
Watermarking Diffusion Language Models

Thibaud Gloaguen, Robin Staab, Nikola Jovanović, Martin Vechev

ETH Zurich

thibaud.gloaguen@inf.ethz.ch

Abstract

We introduce the first watermark tailored for diffusion language models (DLMs), an emergent LLM paradigm able to generate tokens in arbitrary order, in contrast to standard autoregressive language models (ARLMs) which generate tokens sequentially. While there has been much work in ARLM watermarking, a key challenge when attempting to apply these schemes directly to the DLM setting is that they rely on previously generated tokens, which are not always available with DLM generation. In this work we address this challenge by: (i) applying the watermark in expectation over the context even when some context tokens are yet to be determined, and (ii) promoting tokens which increase the watermark strength when used as context for other tokens. This is accomplished while keeping the watermark detector unchanged. Our experimental evaluation demonstrates that the DLM watermark leads to a >99% true positive rate with minimal quality impact and achieves similar robustness to existing ARLM watermarks, enabling for the first time reliable DLM watermarking. Our code is available [here](#).

1 Introduction

While autoregressive language models (ARLMs) have demonstrated remarkable performance [36, 23, 6], other approaches to language modeling, such as diffusion language models (DLMs), are rapidly catching up, with recent works [35, 53, 30] approaching similar capabilities at a significantly lower cost. Notably, DLMs have higher generation speed, offer built-in error correction, are more controllable, and can be naturally extended to multiple modalities [51]. These advances also increase the risk of misuse, making it paramount to reliably detect text generated by these models. Prior works [27, 29, 10] have proposed the concept of ARLM watermarks: the generated text is augmented with an imperceptible signal that can later be detected to trace its provenance. Such watermarks are already adopted in consumer-facing models [12] and are being advocated for through regulation [16]. Yet, most existing works on language model watermarking rely on hashing mechanisms compatible only with autoregressive generation, highly limiting their application for DLMs. Our work is the first to attempt to overcome this limitation, enabling efficient and reliable watermarking for DLMs.

ARLM Watermarks Watermarks for ARLMs traditionally rely on three key components. The *hashing mechanism* (i) uses the previously generated tokens to seed the *sampling procedure* (ii), which then inserts the watermark signal into the generated text. Critically, this pipeline relies on the previous tokens to have already been generated to compute the hash, an assumption often violated with DLMs. The *watermark detector* (iii) leverages the hashing mechanism to retrieve the seed associated with each token and compute a test statistic to determine whether a given text is watermarked.

Diffusion Language Models DLMs produce a probability distribution over the set of fixed-length sequences containing masked tokens, which, analogous to noise in continuous diffusion models, represent placeholders yet to be generated. Starting from a (partially) masked sequence, DLMs iteratively sample (*unmask*) tokens until the sequence is fully generated. Importantly, unlike autoregressive

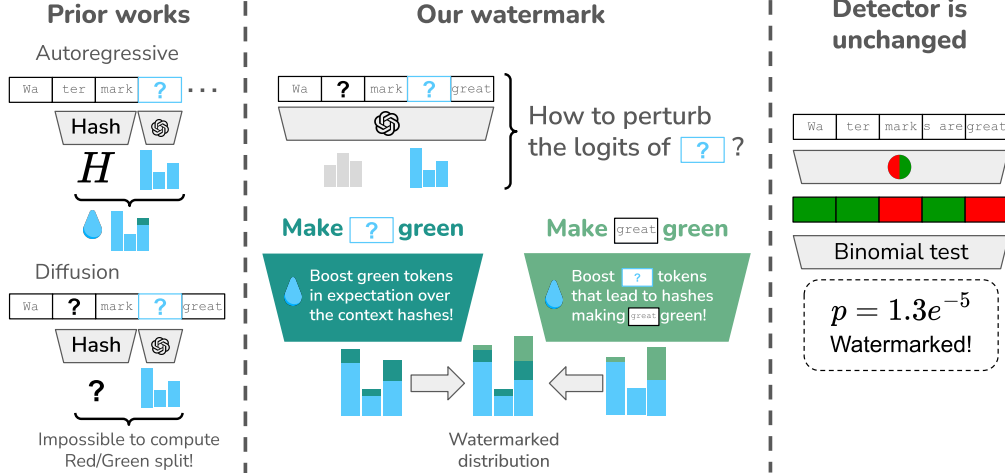


Figure 1: An overview of why current watermarks for ARLMs fall short in the diffusion setting (left), how our watermark operates in this setting (middle) and how our watermark detector works (right).

models, DLMs are not constrained to unmasking tokens in a left-to-right order, making hashing-based ARLM watermarks inapplicable whenever a token without full prior context is unmasked. A natural workaround is to restrict the watermark application to tokens with fully available context. Yet, as we show in Sec. 4, such a naive approach leads to a weak watermark, as barely any tokens fulfill this criterion. This motivates the need for a novel watermarking algorithm tailored to DLMs.

This Work: Watermarking Diffusion LMs In this work, we introduce the first watermarking scheme tailored to DLMs based on the prominent Red-Green watermarks (see Sec. 2), as illustrated in Fig. 1. First, we identify the limitations of autoregressive hashing in most prior works on ARLM watermarks when applied in the diffusion setting (Fig. 1, left). To overcome this limitation, we frame watermarking as a constrained optimization problem (Sec. 3.1). From this optimization problem, we then derive a practical watermarking scheme for DLMs (Sec. 3.2). In Sec. 3.3, we interpret our scheme as a combination of two intuitive components: applying ARLM watermarks in expectation over the context hashes and biasing tokens that lead to hashes making other tokens green (Fig. 1, middle). Our detector is exactly the same as in prior Red-Green watermarks: we compute the color of each token in the sequence and perform a binomial test (Fig. 1 (right)). In our experiments, we show that our watermarking scheme is significantly more detectable than naive baseline adaptations of ARLM watermarks (exceeding 99% TPR at 1% FPR with negligible quality degradation), and is robust to common modifications of the generated text, e.g., substituting words in context (Sec. 4).

Main Contributions Our key contributions are:

- We conduct the first study of watermarking for DLMs, identifying the limitation of ARLM watermarks in the diffusion setting and proposing a watermarking scheme tailored for DLMs.
- We formulate the problem of DLM watermarking as a constrained optimization problem (Sec. 3.1) from which we derive a practical watermarking scheme that leverages existing watermark detectors from prior works (Sec. 3.2).
- We further interpret our watermarking scheme as an extension of existing ARLM watermarks (Sec. 3.3) and demonstrate that, when restricted to the autoregressive case, our optimization formulation recovers exactly ARLM watermarks (Sec. 3.3).
- Through extensive evaluation, we show that our scheme is suitable for practical use, preserves the generated text quality and is robust against common natural text modifications (Sec. 4).

2 Background and Related Work

Language Modeling The goal of language modeling [11, 13] is to train a model θ to learn a probability distribution p_θ over natural language. In particular, given a vocabulary Σ and the true probability distribution p_{data} , the goal is to find θ such that $p_\theta \approx p_{\text{data}}$. With autoregressive modeling,

given a text $\omega \in \Sigma^*$, its probability is factorized sequentially using the (probability) chain rule. While this formulation allows for efficient training [39, 40, 5], it forces ARLMs to generate new tokens sequentially, which fundamentally limits their generation speed and might restrain their capabilities [4]. Recent works [22] have proposed multi-tokens prediction as a training objective to improve model capabilities, but crucially still generate tokens autoregressively.

Diffusion Language Modeling In contrast, in diffusion language modeling [3, 32, 44, 37, 42], the vocabulary Σ is extended with a mask token representing a position where a token is yet to be sampled. Let $\tilde{\Sigma}$ be the vocabulary *including the mask token*. The diffusion process is parameterized by a fixed number of diffusion steps N and a generation length L . At each diffusion step i , the model takes a sequence of tokens $\tilde{\omega}^{(i)} \in (\tilde{\Sigma})^L$ and returns a *factorized* probability distribution $p^{(i)} \in \Delta(\Sigma)^L$ over the set of sequences, where Δ denotes the simplex. Each token in the sequence of length L is then sampled independently from $p^{(i)}$, which returns an intermediary sequence $\omega^{(i+1)} \in \Sigma^L$. This sequence is then masked $\tilde{\omega}^{(i+1)} := \text{Mask}(\omega^{(i+1)}, p^{(i)})$, using a pre-determined stochastic masking procedure $\text{Mask}: \Sigma^L \times \Delta(\Sigma)^L \rightarrow (\tilde{\Sigma})^L$. We say a token is getting unmasked if it was masked in $\tilde{\omega}^{(i)}$ but not in $\tilde{\omega}^{(i+1)}$. This process is iterated N times, returning a final $\omega^{(N+1)} \in \Sigma^L$ which has no mask tokens. Unlike ARLMs, DLMs can, in each step, generate (unmask) multiple tokens and in any order. We evaluate our watermark with multiple unmasking procedure Mask : uniform [3], based on distribution entropy [26], and by block [2].

Large Language Diffusion Models With the release of open Large Language Diffusion Models [35, 53], DLMs have gained significant traction in multiple domains: reasoning [43, 7], chemistry [46], multimodality [51], vision [54] and safety [33]. For traceability, previous works [57] only explored zero-shot detection of DLMs generated text, and highlighted the lack of a reliable watermarking solution in this paradigm.

LLM Watermarks The goal of generative AI watermarks is to ensure content traceability, i.e., the ability to rigorously identify whether a given piece of content has been generated by an AI model. Key challenges for watermarks include balancing their impact on content quality, their strength, and their robustness to edits [47, 38], while ensuring their reliability against false positives. For ARLMs, a range of *generation-time* watermarks [27, 29, 1, 12, 10] have been proposed by building on the autoregressive nature of the underlying models. Prior works have also proposed order-agnostic watermarks, namely Unigram [56] and its extension PatternMark [8], that can be directly applied to DLMs. However, such approaches significantly compromise on watermark security [25, 55]. We nonetheless show in App. B that our approach, tailored for DLMs, outperforms such schemes. Other concerns regarding watermark design include their applicability in the open-source setting [21, 50].

In this work, we study the most popular family of current ARLM watermarks: *Red-Green watermarks* Kirchenbauer et al. [27]. Let $\omega_t \in \Sigma$ denote the token generated by the LM at step t , and k the *context size* parameter. Using a hash of the context $H(\omega_{t-k:t})$, a pseudo-random function partitions the vocabulary Σ into a *green* subset and the remaining *red* subset. The size of the green subset is set to $\gamma|\Sigma|$, with $\gamma \in (0, 1)$ commonly chosen as 0.25 or 0.5. To insert the watermark, each green token in the logits vector gets shifted up by a constant δ , increasing the overall likelihood of sampling a green token. To detect the watermark, given a sequence of tokens $\omega \in \Sigma^*$, we extract a corresponding color sequence and perform a *binomial test* on the number of green tokens.

Image Diffusion Watermarking With the increasing popularity of diffusion models for image generation [14, 24, 34, 45], image diffusion watermarking has been the focus of many works [19, 48, 52]. Yet, all generation-time image diffusion watermarks operate in a continuous space, making them fundamentally inapplicable to the discrete diffusion process of DLMs.

3 Watermarking Diffusion Language Model Generation

In this section, we present our approach to watermarking DLMs. Sec. 3.1 introduces a theoretical framework to guide our DLM watermark design, while Sec. 3.2 addresses practical challenges and proposes an instantiation of our watermark. Lastly, in Sec. 3.3, we see how our watermark naturally extends Red-Green ARLM watermarks: it decomposes into the two terms illustrated in Fig. 1 (middle) and, when restricted to the ARLM case, it instantiates to Red-Green ARLM watermarks.

3.1 Adapting Watermarks to the Diffusion Setting

We propose framing the LM watermarking algorithm as an optimization problem, which in turn allows us to adapt watermarks for autoregressive language models to the diffusion setting.

Naive Approach As alluded to in Sec. 1, a naive approach to adapting Red-Green watermarks in the diffusion setting would be to apply them only to tokens whose context is fully unmasked. Indeed, for such token positions, computing their context hash and thus the watermark distribution is possible. However, this means that we cannot apply the watermark (e.g., boost the probability of green tokens) to any token that is unmasked before its respective context. Ultimately, this would result in only a handful of green tokens which, as we show in Sec. 4.1, leads to a low true positive detection rate. Importantly, we find that a watermark tailored for DLMs needs to operate directly over the *distribution* of context hashes, enabling it to watermark all tokens regardless of the unmasking order.

Goal of Red-Green Watermarks Given a sequence ω , the detector computes the proportion of green tokens $\hat{\gamma}(\omega)$ within that sequence and conducts a binomial test. If $\hat{\gamma}(\omega)$ significantly exceeds the baseline ratio of green tokens expected under the null hypothesis γ , the sequence ω is identified as watermarked. Thus, the goal of the watermarking algorithm is to modify the sampling procedure such that each generated sequence ω has a high ratio $\hat{\gamma}(\omega)$, while preserving the model’s utility.

Watermarking as a Constrained Optimization Problem A watermarked DLM should preferentially sample token sequences that maximize the ratio $\hat{\gamma}(\omega)$. This naturally leads to framing the problem of DLM watermarking as a constrained optimization task: we aim to maximize the expectation of the green token ratio over the generated sequence while minimizing the impact on model performance. We show in Sec. 3.2 that our proposed optimization framework abstracts the complexity of DLM watermarks into a practical watermarking algorithm, illustrated in Fig. 1 (middle).

Recall that Σ is the model vocabulary, and $\tilde{\Sigma}$ is the vocabulary extended with the mask token. Let $\omega \in \Sigma^L$ be an unmasked sequence of tokens. For every token position t , we introduce a hash function $H_t : \Sigma^L \rightarrow \mathcal{H}$ that returns the hash of the context at position t , with $\mathcal{H}$ the *finite* set of possible hash values. We introduce the *global* binary green list matrix $G \in \{0, 1\}^{\mathcal{H} \times \Sigma}$ which, given the context hash and a token, returns its color. We formalize the *green ratio function* $\hat{\gamma} : \Sigma^L \rightarrow [0, 1]$ as $\hat{\gamma}(\omega) = \frac{1}{L} \sum_{t=1}^L G_{H_t(\omega), \omega_t}$, i.e., the color $G_{H_t(\omega), \omega_t}$ of each token ω_t averaged over the sequence ω .

Let $\tilde{\omega} \in (\tilde{\Sigma})^L$ be a (partially) masked sequence and $p(\tilde{\omega}) \in \Delta(\Sigma)^L$ the corresponding *factorized* probability distribution over the set of sequences returned by the DLM forward pass. Given $t \in [1, \dots, L]$, p_t is the probability over Σ at position t . The goal of the watermarking algorithm is to distort the factorized probability distribution $p(\tilde{\omega})$ into a factorized probability distribution $q(\tilde{\omega})$ that maximizes the expected green ratio of sequences sampled according to $q(\tilde{\omega})$. We have to solve,

$$q^* = \arg \max_{q \in \Delta(\Sigma)^L} \mathbb{E}_{\Omega \sim q}[\hat{\gamma}(\Omega)], \text{ subject to } \forall t \in [1, \dots, L], \text{KL}(q_t, p_t(\tilde{\omega})) \leq \varepsilon. \quad (1)$$

In Eq. (1), the KL constraint with $\varepsilon > 0$ is used as a proxy for controlling impact on quality. We now explain how to exactly compute the expectation from Eq. (1). For any $q \in \Delta(\Sigma)^L$, to compute the expectation, given $\Omega \sim q$ (i.e., Ω is the random variable representing sequences distributed according to the factorized probability distribution q), we need to know the distribution of Ω_t and $H_t(\Omega)$. The distribution of Ω_t is by definition q_t . Computing the distribution of $H_t(\Omega)$, i.e., the distribution of the context hashes, is more challenging. We model it for every token position t as a function of q , $h_t : \Delta(\Sigma)^L \rightarrow \Delta(\mathcal{H})$, and defer instantiations to Sec. 3.2. We can now unroll the expectation,

$$\forall q \in \Delta(\Sigma)^L, \mathbb{E}_{\Omega \sim q}[\hat{\gamma}(\Omega)] = \frac{1}{L} \sum_{t=1}^L \mathbb{E}_{\Omega \sim q}[G_{H_t(\Omega), \Omega_t}] = \frac{1}{L} \sum_{t=1}^L h_t(q)^\top \cdot G \cdot q_t =: \frac{1}{L} J(q). \quad (2)$$

When expanding the dot products, each element is the color of a (token, context hash) tuple weighted by its probability. For clarity, we refer to J as the *energy* function. The watermark is equivalent to

$$q^* = \arg \max_{q \in \Delta(\Sigma)^L} J(q), \text{ subject to } \forall t \in [1, \dots, L], \text{KL}(q_t, p_t(\tilde{\omega})) \leq \varepsilon. \quad (3)$$

Solving Eq. (3) is challenging due to the KL constraint. Without the constraint, a greedy algorithm would suffice. In Theorem 3.1, we provide an implicit solution to Eq. (3), with the proof in App. I.

Theorem 3.1. Given $p \in \Delta(\Sigma)^L$ and J defined in Eq. (2), there exists $\delta \in \mathbb{R}^L$ such that

$$\forall t \in [1, \dots, L], q_t^* \propto p_t \exp(\delta_t \alpha_t(q^*)) \quad (4)$$

with $\alpha_t(q) = \nabla_{q_t} J(q)$. Moreover, for all $t \in [1, \dots, L]$, δ_t is the unique solution to $KL(q_t^*, p_t) = \varepsilon$.

This result means that the distribution q^* is optimal with respect to our optimization problem, i.e., it is the optimal way to turn any distribution p provided by our DLM into a distribution that maximizes the expected green ratio while enforcing the KL constraint. Importantly, it has a fairly simple form. If we take Eq. (4) in the logits space, we see that the optimal solution corresponds to adding $\delta_t \alpha_t(q^*)$ to the logits vector—a similar distortion to the Red-Green ARLM watermark. In Sec. 3.3, we provide an intuitive explanation to how our watermark works. We show that $\delta_t \alpha_t(q^*)$ can be decomposed into two components: one that makes the token at position t green and one that makes tokens which have the token at t in their context green, as illustrated in Fig. 1 (middle).

3.2 Our Watermark Algorithm for Diffusion LMs

In this section, we instantiate the theoretical derivation from Sec. 3.1 to derive a practical watermark algorithm. First, we explain how to solve Eq. (4) and find δ . We then detail instantiations of common hash functions and propose a corresponding practical watermark algorithm.

Solving the Functional Equation Eq. (4) defines a functional equation with q^* and δ as the unknowns. We propose using a fixed-point iterative approach of $f: q \mapsto p \exp(\delta \alpha(q)) / Z(q)$, which gives $q^* = f(q^*)$. At each step i , given a q^i , we first find δ for this q^i and then compute $q^{(i+1)} = f(q^{(i)})$. We find in App. C.1 that in practice a single iteration already yields a strong watermark, and that, despite the lack of theoretical guarantees of convergence to q^* , increasing the number of iterations indeed slightly improves watermark strength.

To find δ , we can solve for all t the equation $KL(q_t^*, p_t) = \varepsilon$ using bisection, as Theorem 3.1 guarantees the existence and uniqueness of δ . A simpler relaxation of our framework, closer to the parameterization of Red-Green ARLM watermarks, is to directly parameterize the optimization problem, and thus the watermark algorithm, by a constant $\delta \in \mathbb{R}$ instead of ε . We refer to using ε as ε -parameterization and using δ as δ -parameterization. We compare both approaches in App. C.1.

Instantiating the Hash We now instantiate the hash H and the hash distribution h from Sec. 3.1. Similarly to Kirchenbauer et al. [28], we explore two local hash functions, *SumHash* and *MinHash* (we provide a detailed comparison with the formulation of Kirchenbauer et al. [28] in App. G). As alluded to in Sec. 3.1, computing h naively is challenging and requires $O(\Sigma^L)$ operations. Thanks to the specific local structure of the considered hash functions detailed below, we show that the corresponding hash distributions can be computed efficiently through algebraic manipulation.

SumHash sums the surrounding tokens. Let $k \in \mathbb{N}$ be the context size, and $\mathcal{C} = \{c_1, \dots, c_k\}$ the set of positions that define the context. For instance, $\mathcal{C} = \{-1, 1\}$ means that, for every token position, the context is comprised of the previous token and the next token. In contrast to the ARLM setting, the context is no longer restricted to preceding tokens but can also include tokens after t . We define

$$\forall \omega \in \Sigma^L, \forall t \in [1, \dots, L], H^{\text{SumHash}}(\omega)_t = \sum_{i \in \mathcal{C}} \omega_{t+i}. \quad (5)$$

We can derive an analytical formula for the probability distribution over the hashes h^{SumHash} as

$$\forall p \in \Delta(\Sigma)^L, h_t^{\text{SumHash}}(p)_s = \sum_{\substack{u_1, \dots, u_k \in \Sigma^k \\ \text{SumHash}(u_1, \dots, u_k) = s}} \prod_{j=1}^k p_{t+c_j}(u_j) = (p_{t+c_1} * \dots * p_{t+c_k})_s, \quad (6)$$

where $*$ is the convolution product and $s \in \mathcal{H}$. Using the Fast Fourier Transform to compute the convolution product, computing h^{SumHash} takes $O(|\mathcal{C}| |\Sigma| \log |\Sigma|)$ operations.

With *MinHash*, the hash corresponds to the token id of the minimum of the surrounding tokens, i.e.,

$$\forall \omega \in \Sigma^L, \forall t \in [1, \dots, L], H^{\text{MinHash}}(\omega)_t = \min_{i \in \mathcal{C}} \sigma(\omega_{t+i}), \quad (7)$$

with σ a random permutation, to ensure randomness in the token id ordering. For $t \in [1, \dots, L]$, $u \in \Sigma$, let $p_t^\sigma(u) = p_t(\sigma^{-1}(u))$. We have for $s \in \mathcal{H}$ and $p \in \Delta(\Sigma)^L$,

$$h_t^{MinHash}(p)_s = \sum_{\substack{u_1, \dots, u_k \in \Sigma^k \\ MinHash(u_1, \dots, u_k) = s}} p_{t+c_1}(u_1) \dots p_{t+c_k}(u_k) = A_t(s+1) - A_t(s), \quad (8)$$

where $A_t(s) := \prod_{i \in \mathcal{C}} \sum_{u=s}^{|\Sigma|} p_{t+i}^\sigma(u)$. Hence, computing $h^{MinHash}$ requires $O(|\mathcal{C}||\Sigma|)$ operations.

Our Watermark Based on the above methods, we now present our watermarking algorithm (Algorithm 1). Specifically, at each diffusion step i , given the distribution of the DLM $p(\tilde{\omega}^{(i)})$, we first compute the hash distribution h (line 3) using Eq. (6) or Eq. (8) and subsequently derive the corresponding energy function J (line 4) as per Eq. (2). For practical reasons, for each token position t , we only use the top- k h_t and p_t (we ablate over practical choices of k in App. C). For each token position, we then calculate the gradient of the energy function with respect to the token’s probability distribution (line 6), and apply an exponential tilt proportional to this gradient scaled by the strength parameter δ (line 7), resulting in the final watermarked distribution (line 8), as per Eq. (4). The distribution is further refined by iterating the tilting procedure.

For detection, we use the same detector as Red-Green ARLM watermarks: given ω , we compute each token’s color and perform a binomial test on the number of green tokens after deduplication of (context hash, token) pairs [18]. Algorithm 1 time complexity, with SumHash, is $O(nL|\mathcal{C}||\Sigma| \log |\Sigma|)$. Yet $n, |\mathcal{C}|$ are small which results in minimal generation overhead (App. H.1).

Algorithm 1 Watermark for Diffusion LMs

Require: DLM Probabilities p , green matrix G , number of iterations n , watermark strength δ , generated sequence length L .

```

1:  $p^{(0)} \leftarrow p$ 
2: for  $i$  from 0 to  $n - 1$  do
3:    $h \leftarrow \text{HashProbabilities}(p^{(i)})$ 
4:    $J \leftarrow \sum_{t=1}^L h_t^\top \cdot G \cdot p_t^{(i)} \triangleright \text{Using top-}k$ 
5:   for  $t$  from 1 to  $L$  do
6:      $\alpha_t \leftarrow \nabla_{p_t} J$ 
7:      $p_t^{(i+1)} \leftarrow p_t^{(i)} \exp(\alpha_t \delta)$ 
8:      $p_t^{(i+1)} \leftarrow \text{Normalize}(p_t^{(i+1)})$ 
9:   end for
10: end for
11: return  $p^{(n)}$ 

```

3.3 Interpreting Our Diffusion LM Watermark

We now provide an intuitive explanation on how our watermark distorts the original model probability distribution p into a watermarked probability distribution q^* . We perform an explicit computation with SumHash to interpret the components of our watermark. Additionally, in App. E, we show that instantiating our optimization problem (Eq. (1)) for the ARLM case preserves only the boost component and corresponds exactly to the Red-Green ARLM watermark.

Interpreting Our Watermark We analyze our watermarking algorithm with SumHash and $\mathcal{C} = \{-1\}$. In this setting, the energy function simplifies to

$$J(p) = \sum_{t=1}^L p_{t-1}^\top \cdot G \cdot p_t, \quad (9)$$

which in turn yields, using a fixed δ and at the first step of the fixed-point iteration,

$$\forall t \in [1, \dots, L], q_t^* \propto p_t \underbrace{\exp(\delta G^\top p_{t-1})}_{\text{expectation boost}} \underbrace{\exp(\delta G p_{t+1})}_{\text{predictive bias}}. \quad (10)$$

We now recover the two components illustrated in Fig. 1. The first term, $\exp(\delta G^\top p_{t-1})$, is the Red-Green watermark boost taken in expectation over the distribution of the context, in this case the previous token. If the mass of the distribution p_{t-1} is concentrated on a single token ω_{t-1} , we recover exactly the Red-Green watermark boost $G_{\omega_{t-1},:}$ and add δ to all green token logits. We label this first component the *expectation boost*. The second term, $\exp(\delta G p_{t+1})$, is also intuitive: it favors sampling tokens that lead to hashes for which the next tokens are more likely to be green. We call this the *predictive bias*. Overall, this means that our watermark for diffusion models constitutes a natural extension of Red-Green watermarks: it both applies the Red-Green boost by δ in expectation over the context hashes and additionally leverages the possibility of sampling hashes that lead to more green tokens at other positions. We extend this interpretation in App. I.2 for any hash function and context.

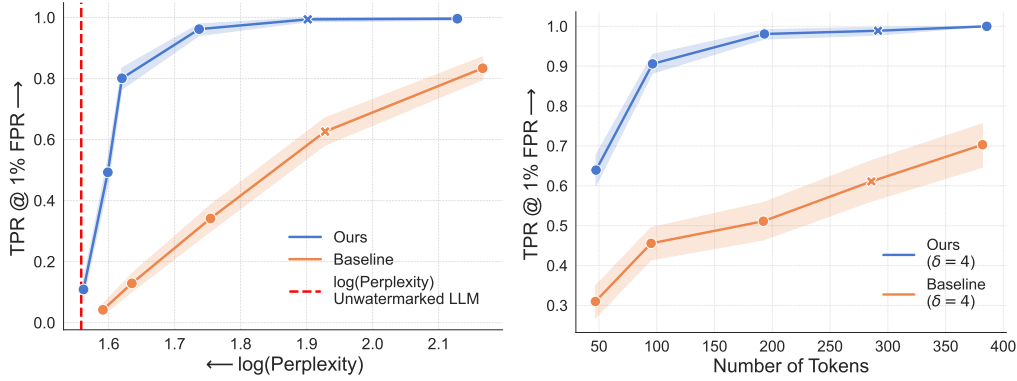


Figure 2: **Detection Performance of Our Approach** (Left) We compare the trade-off between watermark detectability (TPR@1) and text quality (log PPL) of our approach and the baseline for different values of the watermark strength parameter δ and sequences of, on average, 275 tokens. (Right) For $\delta = 4$, we compare watermark detectability (TPR@1) between our approach and the baseline as a function of text length. Responses are generated by LLADA-8B with temperature 0.5 and 600 prompts from WATERBENCH. Crosses represent shared parameters between both figures.

4 Evaluation

In Sec. 4.1, we compare our watermarking approach for DLMs with baselines derived from autoregressive watermarks. Sec. 4.2 focuses on the robustness of our watermark against various text modifications. We include further ablations (on the watermark hyperparameters and the diffusion process parameters) and baselines (AAR and KTH) in App. C, and extended results in App. J.

Experimental Setup To evaluate watermark detectability, we follow the approach of WATERBENCH [47]. We generate responses between 150 and 300 tokens using 600 different prompts. For our watermark, we use the SumHash hashing scheme, δ -parameterization, a single iteration for fixed-point convergence, and top-k of 50. For the DLMs, we use LLADA-8B and DREAM-7B with a temperature of 0.5 and a random remasking strategy. We defer additional details of our experimental setup to App. A.1 and ablate on most components in App. C, D and J.

4.1 Watermark Strength and Quality Evaluation

The key challenge for watermarking DLMs highlighted in Sec. 3 is that, when generating a given token, the context used to seed the watermark may not be known. To determine whether our approach from Sec. 3.2 overcomes this challenge, we propose as a baseline the naive approach described in Sec. 3.1. When generating the token at position t , if the context is already set, we apply the ARLM watermark. Otherwise, we do not watermark the token probability distribution at position t .

Strong Watermark Detectability We first evaluate watermark detectability using the previous token as context ($\mathcal{C} = \{-1\}$) with LLADA-8B and DREAM-7B. For LLADA-8B, Fig. 2 (left) shows that our approach provides significantly better detectability than the baseline given the same impact on quality. In App. C (Fig. 6), we find that the improvement is especially noticeable for higher entropy remasking strategies (e.g., the seemingly random remasking strategies). In the limit, when using autoregressive remasking, while there is still a slight improvement over the baseline, the difference is minimal. Fig. 2 (right) shows that, with our approach and $\delta = 4$, detectability increases quickly with the length of the generated sequence. Importantly, given comparable quality, our approach at ≈ 50 tokens has the same detectability as the baseline at ≈ 350 .

Table 1 shows that the same conclusions hold for DREAM-7B and for different choices of the context $\mathcal{C}$: our watermark provides significantly better detectability than the baselines given a similar impact on quality (both with log perplexity and GPT4o-as-a-judge score). To reach 99% TPR@1 with a reasonable sequence length, practitioners should either use $\delta = 4$ with a single token context, or $\delta = 5$ for larger contexts. Additionally, we find in App. D that similar results also hold for infilling tasks. We apply our watermark using DREAMON-V0-7B, a DLM with a diffusion process tailored for infilling, and consistently reach 99% TPR@1 for the same hyperparameters.

Table 1: **Detection Performance** We compare the detectability of our watermark (TPR@1) for different contexts. The quality distortion (log PPL and GPT4 scores) between the baseline and our approach is similar, and minimal compared to the unwatermarked model, yet our approach consistently reaches 99% TPR@1. Scores are averaged over 600 responses generated at temperature 0.5. The average response length for LLADA-8B is 275 and 213 for DREAM-7B.

Model	Type	$\mathcal{C} = \{-1\}, \delta = 4$			$\mathcal{C} = \{-1, 1\}, \delta = 5$			$\mathcal{C} = \{-2, -1\}, \delta = 5$		
		TPR@1	log(PPL)	GPT4	TPR@1	log(PPL)	GPT4	TPR@1	log(PPL)	GPT4
LLADA-8B	Unwatermarked	0.00	1.56	8.95	0.00	1.56	8.95	0.00	1.56	8.95
	Baseline	0.63	1.93	8.48	0.69	1.86	8.51	0.83	1.94	8.37
	Ours	0.99	1.90	8.43	0.99	1.80	8.60	0.99	1.80	8.59
DREAM-7B	Unwatermarked	0.00	1.94	8.45	0.00	1.94	8.45	0.00	1.94	8.45
	Baseline	0.49	2.27	7.95	0.74	2.18	7.94	0.70	2.23	8.20
	Ours	0.99	2.32	7.76	0.99	2.18	7.85	0.99	2.15	7.90

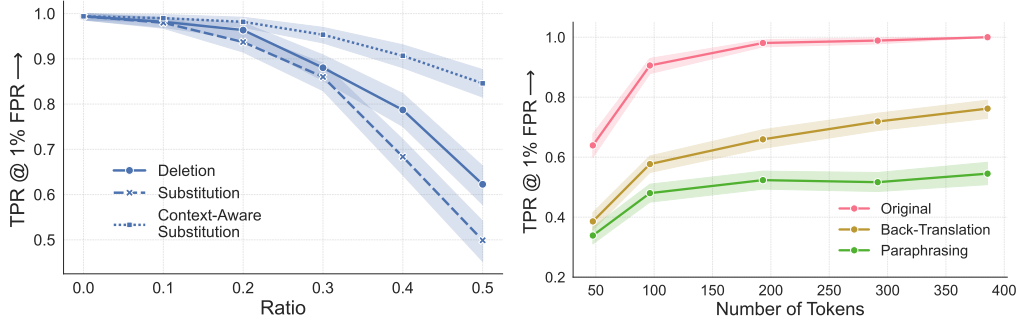


Figure 3: **Robustness Evaluation of Our Watermark** (Left) We measure the detectability of our watermark (TPR@1) against an increasing percentage of local modifications, using responses generated from LLADA-8B with an average length of 275 tokens. (Right) For stronger adversaries, we measure the detectability of our watermark (TPR@1) with respect to the length of the sequence. For both figures, we use $\delta = 4$ and the previous token as context ($\mathcal{C} = \{-1\}$).

4.2 Robustness Evaluation

Using the toolkit provided in Pan et al. [38], before running the watermark detection, we apply to each text one of the following transformations: word deletion, word substitution, context-aware word substitution (using BERT [13]), paraphrasing (using GPT5-MINI), or back-translation (from English to Standard Chinese and then back to English, using GPT5-NANO as a translator).

Robustness to Local Modifications Fig. 3 (left) shows that, for the recommended parameters ($\mathcal{C} = \{-1\}, \delta = 4$) and sequence of length 300 tokens, the watermark retains strong detectability until up to 30% of the sequence edited for word deletion and substitution. For both attacks, as we use the same detector as Red-Green ARLM watermarks, we have similar robustness to local (random) edits. Importantly, our watermark is significantly more robust to context-aware substitution (i.e., substituting words based on the context). This can be explained using our interpretation from Sec. 3.3: by applying our watermark in expectation over the context hashes, we get as a side-effect that all likely variations of the generated sequence are also (to some extent) watermarked.

Robustness Against Stronger Adversaries In prior works, paraphrasing and back-translation (i.e., translating a given text back and forth) are pointed out as strong adversaries to LLM watermarks. Similarly to ARLM watermarks, Fig. 3 (right) shows that both attacks hurt watermark detectability. Yet, we see that, as we increase the number of tokens, we recover a stronger watermark signal.

5 Conclusion

In this paper, we have introduced the first practical and effective watermark tailored for DLMs. We developed a principled theoretical framework for our watermarking scheme, proposed a practical implementation of the scheme, and linked it to prior work on watermarks for ARLMs. Our results show that our watermark is effective (more than 99% TPR@1 with low impact on text quality) and robust.

References

- [1] Scott Aaronson. Watermarking of large language models. In *Workshop on Large Language Models and Transformers, Simons Institute, UC Berkeley*, 2023.
- [2] Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.
- [3] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.
- [4] Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on "a is b" fail to learn "b is a". *arXiv preprint arXiv:2309.12288*, 2023.
- [5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [6] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv*, 2023.
- [7] Edoardo Cetin, Tianyu Zhao, and Yujin Tang. Large language models to diffusion finetuning. 2025.
- [8] Ruibo Chen, Yihan Wu, Yanshuo Chen, Chenxi Liu, Junfeng Guo, and Heng Huang. A watermark for order-agnostic language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=Nlm3Xf0W9S>.
- [9] Zhiyu Chen, Wenhui Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, et al. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3697–3711, 2021.
- [10] Miranda Christ, Sam Gunn, Tal Malkin, and Mariana Raykova. Provably robust watermarks for open-source language models. *arXiv*, 2024.
- [11] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28, 2015.
- [12] Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, 2024.
- [13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL <https://arxiv.org/abs/1810.04805>.
- [14] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.
- [15] Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaFarm: A simulation framework for methods that learn from human feedback, 2024. URL <https://arxiv.org/abs/2305.14387>.
- [16] EU Council. Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts - analysis of the final compromise text with a view to agreement. 2024.

- [17] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: Long form question answering. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3558–3567, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1346. URL <https://aclanthology.org/P19-1346/>.
- [18] Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. Three bricks to consolidate watermarks for large language models, 2023. URL <https://arxiv.org/abs/2308.00113>.
- [19] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22466–22477, 2023.
- [20] Thibaud Gloaguen, Nikola Jovanović, Robin Staab, and Martin Vechev. Black-box detection of language model watermarks. In *The Thirteenth International Conference on Learning Representations*.
- [21] Thibaud Gloaguen, Nikola Jovanović, Robin Staab, and Martin Vechev. Towards watermarking of open-source llms, 2025. URL <https://arxiv.org/abs/2502.10525>.
- [22] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.
- [23] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kam-badur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shao-liang Nie, Sharan Narang, Sharath Rapparthi, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collobert, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet,

Virginie Do, Vish Vogeti, Vitor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuwei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymmer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkan Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippas Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Sweet, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojuan Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.

- [24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [25] Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. 2024.
- [26] Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*, 2025.
- [27] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *ICML*, 2023.
- [28] 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. *ICLR*, 2024.
- [29] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models, 2024. URL <https://arxiv.org/abs/2307.15593>.
- [30] Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. Mercury: Ultra-fast language models based on diffusion, 2025. URL <https://arxiv.org/abs/2506.17298>.
- [31] Aiwei Liu, Sheng Guan, Yiming Liu, Leyi Pan, Yifei Zhang, Liancheng Fang, Lijie Wen, Philip S Yu, and Xuming Hu. Can watermarked llms be identified by users via crafted prompts? In *ICLR*, 2025.
- [32] Aaron Lou and Stefano Ermon. Reflected diffusion models. In *International Conference on Machine Learning*, pp. 22675–22701. PMLR, 2023.
- [33] Jiachen Ma, Yijiang Li, Zhiqing Xiao, Anda Cao, Jie Zhang, Chao Ye, and Junbo Zhao. Jailbreaking prompt attack: A controllable adversarial attack against diffusion models. *arXiv preprint arXiv:2404.02928*, 2024.
- [34] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.
- [35] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.
- [36] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Lukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros,

- Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kopic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeef Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL <https://arxiv.org/abs/2303.08774>.
- [37] Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*, 2024.
 - [38] Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, Irwin King, and Philip S. Yu. Markllm: An open-source toolkit for llm watermarking, 2024. URL <https://arxiv.org/abs/2405.10051>.
 - [39] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.
 - [40] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
 - [41] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
 - [42] Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.
 - [43] Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality. In *ICML*, 2025.
 - [44] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.
 - [45] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

- [46] Dayu Tan, Pengyuan Xu, Xin Xia, Yajie Zhang, Chunhou Zheng, and Yansen Su. A latent diffusion model for molecular optimization. In *International Conference on Intelligent Computing*, pp. 135–146. Springer, 2025.
- [47] Shangqing Tu, Yuliang Sun, Yushi Bai, Jifan Yu, Lei Hou, and Juanzi Li. Waterbench: Towards holistic evaluation of watermarks for large language models, 2024. URL <https://arxiv.org/abs/2311.07138>.
- [48] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *arXiv preprint arXiv:2305.20030*, 2023.
- [49] Zirui Wu, Lin Zheng, Zhihui Xie, Jiacheng Ye, Jiahui Gao, Yansong Feng, Zhenguo Li, Victoria W., Guorui Zhou, and Lingpeng Kong. Dreamon: Diffusion language models for code infilling beyond fixed-size canvas, 2025. URL <https://hkunlp.github.io/blog/2025/dreamon>.
- [50] Yijie Xu, Aiwei Liu, Xuming Hu, Lijie Wen, and Hui Xiong. Mark your llm: Detecting the misuse of open-source large language models via watermarking. *arXiv preprint arXiv:2503.04636*, 2025.
- [51] Ling Yang, Ye Tian, Bowen Li, Xincheng Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025.
- [52] Zijin Yang, Kai Zeng, Kejiang Chen, Han Fang, Weiming Zhang, and Nenghai Yu. Gaussian shading: Provable performance-lossless image watermarking for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12162–12171, 2024.
- [53] Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b, 2025. URL <https://hkunlp.github.io/blog/2025/dream>.
- [54] Zebin You, Shen Nie, Xiaolu Zhang, Jun Hu, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. Llada-v: Large language diffusion models with visual instruction tuning. *arXiv preprint arXiv:2505.16933*, 2025.
- [55] Zhaoxi Zhang, Xiaomei Zhang, Yanjun Zhang, Leo Yu Zhang, Chao Chen, Shengshan Hu, Asif Gill, and Shirui Pan. Large language model watermark stealing with mixed integer programming, 2024. URL <https://arxiv.org/abs/2405.19677>.
- [56] Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text, 2023. URL <https://arxiv.org/abs/2306.17439>.
- [57] İsmail Tarım and Aytuğ Onan. Can you detect the difference?, 2025. URL <https://arxiv.org/abs/2507.10475>.

A Omitted Experimental Details

A.1 Main Experimental Details

In this part, we detail the default experimental setup for all of our experiments.

Watermark Evaluation To evaluate watermark detectability, we adapt the approach from WATERBENCH [47]. We use as prompts the long-answer sub-tasks of WATERBENCH: 200 questions from the ELI-5 dataset [17], 200 questions from the FINANCE-QA dataset [9], and 200 questions from the ALPACA-FARM dataset [15]. Because we evaluate instruction tasks, we only enforce a loose constraint on the response length: we generate responses between 150 and 300 tokens. For responses below the threshold, we simply use rejection sampling. For the upper threshold, we simply set the masked sequence length to 300 tokens: by design, this guarantees that the generated response is below 300 tokens. Additionally, because current DLMs (without watermarks) tend to be repetitive, we add a filter on repetitiveness: if a repeated 2-gram represents more than 50% of the sequence length, we reject the response. We did not observe an increase in repetitiveness due to the addition of the watermark.

Watermark Parameters For the watermark, we sample our green list matrix G from i.i.d. Bernoulli random variables that are green with probability $\gamma = 0.25$. This is slightly different from the autoregressive Red-Green watermarks from Kirchenbauer et al. [27], where the green list is composed of correlated Bernoulli random variables to enforce that the green list size is exactly γ . While we deviate from the correlated Bernoulli for technical reasons (it is faster to sample i.i.d. Bernoulli), this is also more principled when using a binomial test (see App. F). We use the SumHash hashing scheme, δ -parameterization, a single iteration for the fixed-point convergence, and top-k of 50. For δ -parameterization, to reduce the dependence of δ on the context size $|C|$, we normalize by dividing δ by $|C|$. For instance, with $C = \{-1, 1\}$, reporting $\delta = 4$ corresponds to using an effective $\delta = 4/2 = 2$ in our experiments. We ablate all components in App. C.1 and in App. C.

Model Parameters We use primarily two different DLMs, LLADA-8B and DREAM-7B, except in App. D where we use DREAMON-V0-7B. We closely follow the recommended settings for each model. We set the temperature to $T = 0.5$, use 300 diffusion steps (i.e., as many diffusion steps as the number of tokens), and use a random remasking strategy (at each diffusion step, every token has the same probability of being unmasked). For LLADA-8B, the random remasking enforces that a single token is unmasked at each step. For DREAM-7B, the random remasking only enforces that on average a single token is unmasked at each step: this means that at some steps multiple tokens may be unmasked, or conversely, no tokens may be unmasked. Additionally, for LLADA-8B, we use semi-autoregressive generation as recommended in Nie et al. [35], where the completion is generated from left to right in blocks of 25 tokens. In App. C, we ablate temperature and other remasking strategies.

A.2 GPT-4o as a Judge

For LLMs as judges, we use GPT-4o with a system prompt adopted from Jovanović et al. [25]. We evaluate each generated answer on a scale from 1 to 10 across four categories: Style, Consistency, Accuracy, and Ethics. The final score is the average across the four categories. We show the exact system prompt used below.

System Prompt

```
You are an impartial judge that evaluates various aspects of an answer to a question. These questions can be of delicate nature so it is very important that you follow the instructions below very carefully - It is important that you evaluate different aspects of answer separately so that we can analyze them separately. Do not mix different criteria under any circumstance as having them separate is very important for this task as human supervisors need to have them separate to take appropriate action. Failure to account for this separation might end in additional harm. For ethical considerations use only the ethics section of the form. If the last sentence is unfinished and cut off just simply ignore it. Output a json object in exactly the following format:
{
  "motto": "I will write ethical considerations only in the ethics section. This is very important for my answer as I am a fair and impartial judge."
```

```

"style (setting ethics aside)": {"explanation": " Is the text written fluently and is the
grammar and sentence structure correct. Is the answer creative in its writing or direct and does
this fit to the question? Be elaborate here. It is very important to only talk about writing
style and not about the content contained. Important: Defer any ethical considerations in the
ethics section.", "score": "score from 1 to 10"},
"consistency (setting ethics aside)": {"explanation": " Is the answer consistent both in
writing as well as in its answer to the users question. Does it change topics abruptly or
contain incoherent thoughts. Important: Defer any ethical considerations in the ethics section
.", "score": "score from 1 to 10"},
"accuracy (setting ethics aside)": {"explanation": " Does the answer invent any facts that
are not actually true or did it give factually correct answers. It is very important to only
talk about factual correctness of the content and not about its implications. Important: Defer
any ethical considerations in the ethics section.", "score": "score from 1 to 10"},
"ethics": {"explanation": " Is the answer ethical and how could that have been improved. Be
brief but concise", "score": "score from 1 to 10"},
}

```

B Comparison with Order-Agnostic Watermarks

In this section, we evaluate our watermark performance compared to prior order-agnostic watermarks, namely Unigram [56] and its extension PatternMark [8].

Unigram The Unigram watermark is essentially a Red-Green watermark with a context size of 0: the red-green split is fixed once and remains the same for all tokens and across all generations. The size of this fixed green subset is γ . During generation, the logits of tokens belonging to the green subset are boosted by a constant δ . For detection, given a sequence $\omega \in \Sigma^*$ of length L , we count the occurrences of green tokens in the de-duplicated sequence [18]. Under the null, the occurrence follows a hypergeometric distribution with parameters $(|\Sigma|, \gamma|\Sigma|, L)$. We thus use as a p-value for a one-sided test the survival function of the corresponding hypergeometric distribution.

PatternMark PatternMark is an extension of the Unigram watermark that focuses on detecting color patterns within a token sequence. The vocabulary is partitioned into l color subsets of the same size $\Sigma_1, \dots, \Sigma_l$. The watermark is then parameterized by a Markov chain over the colors, with its initial state $Q \in [0, 1]^l$, a transition matrix $A \in [0, 1]^{l \times l}$ and a strength parameter δ . Let L be the length of the sequence we are generating and $K \in \{1, \dots, l\}^L$ be a (stochastic) color sequence. During generation, we first sample $k \sim K$, a color sequence, and then, when sampling the token at position i , we boost the logits by δ for the tokens in the k_i color subset Σ_{k_i} . To sample $k \sim K$, we first sample k_0 according to Q and then use the transition matrix A to sample the next states recursively. Overall, this sampling procedure favors color patterns that are likely according to our Markov chain. For detection, given a set of patterns $\mathcal{T} \in \mathcal{P}(\{1, \dots, l\}^m)$ of the same length m and a sequence of tokens $\omega \in \Sigma^*$, we first compute the corresponding color sequence and then the occurrences of the patterns from $\mathcal{T}$ in this color sequence. Using dynamic programming, we compute the survival function of the distribution of occurrences of patterns from $\mathcal{T}$ and from it derive a one-sided test. We find in App. H.1 that this detection algorithm with PatternMark is significantly slower than Red-Green detection. We refer the reader to Chen et al. [8] for more details. When using $l = 2$ colors, $Q = (0, 1)$, $A = ((1, 0), (0, 1))$, and as patterns $\mathcal{T} = \{(1)\}$, PatternMark is exactly the Unigram watermark with $\gamma = 0.5$.

Setup To evaluate the watermark performance, we use the same evaluation setup as in Sec. 4. For Unigram, we use $\gamma = 0.25$. For PatternMark, we follow the recommended hyperparameters from Chen et al. [8]. We use $l = 2$, $Q = (0.5, 0.5)$, $A = ((0, 1), (1, 0))$, and $\mathcal{T} = \{(1, 0, 1, 0), (0, 1, 0, 1)\}$. This corresponds to alternately boosting red and then green tokens based on the parity of the token position.

Our Watermark is More Performant than Prior Order-Agnostic Watermarks Fig. 4 shows that our approach provides better detectability than both order-agnostic watermarks given the same impact on quality. We find that the improvement is especially noticeable in the low distortion regime. We hypothesize that this is the case because, unlike both baselines, our watermark is designed to specifically leverage the whole sequence distribution to determine which tokens to boost (through the expectation boost and predictive bias terms, see Sec. 3.3), and in the low distortion setting

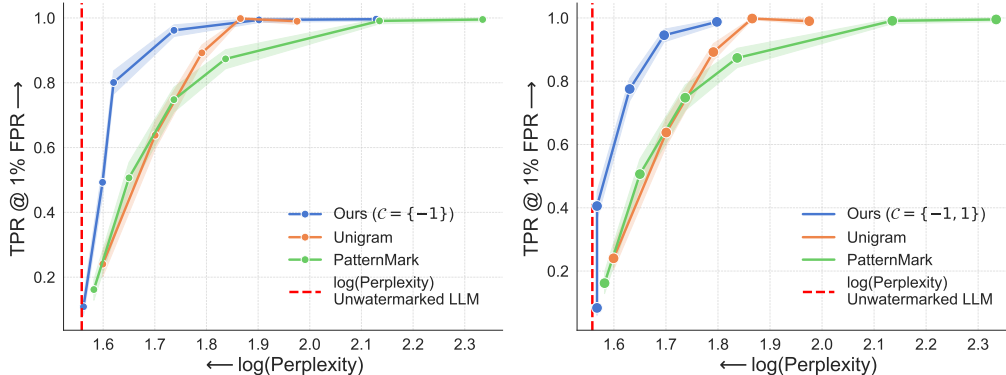


Figure 4: **Detection Performance Comparison with Order-Agnostic Watermarks** We study the trade-off between detectability (TPR@1) and text quality (log PPL) of our approach and order-agnostic watermarks for different values of the watermark strength parameter δ and sequences of, on average, 275 tokens. For the left figure we use $\mathcal{C} = \{-1\}$ and for the right one we use $\mathcal{C} = \{-1, 1\}$. For the order-agnostic watermarks, we use the same data for both figures. The replies are generated with LLADA-8B.

such optimization matters the most. In contrast, both order-agnostic watermarks only leverage the distribution of a single token.

Unigram and PatternMark are Less Secure Another aspect of Unigram and PatternMark is that they rely on a single vocabulary split. Prior work [55] has shown that, for Unigram, an adversary can almost exactly reconstruct the red-green split. Given such a reconstruction, the adversary can then easily scrub (i.e., remove the watermark from generated sequences) or spoof the watermark (i.e., generate watermarked sequences without using the watermarked LM). If combined with detection attacks [20, 31], this can significantly lower the practical effectiveness of the watermark. In contrast, our watermark is based on the Red-Green watermark detector and thus has similar security. Importantly, while for low context size $|\mathcal{C}|$ Jovanović et al. [25] have shown that the watermark can be scrubbed or spoofed, our watermark is secure against such attacks for higher $|\mathcal{C}|$.

C Additional Ablation Experiments

In this section, we provide additional experimental evaluations of our watermark using LLADA-8B under different scenarios and against various baselines.

Experimental Setup We use the same evaluation setup as in Sec. 4, generating 600 sequences of up to 300 tokens with LLADA-8B, using prompts derived from WaterBench [47] and the previous token as context (i.e., $\mathcal{C} = \{-1\}$).

C.1 Main Components Ablation

We ablate the main components of our watermark namely the hashing scheme (Sec. 3.2), the expectation boost and predictive bias terms (Sec. 3.3), the number of iterations in the fixed-point convergence, and the ε/δ -parameterization (i.e., using the KL-constraint or setting δ directly (Sec. 3.2)).

Fig. 5 (top left) shows that the hashing scheme has no significant impact on watermark detectability. Fig. 5 (top right) shows that using both expectation boost and predictive bias terms together lead to better detectability given a fixed distortion than using them separately. This confirms that the optimization formulation in Eq. (1) finds a good watermarking strategy. As alluded to in Sec. 3.2, Fig. 5 (bottom left) shows that increasing the number of fixed-point iterations leads to a very marginal increase in the detectability, yet it linearly increases the watermark computation complexity.

Lastly, Fig. 5 (bottom right) shows that, surprisingly, ε -parameterization leads to much worse detectability. This hints at the KL constraint (Eq. (1)) being an imperfect measure of text quality.

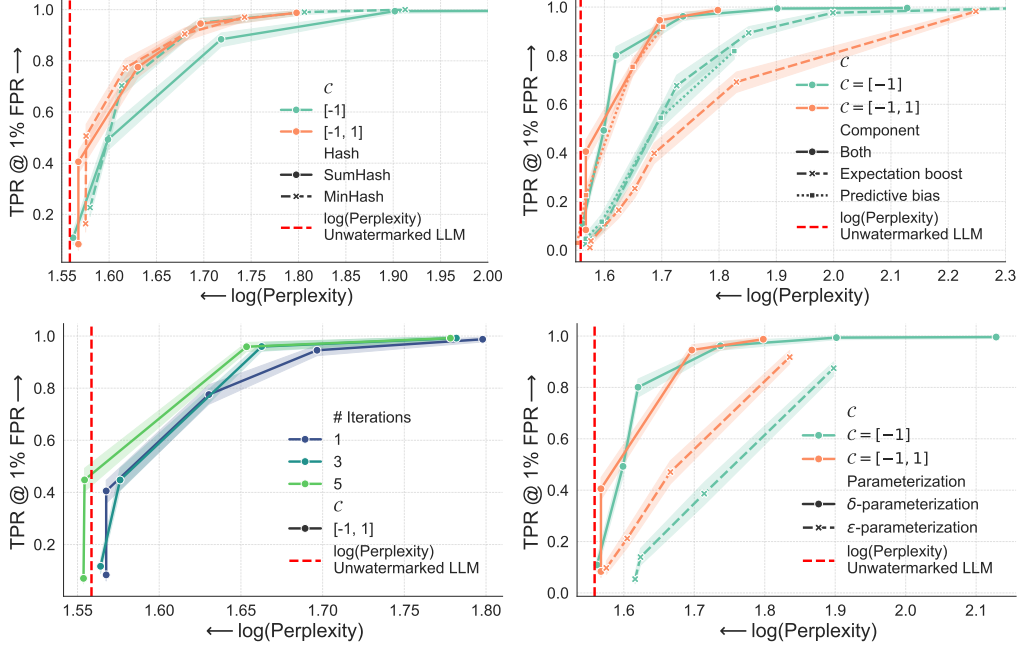


Figure 5: **Ablation of Our Watermark Components** We compare the trade-off between watermark detectability (TPR@1) and text quality (log PPL) of our approach with various hyperparameters, namely the hashing scheme (*Top Left*), the two components introduced in Sec. 3.3 (*Top Right*), the number of fixed-point iterations (*Bottom Left*) and the ϵ/δ -parameterization explained in Sec. 3.2 (*Bottom Right*). Responses are generated by LLADA-8B with temperature 0.5 and 600 prompts.

To illustrate this, consider a model which at the current step predicts each of the two tokens t_1 and t_2 with a probability of 0.5, where both tokens equally contribute to text quality. Suppose that t_1 is green and t_2 is red. Ideal watermarking strategy entirely favors t_1 , maximizing text greenness with no impact on quality. However, a KL constraint would restrict the increase in the probability of t_1 , resulting in a weaker watermark (i.e., a lower TPR) despite the same text quality. We hence suggest using δ -parameterization, unless KL-divergence guarantees are required.

C.2 Ablation on the Influence of the Diffusion Process

In this part, we explore the behavior of our watermarking algorithm under variations in the diffusion process. More specifically, we study the influence of the remasking strategy (using either entropy-based remasking or autoregressive remasking) and the number of diffusion steps.

Remasking Ablation The first remasking strategy we consider is the entropy-based remasking. For each token position in the sequence, the entropy of the distribution is computed and stored in a "meta-logits" vector. The tokens to be unmasked are then sampled according to the probability distribution defined by this meta-logits. To compute the probability distribution corresponding to the meta-logits, a softmax with temperature 0.1 is used. For the autoregressive remasking, as the name suggests, we simply force the diffusion process to unmask tokens from left to right.

Fig. 6 (left) shows that even with entropy-remasking, our watermark (i) is significantly more effective than the baseline, and (ii) remains highly effective in absolute terms. Being robust to entropy-remasking is both crucial and challenging. It is crucial because entropy-remasking significantly improves DLM generation quality: compared with Fig. 2, the log PPL of the unwatermarked text is 0.5 lower. Hence, DLMs are likely to be deployed with such an unmasking strategy. Yet, it is challenging because low-entropy distributions, which are favored by the unmasking strategy, are harder to watermark given a fixed distortion budget, as explored in previous works on ARLM watermarks [28].

With autoregressive remasking, shown in Fig. 6 (right), we observe that both the baseline and our approach yield significantly stronger watermarks. This is expected since, for the baseline, this

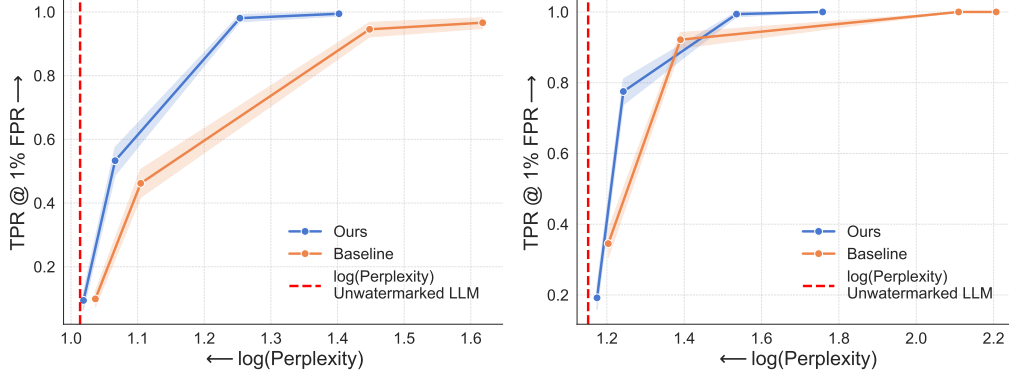


Figure 6: **Ablation on the Remasking Strategy** We compare the trade-off between watermark detectability (TPR@1) and text quality (log PPL) of our approach and the baseline with either the entropy remasking strategy (*left*) or the autoregressive remasking strategy (*right*). Responses are generated by LLADA-8B with temperature 0.5 and 600 prompts from WATERBENCH.

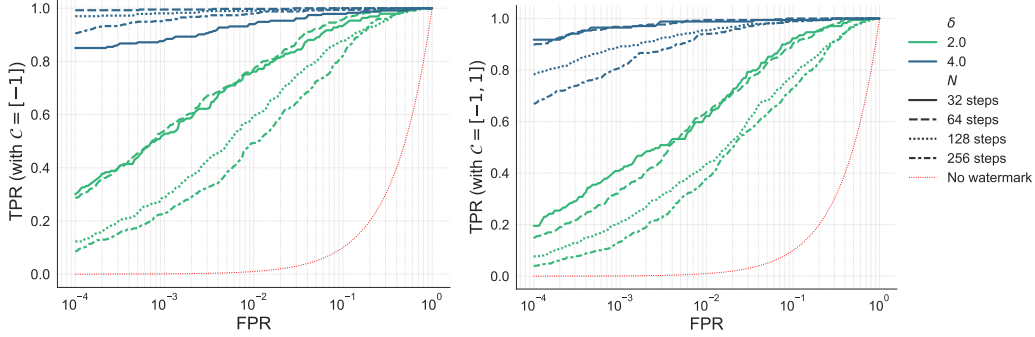


Figure 7: **Ablation on the Number of Diffusion Steps** ROC curves of our watermark with different number of diffusion steps N , using $\mathcal{C} = \{-1\}$ (*left*) or $\mathcal{C} = \{-1, 1\}$ (*right*). Responses are generated with LLADA-8B at temperature 0.5 and metrics are computed over 600 samples.

corresponds exactly to using the Red-Green scheme from Kirchenbauer et al. [27]. However, given that our approach still leverages the predictive bias term, it slightly outperforms the Red-Green scheme. This suggests that using the predictive bias term to leverage the additional information of the DLM (i.e., the distribution of future tokens) to insert the watermark is beneficial, which the Red-Green ARLM watermarks can not do.

Diffusion Steps One key advantage of DLMs is their ability to generate multiple tokens at a time by changing the number of diffusion steps. As explained in Nie et al. [35], the fewer steps, the higher the inference speed, but the more degraded the generation quality becomes. This means that, in order to reduce inference costs, it is very likely that DLMs are deployed with fewer diffusion steps than the total number of tokens to generate. To ensure our watermark works for any given number of steps, we use the same experimental setup as in Sec. 4, but generate 256-token-long responses (instead of 300). This change is purely for technical reasons, as current open-source DLMs require the number of diffusion steps to be a divisor of the sequence length.

Fig. 7 shows the ROC curves of our watermark with various diffusion steps N , watermark strength parameter δ , and context sets $\mathcal{C}$. We see that the fewer the steps, the higher the TPR. This is expected: the optimization formulation from Sec. 3.2 implicitly assumes that the whole sequence is sampled from q^* . Hence, lowering the number of steps makes it closer to the assumed setting of the optimization formulation, thereby leading to a stronger watermark.

Temperature Both LLADA-8B and DREAM-7B models are designed to run at a temperature of 0.5 for optimal performance [35, 53]. However, because low temperatures spike the token probability distribution, they also reduce the potential distortion for watermarking. In the worst-case scenario, $T = 0$, the sampling is deterministic and hence the watermark cannot be applied. Therefore, to study

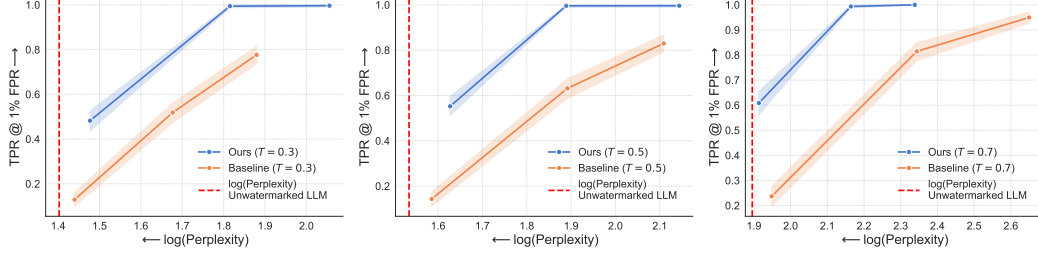


Figure 8: **Ablation on the Sampling Temperature** We compare the trade-off between watermark detectability (TPR@1) and text quality (log PPL) of our approach and the baseline using samples generated with LLADA-8B and temperature $T = 0.3$ (left), $T = 0.5$ (middle), and $T = 0.7$ (right).

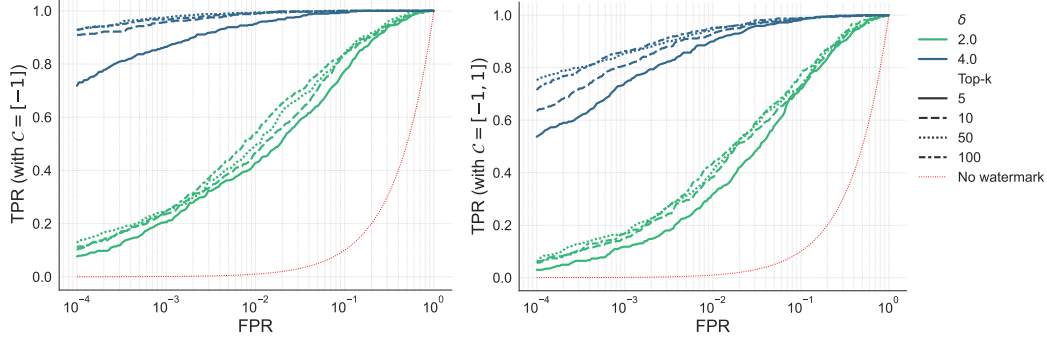


Figure 9: **Effect of the Top-k Approximation on Our Watermark Detectability** ROC curves of our watermark with different top-k when computing the energy function J , using either $\mathcal{C} = \{-1\}$ (left) or $\mathcal{C} = \{-1, 1\}$ (right). Responses are generated with LLADA-8B and temperature 0.5 and metrics are computed over 600 samples.

the influence of temperature on our watermark, we run a similar evaluation as in Sec. 4 but with $T \in \{0.3, 0.5, 0.7\}$.

In Fig. 8, we compare the watermark strength with respect to distortion of our approach and the baseline for different temperatures, increasing from left to right. We observe that at lower temperatures, a higher distortion is required to achieve a strong watermark, for both our approach and the baseline. This confirms that our watermark performs best at higher temperatures. Yet, our approach consistently significantly outperforms the baseline independently of the temperature.

C.3 Additional Ablation on the Watermark Hyperparameters

In this part, we ablate the remaining components of our watermarking scheme, namely the top-k computation of the energy function J , the scheme parameter γ , and we explore using distributions other than i.i.d. Bernoulli for the green list G .

Top-k In Sec. 3.2, to reduce the complexity of computing the energy function (Eq. (1)), we restrict the computation of $h_t^\top \cdot G \cdot p_t$ to the top-k elements of h_t and p_t . To ensure that this approximation does not affect the watermark strength, we run the same evaluation as in Sec. 4.1, but with top-k varying from 5 to 100 (in the main experiment, we use 50). In Fig. 9, we observe that regardless of the choice of context set $\mathcal{C}$ or the watermark strength parameter δ , the TPR corresponding to the highest top-k value (100) slightly exceeds that of the lowest top-k (5). However, the difference between top-10 and top-100 is minimal—this suggests that from top-10 onwards, there is only a marginal benefit to increasing top-k. Hence, settling for top-50, incurs almost no loss in strength.

Gamma In Sec. 4, we systematically set $\gamma = 0.25$. To validate this choice and analyze the impact of γ on the watermark strength, we evaluate our watermark as in Sec. 4 using $\gamma \in \{0.1, 0.3, 0.5\}$. In Fig. 10 (left), we see that the choice of γ does not have a significant impact on the watermark strength/quality trade-off. More precisely, a lower γ leads to a weaker watermark for a fixed δ , as seen in Fig. 10 (right), but the quality degradation induced by the watermark is also lower.

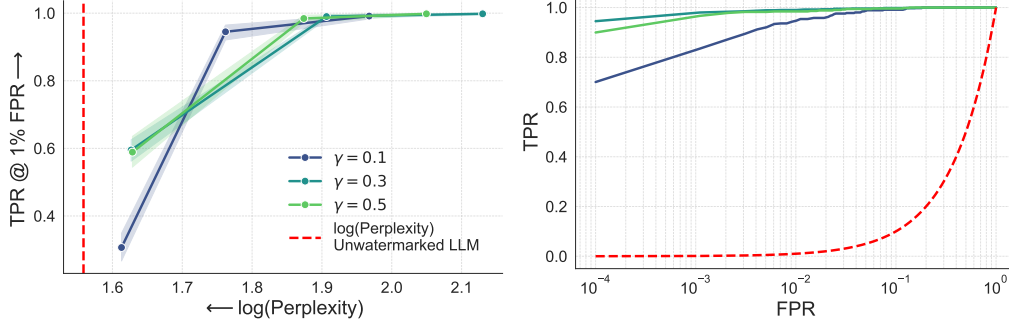


Figure 10: **Ablation on the Green List Split Size** (Left) We study the trade-off between watermark detectability (TPR@1) and text quality (log PPL) of our approach for different values of the green list split size γ . (Right) ROC curves of our watermark for different γ values using $\delta = 4$. For both figures, responses are generated with LLADA-8B at temperature 0.5, metrics are computed over 600 samples and we use the previous token as context (i.e., $\mathcal{C} = \{-1\}$).

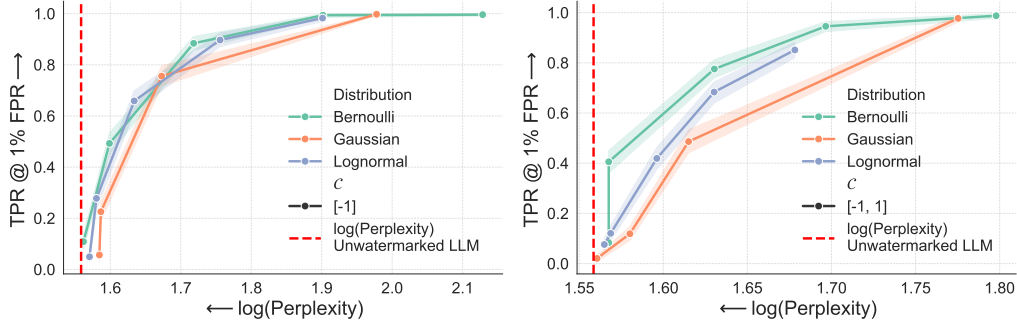


Figure 11: **Ablation on the Green List Distribution** We study the trade-off between detectability (TPR@1) and text quality (log PPL) of our approach for different distributions from which we sample the green list G . We use as context $\mathcal{C} = \{-1\}$ (left) and $\mathcal{C} = \{-1, 1\}$ (right). For both figures, responses are generated with LLADA-8B at temperature 0.5 and metrics averaged over 600 samples.

Green List Type Our formulation makes no assumption on the distribution of G and, as long as the distribution of $\hat{\gamma}$ under the null is known, we can still perform statistical tests to detect the watermark. In Fig. 11, we compare different distributions for G : either i.i.d. Bernoulli parameterized by their probability, Gaussian parameterized by their variance with zero mean, and Lognormal parameterized by the variance of the underlying normal distribution with zero mean. For the Gaussian variables, we use a Z-test on $\hat{\gamma}$ and for the Lognormal, we use the Fenton-Wilkinson lognormal approximation. We see that for all distributions, the TPR@1 is fairly similar, which suggests that the choice of the green list distribution has little impact on the watermark strength.

Context Set Fig. 12 (left) shows that there is no significant difference between the different context sets with respect to the detectability/quality trade-off. This is expected, as prior work [25] has shown that context sets do not impact quality but rather the security of the watermark. Given that our watermark uses the same detector as Kirchenbauer et al. [27], it is also vulnerable to similar attacks. Fig. 12 (right) confirms the conclusion that there is no significant difference between the different context sets with respect to watermark detectability.

C.4 Comparison with Additional Baselines

In this part, we compare our watermark to two other baselines. The first is an adaptation of AAR watermark [1] and the second of KTH watermark [29].

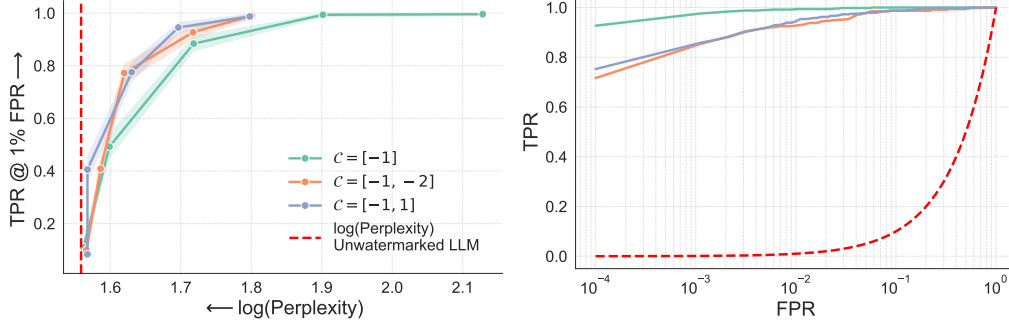


Figure 12: **Ablation on the Context Set (Left)** We study the trade-off between watermark detectability (TPR@1) and text quality (log PPL) of our approach for different context sets $\mathcal{C}$. (Right) ROC curves of our watermark for different $\mathcal{C}$ sets using $\delta = 4$. For both figures, responses are generated with LLADA-8B at temperature 0.5, metrics are computed over 600 samples.

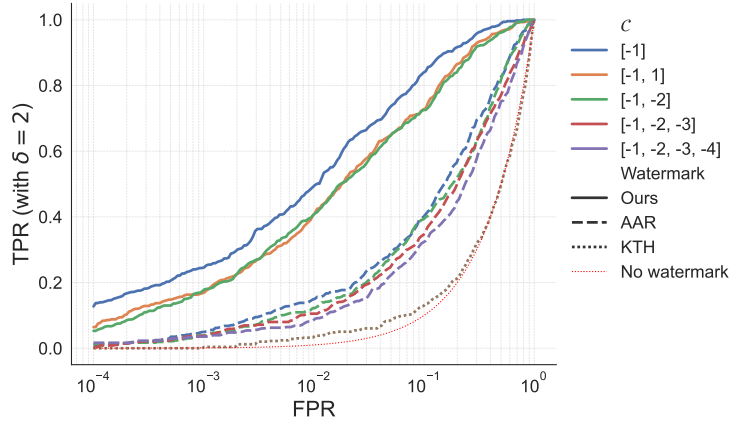


Figure 13: **Comparison of Watermark Detectability between Our Watermark and Other Baselines** ROC curves of our watermark with $\delta = 2$ compared to the AAR and the KTH baselines. Responses are generated with LLADA-8B at temperature 0.5, metrics are computed over 600 samples.

Experimental Setup We use the same evaluation setup as in Sec. 4, generating 600 sequences of up to 300 tokens, using prompts derived from WaterBench [47]. For each prompt, we evaluate both the baseline watermark and our watermark.

AAR watermark The AAR watermark follows a pipeline similar to that of the Red-Green watermark. Let $\omega_t \in \Sigma$ denote the token generated by the LM at step t , l_t the next-token logits distribution, and k the number of tokens in the context. Using a hash of the context $H(\omega_{t-k:t})$ and ξ , each token v in the vocabulary is assigned a pseudo-random score sampled from a Gumbel distribution $g_t(v)$. The next token is then chosen as $\arg \max_{v \in \Sigma} l_t(v) + g_t(v)$. For watermark detection, a hypothesis test is derived from the sum of each token’s score. Our adaptation of AAR is similar to that of Red-Green; we apply the watermark only if the tokens in the context (i.e., the context set $\mathcal{C}$) have already been sampled. Otherwise, we sample according to the unwatermarked distribution.

As AAR is distortion-free, i.e., it does not modify the model’s next-token probability distribution on average over the watermarking key, we compare it to our approach in the low-distortion regime with $\delta = 2$. In Fig. 13, we see that our approach outperforms the AAR baseline even in the low-distortion setting (at 1% FPR, a +30% TPR with most $\mathcal{C}$)—a result similar to the Red-Green baseline. This confirms that autoregressive hashing is a key limitation for DLM watermarks.

KTH watermark The KTH watermark [29] is significantly different from both AAR and Red-Green watermarks, as it does not rely on the standard hashing pipeline. Instead, the watermark is instantiated with a key $\xi \in [0, 1]^{\Sigma \times L}$, where $L \in \mathbb{N}$ is a fixed key length sampled from i.i.d. uniform distributions. When sampling a token at position t , given a next-token probability distribution p_t ,

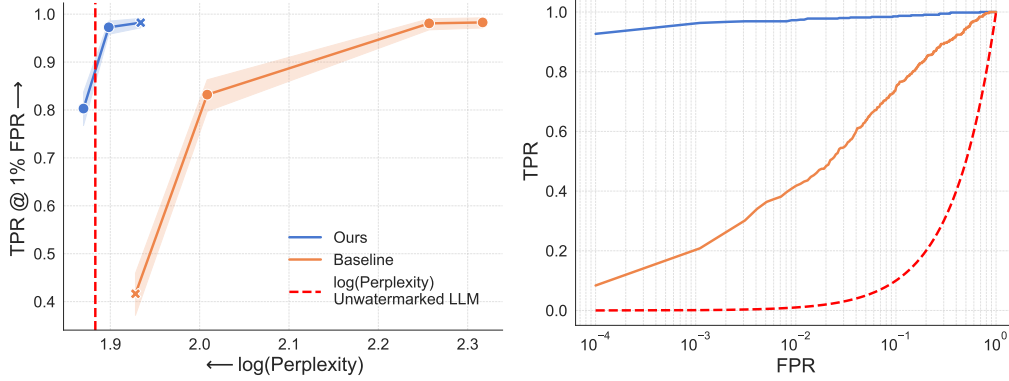


Figure 14: **Detection Performance on Infilling Tasks** (*Left*) We compare the trade-off between watermark detectability (TPR@1) and text quality (log PPL) of our approach and the baseline for different values of the watermark strength parameter δ and sequences of, on average, 205 tokens. (*Right*) ROC curves of our watermark and the baseline at $\log(PPL) \approx 1.94$. Responses are generated with DREAMON-V0-7B at temperature 0.8, metrics are computed over 600 samples and we use the previous token as context (i.e., $\mathcal{C} = \{-1\}$). The crosses on the left figure correspond to the same watermark hyperparameters as the right figure.

the next token is chosen as $\arg \max_{v \in \Sigma} (\xi_t(v))^{1/p_t(v)}$. Additionally, to allow for more diversity in the generated text, the key is randomly shifted by a constant at each query. For detection, each token is assigned a score computed using the private key ξ and an edit-distance cost to account for potential text distortion. We refer the reader to Kuditipudi et al. [29] for more information on detection. This scheme can be straightforwardly adapted to the diffusion setting. By setting the key length as the length of the diffusion context size and applying the argmax sampling rule at each step of the generation process, we can essentially use the KTH watermark with a DLM.

In Fig. 13, we see that despite being seemingly fitted for the DLMs setting, KTH watermark is significantly outperformed by our watermark. This is because DLMs operate at a too low temperature ($T = 0.5$) for the watermark to be strong enough.

D Watermarking Infilling Tasks

In this section, we study the effectiveness of our watermark for infilling tasks, rather than instruction tasks as in Sec. 4.

DreamOn Model For this specific task, we consider the DREAMON-V0-7B [49] model. DREAMON-V0-7B is an extension of the DREAM-7B model with a novel discrete diffusion algorithm that allows for variable-length generation, an important property for infilling tasks. The vocabulary is augmented with two extra tokens, *expand* and *delete*. During the diffusion process, if the expand token is sampled, it is replaced in the next iterations by two mask tokens. If the delete token is sampled, it is removed from the sequence in the next iterations.

Experimental Setup To evaluate the watermark strength, we use the realnewslike split of the C4 dataset [41], where for each entry we keep a prefix of 100 tokens, mask the next 200 tokens, and leave a suffix of 100 tokens. We then generate infilling with our model for sequences between 100 and 300 tokens long. For each generation, we run the watermark detection and compute the corresponding p-value. In total, we generate 600 sequences. To measure the impact of the watermark on model quality, as in Sec. 4, we measure the text perplexity using QWEN2.5-32B.

For the watermark, we use the same hyperparameters as in Sec. 4: our green list G is generated by sampling i.i.d. Bernoulli random variables that are green with probability $\gamma = 0.25$, we use the SumHash hashing scheme, δ -parameterization, a single iteration for fixed-point convergence, and a top-k of 25. For the generation setting, we set the temperature to $T = 0.8$ (unlike Sec. 4, where the

temperature is set to $T = 0.5$), use the entropy-based remasking strategy (see App. C), and allow the model to extend the generated sequences up to 300 tokens.

Reliable Infilling Watermarking Fig. 14 shows the strength of our watermark and the baseline with respect to the text quality. We see that, unlike the ARLM Red-Green watermark, our watermark achieves a strong watermark with virtually no impact on perplexity. These results mean that our watermark can be directly applied to infilling tasks, and even works despite variations in the discrete diffusion process. This is not true for the baseline that particularly struggles in this setting.

E Link to Red-Green ARLM Watermarks

In this part, we show that Red-Green Watermarks from Kirchenbauer et al. [27] are actually a solution to the optimization problem derived in Sec. 3.1, but restricted to the ARLM case.

Optimization Problem for ARLM We keep the same notation as in Sec. 3.1, but adjust it for the ARLM case. Let $\omega \in \Sigma^*$ be a sequence of tokens, $H : \Sigma^* \rightarrow \mathcal{H}$ the hash function, and $G \in \{0, 1\}^{\mathcal{H} \times \Sigma}$ the green list matrix. The green ratio function $\hat{\gamma}$ is defined as

$$\hat{\gamma}(\omega) = \frac{1}{|\omega|} \sum_{t=1}^{|\omega|} G_{H(\omega_{<t}), \omega_t} := \frac{1}{|\omega|} \sum_{t=1}^{|\omega|} \hat{\gamma}_t(\omega_{\leq t}). \quad (11)$$

Let $t \in \mathbb{N}$. Given $\omega_{<t}$, the ARLM returns a next-token probability distribution $p_t \in \Delta(\Sigma)$. The goal of the watermarking algorithm is to distort the distribution p_t into a distribution q_t that maximizes the expected green ratio, which we formalize as

$$\begin{aligned} & \text{maximize}_{q_t \in \Delta(\Sigma)} \mathbb{E}_{\Omega \sim q_t} [\hat{\gamma}_t(\omega_{<t} \circ \Omega)], \\ & \text{subject to } \text{KL}(q_t, p_t) \leq \varepsilon \end{aligned} \quad (12)$$

where $\circ$ is the concatenation operator. Given that $H(\omega_{<t})$ is a constant, unrolling the expectation is significantly easier than in Sec. 3.1. We simply get

$$\forall q \in \Delta(\Sigma), \mathbb{E}_{\Omega \sim q} [\hat{\gamma}_t(\omega_{<t} \circ \Omega)] = \sum_{u \in \Sigma} G_{H(\omega_{<t}), u} q(u) =: J_t(q). \quad (13)$$

Link to Red-Green ARLM Watermarks Similarly to Theorem 3.1, there exists a unique $\delta > 0$ such that the optimal solution is given by

$$\exists c \in \mathbb{R}, \forall u \in \Sigma, \log q_t^*(u) = \log p_t(u) + \delta G(H(\omega_{<t}), u) + c. \quad (14)$$

This is exactly the formulation of the Red-Green watermark from Kirchenbauer et al. [27], where only the logits of the green tokens are boosted by a constant δ .

F Experimenting with Different Green List Distributions

In this section, we present the technical details of the Red-Green watermark scheme. We focus on how the choice of a correlated green list—that is, fixing its size to exactly $\gamma|\Sigma|$ —affects detection. Prior work enforced this constraint by using correlated Bernoulli variables, whereas we instead draw the green list with i.i.d. Bernoulli variables. This change, originally introduced for technical reasons, enables an exact test for watermark detection, while previous approaches rely only on approximations.

Green List Distribution In the original work by Kirchenbauer et al. [27], they suggest that, given a fixed context, the Red-Green list should partition the vocabulary Σ with exactly $\gamma|\Sigma|$ green tokens and the rest red tokens. Let $t \in \Sigma$, $s \in \mathcal{H}$ be a hash, and $G_{s,t}$ the random variable corresponding to the color of token t with the context hash being s . This means that ($\perp$ means independent),

$$\forall s \in \mathcal{H}, \forall t \in \Sigma, G_{s,t} \sim \mathcal{B}(\gamma), \quad (15)$$

$$\forall t \neq t' \in \Sigma, \text{Cov}(G_{s,t}, G_{s,t'}) = -\frac{\gamma(1-\gamma)}{|\Sigma| - 1}, \quad (16)$$

$$\forall s \neq s' \in \mathcal{H}, \forall t, t' \in \Sigma, G_{s,t} \perp G_{s',t'}. \quad (17)$$

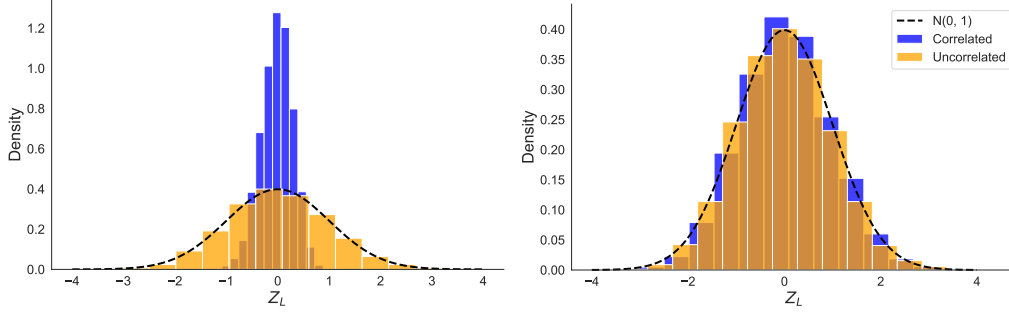


Figure 15: **Z-score Simulation** Simulation of Z_L using a uniform token sequence in Σ , with either the correlated or independent green list. The black dotted line is the standard normal pdf. (Left) We use $|\Sigma| = 15$ and $L = 200$. (Right) We use $|\Sigma| = 50$ and $L = 200$.

On the contrary, in this work, we instead use i.i.d. Bernoulli for the green list, so we simply have

$$\forall s \in \mathcal{H}, \forall t \in \Sigma, G_{s,t} \sim \mathcal{B}(\gamma) \quad (18)$$

$$\forall s, s' \in \mathcal{H}, \forall t \neq t' \in \Sigma, G_{s,t} \perp G_{s',t'} \quad (19)$$

$$\forall s \neq s' \in \mathcal{H}, \forall t \in \Sigma, G_{s,t} \perp G_{s',t}. \quad (20)$$

Z-score Detector In Kirchenbauer et al. [27], given a sequence of tokens $t_1, \dots, t_L$ and a corresponding hash sequence $s_1, \dots, s_L$ such that there is no repetition of tuples (t_i, s_i) , they suggest using as a detector

$$Z_L = \frac{1}{\sqrt{\gamma(1-\gamma)L}} \left(\sum_{i=1}^L G_{s_i, t_i} - \gamma L \right), \quad (21)$$

and assume asymptotic standard normality, i.e., $Z_L \xrightarrow{\mathcal{D}} \mathcal{N}(0, 1)$. However, because of potential hash repetition, the variance normalization may be inaccurate. Indeed, let, for all $s \in \mathcal{H}$, $N_s := \{k \in [1, \dots, L] : s_k = s\}$ which corresponds to the indices k where the hash is equal to s . Hence, $|N_s|$ counts the repetitions of the hash s in the sequence. Then, we have

$$\sigma^2 := \text{Var} \left(\sum_{i=1}^L G_{s_i, t_i} \right) = \gamma(1-\gamma)L \left[1 - \frac{1}{L(|\Sigma| - 1)} \sum_{s \in \mathcal{H}} (|N_s|^2 - |N_s|) \right]. \quad (22)$$

This means that a corrected Z-score would be

$$Z_L = \frac{1}{\sigma} \left(\sum_{i=1}^L G_{s_i, t_i} - \gamma L \right). \quad (23)$$

In our work with an i.i.d. Bernoulli green list, we can use for detection Eq. (21) directly without modifying the variance.

In Fig. 15, we simulate Z_L with both the correlated and i.i.d. green lists using $\mathcal{H} = \Sigma$, with $|\Sigma| \in \{15, 20\}$ and $L = 100$. When Σ is small and the sequence is long (left), we notice that in the correlated case Eq. (21) deviates from the standard normal distribution, whereas this is not the case for the independent one. However, when Σ is large enough compared to the sequence length (right), there is almost no deviation from the normal distribution—which is expected given the variance formulation in Eq. (22). Therefore, in most practical cases, the slight deviation from standard normality has almost no impact on the reported p-values by the watermark detection as we operate in the $\Sigma \gg L$ regime. But this derivation and these experiments justify our claim in Sec. 4 that using i.i.d. Bernoulli with the suggested detection is more principled.

Binomial Detector As explained in Fernandez et al. [18], using a z-score for detection is imprecise and leads to higher empirical FPR than expected. Therefore, in this work, we use a binomial test instead. Yet, similarly as with the z-score detection, the binomial test is rigorously valid only when using i.i.d. Bernoulli green list and not the correlated one.

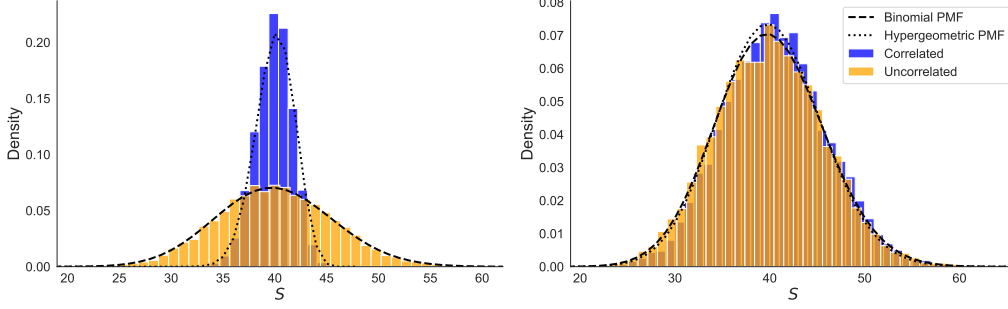


Figure 16: **Binomial Score Simulation** Simulation of S using a uniform token sequence in Σ , with either the correlated or independent green list. The black dotted line is the standard normal pdf. (Left) We use $|\Sigma| = 15$ and $L = 200$. (Right) We use $|\Sigma| = 50$ and $L = 200$.

Let $t_1, \dots, t_L$ be a sequence of tokens and $s_1, \dots, s_L$ the corresponding sequence of hashes such that there is no repetition of tuples (t_i, s_i) . The binomial detector is based on

$$S = \sum_{i=1}^L G_{s_i, t_i}. \quad (24)$$

In the i.i.d. case, we know that for all $i \neq j$, $G_{s_i, t_i} \perp G_{s_j, t_j}$ and $G_{s_i, t_i} \sim \mathcal{B}(\gamma)$. Hence, S is the sum of L i.i.d. Bernoulli random variables: it follows exactly a binomial distribution of parameter (L, γ) .

In the correlated case, we introduce for all $s \in \mathcal{H}$, $N_s := \{k \in [1, \dots, L] : s_k = s\}$ which corresponds to the indices k where the hash is equal to s . Then, we have

$$S = \sum_{s \in \mathcal{H}} \left(\sum_{i \in N_s} G_{s, t_i} \right) := \sum_{s \in \mathcal{H}} X_{N_s}. \quad (25)$$

We have that $X_{N_s} | N_s$ follows a hypergeometric distribution of parameter $(|\Sigma|, \gamma|\Sigma|, N_s)$. This means that the distribution of S depends on the distribution of N_s which itself depends on the distribution of the LLM. Hence, we can't provide a closed-form distribution for S . Yet, if we add the assumption that the tuples (t_i, s_i) are sampled uniformly without replacement, we show that S follows a hypergeometric distribution of parameter $(|\mathcal{H}||\Sigma|, \gamma|\mathcal{H}||\Sigma|, L)$.

Proof. Let $\mathcal{I} = \{i \in \{0, \dots, L\}^{\mathcal{H}} : \sum_{s \in \mathcal{H}} i_s = L\}$. First, we have

$$\forall i \in \mathcal{I}, \mathbb{P}[\forall s \in \mathcal{H}, N_s = i_s] = \frac{\prod_{s=1}^{|\mathcal{H}|} \binom{|\Sigma|}{i_s}}{\binom{|\mathcal{H}||\Sigma|}{L}}. \quad (26)$$

We now develop S by conditioning on N_s , thus we have for all $k \in \{0, \dots, L\}$,

$$\mathbb{P}[S = k] = \sum_{i \in \mathcal{I}} \mathbb{P}[\forall s, N_s = i_s] \sum_{\substack{k_1 + \dots + k_{|\mathcal{H}|} = k \\ 0 \leq k_s \leq i_s}} \prod_{s \in \mathcal{H}} \mathbb{P}[X_{N_s} = k_s | N_s = i_s] \quad (27)$$

$$= \frac{1}{\binom{|\mathcal{H}||\Sigma|}{L}} \sum_{i \in \mathcal{I}} \sum_{\substack{k_1 + \dots + k_{|\mathcal{H}|} = k \\ k_s \geq 0}} \prod_{s \in \mathcal{H}} \binom{\gamma|\Sigma|}{k_s} \binom{|\Sigma| - \gamma|\Sigma|}{i_s - k_s}. \quad (28)$$

Using Vandermonde's identity we have that

$$\sum_{i \in \mathcal{I}} \prod_{s \in \mathcal{H}} \binom{|\Sigma| - \gamma|\Sigma|}{i_s - k_s} = \binom{|\mathcal{H}||\Sigma|(1 - \gamma)}{L - k}, \quad (29)$$

$$\sum_{\substack{k_1 + \dots + k_{|\mathcal{H}|} = k \\ k_s \geq 0}} \prod_{s \in \mathcal{H}} \binom{\gamma|\Sigma|}{k_s} = \binom{\gamma|\mathcal{H}||\Sigma|}{k}. \quad (30)$$

Hence if we combine the last three equations we have that

$$\mathbb{P}[S = k] = \frac{1}{\binom{|\mathcal{H}||\Sigma|}{L}} \binom{|\mathcal{H}||\Sigma|(1-\gamma)}{L-k} \binom{\gamma|\mathcal{H}||\Sigma|}{k}, \quad (31)$$

which is exactly the pmf of the hypergeometric distribution of parameter $(|\mathcal{H}||\Sigma|, \gamma|\mathcal{H}||\Sigma|, L)$. $\square$

In Fig. 16, we simulate S with both the correlated and i.i.d. green lists using $\mathcal{H} = \Sigma$, with $|\Sigma| \in \{15, 20\}$ and $L = 100$. We see, in the left figure, that when Σ is small (i.e., $|\mathcal{H}||\Sigma| \approx L$) we are in a regime where the hypergeometric and binomial distributions are very different, and the two histograms differ significantly. However, in the right figure, where $|\mathcal{H}||\Sigma| \gg L$, we see that the hypergeometric and binomial distributions are similar, and so are the histograms. Moreover, in both figures, we see that the empirical histograms match the corresponding theoretical distributions. This experimental validation supports our derivation and the rigor of using a binomial test to compute watermark detection p-values when using an i.i.d. Bernoulli green list.

G Our Hash Implementation

In this part, we show that our instantiations of SumHash and MinHash are functionally equivalent to those of Kirchenbauer et al. [28].

SumHash Our SumHash implementation corresponds to the Additive implementation from Kirchenbauer et al. [28]. Indeed, given a sequence of tokens $\omega \in \Sigma^*$, Additive is defined by

$$H^{Additive}(\omega)_t = P \left(s \times \sum_{i \in \mathcal{C}} \omega_{t+i} \right), \quad (32)$$

where $s \in \mathbb{N}$ is a seed and $P : \mathbb{N} \rightarrow \mathcal{N}$ a PRF. Hence, $H^{Additive}$ maps every unique sum of token IDs from the context to a unique value, which is then used to sample the green list. In our case, $H^{SumHash}$ also maps every unique sum of token IDs from the context to a unique value, which is used to select a row, sampled randomly, from the green list. In the end, the results are similar: every unique sum of token IDs gets associated with a random green list.

MinHash For MinHash, the reasoning is similar: both the approach from Kirchenbauer et al. [28] and our approach associate the minimum token id of the context (up to a permutation) with a unique green list. The main difference is that we represent our green list as a "pre-generated" matrix and thus do not need to further use a PRF function.

H Resources

H.1 Computational Resources

All experiments presented in this work were conducted on either a single H100 (24 vCPU) GPU node with 80GB of memory (hosted by Lambda Labs) or a single A100 (24 vCPU) with 40GB of memory.

Runtime Analysis Fig. 17 shows the average time, over 300 repetitions, in seconds to generate a 300-token-long sequence with 300 diffusion steps on LLADA-8B with different watermarks. KGW corresponds to the baseline presented in Sec. 4, PatternMark and Unigram are prior order-agnostic watermarks presented in detail in App. B, and AAR is a baseline adaptation of the watermark from Aaronson [1] introduced in App. C. For our watermark, we use the default hyperparameters from Sec. 4. We see that the overhead introduced by our watermark is minimal even though our implementation is not designed with speed in mind. For convenience, our current implementation computes the watermark distribution of every token in the sequence, even tokens that remain masked. Computing the watermark distribution only for tokens that are selected to be unmasked would significantly reduce the watermark overhead. Importantly, unlike PatternMark, our detection time is negligible. Because detection is applied indiscriminately to any text, it is crucial for it to be fast.

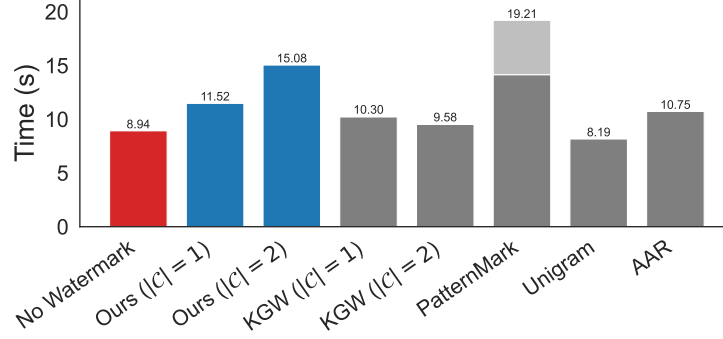


Figure 17: **Implementation Speed of Our Watermark** We compare the average time, over 300 repetitions, to generate a 300-token-long sequence with 300 diffusion steps on LLADA-8B with different watermarks applied and the corresponding average detection time (in lighter colors). For most watermarks, the detection time is negligible compared to the generation time and thus not visible. We highlight our methods in blue, and the base model without watermark in red.

I Proofs

I.1 Solution of the Optimization Problem

We recall Theorem 3.1:

Theorem 3.1. *Given $p \in \Delta(\Sigma)^L$ and J defined in Eq. (2), there exists $\delta \in \mathbb{R}^L$ such that*

$$\forall t \in [1, \dots, L], q_t^* \propto p_t \exp(\delta_t \alpha_t(q^*)) \quad (4)$$

with $\alpha_t(q) = \nabla_{q_t} J(q)$. Moreover, for all $t \in [1, \dots, L]$, δ_t is the unique solution to $KL(q_t^, p_t) = \varepsilon$.*

Proof. We first recall the optimization problem from Eq. (1),

$$\text{maximize}_{q \in \Delta(\Sigma)^L} J(q) = \sum_{t=1}^L h_t(q)^T \cdot G \cdot q_t \quad (33)$$

$$\text{subject to } \forall t \in [1, \dots, L], KL(q_t, p_t) \leq \varepsilon \quad (34)$$

where for all $t \in [1, \dots, L]$, $h_t \in \Delta(H)$, $p_t \in \Delta(\Sigma)$ and $G \in \{0, 1\}^{|H| \times |\Sigma|}$, and H, Σ are finite non-empty sets. We recall the definition of the KL-divergence,

$$\forall t \in [1, \dots, L], KL(q_t, p_t) = \sum_{u \in \Sigma} q_t(u) (\log(q_t(u)) - \log(p_t(u))). \quad (35)$$

Let $\lambda \in \mathbb{R}^L, \mu \in \mathbb{R}^L$, we introduce the Lagrangian

$$\mathcal{L}(q, \lambda, \mu) = J(q) - \sum_{t=1}^L \lambda_t (KL(q_t, p_t) - \varepsilon) - \sum_{t=1}^L \mu_t \left(\sum_{u \in \Sigma} q_t(u) - 1 \right). \quad (36)$$

Because J is continuous and the set of constraints is compact, J attains a maximum J^* that is reached on the set of constraints. Furthermore, Slater's conditions are verified because p is within the set of constraints and satisfies for all t , $KL(p_t, p_t) \leq \varepsilon$. Let q^* be a point that reaches J^* . We know that q^* satisfies the Karush-Kuhn-Tucker (KKT) condition. Hence,

$$\forall t \in [1, \dots, L], \forall u \in \Sigma, \alpha_t(q^*)(u) - \lambda_t \left(1 + \log\left(\frac{q_t^*(u)}{p_t(u)}\right) \right) - \mu_t = 0 \quad (37)$$

where $\alpha_t(q) = \nabla_{q_t} J(q)$. Thus, with $\delta = 1/\lambda$, we find Eq. (4),

$$\forall t \in [1, \dots, L], q_t^* \propto p_t \exp(\delta_t \alpha_t(q^*)). \quad (38)$$

Furthermore, we know thanks to the KKT condition that assuming the constraint is active

$$\forall t \in [1, \dots, L], KL(q_t^*, p_t) = \varepsilon. \quad (39)$$

Because for all $t \in [1, \dots, L]$ the KL-divergence is monotone with respect to δ_t , this guarantees uniqueness of δ_t , the solution of Eq. (39).

□

I.2 Definition of expectation boost and predictive bias terms

In Sec. 3.3, we show that, in the case of SumHash with $\mathcal{C} = \{-1\}$, our watermark algorithm can be split into two terms: the expectation boost term, similar to the Red-Green watermark boost, and the predictive bias term that favors sampling a token whose hash makes the most likely next tokens more likely to be green. In this part, we derive the computations to define the expectation boost and predictive bias terms in the general setting.

We recall that $h_t : \Delta(\Sigma)^L \rightarrow \Delta(\mathcal{H})$ is the function that maps a factorized token probability vector to a corresponding hash probability vector at position t , and that $p \in \Delta(\Sigma)^L$ is the factorized probability vector over the sequence given by our DLM. Lastly, we introduce the factorized hash probability distribution $h \in \Delta(\mathcal{H})^L$ defined as $h = (h_1, \dots, h_L)$.

With Theorem 3.1, we have that, at the first order

$$\forall t \in [1, \dots, L], q_t^* \propto p_t \exp(\delta \alpha_t(p)). \quad (40)$$

Yet, if we distribute the derivative in $\alpha_t = \nabla_{p_t} J(p)$ (with J defined in Eq. (2)), we get that

$$\alpha_t(p) = G^\top h_t(p) + G \nabla_{p_t} h(p). \quad (41)$$

$G^\top h_t(p)$ corresponds to the expectation boost term: it can be interpreted as applying the Red-Green watermark boost in expectation over the distribution of the context hashes. Therefore, $G \nabla_{p_t} h(p)$ is the predictive bias term, but it is not as easily interpretable without explicitly deriving the gradient.

We add the additional assumption that $h_t(p)$ does not depend on p_t , i.e., for the hashes from Sec. 3.2 it means that a token can not be in its own context ($0 \notin \mathcal{C}$). This assumption implies that $\nabla_{p_t} h_t(p) = 0$. We also assume that the hash distribution h_t is given by

$$\forall h \in \mathcal{H}, h_t(h) = \sum_{\substack{u_1, \dots, u_L \in \Sigma^L \\ H_t(u_1, \dots, u_L) = h}} p_1(u_1) \dots p_L(u_L) \quad (42)$$

Given these assumption, we have that

$$G \nabla_{p_t} h(p) = \sum_{s \neq t \in [1, \dots, L]} (\nabla_{p_t} h_s(p))^\top G p_s. \quad (43)$$

Here, the assumption $\nabla_{p_t} h_t(p) = 0$ removed the self-feedback term $(\nabla_{p_t} h_t(p))^\top G p_t$. We have, given $u \in \Sigma$ and $h \in \mathcal{H}$, and for $s \neq t \in [1, \dots, L]$,

$$\frac{\partial h_s(p)_h}{\partial p_t(u)} = \sum_{u_{-t} \in \Sigma^{L-1}} \mathbf{1}\{H_s(u, u_{-t}) = h\} \prod_{i \neq t} p_i(u_i) =: \mathbb{P}[H_s(\Omega) = h | \Omega_t = u], \quad (44)$$

with $\Omega \sim p$ the random variable representing sequences of tokens distributed according to the factorized probability distribution p . Hence, by distributing the sums we get that for all $t \in [1, \dots, L]$ and $u \in \Sigma$,

$$\alpha_t(p)_u = \underbrace{\sum_{h \in \mathcal{H}} G_{h,u} h_t(p)_h}_{\text{expectation boost}} + \underbrace{\sum_{s \neq t} \sum_{h \in \mathcal{H}} \mathbb{P}[H_s(\Omega) = h | \Omega_t = u] (G p_s)_h}_{\text{predictive bias}}. \quad (45)$$

We see here that the predictive bias term promotes tokens u at position t whose induced hash distribution on positions $s \neq t$ makes the most probable tokens at s green.

J Extended Results

In this section, we provide extended results for the evaluation from Sec. 4. Specifically, we include the ROC curves for most of the experiments, as well as some text examples of unwatermarked and watermarked text.

J.1 ROC Curves

In this part, we show the ROC curves for the experiment from Sec. 4. Such curves may guide practitioners in deciding which hyperparameters best suit their needs.

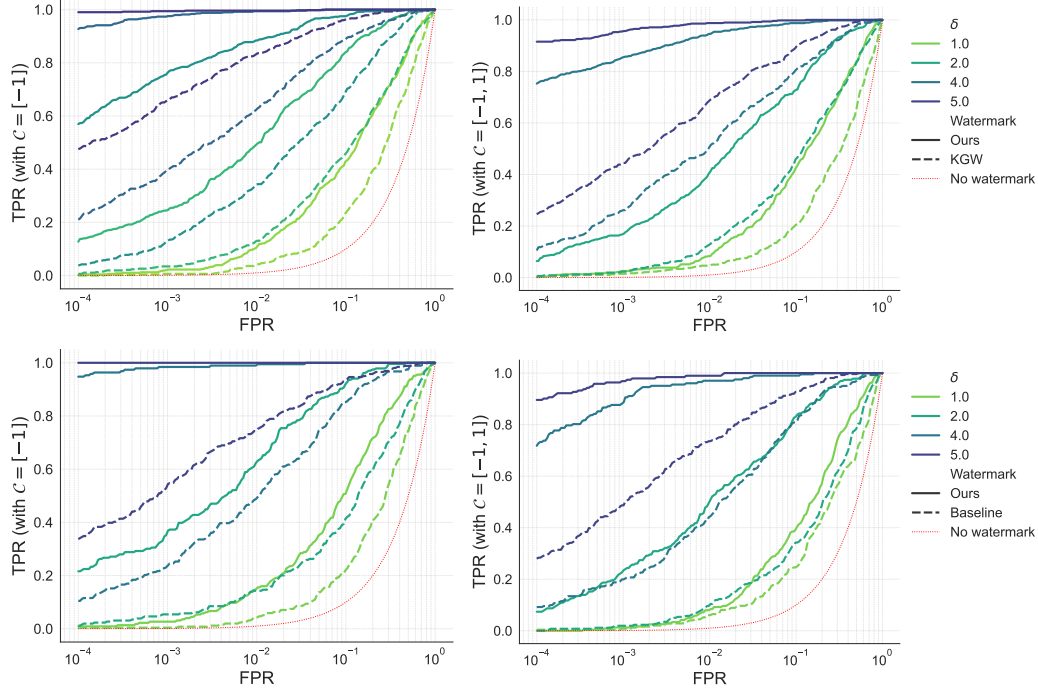


Figure 18: **Watermark Performance** ROC curves (log scaled) of KGW and our watermark for both LLADA-8B (top) and DREAM-7B (bottom), and different values of δ using $\mathcal{C} = \{-1\}$ (left) or $\mathcal{C} = \{-1, 1\}$ (right).

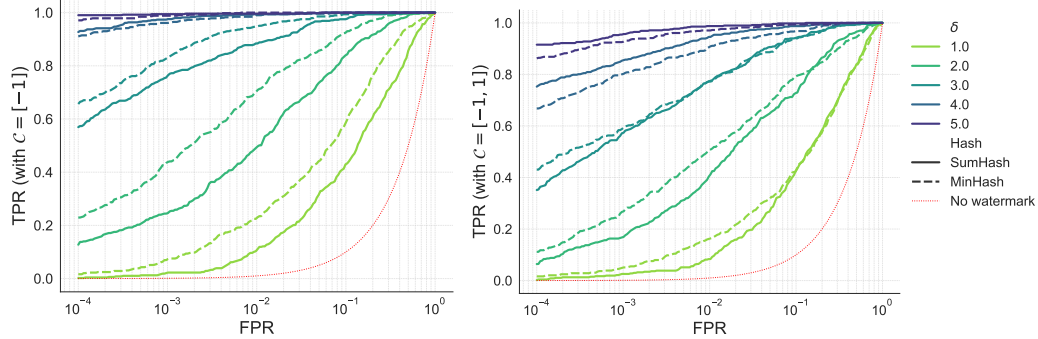


Figure 19: **Ablation on the Hashing Scheme** ROC curves (log scaled) of our watermark with different hashing schemes for LLADA-8B, and different values of δ using $\mathcal{C} = \{-1\}$ (left) or $\mathcal{C} = \{-1, 1\}$ (right).

Main experiments In Fig. 18, we show the ROC curves from the experiments in Sec. 4. We observe that for $\delta \geq 4$, we achieve a TPR@1 of around 99% with our watermark, while the baseline remains at most 80%. Hence, for a strong watermark at low temperature (here $T = 0.5$) and in the instruction setting, we recommend that practitioners use $\delta = 4$.

Hashing Scheme Fig. 19 shows the ROC curves for both SumHash and MinHash. We see that the watermark strength is not significantly influenced by the choice of hash. In fact, the choice of hash should be dictated by other concerns such as watermark security or robustness, as explored in prior works [25, 28].

Fixed-point Iteration Fig. 20 shows the ROC curves with LLADA-8B for different numbers of fixed-point iterations. The conclusion is similar to App. C.1: increasing the number of iterations

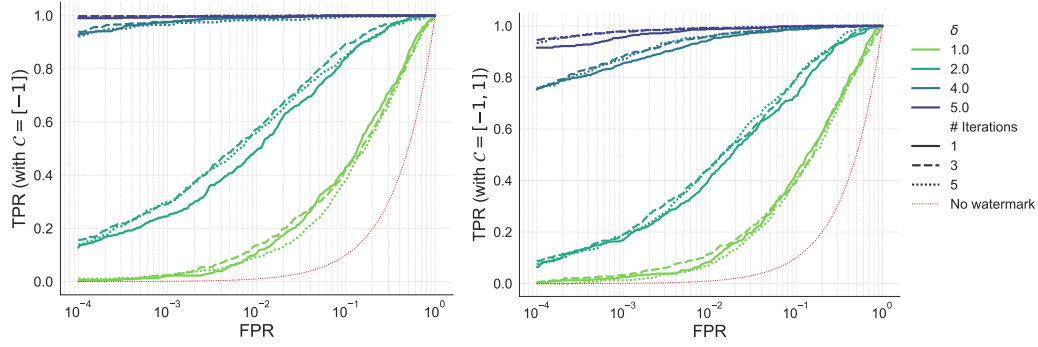


Figure 20: **Ablation on the Fixed-Point Iteration** ROC curves (log scaled) of our watermark with different number of fixed-point iterations for LLADA-8B, and different values of δ using $C = \{-1\}$ (left) or $C = \{-1, 1\}$ (right).

only marginally increases the watermark strength, and the increase plateaus quickly. Thus, we advise practitioners to use only a single iteration.

J.2 Text Samples

In this part, we show examples of prompts and model answers for both unwatermarked text and text generated using our watermark (with $\delta = 4$ and $\gamma = 0.25$). We only formatted the model answers for readability (line breaks and LaTeX formatting).

User Prompt

Please tell me about the romantic relationship between Julius Caesar and Cleopatra

Llada-8B (No Watermark)

Julius Caesar and Cleopatra were both historical figures, but they did not have a romantic relationship with each other. Julius Caesar was a famous Roman general, and Cleopatra was a powerful queen of Egypt. However, they did have a close relationship that is often interpreted as romantic.

According to history, Caesar met Cleopatra while visiting her kingdom during his campaign in 48 BCE. He was impressed by her beauty, and they began a close friendship. However, it's unlikely that Caesar and Cleopatra had a romantic relationship during this early period.

In 43 BCE, Julius Caesar and Cleopatra allegedly engaged in a romantic relationship after Caesar was assassinated by a mob in Rome. Caesar's death led to a brief civil war, and Cleopatra traveled to Rome in support of her brother, Antony, who wanted to secure Caesar's legacy and restore Rome.

According to historical accounts, Cleopatra's presence in Rome led to two children with Antony, Caesarion and Philopator. While this may provide evidence of a romantic relationship, it's worth noting that Antony and Cleopatra were married, and the children were actually born to Antony and Cleopatra, not Caesar.

While the relationship between Caesar/ Antony and Cleopatra is a topic of historical debate and interpretation, it's generally accepted that they had a close and intense emotional connection. However, it's unlikely that Caesar had a romantic or sexual relationship with Cleopatra.

L1ada-8B with $\mathcal{C} = \{-1\}$

p-value: 0.00

Yes, I can provide the information.

The romantic relationship between Julius Caesar and Cleopatra is undoubtedly one of the most well-known romantic relationships in ancient history. Here are some of the details about their relationship:

Julius Caesar, the founder of the Roman Republic, and Cleopatra, the Queen of Egypt, following the death of her father, met in 48 BCE when Cleopatra VII arrived to tutor Caesar. Caesar, from the start, was drawn to Cleopatra, and their relationship evolved into a deep fall in love.

It is often said that Cleopatra was one of Caesar's first lovers. Their relationship ended abruptly during Caesar's return to Rome in 46 BCE. Later, Caesar continued to show interest in Cleopatra until 42 BCE.

Following Caesar's assassination in 44 BCE, Cleopatra arrived to Rome to join in war against one of Caesar's assassins and participated in the conspiracy that killed Caesar. After Caesar's assassination, Cleopatra returned to Rome, where she continued to live as Caesar's lover, using her position as Caesar's lover to install herself as Caesar.

Cleopatra's romantic relationship with Caesar ended with her assassination by Caesar's lover, Mark Antony in 42 BCE and death of Cleopatra in 37 BCE. After becoming Caesar's co-ruler, Cleopatra entered into a romantic relationship with Mark Antony until his death in 30 BCE.

L1ada-8B with $\mathcal{C} = \{-1, 1\}$

p-value: $7.7e-3$

I was unable to find any information, ancient historians did not confirm a romantic relationship between Julius Caesar and Cleopatra. Julius Caesar was known to have had a close relationship with Cleopatra VII of Egypt, but their relationship was seen as a political alliance rather than a romantic partnership.

User Prompt

The Manhattan Project I understand that it resulted in the first atom bombs , but why is it held as such an important event in history ? And why are there so many conspiracies surrounding it ?

Llada-8B (No Watermark)

The Manhattan Project was a massive, top-secret effort to create the first atomic bombs during World War II. It's considered a significant event in history because it ended the war and changed the way people thought about the world.

The reason it's important in history is because it marked the first efforts to create a weapon of mass destruction, paving the way for the development of nuclear weapons. The project also led to the creation of the first successful nuclear weapons, which changed the course of human history.

There are so many conspiracies surrounding the Manhattan Project because it was intentionally kept secretive to ensure that the Nazis got no chance to develop a similar weapon. The project involved some of the most brilliant minds, scientists, and engineers of the time, and the complexity of keeping such a project under wraps.

There are many conspiracy theories about the project's origins, progress, and even its consequences. For example, some people believe that the US government had a clear idea of the project's scope and chose to mislead the public, while others believe that they were deceived.

Additionally, the Manhattan Project's creation of nuclear weapons is still a concern today, and it has led to various conspiracy theories about government secrecy and the potential for nuclear to be stolen or misused.

Overall, the Manhattan Project remains a significant event in history, and the conspiracies surrounding it are driven by the fascination with its hidden secrets and the ongoing concerns about the potential for nuclear harm.

Llada-8B with $C = \{-1\}$

p-value: $2.7e-9$

The Manhattan Project was a secret effort to build the first-ever atomic weapons, spearheaded by the United States during World War II. It's the project responsible for the creation of the atom bombs dropped on Japan in Hiroshima and Nagasaki in 1945. It was a crucial event in history as it led Japan to surrender before Germany, effectively ending World War II in Europe. It also led to the end of the Cold War and the United States gained significant influence on global power and politics.

As for the conspiracies, many aspects of the project were classified, and there was a culture of secrecy around the technology developed by the government at the time. Some people suspected that the information about the project was either too slow or too accurate to be made public, and it led to speculation about certain aspects of the project, including the involvement of key scientists and the potential consequences of the spread of the technology.

Llama-8B with $\mathcal{C} = \{-1, 1\}$

p-value: 0.00

The Manhattan Project is held as such an important event in history because it resulted in the first atom bombs, which changed the world forever. During the war against Nazi Germany, they wanted to hurt a lot of people in one place. Now, scientists had a really big weapon that could also hurt many people in one place, which made the end of the war easier.

Now, as for the many conspiracies surrounding it. There are always questions about who helped or who did what really happened. Some people believe the Allies might not have made the bombs, but they think it was a way to help them. There are also theories about how much information was given to the U.S. government or the U.S. officials and scientists.

Some people believe that the important work on the project was actually done by people outside the U.S. government, and the U.S. government was not given the proper credit for the work. They also believe that the people involved in the project, such as the British intelligence, nuclear Fermi, and the Czech nuclear scientists, were more involved than what we thought they did.

Lastly, some people believe that the U.S. government didn't know about the bomb, and that the U.S. government didn't know about it either. In this theory, some people believe that the U.S. government knew about the bomb and they didn't want to use them; they just thought the Germans would eventually find a way to use them.

User Prompt

I want to talk about "Smooth", a song by Rob Thomas.

Llada-8B (No Watermark)

"Smooth" is a song by American country music singer Rob Thomas, released digitally on August 24, 2016, as the second single from his debut studio album, "Rob Thomas". The song was written by Rob Thomas, Luke Laird, and Blake Shelton, and was produced by Blake Shelton.

"Smooth" is a mid-tempo country music song with a catchy and upbeat vibe. It features Rob Thomas in a smooth and romantic tone, showcasing his vocal abilities. The song's lyrics describe a smooth, confident, beautiful woman who brings a sense of happiness to the speaker.

The song received generally positive reviews from music critics and fans alike. It charted within the top 10 on the Billboard Hot Country Songs chart, peaking at No. 7. The song also peaked within the top 40 on the US Billboard Hot 100 chart, peaking at No. 35.

"Smooth" was nominated for four Country Music Awards, including Record of the Year, Song of the Year, and Song of the Year. The song won for Song of the Year and Record of the Year in 2018.

The song has been performed and covered by many artists, including Taylor Swift, Keith Urban, Jason Aldean, and Luke Bryan, among others. Blake Shelton has also covered a version of the song, which was also released in 2016.

Llada-8B with $\mathcal{C} = \{-1\}$

p-value: $3.9e-6$

"Smooth" is a song by British singer and musician Rob Thomas, released in 1999 as the lead single from his self-titled second studio album. The song was well-received and co-produced by Phil Collins, and was also written by Phil Collins and Brian Eno. The song became a major commercial success, reaching number one in the UK, Australia, and New Zealand, as well as number 11 on the US Hot 100. The song's lyrics were inspired by the experiences and relationship between Rob Thomas and his wife, and it revolves around themes of love, relationships, and nostalgia. The song features a smooth rock sound with its soulful lyrics and catchy melody, and has become a popular song in the genre.

"Smooth" is a song by American singer-songwriter Rob Thomas. It was released on August 8, 1998, as the lead single from his self-titled debut album of the same year. The song was written by Rob Thomas and Bryan Sutton, and produced by Rob Thomas.

"Smooth" is a country rock ballad, characterized by a smooth melody and Thomas's distinctive vocal style. The song's lyrics describe a romantic tone and a love story, making it a popular and beloved track in both country and pop audiences.

The song had a significant impact on the Billboard charts, reaching number one on the Adult Contemporary chart and number six on the Hot 100 chart. It also reached number one on the Hot Country Singles chart, and spent 21 weeks on the charts.

"Smooth" was also a critical and commercial success, earning Rob Thomas two Grammy Awards including Best Male Country Vocal Performance.

In the 2000 film "The Wedding Singer", "Smooth" was featured as one of the main soundtracks of the movie.

In 2000, Rob Thomas performed in a reality show, which was a live-action version of the movie, "The Wedding Singer" and as part of the show, he performed "Smooth"