

ClusterMark: Towards Robust Watermarking for Autoregressive Image Generators with Visual Token Clustering

Denis Lukovnikov*
 Ruhr University Bochum
 denis.lukovnikov@rub.de

Andreas Müller*
 Ruhr University Bochum
 andreas.mueller-t1x@rub.de

Erwin Quiring
 _fbeta
 erwin.quiring@fbeta.de

Asja Fischer
 Ruhr University Bochum
 asja.fischer@rub.de

Abstract

In-generation watermarking for latent diffusion models has recently shown high robustness in marking generated images for easier detection and attribution. However, its application to autoregressive (AR) image models is underexplored. Autoregressive models generate images by autoregressively predicting a sequence of visual tokens that are then decoded into pixels using a VQ-VAE decoder. Inspired by KGW watermarking for large language models, we examine token-level watermarking schemes that bias the next-token prediction based on prior tokens. We find that a direct transfer of these schemes works in principle, but the detectability of the watermarks decreases considerably under common image perturbations. As a remedy, we propose a watermarking approach based on visual token clustering, which assigns similar tokens to the same set (red or green). We investigate token clustering in a training-free setting, as well as in combination with a more accurate fine-tuned token or cluster predictor. Overall, our experiments show that cluster-based watermarks greatly improve robustness against perturbations and regeneration attacks while preserving image quality, outperforming a set of baselines and concurrent works. Moreover, our methods offer fast verification runtime, comparable to lightweight post-hoc watermarking techniques. The source code is available at <https://github.com/lukovnikov/ClusterMark>

1. Introduction

Watermarking of AI-generated content is a key tool to address misuse [23], to achieve content authenticity, and to maintain the quality of training data by filtering out AI-generated material. Besides classic post-hoc watermarking,

one line of research explores techniques for embedding watermarks already during generation. Most research has focused on diffusion models [11, 37, 41], while autoregressive (AR) image models received less attention. However, recent progress in this field motivates a comprehensive exploration of AR image model watermarking.

AR models for image generation [3, 4, 9, 21, 29, 31, 32, 34, 35, 43, 44] generate images by predicting a sequence of tokens in a latent space that is then transformed to pixels using a decoder. A common approach is to use discrete tokens which are obtained after a quantization of the latent space.

In this work, we explore the embedding of watermarks during the autoregressive generation process. To this end, we consider token-level watermarks which have previously been studied in the context of large language models (LLMs) generating text. In particular, we adopt the KGW watermarking approach proposed by Kirchenbauer et al. [18] due to its simplicity and effectiveness. This approach builds on the idea of partitioning tokens into red and green sets, then biasing the model to favor green tokens during generation. This enables watermark detection based on the proportion of green tokens in a given text.

As a baseline, we first examine the direct transfer of KGW-style watermarks. Although this approach works in principle, we find that the watermarks lack robustness under common image perturbations. This can be explained by the following observations: To verify the watermark, a given image needs to be mapped back to the tokens that were originally produced by the AR model to generate the image. Firstly, the tokens might not be perfectly reconstructed due to a mismatch between the VQ-VAE’s image encoding and decoding procedures. Secondly, image perturbations further amplify the discrepancy between the original and the reconstructed tokens. Hence, a naive application of LLM watermarks is not suitable for image generators.

As a remedy, we introduce a new watermarking approach

*Equal contribution

based on visual token clustering. By clustering similar tokens together and building the green and red sets on cluster level instead of token level, the watermark detection robustness improves considerably. Even after image perturbations (e.g. blur), the reconstructed tokens are more likely to fall into the same cluster as the originally generated tokens.

We investigate the clustering approach in two variants: (1) a training-free version that only adapts the sampling procedure and (2) in combination with a trained token or cluster classifier for more robust token reconstruction.

In summary, our contributions are as follows:

- **Clustering for KGW-style watermarking for AR image models.** We are among the first to investigate token-level watermarking for AR image models and propose two novel approaches based on token clustering.
- **Robustness analysis.** We conduct an extensive analysis demonstrating the high robustness of our methods against common image perturbations and regeneration attacks.
- **Extensive evaluation.** We thoroughly evaluate the proposed techniques on different left-to-right autoregressive image models, i.e., LlamaGen [29] (versions GPT-B and GPT-L) and RAR-XL [44]. Our methods achieve state-of-the-art results with low verification runtime, outperforming the most recently developed concurrent in-generation watermarks for AR image generators with available code [16, 33].

2. Preliminaries

2.1. Autoregressive Image Generation

There are various works [3, 4, 9, 21, 29, 31, 32, 34, 35, 43, 44] that have developed autoregressive methods for image generation with different sampling approaches. In the following, we focus on common approaches of decoding discrete tokens from a latent space, that is, on approaches that rely on vector-quantized variational autoencoders (VQ-VAE) for mapping an image to a sequence of tokens.

Here, a color image $x \in \mathbb{R}^{3 \times H \times W}$ is first mapped to a latent space using an encoder $\mathcal{E}$, resulting in continuous feature vectors $\mathcal{E}(x) = z \in \mathbb{R}^{d \times h \times w}$. This latent representation is then quantized to visual tokens $q \in \mathbb{V}^{h \times w}$ based on a codebook. More precisely, there exists a fixed vocabulary $\mathbb{V}$ with a corresponding codebook $\mathbb{Q} \in \mathbb{R}^{|\mathbb{V}| \times d}$ containing d -dimensional vectors. These vectors are used to look up the nearest token to each of the $h \times w$ d -dimensional vectors in z , obtaining a collection of visual tokens q . The tokens can be mapped back to an image by first dequantizing them using the same codebook $\mathbb{Q}$ and then mapping the latent vectors into pixel space using a corresponding decoder $D : \mathbb{R}^{d \times h \times w} \rightarrow \mathbb{R}^{3 \times H \times W}$.

An autoregressive image generator models the joint distribution of image tokens $p(q_1, q_2, \dots, q_{h \cdot w} | s)$ given a se-

mantic context vector s (such as a class or text) by decomposing it into $\prod_i^{h \cdot w} p_\theta(q_i | q_{1:i-1}, s)$. During generation, a sequence of tokens is consecutively generated by sampling from p_θ at every step. The default setting is to generate an image by producing tokens describing parts of the image, in a left-to-right, top-down scanning order. We consider this setting for our watermarking approach. Note, however, that alternative orderings and parallel decoding [3, 4, 21, 32] have also been explored.

2.2. LLM Watermarking

Several methods have been proposed for watermarking LLM-generated text [5, 18–20, 26, 38, 42]. In this work, we adopt the KGW watermarking technique by Kirchenbauer et al. [18] due to its simplicity and effectiveness. At every generation step, this technique uses the previously decoded tokens to seed a random function that is used to partition the vocabulary of the generator into two sets, one referred to as “green”, the other as “red”. More formally, given the sequence of tokens decoded so far, $y_{1:i-1} = (y_1, \dots, y_{i-1}) \in \mathbb{V}^{i-1}$, we first compute a hash $o_i = \text{hash}(\kappa, y_{i-l}, \dots, y_{i-1})$ using a random hashing function like SHA256. Here, κ is a predefined secret key that remains the same across multiple generations, and l is the number of preceding tokens that are used. The hash o_i is then used to seed a pseudorandom number generator (PRNG) to randomly select a fraction γ of tokens from $\mathbb{V}$ to form the set of green tokens G_i of size $\lfloor \gamma * |\mathbb{V}| \rfloor$. We refer to this process as $g(\cdot)$, which computes the green set based on previously generated tokens: $G_i = g(\kappa, y_{i-l}, \dots, y_{i-1})$. The red set of tokens is then its complement: $R_i = \mathbb{V} \setminus G_i$. Given G_i , the bias mask $m_i \in \{0, 1\}^\mathbb{V}$ is defined by setting the mask to one for green tokens: $m_i[v] = \mathbf{1}_{G_i}(v)$. Next, the model’s distribution over $\mathbb{V}$ is biased to favor tokens from the green set by adding some value δ to their logits $f_\theta(y_{1:i-1}) \in \mathbb{R}^\mathbb{V}$ predicted by the model before normalizing using the softmax:

$$p'_\theta(y_i | y_{1:i-1}) = \text{softmax}(f_\theta(y_{1:i-1}) + m_i * \delta) .$$

The smaller the value of δ , the less the generator’s distribution is affected, but the weaker the watermark signal.

In order to verify the watermark for the given sequence $(y_1, \dots, y_T)$, we can simply recompute the green sets G_i for every given token given the preceding context. This is fast as there is no need to access the original model. Finally, to verify the watermark, we perform a one-side (right-tailed) test assessing how unlikely it is to observe the given number of green tokens, $N_g = \#_{\text{green}}$, in the entire sequence by chance. The null hypothesis is that N_g is distributed according to a binomial distribution: $H_0 : N_g \sim \text{Binomial}(T, \gamma)$. We compute the right-tailed p-value $p = \Pr(X \geq N_g)$ and reject H_0 for a sequence if its p-value p is lower than a chosen threshold ρ .

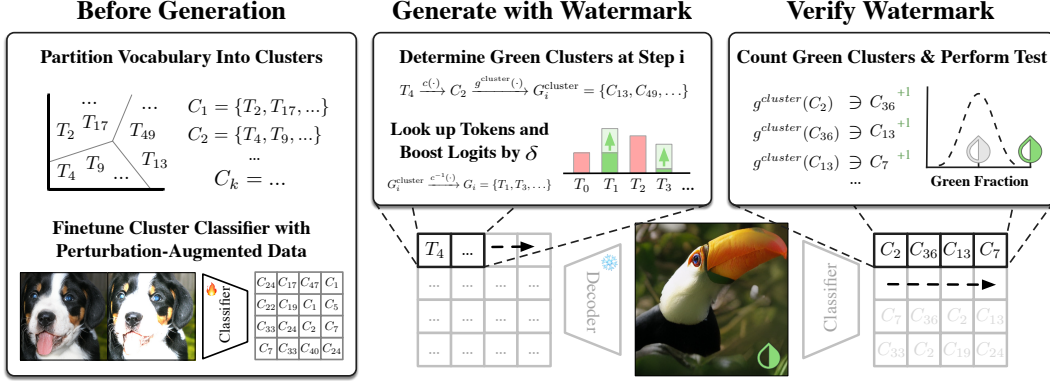


Figure 1. Overview of our proposed cluster-based watermark for AR image generators. Before generation, tokens are clustered based on codebook vector similarity, partitioning the vocabulary into a set of clusters. Additionally, a token or cluster classifier can be trained to further boost robustness (This figure shows the variant with the cluster classifier). At every generation step, the set of clusters is partitioned into green and red sets and sampling is biased towards tokens in green clusters. During verification, if the fraction of green tokens is significantly higher than random chance, the image is considered watermarked.

3. Watermarking for AR Image Generators

First, we introduce a baseline watermark for autoregressive image generation. Next, we describe our token clustering and fine-tuning strategies for improving robustness.

3.1. Baseline AR Watermark

As a **baseline** variant, we consider the naive application of KGW watermarking on the token sequence $\{q_i\}_1^{h.w} = (q_1, \dots, q_{h.w})$ produced by the AR image generation model. The **generation** of the image consists of two phases: (1) sampling the token sequence $\{q_i\}_1^{h.w}$ using the AR model, and (2) mapping the token sequence to an image x using the VQ-VAE. During sampling, the hash for the red-green partitioning is computed from the previous token: $o_i = \text{hash}(\kappa, q_{i-1})$. We refer to κ as the (hash) prefix. For the **verification** of a given image x , we first use the encoder and quantizer of the VQ-VAE to produce a token sequence $\{q'_i\}_1^{h.w}$, and then count the fraction of green tokens along the generation order.

3.2. Cluster-Based AR Watermark

Due to the quantization in the encoding step of the VQ-VAE, even small image perturbations can lead to entirely different tokens being predicted for a large fraction of the tokens (since it is sufficient that the perturbation moves the latent representation closer to another token in the codebook). This sensitivity of the token reconstruction leads to poor robustness under image perturbations, as similar tokens are randomly assigned to either red or green sets. Thus, we cluster the codebook tokens based on their embeddings to improve robustness. Tokens are partitioned into green and red sets such that all tokens belonging to the same cluster are either in the green or in the red set.

Clustering the vocabulary. Prior to generation, we apply k -means clustering to partition the vocabulary $\mathbb{V}$ into k disjoint subsets $(C_1, \dots, C_k)$. This partitioning is computed using the Euclidean distances between the token representations in the codebook $\mathbb{Q}$. For convenience, we define a cluster assignment function $c(q) \in \{1, \dots, k\}$, which returns the index of the cluster containing token q .

Watermarking with clusters. During **generation**, there are two main differences compared to the basic case. The hash o_i is computed based on the cluster of the preceding token: $o_i = \text{hash}(\kappa, c(q_{i-1}))$, where κ is the (hash) prefix that is kept constant across multiple generations. The hash is then used to compute a green set G_i^{cluster} over clusters $(C_1, \dots, C_k)$, rather than over the vocabulary $\mathbb{V}$. The green set over $\mathbb{V}$ is simply the union over green clusters: $G_i = \bigcup_{C_i \in G_i^{\text{cluster}}} C_i$. This green set of tokens G_i is then used to bias the generative distribution towards generating one of the green tokens, similarly to the baseline variant. Note that the red/green assignment is shared by all tokens in a cluster. This should improve robustness to image perturbations, under the assumption that perturbed images are likely to decode to semantically or geometrically similar tokens within the same cluster. **Verification** works similarly to the baseline variant with the difference that hashing and masking are again done on cluster level. Algorithm 1 describes the generation and verification process in detail.

3.3. Token and Cluster Reconstruction

The approach discussed so far is training-free and only relies on the clustering of tokens based on their codebook vectors. As an extension of this cluster-based watermarking approach, we also investigate **fine-tuning the VAE encoder** to improve cluster reconstructions. Here, we experiment with

two approaches: (1) a token predictor that predicts the token indices that can then be mapped to cluster indices and (2) a cluster predictor that directly outputs the cluster indices. To increase robustness, fine-tuning is performed adversarially with perturbation-augmented images by applying a perturbation function $\phi(\cdot)$.

The token and cluster classifier ($\mathcal{M}_T$ and $\mathcal{M}_C$, respectively) are implemented by taking a copy of the VQ-VAE encoder $\mathcal{E}$ without the pre-quantization layer, and adding a simple classification output layer. They are fine-tuned using unwatermarked images generated by the AR model. The token classifier is trained with the following loss, which tries to produce the original tokens:

$$\mathcal{L}_{TC} = \mathbb{E}_{(x, \{q_i\}_1^{h.w}) \sim p_\theta} \left[\sum_i \text{CE}(\mathcal{M}_T(\phi(x))_i, q_i) \right], \quad (1)$$

whereas the cluster classifier is trained with another loss, which tries to reproduce the clusters of the original tokens directly:

$$\mathcal{L}_{CC} = \mathbb{E}_{(x, \{q_i\}_1^{h.w}) \sim p_\theta} \left[\sum_i \text{CE}(\mathcal{M}_C(\phi(x))_i, c(q_i)) \right], \quad (2)$$

Note that the use of these classifiers does not require any changes to the watermarked image generation procedure.

3.4. Prefix Tuning

In preliminary experiments, we observed that the choice of hash prefix κ (the secret) significantly affects watermark detection performance. For certain images (e.g., those with large uniform regions), specific cluster transitions occur more frequently, leading to unusually high green counts even in unwatermarked images and resulting in false positives. To address this, we adopt a simple empirical approach by evaluating multiple κ values and selecting the best-performing one.

4. Evaluation

In this section, we evaluate the effectiveness of our proposed watermarking approach. We first detail the experimental setup, including the models and metrics used, before presenting our main findings and an ablation study.

4.1. Experimental Setup

Models. We conduct experiments on three left-to-right autoregressive models: LlamaGen [29] (GPT-B and GPT-L) and RAR-XL [44]. For both models, their class-to-image variants are used, which generate images conditioned on ImageNet [28] classes. We use the best reported generation settings. See Section A in Suppl. Material for details.

Evaluation Metrics. Similarly to previous work, we report TPR@FPR=1% and AUC to measure watermark detection performance. The main evaluation (Tab. 1) uses 2,000

Algorithm 1 Cluster-Based Watermarking

```

1: function COMPUTEGREENSET( $q_{i-l:i-1}, \mathbb{V}, C, \kappa, \gamma$ )
2:    $\mathcal{G} \leftarrow \text{PRNG}(\text{hash}(\kappa, c(q_{i-l:i-1})))$   $\triangleright$  Seed PRNG
3:   for  $j = 1, 2, \dots, k$  do
4:      $r_j \leftarrow \mathcal{G}.\text{uniform}(0, 1)$ 
5:   end for
6:    $k_{\text{green}} \leftarrow \lfloor \gamma \cdot k \rfloor$   $\triangleright$  Number of green clusters
7:    $G^{\text{cluster}} \leftarrow \text{argsort}((r_1, r_2, \dots, r_k))[:k_{\text{green}}]$ 
8:    $G = \bigcup \{C_j : C_j \in G^{\text{cluster}}\}$   $\triangleright$  Merge green
9:   return  $G$ 
10: end function

11: function GENERATEWITHWM( $f_\theta, \mathbb{V}, C, \kappa, \gamma, \delta, s$ )
12:    $q_1 \sim \text{softmax}(f_\theta(s))$   $\triangleright$  Sample first token
13:   for  $i = 2, \dots, h.w$  do  $\triangleright$  Generate  $h.w$  tokens
14:      $G_i \leftarrow \text{ComputeGreenSet}(q_{i-l:i-1}, \mathbb{V}, C, \kappa, \gamma)$ 
15:      $z_i \leftarrow f_\theta(q_{1:i-1}, s)$   $\triangleright$  Compute model logits
16:     for  $v \in G_i$  do
17:        $z_i[v] \leftarrow z_i[v] + \delta$   $\triangleright$  Bias green
18:     end for
19:      $q_i \sim \text{softmax}(z_i)$   $\triangleright$  Sample next token
20:   end for
21:   return  $q_{1:h.w}$ 
22: end function

23: function VERIFYWATERMARK( $q_{1:h.w}, \mathbb{V}, C, \kappa, \gamma$ )
24:    $\text{green\_count} \leftarrow 0$ 
25:   for  $i = 2, \dots, h.w$  do
26:      $G_i \leftarrow \text{ComputeGreenSet}(q_{i-l:i-1}, \mathbb{V}, C, \kappa, \gamma)$ 
27:     if  $q_i \in G_i$  then
28:        $\text{green\_count} \leftarrow \text{green\_count} + 1$ 
29:     end if
30:   end for
31:    $p\text{-value} \leftarrow 1 - \text{CDF}_{\text{Binom}}(\text{green\_count} - 1, h.w, \gamma)$ 
32:   return  $p\text{-value}$ 
33: end function

```

watermarked images and provides empirical numbers computed against a set of 2,000 generated, unwatermarked images. Image quality is assessed by computing the FID of 50k generated images against the ImageNet validation set.

Watermark Settings. We explore different settings for the cluster-based watermark by varying the number of clusters ($k \in 8, 16, 32, 64, 128$), the strength of the watermark ($\delta \in 2, 5$), and the fraction of green tokens ($\gamma \in 0.5, 0.25$). This is done for different hash prefixes ($\kappa \in 1, \dots, 8$) from which we pick the best-performing one. The optimal settings were determined on a validation set disjoint from the test set used in Table 1 and by using different perturbations. This procedure is described in detail in Section B in the Supplementary Material.

Model	FID ↓	Clean	JPEG 20	Gauss. Blur R 3	Gauss. std .2	Salt&Pepper .1	Color Jitter	Regeneration
LlamaGen GPT-B 256 × 256 (Baseline FID=6.01)								
DWT-DCT-SVD	6.51	1.000 — 1.000	0.415 — 0.006	0.975 — 0.785	0.503 — 0.005	0.506 — 0.009	0.542 — 0.161	0.826 — 0.494
RivaGAN	6.34	1.000 — 0.999	0.928 — 0.510	0.996 — 0.963	0.951 — 0.661	0.958 — 0.701	0.848 — 0.752	0.853 — 0.475
TrustMark	6.06	N/A — 1.000	N/A — 0.154	N/A — 0.320	N/A — 0.023	N/A — 0.022	N/A — 0.650	N/A — 0.373
SSL	6.19	1.000 — 1.000	0.837 — 0.242	0.994 — 0.938	0.553 — 0.035	0.545 — 0.022	0.982 — 0.919	0.937 — 0.685
IndexMark (+IE)	5.84	1.000 — 1.000	0.969 — 0.821	0.761 — 0.171	0.631 — 0.055	0.635 — 0.071	0.907 — 0.732	0.951 — 0.761
Ours (No Clustering)	5.90	0.999 — 0.999	0.928 — 0.692	0.621 — 0.068	0.626 — 0.075	0.621 — 0.069	0.851 — 0.573	0.916 — 0.710
+ Token Classifier	5.90	1.000 — 1.000	0.877 — 0.564	0.818 — 0.366	0.912 — 0.651	0.999 — 0.998	0.974 — 0.926	0.881 — 0.677
Ours (Clustering, k=64)	6.12	1.000 — 1.000	0.993 — 0.956	0.951 — 0.663	0.861 — 0.369	0.875 — 0.402	0.926 — 0.792	0.997 — 0.972
+ Token Classifier	6.12	1.000 — 1.000	0.978 — 0.875	0.995 — 0.949	0.980 — 0.900	1.000 — 1.000	0.960 — 0.917	0.996 — 0.968
+ Cluster Classifier	6.12	1.000 — 1.000	0.982 — 0.893	0.992 — 0.925	0.982 — 0.895	1.000 — 0.999	0.985 — 0.951	0.993 — 0.935
LlamaGen GPT-L 384 × 384 (Baseline FID=4.50)								
DWT-DCT-SVD	4.54	1.000 — 0.999	0.461 — 0.010	0.975 — 0.815	0.518 — 0.009	0.502 — 0.008	0.544 — 0.195	0.888 — 0.628
RivaGAN	4.62	1.000 — 0.999	0.958 — 0.702	0.999 — 0.983	0.950 — 0.682	0.959 — 0.733	0.859 — 0.790	0.910 — 0.596
TrustMark	4.46	N/A — 0.997	N/A — 0.338	N/A — 0.947	N/A — 0.029	N/A — 0.025	N/A — 0.636	N/A — 0.523
SSL	4.35	1.000 — 1.000	0.905 — 0.453	0.998 — 0.984	0.612 — 0.058	0.571 — 0.048	0.988 — 0.954	0.986 — 0.893
IndexMark (+IE)	4.49	0.999 — 0.999	0.976 — 0.836	0.841 — 0.307	0.642 — 0.035	0.654 — 0.041	0.929 — 0.793	0.983 — 0.834
Ours (No Clustering)	4.36	1.000 — 1.000	0.960 — 0.8255	0.726 — 0.193	0.657 — 0.097	0.634 — 0.082	0.884 — 0.683	0.961 — 0.812
+ Token Classifier	4.36	1.000 — 1.000	0.901 — 0.641	0.887 — 0.550	0.9128 — 0.698	0.999 — 1.000	0.980 — 0.954	0.929 — 0.729
Ours (Clustering, k=64)	4.85	1.000 — 1.000	0.996 — 0.975	0.983 — 0.905	0.899 — 0.483	0.923 — 0.543	0.931 — 0.849	0.999 — 0.997
+ Token Classifier	4.85	1.000 — 1.000	0.985 — 0.931	0.999 — 0.991	0.984 — 0.925	1.000 — 1.000	0.986 — 0.975	0.999 — 0.991
+ Cluster Classifier	4.85	1.000 — 1.000	0.989 — 0.944	0.995 — 0.971	0.991 — 0.956	1.000 — 1.000	0.982 — 0.963	0.999 — 0.990
RAR-XL 256 × 256 (Baseline FID=3.13)								
DWT-DCT-SVD	3.87	1.000 — 1.000	0.411 — 0.003	0.983 — 0.802	0.504 — 0.004	0.494 — 0.004	0.534 — 0.135	0.819 — 0.462
RivaGAN	3.64	1.000 — 1.000	0.943 — 0.550	0.998 — 0.983	0.956 — 0.651	0.963 — 0.730	0.860 — 0.785	0.870 — 0.496
TrustMark	3.31	N/A — 0.998	N/A — 0.118	N/A — 0.248	N/A — 0.024	N/A — 0.021	N/A — 0.651	N/A — 0.350
SSL	4.31	1.000 — 1.000	0.855 — 0.227	0.996 — 0.955	0.598 — 0.029	0.572 — 0.017	0.985 — 0.927	0.953 — 0.699
WMAR (FT+AUGS)	4.23	1.000 — 1.000	0.992 — 0.954	0.964 — 0.652	0.879 — 0.476	0.861 — 0.423	0.914 — 0.718	0.954 — 0.758
Ours (No Clustering)	3.14	1.000 — 0.999	0.927 — 0.586	0.638 — 0.045	0.503 — 0.026	0.528 — 0.021	0.859 — 0.532	0.947 — 0.750
+ Token Classifier	3.14	1.000 — 1.000	0.958 — 0.792	0.930 — 0.565	0.915 — 0.626	1.000 — 0.999	0.984 — 0.919	0.966 — 0.818
Ours (Clustering, k=64)	3.96	1.000 — 1.000	0.992 — 0.901	0.772 — 0.100	0.723 — 0.087	0.739 — 0.093	0.935 — 0.661	0.986 — 0.868
+ Token Classifier	3.96	1.000 — 1.000	0.995 — 0.960	0.986 — 0.837	0.984 — 0.885	1.000 — 1.000	0.991 — 0.956	0.994 — 0.945
+ Cluster Classifier	3.96	1.000 — 1.000	0.996 — 0.964	0.979 — 0.770	0.975 — 0.800	1.000 — 1.000	0.987 — 0.943	0.993 — 0.922

Table 1. Main results on a set of challenging perturbations. Detailed perturbation configurations are provided in Appendix A of the Supplementary Material. We report AUC and TPR@FPR=1% across various perturbation and regeneration attacks. For *Ours (No Clustering)* we set the penalty $\delta = 5$ and the green-token fraction $\gamma = 0.25$. For *Ours (Clustering)* and for our method using the token/cluster classifier we use $k = 64$ clusters and the same δ and γ . To improve readability, the **best** and **second best** entries in each column are highlighted.

Robustness analysis. The robustness is evaluated w.r.t. both perturbations and regeneration attacks on a distinct set of 2,000 images. We use an extensive set of image perturbations, such as Gaussian noise and blur. For regeneration attacks, we use the SD1.5 and FLUX.1 autoencoders, regenerating the watermarked images by first encoding them with the VAE encoder and then decoding them with the VAE decoder. We further add a diffusion-based regeneration attack [46] with default settings, i.e. 60 steps of denoising using Stable Diffusion 2.1 [27]. Visual examples of all applied perturbations as well as the detailed perturbation settings can be found in Section A in the Supplementary Material.

We additionally evaluated geometric transformations and found that our watermark is vulnerable to rotation and cropping, which is in line with previous in-generation watermarks [13, 16, 33, 37, 41]. This issue can be mitigated by using an *image synchronization layer* such as SyncSeal [12]. A detailed analysis of SyncSeal’s ability to undo

geometric transformations and restore successful detection of our watermark is provided in Section B in the Supplementary Material.

Baselines. We benchmark our AR watermarking approach against post-hoc watermarks: *DWT-DCT-SVD* [25], *RivaGAN* [45], *TrustMark* [2], and *SSL* [10], applied after generating unwatermarked images. We also compare against concurrent AR watermarks for which source code is readily available: (1) *IndexMark* [33], which was implemented only on LlamaGen, and (2) *WMAR* [16], which was developed for RAR-XL among two other models. For both, we evaluate with their best settings, namely IndexMark with Index Encoder and WMAR with fine-tuning of the VAE encoder and decoder with data augmentation. Besides these schemes, most other in-generation watermarks target diffusion models and are not applicable to AR models.

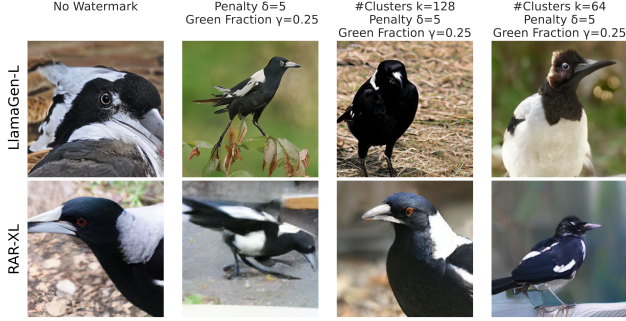


Figure 2. Examples of images generated by LlamaGen-L (top) and RAR-XL (bottom), shown (column-wise) (1) unwatermarked images, (2) watermarked images without clustering, and (3,4) watermarked images with clustering enabled. Note that while images were generated using the same seed, they differ visually because our watermarking method modifies the generation process rather than applying a post-hoc watermark to given cover images. For more examples, see Section C in the Supplementary Material.

Training details. To fine-tune the token/cluster classifier, we use 100k images generated using the model *without* applying watermarks. A predefined set of 64 clusters is used, obtained by k-means clustering. The model was fine-tuned for 30 epochs with batch size 16, which took ~ 12 hours on one A40 GPU. The set of perturbations used during fine-tuning is given in Section A in the Supp. Material. We use a linearly increasing perturbation schedule.

4.2. Results

Table 1 shows our main results. The reported watermarking hyper-parameters ($k = 64$, $\delta = 5$, $\lambda = 0.25$) are determined by choosing the best trade-off between image quality (measured using FID) and robustness. The ablation study in Section 4.3 gives further details on the selection. Overall, the results show that cluster-based watermarking indeed significantly improves robustness, even in its training-free variant. When combined with a token classifier or cluster classifier, it achieves the best overall results in the comparison, also outperforming concurrent work. Visual examples of generated images are provided in Figure 2, with more examples provided in Section C in the Supplementary Material.

Baseline variant. Our simple token-level baseline watermarking method (*Ours (No clustering)*) shows near perfect TPR@FPR1% in the clean setting, with image quality matching that of non-watermarked images, as measured by the FID. However, under even mild image perturbations, such as Gaussian noise, TPR significantly drops, rendering the watermark unusable.

Effect of clustering. The training-free cluster-based approach (*Ours (Clustering)*) drastically increases robustness

Model Resolution	LlamaGen		RAR
	256 ² (GPT-B)	384 ² (GPT-L)	256 ² (XL)
DWT-DCT-SVD	26.5 $\pm$ 11.7	68.4 $\pm$ 26.9	23.7 $\pm$ 9.9
RivaGAN	24.8 $\pm$ 13.6	61.5 $\pm$ 20.4	57.6 $\pm$ 23.7
TrustMark	98.7 $\pm$ 15.5	95.2 $\pm$ 17.9	95.0 $\pm$ 17.9
SSL	219.7 $\pm$ 0.4	222.8 $\pm$ 0.9	222.2 $\pm$ 1.1
Ours	12.3 $\pm$ 10.9	23.6 $\pm$ 10.8	14.9 $\pm$ 13.2
+ Token Classifier	11.3 $\pm$ 9.3	22.2 $\pm$ 10.5	10.5 $\pm$ 8.4
+ Cluster Classifier	12.8 $\pm$ 10.3	22.7 $\pm$ 8.4	9.6 $\pm$ 8.6

Table 2. Verification runtime (in milliseconds per image) of our methods against baselines across different models.

against various perturbations for both LlamaGen variants as well as RAR-XL. While watermark detectability is significantly improved, robustness to blurring and to salt-and-pepper noise remains hard to achieve. However, note that blurring and salt-and-pepper noise are very noticeable and destructive (see Section A in the Supplementary Material for an example), which severely limits the usefulness of the perturbed image. With a sufficiently large number of clusters ($k = 64$ in Table 1), the FID remains low.

Effect of token or cluster classifier. Using the *Token Classifier* or *Cluster Classifier* boosts robustness to several image perturbations for all studied models, outperforming all baselines. The effect is most pronounced for salt-and-pepper noise, color jitter, and blurring, which are also the most destructive transformations for the training-free variants. For example, the cluster classifier successfully learned to undo salt-and-pepper noise, yielding near-perfect watermark detection—a clear increase from 9.3–54.3% TPR@1%FPR when using only clusters ($k = 64$).

Comparison to baselines. The results show that our best method outperforms all baselines, including the concurrently developed in-generation AR watermarks IndexMark [33] and WMAR [16], by a large margin across all tested models on almost all perturbations. The exception is Gaussian Blur, where RivaGAN achieves slightly higher robustness. Comparing the FID shows that our methods achieve generator quality close to unwatermarked generation, unlike WMAR, which achieves worse FID in our evaluation using its code and pretrained models. We also compute CLIP scores and see no significant degradation in text conditioning performance compared to the non-watermarked case. All CLIP scores are in the 29–31 range and are reported in Section B in the Supp. Material.

Runtime. Table 2 shows the **verification runtime** of the methods evaluated in this work. Time is measured by running watermark verification on single-example batches for 1,000 images. For the baselines, we use a commonly

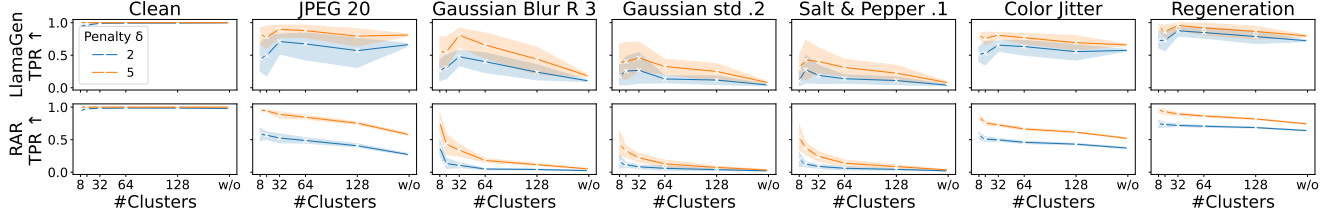


Figure 3. Empirical TPR@FPR=1% under different perturbations for LlamaGen (GPT-L) and RAR-XL for different numbers of clusters k and penalties δ , with green token fraction $\gamma = 0.25$ for the training-free approach. Results are reported over multiple prefixes (8 for RAR-XL and LlamaGen): Lines indicate the average and the shaded area borders the standard deviation.

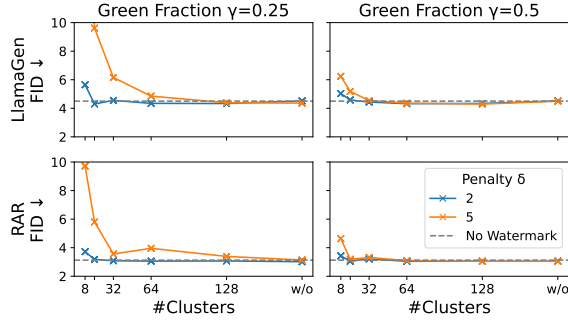


Figure 4. FID obtained for LlamaGen (GPT-L) and RAR-XL across different number of clusters k , different fractions of green tokens γ , and penalties δ . See Section B in the Supplementary Material for full results across all settings.

used implementation for DWT-DCT-SVD and RivaGAN¹ as well as the official implementations of TrustMark² and SSL³, all with GPU acceleration enabled. The results indicate that the verification of token-based watermarks for AR models is as lightweight as for DWT-DCT-SVD (10-25 milliseconds). This is in stark contrast to recently proposed in-generation watermarking methods for diffusion models (e.g., Tree-Ring [37]), which in addition to encoding the image to latent space also require full inversion of the diffusion process, resulting in roughly the same overhead for verification as for generation. For IndexMark and WMAR, we observe verification times similar to those of our method. For example, WMAR deployed with RAR-XL takes ~ 10 ms (after filling their green lookup table cache), while IndexMark takes 15 ms (GPT-B) and 30 ms (GPT-L).

Lastly, the **generation runtime overhead** is negligible with an average overhead of 0.06s out of 3.7s total generation time and 0.3s out of 14.3s for LlamaGen GPT-B and GPT-L, respectively.

4.3. Ablation Study

We investigate the effect of the number of clusters k and watermark strength δ on the TPR in Fig. 3, as well as the effect of k , δ and green token fraction γ on the FID in Fig. 4. Additional ablation details are reported in Section B in the Supplementary Material.

Effect of the number of clusters. We tested extremely low numbers of clusters (down to 8) in order to study their effects on image degradation and watermark robustness. From the TPR plots in Figure 3, we can see that generally, a lower number of clusters results in higher robustness to various perturbations. This matches our expectations since fewer and larger clusters result in the reconstructed tokens becoming less likely to leave their original token’s cluster after stronger perturbations. However, we also see a higher variance across different prefixes. The FID reported in Figure 4 indicate that with a number of clusters lower than 64, there is a significant drop in generation quality.

Effect of green token fraction γ and penalty δ . Regarding image quality (Fig. 4), we observe that $\gamma = 0.25$ results in worse FID compared to $\gamma = 0.5$, especially when combined with a stronger penalty at extremely low numbers of clusters. On the other hand, a higher penalty value δ always results in significantly higher watermark robustness. A lower green set fraction $\gamma = 0.25$ results in significantly higher robustness. This can be seen in Figure 9 in the Supplementary Material.

Effect of hash prefix. The prefix κ used to seed the pseudo-random number generator for green set generation can have a significant impact on robustness. As shown in Figure 3, lower cluster counts exhibit this effect more strongly, with noticeably higher variance.

To investigate this, we compare the impact of the two hash prefixes which show the best ($\kappa = 5$) and worst ($\kappa = 1$) empirical performance on the green tokens ratio recovered from 2,000 watermarked and unwatermarked images for an extremely low number of clusters $k = 8$. We use

¹<https://github.com/ShieldMnt/invisible-watermark>

²<https://github.com/adobe/trustmark>

³https://github.com/facebookresearch/ssl_watermarking

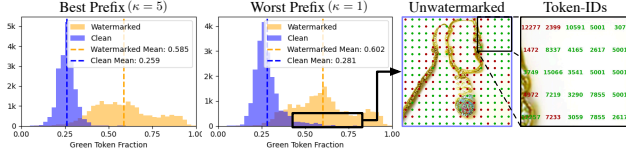


Figure 5. Distribution of green token ratios for watermarked and unwatermarked images across two hashing prefixes κ with a very low number of clusters $k = 8$. Poorly performing hash prefixes suffer from large uniform regions in unwatermarked images, which produce repetitive token bigrams that are spuriously counted as green.

LlamaGen (GPT-B) without the cluster/token-classifier, and set $\delta = 5$ and $\gamma = 0.25$. Watermarked images are distorted with the strong perturbation set B to increase the overlap of the two distributions. This is shown in Figure 5. Ideally, the green token ratio distribution of unwatermarked images should follow a Gaussian centered around $\gamma = 0.25$. While $\kappa = 5$ closely approximates this, $\kappa = 1$ exhibits a heavy right tail that ultimately reduces separability.

This is due to the presence of unwatermarked images with large white areas. For some prefixes, green cluster patterns include those transitioning from a flat white area to a nearly identical area (as shown in Fig. 5 on the right). This inflates the green token ratio in unwatermarked images and triggers false positives. Lower cluster counts make this behaviour more likely.

Prefix tuning is an empirical method to prevent such assignments and can be used to optimize performance during deployment; we leave the exploration of potentially better solutions to future work.

5. Related Work

The integration of watermarks into AI-generated images can be realized in different ways. Post-processing techniques [1, 8, 22, 30, 45, 47] apply watermarking to existing images in a post-hoc manner. Another possibility is to integrate watermarking directly during the generation process. Various techniques have been proposed for text-to-image diffusion models [6, 7, 11, 13, 37, 41]. For instance, the decoder of a latent diffusion model (LDM) can be fine-tuned to always create watermarked images from the latents [7, 11, 40]. Alternatively, inversion-based semantic watermarking modifies the initial latent noise during generation to embed a pattern, which is then recovered through the inversion of the denoising process [6, 13, 37, 41]. Since the initial latent is modified, the watermark becomes an inherent part of the generated image. In-generation (semantic) watermarking thus exploits the fact that there is a myriad of different ways to generate an image that conforms to user specifications (e.g. the prompt) by partly fixing randomness in the generation procedure. In this work, we inte-

grate watermarking during generation by biasing the token sampling for AR models for images.

Concurrently with our work, some recent works have also investigated token-level watermarking for AR image models [16, 17, 33, 39]. Wu et al. [39] propose a different clustering-based approach evaluated on a different model (Emu-3), and only provides a limited robustness study by evaluating robustness to l_2 and l_∞ noise. Tong et al. [33] (IndexMark) compute pairs of similar tokens but assign one token in the pair to the green set and the other one to the red set. We assign similar tokens to the same set. They also train an index reconstruction model to improve token reconstruction accuracy. Similarly to Tong et al. [33], Jovanović et al. [16] (WMAR) also train a token reconstruction model. However, they also consider data augmentation with perturbations during training to improve detection robustness. In contrast, our base clustering approach is training-free, simple to use, and improves robustness while preserving image quality. Our token and cluster classifiers are similar to the trained token reconstructors of IndexMark and WMAR, with a few differences. WMAR also fine-tunes the VAE decoder, which complicates the training process and further affects the appearance of generated images. In contrast, our training-based approach is simpler to train and does not require updating the decoder.

Separate from the detection of AI-generated content, Wang et al. [36] use AR models to embed a hidden image into a given cover image in a post-hoc fashion. Finally, related to our cluster-based watermarking methods are semantic watermarking methods for LLMs (e.g., Ren et al. [26]), developed to resist synonym and paraphrasing attacks by grouping tokens with similar embeddings.

6. Conclusion

In this work, we investigate clustering as a promising solution to address the poor robustness of KGW-style watermarking in autoregressive models for image generation. We thoroughly ablate various design choices across multiple models. The results indicate that clustering is effective in improving robustness against common perturbations and attacks. In fact, it significantly boosts watermark detection, except in the most challenging and destructive settings, namely strong salt-and-pepper noise as well as blurring. In addition to clustering, which is training-free, this work also investigates training token or cluster predictors in combination with clustering. The results indicate that clustering also boosts robustness in this setting, outperforming established post-hoc watermarking baselines and concurrent works on watermarking AR image models. This work provides a robust alternative for the detection of AI-generated content with low verification runtime overhead, contributing to trustworthy deployment of generative AI.

Acknowledgements

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CASA – 390781972 and by the Ministry of Culture and Science of North Rhine-Westphalia as part of the Lamarr Fellow Network.

References

- [1] Ali Al-Haj. Combined dwt-dct digital image watermarking. *Journal of computer science*, 3(9):740–746, 2007. 8
- [2] Tu Bui, Shruti Agarwal, and John Collomosse. Trustmark: Robust watermarking and watermark removal for arbitrary resolution images. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 5, 11
- [3] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2
- [4] Huiwen Chang, Han Zhang, Jarred Barber, Aaron Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Patrick Murphy, William T. Freeman, Michael Rubinstein, Yuanzhen Li, and Dilip Krishnan. Muse: Text-to-image generation via masked generative transformers. In *Proc. of Int. Conference on Machine Learning (ICML)*. PMLR, 2023. 1, 2
- [5] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. In *Proc. of the Annual Conference on Learning Theory (COLT)*, 2024. 2
- [6] Hai Ci, Pei Yang, Yiren Song, and Mike Zheng Shou. RingID: Rethinking tree-ring watermarking for enhanced multi-key identification. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2024. 8, 15
- [7] Hai Ci, Yiren Song, Pei Yang, Jinheng Xie, and Mike Zheng Shou. WMAAdapter: Adding WaterMark control to latent diffusion models. In *Proc. of Int. Conference on Machine Learning (ICML)*, 2025. 8
- [8] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital Watermarking and Steganography*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 edition, 2007. 8
- [9] Lijie Fan, Tianhong Li, Siyang Qin, Yuanzhen Li, Chen Sun, Michael Rubinstein, Deqing Sun, Kaiming He, and Yonglong Tian. Fluid: Scaling autoregressive text-to-image generative models with continuous tokens. In *International Conference on Learning Representations (ICLR)*, 2025. 1, 2
- [10] Pierre Fernandez, Alexandre Sablayrolles, Teddy Furon, Hervé Jégou, and Matthijs Douze. Watermarking images in self-supervised latent spaces. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022. 5, 11
- [11] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 8
- [12] Pierre Fernandez, Tomáš Souček, Nikola Jovanović, Hady Elsahar, Sylvestre-Alvise Rebuffi, Valeriu Lacatusu, Tuan Tran, and Alexandre Mourachko. Geometric image synchronization with deep watermarking. *arXiv: 2509.15208*, 2025. 5, 14
- [13] Sam Gunn, Xuandong Zhao, and Dawn Song. An undetectable watermark for generative image models. In *International Conference on Learning Representations (ICLR)*, 2025. 5, 8, 15
- [14] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021. 14
- [15] Anubhav Jain, Yuya Kobayashi, Naoki Murata, Yuhta Takida, Takashi Shibuya, Yuki Mitsufuji, Niv Cohen, Nasir Memon, and Julian Togelius. Forging and removing latent-noise diffusion watermarks using a single image. *arXiv: 2504.20111*, 2025. 15
- [16] Nikola Jovanović, Ismail Labiad, Tomas Soucek, Martin Vechev, and Pierre Fernandez. Watermarking autoregressive image generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 2, 5, 6, 8, 11, 15
- [17] Louis Kerner, Michel Meintz, Bihe Zhao, Franziska Boenisch, and Adam Dziedzic. Bitmark for infinity: Watermarking bitwise autoregressive image generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 8
- [18] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *Proc. of Int. Conference on Machine Learning (ICML)*, 2023. 1, 2
- [19] 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 *International Conference on Learning Representations (ICLR)*, 2024.
- [20] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research (TMLR)*, 2024. 2
- [21] Hermann Kumbong, Xian Liu, Tsung-Yi Lin, Ming-Yu Liu, Xihui Liu, Ziwei Liu, Daniel Y. Fu, Christopher Re, and David W. Romero. Hmar: Efficient hierarchical masked auto-regressive image generation. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 1, 2
- [22] Shilin Lu, Zihan Zhou, Jiayou Lu, Yuanzhi Zhu, and Adams Wai-Kin Kong. Robust watermarking using generative priors against image editing: From benchmarking to advances. In *International Conference on Learning Representations (ICLR)*, 2025. 8
- [23] Nahema Marchal, Rachel Xu, Rasmi Elasmr, Iason Gabriel, Beth Goldberg, and William Isaac. Generative ai misuse: A taxonomy of tactics and insights from real-world data. *arXiv: 2406.13843*, 2024. 1
- [24] Andreas Müller, Denis Lukovnikov, Jonas Thietke, Asja Fischer, and Erwin Quiring. Black-box forgery attacks on se-

- mantic watermarks for diffusion models. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 15
- [25] K. A. Navas, Mathews Cheriyan Ajay, M. Lekshmi, Tampy S. Archana, and M. Sasikumar. Dwt-dct-svd based watermarking. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, 2008. 5, 11
- [26] Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. A robust semantics-based watermark for large language model against paraphrasing. In *Findings of the Association for Computational Linguistics: NAACL*, 2024. 2, 8
- [27] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 5, 11
- [28] 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 (IJCV)*, 115 (3):211–252, 2015. 4
- [29] 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: 2406.06525*, 2024. 1, 2, 4, 11
- [30] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 8
- [31] Haotian Tang, Yecheng Wu, Shang Yang, Enze Xie, Junsong Chen, Junyu Chen, Zhuoyang Zhang, Han Cai, Yao Lu, and Song Han. HART: Efficient visual generation with hybrid autoregressive transformer. In *International Conference on Learning Representations (ICLR)*, 2025. 1, 2
- [32] Keyu Tian, Yi Jiang, Zehuan Yuan, BINGYUE PENG, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 1, 2
- [33] Yu Tong, Zihao Pan, Shuai Yang, and Kaiyang Zhou. Training-free watermarking for autoregressive image generation. *arXiv: 2505.14673*, 2025. 2, 5, 6, 8, 11, 15
- [34] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiyang Yu, Yingli Zhao, Yulong Ao, Xuebin Min, Tao Li, Boya Wu, Bo Zhao, Bowen Zhang, Liangdong Wang, Guang Liu, Zheqi He, Xi Yang, Jingjing Liu, Yonghua Lin, Tiejun Huang, and Zhongyuan Wang. Emu3: Next-token prediction is all you need. *arXiv: 2409.18869*, 2024. 1, 2
- [35] Yuqing Wang, Zhijie Lin, Yao Teng, Yuanzhi Zhu, Shuhuai Ren, Jiashi Feng, and Xihui Liu. Bridging continuous and discrete tokens for autoregressive visual generation. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 1, 2
- [36] Ziyi Wang, Songbai Tan, Gang Xu, Xuerui Qiu, Hongbin Xu, Xin Meng, Ming Li, and Fei Richard Yu. Safe-VAR: Safe visual autoregressive model for text-to-image generative watermarking. *arXiv: 2503.11324*, 2025. 8
- [37] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-Ring watermarks: Invisible fingerprints for diffusion images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 1, 5, 7, 8, 15
- [38] Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. A resilient and accessible distribution-preserving watermark for large language models. In *Proc. of Int. Conference on Machine Learning (ICML)*, 2024. 2
- [39] Yihan Wu, Xuehao Cui, Ruibo Chen, Georgios Milis, and Heng Huang. A watermark for auto-regressive image generation models. *arXiv: 2506.11371*, 2025. 8
- [40] Cheng Xiong, Chuan Qin, Guorui Feng, and Xinpeng Zhang. Flexible and secure watermarking for latent diffusion model. In *Proc. ACM International Conference on Multimedia (ACM MM)*, 2023. 8
- [41] Zijin Yang, Kai Zeng, Kejiang Chen, Han Fang, Weiming Zhang, and Nenghai Yu. Gaussian Shading: Provable performance-lossless image watermarking for diffusion models. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 5, 8, 15
- [42] KiYoon Yoo, Wonhyuk Ahn, Jiho Jang, and Nojun Kwak. Robust multi-bit natural language watermarking through invariant features. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. 2
- [43] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Guntan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*, 2022. 1, 2
- [44] Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 1, 2, 4, 11
- [45] Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust invisible video watermarking with attention. *arXiv: 1909.01285*, 2019. 5, 8, 11
- [46] Xuandong Zhao, Kexun Zhang, Zihao Su, Saastha Vasan, Ilya Grishchenko, Christopher Kruegel, Giovanni Vigna, Yu-Xiang Wang, and Lei Li. Invisible image watermarks are provably removable using generative AI. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 5, 11
- [47] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. 8