PREDICTING TOKEN-LEVEL WATERMARK RETENTION FROM UNWATERMARKED IMAGES FOR AUTOREGRESSIVE IMAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Token-level watermarks make autoregressive (AR) image models more likely to sample tokens selected by a secret key, creating a signal that identifies watermarked images. However, the signal can remain detectable after image processing on one AR image model and become difficult to detect on another. Existing evaluation requires generating watermarked images for each model and watermark. In this paper we propose a metric η for the signal retained when generated tokens are rendered into an image and recovered for detection. We prove that higher η gives higher detection success for a fixed model, watermark, and sampling bias at a fixed false-positive rate under stated statistical assumptions. We then propose RenderGap, an evaluation method that predicts η through a derived approximation using only unwatermarked generations. RenderGap compares generated tokens with tokens recovered from the images and scores both using the watermark key. Experiments with independent calibration on various AR image models show median absolute relative prediction errors of 0.009–0.073 across JPEG, blur, and tokenizer variants. RenderGap therefore evaluates retention for a given AR image model and watermark before watermarked images are generated.

1 INTRODUCTION

Token-level watermarking embeds a hidden signal during autoregressive (AR) image generation to verify the origin of generated images. An AR image model samples discrete visual tokens and renders them into an image, which the detector converts back into tokens. Kirchenbauer et al. (2023) proposed a token-level watermark for large language models that biases sampling toward a *green list* of tokens selected by a secret key. Recent studies extended token-level watermarks to AR image models (Jovanović et al., 2025; Kerner et al., 2025; Lukovnikov et al., 2026; Tong et al., 2025; Wu et al., 2025; Hui et al., 2025; Meintz et al., 2025), but watermark detection after image processing varies across models. Jovanović et al. (2025) show that the tokens recovered from an image can differ from the generated tokens, weakening the watermark signal. In this paper, we predict how much of a given watermark's signal an AR image model preserves, without generating watermarked images.

To compare watermark detection across AR image models, existing studies report the true-positive rate (TPR) using watermarked images and unwatermarked reference images (Figure 1(b)). Increasing the sampling bias can produce a high detection rate even when the tokens recovered from the image retain only a small part of the watermark signal. The detection rate alone therefore does not show how well an AR image model preserves a given watermark. Measuring preservation requires comparing the watermark signal in the generated tokens with that in the tokens recovered from the image.

We first propose the metric η to measure how much watermark signal is preserved through the *render channel*, which is the process of decoding generated tokens into an image, applying optional image processing, and then re-encoding the image into tokens. For a given keyed watermark, η is the squared ratio of the average standardized green-count score after the render channel to the corresponding average score before it. We theoretically prove that higher η gives a higher true-positive rate for a fixed model and keyed watermark at a fixed sampling bias and false-positive rate (FPR), under the statistical assumptions of Proposition 1. However, measuring η also requires generated watermarked images and unwatermarked reference images.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

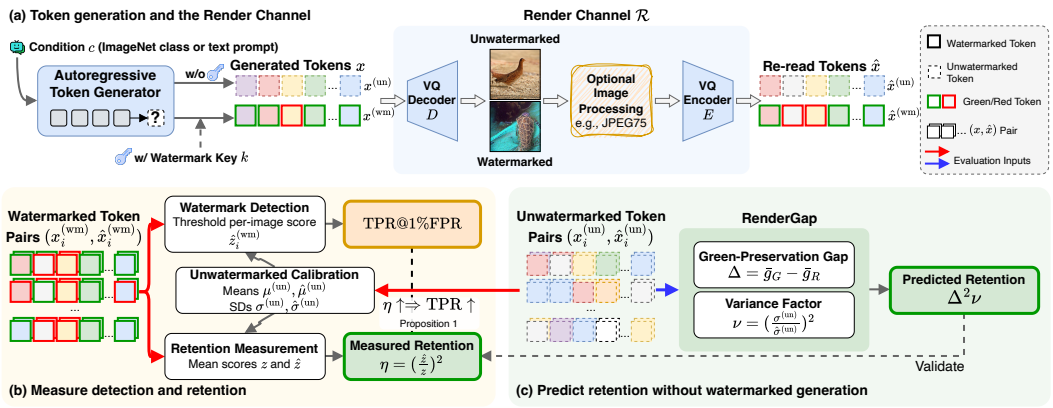


Figure 1: **Overview of watermark retention evaluation.** (a) Generated tokens x are decoded into an image and re-encoded into $\hat{x}$ after optional processing, forming the render channel $\mathcal{R}$. (b) Unwatermarked green-count means and standard deviations calibrate the scores before and after rendering. Hats mark the after-render quantities. Watermarked pairs give per-image detection scores and mean scores for measured retention η. (c) RenderGap uses only unwatermarked pairs, scored with the same key, to predict retention as $\Delta^2\nu$. Proposition 1 relates retention to TPR at fixed generation and FPR. The dashed arrow compares predicted and measured retention. We derive $\eta \approx \Delta^2\nu$ and test it against measured retention.

We then propose **RenderGap**, an evaluation method that predicts η using only unwatermarked images. After generation, it applies the watermark key to compare generated tokens with those recovered through the render channel (Figure 1(c)). The *green-preservation gap* Δ measures how much more often a green token returns green than a token outside the green list does. The *variance factor* ν compares the variation in green counts across unwatermarked images before and after the render channel. Our analysis gives $\eta \approx \Delta^2\nu$ under certain conditions, which RenderGap uses as its prediction. The same unwatermarked token pairs can be reused to evaluate multiple keys.

Experiments on Janus-Pro-1B and LlamaGen-L show that RenderGap's predictions agree with independently measured retention across the tested JPEG and blur settings. Among three RAR-XL tokenizer variants, RenderGap identifies the one with the highest measured retention under JPEG compression using only unwatermarked images.

2 Related Work

Token-level watermarking for AR image generation. Green-list watermarking biases token sampling toward a subset selected by a secret key (Kirchenbauer et al., 2023). Unigram-Watermark keeps that subset fixed across positions (Zhao et al., 2024). Recent methods embed watermarks in AR image tokens (Kerner et al., 2025; Tong et al., 2025; Yilmaz et al., 2026; Wu et al., 2025; Hui et al., 2025; Meintz et al., 2025). WMAR improves token recovery by fine-tuning the VQ encoder and decoder (Jovanović et al., 2025). ClusterMark groups visually similar tokens to preserve watermark labels when token identities change (Lukovnikov et al., 2026). Decoder choice also affects token recovery (Rebuffi et al., 2026). While these methods improve embedding or recovery, RenderGap evaluates an AR image model's retention of a specified watermark signal.

Watermark evaluation and robustness. Text-watermark studies analyze detection power (Li et al., 2025a;b) and robustness to paraphrasing and mixing watermarked text into longer documents (Kirchenbauer et al., 2024). WAVES evaluates detection and identification across distortions and attacks (An et al., 2024), while W-Bench focuses on regeneration and global or local generative edits (Lu et al., 2025). Saberi et al. (2024) bound the combined false-negative and false-positive error under diffusion purification for image watermarks with small perturbations. Transfer attacks optimize perturbations against surrogate models without access to the target detector (Hu et al., 2025), and removal and forgery attacks also affect AR-image watermarks (Müller et al., 2026). Provenance tests based on quantization errors use unwatermarked AR images to determine whether a specified

model generated an image (Zhao et al., 2026). The detection benchmarks measure performance on watermarked test images, whereas RenderGap predicts signal retention through a specified image round trip using only unwatermarked generations.

3 PROBLEM SETUP

AR image generation. An AR image model combines an AR token generator p_θ with a vector-quantized (VQ) decoder (van den Oord et al., 2017; Esser et al., 2021). Given a conditioning input c, the generator with parameters θ samples a length-T sequence $x = (x_1, \ldots, x_T)$ from a vocabulary $\mathcal{V}$ of discrete visual tokens. At generation step t, the preceding tokens $x_{<t}$ and c determine the next-token probabilities $p_t(v) = p_\theta(v \mid x_{<t}, c)$ for $v \in \mathcal{V}$. The decoder D maps the completed sequence to an image $I = D(x)$. Here c is a text prompt for Janus-Pro (Chen et al., 2025) and an ImageNet class label (Russakovsky et al., 2015) for LlamaGen and RAR (Sun et al., 2024; Yu et al., 2024). Multiple images can share the same conditioning input c, with a token sequence sampled separately for each image.

Green-list watermark. A secret key k selects a set $G_t(k) \subset \mathcal{V}$ of *green* tokens, and the remaining tokens are *red*. The indicator $w_t(v) = \mathbb{1}[v \in G_t(k)]$ records membership. Following Kirchenbauer et al. (2023), watermarking adds a bias $\delta > 0$ to green-token logits before sampling. Writing $\ell_t(v)$ for token v's pre-softmax score, the unwatermarked and watermarked probabilities are

$$p_t(v) = \frac{e^{\ell_t(v)}}{\sum_{u\in\mathcal{V}} e^{\ell_t(u)}}, \qquad q_t(v;\delta) = \frac{e^{\ell_t(v)+\delta w_t(v)}}{\sum_{u\in\mathcal{V}} e^{\ell_t(u)+\delta w_t(u)}}. \tag{1}$$

The next token is sampled from q_t during watermarked generation and from p_t without watermarking. Our main constructions fix the green sets independently of the generated prefix. Appendix E.5 studies a previous-token rule separately.

We measure the change in the full next-token distribution by the Kullback–Leibler (KL) divergence,

$$\mathrm{KL}(q_t\|p_t) = \sum_{v\in\mathcal{V}} q_t(v;\delta)\log\frac{q_t(v;\delta)}{p_t(v)}, \qquad \varepsilon = \mathbb{E}_{c,x}\left[\frac{1}{T}\sum_{t=1}^{T}\mathrm{KL}(q_t\|p_t)\right]. \tag{2}$$

Both distributions in each KL term use the same conditioning and prefix. The expectation averages over conditioning inputs c from the evaluation protocol and sequences x generated by successively sampling from q_t. Its empirical value averages the recorded KL terms over images and positions. Natural logarithms give the per-token budget ε in nats (Appendix B.2).

Render channel. The generated tokens are decoded by VQ decoder D into an image, optionally processed by $\mathcal{A}$, and re-encoded by the VQ encoder E for detection:

$$x \xrightarrow{D} I \xrightarrow{\mathcal{A}} I' \xrightarrow{E} \hat{x}, \qquad \hat{x} = \mathcal{R}(x), \quad \mathcal{R} = E \circ \mathcal{A} \circ D. \tag{3}$$

We call this token-to-image-to-token mapping $\mathcal{R}$ the *render channel*. It uses no watermark key, and the re-read tokens $\hat{x}$ can differ from x even when $\mathcal{A}$ leaves the image unchanged. Before and after refer to the two ends of this complete round trip. A model, keyed watermark construction, and render channel specify an evaluation setting.

Watermark detection. The detector flags images with unusually many green re-read tokens, using unwatermarked images processed through the same render channel to calibrate its threshold. The true-positive rate (TPR) is the fraction of watermarked images flagged, and the false-positive rate (FPR) is the fraction of unwatermarked images flagged. Following Jovanović et al. (2025) and Lukovnikov et al. (2026), we report TPR@1%FPR: an image is flagged when its score strictly exceeds the 99th percentile of the unwatermarked calibration scores. This threshold targets 1% FPR, but the achieved rate on new images depends on the finite calibration sample. We define the scores and their before/after comparison next.

4 MEASURING AND PREDICTING WATERMARK RETENTION

4.1 RETENTION AND DETECTION

Direct measurement compares watermarked images with unwatermarked calibration images (Figure 1(b)). We use superscripts (un) and (wm) to distinguish generation without and with watermark bias, and a hat to mark quantities after the render channel. For image i, the generated sequence x_i is paired with its own re-read sequence $\hat{x}_i$. The two image pools can have different sizes. The green counts for unwatermarked and watermarked images are

$$\begin{aligned} S_i^{(\mathrm{un})} &= \sum_{t=1}^{T} w_t(x_{i,t}^{(\mathrm{un})}), \qquad \hat{S}_i^{(\mathrm{un})} = \sum_{t=1}^{T} w_t(\hat{x}_{i,t}^{(\mathrm{un})}), \\ S_i^{(\mathrm{wm})} &= \sum_{t=1}^{T} w_t(x_{i,t}^{(\mathrm{wm})}), \quad \hat{S}_i^{(\mathrm{wm})} = \sum_{t=1}^{T} w_t(\hat{x}_{i,t}^{(\mathrm{wm})}). \end{aligned} \tag{4}$$

Let $\mu^{(\mathrm{un})}$ and $\hat{\mu}^{(\mathrm{un})}$ be the means of the unwatermarked counts before and after rendering, and let $\sigma^{(\mathrm{un})} > 0$ and $\hat{\sigma}^{(\mathrm{un})} > 0$ be their standard deviations (SDs). The corresponding watermarked means are $\mu^{(\mathrm{wm})}$ and $\hat{\mu}^{(\mathrm{wm})}$. Each count is standardized using the unwatermarked mean and SD at the same stage:

$$\begin{aligned} z_i^{(\mathrm{un})} &= \frac{S_i^{(\mathrm{un})} - \mu^{(\mathrm{un})}}{\sigma^{(\mathrm{un})}}, \qquad \hat{z}_i^{(\mathrm{un})} = \frac{\hat{S}_i^{(\mathrm{un})} - \hat{\mu}^{(\mathrm{un})}}{\hat{\sigma}^{(\mathrm{un})}}, \\ z_i^{(\mathrm{wm})} &= \frac{S_i^{(\mathrm{wm})} - \mu^{(\mathrm{un})}}{\sigma^{(\mathrm{un})}}, \quad \hat{z}_i^{(\mathrm{wm})} = \frac{\hat{S}_i^{(\mathrm{wm})} - \hat{\mu}^{(\mathrm{un})}}{\hat{\sigma}^{(\mathrm{un})}}. \end{aligned} \tag{5}$$

Detection applies the same threshold to the per-image after-scores. To measure retention, we average the scores over watermarked images, giving the population means

$$z = \frac{\mu^{(\mathrm{wm})} - \mu^{(\mathrm{un})}}{\sigma^{(\mathrm{un})}}, \qquad \hat{z} = \frac{\hat{\mu}^{(\mathrm{wm})} - \hat{\mu}^{(\mathrm{un})}}{\hat{\sigma}^{(\mathrm{un})}}. \tag{6}$$

For $z \neq 0$, we define retention as

$$\eta := \left(\frac{\hat{z}}{z}\right)^2. \tag{7}$$

Thus $\eta = 1$ means unchanged squared mean score, while $\eta < 1$ means a decrease and $\eta > 1$ an increase. In experiments, these symbols denote sample estimates, and the resulting ratio is measured η (Appendix B.4).

Proposition 1 (Retention and detection)**.** *Fix the AR token generator p_θ, the distribution of conditioning inputs c, the keyed green lists $G_t(k)$, and the watermark bias δ, so $z > 0$ is fixed. Suppose the per-image after-scores satisfy*

$$\hat{z}_i^{(\mathrm{un})} \sim \mathcal{N}(0,1), \qquad \hat{z}_i^{(\mathrm{wm})} \sim \mathcal{N}(\hat{z},1), \qquad \hat{z} \geq 0.$$

At false-positive rate $\lambda \in (0,1)$,

$$\mathrm{TPR} = 1 - \Phi\big(\Phi^{-1}(1-\lambda) - \sqrt{\eta}\, z\big), \tag{8}$$

where Φ and Φ^{-1} are the standard normal cumulative distribution and quantile functions. Consequently, TPR increases with η at fixed z and λ.

The proof and the corresponding joint-image sample-size relation are in Appendix A. Across AR token generators or watermark constructions, detection success depends on both retention and the generated-token mean score z (Appendix D.1).

Verifying that TPR increases with retention. With generation and the green list fixed, we vary image processing or the tokenizer. Higher retention generally corresponds to higher TPR@1%FPR (Figure 2), consistent with Proposition 1. Appendix C.1 reports the protocol and Gaussian approximation errors.

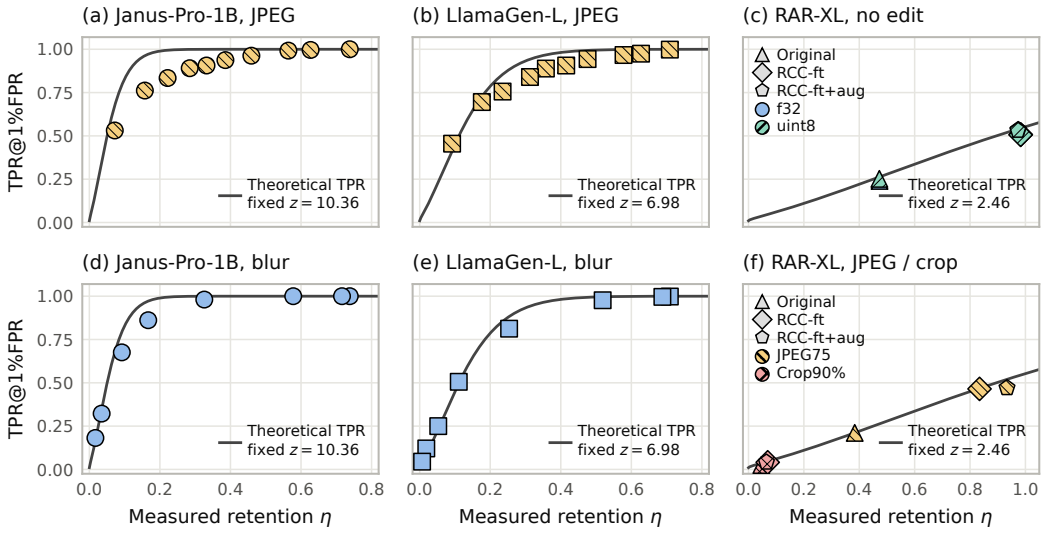


Figure 2: **Retention and detection at fixed generation.** Solid curves give Proposition 1's Gaussian prediction of TPR@1%FPR as retention η varies. The mean score z of watermarked tokens before rendering is fixed at its measured value for each model. Markers show measured retention and observed TPR. Global key 42, with bias 1 for Janus-Pro-1B/LlamaGen-L and 0.4 for RAR-XL. Panels (a,b) vary JPEG quality and panels (d,e) vary Gaussian-blur radius. Panels (c,f) compare RAR tokenizers under the indicated processing paths. RAR-XL uses its original tokenizer or variants fine-tuned for reverse cycle-consistency (RCC), without image augmentations (RCC-ft) or with them (RCC-ft+aug) (Jovanović et al., 2025).

4.2 RenderGap prediction from unwatermarked round trips

Although η quantifies watermark retention through the render channel, measuring it directly still requires watermarked images. We therefore propose RenderGap, an evaluation method that predicts η using only unwatermarked images. Specifically, we generate tokens without watermark bias ($\delta = 0$), pass them through the render channel, and apply the watermark key to both the generated and re-read token sequences (Figure 1(c)). The paired green/red labels estimate the *green-preservation gap*

$$\Delta = \bar{g}_G - \bar{g}_R, \qquad \bar{g}_G = \Pr[\hat{x}_t \in G_t \mid x_t \in G_t], \quad \bar{g}_R = \Pr[\hat{x}_t \in G_t \mid x_t \notin G_t]. \tag{9}$$

We pool token pairs across unwatermarked images and positions: $\bar{g}_G$ divides the number of green-to-green pairs by the number of generated green tokens, and $\bar{g}_R$ divides red-to-green pairs by generated red tokens. Thus each conditional proportion weights positions by their number of source green or red tokens. Variation in the per-image green counts gives the *variance factor* $\nu = (\sigma^{(\mathrm{un})}/\hat{\sigma}^{(\mathrm{un})})^2$. Substituting the two mean scores from Eq. 6 gives the exact identity

$$\eta = \left(\frac{\hat{\mu}^{(\mathrm{wm})} - \hat{\mu}^{(\mathrm{un})}}{\mu^{(\mathrm{wm})} - \mu^{(\mathrm{un})}} \right)^2 \nu. \tag{10}$$

The following result connects that mean-count ratio to unwatermarked transitions.

Proposition 2 (Predicting retention from unwatermarked transitions)**.** *Fix each next-token distribution p_t, green list G_t, and a token-local transition matrix M that is independent of position and other tokens. Its entry $M_{uv} = \Pr[\hat{x}_t = v \mid x_t = u]$ is the probability that a generated token with ID u is re-encoded as token v after passing through the render channel. Let $\pi_t = \Pr_{p_t}[x_t \in G_t] \in (0, 1)$ be the probability of sampling a green token at position t without watermark bias, and let $d_t = \Pr[\hat{x}_t \in G_t \mid x_t \in G_t] - \Pr[\hat{x}_t \in G_t \mid x_t \notin G_t]$ be the green-preservation gap at that position. The weighted green-preservation gap Δ_* averages these position-specific gaps with weights $\pi_t(1 - \pi_t)$. With positive unwatermarked green-count SDs, as the watermark bias $\delta \to 0^+$,*

$$\eta \longrightarrow \Delta_*^2 \nu, \qquad \Delta_* := \frac{\sum_t \pi_t(1 - \pi_t) d_t}{\sum_t \pi_t(1 - \pi_t)}. \tag{11}$$

The weighted gap Δ_* equals the pooled gap Δ when π_t is constant across positions.

Table 1: **Predicted and independently measured retention.** Global key 42, with original Janus-Pro/LlamaGen tokenizers at $\delta = 1$ and RAR tokenizers at $\delta = 0.4$. Subscripts give one bootstrap SD. Ratios closest to 1 indicate agreement. RCC-ft and RCC-ft+aug are RAR tokenizers fine-tuned without and with augmentations, respectively. The paths use floating-point pixels (`f32`), 8-bit conversion (`uint8`), or JPEG quality 75.

Model / tokenizer	Processing	η	$\Delta^2\nu$	$\Delta^2\nu/\eta$
Janus-Pro-1B	`uint8`	$0.7382_{\pm 0.0157}$	$0.7315_{\pm 0.0294}$	0.99
LlamaGen-L	`uint8`	$0.7091_{\pm 0.0227}$	$0.7416_{\pm 0.0355}$	1.05
RAR-XL / Original	`f32`	$0.4732_{\pm 0.0177}$	$0.4525_{\pm 0.0170}$	0.96
RAR-XL / Original	`jpeg75`	$0.3836_{\pm 0.0186}$	$0.3533_{\pm 0.0160}$	0.92
RAR-XL / RCC-ft	`f32`	$0.9833_{\pm 0.0063}$	$0.9759_{\pm 0.0076}$	0.99
RAR-XL / RCC-ft	`jpeg75`	$0.8343_{\pm 0.0193}$	$0.8222_{\pm 0.0237}$	0.99
RAR-XL / RCC-ft+aug	`f32`	$0.9741_{\pm 0.0075}$	$0.9763_{\pm 0.0101}$	1.00
RAR-XL / RCC-ft+aug	`jpeg75`	$0.9334_{\pm 0.0126}$	$0.9257_{\pm 0.0161}$	0.99

The weights reflect how strongly each position's green-token probability responds to a small bias. The proof shows that the ratio of mean-count differences in Eq. 10 converges to Δ_* (Appendix A). RenderGap substitutes the pooled gap to obtain its finite-bias prediction,

$$\boxed{\eta \approx \Delta^2\nu.} \tag{12}$$

Both factors are computed from unwatermarked pairs: pool label transitions for Δ, and compute the two green-count SDs for ν. The variance factor accounts for the score's changing normalization. Equal null SDs give $\nu = 1$ and the simpler prediction Δ^2. Fixed contexts and token-local transitions approximate AR generation and neural tokenizers. Section 5 tests the resulting prediction at finite bias.

Verifying the retention prediction. Our experiments show that RenderGap predictions from only unwatermarked images closely match independently measured retention for the no-attack and JPEG settings in Table 1. Section 5 reports the experimental settings and full processing sweeps.

5 Experimental evaluation

5.1 Experimental settings

We evaluate RenderGap on three AR image models: Janus-Pro-1B, LlamaGen-L, and RAR-XL. For native Janus-Pro-1B and LlamaGen-L, we vary the strength of JPEG image compression and Gaussian blur while keeping their original tokenizers (Appendices D.2 and D.3). For RAR-XL, the original tokenizer and WMAR's variants fine-tuned without augmentations (RCC-ft) and with augmentations (RCC-ft+aug) process the same generated tokens (Jovanović et al., 2025). This holds the generation-side watermark signal fixed while changing how the image is decoded and re-encoded (Appendix E.3). Appendix B gives model configurations and sample counts.

We compare two ways of constructing the green list, called *Global* and *Cluster-64*. In Global, the watermark key selects one green list shared by all token positions (Zhao et al., 2024). In Cluster-64, we group codebook vectors into 64 clusters using k-means and select whole clusters using the key, following the label-preservation motivation of ClusterMark (Lukovnikov et al., 2026). Global uses green vocabulary fraction $\gamma = |G_t|/|\mathcal{V}| = 0.5$, while Cluster-64 adds whole clusters in keyed random order until at least half the vocabulary is green. This fraction is distinct from the model's probability of sampling green tokens. The fixed Cluster-64 partition is a controlled construction, whereas ClusterMark's context-dependent rule is evaluated separately in Appendix E.1.

We vary image processing between decoding and re-encoding, starting with two no-attack conditions. We either re-encode the floating-point decoder output directly (`f32`) or first convert it to an 8-bit image (`uint8`). After 8-bit conversion, we also test JPEG compression at quality 75 (`jpeg75`) and center-cropping to 90% of each dimension followed by resizing (`crop09`).

Our main independent-reference comparisons use Global key 42 at bias 1 for Janus-Pro-1B/LlamaGen-L and 0.4 for RAR-XL. We predict retention from 1,000 unwatermarked images and measure the

reference retention using a separate $4{,}000$ unwatermarked and 512 watermarked images. Generated tokens are reused across processing levels and compatible tokenizers (Appendix B.3).

5.2 Prediction across controlled changes

Table 2: **Errors in predicting watermark retention η.** For RenderGap, absolute relative error is $|\Delta^2\nu - \eta|/\eta$, where η is independently measured retention. For the baselines, replace $\Delta^2\nu$ with Δ^2 or p^2_{same}. Median and mean summarize these errors across the settings in each row. Bold marks the lowest error for each statistic. Processing sweeps have ten JPEG and eight blur settings per model. Three RAR tokenizers give nine settings across `f32`/`uint8`/`jpeg75` and three under `crop09`.

	Absolute relative error ↓					
	RenderGap ($\Delta^2\nu$)		Δ^2		p^2_{same}	
Settings	Median	Mean	Median	Mean	Median	Mean
Image-processing strengths: original tokenizers						
Janus-Pro JPEG	**0.0285**	**0.0332**	0.0593	0.0633	0.0620	0.0635
Janus-Pro blur	**0.0375**	**0.0402**	0.0400	0.0411	0.0468	0.0477
LlamaGen JPEG	0.0460	**0.0476**	**0.0436**	0.0490	0.0486	0.0512
LlamaGen blur	0.0734	0.0717	**0.0394**	**0.0580**	0.0518	0.0629
Tokenizer variants: RAR-XL						
`f32`, `uint8`, `jpeg75`	**0.0090**	0.0247	0.0114	**0.0192**	0.0209	0.0233
`crop09`	**0.5531**	0.5817	0.5926	0.5958	0.5735	**0.5797**

We test whether RenderGap predicts how image processing and tokenizer changes affect retention. For Janus-Pro-1B and LlamaGen-L, we hold each original tokenizer fixed and vary JPEG quality and blur radius at bias $\delta = 1$. JPEG quality 100 is an identity 8-bit image path without a JPEG resave and coincides with zero blur. For RAR-XL, we apply three tokenizer variants to the same generated tokens under `f32`, `uint8`, `jpeg75`, and `crop09`, at $\delta = 0.4$. Both comparisons use Global key 42 and reuse generated tokens across the tested processing conditions.

We use exact-token agreement p_{same}, the fraction of tokens recovered unchanged at the same positions, to construct a simple baseline. Suppose green and red tokens are equally likely to remain unchanged, and a changed token is green with probability γ regardless of its original label. Then $\Delta = p_{\text{same}}$, and if $\nu \approx 1$, Eq. 12 gives $\eta \approx p^2_{\text{same}}$. We compare RenderGap's $\Delta^2\nu$ with the gap-only Δ^2 and squared exact-token agreement p^2_{same}. For each predictor, we report the absolute difference from measured retention η, divided by η. All three predictors use only unwatermarked token pairs. Appendix C.4 also reports unsquared agreement and the constant prediction $\eta = 1$. In the comparisons without crop, RenderGap's median errors stay below 0.08 against the independent reference (Table 2). Unwatermarked token pairs therefore capture changes in retention as image processing or the tokenizer changes, without fitting to watermarked images. Figure 3 compares individual predictions with their independent measurements across all three architectures. The variance correction improves the Janus-Pro sweeps, while Δ^2 performs similarly or better in the other comparisons. Paired bootstrap comparisons leave the mean-error differences unresolved in every family (Appendix C.4).

Crop-and-resize produces large prediction errors (Table 2), with consistent underprediction in the per-setting comparisons (Appendix C.4, Figure 6). Crop changes the correspondence between generated and re-read token positions, which the same-position transition approximation does not represent (Appendix D.4).

5.3 Selecting a tokenizer without watermarked generation

We use the predictions to choose among RAR-XL's three compatible tokenizers while fixing the generated tokens and Global partition. Under JPEG, RenderGap selects the tokenizer fine-tuned with augmentations, which also has the highest independently measured retention (Table 1). Both fine-tuned variants preserve more than the original tokenizer and also have higher measured TPRs under this fixed-generation comparison (Table 10 in Appendix C.2). Without image processing,

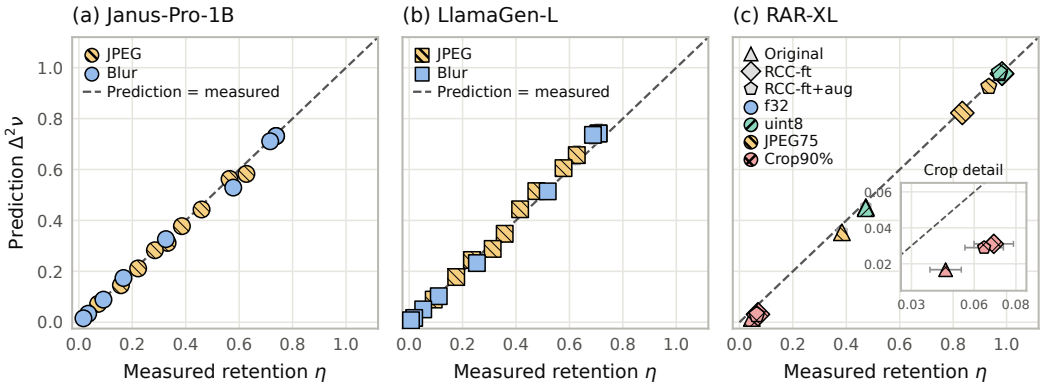


Figure 3: **Predicted versus independently measured retention.** Global key 42, with 1,000 unwatermarked images for prediction and separate 4,000 unwatermarked plus 512 watermarked images for measurement. Bias is 1 for Janus-Pro-1B/LlamaGen-L and 0.4 for RAR-XL. The diagonal marks equality, and bars give one bootstrap SD on each axis. JPEG and blur share their unchanged 8-bit control. The inset enlarges the three RAR crop cases.

Table 3: **Effects of label preservation and null variance on prediction.** (a) Original RAR-XL tokenizer, Cluster-64, key 42, $\delta = 0.4$, and an independent 128-image watermarked reference. (b) Two quantization rules with BF16 arithmetic on identical Janus-Pro-1B Global/`f32` tokens and weights. Retention averages six biases with shared null calibration. Subscripts give one bootstrap SD, and ratios closer to 1 indicate more accurate predictions.

(a) Label preservation (RAR-XL)

Channel	$\Delta^2\nu/\eta$	Δ^2/η	p^2_{same}/η
`f32`	0.97	1.01	0.80
`uint8`	0.97	1.01	0.80
`jpeg75`	1.00	1.06	0.76

(b) Variance correction (Janus-Pro-1B)

Quantization rule	Measured η	$\Delta^2\nu/\eta$	Δ^2/η
Unnormalized Euclidean	$0.0075_{\pm 0.0032}$	0.82	2.80
L_2-normalized (native)	$0.7161_{\pm 0.0272}$	0.97	1.00

the two fine-tuned variants have similar retention and their paired difference is unresolved. The predictions identify tokenizer changes that produce a substantial preservation gain.

5.4 Label preservation and variance correction

Label preservation. Token substitutions can preserve the watermark label even when exact-token agreement falls (Appendix E.2). Both gap-based predictors capture this preservation under RAR-XL's fixed Cluster-64 partition, where squared exact-token agreement underpredicts retention (Table 3(a)). This fixed partition differs from the context-dependent ClusterMark method.

Variance correction. To test when the variance factor improves prediction, we change Janus-Pro-1B's nearest-code selection while holding generated tokens and tokenizer weights fixed (Table 3(b)). Hugging Face Transformers 4.52.4 uses unnormalized Euclidean distance, whereas native Janus first L_2-normalizes the encoder output and codebook vectors. Both use bfloat16 (BF16) arithmetic and produce identical decoded images, but different re-read tokens. Unnormalized search increases the unwatermarked green-count variance, so Δ^2 overestimates retention. Multiplying by ν accounts for much of this error. Native search leaves the before/after variances similar, and Δ^2 is already close to measured retention (Appendix E.4).

5.5 Calibration size and evaluation cost

We compare the cost and accuracy of predicting η with those of measuring it directly. With the same 128 unwatermarked images, RenderGap computes Δ and ν, whereas direct measurement also requires 128 watermarked images to compute the mean scores (Table 4(a)). The two procedures have

Table 4: **Retention error and image-generation cost.** Errors are dimensionless median absolute relative errors against reference η measured with independent $4{,}000$ unwatermarked and 512 watermarked images. Direct measurement also estimates η. Panel (a) uses the same 128 unwatermarked images for every procedure over 200 fitting subsamples, with an additional 128 watermarked images for direct measurement. Panel (b) uses the full cached pool. GPU times are projected for RTX 4090s. Settings cover ten LlamaGen-L JPEG levels and three RAR-XL tokenizers under `f32`, `uint8`, and `jpeg75`.

	New images		LlamaGen-L		RAR-XL	
Procedure	Unwatermarked	Watermarked	Median error (fraction) ↓	GPU time (s) ↓	Median error (fraction) ↓	GPU time (s) ↓
(a) Matched unwatermarked pool: 128 images						
Direct measurement	128	128	0.0995	203.84	0.0357	170.69
RenderGap $\Delta^2\nu$	128	0	0.0980	101.67	0.0288	85.22
Gap-only Δ^2	128	0	0.0504	101.67	0.0084	85.22
(b) Reusing 1,000 unwatermarked token pairs: additional cost						
Direct measurement	0	128	0.0423	102.17	0.0183	85.47
RenderGap $\Delta^2\nu$	0	0	0.0460	0.00	0.0090	0.00
Gap-only Δ^2	0	0	0.0436	0.00	0.0114	0.00

GPU time includes sampling, decoding, and re-encoding, excluding loading, CPU scoring, I/O, and reference construction. In (b), both token sequences are cached for every evaluated setting, and zero denotes no additional GPU work.

similar median errors in these comparisons, while RenderGap avoids the GPU work of generating and processing the additional watermarked images. The gap-only predictor is more accurate at this sample size because estimating ν adds noise.

For a fixed AR token generator and input distribution, cached unwatermarked token pairs let both predictors evaluate new keys using CPU scoring alone. Direct measurement requires new watermarked images. All three methods reuse generated tokens across image-processing levels and compatible tokenizers. A new image-processing or tokenizer setting requires its own round trip. Larger pools reduce sampling error, while crop errors persist (Figure 7 in Appendix C.5). The same appendix details the allocations and timings.

Other strengths and keys. The bias sweeps support reusing a Global prediction across the tested strengths, while Cluster-64 stability depends on the setting (Appendix C.3). Independent calibration leaves ranking among the ten shared random keys unresolved on Janus-Pro-1B and LlamaGen-L (Table 18 in Appendix C.6). Appendices E.5 and E.1 extend the evaluation to previous-token green lists and adapted watermarking methods.

6 Conclusion and Limitations

We introduced RenderGap to predict watermark retention from unwatermarked generations by measuring green-label transitions and changes in unwatermarked green-count variance across the render channel. The retention metric η compares standardized watermark strength before and after rendering, and our analysis relates it to detection at fixed generation and false-positive rate under the conditions of Proposition 1. Experiments on three AR image models show close agreement with independently measured retention for the tested JPEG and blur sweeps and tokenizer comparisons without cropping. These results support using RenderGap to compare compatible tokenizers before generating watermarked images, while reusing cached token pairs to evaluate additional keys.

Limitations. The small-bias analysis fixes token contexts and models token transitions locally. Extensions could account for autoregressive dependencies and, given the crop-and-resize errors, spatial correspondence. Small calibration sets can make variance estimates noisy, while independent tests do not establish reliable rankings among the tested keys. More stable variance estimation and explicit uncertainty estimates could make comparisons between similar retention values more reliable.

AI USE STATEMENT

Generative AI tools were used to assist with implementing experimental code and polishing the manuscript text for clarity and readability. All AI-assisted code and text were reviewed and verified by the authors. The authors take full responsibility for the final content and results of this work.

ETHICS STATEMENT

RenderGap lets defenders assess watermark retention for a given AR token generator, tokenizer, and keyed watermark before deployment. An adversary could use our reported processing results to guide attempts to weaken watermark detection. We judge disclosure to benefit defenders because the predictions help compare retention across models and image-processing conditions. We use publicly released weights for Janus-Pro-1B, LlamaGen-L, RAR-XL, and WMAR's finetuned tokenizers, together with public image data. The study involves no human subjects or personally identifying data.

REPRODUCIBILITY STATEMENT

The supplementary archive provides the core RenderGap prediction and retention-measurement code, together with input specifications and usage instructions. Sections 3–4 define the quantities and estimators, and Appendix A provides the proofs. Section 5 and Appendix B specify the models, image processing, sampling, calibration, and computational setup. Appendices C–E report the evaluation protocols and additional results.

REFERENCES

Bang An, Mucong Ding, Tahseen Rabbani, Aakriti Agrawal, Yuancheng Xu, Chenghao Deng, Sicheng Zhu, Abdirisak Mohamed, Yuxin Wen, Tom Goldstein, and Furong Huang. WAVES: Benchmarking the robustness of image watermarks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 1456–1492. PMLR, 2024.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11315–11325, 2022.

Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-Pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2nd edition, 2006. ISBN 9780471241959. doi: 10.1002/047174882X.

B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. doi: 10.1214/aos/1176344552.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12873–12883, 2021.

Danilo Francati, Yevin Nikhel Goonatilake, Shubham Vivek Pawar, Daniele Venturi, and Giuseppe Ateniese. The coding limits of robust watermarking for generative models. In *2026 IEEE 11th European Symposium on Security and Privacy (EuroS&P)*, pp. 1343–1356. IEEE, 2026.

Peter Hall and Susan R. Wilson. Two guidelines for bootstrap hypothesis testing. *Biometrics*, 47(2): 757–762, 1991. doi: 10.2307/2532163.

Yuepeng Hu, Zhengyuan Jiang, Moyang Guo, and Neil Gong. A transfer attack to image watermarks. In *International Conference on Learning Representations*, volume 2025, pp. 18101–18127, 2025.

Siqi Hui, Yiren Song, Sanping Zhou, Ye Deng, Wenli Huang, and Jinjun Wang. Autoregressive images watermarking through lexical biasing: An approach resistant to regeneration attack. *arXiv preprint arXiv:2506.01011*, 2025.

Nikola Jovanović, Ismail Labiad, Tomáš Souček, Martin Vechev, and Pierre Fernandez. Watermarking autoregressive image generation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 71801–71848. Curran Associates, Inc., 2025. doi: 10.52202/085713-2411.

Louis Kerner, Michel Meintz, Bihe Zhao, Franziska Boenisch, and Adam Dziedzic. BitMark: Watermarking bitwise autoregressive image generative models. *Advances in Neural Information Processing Systems*, 38:99606–99640, 2025.

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

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 *The Twelfth International Conference on Learning Representations*, 2024.

Xiang Li, Feng Ruan, Huiyuan Wang, Qi Long, and Weijie J. Su. A statistical framework of watermarks for large language models: Pivot, detection efficiency and optimal rules. *The Annals of Statistics*, 53(1):322–351, 2025a. doi: 10.1214/24-AOS2468.

Xingchi Li, Guanxun Li, and Xianyang Zhang. A likelihood based approach for watermark detection. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 1675–1683. PMLR, 2025b.

Shilin Lu, Zihan Zhou, Jiayou Lu, Yuanzhi Zhu, and Adams Kong. Robust watermarking using generative priors against image editing: From benchmarking to advances. In *The Thirteenth International Conference on Learning Representations*, 2025.

Denis Lukovnikov, Andreas Müller, Erwin Quiring, and Asja Fischer. ClusterMark: Towards robust watermarking for autoregressive image generators with visual token clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9213–9222, 2026.

Michel Meintz, Jan Dubiński, Franziska Boenisch, and Adam Dziedzic. Radioactive watermarks in diffusion and autoregressive image generative models. *arXiv preprint arXiv:2506.23731*, 2025.

Andreas Müller, Denis Lukovnikov, Shingo Kodama, Minh Pham, Anubhav Jain, Jonathan Petit, Niv Cohen, and Asja Fischer. On the robustness of watermarking for autoregressive image generation. *arXiv preprint arXiv:2604.11720*, 2026.

Sylvestre-Alvise Rebuffi, Tuan Tran, Valeriu Lacatusu, Pierre Fernandez, Tomáš Souček, Nikola Jovanović, Tom Sander, Hady Elsahar, and Alexandre Mourachko. Learning to watermark in the latent space of generative models. *arXiv preprint arXiv:2601.16140*, 2026.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Mehrdad Saberi, Vinu Sankar Sadasivan, Keivan Rezaei, Aounon Kumar, Atoosa Chegini, Wenxiao Wang, and Soheil Feizi. Robustness of AI-image detectors: Fundamental limits and practical attacks. In *International Conference on Learning Representations*, volume 2024, pp. 27500–27526, 2024.

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.

Yu Tong, Zihao Pan, Shuai Yang, and Kaiyang Zhou. Training-free watermarking for autoregressive image generation. *arXiv preprint arXiv:2505.14673*, 2025.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Yihan Wu, Xuehao Cui, Ruibo Chen, Georgios Milis, and Heng Huang. A watermark for auto-regressive image generation models. *arXiv preprint arXiv:2506.11371*, 2025.

Yigit Yilmaz, Elena Petrova, Mehmet Kaya, Lucia Rossi, and Amir Rahman. UniMark: Unified adaptive multi-bit watermarking for autoregressive image generators. *arXiv preprint arXiv:2604.11843*, 2026.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. Advancing beyond identification: Multi-bit watermark for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4031–4055, 2024.

Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation. *arXiv preprint arXiv:2411.00776*, 2024.

Bihe Zhao, Louis Kerner, Michel Meintz, Tameem Bakr, Franziska Boenisch, and Adam Dziedzic. Data provenance for image auto-regressive generation. In *The Fourteenth International Conference on Learning Representations*, 2026.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*, 2024.

APPENDIX CONTENTS

	Notation and terminology	13
A	Proofs of the retention results	14
B	Experimental setup and estimation	15
C	Validation of the retention metric and prediction	18
D	Effects of image processing	29
E	Watermark constructions and model variations	31

NOTATION AND TERMINOLOGY

In experiments, the score and calibration symbols denote sample estimates, and η denotes measured retention. A hat marks the after-render stage, and superscripts (un) and (wm) identify unwatermarked and watermarked generation. Both measured η and its prediction $\Delta^2\nu$ have sampling uncertainty.

Symbol or term	**Meaning**
$x_t, \hat{x}_t$	Generated and re-read tokens at position t; $x_{1:T}$ and $\hat{x}_{1:T}$ are the paired sequences for one image.
$c, T, \mathcal{V}$	Prompt or class label, tokens per image, and token vocabulary; $\lvert\mathcal{V}\rvert$ is its size.
p_t, q_t	Next-token distributions before and after applying watermark bias.
$\mathbb{E}_{c,x}$	Expectation over conditioning inputs c and watermarked token sequences x generated from q_t.
$G_t, w_t(v)$	Green list at position t and membership indicator $w_t(v) = \mathbb{1}[v \in G_t]$ for token v.
γ, π_t	Fraction of vocabulary entries that are green / probability of sampling a green token at position t without watermark bias.
δ, ε	Bias added to green-token logits and average per-token KL budget in nats; εT is the total budget.
$S_i^{(\mathrm{un})}, S_i^{(\mathrm{wm})}$	Green counts before rendering for unwatermarked and watermarked image i, respectively.
$\mu^{(\mathrm{un})}, \mu^{(\mathrm{wm})}$	Mean green counts before rendering for unwatermarked and watermarked generation.
$\sigma^{(\mathrm{un})}, \hat{\sigma}^{(\mathrm{un})}$	Unwatermarked green-count SDs before and after rendering.
$z_i^{(\mathrm{un})}, z_i^{(\mathrm{wm})}$	Per-image green counts before rendering, standardized against the unwatermarked mean and SD.
$z, \hat{z}$	Mean standardized scores over watermarked images before and after the complete render channel.
η	Retention metric $(\hat{z}/z)^2$, defined for $z \neq 0$; direct measurement uses watermarked images and unwatermarked calibration.
$\Delta^2\nu$	RenderGap prediction $\Delta^2\nu$, estimated from unwatermarked token pairs under the watermark key.
$\bar{g}_G, \bar{g}_R$	Probabilities that a re-read token is green given that its generated token was green or red, respectively.
Δ	Pooled green-preservation gap $\bar{g}_G - \bar{g}_R$.
d_t, Δ_*	Green-preservation gap at position t and its weighted average with weights $\pi_t(1-\pi_t)$; Δ_* gives the small-bias ratio of mean shifts after and before rendering.
ν	Variance factor $(\sigma^{(\mathrm{un})}/\hat{\sigma}^{(\mathrm{un})})^2$.
Nat	Information unit for natural logarithms; one nat equals $1/\ln 2 \approx 1.4427$ bits.
Render channel	Decoding generated tokens into an image, applying optional image processing, and re-encoding to re-read tokens.
Empirical null	Unwatermarked generations used to estimate green-count statistics and calibrate detection under a given key.
`f32`, `uint8`	Re-encode floating-point decoder output directly / after 8-bit image conversion; neither applies an attack.
Exact-token agreement p_{same} / alignment	Fraction of tokens recovered unchanged at the same positions in unwatermarked round trips / preservation of green or red labels when tokens change.
TPR / FPR	Fractions of watermarked / unwatermarked images flagged by the detector at a given threshold.

A PROOFS OF THE RETENTION RESULTS

A.1 PROOFS OF THE MAIN RESULTS

Proposition 1. Eq. 7 with $z > 0$ and $\hat{z} \geq 0$ gives $\hat{z} = \sqrt{\eta}\, z$. Under the Gaussian assumption, the null-standardized post-render score of a watermarked image is normal with mean $\hat{z}$ and variance 1, and the null score is standard normal. At false-positive rate λ the threshold is $\Phi^{-1}(1-\lambda)$, so substituting $\hat{z} = \sqrt{\eta}\, z$ gives Eq. 8. The true-positive rate increases with $\hat{z}$ and hence with η at fixed z. For n independent, identically distributed unwatermarked images, the aggregate statistic $n^{-1/2} \sum_{i=1}^{n} \hat{z}_i^{(\mathrm{un})}$ is standard normal. For n independent, identically distributed watermarked images, $n^{-1/2} \sum_{i=1}^{n} \hat{z}_i^{(\mathrm{wm})}$ has mean $\sqrt{n}\,\hat{z} = \sqrt{n\eta}\, z$ and variance 1. For $\eta > 0$ and target true-positive rate $\tau \in (\lambda, 1)$, solving the Gaussian power inequality gives the minimum integer sample count

$$n_{\min} = \left\lceil \frac{\left(\Phi^{-1}(1-\lambda) + \Phi^{-1}(\tau)\right)^2}{\eta\, z^2} \right\rceil .$$

The numerator before squaring is positive because $\tau > \lambda$. Thus the required number of images for this joint decision scales as $1/\eta$ before integer rounding. □

Proposition 2. Hold the distributions p_t, green lists G_t, and channel M fixed. Let $m_t(v) = \mathbb{E}[w_t(\hat{x}_t) \mid x_t = v]$ be the probability that input token v returns green at position t. This expectation uses row v of the transition matrix M, which gives the distribution of re-read tokens for input token v. The probability of sampling a green token without watermark bias is $\pi_t = \mathbb{E}_{x_t \sim p_t}[w_t(x_t)]$, whereas $\gamma = |G_t|/|\mathcal{V}|$ counts the fraction of codebook entries in the green list. Since $0 < \pi_t < 1$, the probabilities of returning green conditional on a generated green or red token are $\bar{g}_{G,t} = \mathbb{E}_{x_t \sim p_t}[m_t(x_t) \mid x_t \in G_t]$ and $\bar{g}_{R,t} = \mathbb{E}_{x_t \sim p_t}[m_t(x_t) \mid x_t \notin G_t]$, respectively, with gap $d_t = \bar{g}_{G,t} - \bar{g}_{R,t}$.

Adding the green-token logit bias gives the normalized distribution below. Its derivative with respect to δ, evaluated at $\delta = 0$, measures how the token probability changes as the bias increases from zero.

$$q_t(v;\delta) = \frac{p_t(v) e^{\delta w_t(v)}}{1 - \pi_t + \pi_t e^{\delta}}, \qquad \left.\frac{\partial q_t(v;\delta)}{\partial \delta}\right|_{\delta=0} = p_t(v)\big(w_t(v) - \pi_t\big).$$

Summing this derivative against any fixed function f on the finite codebook gives

$$\begin{aligned} &\left.\frac{\partial}{\partial \delta} \mathbb{E}_{x_t \sim q_t(\cdot;\delta)}[f(x_t)]\right|_{\delta=0} \\ &\quad = \mathbb{E}_{x_t \sim p_t}[f(x_t) w_t(x_t)] - \mathbb{E}_{x_t \sim p_t}[f(x_t)]\pi_t \\ &\quad = \mathrm{Cov}_{p_t}(f, w_t). \end{aligned}$$

Here Cov_{p_t} is covariance under the unwatermarked token distribution, so the initial change in the expected value of f depends on its association with green-list membership. For the re-read green indicator, take $f = m_t$ and separate the green and red cases:

$$\mathrm{Cov}_{p_t}(m_t, w_t) = \pi_t \bar{g}_{G,t} - \pi_t\big[\pi_t \bar{g}_{G,t} + (1-\pi_t)\bar{g}_{R,t}\big] = \pi_t(1-\pi_t) d_t.$$

For the generated-token green indicator, taking $f = w_t$ instead gives the derivative $\pi_t(1-\pi_t)$. These are first-order coefficients, so the finite mean shifts satisfy

$$\begin{aligned} \hat{\mu}^{(\mathrm{wm})} - \hat{\mu}^{(\mathrm{un})} &= \delta \sum_t \pi_t(1-\pi_t) d_t + O(\delta^2), \\ \mu^{(\mathrm{wm})} - \mu^{(\mathrm{un})} &= \delta \sum_t \pi_t(1-\pi_t) + O(\delta^2). \end{aligned}$$

The $O(\delta^2)$ remainders are bounded in magnitude by a constant times δ^2 near zero with T, $|\mathcal{V}|$, p_t, G_t, and M fixed, since the finite sums above are analytic in δ. The token-side coefficient is positive. Dividing the two expansions and cancelling δ therefore gives

$$\frac{\hat{\mu}^{(\mathrm{wm})} - \hat{\mu}^{(\mathrm{un})}}{\mu^{(\mathrm{wm})} - \mu^{(\mathrm{un})}} \longrightarrow \frac{\sum_t \pi_t(1-\pi_t) d_t}{\sum_t \pi_t(1-\pi_t)} = \Delta_*.$$

Pooling uniformly over token positions weights green and red conditionals differently:

$$\Delta = \frac{\sum_t \pi_t \bar{g}_{G,t}}{\sum_t \pi_t} - \frac{\sum_t (1-\pi_t)\bar{g}_{R,t}}{\sum_t (1-\pi_t)}.$$

When π_t is constant across positions, both this expression and Δ_* reduce to $T^{-1}\sum_t d_t$. In general they differ, even if d_t is constant, because the two pooled conditional means use different weights. □

Substituting this limit into Eq. 10 proves $\eta \to \Delta_*^2 \nu$, since the null variances are fixed as δ varies. For $\Delta_* \neq 0$, $\eta/\Delta_*^2 \to \nu$, so the gap-only approximation has the correct relative limit exactly when $\nu = 1$. At $\delta = 0$, both population mean scores vanish and their ratio is undefined. The proposition concerns the limit from positive bias.

Fixed context and AR sampling. The calculation above holds one context fixed at each position. During AR sampling, a logit bias changes the distribution of earlier tokens and therefore the contexts seen by later sampling steps. The conditional model distribution $p_\theta(\cdot \mid x_{<t}, c)$ remains fixed for a specified prefix, but averaging over the changing prefix distribution introduces additional cross-position contributions to the mean shift. Proposition 2 does not bound these contributions, so we assess their effect through the empirical prediction residual.

The empirical predictor pools transitions across positions, so its gap Δ may differ from the weighted Δ_* even before accounting for changes in AR prefixes. When $\Delta \neq 0$ and the before-render mean difference is nonzero, the residual

$$\left(\frac{\hat{\mu}^{(\mathrm{wm})} - \hat{\mu}^{(\mathrm{un})}}{\Delta \left(\mu^{(\mathrm{wm})} - \mu^{(\mathrm{un})} \right)} \right)^2$$

compares the exact squared mean-count ratio with Δ^2. It equals 1 when the pooled approximation is exact and also reflects finite bias and interactions among positions.

B Experimental setup and estimation

B.1 Models and sampling

Janus-Pro-1B uses the official `deepseek-ai/Janus-Pro-1B` VQ weights with its normalized nearest-code search. It emits a 384×384 image as T=576 tokens from a codebook of $|\mathcal{V}|$=16,384 entries. LlamaGen-L is the class-conditional `c2i_L_256` checkpoint with the `vq_ds16_c2i` tokenizer. It emits a 256×256 image as T=256 tokens from a codebook of $|\mathcal{V}|$=16,384 entries. RAR-XL generates T=256 tokens from a MaskGIT-VQGAN codebook of $|\mathcal{V}|$=1,024 entries (Chang et al., 2022). Appendix E.3 describes its tokenizer variants and generation protocol. The native Janus VQ search normalizes latent and codebook vectors before computing squared nearest-code distances in BF16, whereas LlamaGen-L's VQ path uses normalized FP32 search and RAR-XL's uses raw Euclidean FP32 search. BF16 denotes bfloat16, while FP16 and FP32 denote 16-bit and 32-bit IEEE floating-point arithmetic, respectively. These arithmetic choices describe token recovery, whereas `f32` and `uint8` name image-processing paths (Section 5).

Janus-Pro images are conditioned on text prompts of the form "A photo of {subject}." over a fixed list of 60 subjects, whereas LlamaGen-L images are conditioned on a fixed list of 60 ImageNet class labels. Every generation run cycles through its list in order. Our sampling loop draws each token from the full conditional distribution at temperature 1, with no top-k or nucleus truncation and no classifier-free guidance.

B.2 Payload allocation and embedding budgets

For multi-bit payloads we adapt the position-allocation idea of Yoo et al. (2024) by cycling through the payload bits across token positions and setting the sign of δ from the assigned bit. For randomly sampled payloads the budget also averages over messages. To decode a bit, we compute the green-token fraction over the positions assigned to it. We predict 1 when this fraction exceeds its mean on unwatermarked reference images at the same stage, and 0 otherwise. Bit accuracy averages agreement with the embedded bits over all bit positions and images.

We compute the per-token KL divergence of Eq. 2 exactly by summing over all $|\mathcal{V}|$ tokens at each step. Natural logarithms give KL in nats, with one nat equal to $1/\ln 2 \approx 1.4427$ bits (Cover & Thomas, 2006). Using bits would rescale every KL budget by this constant without changing the distribution or the selected watermark. For each target KL budget we select the embedding bias with the closest recorded average KL in a calibration sweep. The nominal targets specify the experimental settings, while reported ε values are the recorded averages.

We select biases using nominal per-token KL targets $\{0.005, 0.01, 0.02, 0.05, 0.1\}$ and add a $\delta = 2$ anchor above the low embedding budget. The reported retention sweeps cover Global and Cluster-64, with the tested biases listed in Table 5. At each bias we generate watermarked images, apply each render channel, and compute detection scores.

B.3 Unwatermarked calibration and generation runs

For the supplementary bias sweeps, calibration uses $1{,}024$ unwatermarked images for Janus-Pro-1B and $5{,}000$ for LlamaGen-L and each RAR-XL tokenizer (Appendix E.3). These images follow the watermarked generation protocol with $\delta = 0$. The key and construction studies use the calibration pools listed in Table 5, while the controlled prediction comparisons use the separate pools described below. Every channel is applied to the null and to the watermarked images alike, so the detection threshold, Δ, and ν of a channel all come from the null pushed through that channel. The channel parameters are given in Section 5, and the blur radii in Appendix D.3.

Independent-reference prediction assessment. For each of the controlled Janus-Pro-1B and LlamaGen-L JPEG and blur comparisons, we use one Global key-42 unwatermarked pool of $5{,}000$ generated images, a 128-image watermarked fitting pool at $\delta = 1$, and a separate 512-image watermarked reference pool at the same bias. The same generated token sequences are evaluated at all ten JPEG settings and eight blur settings, including their common identity setting. For the controlled RAR-XL comparison, the three tokenizer variants process the same $5{,}000$ unwatermarked token rows, the same 128 watermarked fitting rows, and the same separate 512 watermarked reference rows at Global key 42 and $\delta = 0.4$. Each tokenizer and processing channel produces its own recovered tokens and unwatermarked calibration statistics. For all three models, the first $1{,}000$ unwatermarked rows fit Δ and ν, while the other $4{,}000$ and the separate 512 watermarked images measure the reference η. The 128 fitting watermarked images are used only for comparisons with direct estimation and never enter the reference. The reference is an independently calibrated finite-sample measurement. The Cluster-64 RAR secondary grid instead has 128 watermarked images per setting and is reported separately from the 512-image primary reference.

Table 5 summarizes the watermarked image sets used for retention and detection. Each of our own generation runs uses the constructions of Section 5 with a target green fraction of 0.5, and the existing constructions carry their own parameters (Appendix E.1). Seeds are fixed per generation run and recorded with the raw results. The random keys were drawn with seed 0, and each random-key generation run uses its key as the seed.

B.4 Estimation and uncertainty

For N_{un} unwatermarked and N_{wm} watermarked images, the sample means and SDs before the render channel are

$$\mu^{(\mathrm{un})} = \frac{1}{N_{\mathrm{un}}} \sum_i S_i^{(\mathrm{un})}, \qquad (\sigma^{(\mathrm{un})})^2 = \frac{1}{N_{\mathrm{un}} - 1} \sum_i (S_i^{(\mathrm{un})} - \mu^{(\mathrm{un})})^2,$$

$$\mu^{(\mathrm{wm})} = \frac{1}{N_{\mathrm{wm}}} \sum_i S_i^{(\mathrm{wm})}, \qquad z = \frac{\mu^{(\mathrm{wm})} - \mu^{(\mathrm{un})}}{\sigma^{(\mathrm{un})}}.$$

Applying the same formulas to the re-read counts $\hat{S}_i^{(\mathrm{un})}$ and $\hat{S}_i^{(\mathrm{wm})}$ gives $\hat{\mu}^{(\mathrm{un})}$, $\hat{\sigma}^{(\mathrm{un})}$, $\hat{\mu}^{(\mathrm{wm})}$, and $\hat{z}$. Hats mark the after-render stage throughout, and all empirical means and SDs are sample estimates. Each token sequence remains paired with its own re-read sequence, while watermarked images need not be paired with unwatermarked images. RenderGap estimates $\bar{g}_G$ by dividing the number of green-to-green token pairs by the number of generated green tokens, and estimates $\bar{g}_R$ analogously

Table 5: **Watermarked image sets for retention and detection experiments.** Rows specify the model, keys, logit bias δ, and number of watermarked images. Image counts are per model and construction, and per key or budget where marked. Rows marked both cover Janus-Pro-1B and LlamaGen-L. The null contains $1{,}024$ Janus-Pro-1B, $5{,}000$ LlamaGen-L, or $5{,}000$ RAR-XL images unless a null size is given. The controlled comparison rows distinguish fitting and reference pools. Processing levels and tokenizer variants reuse their generated token rows. The blur sweep with shared calibration reuses the controlled LlamaGen fitting images. The existing constructions carry their own keys and budgets, listed in Appendix E.1.

experiment	model	keys	δ	images
budget sweep, global and cluster-64 (Section 5)	Janus-Pro-1B	42	0.2, 0.3, 0.4, 0.7, 1.0, 2.0	128 per budget
budget sweep, global; cluster-64 has three biases	LlamaGen-L	42	0.2, 0.3, 0.4, 0.7, 1.0, 2.0; cluster: 0.4, 1.0, 2.0	128 per budget
calibration-size factor analysis (Appendix C.5)	Janus-Pro-1B	11 keys	2.0	128 per key
shared random keys (Section 5)	both	10 keys	2.0	512 per key
shared random keys, low embedding budget (Appendix C.6)	Janus-Pro-1B	10 keys	0.4	512 per key
control run (Appendix C.6)	both	42	2.0	128
controlled JPEG and blur sweeps (Section 5)	both	42	1.0	128 fit, 512 reference, null 5,000
blur sweep, shared calibration (Appendix D.3)	LlamaGen-L	42	1.0	128, null 1,000
budget sweep, global and cluster-64, three tokenizers (Appendix E.3)	RAR-XL	42	0.2, 0.3, 0.4, 0.7, 1.0, 2.0	128 per budget
fixed keys, three tokenizers (Appendix E.3)	RAR-XL	10 keys	2.0	128 per key
low embedding budget, three tokenizers (Appendix E.3)	RAR-XL	42	0.4	512
controlled tokenizer comparison (Section 5)	RAR-XL	42	0.4	128 fit, 512 reference, null 5,000
random keys, three tokenizers (Appendix E.3)	RAR-XL	10 keys	2.0	512 per key
existing constructions (Appendix E.1)	both	—	—	512 per budget

for red-to-green pairs. The predictor is undefined if either source color is absent or the required null SD is zero. Measured η additionally requires nonzero z.

We estimate uncertainty by resampling whole images with replacement (Efron, 1979). The pooled η in Table 15 is the arithmetic mean over the included embedding biases. Its uncertainty uses the joint bootstrap described in Appendix C.3, including the pairing of watermarked rows across biases when generation seeds are shared. LlamaGen-L cluster-64 uses $\delta \in \{0.4, 1.0, 2.0\}$, while the other entries use $\delta \in \{0.2, 0.3, 0.4, 0.7, 1.0, 2.0\}$. The original RAR-XL global `f32` result is repeated in panel (b) to compare the tokenizer variants with the generated tokens held fixed.

Subscript $\pm$ values in the model, construction, and channel tables are standard deviations from $4{,}000$ image-bootstrap resamples. In Tables 15 and 25, the prediction $\Delta^2\nu$ and the measured η carry a subscript, while Δ, ν, and TPR@1%FPR are point estimates. Table 18 reports point estimates and their spread across the shared random keys. Its disjoint-null tests use 50 partitions from 25 seeds and $20{,}000$ key-label permutations (Appendix C.6). The Janus-Pro-1B factor analysis in Appendix C.5 draws 50 subsamples per size with seed 0. TPR is reported at 1% FPR with the threshold set at the tail of the null (Section 3).

The controlled independent-reference analysis uses $2{,}000$ whole-image bootstrap draws, resampling the fit null, reference null, and reference watermarked pools separately while preserving each image's

Table 6: **Gaussian detection-model errors.** Mean absolute TPR errors and FPR ranges are reported as fractions, with FPR measured on disjoint unwatermarked images. The first formula uses the population Gaussian threshold, the second the empirical threshold, and the third also fits the watermarked SD.

Model	Settings	Gaussian ↓	Emp. threshold ↓	Unequal SD ↓	FPR range
Janus-Pro-1B	17	0.0652	0.0631	0.0142	0.0050–0.0130
LlamaGen-L	17	0.0409	0.0442	0.0139	0.0040–0.0140
RAR-XL	12	0.0164	0.0187	0.0188	0.0030–0.0140

pairing across processing levels or tokenizers. The matched-null cost comparison uses the same 128 unwatermarked images for all methods, with 128 additional watermarked images for direct measurement, over 200 paired subsets of the fitting null pool. For its equal-total-budget sample-size curves, 200 paired subsets are drawn from the available fitting pools at total budgets 32, 64, 128, and 256 images. Null-only methods use the entire budget for unwatermarked images, whereas the direct estimator receives half unwatermarked and half watermarked images. The same independent reference is used for every subset, so the curves measure sensitivity to subsampling within the available pools. Cost projections count each distinct AR generation pool once and allow both methods to reuse generated tokens and decoded images across compatible processing settings. Stage timings use RTX 4090 GPUs, batch size 8, and 256 tokens per image, with ten measured batches after two warmup batches per condition. We alternate unwatermarked and watermarked batches and synchronize CUDA around AR sampling, decoding, and re-encoding. Both models use PyTorch 2.6.0. LlamaGen-L uses BF16 for AR sampling and FP32 for its tokenizer, while RAR-XL uses FP32 for both. Both use eight CPU threads, with TensorFloat-32 (TF32) disabled for matrix multiplication and enabled for cuDNN convolutions. Watermarked sampling uses bias 1.0 for LlamaGen-L and 0.4 for RAR-XL, matching the evaluated settings, and includes exact KL calculation. The reported GPU-stage times exclude loading.

Computational resources. The random-key images and their control run were generated on NVIDIA V100 GPUs with 32 GB of memory. Native Janus VQ evaluation uses BF16 arithmetic on an NVIDIA RTX 4090. Statistical analysis runs on CPU with NumPy 2.4.6 for the joint bootstrap and torch 2.6.0 with NumPy 2.2.6 for the other analyses.

C Validation of the retention metric and prediction

C.1 Empirical check of the Gaussian detection relation

We compare Proposition 1's detection formula with observed TPR while fixing the generated-token distribution within each model. The controlled pools of Appendix B.3 provide 4,000 unwatermarked calibration images, 512 watermarked images, and 1,000 separate unwatermarked images on which to check FPR. The last pool also supplies RenderGap's prediction factors, but is disjoint from threshold calibration. The per-model settings are ten JPEG levels and eight blur levels for Janus-Pro-1B/LlamaGen-L, and three tokenizers under four processing paths for RAR-XL. JPEG quality 100 and zero blur are the same identity path and are counted once in summaries. The same generated rows are reused within each model, giving paired observations across settings. The RAR-XL tables label the original tokenizer as Original, the variant fine-tuned without image augmentations as RCC-ft, and the variant fine-tuned with augmentations as RCC-ft+aug.

We check the Gaussian detection model using the mean score and TPR measured on the same watermarked images. We compare three fitted formulas: Eq. 8; the same unit-variance formula with the empirical null threshold; and a Gaussian formula that also uses the measured watermarked SD. Here $\hat{\sigma}^{(\mathrm{wm})}$ is the SD of green counts across watermarked images after the render channel. Writing u for the empirical threshold in null-standardized units and $r = \hat{\sigma}^{(\mathrm{wm})}/\hat{\sigma}^{(\mathrm{un})}$ for the watermarked-to-null SD ratio, the last formula is $1 - \Phi((u - \hat{z})/r)$. The second formula sets $r = 1$.

Adjusting only the threshold changes the errors little, whereas fitting the watermarked SD substantially reduces the Janus-Pro-1B and LlamaGen-L discrepancies (Table 6). Unequal variances therefore explain much of the gap between observed detection and the unit-variance formula in

Table 7: **Gaussian detection check on Janus-Pro-1B.** Global key 42, with 4,000 calibration and 512 watermarked images. SD ratio is watermarked SD divided by null SD. TPR error is Gaussian TPR minus observed TPR. Subscripts give one bootstrap SD. FPR is measured on 1,000 different unwatermarked images.

Setting	η	TPR	Gaussian TPR	TPR error	FPR	SD ratio
JPEG 100	0.7382	1.0000	1.0000	$0.0000_{\pm 0.0000}$	0.0110	1.2613
JPEG 95	0.6271	0.9961	1.0000	$0.0039_{\pm 0.0027}$	0.0120	1.8152
JPEG 90	0.5641	0.9922	1.0000	$0.0078_{\pm 0.0041}$	0.0100	1.9419
JPEG 80	0.4593	0.9629	1.0000	$0.0371_{\pm 0.0084}$	0.0120	2.2775
JPEG 70	0.3856	0.9375	1.0000	$0.0625_{\pm 0.0115}$	0.0120	2.3674
JPEG 60	0.3321	0.9062	0.9999	$0.0936_{\pm 0.0134}$	0.0050	2.4480
JPEG 50	0.2845	0.8906	0.9993	$0.1087_{\pm 0.0140}$	0.0100	2.4810
JPEG 40	0.2218	0.8340	0.9947	$0.1607_{\pm 0.0177}$	0.0070	2.2922
JPEG 30	0.1573	0.7617	0.9628	$0.2011_{\pm 0.0161}$	0.0090	2.3255
JPEG 20	0.0721	0.5312	0.6760	$0.1447_{\pm 0.0280}$	0.0060	1.7918
Blur 0.4	0.7161	1.0000	1.0000	$0.0000_{\pm 0.0000}$	0.0130	1.2476
Blur 0.8	0.5779	1.0000	1.0000	$0.0000_{\pm 0.0000}$	0.0130	1.3282
Blur 1.2	0.3258	0.9805	0.9998	$0.0194_{\pm 0.0066}$	0.0090	1.5864
Blur 1.6	0.1669	0.8613	0.9718	$0.1105_{\pm 0.0175}$	0.0050	1.6212
Blur 2.0	0.0920	0.6758	0.7932	$0.1174_{\pm 0.0210}$	0.0120	1.4706
Blur 2.5	0.0348	0.3223	0.3473	$0.0250_{\pm 0.0254}$	0.0080	1.2701
Blur 3.0	0.0171	0.1816	0.1661	$-0.0155_{\pm 0.0180}$	0.0060	1.1194

Table 8: **Gaussian detection check on LlamaGen-L.** Global key 42, with 4,000 calibration and 512 watermarked images. SD ratio is watermarked SD divided by null SD. TPR error is Gaussian TPR minus observed TPR. Subscripts give one bootstrap SD. FPR is measured on 1,000 different unwatermarked images.

Setting	η	TPR	Gaussian TPR	TPR error	FPR	SD ratio
JPEG 100	0.7091	0.9980	0.9998	$0.0018_{\pm 0.0021}$	0.0060	1.1549
JPEG 95	0.6275	0.9746	0.9993	$0.0247_{\pm 0.0075}$	0.0070	1.3368
JPEG 90	0.5779	0.9668	0.9986	$0.0318_{\pm 0.0082}$	0.0100	1.3857
JPEG 80	0.4759	0.9434	0.9936	$0.0502_{\pm 0.0099}$	0.0040	1.4585
JPEG 70	0.4154	0.9062	0.9851	$0.0788_{\pm 0.0129}$	0.0060	1.5515
JPEG 60	0.3577	0.8887	0.9677	$0.0790_{\pm 0.0136}$	0.0050	1.5340
JPEG 50	0.3127	0.8398	0.9426	$0.1027_{\pm 0.0158}$	0.0070	1.5119
JPEG 40	0.2350	0.7559	0.8548	$0.0990_{\pm 0.0191}$	0.0040	1.5414
JPEG 30	0.1762	0.6953	0.7268	$0.0315_{\pm 0.0262}$	0.0070	1.5095
JPEG 20	0.0919	0.4551	0.4167	$-0.0384_{\pm 0.0244}$	0.0100	1.3889
Blur 0.4	0.6880	0.9961	0.9997	$0.0036_{\pm 0.0027}$	0.0050	1.1499
Blur 0.8	0.5187	0.9766	0.9965	$0.0200_{\pm 0.0070}$	0.0050	1.2853
Blur 1.2	0.2535	0.8125	0.8826	$0.0701_{\pm 0.0188}$	0.0140	1.4609
Blur 1.6	0.1108	0.5059	0.4988	$-0.0071_{\pm 0.0335}$	0.0110	1.4243
Blur 2.0	0.0526	0.2500	0.2342	$-0.0158_{\pm 0.0275}$	0.0110	1.3046
Blur 2.5	0.0187	0.1211	0.0852	$-0.0358_{\pm 0.0154}$	0.0140	1.2501
Blur 3.0	0.0069	0.0449	0.0402	$-0.0047_{\pm 0.0097}$	0.0100	1.1400

these settings. We estimate uncertainty with 2,000 whole-image bootstrap draws, independently resampling the calibration, watermarked, and FPR-test pools and preserving rows across settings. This resampling accounts for uncertainty in means, SDs, thresholds, and TPR, together with their shared-data dependence. Tables 7–9 report every unique setting, with signed errors defined as formula TPR minus observed TPR.

C.2 Fixed-generation tokenizer selection

We choose the RAR-XL tokenizer with the largest predicted retention for each processing path using the 1,000 unwatermarked fitting images. The three candidates use the same generated tokens and Global key-42 partition, so their generated-token score is identical. The independent reference uses

Table 9: **Gaussian detection check on RAR-XL.** Global key 42, with 4,000 calibration and 512 watermarked images. SD ratio is watermarked SD divided by null SD. TPR error is Gaussian TPR minus observed TPR. Subscripts give one bootstrap SD. FPR is measured on 1,000 different unwatermarked images.

Setting	η	TPR	Gaussian TPR	TPR error	FPR	SD ratio
Original/f32	0.4732	0.2402	0.2630	$0.0228_{\pm 0.0188}$	0.0060	1.0016
Original/uint8	0.4726	0.2520	0.2626	$0.0107_{\pm 0.0211}$	0.0060	0.9991
Original/jpeg75	0.3836	0.2109	0.2111	$0.0001_{\pm 0.0236}$	0.0090	1.0056
Original/crop09	0.0464	0.0254	0.0362	$0.0108_{\pm 0.0070}$	0.0140	0.9528
RCC-ft/f32	0.9833	0.5078	0.5450	$0.0372_{\pm 0.0253}$	0.0080	1.0076
RCC-ft/uint8	0.9826	0.5059	0.5447	$0.0388_{\pm 0.0272}$	0.0090	1.0101
RCC-ft/jpeg75	0.8343	0.4648	0.4684	$0.0035_{\pm 0.0260}$	0.0050	1.0372
RCC-ft/crop09	0.0694	0.0410	0.0466	$0.0056_{\pm 0.0119}$	0.0030	0.9674
RCC-ft+aug/f32	0.9741	0.5352	0.5405	$0.0053_{\pm 0.0249}$	0.0070	1.0005
RCC-ft+aug/uint8	0.9726	0.5312	0.5397	$0.0085_{\pm 0.0240}$	0.0070	1.0003
RCC-ft+aug/jpeg75	0.9334	0.4707	0.5201	$0.0494_{\pm 0.0257}$	0.0090	0.9862
RCC-ft+aug/crop09	0.0648	0.0410	0.0446	$0.0035_{\pm 0.0085}$	0.0040	0.9746

Table 10: **Paired differences between RAR-XL tokenizer candidates.** Entries are differences measured on the independent reference, with one paired whole-image bootstrap SD shown as a subscript. Original denotes the original tokenizer, while RCC-ft and RCC-ft+aug are fine-tuned without and with augmentations, respectively. Negative differences favor the right-hand candidate.

Processing	Tokenizer pair	η difference	TPR difference
f32	Original − RCC-ft	$-0.5101_{\pm 0.0182}$	$-0.2676_{\pm 0.0309}$
f32	Original − RCC-ft+aug	$-0.5009_{\pm 0.0185}$	$-0.2949_{\pm 0.0310}$
f32	RCC-ft − RCC-ft+aug	$0.0092_{\pm 0.0053}$	$-0.0273_{\pm 0.0227}$
uint8	Original − RCC-ft	$-0.5100_{\pm 0.0182}$	$-0.2539_{\pm 0.0337}$
uint8	Original − RCC-ft+aug	$-0.5000_{\pm 0.0184}$	$-0.2793_{\pm 0.0312}$
uint8	RCC-ft − RCC-ft+aug	$0.0100_{\pm 0.0056}$	$-0.0254_{\pm 0.0229}$
jpeg75	Original − RCC-ft	$-0.4507_{\pm 0.0228}$	$-0.2539_{\pm 0.0358}$
jpeg75	Original − RCC-ft+aug	$-0.5499_{\pm 0.0212}$	$-0.2598_{\pm 0.0354}$
jpeg75	RCC-ft − RCC-ft+aug	$-0.0992_{\pm 0.0182}$	$-0.0059_{\pm 0.0284}$
crop09	Original − RCC-ft	$-0.0230_{\pm 0.0097}$	$-0.0156_{\pm 0.0149}$
crop09	Original − RCC-ft+aug	$-0.0184_{\pm 0.0096}$	$-0.0156_{\pm 0.0121}$
crop09	RCC-ft − RCC-ft+aug	$0.0046_{\pm 0.0052}$	$0.0000_{\pm 0.0117}$

4,000 different unwatermarked and 512 watermarked images at $\delta = 0.4$. Under JPEG, prediction and reference both favor the tokenizer fine-tuned with augmentations. Its retention exceeds the variant trained without augmentations, but their TPR difference is unresolved (Table 10). Both fine-tuned variants exceed the original in retention and TPR under the three paths without crop. For f32 and uint8, paired resampling leaves the retention ordering of the two fine-tuned variants unresolved. Under crop, prediction underestimates retention for all candidates (Figure 6).

C.3 Retention across embedding budgets

We vary the sampling bias to assess whether a prediction computed once from unwatermarked images remains useful across watermark strengths. The sampling and budget-calibration procedures are given in Appendix B.2. The Global/f32 sweeps show little variation in retention as sampling bias changes (Figure 5). We summarize each sweep by the arithmetic mean of its per-bias estimates and test for variation around that mean (Tables 11 and 12).

Because every $\eta(\delta)$ is standardized against one shared null within a model and channel, we resample that null once per bootstrap draw and preserve generated/re-read image pairs. Corresponding Janus-Pro-1B and LlamaGen-L watermarked rows are resampled together across biases because they share generation seeds and conditioning order. RAR-XL uses independent bias samples. For J biases, let η_j be measured retention at the jth tested bias and $e_j = \eta_j - J^{-1}\sum_k \eta_k$ its deviation from the mean across biases. Let s_j be the joint-bootstrap standard deviation of this deviation. We test equality of

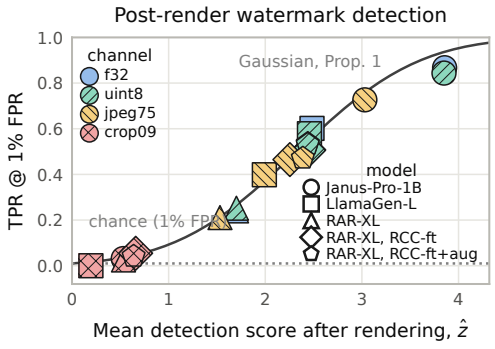


Figure 4: **Detection and the mean post-render score.** Global watermarks at the nominal low budget, across models and channels. RCC-ft and RCC-ft+aug denote RAR-XL tokenizers fine-tuned without and with augmentations, respectively. The curve is Proposition 1's Gaussian prediction with measured $\hat{z} = \sqrt{\eta}\, z$.

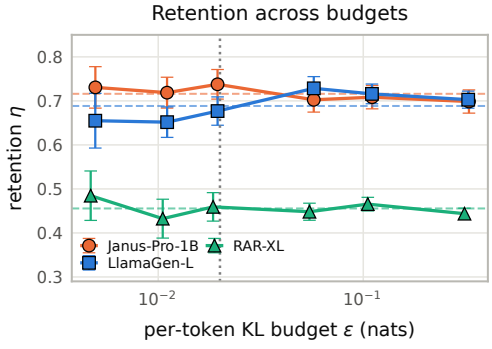


Figure 5: **Retention across sampling budgets.** Global construction with `f32`. Error bars show one joint-bootstrap SD and horizontal dashed lines show pooled means. The vertical line marks the nominal low budget of 0.02 nats per token.

Table 11: **Retention ranges and joint tests across biases.** Minimum and maximum η are taken over the tested sampling biases. The p-values use the joint bootstrap test across biases. All rows use key 42, with 1,024 null images for Janus-Pro-1B and 5,000 for LlamaGen-L.

Construction	Channel	No. of biases	min η	max η	p
Global	`f32`	6	0.6986	0.7377	0.5239
Global	`uint8`	6	0.6951	0.7353	0.6043
Global	`jpeg75`	6	0.3856	0.4557	0.1417
Global	`crop09`	6	0.0073	0.0177	0.7166
Cluster-64	`f32`	6	0.7738	0.8115	0.8658
Cluster-64	`uint8`	6	0.7458	0.8009	0.4741
Cluster-64	`jpeg75`	6	0.4388	0.5101	0.3377
Cluster-64	`crop09`	6	0.0760	0.1330	0.1052
LlamaGen-L, for comparison					
Global	`f32`	6	0.6517	0.7285	0.1642
Cluster-64	`f32`	3	0.7796	0.8329	0.0130

retention using the largest standardized deviation U_{obs} and its null-centered bootstrap counterpart $U^{(b)}$:

$$U_{\text{obs}} = \max_j |e_j|/s_j, \qquad U^{(b)} = \max_j |e_j^{(b)} - e_j|/s_j, \qquad p = \frac{1 + \sum_{b=1}^{B_{\text{boot}}} \mathbb{1}[U^{(b)} \geq U_{\text{obs}}]}{B_{\text{boot}} + 1}.$$

The superscript (b) denotes bootstrap draw b, and B_{boot} is the number of draws. The centering $e_j^{(b)} - e_j$ approximates deviations under equal retention (Hall & Wilson, 1991), and the approximate p-value compares U_{obs} with $U^{(b)}$. With $B_{\text{boot}} = 4{,}000$ joint draws, the tests detect variation for LlamaGen-L Cluster-64 ($p = 0.0130$), but not for the Global/`f32` sweeps on Janus-Pro-1B or LlamaGen-L (Table 11). Thus stability across biases depends on the construction. The Global results are consistent with a common retention value at the tested biases and sample sizes.

C.4 Variance calibration

Figure 6 shows the per-setting prediction-to-measurement ratios underlying Table 2.

To assess what squared score scaling and variance calibration add, we compare the three main predictors with unsquared exact-token agreement and the constant prediction 1 (Table 13). The first four predictors use unwatermarked token pairs, whereas the constant requires no fitting images.

Table 12: **Janus-Pro-1B retention with native VQ search.** Each pooled η is the mean of six per-bias values ($\delta = 0.2, 0.3, 0.4, 0.7, 1, 2$), with uncertainty given by the joint image-bootstrap SD. All channels use the same generated token rows and their own re-read null calibration. `f32` denotes the image path. Higher η means greater retention, and ratios closer to 1 indicate more accurate predictions.

Construction	Channel	η	Δ	ν	$\Delta^2\nu/\eta$
Global	`f32`	$0.7161_{\pm 0.0272}$	0.8452	0.9769	0.97
Global	`uint8`	$0.7114_{\pm 0.0276}$	0.8379	0.9929	0.98
Global	`jpeg75`	$0.4167_{\pm 0.0301}$	0.6294	0.9704	0.92
Global	`crop09`	$0.0112_{\pm 0.0027}$	0.0140	1.0270	0.02
Cluster-64	`f32`	$0.7988_{\pm 0.0176}$	0.9041	0.9815	1.00
Cluster-64	`uint8`	$0.7879_{\pm 0.0202}$	0.8992	0.9868	1.01
Cluster-64	`jpeg75`	$0.4863_{\pm 0.0214}$	0.7454	0.9326	1.07
Cluster-64	`crop09`	$0.1086_{\pm 0.0166}$	0.0714	1.0080	0.05

Table 13: **Complete comparison of retention predictors.** Median and mean absolute relative errors use the independent reference protocol of Table 2. Bold marks the lowest displayed error in each row.

(a) Median relative error ↓

Setting	$\Delta^2\nu$	Δ^2	p^2_{same}	p_{same}	1
Image-processing strengths: original tokenizers					
Janus-Pro JPEG	**0.0285**	0.0593	0.0620	0.6141	1.8022
Janus-Pro blur	**0.0375**	0.0400	0.0468	1.0500	3.5305
LlamaGen JPEG	0.0460	**0.0436**	0.0486	0.5529	1.6015
LlamaGen blur	0.0734	**0.0394**	0.0518	1.4529	5.4854
Tokenizer variants: RAR-XL					
`f32`, `uint8`, `jpeg75`	**0.0090**	0.0114	0.0209	0.0207	0.0713
`crop09`	**0.5531**	0.5926	0.5735	1.5951	14.4341

(b) Mean relative error ↓

Setting	$\Delta^2\nu$	Δ^2	p^2_{same}	p_{same}	1
Image-processing strengths: original tokenizers					
Janus-Pro JPEG	**0.0332**	0.0633	0.0635	0.8636	3.0756
Janus-Pro blur	**0.0402**	0.0411	0.0477	1.9429	12.9377
LlamaGen JPEG	**0.0476**	0.0490	0.0512	0.7503	2.6050
LlamaGen blur	0.0717	**0.0580**	0.0629	3.2213	28.5071
Tokenizer variants: RAR-XL					
`f32`, `uint8`, `jpeg75`	0.0247	**0.0192**	0.0233	0.1732	0.4662
`crop09`	0.5817	0.5958	**0.5797**	1.6685	16.1403

Across the JPEG and blur sweeps, unsquared agreement and the constant have much larger errors than the three squared predictors. Recovering a fraction of token identities therefore does not translate directly into retaining the same fraction of the squared mean score.

The exact factorization in Eq. 10 retains ν because the null standard deviation may change through the render channel. For native Janus-Pro-1B Global/`f32`, the before/after variances are similar, so Δ^2 already approximates retention closely (Table 12). Estimating ν then adjusts a prediction that needs little correction, whereas the unnormalized-search control exhibits a substantial variance change (Table 27). The contribution of ν is to account for that change, but its practical benefit also depends on estimation uncertainty. Table 14 gives the paired error differences against independent references for the controlled Janus-Pro-1B, LlamaGen-L, and RAR-XL settings. Paired resampling leaves the differences in mean error statistically unresolved in every comparison. Applying a binomial variance at both stages would force $\nu = 1$, whereas our estimator measures each stage's null standard deviation from unwatermarked images.

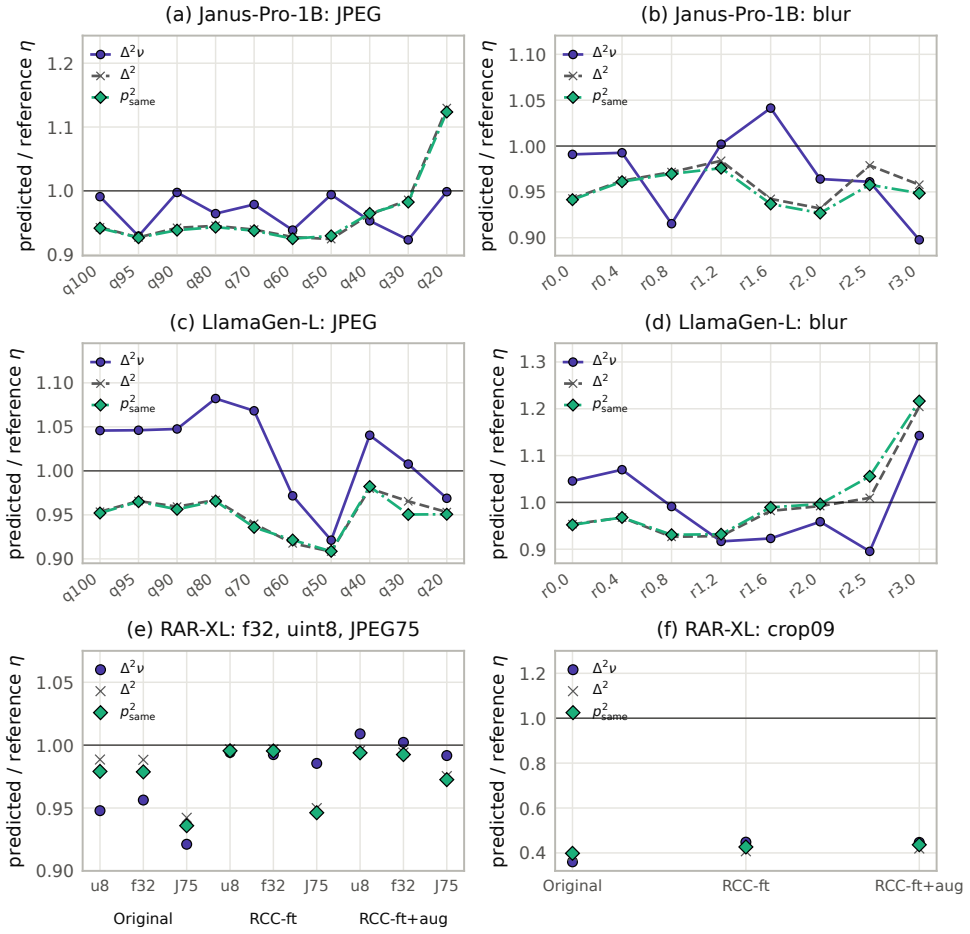


Figure 6: **Retention predictions across controlled processing and tokenizer changes.** Each point is the prediction divided by an independently calibrated reference η, and the horizontal line indicates agreement. JPEG and blur connect the same ordered processing levels for Janus-Pro-1B and LlamaGen-L, while RAR settings are separate tokenizer/channel categories. In the RAR panels, Original denotes the original tokenizer, while RCC-ft and RCC-ft+aug are fine-tuned without and with augmentations, respectively. The labels u8 and J75 denote 8-bit conversion and JPEG quality 75. JPEG ticks give quality q, with q100 denoting the unchanged 8-bit control without JPEG re-saving. Blur ticks give radius r in pixels.

C.5 Calibration sample size

To measure sensitivity to calibration size in the controlled processing comparisons, we use total fitting budgets of $B = 32, 64, 128, 256$ images and 200 paired subsamples of each fitting pool. Null-only predictors use B unwatermarked images, whereas direct estimation uses $B/2$ unwatermarked and $B/2$ watermarked images. The watermarked fitting pool is separate from the 512-image reference pool, and equal allocation is a specified comparator rather than an optimized allocation. Figure 7 reports the errors against the independent reference. With small fitting pools, the gap-only predictor is often more accurate than the full prediction, indicating that estimating the variance factor can cost more accuracy than it adds. Increasing the pool reduces sampling error, but the persistent crop error points to the approximation itself rather than insufficient calibration data.

To identify which factor contributes most to estimation uncertainty, we vary the calibration size for 11 selected Janus-Pro-1B keys under Global/`f32` (Table 16). We rank keys 1–40 by the unwatermarked variance factor, take eight evenly spaced ranks, add keys 7, 13, and 42, and remove duplicates. Selection uses the unnormalized Euclidean quantizer of Appendix E.4, and evaluation

Table 14: **Paired mean absolute relative-error differences against an independent reference.** Entries are comparator error minus $\Delta^2\nu$ error as a fraction, with one paired whole-image bootstrap SD shown as a subscript. Positive values favor $\Delta^2\nu$. Each model reuses generated rows across its processing settings.

Family	Δ^2 difference	p^2_{same} difference
Janus-Pro-1B JPEG	$+0.0300_{\pm 0.0278}$	$+0.0303_{\pm 0.0278}$
Janus-Pro-1B blur	$+0.0009_{\pm 0.0271}$	$+0.0075_{\pm 0.0283}$
LlamaGen-L JPEG	$+0.0014_{\pm 0.0462}$	$+0.0036_{\pm 0.0471}$
LlamaGen-L blur	$-0.0137_{\pm 0.0336}$	$-0.0088_{\pm 0.0366}$
RAR ordinary	$-0.0055_{\pm 0.0113}$	$-0.0013_{\pm 0.0110}$
RAR crop	$+0.0142_{\pm 0.0217}$	$-0.0020_{\pm 0.0233}$

Table 15: **Prediction across AR image models, constructions, and channels.** Janus-Pro-1B rows use native normalized BF16 VQ search. The prediction $\Delta^2\nu$ is estimated from unwatermarked token pairs. Measured η averages the per-bias estimates specified in Appendix B.4, and subscripts give one bootstrap SD. RCC-ft and RCC-ft+aug denote RAR-XL tokenizers fine-tuned without and with augmentations, respectively. Prediction-to-measurement ratios closer to 1 indicate more accurate predictions.

(a) Original tokenizers

Construction	Channel	Δ	ν	$\Delta^2\nu$	η	Δ^2/η	$\Delta^2\nu/\eta$
Janus-Pro-1B							
global	`f32`	0.8452	0.9769	$0.6979_{\pm 0.0237}$	$0.7161_{\pm 0.0272}$	1.00	0.97
global	`jpeg75`	0.6294	0.9704	$0.3844_{\pm 0.0188}$	$0.4167_{\pm 0.0301}$	0.95	0.92
cluster-64	`f32`	0.9041	0.9815	$0.8022_{\pm 0.0124}$	$0.7988_{\pm 0.0176}$	1.02	1.00
LlamaGen-L							
global	`f32`	0.8292	1.0494	$0.7215_{\pm 0.0125}$	$0.6884_{\pm 0.0225}$	1.00	1.05
global	`jpeg75`	0.6542	1.0619	$0.4544_{\pm 0.0148}$	$0.4296_{\pm 0.0257}$	1.00	1.06
cluster-64	`f32`	0.8999	0.9943	$0.8052_{\pm 0.0060}$	$0.8095_{\pm 0.0131}$	1.00	0.99
RAR-XL							
global	`f32`	0.6807	0.9297	$0.4307_{\pm 0.0070}$	$0.4556_{\pm 0.0159}$	1.02	0.95
global	`jpeg75`	0.6001	0.9641	$0.3472_{\pm 0.0069}$	$0.3573_{\pm 0.0170}$	1.01	0.97
cluster-64	`f32`	0.7635	0.9453	$0.5511_{\pm 0.0054}$	$0.6191_{\pm 0.0159}$	0.94	0.89

(b) RAR-XL tokenizer variants under `f32`

Tokenizer	Δ	ν	$\Delta^2\nu$	η	Δ^2/η	$\Delta^2\nu/\eta$
original tokenizer	0.6807	0.9297	$0.4307_{\pm 0.0070}$	$0.4556_{\pm 0.0159}$	1.02	0.95
RCC-ft	0.9899	0.9994	$0.9792_{\pm 0.0037}$	$0.9804_{\pm 0.0055}$	1.00	1.00
RCC-ft+aug	0.9839	1.0033	$0.9712_{\pm 0.0048}$	$0.9695_{\pm 0.0070}$	1.00	1.00

uses native normalized search with 128 watermarked images per key at $\delta = 2$. We split 1,024 native unwatermarked images into fit and score halves of 512 images each. The fit half supplies Δ and ν, while the score half supplies the null means and SDs for measuring η. We subsample whole images from the fit half, using 10 seeds in both directions and 50 subsets at each size. Relative deviations compare each estimate with the full fit half, while Spearman's rank correlation ρ_s compares prediction with retention measured using the disjoint score half. The variance factor accounts for most of the prediction's finite-pool sensitivity: its relative deviations closely follow those of $\Delta^2\nu$ and exceed those of Δ. At 256 images, the median deviation of $\Delta^2\nu$ is about 0.03, yet the disjoint-null rank correlation is only 0.173. Stable retention estimates at this scale therefore do not resolve the smaller differences between keys.

Measured timing and projected fitting cost. Table 17 expands the main cost comparison in Table 4, including the measured AR batch timings. Panel (a) uses the same 128 unwatermarked images for all methods, with 128 additional watermarked images for direct measurement of η. We draw 200 subsets from the 1,000-image fitting null pool, use each subset for every method and processing setting, and keep the separate 128 watermarked fitting images fixed. Panel (b) instead

Table 16: **Native Janus-Pro-1B calibration-size sensitivity.** Absolute relative deviation is the absolute difference between a subsample estimate and the full 512-image estimate, divided by the latter. Entries are median deviations as fractions, so 0.01 means 1%. Spearman's ρ_s compares predicted and measured retention across 11 fixed keys using separate images for measurement. Lower deviations and higher ρ_s indicate closer agreement.

Fit images m	Median absolute relative deviation (fraction) ↓			Key ρ_s ↑
	Δ	ν	$\Delta^2\nu$	
16	0.0163	0.1816	0.1839	0.0545
32	0.0106	0.1244	0.1255	0.0364
64	0.0073	0.0848	0.0856	0.0727
128	0.0048	0.0557	0.0574	0.0955
256	0.0029	0.0323	0.0328	0.1727
512	0.0000	0.0000	0.0000	0.2455

holds the total number of newly generated images at 256: RenderGap uses all 256 as unwatermarked, while direct measurement uses 128 unwatermarked and 128 watermarked images. The projected GPU costs are similar in this equal-total-budget comparison because watermark bias changes AR sampling time little. Panel (c) starts with 1,000 cached unwatermarked pairs. Direct measurement adds 128 watermarked images, whereas RenderGap and Δ^2 predict η from the cached pairs without further generation, decoding, or re-encoding. CPU scoring is excluded from these GPU-stage times. All errors use the same independent reference of 4,000 unwatermarked and 512 watermarked images. Both models are timed on RTX 4090 GPUs under the protocol in Appendix B.4. Panel (d) reports sampling times over ten batches of eight images after two warmup batches per condition. The error and cost comparisons cover ten JPEG levels for LlamaGen-L and three RAR-XL tokenizers with the `f32`, `uint8`, and `jpeg75` paths. These workloads differ between models. Within each workload, both procedures reuse generated tokens and decoded images across compatible processing settings and tokenizer variants.

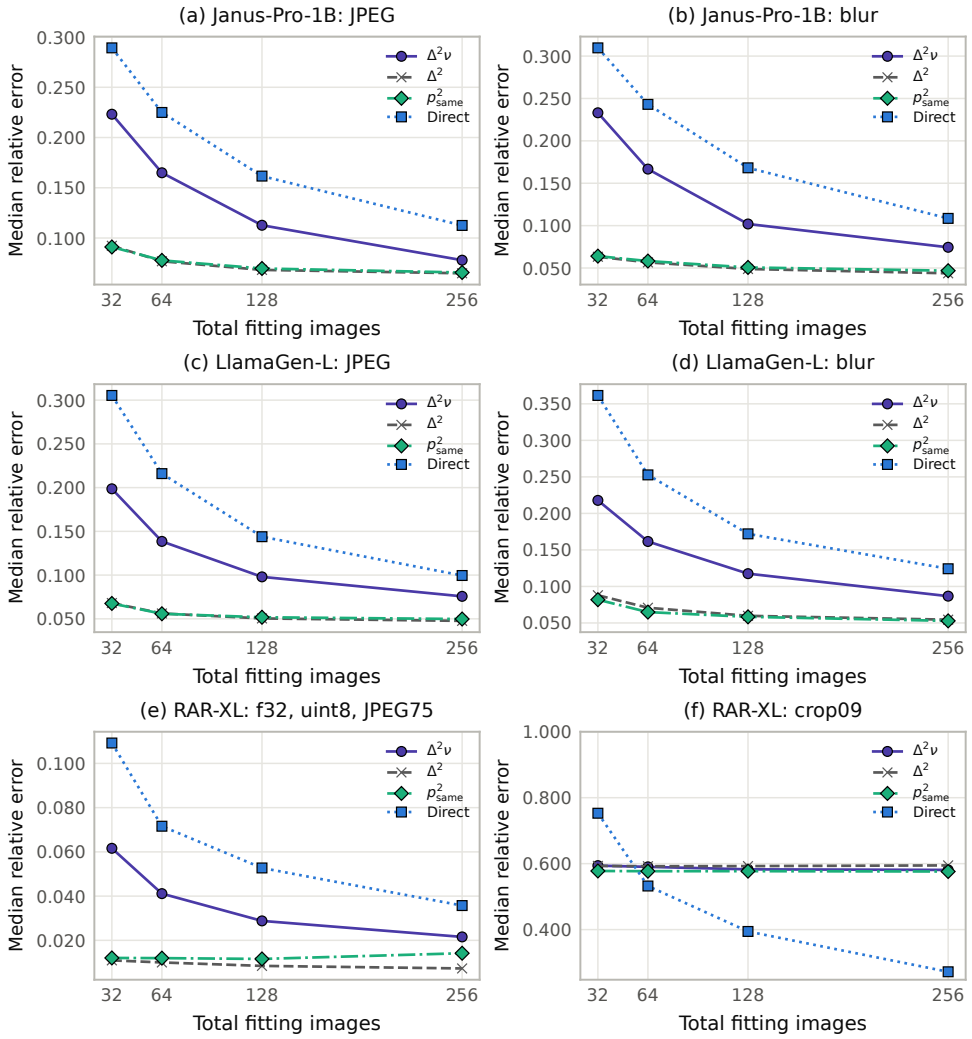


Figure 7: **Retention error at different fitting-image budgets.** Curves show median absolute relative error over paired subsamples of fixed fitting pools against an independent reference. Null-only methods use all B images as unwatermarked, whereas direct estimation divides B equally between unwatermarked and watermarked images. The crop panel displays the persistent approximation error of the null-only methods.

Table 17: **Cost and error of retention measurement and prediction.** Errors are dimensionless median absolute relative errors against reference η measured with independent $4{,}000$ unwatermarked and 512 watermarked images. Direct measurement also estimates η. Panels (a)–(b) summarize 200 fitting subsamples across settings. Panel (a) shares the same 128 unwatermarked images across procedures, whereas (b) fixes the total number of new images. Panel (c) uses the full cached pool, and (d) reports measured AR batch times as mean $\pm$ SD. GPU-stage times are projected for RTX 4090s.

Model	Procedure	New images: Unwatermarked	New images: Watermarked	Median error (fraction) ↓	GPU time (s) ↓
(a) Matched unwatermarked pool: 128 images					
LlamaGen-L	Direct measurement (η)	128	128	0.0995	203.84
	RenderGap prediction ($\Delta^2\nu$)	128	0	0.0980	101.67
	Gap-only prediction (Δ^2)	128	0	0.0504	101.67
RAR-XL	Direct measurement (η)	128	128	0.0357	170.69
	RenderGap prediction ($\Delta^2\nu$)	128	0	0.0288	85.22
	Gap-only prediction (Δ^2)	128	0	0.0084	85.22
(b) Equal total budget: 256 new images					
LlamaGen-L	Direct measurement (η)	128	128	0.0995	203.84
	RenderGap prediction ($\Delta^2\nu$)	256	0	0.0757	203.35
	Gap-only prediction (Δ^2)	256	0	0.0477	203.35
RAR-XL	Direct measurement (η)	128	128	0.0357	170.69
	RenderGap prediction ($\Delta^2\nu$)	256	0	0.0216	170.44
	Gap-only prediction (Δ^2)	256	0	0.0073	170.44
(c) Reusing 1,000 unwatermarked token pairs: additional cost					
LlamaGen-L	Direct measurement (η)	0	128	0.0423	102.17
	RenderGap prediction ($\Delta^2\nu$)	0	0	0.0460	0.00
	Gap-only prediction (Δ^2)	0	0	0.0436	0.00
RAR-XL	Direct measurement (η)	0	128	0.0183	85.47
	RenderGap prediction ($\Delta^2\nu$)	0	0	0.0090	0.00
	Gap-only prediction (Δ^2)	0	0	0.0114	0.00

(d) Measured AR sampling time per batch of eight images

Model	Unwatermarked (s) ↓	Watermarked (s) ↓
LlamaGen-L	5.89 ± 0.07	5.93 ± 0.05
RAR-XL	4.90 ± 0.03	4.92 ± 0.03

GPU time covers sampling, decoding, and re-encoding, excluding loading, CPU scoring, I/O, and reference construction. Settings are ten LlamaGen-L JPEG levels and three RAR-XL tokenizers under `f32`, `uint8`, and `jpeg75`, with generated tokens reused across settings. In (c), cached pairs include both generated and re-read tokens for every evaluated setting, so both predictors require only CPU scoring.

C.6 PREDICTION ACROSS A SHARED RANDOM KEY SET

We test prediction accuracy and independent key ranking on a common random key set. Ten distinct integer keys were sampled uniformly without replacement from 43 to 999 using seed 0 and fixed for both Janus-Pro-1B and LlamaGen-L. Each key has 512 watermarked images at $\delta = 2$, and Janus-Pro-1B has another 512 at $\delta = 0.4$. Each disjoint test splits the null into prediction and measurement halves, with 25 seeds used in both directions. We assess rank agreement using 20,000 key-label permutations, applying the same permutation across all 50 split directions in each draw. The permutation-test p-value compares the observed rank agreement with that obtained after shuffling key labels. The predictions lie close to the measurements under shared calibration on both models (Figure 8). For native Janus-Pro-1B at $\delta = 2$, the rank correlation falls from 0.9394 with shared calibration to 0.2235 with disjoint calibration (Table 18). Shared calibration can therefore make agreement between small key-to-key differences appear stronger than it is under independent evaluation. The disjoint tests leave ranking unresolved for both models, including Janus-Pro-1B at the lower bias $\delta = 0.4$. We also evaluate the ten keys using a separate 5,000-image unwatermarked

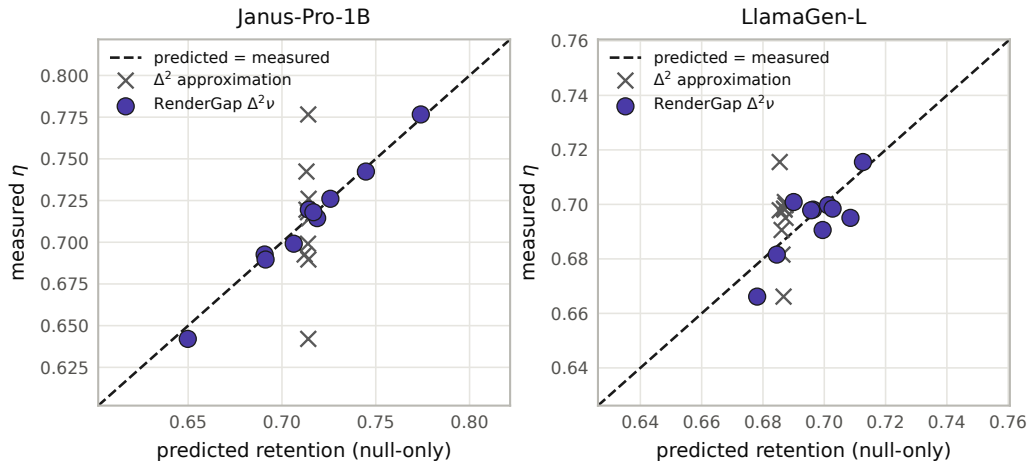


Figure 8: **Prediction at the same ten random keys on both models.** Global/`f32`, $\delta = 2$, and 512 watermarked images per key. Both predictions share null calibration with the measured retention, and the diagonal marks equality. Table 18 separately reports ranking with disjoint calibration.

Table 18: **Prediction and ranking at ten shared random keys.** Global/`f32` uses 512 watermarked images per key and bias. Disjoint results use 25 reused split seeds in both directions with a common key-label permutation. The two Janus null pools are separate datasets. Ratio ranges use shared calibration, and p is the key-label permutation-test p-value under disjoint calibration. Ratios closer to 1 indicate more accurate predictions, and higher Spearman's ρ_s indicates stronger rank agreement.

Model	δ	null images	$\Delta^2\nu/\eta$ range	shared $\rho_s \uparrow$	disjoint $\rho_s \uparrow$	Permutation p
Janus-Pro-1B	2.0	1,024	0.9928–1.0121	0.9394	0.2235	0.1338
Janus-Pro-1B	0.4	1,024	0.9721–1.0182	0.8788	0.1721	0.1966
Janus-Pro-1B	2.0	5,000	0.9860–1.0052	0.9758	0.3559	0.0529
Janus-Pro-1B	0.4	5,000	0.9723–1.0087	0.9030	0.2815	0.0951
LlamaGen-L	2.0	5,000	0.9845–1.0194	0.5515	−0.0538	0.6160

pool with the native tokenizer. Its disjoint-null correlations are higher at both biases, but ranking remains unresolved (Table 18). This comparison changes both the image pool and its size.

Table 19: **Detection before and after the render channel.** The global construction uses key 42 and sampling bias $\delta = 0.4$. Mean scores are z in the generated-token row and $\hat{z}$ in the remaining rows. Detection rates are TPR@1%FPR at the corresponding stage. The Janus-Pro-1B entries use native normalized BF16 VQ search.

Stage	Janus-Pro-1B		LlamaGen-L	
	Mean score ↑	TPR ↑	Mean score ↑	TPR ↑
Generated tokens (before)	4.4894	0.9609	3.0188	0.7891
`f32` (after)	3.8558	0.8672	2.4836	0.6016
`uint8` (after)	3.8496	0.8438	2.4573	0.5781
`jpeg75` (after)	3.0304	0.7266	1.9906	0.3984
`crop09` (after)	0.5306	0.0312	0.1957	0.0000

Table 20: **Retention and detection under Global and Cluster-64.** Native Janus-Pro-1B, key 42, $\delta = 0.4$, 128 watermarked images, and a shared 1,024-image null. Entries are point estimates. The recorded per-token KL budgets are 0.0195 and 0.0176 nats for Global and Cluster-64, respectively.

Construction	Channel	η	$\Delta^2\nu$	z	$\hat{z}$	TPR@1%FPR ↑
Global	`f32`	0.7377	0.6979	4.4894	3.8558	0.8672
Global	`uint8`	0.7353	0.6971	4.4894	3.8496	0.8438
Global	`jpeg75`	0.4557	0.3844	4.4894	3.0304	0.7266
Global	`crop09`	0.0140	0.0002	4.4894	0.5306	0.0312
Cluster-64	`f32`	0.8026	0.8022	1.9958	1.7880	0.1562
Cluster-64	`uint8`	0.8009	0.7979	1.9958	1.7862	0.1328
Cluster-64	`jpeg75`	0.4927	0.5182	1.9958	1.4009	0.0547
Cluster-64	`crop09`	0.0966	0.0051	1.9958	0.6203	0.0469

D Effects of image processing

D.1 Detection across render channels

Detection at the low embedding budget. We test Proposition 1's relation between retention and TPR@1%FPR across channels with the model, construction, key, and generated tokens fixed. The relation holds under the proposition's score assumptions with a fixed generated-token mean score. For the nominal low embedding budget of 0.02 nats per token, calibration gives δ=0.4 and we score each channel against the null. Table 19 reports the mean scores and detection rates for the global construction before and after each channel. Detection weakens even under `f32`, showing that the tokenizer round trip alone can reduce the signal. JPEG weakens it further, while crop-and-resize brings the observed detection rates near chance on both models. Because the generated tokens are fixed across channels, these comparisons isolate changes in preservation rather than changes in watermark bias. The before/after metric quantifies that preservation, and $\sqrt{\eta}\, z$ connects it to the Gaussian detection relation in Figure 4. Across constructions, both retention and the generated-token score change: Cluster-64 retains more than Global on Janus-Pro-1B but has lower detection success (Table 20). This is why retention and TPR answer different evaluation questions.

D.2 JPEG compression

To test prediction across compression strengths, the JPEG sweeps of Section 5 apply qualities $\{100, 95, 90, 80, 70, 60, 50, 40, 30, 20\}$ to Janus-Pro-1B and LlamaGen-L images with generated tokens held fixed. The quality-100 label denotes the unchanged 8-bit control without JPEG re-saving. In the controlled comparison, predictions use 1,000 unwatermarked images and measured retention uses a separate 4,000-image null and 512 watermarked images at δ=1. The three squared predictors track retention across compression strengths, with the variance correction helping the Janus-Pro-1B sweep more than the LlamaGen-L sweep (Table 2). Thus unwatermarked transitions capture the effect of compression, while the value of the variance correction depends on the model and calibration. The separate native Janus-Pro-1B `jpeg75` comparison at $\delta = 0.4$ also shows an underestimate

(Table 20). Its different bias and calibration pool make it an additional accuracy check rather than a matched test of bias dependence.

D.3 GAUSSIAN BLUR

The Gaussian-blur sweeps test whether prediction tracks retention as blur increases, using radius $r \in \{0, 0.4, 0.8, 1.2, 1.6, 2.0, 2.5, 3.0\}$ in pixels at each model's native resolution. Table 21 separates the contributions of the green-preservation gap and variance factor in a LlamaGen-L blur sweep. It uses the 128 watermarked fitting images at $\delta = 1$ and 1,000 unwatermarked fitting images from Section 5, sharing null calibration between prediction and measurement. As blur increases, Δ follows exact-token agreement p_{same} downward while ν stays close to 1, so the gap accounts for most of the decline in retention. Section 5 evaluates prediction accuracy using independent reference pools for both Janus-Pro-1B and LlamaGen-L.

Table 21: Gaussian-blur intervention on LlamaGen-L's render channel, tokens held fixed. The fraction of tokens recovered unchanged is the exact-token agreement p_{same}. η spans $117\times$ in this separate LlamaGen-L sweep. An accurate prediction $\Delta^2\nu$ is close to measured η.

radius r	p_{same}	Δ	ν	$\Delta^2\nu$	η
0.0	0.8217	0.8223	1.0832	0.7323	0.7344
0.4	0.8161	0.8157	1.0993	0.7314	0.7346
0.8	0.6949	0.6932	1.0552	0.5070	0.4981
1.2	0.4861	0.4850	0.9792	0.2303	0.2287
1.6	0.3309	0.3289	0.9413	0.1018	0.1084
2.0	0.2291	0.2283	0.9525	0.0496	0.0533
2.5	0.1407	0.1387	0.8770	0.0169	0.0183
3.0	0.0913	0.0900	0.9705	0.0079	0.0063

The close agreement between Δ and p_{same} explains why squared exact-token agreement is competitive here, whereas it misses label-preserving substitutions in the clustered comparison of Table 3(a).

D.4 CROP-AND-RESIZE

Crop-and-resize changes token-position correspondence, testing the limits of the token-local prediction. For native Janus-Pro-1B, crop reduces both the green-preservation gap Δ and measured retention η (Table 12). Francati et al. (2026) show crop-and-resize breaks a pseudorandom-code (PRC) watermark by key misalignment, whereas here crop-and-resize changes the re-read tokens themselves. The gap falls even for the Global partition, whose green list is independent of position. The failure therefore involves changes to token content, beyond losing alignment with a position-dependent key. Cluster-64 partly preserves the signal under crop, but measured retention and detection remain low (Tables 12 and 20).

Crop prediction errors. Eq. 12 is derived under the token-local approximation. We assess its accuracy empirically on `f32`, `uint8`, JPEG compression, and Gaussian blur. Crop additionally remaps token positions, which the token-local M omits. Under crop Δ collapses, yet on Janus-Pro-1B pooled measured η exceeds $\Delta^2\nu$ by more than an order of magnitude for both global and cluster-64 (Table 12). Controlled RAR-XL crop settings also show large errors across the null-only predictors (Table 2, Figure 6). These errors show that same-position token transitions are insufficient for predicting retention under crop-and-resize. Extending the prediction to geometric transformations requires accounting for the changed correspondence between token positions.

E WATERMARK CONSTRUCTIONS AND MODEL VARIATIONS

E.1 EXISTING WATERMARK CONSTRUCTIONS

We evaluate controlled adaptations of IndexMark and ClusterMark on Janus-Pro-1B and LlamaGen-L to examine what retention reveals alongside detection rates. Our IndexMark adaptation samples from the unmodified AR token generator's distribution and then swaps every red token to a paired green token (Tong et al., 2025). The swap happens after sampling and replaces a fraction 0.4988 of the tokens on LlamaGen-L and 0.5012 on Janus-Pro-1B (Table 23). The recorded sampling KL is 0 because the generator's sampling distribution is unchanged. It excludes distortion introduced by the swaps. Our ClusterMark adaptation biases sampling toward a keyed green set of codebook clusters chosen by the cluster of the previous token (Lukovnikov et al., 2026). We evaluate adapted IndexMark and ClusterMark embedding rules and score functions with each model's original tokenizer, with 512 watermarked images per construction and model. Sampling follows every other generation run in this paper, with plain multinomial sampling and no classifier-free guidance. The IndexMark adaptation uses the original VQ encoder in place of the trained Index Encoder, and the ClusterMark adaptation uses the cluster-bigram statistic without the optional learned predictors. Both models are scored against a $5{,}000$-image null. For LlamaGen-L this is the unwatermarked pool of Appendix B.3, while Janus-Pro-1B uses a separate $5{,}000$-image pool generated under the same protocol.

The adaptations differ from the published methods in pairing, clustering, and sampling as well as IndexMark's encoder. Since IndexMark releases its codebook pairing for LlamaGen-L only, we rebuild the Janus-Pro-1B pairing with the published recipe. We form a candidate graph from the 10 nearest neighbors of each codebook entry by cosine similarity. At $|\mathcal{V}|$=16,384 we then match greedily instead of by the recipe's exact maximum-weight blossom matching, and 94.8608% of the greedy pairs are edges of the candidate graph. ClusterMark's keyed green sets are built over 64 cosine k-means clusters of each model's own codebook with a target green fraction of 0.25, identically on both models. Our clustering uses ordinary cosine k-means, whereas ClusterMark uses constrained balanced Euclidean k-means. The realized token-level green mass on Janus-Pro-1B is 0.242–0.260.

Table 22: Existing constructions on Janus-Pro-1B and LlamaGen-L. Each row evaluates an adapted embedding rule with the model's original tokenizer. All post-render columns use the `f32` channel. η is the measured retention of each construction's own detection statistic. The sampling budget ε is the per-token KL in nats. The two ClusterMark rows at δ=5 spend $65.41\times$ and $66.89\times$ the nominal low embedding budget of 0.02 nats per token on LlamaGen-L and Janus-Pro-1B.

construction	model	ε (nats/token)	z	$\hat{z}$	η	TPR@1%FPR ↑
IndexMark, swap-all	LlamaGen-L	0.0000	15.4296	12.3801	0.6438	1.0000
	Janus-Pro-1B	0.0000	22.8944	19.2758	0.7089	1.0000
ClusterMark, δ=0.45	LlamaGen-L	0.0191	2.7110	2.2367	0.6807	0.4629
	Janus-Pro-1B	0.0185	3.8419	3.1098	0.6552	0.7559
ClusterMark, δ=5	LlamaGen-L	1.3081	23.7998	20.0765	0.7116	1.0000
	Janus-Pro-1B	1.3378	34.2994	28.6633	0.6984	1.0000

Table 22 illustrates why detection success alone cannot measure preservation. Adapted IndexMark detects every evaluated watermarked image on both models even though the round trip reduces its squared mean score. Its token swaps produce a strong green-count signal before rendering (Table 23). For adapted ClusterMark, increasing the bias raises TPR to 1.0000 on both models while retention changes much less. The $\delta = 5$ setting spends more than 65 times the nominal low budget, so this detection improvement accompanies a much stronger watermark. ClusterMark detects with a cluster-bigram statistic conditioned on the previous token, whereas the law of Section 4 is defined on the unigram green count, so these experiments compare measured retention and detection across models. Measuring retention alongside TPR reveals the signal reduction even when detection has saturated.

Table 23: **Token replacement and recovered green tokens under IndexMark.** Both models use 512 watermarked images. The replaced-token fraction is measured before rendering, and the final column is the mean green-token fraction recovered from the image under `f32`.

Model	Replaced-token fraction	Green fraction after `f32`
Janus-Pro-1B	0.5012	0.9124
LlamaGen-L	0.4988	0.8983

Table 24: **Prediction factors across** 100 **watermark keys.** Entries average unwatermarked estimates for Janus-Pro-1B under `f32`. The final row divides the cluster-64 mean by the global mean in each column. The prediction column averages $\Delta^2\nu$ for each key.

Construction	Mean Δ^2	Mean ν	Mean $\Delta^2\nu$
global	0.7139	0.9840	0.7024
cluster-64	0.8185	0.9804	0.8025
cluster-64 / global	1.15	1.00	1.14

E.2 Clustering and label preservation

The factorization identifies which part of the prediction changes under clustering. Across 100 Janus-Pro-1B keys, the increase in predicted retention comes mainly from the green-preservation gap, while the mean variance factor stays close to one for both partitions (Table 24). Thus the unwatermarked statistics associate the gain with preserving green labels when token identities change.

Retention and detection under clustering. The mean detection score after the round trip is $\hat{z} = \sqrt{\eta}\, z$. Changing the construction affects both the retained fraction and the generated-token mean score. Cluster-64 improves retention on Janus-Pro-1B but lowers z enough to reduce detection success (Table 20). The settings share a nominal KL target, with recorded per-token budgets of 0.0176 and 0.0195 nats for Cluster-64 and Global, respectively. Preserving labels therefore addresses one part of watermark design, but the construction must also create a strong enough signal during generation. The retention metric makes these two effects visible separately.

E.3 RAR-XL tokenizer variants

We test whether unwatermarked prediction captures the effect of tokenizer finetuning while AR generation stays fixed. RAR-XL (Yu et al., 2024) generates a 256×256 image as $T{=}256$ tokens from the MaskGIT-VQGAN codebook of $|\mathcal{V}|{=}1{,}024$ entries (Chang et al., 2022), and it is the testbed of WMAR (Jovanović et al., 2025). WMAR releases two encoder–decoder pairs finetuned for the round trip, one trained without and one with augmentations, as weight deltas on the original tokenizer. We sample with the same loop, budget calibration, and channels as for the other models, conditioned on the 60 ImageNet class labels of LlamaGen-L and without classifier-free guidance. We use the released raster-order sampling path, so generation order and decoder positions coincide. We can therefore pair generated and re-read tokens without a token permutation. At $\delta{=}0$ the loop returns the tokens of the released sampler exactly. Each set of generated tokens is then rendered through the original tokenizer and through both finetuned pairs, so the three render channels differ only in the tokenizer. The nominal low embedding budget of 0.02 nats per token again gives $\delta{=}0.4$, and the null holds 5,000 unwatermarked images per tokenizer.

To compare token agreement on real and generated images, we also evaluate the original RAR-XL tokenizer on the first 512 Imagenette2-160 validation JPEGs in sorted path order. Each image is converted to RGB, resized to a shorter edge of 256 pixels using bicubic interpolation, and center-cropped to 256×256. We compare its encoded tokens with those obtained after decoding and re-encoding, using floating-point decoder output without clamping or 8-bit conversion. We average the fraction of matching tokens over the 256 positions in each image and then over images. Table 25 compares this real-image agreement with agreement on generated unwatermarked tokens for all three tokenizers. Agreement is higher on the real-image inputs, motivating calibration on the AR token generator's own distribution when predicting its watermark retention.

Table 25: **Additional retention and detection results for RAR-XL.** The columns are the original MaskGIT-VQGAN and WMAR's two encoder–decoder pairs finetuned for the round trip, without (RCC-ft) and with augmentations (RCC-ft+aug), applied to the same generated tokens. Rows use the global construction unless stated. Pooled η averages the tested biases, and subscript $\pm$ values on $\Delta^2\nu$ and η are one bootstrap SD. Global/`f32` factors and pooled retention are in Table 15. The equal-retention p-value uses the joint bootstrap test of Appendix C.3. Real-image token agreement uses the 512 Imagenette images described above. For the fixed keys, the median ratio uses disjoint calibration and the retention range uses the full unwatermarked pool. The ten random keys of Appendix C.6 use shared calibration for ratio summaries and disjoint calibration for rank correlations and permutation p-values. The detection rows are at the low embedding budget with 512 images, and z is the mean detection score of the generated tokens before rendering. Ratios closer to 1 indicate more accurate predictions, and higher TPR and ρ_s indicate better detection and rank agreement, respectively.

	original	RCC-ft	RCC-ft+aug
(a) Exact-token agreement			
Null round-trip agreement `f32`	0.6790	0.9893	0.9832
Real-image token agreement `f32`	0.7777	0.9980	0.9963
(b) Retention across biases			
Equal-retention p `f32`	0.9445	0.3584	0.7166
Pooled η `jpeg75`	$0.3573_{\pm 0.0170}$	$0.8193_{\pm 0.0172}$	$0.9144_{\pm 0.0115}$
$\Delta^2\nu$ `jpeg75`	$0.3472_{\pm 0.0069}$	$0.8261_{\pm 0.0094}$	$0.9195_{\pm 0.0074}$
Cluster-64 pooled η `f32`	$0.6191_{\pm 0.0159}$	$0.9785_{\pm 0.0041}$	$0.9655_{\pm 0.0049}$
Cluster-64 $\Delta^2\nu$ `f32`	$0.5511_{\pm 0.0054}$	$0.9790_{\pm 0.0021}$	$0.9680_{\pm 0.0027}$
(c) Prediction across keys			
Fixed-key median $\Delta^2\nu/\eta$	0.95	1.00	1.00
Fixed-key η range	0.4376–0.4772	0.9726–0.9900	0.9583–0.9813
Random-key median $\Delta^2\nu/\eta$	0.96	1.00	1.00
Random-key $\Delta^2\nu/\eta$ range	0.92–0.97	1.00–1.01	1.00–1.01
Random-key max/min η	1.08	1.03	1.04
Random-key disjoint-null $\rho_s \uparrow$	0.2090	0.6674	0.6621
Random-key permutation p	0.1726	0.0030	0.0022
(d) Detection at the low embedding budget			
TPR@1%FPR $\uparrow$ `f32`	0.2402	0.5078	0.5352
TPR@1%FPR $\uparrow$ `uint8`	0.2520	0.5059	0.5312
TPR@1%FPR $\uparrow$ `jpeg75`	0.2109	0.4648	0.4707
TPR@1%FPR $\uparrow$ `crop09`	0.0254	0.0547	0.0410
z	2.4709	2.4709	2.4709
Per-token ε	0.0185	0.0185	0.0185

We also evaluate the fixed keys $\{3, 6, 7, 13, 22, 26, 33, 38, 39, 42\}$ under Global/`f32` at $\delta = 2$, with 128 watermarked images per key and 5,000 unwatermarked images. Retention ranges use the full unwatermarked pool, while prediction ratios use separate 2,500-image halves for prediction and measurement. The median ratio summarizes keys and 50 split directions from 25 seeds.

Holding generated tokens fixed isolates the tokenizer's effect on retention (Tables 15 and 25). Both finetuned pairs preserve nearly all of the squared mean score under `f32`, whereas the original tokenizer retains less than half. Under JPEG, training with augmentations preserves more signal than training without them. RenderGap predicts both the improvement from finetuning and the additional JPEG retention from augmentations using unwatermarked token pairs. The corresponding detection gains occur with the generated-token score and sampling budget held fixed. This comparison demonstrates a practical use of the predictor: assessing tokenizer changes before generating watermarked images.

The fixed-key and random-key results show that agreement in retention level extends beyond the key used in the tokenizer comparison. For the ten random keys drawn before calibration and evaluated with 512 watermarked images each, predictions slightly underestimate retention with the original tokenizer and are close to measurements with the finetuned pairs (Table 25). Disjoint-null rank tests

Table 26: **Measured retention and prediction ratios for a fixed clustered green list.** Original RAR-XL tokenizer, Cluster-64, key 42, and $\delta = 0.4$. Predictions use 1,000 null images, while measurement uses an independent 4,000-image null and 128 watermarked images. Ratios closer to 1 indicate more accurate predictions.

Channel	Reference η	$\Delta^2\nu/\eta$	Δ^2/η	p^2_{same}/η
`f32`	0.5816	0.97	1.01	0.80
`uint8`	0.5794	0.97	1.01	0.80
`jpeg75`	0.4734	1.00	1.06	0.76

Table 27: **Effect of VQ normalization and arithmetic on retention.** Matched Janus-Pro-1B Global/`f32` inputs and weights, with 1,024 null images calibrating both prediction and measurement. Measured η averages six biases with 128 watermarked images each. Subscripts give one joint-bootstrap SD. VQ denotes network arithmetic. FP32 dist. promotes the distance calculation, while FP32 norm.+dist. also promotes normalization. All FP32 controls promote the same stored checkpoint values. Prediction-to-measurement ratios closer to 1 indicate more accurate predictions.

Quantization rule	VQ	Search arithmetic	η	ν	Δ^2/η	$\Delta^2\nu/\eta$
Unnormalized Euclidean	BF16	BF16	$0.0075_{\pm 0.0032}$	0.2946	2.80	0.82
Unnormalized Euclidean	BF16	FP32 dist.	$0.2718_{\pm 0.0234}$	0.7480	1.32	0.99
Unnormalized Euclidean	FP32	FP32	$0.2870_{\pm 0.0240}$	0.7655	1.26	0.96
L_2-normalized (native)	BF16	BF16	$0.7161_{\pm 0.0272}$	0.9769	1.00	0.97
L_2-normalized (native)	BF16	FP32 dist.	$0.7234_{\pm 0.0273}$	0.9857	0.99	0.97
L_2-normalized (native)	BF16	FP32 norm.+dist.	$0.7232_{\pm 0.0275}$	0.9835	0.99	0.97
L_2-normalized (native)	FP32	FP32	$0.7204_{\pm 0.0266}$	0.9814	1.00	0.98

also support agreement for the two finetuned pairs, while ranking remains unresolved for the original tokenizer. Clustering improves retention under the original tokenizer, although its predicted value is less accurate than the Global prediction in this comparison. Crop produces large prediction errors with all three tokenizers (Table 2).

Table 3(a) reports the fixed Cluster-64 comparison under the original RAR-XL tokenizer. Its reference uses 128 watermarked images, compared with 512 in Table 2, and an independent 4,000-image null.

E.4 Quantization-rule sensitivity

We test whether RenderGap captures retention changes caused by the nearest-code search rule and its numerical precision. The Janus-Pro-1B checkpoint gives different re-read tokens when its VQ search uses unnormalized Euclidean distance or the native L_2-normalized encoder output and codebook vectors. The unnormalized Euclidean implementation is the Janus VQ search in Hugging Face Transformers 4.52.4. Table 27 applies both rules to the same generated Global/`f32` token sequences and VQ weights, varying the arithmetic of the VQ network, normalization, and distance calculation. Both decoded pixels and prequantizer encoder outputs agree between the implementations at matching precision. The nearest-code selection produces the difference. With unnormalized search, the gap-only predictor overestimates retention and the variance factor corrects much of that error: the predicted-to-measured ratio changes from 2.80 to 0.82 in BF16. Promoting the unnormalized distance calculation to FP32 improves retention, but it remains below the normalized-search result. Native normalized search gives similar retention across the tested arithmetic choices, with ν near 1 and little correction needed. The control shows why the evaluation must specify the quantization rule as well as the checkpoint, and where measuring null variance adds information beyond the green-preservation gap.

E.5 Previous-token KGW watermark

To examine prediction when green lists depend on context, we adapt the previous-token rule of Kirchenbauer et al. (2023) to image tokens. A CPU random permutation seeded by the key times the preceding token ID selects half of the codebook as green. The first image token is unbiased and

Table 28: **Retention prediction with previous-token green lists.** Each setting uses 120 unwatermarked and 120 watermarked images, with shared null calibration. Subscripts give one SD from 4,000 whole-image bootstrap draws. Ratios closer to 1 indicate more accurate predictions.

Model	δ	Channel	Measured η	$\Delta_{\mathrm{ctx}}^2/\eta$	$\Delta_{\mathrm{ctx}}^2\nu/\eta$
LlamaGen-L	0.4	`f32`	$0.4867_{\pm 0.0782}$	1.00	0.96
		`uint8`	$0.4587_{\pm 0.0781}$	1.04	1.00
		`jpeg75`	$0.2828_{\pm 0.0633}$	0.82	0.87
		`crop09`	$2.8286\times10^{-6}{}_{\pm 0.0029}$	$68.95^{\dagger}$	$68.77^{\dagger}$
	2	`f32`	$0.5021_{\pm 0.0652}$	0.96	0.93
		`uint8`	$0.5007_{\pm 0.0698}$	0.95	0.92
		`jpeg75`	$0.2843_{\pm 0.0481}$	0.82	0.86
		`crop09`	$1.1558\times10^{-7}{}_{\pm 0.0002}$	$1687.35^{\dagger}$	$1682.91^{\dagger}$
Janus-Pro-1B	0.4	`f32`	$0.5088_{\pm 0.0820}$	1.02	1.06
		`uint8`	$0.4716_{\pm 0.0719}$	1.07	0.98
		`jpeg75`	$0.1612_{\pm 0.0374}$	1.42	1.11
		`crop09`	$0.0025_{\pm 0.0026}$	$3.33\times10^{-3\dagger}$	$2.53\times10^{-3\dagger}$
	2	`f32`	$0.5651_{\pm 0.0891}$	0.92	0.95
		`uint8`	$0.4896_{\pm 0.0726}$	1.03	0.95
		`jpeg75`	$0.1956_{\pm 0.0371}$	1.17	0.92
		`crop09`	$0.0010_{\pm 0.0005}$	$8.15\times10^{-3\dagger}$	$6.20\times10^{-3\dagger}$

† Crop ratios are unstable because the measured post-render mean shift is close to zero.

unscored, and every subsequent bigram is counted, including repeats. During detection, the re-read predecessor selects the green list because the original predecessor is unavailable to the detector.

Each model uses 120 unwatermarked images and 120 watermarked images at each bias $\delta = 0.4$ and 2, with key 15485863. The same 60 model-specific conditions occur twice in each pool, with disjoint generation seeds. AR generation uses FP16, temperature 1, full-vocabulary sampling, and no classifier-free guidance. The VQ network uses FP32 for LlamaGen-L and native normalized BF16 search for Janus-Pro-1B. Prediction and measurement share the 120-image unwatermarked calibration pool.

We estimate a context-dependent gap Δ_{ctx} from paired generated and re-read bigrams. It is the probability that a re-read bigram is green conditional on its generated bigram being green, minus the corresponding probability conditional on the generated bigram being red. The two green labels use their respective predecessors. We compare Δ_{ctx}^2 and $\Delta_{\mathrm{ctx}}^2\nu$ with retention measured from the mean standardized bigram scores. We evaluate this context-dependent predictor empirically, whereas Proposition 2 assumes fixed green lists.

Across `f32`, `uint8`, and `jpeg75`, the corrected prediction-to-measurement ratios are 0.86–1.00 for LlamaGen-L and 0.92–1.11 for native Janus-Pro-1B (Table 28). Computing each green label from its own stage's predecessor therefore gives useful retention predictions in these single-key, shared-calibration tests, even though the green list changes with context. The variance correction improves some settings and worsens others. Crop-and-resize breaks the approximation on both models. For LlamaGen-L under crop, bootstrap resampling frequently changes the sign of the post-render mean shift, so the large point ratios do not give reliable fold-error estimates.