.95
	FT+AUGS+SYNC	0.98	0.83	0.82	0.69	0.80	0.97	0.76	0.64	0.81	0.86	1.00	0.89	0.82	0.78	0.94
Post-hoc	CIN	1.00	0.96	0.00	0.03	0.02	1.00	0.99	0.00	0.14	0.16	1.00	0.95	0.00	0.04	0.01
	MBRS	1.00	0.98	0.02	0.36	0.31	1.00	0.99	0.02	0.27	0.56	1.00	0.99	0.01	0.38	0.34
	TRUSTMARK	1.00	0.98	0.75	0.40	0.86	1.00	0.97	0.74	0.64	0.99	1.00	0.99	0.75	0.42	0.90
	WAM	1.00	0.89	0.98	0.06	0.02	1.00	0.97	0.95	0.26	0.48	1.00	0.92	0.98	0.08	0.02

Finetuning enables valuemetric and attack robustness. In [Figure 6](#), we see that the watermark is fragile to valuemetric and geometric transformations when used on BASE. When we use it on FT+AUGS, robustness to valuemetric transformations greatly improves, validating our focus on RCC finetuning. Surprisingly, finetuning also improves robustness to (i) neural compressors [[5](#), [14](#), [69](#)] of different strengths (see details in [Appendix E.2](#)), including FLUX and SD VAEs [[13](#), [53](#), [75](#), [80](#)], and (ii) the challenging diffusion purification attack [[71](#)]. We remark that high values such as $t = 0.3$ were found to excessively alter images, making this regime less relevant [[81](#)]. This effect holds for RAR-XL but is less pronounced for CHAMELEON in [Table 2](#), where our watermark is already robust to these attacks even without RCC finetuning, likely due to the detector scoring more tokens for larger images.

Synchronization enables geometric robustness. Geometric robustness (*bottom left* in [Figure 6](#)) remains low across all models, even with FT+AUGS, which is expected as autoregressive models cannot preserve token sequences under semantic changes like flips, and the watermark is lost. This motivated our synchronization layer ([Section 3.2](#)), which estimates and reverts such changes. As [Figure 6](#) and [Table 2](#) show, synchronization significantly improves geometric robustness while preserving watermark power on unmodified images. However, this comes at the cost of a minor drop in valuemetric robustness. Namely, moderate valuemetric transformations sometimes disrupt the synchronization signal, causing us to estimate and revert a non-existent geometric transformation. Such a mistake corrupts most tokens and breaks the watermark. This does not happen for strong valuemetric transformations that often fully destroy the synchronization signal, as in this case we keep the image intact. A more elaborate and robust synchronization layer could likely minimize this undesirable effect while further extending the scope of supported transformations; we propose several concrete directions in [Section 6](#).

Comparison to post-hoc methods. As noted above, no prior work targets watermarking of autoregressive image generation. Thus, in [Table 2](#) we compare to *post-hoc* methods [[9](#), [41](#), [65](#), [85](#)] applied on top of generated images. While they are comparably or more robust than our watermark on valuemetric transformations, each post-hoc watermark is either fully removed by geometric ones or not robust to attacks (adversarial purification and neural compressors). Further, our watermark yields p-values grounded in randomness, with theoretical guarantees inherited from LLM watermarking [[27](#), [47](#), [119](#)] and empirically validated in [Appendix F.5](#). In contrast, post-hoc methods use neural extractors to recover messages and may introduce bias in their p-value estimators [[28](#), App. B.5] and [[83](#), App. B]. Finally, as a token-level generation-time method, our method is the only one able to watermark content via semantic modifications (see [Figure 2](#)). In [Appendix F.6](#) we provide an additional comparison to generation-time watermarks for diffusion models, despite their inapplicability to our target models.

4.3 Joint Watermarking of Interleaved Modalities

Finally, we explore joint watermarking of multiple modalities generated by the same autoregressive model. [Equation 2](#) shows that scoring more equally watermarked tokens improves power. However, acquiring more tokens is not always possible—in such cases, jointly watermarking multiple modalities may be necessary to reliably detect the watermark. For example, consider that Alice aims to prove if

an online article was generated by her model \mathcal{M} . To simulate this, we run CHAMELEON (FT+AUGS) in interleaved mode on 1000 prompts to produce text and an image, and we model attempts to conceal the use of \mathcal{M} by randomly changing text tokens (a proxy for paraphrasing [50]).

Benefits of joint watermark detection. The orange line in Figure 7 shows TPR at 1% FPR when only text is watermarked, quickly degrading with more text corruption. As all suspect text is used, it is hard for Alice to improve detection in this scenario. However, if both text and the image were originally watermarked with our method, detection on combined tokens as described in Section 3 significantly boosts watermark power (top purple, *Clean*). At 10% text corruption TPR improves from ≈ 0.9 to 1.0, and stays above 0.94 even in the hardest case, where text-only TPR drops to ≈ 0 . Alice also gets a rigorous p-value, which would be hard if modalities were watermarked separately. As our method is robust to moderate image transformations, a similar trend holds when adding Gaussian noise with $\sigma = 0.1$ (middle purple, *Weak Noise*).

Importantly, there is a tradeoff—*integrating a weak watermarking signal can degrade detection*. We see this for $\sigma = 0.3$ (bottom purple, *Strong Noise*), where TPR drops below 0.6 at 10% corruption, i.e., text-only detection is preferable. In Appendix H we provide a more detailed discussion and present extended results and example interactions with CHAMELEON. Inspired by this, in the following section we explore the extension of our method to additional modalities.

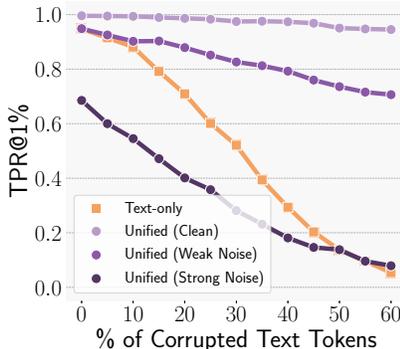


Figure 7 Joint watermark detection on text and image generations.

5 Extension to Additional Modalities: Audio Case Study

In this final section we ask: *Can our approach be extended to other modalities?* In an attempt to answer this, we conduct a preliminary study on autoregressive audio generation [8, 18, 24, 70, 113, 117] focusing on MOSHI [24], a transformer-based speech-text foundation model. We observe similar challenges and main results as for images, while noting several important differences. We defer many details to Appendix C.2 (RCC experiment), Appendix D.2 (synchronization details), Appendix E.3 (experimental details), and present extended results of audio experiments in Appendix F.7.

Audio tokenization. MOSHI’s tokenizer (MIMI) relies on *residual* vector quantization (RVQ) [23, 52, 55, 110]. RVQ iteratively quantizes the residuals of the previous quantizer, such that $s_i = (s_i^1, \dots, s_i^K)$ for K different codebooks C^1, \dots, C^K (K streams) Each token here represents ≈ 80 ms.

RCC in audio. Défossez et al. [24, Sec. 6.4] already investigate the RCC properties of MOSHI’s tokenizer. They observe that the first stream is somewhat cycle-consistent, while this degrades for later ones. We obtain similar results and also show that TM further worsens under transformations (highpass, speedup). For instance, in an experiment on 1000 generated sequences, we measure average TM of 0.36 (original), 0.21 (highpass 500 Hz), and 0.16 ($1.1\times$ speedup). This motivates an adaptation of RCC finetuning (Section 3.1) and synchronization (Section 3.2) to audio.

RCC finetuning. To instantiate the finetuning procedure from Section 3.1 we make the following changes to Equations 4 and 5. We use the pre-projection soft latents as target since the quantization is done in a projected space [52]. We replace LPIPS with multi-resolution STFT loss [104]. Finally, during training we apply augmentations from a set \mathcal{A} that includes audio-specific valuemetric edits (high/low/bandpass, gaussian/pink noise, etc.) as well as small (1-10 ms) time-frequency shifts.

Synchronization. In contrast to images, no localized audio embedder proved robust: we found San Roman et al. [83] to not be precise enough under time-frequency edits (see Appendix D.2). Thus, our following experiment focuses primarily on the influence of finetuning. Future work on audio-specific localized watermarking could improve on this, motivated by the synchronization usecase.

Watermarking multiple streams. Early RVQ streams are more reverse cycle-consistent and thus more likely to preserve the watermark signal. We observe however that limiting watermarking to a single stream provides too few tokens for reliable statistical testing, significantly increasing p-values. On the other hand, watermarking all streams introduces noise due to the lack of RCC in later codes. Empirically, we find that watermarking the first four streams achieves a good balance.

Experimental setting. We perform RCC finetuning to obtain FT and FT+AUQS using the VoxPopuli [99] dataset, such that final PESQ [79] (perceptual speech quality metric) is 4.3 w.r.t. BASE samples. For evaluation, we generate 12s watermarked audio samples with MOSHI using 1000 text prompts generated by LLAMA3.1-8B-INSTRUCT and synthesized to audio with SEAMLESSV2 [6]. We set $h = 0$ and $\delta = 2$ and watermark the first four audio streams as described above. As in Section 4, we evaluate TPR at 1% FPR and quality, for which in this case we use the MOSNet [24] metric.

Results. We present the results in Table 3. As for images, we do not observe notable quality degradation due to watermarking, and obtain high watermark strength even without RCC finetuning. We find that BASE already has nonzero time-frequency robustness, likely due to non-semantic streams being used to carry the watermark. Finetuning without augmentations, interestingly, impairs time-frequency RCC, which is later recovered by FT+AUQS. We hypothesize that this drop is due to catastrophic forgetting [48] as the model learns to detokenize the audio in a way that is not robust to time-frequency

transformations. This suggests that augmentations are a key component of finetuning, matching our results on images. Yet, while FT+AUQS improves valuemetric robustness, it fails to raise time-frequency robustness to satisfactory levels, leaving this question open for future work. Finally, while we are not aware of audio equivalents of diffusion purification used in Section 4, we evaluate robustness to neural compression (DAC [52] and EnCodec [23]). We observe results comparable to post-hoc AUDIOSEAL [83], even though in contrast to AUDIOSEAL we do not explicitly train against EnCodec during finetuning.

Table 3 TPR at 1% FPR and MOSNet [62] of our watermark, under valuemetric (**Val.**), time-frequency (**Time**) transformations, and neural compression (**NC**). MOSNet is 3.80 for unwatermarked generation.

		None	Val.	Time	NC	MOSNet
Ours	BASE	0.97	0.62	0.24	0.80	3.82
	FT	0.99	0.64	0.14	0.84	3.83
	FT+AUQS	0.99	0.80	0.24	0.86	3.73
Post-hoc	AUDIOSEAL	1.00	0.84	0.55	0.85	3.78

6 Conclusion and Limitations

Our work successfully applies watermarking to the previously unexplored setting of autoregressive image generation, addressing low reverse cycle-consistency (RCC) through a custom finetuning stage and a synchronization layer. Experiments demonstrate the power, robustness, and practicality of our watermark across a range of settings. By broadening the scope of generative model watermarking, we believe this work takes an important step towards more reliable content provenance.

Limitations. Our method’s scope could be extended in several important ways. As noted in Section 2, we target the most prominent models that tokenize images via VQ. Our method is thus not applicable to models that use continuous representations or hybrids that combine autoregressive and diffusion models [26, 56, 66, 120]. Another dimension is modality: we present initial audio experiments in Section 5, but this direction could be investigated further, e.g., finding ways to improve time-frequency robustness. Next, our synchronization relies on off-the-shelf localized watermarks to embed a fixed pattern (*quadrants*, see Figure 4), which is suboptimal for several reasons. First, localized watermarks are trained to embed arbitrary patterns. Training a custom synchronization layer with a known pattern, potentially integrated with RCC finetuning, would be a more principled approach that could significantly improve robustness. Second, using more elaborate patterns could both improve the quality of our transformation estimation step, but also expand its scope, e.g., to support arbitrary crops as discussed in Section 3.2. Finally, our method is not robust to combined removal attacks (to disrupt synchronization) and geometric attacks (to decrease token match)—to the best of our knowledge, this attack would also break most other contemporary watermarks. We leave this question open for future work.

References

- [1] Scott Aaronson and Hendrik Kirchner. Watermarking gpt outputs, 2023.
- [2] Armen Aghajanyan, Lili Yu, Alexis Conneau, Wei-Ning Hsu, Karen Hambardzumyan, Susan Zhang, Stephen Roller, Naman Goyal, Omer Levy, and Luke Zettlemoyer. Scaling laws for generative mixed-modal language models. In *ICML*, 2023.
- [3] 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 *ICML*, 2024.
- [4] Dara Bahri and John Wieting. A watermark for black-box language models. *arXiv preprint arXiv:2410.02099*, 2024.
- [5] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *ICLR*, 2018.
- [6] Loïc Barrault, Yu-An Chung, Mariano Cora Meglioli, David Dale, Ning Dong, Paul-Ambroise Duquenne, Hady Elsahar, Hongyu Gong, Kevin Heffernan, John Hoffman, et al. Seamlessm4t-massively multilingual & multimodal machine translation. *arXiv preprint arXiv:2308.11596*, 2023.
- [7] Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. Compressai: a pytorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029*, 2020.
- [8] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matthew Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. Audiolm: A language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2022.
- [9] Tu Bui, Shruti Agarwal, and John Collomosse. Trustmark: Universal watermarking for arbitrary resolution images. *arXiv preprint arXiv:2311.18297*, 2023.
- [10] Tu Bui, Shruti Agarwal, Ning Yu, and John Collomosse. Rosteals: Robust steganography using autoencoder latent space. In *CVPR*, 2023.
- [11] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024.
- [12] Yapei Chang, Kalpesh Krishna, Amir Houmansadr, John Wieting, and Mohit Iyyer. Postmark: A robust blackbox watermark for large language models. In *EMNLP*, 2024.
- [13] Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. *arXiv*, 2024.
- [14] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *CVPR*, 2020.
- [15] Ethan Chern, Jiadi Su, Yan Ma, and Pengfei Liu. Anole: An open, autoregressive, native large multimodal models for interleaved image-text generation. *arXiv preprint arXiv:2407.06135*, 2024.
- [16] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *Cryptology ePrint Archive*, 2023.
- [17] Hai Ci, Pei Yang, Yiren Song, and Mike Zheng Shou. Ringid: Rethinking tree-ring watermarking for enhanced multi-key identification. *arXiv preprint arXiv:2404.14055*, 2024.
- [18] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. *NeurIPS*, 2024.
- [19] Gabriela Csurka, Frédéric Deguillaume, Joseph Ó Ruanaidh, and Thierry Pun. A bayesian approach to affine transformation resistant image and video watermarking. In *Information Hiding*, 1999.
- [20] Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, Jamie Hayes, Nidhi Vyas, Majd Al Mery, Jonah Brown-Cohen, Rudy Bunel, Borja Balle, Taylan Cemgil, Zahra Ahmed, Kitty

- Stacpoole, Ilya Shumailov, Ciprian Baetu, Sven Gowal, Demis Hassabis, and Pushmeet Kohli. Scalable watermarking for identifying large language model outputs. *Nature*, 2024.
- [21] DeepMind. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- [22] Google DeepMind. Identifying ai-generated images with synthid, 2023. Accessed on Jun 15, 2025.
- [23] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.
- [24] Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024.
- [25] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.
- [26] Lijie Fan, Tianhong Li, Siyang Qin, Yuanzhen Li, Chen Sun, Michael Rubinstein, Deqing Sun, Kaiming He, and Yonglong Tian. Fluid: Scaling autoregressive text-to-image generative models with continuous tokens. *arXiv preprint arXiv:2410.13863*, 2024.
- [27] Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. Three bricks to consolidate watermarks for large language models. In *IEEE WIFS*, 2023.
- [28] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *ICCV*, 2023.
- [29] Pierre Fernandez, Hady Elsahar, I Zeki Yalniz, and Alexandre Mourachko. Video seal: Open and efficient video watermarking. *arXiv preprint arXiv:2412.09492*, 2024.
- [30] Philip Gage. A new algorithm for data compression. *C Users J.*, 1994.
- [31] Yuying Ge, Sijie Zhao, Ziyun Zeng, Yixiao Ge, Chen Li, Xintao Wang, and Ying Shan. Making llama see and draw with seed tokenizer. *arXiv preprint arXiv:2310.01218*, 2023.
- [32] Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.
- [33] Hengchang Guo, Qilong Zhang, Junwei Luo, Feng Guo, Wenbin Zhang, Xiaodong Su, and Minglei Li. Practical deep dispersed watermarking with synchronization and fusion. In *ACM MM*, 2023.
- [34] Frank Hartung, Jonathan Su, and Bernd Girod. Spread spectrum watermarking: Malicious attacks and counterattacks. *Security and Watermarking of Multimedia Contents*, 2000.
- [35] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.
- [36] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.
- [37] Abe Bohan Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. Semstamp: A semantic watermark with paraphrastic robustness for text generation. *arXiv preprint arXiv:2310.03991*, 2023.
- [38] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. In *ICML*, 2022.
- [39] Yufang Huang, Wentao Zhu, Deyi Xiong, Yiye Zhang, Changjian Hu, and Feiyu Xu. Cycle-consistent adversarial autoencoders for unsupervised text style transfer. In *COLING*, 2020.
- [40] Yufang Huang, Wentao Zhu, Deyi Xiong, Yiye Zhang, Changjian Hu, and Feiyu Xu. Cycle-consistent adversarial autoencoders for unsupervised text style transfer. *arXiv preprint arXiv:2010.00735*, 2020.
- [41] Zhaoyang Jia, Han Fang, and Weiming Zhang. MBRS: enhancing robustness of dnn-based watermarking by mini-batch of real and simulated JPEG compression. In *ACM Multimedia*, 2021.
- [42] Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. In *ICML*, 2024.

- [43] Nikola Jovanović, Robin Staab, Maximilian Baader, and Martin Vechev. Ward: Provable rag dataset inference via llm watermarks. In *ICLR*, 2025.
- [44] Jun-Hyuk Kim, Soobeom Jang, Jun-Ho Choi, and Jong-Seok Lee. Instability of successive deep image compression. In *ACM MM*, 2020.
- [45] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.
- [46] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. *ICML*, 2023.
- [47] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. On the reliability of watermarks for large language models. In *ICLR*, 2024.
- [48] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017.
- [49] Andreas Köpf, Yannic Kilcher, Dimitri Von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richard Nagyfi, et al. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36: 47669–47681, 2023.
- [50] Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *NeurIPS*, 2023.
- [51] Rohith Kudritipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023.
- [52] Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. *NeurIPS*, 2023.
- [53] Black Forest Labs. Flux. <https://github.com/black-forest-labs/flux>, 2024.
- [54] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.
- [55] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *CVPR*, 2022.
- [56] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *NeurIPS*, 2024.
- [57] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.
- [58] 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 *ECCV*. Springer, 2014.
- [59] Xi Victoria Lin, Akshat Shrivastava, Liang Luo, Srinivasan Iyer, Mike Lewis, Gargi Ghosh, Luke Zettlemoyer, and Armen Aghajanyan. Moma: Efficient early-fusion pre-training with mixture of modality-aware experts. *arXiv preprint arXiv:2407.21770*, 2024.
- [60] Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. World model on million-length video and language with blockwise ringattention. *arXiv preprint arXiv:2402.08268*, 2024.
- [61] Wenrui Liu, Zhifang Guo, Jin Xu, Yuanjun Lv, Yunfei Chu, Zhou Zhao, and Junyang Lin. Analyzing and mitigating inconsistency in discrete audio tokens for neural codec language models. *arXiv preprint arXiv:2409.19283*, 2024.
- [62] Chen-Chou Lo, Szu-Wei Fu, Wen-Chin Huang, Xin Wang, Junichi Yamagishi, Yu Tsao, and Hsin-Min Wang. Mosnet: Deep learning based objective assessment for voice conversion. In *Proc. Interspeech 2019*, 2019.
- [63] Xiyang Luo, Ruohan Zhan, Huiwen Chang, Feng Yang, and Peyman Milanfar. Distortion agnostic deep watermarking. In *CVPR*, 2020.

- [64] Xiyang Luo, Michael Goebel, Elnaz Barshan, and Feng Yang. Leca: A learned approach for efficient cover-agnostic watermarking. *arXiv preprint arXiv:2206.10813*, 2022.
- [65] Rui Ma, Mengxi Guo, Yi Hou, Fan Yang, Yuan Li, Huizhu Jia, and Xiaodong Xie. Towards blind watermarking: Combining invertible and non-invertible mechanisms. In *ACM Multimedia*, 2022.
- [66] Yiyang Ma, Xingchao Liu, Xiaokang Chen, Wen Liu, Chengyue Wu, Zhiyu Wu, Zizheng Pan, Zhenda Xie, Haowei Zhang, Liang Zhao, et al. Janusflow: Harmonizing autoregression and rectified flow for unified multimodal understanding and generation. *arXiv preprint arXiv:2411.07975*, 2024.
- [67] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023.
- [68] AI Meta. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. <https://ai.meta.com/blog/llama-4-multimodal-intelligence/>, 2025.
- [69] David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *NeurIPS*, 2018.
- [70] Tu Anh Nguyen, Benjamin Muller, Bokai Yu, Marta R Costa-Jussa, Maha Elbayad, Sravya Popuri, Christophe Ropers, Paul-Ambroise Duquenne, Robin Algayres, Ruslan Mavlyutov, et al. Spirit-lm: Interleaved spoken and written language model. *TACL*, 2025.
- [71] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion models for adversarial purification. *arXiv preprint arXiv:2205.07460*, 2022.
- [72] OpenAI. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.
- [73] Patrick O’Reilly, Prem Seetharaman, Jiaqu Su, Zeyu Jin, and Bryan Pardo. Code drift: Towards idempotent neural audio codecs. In *IEEE ICASSP*, 2025.
- [74] Shelby Pereira, Joseph Ó Ruanaidh, and Thierry Pun. Secure robust digital watermarking using the lapped orthogonal transform. In *Security and Watermarking of Multimedia Contents*, 1999.
- [75] 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. In *ICLR*, 2024.
- [76] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021.
- [77] 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.
- [78] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *NeurIPS*, 2019.
- [79] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*. IEEE, 2001.
- [80] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.
- [81] 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. *ICLR*, 2024.
- [82] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.
- [83] Robin San Roman, Pierre Fernandez, Hady Elsahar, Alexandre Défossez, Teddy Furon, and Tuan Tran. Proactive detection of voice cloning with localized watermarking. In *ICML*, 2024.
- [84] Tom Sander, Pierre Fernandez, Alain Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. In *NeurIPS*, 2024.

- [85] Tom Sander, Pierre Fernandez, Alain Durmus, Teddy Furon, and Matthijs Douze. Watermark anything with localized messages. *ICLR*, 2025.
- [86] Mustafa Shukor, Enrico Fini, Victor Guilherme Turrissi da Costa, Matthieu Cord, Joshua Susskind, and Alaaeldin El-Nouby. Scaling laws for native multimodal models scaling laws for native multimodal models. *arXiv preprint arXiv:2504.07951*, 2025.
- [87] Kaiser Sun, Peng Qi, Yuhao Zhang, Lan Liu, William Yang Wang, and Zhiheng Huang. Tokenization consistency matters for generative models on extractive NLP tasks. In *EMNLP (Findings)*, 2023.
- [88] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.
- [89] Cees H Taal, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In *IEEE ICASSP*, 2010.
- [90] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *CVPR*, 2020.
- [91] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An instruction-following Llama model, 2023.
- [92] Yunfei Teng and Anna Choromanska. Invertible autoencoder for domain adaptation. *Comput.*, (2), 2019.
- [93] Changyao Tian, Xizhou Zhu, Yuwen Xiong, Weiyun Wang, Zhe Chen, Wenhai Wang, Yuntao Chen, Lewei Lu, Tong Lu, Jie Zhou, et al. Mm-interleaved: Interleaved image-text generative modeling via multi-modal feature synchronizer. *arXiv preprint arXiv:2401.10208*, 2024.
- [94] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *NeurIPS*, 2024.
- [95] Andrew Z. Tirkel, Charles F. Osborne, and Thomas E. Hall. Image and watermark registration. *Signal Process.*, (3), 1998.
- [96] Michael Tschanen, Cian Eastwood, and Fabian Mentzer. Givt: Generative infinite-vocabulary transformers. In *ECCV*, 2024.
- [97] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *NeurIPS*, 2017.
- [98] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. <https://github.com/huggingface/diffusers>, 2022.
- [99] Changan Wang, Morgane Riviere, Ann Lee, Anne Wu, Chaitanya Talnikar, Daniel Haziza, Mary Williamson, Juan Pino, and Emmanuel Dupoux. Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation. *arXiv preprint arXiv:2101.00390*, 2021.
- [100] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.
- [101] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *NeurIPS*, 2023.
- [102] Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. *arXiv preprint arXiv:2410.13848*, 2024.
- [103] Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. *arXiv preprint arXiv:2408.12528*, 2024.
- [104] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *IEEE ICASSP*, 2020.

- [105] Zijin Yang, Kai Zeng, Kejiang Chen, Han Fang, Weiming Zhang, and Nenghai Yu. Gaussian shading: Provable performance-lossless image watermarking for diffusion models. In *CVPR*, 2024.
- [106] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019.
- [107] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*, 2021.
- [108] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.
- [109] Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation. *arXiv preprint arXiv:2411.00776*, 2024.
- [110] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE TASLPRO*, 2022.
- [111] Jun Zhan, Junqi Dai, Jiasheng Ye, Yunhua Zhou, Dong Zhang, Zhigeng Liu, Xin Zhang, Ruibin Yuan, Ge Zhang, Linyang Li, et al. Anygpt: Unified multimodal llm with discrete sequence modeling. *arXiv preprint arXiv:2402.12226*, 2024.
- [112] Chaoning Zhang, Adil Karjauv, Philipp Benz, and In So Kweon. Towards robust deep hiding under non-differentiable distortions for practical blind watermarking. In *ACM MM*, 2021.
- [113] Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities. In *EMNLP*, 2023.
- [114] Qian Zhang, Tongda Xu, Yanghao Li, and Yan Wang. Evaluating strong idempotence of image codec. *arXiv*, 2023.
- [115] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.
- [116] Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. REMARK-LLM: A robust and efficient watermarking framework for generative large language models. In *USENIX Security Symposium*, 2024.
- [117] Ziqiang Zhang, Sanyuan Chen, Long Zhou, Yu Wu, Shuo Ren, Shujie Liu, Zhuoyuan Yao, Xun Gong, Lirong Dai, Jinyu Li, et al. Speechlm: Enhanced speech pre-training with unpaired textual data. *IEEE TASLPRO*, 2024.
- [118] Zhaoxi Zhang, Xiaomei Zhang, Yanjun Zhang, Leo Yu Zhang, Chao Chen, Shengshan Hu, Asif Gill, and Shirui Pan. Large language model watermark stealing with mixed integer programming. *arXiv*, 2024.
- [119] Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. *arXiv*, 2023.
- [120] Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arXiv:2408.11039*, 2024.
- [121] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *ECCV*, 2018.
- [122] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.
- [123] Lei Zhu, Fangyun Wei, Yanye Lu, and Dong Chen. Scaling the codebook size of VQ-GAN to 100,000 with a utilization rate of 99%. In *NeurIPS*, 2024.

Appendix

A Ethical Statement

A.1 Societal Impact

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

A.2 Environmental impact

The cost of experiments and model training is high, though order of magnitude less than training the generative models themselves. Finetuning the image tokenizer as done in the paper takes ≤ 32 GPU-hours. We also roughly estimate that the number of GPU-days used for running all our experiments is around 500, i.e., 12k GPU-hours. This amounts to total emissions in the order of 1 ton of CO₂eq. Estimations are conducted using the [Machine Learning Impact Calculator](#) presented by Lacoste et al. [54]. Namely, using a default grid, we compute $250\text{W} \times 12000\text{h} = 3000 \text{ kWh} \times 0.3 \text{ kg eq. CO}_2/\text{kWh} = 900 \text{ kg eq. CO}_2$. We do not consider memory storage, CPU-hours, production cost of GPUs/CPUs, etc.

B Technical Details of LLM Watermarking

We here more thoroughly introduce LLM watermarking, following the notation in [Section 2](#).

Generation. We consider an autoregressive model \mathcal{M} generating a sequence of tokens $s = (s_1, s_2, \dots, s_T)$, where each token s_t is sampled from a probability distribution conditioned on the previous tokens $p(s_t | s_{<t})$. In practice, the model outputs a vector of logits $\ell \in \mathbb{R}^{|V|}$, where V is the vocabulary (which we can assume in the most general case can contain text, audio or image tokens), which is transformed into a probability distribution $p = \text{softmax}(\ell/\tau)$, with τ being a temperature parameter.

The watermark scheme modifies the token selection process using a secret key ξ . A cryptographic hash function takes as input h previous tokens $(s_{t-h}, \dots, s_{t-1})$ (the context window) and the secret key ξ , producing a seed for a random number generator (RNG) that influences the choice of the next token s_t .

Two prominent LLM watermarking approaches are:

- The method of Kirchenbauer et al. [46] (KGW), which uses RNG to randomly partition the vocabulary V into a greenlist \mathcal{G}_t and a redlist \mathcal{R}_t , where \mathcal{G}_t contains a proportion γ of the vocabulary. The logit of each token in the greenlist is increased by $\delta > 0$, effectively boosting the probability of selecting tokens from the greenlist.
- The method of Aaronson and Kirchner [1], which uses a different approach based on the RNG to sample secret values for each token. Although we do not present it in this work for simplicity, it could have been adapted in the same way to watermark autoregressive models.

Detection. For the KGW method that we focus on, the *watermark detection* process analyzes a token sequence s and computes a score S based on the count of green tokens:

$$S = \sum_{t=h+1}^T \mathbb{1}(s_t \in \mathcal{G}_t), \tag{7}$$

where \mathcal{G}_t is the greenlist for position t , which depends on the h preceding tokens and the secret key ξ .

Statistical hypothesis testing. Detection uses a statistical hypothesis test distinguishing between \mathcal{H}_0 : “the sequence is not watermarked with secret key ξ ” and the alternative \mathcal{H}_1 : “the sequence was generated with a watermark with secret key ξ .” Previous approaches, such as those by Kirchenbauer et al. [46] and Aaronson and Kirchner [1], relied on a Z -test to compare the count of green tokens S to its expected value under the null hypothesis \mathcal{H}_0 . In this work we instead adopt an exact test [27], which is more accurate, especially for short sequences.

Under \mathcal{H}_0 , S follows a binomial distribution \mathcal{B} with parameters $(T - h)$ and γ , where γ is the expected proportion of green tokens, T is the total number of tokens, and h is the size of the watermark context window. The p-value determines the likelihood of observing a score as extreme as S under \mathcal{H}_0 , and is calculated as:

$$\text{p-value}(S, T, h, \gamma) = \text{Prob}(X \geq S \mid X \sim \mathcal{B}(T - h, \gamma)) = I_\gamma(S, T - h - S + 1), \quad (8)$$

where $I_x(a, b)$ is the regularized incomplete Beta function.

Sequences are flagged as watermarked if the p-value falls below the desired false positive rate.

Main parameters. The main parameters of the watermarking method are the context window size h , the watermark strength factor δ and the proportion of green tokens γ .

The context window size h determines how many previous tokens determine the greenlist. A smaller h increases robustness against text modifications but may bias generation as the same hash is used more frequently. It typically reduces security since recurring greenlists make the watermark easier to spoof [42, 118]. When $h = 0$, the RNG seed depends solely on the secret key ξ , creating fixed green/red lists for all tokens. For non-text tokens, we hypothesize that $h = 0$ maintains security since tokenizer access is restricted and image tokenizers have more degrees of freedom than text ones.

The watermark strength factor δ determines the amount by which the logits of green tokens are boosted. A higher δ increases the robustness of the watermark, but also increases the risk of generating low-quality text/images. It is tuned for every model and application.

The proportion of green tokens γ affects both detection sensitivity and generation quality. With low δ , a smaller γ reduces green token selection during generation, resulting in lower watermark power. With high δ , it restricts token choice and may lower output quality. During detection, lower γ values yield more significant p-values since green tokens are less likely to appear by chance [46]. At fixed watermark power, higher γ distributes the watermark evenly, while lower values concentrate it on fewer tokens. We set γ to 0.25 in our experiments, as it is a common choice in the literature [1, 46] and consistently yields good results in our experiments.

C More on Reverse Cycle-consistency

In this section, we elaborate on the case of text tokenizers not being perfectly reverse cycle-consistent (RCC), discuss audio tokenizers and our experiment measuring RCC in this setting, expanding on [Section 5](#), and discuss related topics studied in prior work.

C.1 RCC in Text Tokenizers

In BPE tokenizers, the vocabulary is initialized with all characters in the training set, and common character pairs are iteratively merged and added to the vocabulary until the predefined size is reached. Tokenization is performed greedily from left to right, by always selecting the longest possible token from V . Detokenization is simply performed by a lookup into V .

RCC can be violated. Text tokenizers are not immune to the RCC issue. For example, consider the following subset of the GPT-4O tokenizer: `{cons: 9673, istent: 20908, consistent: 173878}`. Due to the

Table 4 Token Match across different streams for 1000 sequences, where audios are subject to transformations before re-tokenization. Sequences of tokens are generated either as reconstructions of 10 second VoxPopuli audios with the MIMI tokenizer, or by the MOSHI model with audio prompts (described in Appendix E.3).

	MIMI tokenizer									MOSHI								
	1	2	3	4	5	6	7	8	Avg.	1	2	3	4	5	6	7	8	Avg.
Identity	0.56	0.31	0.21	0.22	0.18	0.20	0.18	0.16	0.25	0.60	0.55	0.38	0.28	0.26	0.26	0.24	0.25	0.35
<i>Transformations</i>																		
Lowpass 3 kHz	0.38	0.15	0.15	0.17	0.14	0.16	0.15	0.12	0.18	0.50	0.39	0.30	0.21	0.19	0.21	0.20	0.21	0.28
Noise 0.001	0.50	0.33	0.19	0.20	0.17	0.18	0.17	0.14	0.24	0.51	0.34	0.19	0.18	0.17	0.18	0.18	0.13	0.23
MP3 16 kbps	0.44	0.19	0.16	0.18	0.15	0.17	0.16	0.13	0.20	0.54	0.41	0.29	0.20	0.18	0.20	0.20	0.21	0.28
Encodec	0.24	0.13	0.10	0.10	0.07	0.07	0.05	0.04	0.10	0.28	0.38	0.24	0.16	0.14	0.15	0.11	0.15	0.20
Speed $\times 1.25$	0.02	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.04	0.11	0.08	0.05	0.05	0.06	0.04	0.07	0.06
Crop (90% kept)	0.03	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.08	0.15	0.11	0.07	0.07	0.08	0.05	0.09	0.09

greedy property of BPE tokenizers, $\mathcal{D}_{\text{text}}$ is guaranteed to always invert $\mathcal{T}_{\text{text}}$, e.g.,

$$\mathcal{D}_{\text{text}}(\mathcal{T}_{\text{text}}(\text{consistent})) = \mathcal{D}_{\text{text}}([173878]) = \text{consistent},$$

guaranteeing *forward cycle-consistency (FCC)*. In contrast, *reverse cycle-consistency (RCC)*, necessary for a strong watermark, may be violated, e.g.,

$$\mathcal{T}_{\text{text}}(\mathcal{D}_{\text{text}}([9673, 20908])) = \mathcal{T}_{\text{text}}(\text{consistent}) = [173878].$$

RCC approximately holds for text tokenizers in practice: while it is also a prerequisite for successful watermarking in text, no prior art has highlighted this as a hurdle. Some works have even shown that adversaries learning about the watermark can still be successful even if they use a different tokenizer [42], which is only possible if the tokens match across tokenizers.

Experiment. As discussed in the main text, we confirmed this experimentally. We used LLAMA3.1-8B-INSTRUCT to generate 1000 answers to prompts from the Open Assistant dataset [49]. We then compared the generated sequences of tokens with the re-tokenized sequences. Specifically, we took the token IDs from the model’s generation, detokenized them to text, then re-tokenized this text and computed the Levenshtein distance between the original and the new token sequence. Our results showed that the average token match is 99.5%, confirming that text tokenizers exhibit very high reverse cycle-consistency in practice.

C.2 RCC in Audio Tokenizers

A study of the RCC issue in MOSHI’s tokenizer is already given by the authors [24] (called *idempotence*). We however observe some differences in our study, as well as other key findings, such as the effect of augmentations, that we summarized in Section 5, and that we discuss in more detail in the following.

RVQ tokenizer. As a reminder, MOSHI’s tokenizer (MIMI) utilizes *residual vector quantization* (RVQ) [23, 52, 55, 110]. In RVQ, the quantization process happens iteratively, where each step quantizes the residual error from the previous quantization. Formally, for each step i , representing an audio frame of 1920 samples, the tokenization results in a sequence of tokens (s_i^1, \dots, s_i^K) corresponding to K different codebooks C^1, \dots, C^K (referred to as K streams). Each token represents approximately 80 ms of audio. The first token (or stream) is referred to as *semantic*, because there is a distillation loss during training with a non-causal model that encourages this first codebook to capture the most semantically relevant information. Défossez et al. [24] note that this semantic token exhibits higher cycle-consistency compared to later streams, which are assumed to progressively capture more fine-grained details, and to be less consistent.

Experimental setup. We measure Token Match (TM) for sequences either (a) generated as reconstructions of 10-seconds audios from VoxPopuli with the MIMI tokenizer, or (b) generated by the MOSHI model, as described in Appendix E.3. This corresponds to ≈ 125 time-steps for both cases, so $\approx 1,000$

audio tokens (counting all the streams). The audio is subjected to various transformations before re-tokenization, which include the three categories: valuemetric (lowpass filtering at 3kHz, addition of strong Gaussian noise at 0.01 amplitude), temporal-frequency (speed modification by 1.25x, cropping 90% of the original audio), and compression-based (MP3 compression at 16kbps, EnCodec compression).

Results. Table 4 presents the results that supplement the study by Défossez et al. [24]. Notably, their study only focuses on pre-existing audio sequences, while we also include generated sequences, which behave differently. For instance, the second stream sometimes shows higher consistency than other streams contradicting the expectation that only the first (semantic) stream could maintain high consistency. Different augmentations affect streams differently: e.g., lowpass has less impact on streams 2 and 3 compared to strong noise addition, while EnCodec strongly decreases TM of the first stream. Importantly, temporal-frequency augmentations (speed, cropping) reduce TM less dramatically for MOSHI (0.04-0.15) than would be expected given our image watermarking results where such transformations typically yield near-zero consistency. This multi-stream aspect presents challenges and opportunities for watermarking: while complicating RCC analysis, it enables potential development of more sophisticated techniques leveraging complementary properties across streams.

C.3 Related Concepts

There are several concepts related to RCC that were studied in prior work.

Codec idempotence. In the context of codecs a relevant property is *codec idempotence* [44, 73, 114]. Directly applying the mathematical definition of idempotence to our case, we let $f(\cdot)$ denote the encoder-decoder pair (e.g., f compresses an image to a JPEG file and then decompresses it back to pixels), and say that a codec is idempotent if it satisfies:

$$f(f(x)) = f(x). \tag{9}$$

This property is naturally of interest to codecs. While we assume that JPEG compressing an image is lossy (i.e., we do not expect $f(x) = x$), we want our codec to not further degrade image quality on successive applications, which can commonly occur in practice (i.e., $f(f(x)) = f(x)$). This is in stark contrast with the notion of RCC relevant to our work. In our case, x are the tokens, while $f(\cdot)$ is the detokenization followed by the tokenization. The first application of $f(\cdot)$ is crucial for us: as explained above, we require $f(x) \approx x$ as otherwise the watermark that was present in the tokens of x is lost. $f(f(x))$, i.e., re-tokenizing the image several times successively, on the other hand is not of particular interest in this case, thus idempotence is not an important concern.

Consistency of tokenizers. Another related concern is the *consistency of tokenizers* [61, 87]. Intuitively a tokenizer is *consistent* if the tokenization of a particular string (assuming the text domain) does not change depending on the surrounding context. While the cited works show that this is a desirable property, it is not as relevant to our motivation of preserving the watermark as RCC. In particular, for generated token sequence x , if RCC is satisfied the watermark will be entirely preserved, even if the tokenization was context-dependent. This may be a concern in the context of various attacks: for example, infilling a part of the image before re-tokenization should ideally change only the tokens corresponding to the infilled part, and not the entire image, which may happen depending on the setup of the convolutions in the tokenizer. We do not explore this angle as part of this work.

Cycle-consistency in other contexts. Finally, a line of works studies cycle-consistency in various generative models [39, 122], most commonly in the context of style transfer: a single *cycle* is the translation from a style A to a style B and back to A , and cycle-consistency can in this case be beneficial as a constraint for the model. Finally, Teng and Choromanska [92] explicitly parametrize encoder-decoder pairs to be inverses of each other and Huang et al. [40] study cycle-consistency in the context of disentangled representations.

D More on Watermark Synchronization

In [Appendix D.1](#) we provide a more detailed description of our watermark synchronization layer ([Section 3.2](#)), and show additional examples. In [Appendix D.2](#) we describe our attempt to use AudioSeal [\[83\]](#) for synchronization in audio.

D.1 Image Synchronization Details

We remark that the problem of watermark synchronization was studied in the past, before the advent of generative models, in the context of digital watermarking. These works suggest approaches such as multiple testing with a carefully controlled number of tests to avoid the false positive rate increase we mentioned in [Section 3.2](#) [\[34\]](#), or similarly to us, embedding a synchronization pattern in addition to the original watermarking pattern to revert the transformation [\[19, 74, 95\]](#). As noted above, we are aware of two works that study this in the context of post-hoc generative model watermarks [\[33, 64\]](#), however their code is not publicly available.

Full algorithm description. Our algorithm consists of two main procedures: embedding a synchronization pattern into the generated and decoded image, and estimating the transformation from an incoming image where previously the watermark and the synchronization pattern were embedded. We assume access to a localized watermark module L that can embed a different message in every pixel of an image, and recover the probability that each pixel has the watermark along with the most probable message in it. As noted above, we instantiate this using Sander et al. [\[85\]](#).

To embed the synchronization pattern, we use four 32-bit messages $\{m_1 = 0^{32}, m_2 = 0^{16}1^{16}, m_3 = 1^{16}0^{16}, m_4 = 1^{32}\}$. For each message, the corresponding mask is one of the quadrants (as in [Figure 4](#)), where given parameter μ , we keep a horizontal and a vertical strip of width μ pixels in the middle of the image free of messages (we use $\mu = 18$ for TAMING and RAR-XL, and $\mu = 36$ for CHAMELEON as we work with images of twice the resolution). Using this mask, we embed the pattern using L .

Given an incoming image, we first obtain and postprocess the predictions of L . Namely, for each pixel, we take the closest message in Hamming distance from the four fixed messages above, as long as the Hamming distance is below 6 bits and the probability of the pixel being watermarked as predicted by L is above 0.5. Then, as a heuristic, we proceed only if we found at least one pixel for each of the 4 messages, and if the total area of the pixels with messages is at least 70% of the image—attempting to estimate the transformation otherwise proved too unreliable.

To estimate the transformation, we sweep over rotations in $[-20, 20]$ degrees, for each rotation *rotate the grid of extracted messages back* by the inverse rotation, and attempt to find the best-fitting pair (i, j) such that the row i of the rotated message grid best separates pixels with messages (m_1, m_3) as well as (m_2, m_4) , and the column j of the rotated message grid best separates pixels with messages (m_1, m_2) as well as (m_3, m_4) . For example, to find j that best separates (m_1, m_2) , we compute the *cost* of each candidate j' as the number of *wrongly positioned* pixels, i.e., pixels with message m_1 that are to the right of j' and pixels with message m_2 that are to the left of j' . We repeat the same cost computation for the horizontally flipped message grid: if we find that this leads to a lower cost, we estimate that the image was flipped. The lowest-cost estimate for j and the estimate if the image was flipped or not are then aggregated over (m_1, m_2) and (m_3, m_4) proportionally to the number of pixels with these messages in the image. The process for i is analogous, with the final result for the fixed suspect rotation being the tuple $(i, j, isFlipped, cost)$.

We finally take such tuple with the minimal *cost*, returning the corresponding rotation and $(i, j, isFlipped)$ as our final estimate that we later revert.

Examples. Complementing [Figure 4](#), in [Figure 8](#) we show real examples of recovered synchronization patterns and estimated transformations for horizontal flip, positive and negative rotation, crop, as well as Gaussian noise, that we found to be the most challenging valuometric transformation in terms of disturbing the synchronization signal.

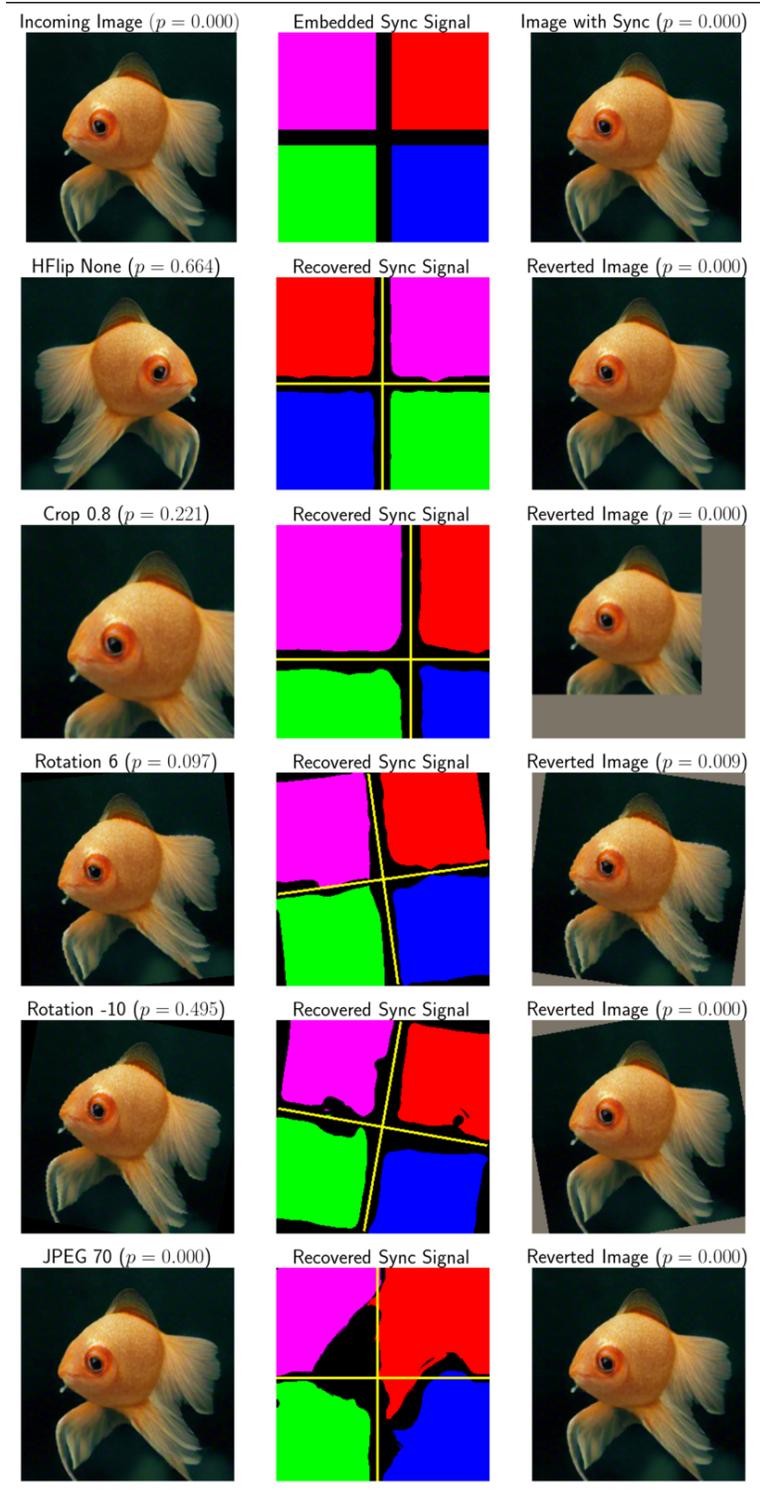


Figure 8 Visualization of our synchronization layer (Section 3.2) on a real example from our experiments. In the four middle rows we see that the watermark detection would have failed on original geometrically transformed images, but has eventually succeeded after the synchronization signal was detected and reverted. In the bottom row we see that a valuemetric transformation can disrupt the signal—in this case a JPEG compression. While this did not hamper detection in this example, it can be problematic in practice as evidenced by the drop in valuemetric robustness with synchronization shown in Table 2.

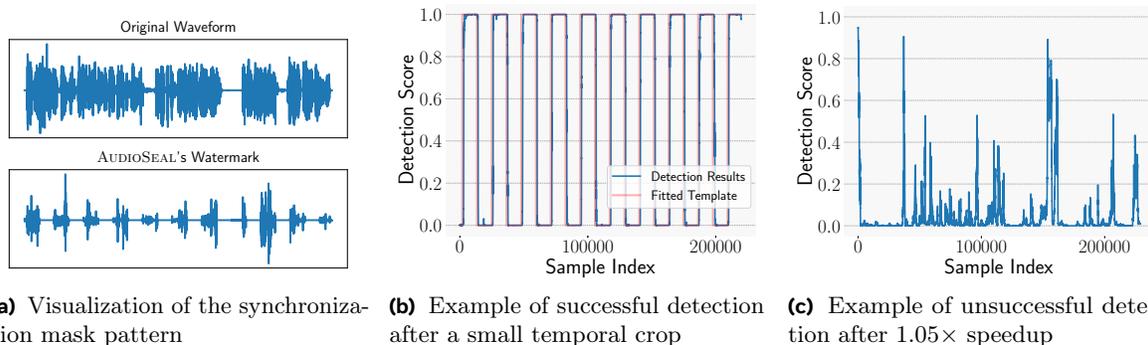


Figure 9 The audio watermark synchronization method we attempted to incorporate. (a) how we embed the watermark periodically in the audio; (b) successful case with a clear periodic pattern detected when the audio is cropped for the first 0.84 seconds; (c) a $1.05\times$ speedup creates a detection signal too noisy to reliably extract the synchronization pattern.

D.2 Audio Synchronization with Localized Audio Watermarking

As explained in [Section 5](#), we attempted to use the localization property of AudioSeal [83] for synchronization similar to [Section 3.2](#), but this approach proved less successful than with images. We describe below the method and experiments supporting this claim and summarize the results in [Figure 9](#).

Synchronization approach and transformation detection. We use AudioSeal’s watermark embedder to embed watermarks with a periodic mask pattern across the audio signal. This enables detection of transformations like time-stretching and phase shifts through cross-correlation analysis. We applied a square wave template with periods of 6 frames, each frame being 1920 samples at 24kHz, as in MIMI.

When audio undergoes transformations, the periodic pattern distorts predictably. Through cross-correlation between the detection signal and template patterns, we can estimate the speedup factor (identified by the period maximizing cross-correlation) and phase shift (located by finding optimal alignment). Once estimated, we can invert the transformation by resampling to original speed and applying phase correction. More specifically, in our implementation, the detection results are first downsampled by a factor of 8 and we sweep the template period from $0.5\times$ to $1.5\times$ the nominal half-period (6 frames = 0.48s) in coarse steps of 10 samples, then refine over ± 10 samples around the best match; this two-stage cross-correlation yields precise estimates of speedup (from the best period) and phase shift, which are used to resample back to 24kHz and correct the alignment.

Challenges and limitations. As shown in [Figure 9\(c\)](#), the detection signal extracted from AudioSeal is not yet robust enough for general use, as it fails to detect the watermark reliably after a speedup of $1.05\times$. Future work could explore more sophisticated synchronization methods.

E Experimental Details

E.1 Omitted Details of Image RCC Evaluation and Finetuning

Here we provide more details on our RCC evaluation experiments shown above in [Table 1](#) and details related to RCC finetuning introduced in [Section 3.1](#).

RCC evaluation. For [Table 1](#), for simplicity, we re-use the watermarked BASE model of TAMING from our main experiments. We confirmed that running the non-watermarked version results in very similar values. We use the full set of valuemetric and geometric transformations as in the main experiments, and for each transformation use the same parameter that was chosen for summarized scores in [Table 2](#), as detailed below in [Appendix E.2](#).

RCC finetuning. To complete our RCC finetuning description from Section 3.1 we provide the omitted details. We note that the exact same hyperparameters were used for all models which demonstrates the transferability of our finetuning procedure. The finetuning is done for 10 epochs with distributed data parallel training on 16 V100 GPUs (TAMING, training takes 2h) and 8 H200 GPUs (CHAMELEON and RAR-XL, training takes 2.5h and 0.5h respectively). We use the Adam optimizer [45] with a learning rate of 10^{-4} , multiplied by a factor of 0.9 each epoch (StepLR). We use a total batch size across all GPUs of 64 (4 per gpu for TAMING and 8 per gpu for CHAMELEON and RAR-XL), and always set $\lambda = 1$. As noted above, we use a set of augmentations \mathcal{A} to improve robustness of our watermark to transformations and attacks. JPEG is not differentiable, therefore we backpropagate only through the difference between the uncompressed and compressed images (straight-through estimator): $x' = x_{\text{aug}} + \text{nograd}(x_{\text{aug, JPEG}} - x_{\text{aug}})$ [106, 112]. We define three progressively harder sets: \mathcal{A}_1 , \mathcal{A}_2 , \mathcal{A}_3 , and use no augmentations for 1 epoch, then \mathcal{A}_1 for 1 epochs, then \mathcal{A}_2 for 4 epochs, and finally \mathcal{A}_3 for the last 4 epochs.

\mathcal{A}_1 uses JPEG compression with qualities {90, 80, 70}, Gaussian blur with kernel sizes {1, 3}, Gaussian noise with standard deviations {0.005, 0.01, 0.015, 0.02}, Brightening with factors {1.0, 1.1, 1.2}, Rotation with angles $\{-1, 1\}$ degrees, and Cropping with % kept from {80, 90}. \mathcal{A}_2 uses JPEG with qualities {80, 60, 40}, Gaussian blur with kernel sizes {3, 5}, Gaussian noise with standard deviations {0.02, 0.04, 0.06}, Brightening with factors {1.2, 1.3, 1.4}, Rotation with angles $\{-3, -2, -1, 1, 2, 3\}$ degrees, and Cropping with % kept from {50, 60, 70, 80, 90}. Finally, \mathcal{A}_3 uses JPEG with qualities {40, 30, 20}, Gaussian blur with kernel sizes {5, 7, 9}, Gaussian noise with standard deviations {0.06, 0.08, 0.1}, Brighten with factors {1.4, 1.7, 2.0}, and the same geometric augmentations as in \mathcal{A}_2 .

E.2 Details of Main Experiments

We provide full details of our main experiments (Section 4.1 and Section 4.2), expanding on the information provided in the main paper.

Models. For TAMING, we use the VQGAN IMAGENET (F=16), 16384 version available in the authors’ repository. For CHAMELEON, we use the 7B model. Since the open-weight version does not include image generation capabilities (as noted in the original paper), we obtained the necessary weights directly from the authors. Alternatively, image generation with CHAMELEON can be approximated using the Anole model [15] and its associated repository: <https://github.com/GAIR-NLP/anole>, though we note that its output quality is somewhat lower. For RAR-XL we use the official model and image tokenizer from the authors’ repository.

Parameters. The results in Figure 5, Figure 6 and Table 2 are obtained from the same experiment, repeated on TAMING, CHAMELEON, and RAR-XL. For TAMING and RAR-XL we set $\delta = 2, \gamma = 0.25, h = 1$ and evaluate (for each transformation/attack) on 1000 generations, 100 per each of the following ImageNet class indices: [1, 9, 232, 340, 568, 656, 703, 814, 937, 975]. For CHAMELEON we set $\delta = 2, \gamma = 0.25, h = 0$. We again use 1000 generations, conditioning the model on a text prompt each time. Following the standard protocol in the literature [76, 77, 80, 82] we use the prompts from the validation set of MS-COCO [58]. To do so, we first retrieve all the captions from the validation set, keep only the first one for each image, and select the first 1000 (or 5000 when computing FID for CHAMELEON). While we did not benchmark this in detail, the computational overhead of our watermark matches that of the LLM watermarking scheme we inherit from *and* the localized watermark we use as the synchronization signal. As discussed in Section 6 more thoroughly integrating these two components could also make our watermark more efficient. A single run (e.g., BASE with all (augmentation, parameter) pairs detailed below on 1000 generations) with TAMING was executed on 25 V100 GPUs, lasting ≈ 30 minutes for BASE, FT, FT+AUGS and ≈ 1.5 h for FT+AUGS+SYNC. For CHAMELEON and RAR-XL, we use 10 H200 GPUs (50 for FT+AUGS+SYNC), taking comparable time as for TAMING.

Split stratification. As noted in the literature [25, 107], trained VQGANs often suffer from *low codebook utilization*, meaning that a certain percentage of the codebook is effectively not used and those codes (*dead codes*) are in practice never emitted by the transformer nor used when tokenizing images. While

later work addresses this issue [123], the VQGAN used in TAMING and in our experiments suffers from this issue and has only 971 *alive codes*, despite the codebook size of 16384.

This can affect the soundness of the watermark. Namely, the null hypothesis assumes that the ratio of green tokens in content produced without the use of the model \mathcal{M} is γ . However, if the number of alive codes n_{alive} is much smaller than the codebook size $|V|$, there is a non-negligible chance that choosing the set of green tokens as a uniformly random subset of $|V|$ of size $\gamma|V|$ results in significantly more or less than γn_{alive} *alive* green tokens. As these are the only tokens emitted by the tokenizer in practice, the green ratio under the null hypothesis can thus be significantly different from γ , making Equation 2 inaccurate. While for $h > 0$ we can hope that this effect averages out across different contexts (as the expected alive green ratio is still γ), for $h = 0$ (fixed red/green split) this can introduce a constant bias.

In particular, assume for simplicity that $\gamma|V|$ and γn_{alive} are both integers. The probability P_g that a uniformly random split of $|V|$ into $\gamma|V|$ green and $|V| - \gamma|V|$ red tokens results in *exactly* g green tokens among alive ones is not given by a binomial distribution but by a [hypergeometric distribution](#):

$$P_g(|V|, n_{\text{alive}}, \gamma) = \frac{\binom{\gamma|V|}{g} \cdot \binom{|V| - \gamma|V|}{n_{\text{alive}} - g}}{\binom{|V|}{n_{\text{alive}}}}. \quad (10)$$

In Figure 10 we plot the distribution of *actual green ratios*, i.e., compute $P_g(|V|, n_{\text{alive}}, \gamma)/n_{\text{alive}}$ for different values of g , for $|V| = 192, n_{\text{alive}} = 12, \gamma = 0.25$ (corresponding roughly to the ratio $n_{\text{alive}}/|V|$ of the VQGAN we use in our experiments). We see (*in red*) that there is in fact only $\approx 25\%$ chance that the green ratio among alive tokens is correctly set to γ . To resolve this, for TAMING we use a *stratified* split, i.e., we separately sample a red/green split on alive and dead codes, ensuring that the green ratio of alive tokens is exactly γ (*green in Figure 10*).

An interesting question, to the best of our knowledge not explored before, is if similar effects can be observed in LLM watermarking. For example, a uniformly random split of a large multilingual vocabulary may introduce a particularly biased split on e.g., Cyrillic tokens, which are effectively the majority of the alive ones when the LLM is prompted to write in a language that uses the Cyrillic script. Especially for $h = 0$, this may point at unfairness towards certain subdomains, where for a particular subdomain the watermark is overly conservative or more importantly has a much higher FPR than stated theoretically.

Image transformations. We next list all image transformations and their parameters used in our main experiments. We evaluate 90 variants (the original image and 89 transformations described below) for each image, i.e., 90,000 images in total per evaluation. For valuemetric transformations we use:

- Gaussian Blur: kernel sizes [0, 1, 3, 5, 7, **9**, 11, 13, 15, 17, 19].
- Gaussian Noise: standard deviations [0, 0.025, 0.05, 0.075, **0.1**, 0.125, 0.15, 0.175, 0.2].
- JPEG Compression: quality factors [100, 95, 85, 75, 65, 55, 45, 35, **25**, 15, 5].
- Brighten: factors [**1**, 1.25, 1.5, 1.75, **2**, 2.25, 2.5, 2.75, 3].

For geometric transformations we use:

- Rotation: angles [-20, -15, -10, -5, 0, 5, **10**, 15, 20].

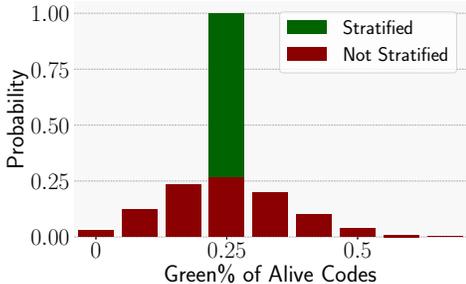


Figure 10 When the number of alive codes n_{alive} is much smaller than the codebook size $|V|$, green/red splits may not correspond to the expected ratio of γ . In this figure, $n_{\text{alive}} = 12, |V| = 192, \gamma = 0.25$. Stratification, i.e., separate splitting of red and green tokens, resolves this issue.

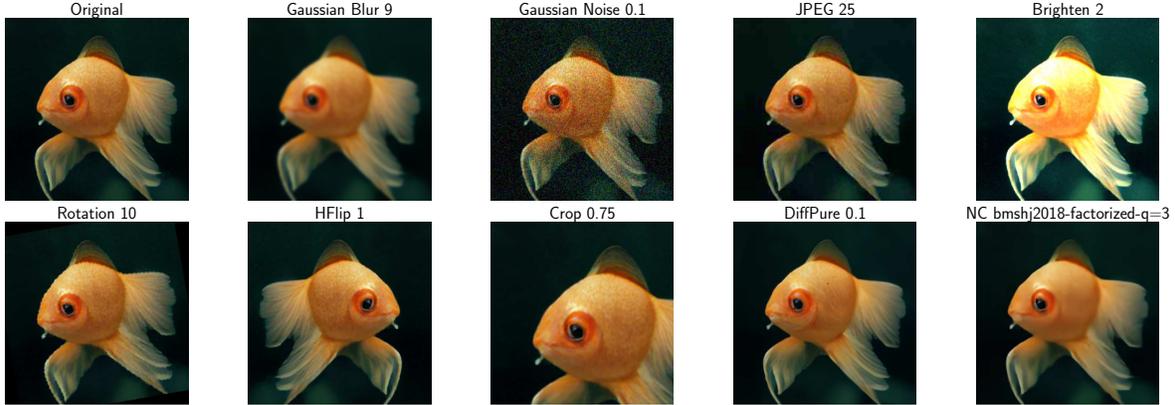


Figure 11 Examples of transformations with parameters used to compute the scores in [Table 2](#).

- Horizontal Flip: parameters $[0, \mathbf{1}]$, where 1 indicates that a flip was performed.
- Crop: percent of the image kept $[\mathbf{1.0}, 0.95, 0.9, 0.85, 0.8, \mathbf{0.75}, 0.7, 0.65, 0.6, 0.55, 0.5]$, where we crop from the top-left corner of the image and then resize it back to the original size.

Finally, we use the following attacks:

- DiffPure: timesteps $[0.01, 0.05, \mathbf{0.1}, 0.2, 0.3]$ with the 256×256 ImageNet diffusion model used in the original attack of Nie et al. [\[71\]](#).
- Neural Compression: a range of 22 models with different quality factors; see details below.

The underlined values above correspond to transformations that do not change the image (showing the maximum of robustness in each subplot of [Figure 6](#)). The **bold** values are used to, following prior work [\[101\]](#), summarize the results to a single score per transformation/attack type in [Table 2](#), where we average the 4 valuemetric scores and 3 geometric scores independently. For neural compression we describe how we compute the score below. Visual examples of each bold transformation/attack are shown on a real TAMING generation in [Figure 11](#).

Neural compression. For neural compression we use the following models from the [CompressAI](#) [\[7\]](#) library:

- BMSHJ18 (FACTORIZED) [\[5\]](#) with quality factors $q \in \{1, 3, 6\}$.
- BMSHJ18 (HYPERPRIOR) [\[5\]](#) with quality factors $q \in \{1, 3, 6\}$.
- CSTK20 (ANCHOR) [\[14\]](#) with quality factors $q \in \{1, 3, 6\}$.
- CSTK20 (ATTENTION) [\[14\]](#) with quality factors $q \in \{1, 3, 6\}$.
- MBT18 [\[69\]](#) with quality factors $q \in \{1, 3, 6\}$.
- MBT18 (SCALE) [\[69\]](#) with quality factors $q \in \{1, 3, 6\}$.

To sort these by compression strength we compute *bpp* (bits per pixel) as done in the library:

$$\frac{\sum_i \log L_i}{-\ln 2 \cdot n_{\text{pix}}}, \quad (11)$$

where L is the likelihood vector and n_{pix} is the number of pixels in the image. Empirically we observe *bpp* of around 0.1 ($q = 1$), 0.3 ($q = 3$), and 1.0 ($q = 6$). When reporting a single score for neural compression we average the six scores with $q = 3$. Additionally, we evaluate the following four autoencoders from the [diffusers](#) [\[98\]](#) library for which we compute *bpp* manually by considering the

downscaling factor in the latent space, the latents size, and 16-bit/32-bit floating precision. We see that our calculations are consistent with the results of neural compressors from **CompressAI**:

- The Stable Diffusion VAE [80] (`stabilityai/sd-vae-ft-ema`; SD VAE (FT-EMA)), with bpp 2.
- The Stable Diffusion XL VAE in half precision [75] (`madebyollin/sdx1-vae-fp16-fix`; SDXL VAE (FP16)), with bpp 1.
- The Deep Compression AE [13] (`mit-han-lab/dc-ae-f64c128-in-1.0-diffusers`; DC-AE), with bpp 1.
- The VAE of Flux [53] (from the `black-forest-labs/FLUX.1-schnell` pipeline; FLUX VAE), with bpp 1.

E.3 Details of Audio Experiments

Audio prompt generation. We observed that MOSHI frequently generates brief responses and typically expects human interaction to continue the conversation. When using conventional text prompts such as those from Alpaca [91] or Open Assistant Conversations [49] datasets, the model rarely produced audio outputs of sufficient length (e.g., 10 seconds). We therefore synthesized specialized prompts designed for this particular use case. These prompts are used when prompting the MOSHI model to generate (possibly watermarked) audio, such as in the experiments described in Section 5, Appendix C.2, and Appendix F.7.

To create a diverse collection of audio monologue topics, we leveraged LLAMA 3.1-8B-INSTRUCT to generate 1000 unique text prompts. We guided the model using a system+user template to produce concise single-sentence requests (each beginning with action verbs like “Describe”, “Talk about”, etc.) covering distinct subjects. We then filtered out near-duplicates by calculating pairwise Rouge-L scores [57] (using a threshold of 0.7) and eliminated texts that fell outside our desired length parameters. Representative examples include: “Describe the life cycle of a butterfly and the symbolic meanings associated with it.”, “Explain the process of photosynthesis in plants and its importance to ecosystems.”, or “Discuss the cultural significance of traditional Japanese tea ceremonies.”. Finally, we converted these text prompts into audio using the SEAMLESSV2 [6] (large) model, saving each resulting waveform alongside its corresponding source prompt. The resulting audio prompts average approximately 4 seconds in length.

Audio transformations. We evaluate robustness to a set of audio edits grouped into valuemetric, time-frequency, and neural compression transformations. When evaluating (e.g., in Table 3), each is applied with the following fixed strengths:

- Valuemetric:
 - Bandpass Filter: (300,3000), (500,5000), (1000,8000) Hz.
 - Highpass Filter: 100, 500, 1000 Hz.
 - Lowpass Filter: 1000, 3000, 8000 Hz.
 - Noise Injection (white): std = 0.001, 0.01, 0.05.
 - Pink Noise: std = 0.01, 0.05, 0.1.
 - Echo: (delay = 0.1 s, vol = 0.2), (0.3 s, 0.5), (0.5 s, 0.7).
 - Smooth: window fraction = 0.001, 0.005, 0.01.
 - Boost Audio: +50 %, +90 %.
 - Duck Audio: −50 %, −90 %.
 - MP3 Compression: bitrate = 16, 64, 128 kbps.

- Time-frequency:
 - Speed: factor = 0.75, 0.9, 1.0, 1.1, 1.25.
 - Temporal Crop: keep 50 %, 70 %, 90 % of duration.
 - Time Shift: shift = 10 ms, 20 ms, 40 ms.
 - Up/Down Resample: intermediate = 24 kHz, 36 kHz, 48 kHz.
- Neural Compression:
 - DAC Compression (24 kHz): full model pass.
 - EnCodec Compression (24 kHz): full model pass.

We use the same implementation as in AudioSeal [83] when the augmentations are available. For DAC [52] and EnCodec [23] we use the official models at 24 kHz.

RCC finetuning. We perform finetuning for 200 epochs with 1000 steps per epoch on batches of 64 audio clips of 10-seconds from VoxPopuli [99], using 2 H200 GPUs for 1 day. We use the AdamW optimizer [45] with a base learning rate of 2×10^{-5} , linear warmup over 5 epochs, and cosine annealing down to 2×10^{-7} . We set λ to 0.01 for the regularization loss in the FT+AUGS model (with transformations), while using 0.001 in the FT model (without transformations). For the regularization loss we use the Multi-Resolution STFT between the audios reconstructed either with the original decoder D_0 or the finetuned decoder D . Following notations from Section 3.1, for the RCC loss we use the MSE between z , the soft latents before the projection and quantization step, and z' , the soft latents generated by the encoder replica E' . To improve robustness for the FT+AUGS model, we apply augmentations \mathcal{A} from the start, sampling one augmentation per batch. The augmentations are chosen randomly at each step, and the parameters are sampled uniformly from the ranges below:

- Lowpass filter: cutoff 2000-6000 Hz
- Highpass filter: cutoff 200-600 Hz
- White noise injection: std 0.001-0.01
- Pink noise: std 0.001-0.01
- Smooth: window fraction 0.001-0.005
- Time shift: 0.3-10 ms

F Additional Experimental Results

In this section, we present additional results: ablations of RCC finetuning (Appendix F.1), investigations of different watermarking parameters (Appendix F.2), additional results for CHAMELEON and RAR-XL on token match, watermark power, quality, and robustness (Appendix F.3), comparison of the original and final decoders via PSNR (Appendix F.4), validation of statistical test correctness (Appendix F.5), comparison to generation-time watermarks (Appendix F.6), and omitted audio results (Appendix F.7).

F.1 Finetuning Ablations

We train five more finetunes of TAMING to test the influence of different parameters:

- $\lambda = 10$ uses a higher regularization weight, i.e., puts less weight on the RCC loss.
- $\lambda = 0.1$ uses a lower regularization weight, i.e., puts more weight on the RCC loss.
- $lr = 10^{-5}$ uses a lower learning rate.
- $lr = 10^{-3}$ uses a higher learning rate.

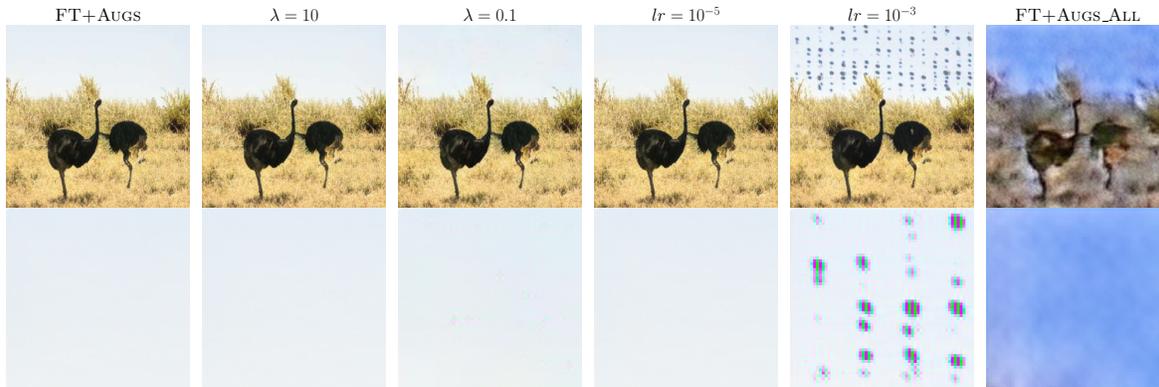


Figure 12 Visual examples of images produced by finetuning ablations introduced in [Appendix F.1](#). The first row shows a detokenized image (based on the same token sequence as we fix the seed) and the second row zooms in on the top-left 64×64 pixel region to more clearly show artifacts. We see that the $\lambda = 0.1$, $\alpha = 10^{-3}$, and FT+AUGS-ALL variants lead to significant degradation in image quality.

- FT+AUGS-ALL finetunes all components of the VQGAN, including the codebook.

The results are presented in [Table 5](#) and visual examples in [Figure 12](#) where the first row shows a detokenized output and the second row zooms in on the top-left 64×64 pixel region. Our baseline here is our FT+AUGS variant (we do not consider synchronization in this experiment). We see that increasing λ to 10 slightly degrades the results (likely within the level of experimental noise) and reducing the learning rate reduces robustness. We also experimented with training for more epochs with a lower learning rate, but this did not lead to better results than our best variant.

We also see that $\lambda = 0.1$, $\alpha = 10^{-3}$, and FT+AUGS-ALL lead to much better results. However, the visual results in [Figure 12](#) show that all three of these variants lead to degradation in image quality. For FT+AUGS-ALL this is the most evident, which motivates our discussion above regarding the importance of carefully choosing which modules to finetune. The artifacts in $\alpha = 10^{-3}$ clearly show that the learning rate is too high for stable training. For $\lambda = 0.1$ the artifacts are clearly visible in the second row of the figure, showing that this setting puts too much weight on the RCC loss.

Training on transformer-generated tokens. Finally, we hypothesized that using token sequences generated by the transformer as the training set instead of tokenizations of ImageNet images would improve finetuning, as the former more closely matches the distribution of inputs that the detokenizer sees at evaluation time. Another experiment we tried was using a mixture of these two token sequences. We did not observe any benefits of this approach.

Table 5 TPR at 1% FPR of finetuning ablations on TAMING in the setting of [Table 2](#). The ablations are described in [Appendix F.1](#). Three of the variants improve the results; however, [Figure 12](#) shows that they also lead to significant degradation in image quality.

	None	Valuematic	Geometric	Adversarial Purification	Neural Compression
BASE	0.99	0.26	0.01	0.43	0.48
FT+AUGS	1.00	0.92	0.01	0.70	0.79
$\lambda = 10$	1.00	0.91	0.01	0.68	0.77
$\lambda = 0.1$	1.00	0.98	0.01	0.85	0.96
$lr = 10^{-5}$	0.99	0.75	0.01	0.63	0.76
$lr = 10^{-3}$	0.98	0.98	0.01	0.81	0.78
FT+AUGS-ALL	1.00	1.00	0.02	0.85	0.98

Table 6 TPR at 1% FPR of watermark parameter ablations (δ, h, γ , context choice) on TAMING in the setting of Table 2 using FT+AUGS. The ablations and their results are discussed in Appendix F.2.

	None	Valuometric	Geometric	Adversarial Purification	Neural Compression	FID
FT+AUGS	1.00	0.92	0.01	0.70	0.79	16.33
$h = 2$	1.00	0.82	0.01	0.41	0.57	15.70
$h = 3$	1.00	0.73	0.01	0.24	0.39	15.87
$h = 3$ (SPATIAL)	1.00	0.69	0.01	0.22	0.36	16.12
$\delta = 1$	1.00	0.69	0.01	0.29	0.40	16.22
$\delta = 4$	1.00	0.97	0.01	0.86	0.93	18.76
$\gamma = 0.5$	1.00	0.86	0.01	0.51	0.63	15.71

F.2 Watermark Parameters

In the following experiments, starting from FT+AUGS on TAMING, we vary the main watermarking parameters: context size h , strength δ , green ratio γ , the choice of watermark context, and the partitioning strategy, presenting the key results in Table 6.

For the context size h , we observed that $h > 1$ generally led to non-robust watermarks. We explore both $h = 2$ and $h = 3$ and both the standard choice of watermark context (preceding h tokens) and the image-specific one (the h tokens spatially close, e.g., for $h = 3$ the tokens above the current token, to the left of it, and top-left of it in the image; denoted SPATIAL). As Table 6 shows, all of these variants significantly degrade robustness compared to the baseline (FT+AUGS). For adversarial purification and neural compression, the results are comparable to the non-ablated results *without finetuning*.

This is in line with the intuitive understanding of h from prior work [46, 119]: large h makes watermark removal easier, as changing any of the preceding h tokens changes the red/green split at the following token. On the other hand, low h makes the watermark less secure, i.e., easier to forge. As noted above, we hypothesize that $h = 0$ is a more viable choice for images (and thus use it for CHAMELEON), as reverse-engineering of the watermark rules (as successfully done for $h = 0$ in text [42, 118]) is likely much more difficult due to the complex image tokenizer being hidden.

Regarding strength, we found that $\delta = 2$ is for both our models the strongest watermark that does not degrade quality. As seen in Table 6, $\delta = 1$ degrades robustness, while $\delta = 4$ greatly boosts it at the cost of generation quality: for all rows in Table 6, FID is lower or comparable to the baseline value, while $\delta = 4$ increases it to 18.7. Across our experiments we found that $\gamma = 0.25$ leads to slightly better tradeoffs overall than $\gamma = 0.5$, as illustrated in the last row of Table 6.

Finally, we briefly experimented with a semantic partitioning strategy, related to similar attempts for LLMs [37]. In particular, instead of choosing green tokens \mathcal{G} uniformly at random from the vocabulary at each partitioning step, we use k-means clustering to partition the hard embeddings of each token in the vocabulary into $k = 100$ clusters and assign colors such that all tokens in the same cluster have the same color, while keeping the overall green ratio at γ . In theory, this should make the watermark more robust to modifications that do not change the semantics, as the resulting change in the token would hopefully not leave the k-means cluster and thus remain green. On the other hand, it might make it harder for the decoding-time watermark to replace a green token with a red one, as all tokens from the same k-means cluster, which may be good alternatives, are also red. In practice, we consistently observed higher robustness but at the cost of significant quality degradation. More work is needed to understand how to find a good tradeoff with this approach.

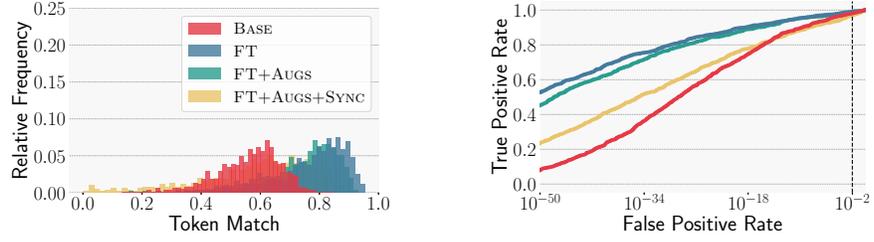


Figure 13 Token match and watermark power results for CHAMELEON, analogous to Figure 5.

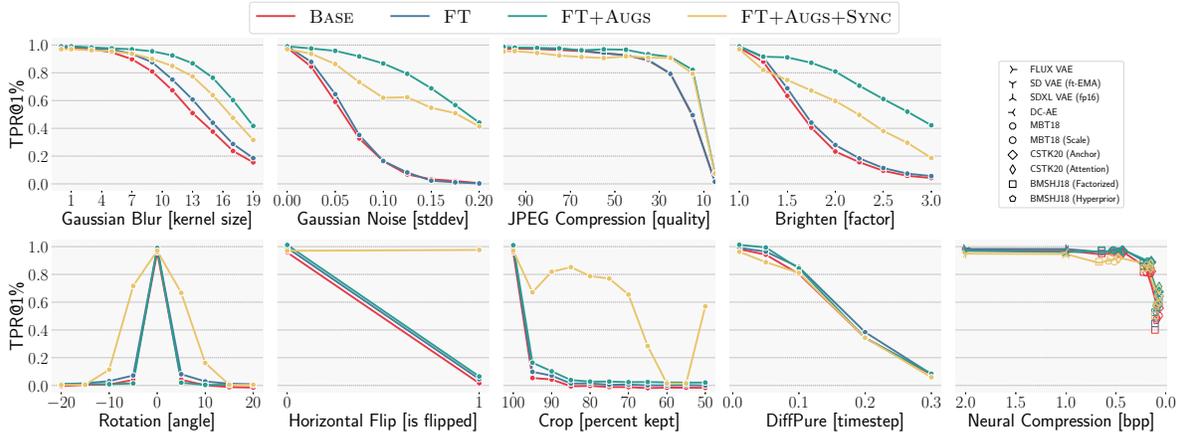


Figure 14 Robustness results for CHAMELEON, analogous to Figure 6.

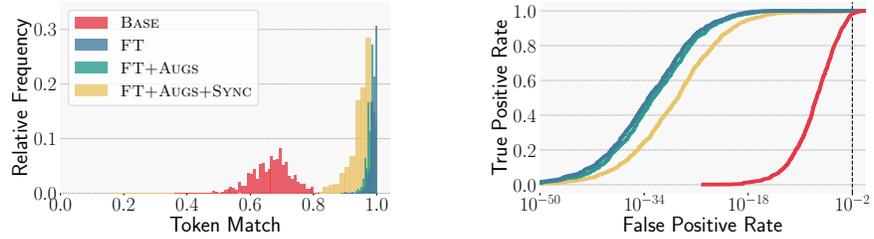


Figure 15 Token match and watermark power results for RAR-XL, analogous to Figure 5.

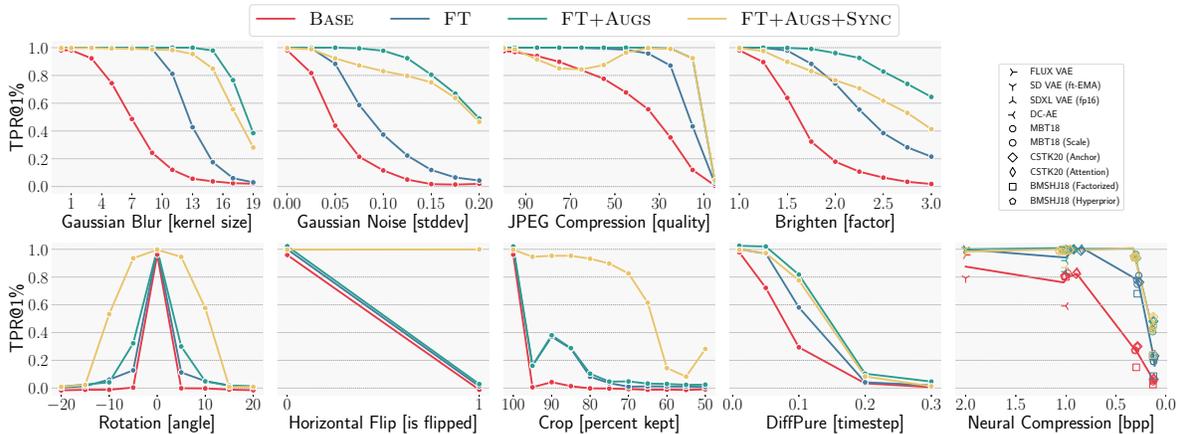


Figure 16 Robustness results for RAR-XL, analogous to Figure 6.

F.3 Full Results for Chameleon and RAR-XL

In [Figure 13](#) and [Figure 14](#) we present token match and watermark power results for CHAMELEON and RAR-XL, complementing the TAMING results presented above in [Figure 5](#). Similarly, in [Figure 15](#) and [Figure 16](#) we present robustness results for CHAMELEON and RAR-XL, similar to those in [Figure 6](#).

Additionally, we report FID results on CHAMELEON and RAR-XL using 50,000 images as for TAMING, where for CHAMELEON we use 10 independent generations per each of the 5000 COCO validation prompts (see [Appendix E.2](#)). Our conclusions match those made in the main text. None of BASE, FT, or FT+AUGS exceed the unwatermarked FID of 19.7 for CHAMELEON and 11.5 for RAR-XL. For these models synchronization does not increase FID. We note that we obtain slightly higher baseline FID for RAR-XL compared to Yu et al. [109]; we suspect that this is due to sampling differences (we used the default parameters from the project repository) and did not investigate this further, as for our study only the *difference* in FID is of interest. For all models, we also visually confirm the quality of the generated images; we present examples in [Appendix G](#).

F.4 Distance to the Original Decoder

FID assesses the quality of the generative model by comparing distributions of generated and real images. Our generation-time watermark modifies each sampling step, which results in a completely different generation (*semantic watermarking*, see [Figure 2](#)). This makes direct comparison between image pairs not viable as a quality metric in our case. However, using the same latent representation and watermarking parameters, we can directly compare the image decoded by the original decoder with the one decoded by the RCC-finetuned decoder. While not necessarily the best proxy for image quality, this experiment gives us some insight into how much the decoder behavior changes after finetuning.

To this end, we compute the average PSNR over 1000 image pairs, where in each pair, one image is decoded using the decoder from BASE, and the other using the decoder from FT or FT+AUGS, or using both the decoder and synchronization layer from FT+AUGS+SYNC. Even with synchronization, which as expected reduces the PSNR the most, the average PSNR is 37.6 for TAMING, 39.5 for CHAMELEON, and 29.0 for RAR-XL, indicating that the final decoding does not significantly deviate from the original.

F.5 Empirical Validation of Statistical Test Correctness

We empirically validate the correctness of our statistical test by computing p-values of our watermark detector on unwatermarked images. In [Figure 17](#) we show the distribution of such p-values on 50,000 unwatermarked images generated with the base model of TAMING, using the corresponding watermark ($\gamma = 0.25, h = 1$) across 10 different random seeds for the watermark. We observe that the distribution is roughly uniform; this holds also for each individual seed.

We push this investigation further by running our watermark detector on huge token sequences, as

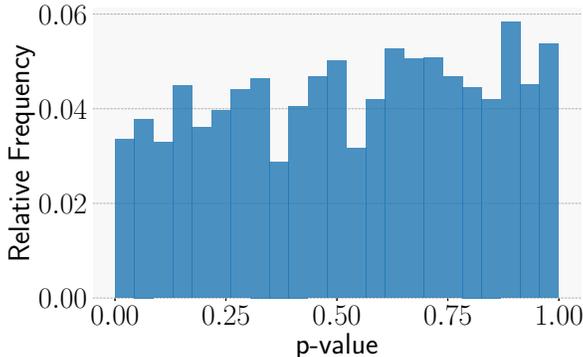


Figure 17 The distribution of p-values on unwatermarked images.

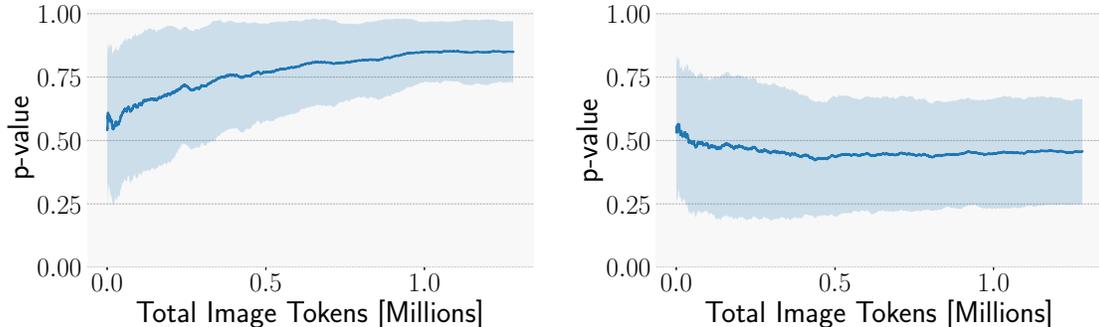


Figure 18 *Left*: On extremely long token sequences, the p-values on unwatermarked data increase as the real expected green ratio $\gamma' = 242/971$ is slightly below $\gamma = 0.25$ used in the test, making the test sound but overly conservative. *Right*: Using γ' in the test rectifies this, and p-values behave as expected.

Table 7 TPR at 1% FPR of different generation-time baselines. As these methods are not applicable to autoregressive image generation models, we show results on the `stabilityai/stable-diffusion-2-1-base` diffusion model. The transformations and attacks are the same as in [Table 2](#)

	None	Valuematic	Geometric	Adversarial Purification	Neural Compression
STABLE SIGNATURE [28]	1.00	0.71	0.71	0.39	0.54
TREE-RING [101]	1.00	0.89	0.36	0.81	0.85
GAUSSIAN SHADING [105]	1.00	1.00	0.01	1.00	1.00

in Sander et al. [84], despite those not being crucial to our usecase. Namely, for 10 random seeds, we 10 times independently concatenate 50,00 images to obtain a sequence of above 1M tokens, and compute the p-value on prefixes of this sequence of increasing length. We show the results in [Figure 18](#) (*left*, mean and standard deviation over 100 runs described above). Interestingly, while we would expect convergence around 0.5, the p-values of extremely long token sequences become as high as 0.8.

The reason for this is the relationship between the effective vocabulary size (i.e., the set of alive codes, see [Appendix E](#)) $n_{\text{alive}} = 971$ and the watermark parameter $\gamma = 0.25$. Specifically, as the number of green tokens $\gamma \cdot n_{\text{alive}} = 242.75$ in the vocabulary assumed by our test ([Equation 2](#)) is not an integer (which is never an issue for common values of $|V|$ and γ in the literature), we chose to conservatively select 242 green tokens. This keeps the statistical test sound as the *real* expected green ratio $\gamma' = 242/971 \approx 0.2492$ is smaller than $\gamma = 0.25$, but may sacrifice some power. To further confirm that this is the main cause for the observed behavior, we repeat the experiment using γ' instead of γ in [Equation 2](#) and present the results in [Figure 18](#) (*right*). We observe that the p-values now converge to slightly below 0.5, which matches results in prior work [84].

F.6 Comparison to Generation-time Watermarks for Diffusion Models

In [Table 7](#) we present results on generation-time watermarks for image models. As no prior work studies autoregressive models, we show results for diffusion models. In particular, we study `stabilityai/stable-diffusion-2-1-base` [80]. For TREE-RING [101] we use the official implementation and set the watermark pattern to `ring`, `w_channels` to 3 and use 50 inference steps for generation and testing. For STABLE SIGNATURE [28] we use the official implementation and set `unconditional_guidance_scale` to 9, `steps` to 50, and use `PLMSSampler` with `ddim_eta` set to 0. For GAUSSIAN SHADING [105] we use the official implementation, enable `chacha_encryption`, set the number of inference steps for generation and inversion to 50 and the number of bits to 256 with `channel_copy` set to 1 and `hw_copy` to 8.

We observe that each watermark is either fragile to geometric transformations or to adversarial purification and neural compression.

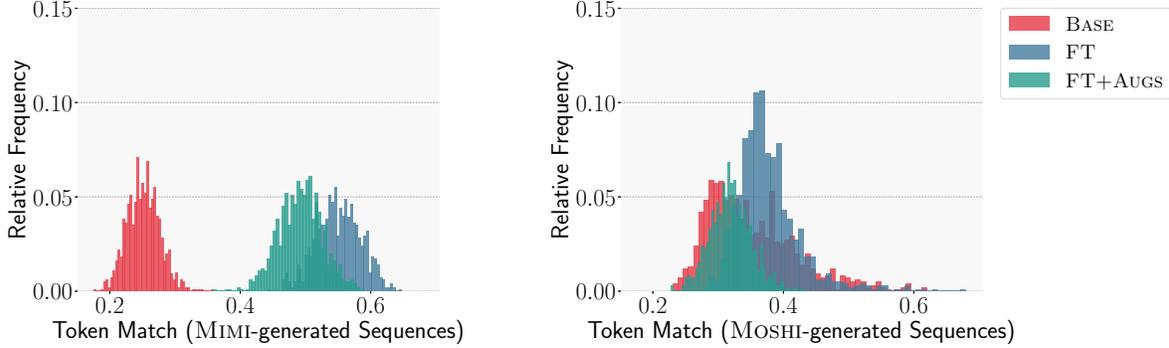


Figure 19 Token match histograms for sequences generated with MIMI (left) and MOSHI (right). We observe substantial consistency gains for MIMI-generated sequences and minimal change for MOSHI-generated ones.

F.7 Omitted Audio Results

RCC finetuning. We presented in [Appendix E.3](#) the details of our finetuning approach. Here we discuss the validation metrics and demonstrate how finetuning influences RCC and token match. We first evaluate the perceived audio quality of the generated samples using the PESQ and STOI metrics. After finetuning, we achieve a PESQ [79] score of 4.3 for both FT and FT+AugS when compared to BASE samples. STOI [89] scores reach 0.98 for FT and 0.99 for FT+AugS (we fixed the audio regularization loss weight to maintain approximately similar values for both approaches). The resulting audio is very hard to discriminate from the original, although we observe that it sometimes lead to light humming artifacts.

RCC and TM results are presented in [Figure 19](#). We observe significant token match improvements after finetuning when considering sequences of tokens generated by reconstructing 10-seconds VoxPopuli audios with the MIMI tokenizer. However, the finetuning process does not substantially improve token match for sequences generated by the MOSHI model with audio prompts. This may explain why finetuning does not significantly enhance watermark power for the MOSHI model, as observed in [Table 3](#) of [Section 5](#). A potential approach for improving watermark TPR would be to specifically finetune the model on generated sequences, which we leave for future work. (We attempted this approach for images but did not observe significant improvements, see [Appendix F.1](#) for details.)

Detailed robustness results. In [Table 3](#) of the main paper, we report the average TPR at 1% FPR over multiple audio-specific augmentations. [Table 8](#) provides the TPR for each individual augmentation that contributes to these averages, with a detailed view of how each transformation impacts watermark robustness. [Table 8](#) also reports TPRs for different δ values. For instance, increasing δ to 4.0 pushes TPR above 0.9 across most augmentations, but the resulting audio quality deteriorates noticeably compared to lower-strength settings.

G Qualitative Examples

In [Figure 20](#) (TAMING), [Figure 21](#) (CHAMELEON), and [Figure 22](#) (RAR-XL) we show qualitative examples of images generated with our watermark and images post-hoc watermarked by the baselines previously evaluated in [Section 4.2](#). We notice no observable degradation in quality, which is also confirmed by FID measurements. Interestingly, RAR-XL shows larger pixel-wise differences in decoder behavior due to finetuning (confirming PSNR results in [Appendix F.4](#)), but the finetuned decoders are still able to generate images of good quality. We remark that we use the same finetuning parameters for all models, which may not be optimal. Tuning the hyperparameters specifically for RAR-XL may thus be able to achieve similar watermark strength and robustness with smaller decoder changes.

Table 8 TPR at 1% FPR for the different tokenizer models, watermarking at different values for δ and different augmentation strengths. Audios are generated with MOSHI using the prompts described in [Appendix E.3](#).

Transformation	$\delta = 0.5$			$\delta = 2.0$			$\delta = 4.0$		
	BASE	FT	FT+ AUGS	BASE	FT	FT+ AUGS	BASE	FT	FT+ AUGS
<i>Identity</i>									
Identity	0.31	0.57	0.46	0.98	0.99	0.99	0.99	0.99	0.99
<i>Time-frequency</i>									
Speed 0.75	0.06	0.04	0.05	0.09	0.03	0.08	0.19	0.07	0.17
Speed 0.9	0.07	0.07	0.08	0.27	0.20	0.30	0.63	0.49	0.70
Speed 1.1	0.04	0.03	0.04	0.21	0.06	0.20	0.59	0.14	0.55
Speed 1.25	0.02	0.01	0.02	0.09	0.02	0.07	0.26	0.04	0.19
Crop 0.5	0.06	0.04	0.06	0.24	0.15	0.23	0.49	0.28	0.46
Crop 0.7	0.06	0.05	0.07	0.30	0.19	0.33	0.59	0.35	0.54
Crop 0.9	0.07	0.05	0.08	0.37	0.21	0.36	0.69	0.40	0.63
Shift 10.0	0.09	0.09	0.14	0.56	0.33	0.80	0.91	0.68	0.97
Shift 20.0	0.06	0.04	0.03	0.26	0.12	0.11	0.65	0.32	0.30
Shift 40.0	0.06	0.08	0.03	0.17	0.24	0.12	0.48	0.64	0.39
<i>Valuetric</i>									
Bandpass (1000, 8000)	0.03	0.04	0.16	0.10	0.15	0.45	0.28	0.38	0.85
Bandpass (300, 3000)	0.15	0.15	0.18	0.66	0.69	0.95	0.92	0.92	0.98
Bandpass (500, 5000)	0.10	0.17	0.18	0.45	0.53	0.94	0.83	0.85	0.98
Boost 50	0.37	0.27	0.22	0.98	0.98	0.97	0.98	0.99	0.98
Boost 90	0.21	0.19	0.14	0.96	0.93	0.88	0.98	0.98	0.98
Duck 50	0.14	0.15	0.12	0.81	0.84	0.81	0.98	0.98	0.97
Duck 90	0.11	0.09	0.14	0.48	0.30	0.58	0.83	0.63	0.90
Echo (0.1, 0.2)	0.12	0.20	0.13	0.87	0.97	0.93	0.98	0.99	0.98
Echo (0.3, 0.5)	0.05	0.09	0.07	0.55	0.79	0.60	0.89	0.97	0.93
Echo (0.5, 0.7)	0.04	0.05	0.04	0.38	0.56	0.40	0.77	0.91	0.79
Highpass 100	0.27	0.49	0.40	0.98	0.99	0.99	0.98	0.99	0.99
Highpass 1000	0.02	0.07	0.11	0.09	0.44	0.42	0.23	0.80	0.81
Highpass 500	0.04	0.16	0.19	0.32	0.81	0.95	0.80	0.98	0.98
Lowpass 1000	0.00	0.01	0.01	0.07	0.07	0.06	0.27	0.23	0.22
Lowpass 3000	0.33	0.34	0.39	0.97	0.98	0.98	0.99	0.98	0.99
Lowpass 8000	0.31	0.40	0.46	0.98	0.99	0.99	0.99	0.99	0.99
MP3 128	0.28	0.48	0.44	0.97	0.99	0.98	0.99	0.99	0.99
MP3 16	0.30	0.32	0.25	0.97	0.98	0.98	0.99	0.99	0.99
MP3 64	0.27	0.43	0.43	0.97	0.99	0.99	0.98	0.99	0.99
Noise 0.001	0.24	0.04	0.47	0.97	0.38	0.99	0.98	0.72	0.99
Noise 0.01	0.03	0.01	0.31	0.50	0.01	0.98	0.86	0.01	0.98
Noise 0.05	0.05	0.00	0.04	0.17	0.00	0.29	0.39	0.00	0.63
Pink 0.01	0.24	0.49	0.43	0.97	0.98	0.99	0.98	0.98	0.99
Pink 0.05	0.17	0.18	0.32	0.97	0.97	0.98	0.98	0.98	0.98
Pink 0.1	0.07	0.06	0.14	0.93	0.80	0.96	0.97	0.96	0.97
Smooth 0.001	0.06	0.04	0.10	0.44	0.24	0.79	0.83	0.52	0.96
Smooth 0.005	0.01	0.00	0.10	0.02	0.01	0.58	0.07	0.01	0.90
Smooth 0.01	0.01	0.01	0.07	0.02	0.02	0.38	0.04	0.03	0.76
UpDown Res. 24000	0.31	0.57	0.46	0.98	0.99	0.99	0.99	0.99	0.99
UpDown Res. 36000	0.30	0.52	0.46	0.98	0.99	0.99	0.99	1.00	0.99
UpDown Res. 48000	0.31	0.52	0.46	0.98	0.99	0.99	0.99	1.00	0.99
<i>Neural Compression</i>									
DAC	0.24	0.17	0.38	0.97	0.96	0.99	0.98	0.98	0.99
EnCodec	0.07	0.06	0.09	0.62	0.55	0.69	0.91	0.91	0.94

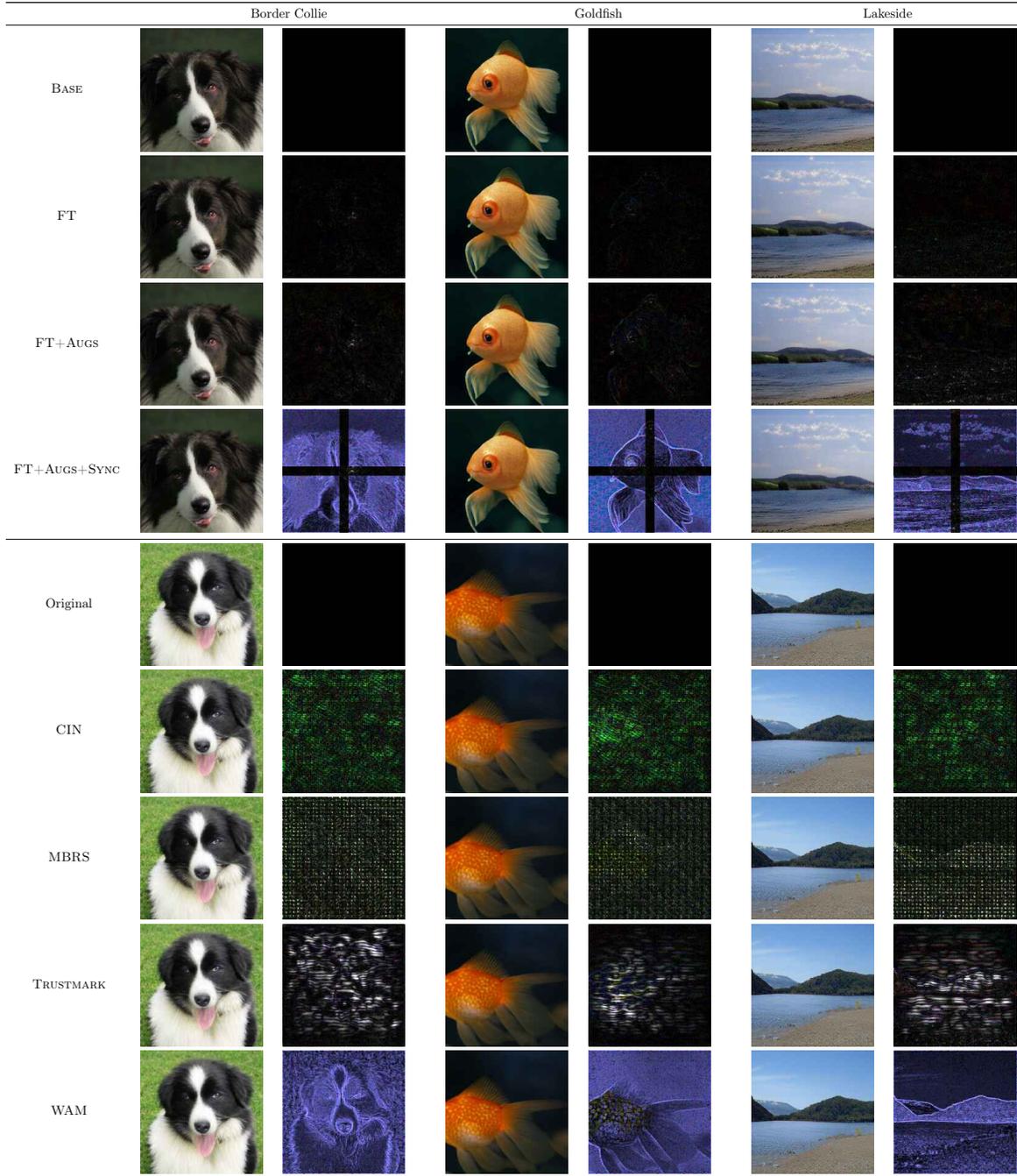


Figure 20 Qualitative results on TAMING with samples from 3 of the ImageNet-1k classes. The left column shows the images and the right column the diffs. For variants of our watermark (*top*) the diff is computed w.r.t. the BASE decoder (as there is no notion of an original unwatermarked image). For post-hoc baselines (*bottom*) the diff is computed w.r.t. the original image. All diffs are displayed with the same postprocessing applied for visibility, namely $\text{clip}(|a - b| \cdot 30, 0, 255)$, where a and b are pixel values of the two images in range $[0, 255]$.

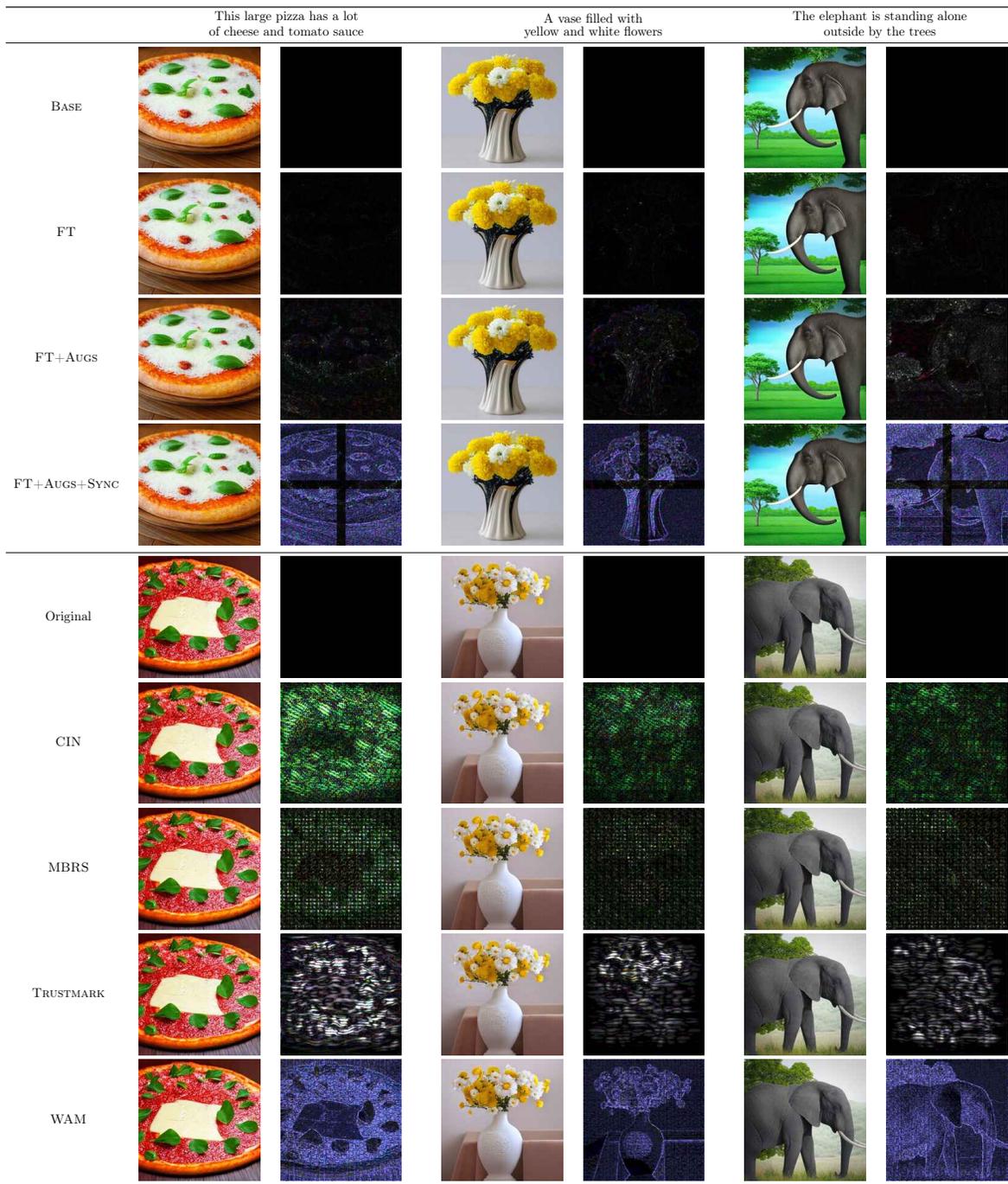


Figure 21 Qualitative results on CHAMELEON with 3 COCO validation prompts. The left column shows the images and the right column the diffs. For variants of our watermark (*top*) the diff is computed w.r.t. the BASE decoder (as there is no notion of an original unwatermarked image). For post-hoc baselines (*bottom*) the diff is computed w.r.t. the original image. All diffs are displayed with the same postprocessing applied for visibility, namely $\text{clip}(|a - b| \cdot 30, 0, 255)$, where a and b are pixel values of the two images in range $[0, 255]$.

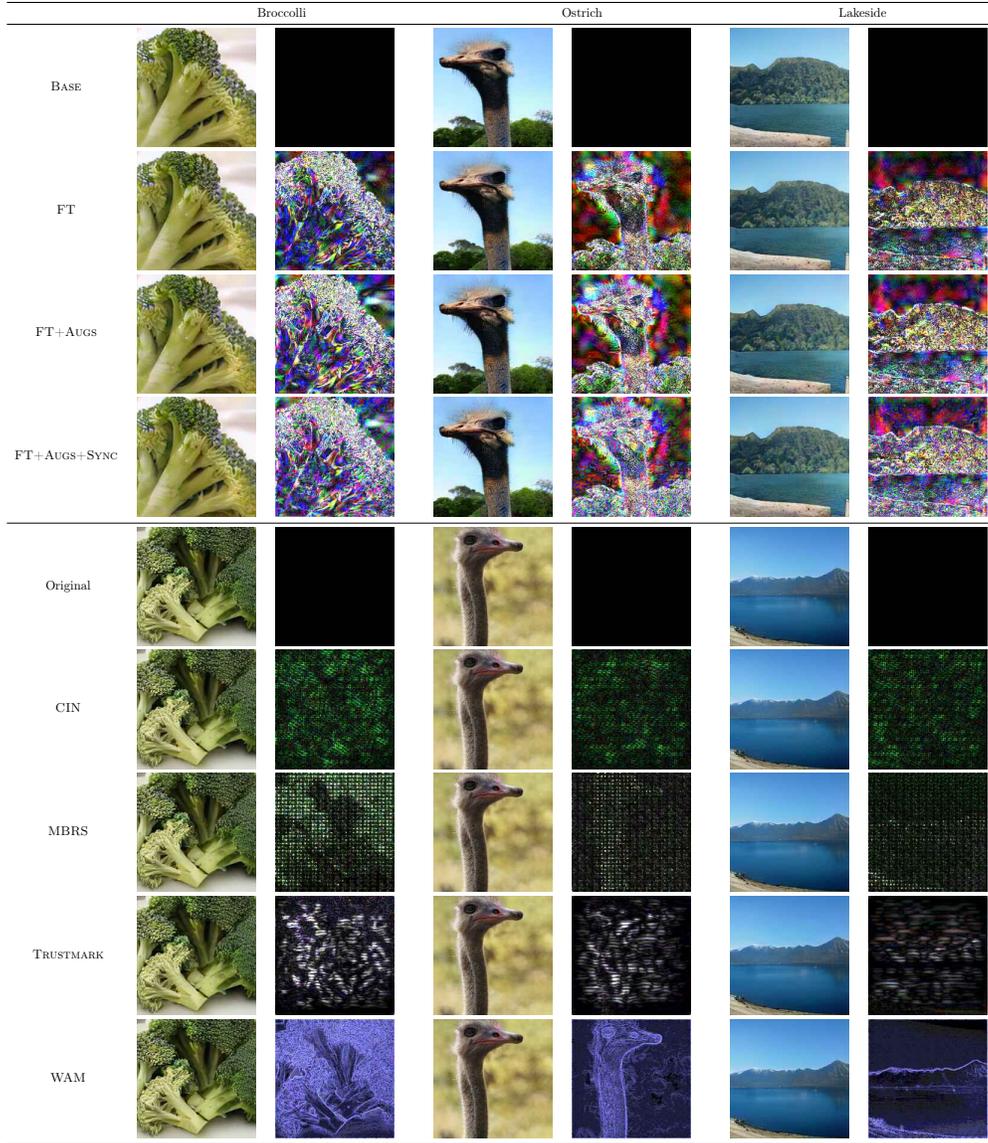


Figure 22 Qualitative results on RAR-XL with samples from 3 of the ImageNet-1k classes, as in Figure 20 and Figure 21. We observe bigger pixel-wise differences of finetuned decoders for our watermark compared to other models, but no observable degradation in quality, confirmed by FID. More elaborate tuning of the finetuning hyperparameters (we use the same ones for all models) could achieve same results with smaller decoder changes.

H On Joint Watermarking of Interleaved Modalities

In this section, we extend our discussion on joint watermarking of interleaved modalities from Section 4.3, provide omitted experimental details, and present extended experimental results.

Experimental details. We query CHAMELEON with 1000 prompts, each generated from one of ImageNet-1K classes, asking the model to teach the user about the notion represented by the class label and illustrate it. We use $\gamma = 0.25$ and set $h = 0$ for images and $h = 1$ for text. Two example interactions are shown in Appendix H.1. For text corruption we explore percentages in $[0, 60]$. For Gaussian noise corruption of images, we use $\sigma \in \{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$.

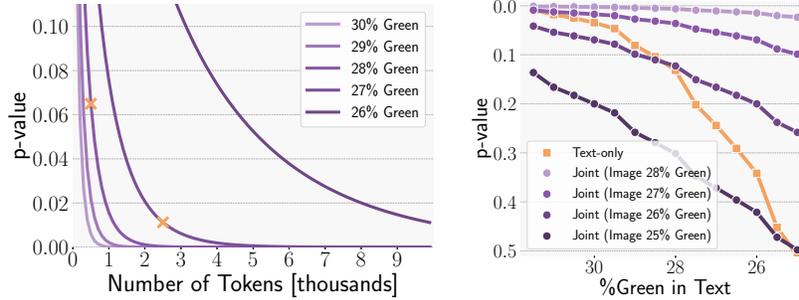


Figure 24 *Left*: Using more tokens at the same green ratio reduces the p-value. As orange crosses show, using $5\times$ more tokens can improve the p-value even if the green ratio drops ($28\% \rightarrow 27\%$). *Right*: Simulated results similar to Figure 7. Both axes are flipped to match Figure 7: x-axis reduces the green% (*weaker signal*) and y-axis reduces the p-value (*stronger detection*). As images carry many tokens, merging e.g., text with 28% green tokens with an image with 26% green tokens still improves detection.

Full experimental results. Extended results are shown in Figure 23. We observe that for high-quality images where the watermark signal is preserved, joint detection is always beneficial, while for highly corrupted images, it almost never is. Between these two, joint detection becomes beneficial starting from some text corruption level. The intuitive understanding is that joint detection that integrates a *better quality* signal is always beneficial. Perhaps unexpectedly, it can be also beneficial when signal is of slightly lower quality if it sufficiently increases the number of tokens. This is important as a single image consists of a large number of tokens, in particular 1024 for CHAMELEON, while the average text length in our experiment is ≈ 227 .

When is joint detection beneficial? First, to elaborate on our claim from the main paper that *scoring more equally watermarked tokens improves power*, we plot the p-value of the watermark detection as a function of token length for a *fixed green ratio* $S/(T-h)$ (Figure 24, left). For each line, increasing the number of tokens can rapidly improve the detection p-value. To illustrate the point made above that increasing the number of tokens can be beneficial even if the green ratio drops, note the two orange crosses. Increasing the number of tokens fivefold ($500 \rightarrow 2500$) while reducing the green ratio from 28% to 27% has still dropped the p-value from around 0.06 to almost 0.01 which for FPR of 1% is the passing threshold.

Rounding the average text length in our experiment to 256, the fivefold increase corresponds exactly to the average case we observed ($256 \rightarrow 1024 + 256$). We simulate this case (a combination of 256 text and 1024 image tokens) while fixing the green ratio for both modalities. Our results in Figure 24 (right) show a similar trend as in our above experiment on real data. For example, merging text with 28% green tokens with an image with only 26% green tokens still improves the p-value.

These results illustrate the benefits of watermarking modalities jointly. In contrast, if text and images were watermarked separately, to reject the null hypothesis of *this online article was not generated by my model \mathcal{M}* in a statistically sound way, we would need to take the maximum of the p-values of the two individual tests, i.e., the improvements seen above would not be possible.

Outlook. We hope this investigation opens the door to more detailed studies of cross-modal watermarking. Other interesting questions include the impact of transformations on the watermark *across modalities*, and the relationship between using more tokens from the same or a different modality. Intuitively, as vocabularies are disjoint, for long sequences the former may add many duplicate tokens, which do not affect the p-value. On the other hand, the probability of such duplicates may be higher for some modalities: in the first example in Appendix H.1, the image mostly contains white tokens.

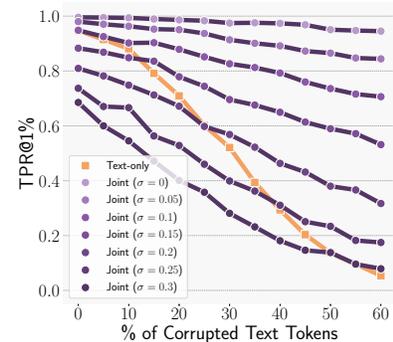


Figure 23 Full results on joint watermark detection, extending Figure 7.

H.1 Example Interactions

We show two real interactions with CHAMELEON from our joint watermark experiment (Section 4.3), along with the corresponding watermark detection results. In the detector view, for both modalities, ■ denotes tokens scored green, ■ denotes tokens scored red, and □ those ignored due to duplication (e.g., plain white visual tokens in the first example, we use gray for visibility).



User Prompt

I am wondering about the following: acoustic guitar. Can you teach me something about it in a way that is easy to understand and also generate a picture of it?



Chameleon

Certainly, I'd be happy to help!

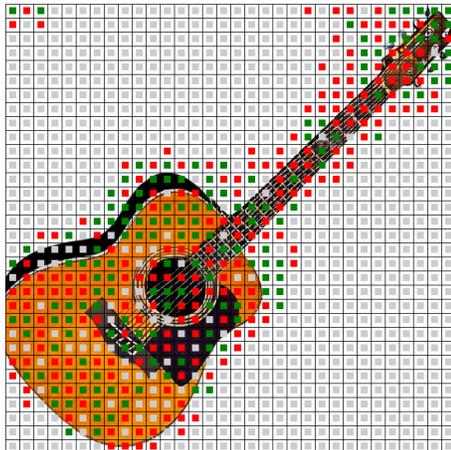
The acoustic guitar, as shown in the image below, is a popular instrument for playing music. It is made of a wooden body, a neck, and strings that are plucked or strummed to produce sound. The sound of an acoustic guitar is created by the vibrations of the strings, which are then amplified by the body of the guitar.



Watermark Detector ($p = 1.7 \cdot 10^{-16}$)

Certainly, I'd be happy to help!

The acoustic guitar, as shown in the image below, is a popular instrument for playing music. It is made of a wooden body, a neck, and strings that are plucked or strummed to produce sound. The sound of an acoustic guitar is created by the vibrations of the strings, which are then amplified by the body of the guitar.





User Prompt

I am wondering about the following: thatch. Can you teach me something about it in a way that is easy to understand and also generate a picture of it?



Chameleon

Sure! Thatch refers to the plant material used to cover roofs and walls in traditional or indigenous architecture. It is made from various types of grasses, leaves, or other vegetation. Here is an image of an example of thatch, used as roofing material in the construction of this house. Thatch is often made from locally available resources, such as reeds, rushes, or grasses, and is often woven or tied together to create a durable and effective roofing material. It requires regular maintenance to keep it in good condition and can last for many years with proper care. In some areas, thatch is still used today as a traditional building material, while in others, it has been largely replaced by modern roofing systems.



Watermark Detector ($p = 1.2 \cdot 10^{-34}$)

Sure! Thatch refers to the plant material used to cover roofs and walls in traditional or indigenous architecture. It is made from various types of grasses, leaves, or other vegetation. Here is an image of an example of thatch, used as roofing material in the construction of this house. Thatch is often made from locally available resources, such as reeds, rushes, or grasses, and is often woven or tied together to create a durable and effective roofing material. It requires regular maintenance to keep it in good condition and can last for many years with proper care. In some areas, thatch is still used today as a traditional building material, while in others, it has been largely replaced by modern roofing systems.

