TREE-BASED SPECULATIVE DECODING FOR WATERMARK GENERATION VIA VOCABULARY PARTITIONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Watermarking helps identify LLM-generated text for data provenance, but its practical deployment requires fast generation. Tree-based speculative decoding could accelerate watermark generation by proposing multiple candidate continuations at each step, so that another candidate can still be accepted if one is rejected. However, applying it to watermarking is challenging: different branches use different pseudorandom sources, making the watermark signal harder to detect. We address this challenge by deterministically partitioning the vocabulary into disjoint subsets and sampling one draft token from each subset to form a distinct branch of the draft tree. The subset containing an observed token identifies the only draft branch that could have produced it, leaving that branch and residual sampling as the two possible sources. We prove that the resulting algorithm preserves the target model's output distribution and achieves a draft acceptance probability at least as high as that of single-sequence speculative decoding. Experiments with Gumbel-max and SynthID watermarks show improved generation throughput while largely preserving detection power. We also show that our method can benefit from advances in drafter design, with an EAGLE-style draft head enabling additional speedups.

1 INTRODUCTION

In the era of generative AI, establishing data provenance for large language models (LLMs) has become increasingly important due to the misinformation and social risks that misuse may incur (Weidinger et al., 2022; Starbird, 2019; Milano et al., 2023; Shumailov et al., 2024; Anthropic, 2026). Watermarking has emerged as a promising approach for this purpose. By carefully coupling the sampling process with recoverable pseudorandom variables (Aaronson, 2023; Kirchenbauer et al., 2023; Kuditipudi et al., 2024), watermark signals can be embedded and reliably detected. However, standard watermarked generation inherits the sequential bottleneck of autoregressive decoding, producing tokens one at a time.

Speculative decoding (Chen et al., 2023; Leviathan et al., 2023) accelerates autoregressive generation by allowing a lightweight draft model to propose tokens that a target model verifies in parallel. By accepting and, when necessary, correcting the draft, it can generate multiple tokens per target-model forward pass while preserving the target distribution. Sampling efficiency, defined as the acceptance rate of draft tokens, is a key factor in the achieved speedup. Recent work (He et al., 2026; Hu & Huang, 2024b) extends this approach to accelerate watermark generation and studies the trade-off between sampling efficiency and watermark strength. These methods, however, remain limited by their single-sequence structure: at each step, the draft model explores only one continuation, with no alternative draft candidate to try after a rejection. For non-watermarked generation, tree-based speculative decoding overcomes this limitation by expanding the draft from a single sequence into multiple branching continuations (Miao et al., 2024; Cai et al., 2024; Li et al., 2024a; Spector & Re, 2023; Yang et al., 2025b; Jeon et al., 2024). The draft tree provides multiple candidate continuations at each branching point, allowing verification to continue along an alternative branch after a rejection and thereby further improving sampling efficiency and accelerating generation.

These advances in tree-based speculative decoding suggest a plausible way to further accelerate watermarked generation and motivate the question in this work: *How can tree-based speculative decoding accelerate watermark generation while preserving unbiasedness and maintaining detectability?* Extending tree-based speculative decoding to watermark generation is non-trivial, as it requires

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

meeting multiple tightly coupled requirements. (**R1**) It must propose diverse draft continuations for a given prefix. This is challenging because using the same watermark pseudorandomness for a given prefix tends to produce the same draft token, rather than diverse candidates (Kuditipudi et al., 2024). (**R2**) It must preserve the target distribution (i.e., unbiasedness) to maintain generative fidelity. (**R3**) It must limit the degradation of watermark detectability. (**R4**) It must improve sampling efficiency over single-sequence speculation by making better use of the generated draft candidates.

These requirements are intertwined: using independent pseudorandom variables helps generate diverse candidates across branches, but it also complicates detection. Because the detector sees only the final text, it cannot tell which branch produced each token. This ambiguity makes it harder to match tokens with the pseudorandom variables carrying their watermark signals and can substantially weaken detection.

Contributions. We address these challenges with a vocabulary partition strategy and propose a tree-based speculative decoding framework for watermark generation. Our contributions are as follows.

- **Tree-based speculative decoding for watermarking**. We introduce a deterministic vocabulary partition strategy that samples from disjoint subsets to propose diverse draft continuations for a given prefix (**R1**). The detector can recover the partition and identify the candidate branch for each token. For a draft tree with k branches at each node, this reduces the number of possible sources for each observed token from $k+1$ to 2, which helps to achieve (**R3**).
- **Theoretical guarantees**. We prove that our algorithm preserves the target distribution (**R2**) and improves sampling efficiency over single-sequence speculation (**R4**).
- **Experimental validation**. Experiments on synthetic distributions and real generation tasks validate the sampling-efficiency gains, while evaluations with Gumbel-max (Aaronson, 2023) and SynthID (Dathathri et al., 2024) show largely preserved detectability. We also show that our framework is compatible with advanced drafter designs: an EAGLE-style draft head (Li et al., 2024a) achieves up to $1.92\times$ the throughput of single-sequence speculative decoding using a Qwen3-0.6B drafter.

Paper organization. Section 2 reviews preliminaries on watermarking and speculative decoding. Section 3 presents our tree-based speculative decoding algorithm along with its theoretical guarantees (unbiasedness and improved sampling efficiency). Section 4 reports empirical results on synthetic distributions and real generation tasks. Related work is deferred to App. A.

2 PRELIMINARIES

Watermark embedding and detection. Let $\boldsymbol{P}$ denote the (conditional) token distribution over the vocabulary $\mathcal{W}$, and write $P(w_t)$ as shorthand for $P(w_t \mid w_{<t})$ when the conditioning context is clear. Watermarking can be interpreted as reweighting $\boldsymbol{P}$ using pseudorandomness (Hu et al., 2024). In particular, a token w is sampled from a modified distribution $\boldsymbol{P}_\zeta = \mathcal{S}(\boldsymbol{P}, \zeta)$, where ζ is a pseudorandom variable and $\mathcal{S}$ is a decoding function that embeds the watermark signal into the generated text. An unbiased watermark recovers the original distribution when averaged over pseudorandomness, i.e., $\mathbb{E}_\zeta[\boldsymbol{P}_\zeta] = \boldsymbol{P}$. During detection, the pseudorandom variable ζ can be exactly recovered from the observed text, and we test whether an observed token w is generated from the original distribution $\boldsymbol{P}$ or its watermarked counterpart $\boldsymbol{P}_\zeta$. Details of the Gumbel-max (Aaronson, 2023) and SynthID (Dathathri et al., 2024) watermarks are provided in App. B.

Accelerating watermark generation with speculative decoding. Speculative decoding Hu & Huang (2024b); Dathathri et al. (2024); He et al. (2026) accelerates watermark generation via a two-stage procedure. Given a prefix $w_{<t}$, lookahead depth L, and two independent pseudorandom variables $\zeta^{(0)}$ and $\zeta^{(1)}$, the procedure consists of:

- *Draft generation and evaluation*. A lightweight draft model $\boldsymbol{Q}$ autoregressively generates L watermarked tokens $\tilde{w}_{t:t+L-1}$, where $\tilde{w}_{t+i} \sim \boldsymbol{Q}_{\zeta^{(1)}}(\cdot \mid w_{<t}, \tilde{w}_{t:t+i-1})$. The target model $\boldsymbol{P}$ then evaluates these tokens in parallel by computing their probability distributions $\boldsymbol{P}(\cdot \mid w_{<t}, \tilde{w}_{t:t+i-1})$.
- *Draft token verification*. Each draft token is accepted with probability $\min\{1, P(\tilde{w}_t)/Q(\tilde{w}_t)\}$. If accepted (i.e., $w_t = \tilde{w}_t$), the algorithm proceeds to verify $\tilde{w}_{t+1}$; otherwise, it samples w_t from $[\boldsymbol{P} - \boldsymbol{Q}]_{+,\zeta^{(0)}}$, the normalized positive part of $\boldsymbol{P} - \boldsymbol{Q}$ with watermarking controlled by $\zeta^{(0)}$.

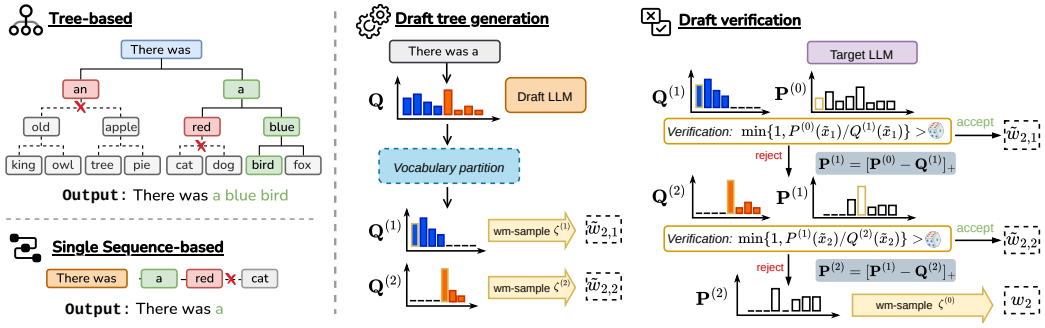


Figure 1: Overview of tree-based speculative decoding for watermark generation. **Left**: standard (sequence-based) versus our tree-based decoding. **Middle**: tree construction via the vocabulary partition strategy for the $k = 2$ case. **Right**: an illustration of draft verification.

This draft-and-verify procedure provably preserves the target distribution whenever $\zeta^{(0)}$ and $\zeta^{(1)}$ are independent (Dathathri et al., 2024; He et al., 2026), thereby accelerating watermark generation without compromising unbiasedness. However, the achievable acceleration is limited by the single-sequence structure: once a draft token is rejected, all subsequent tokens in the lookahead sequence are discarded (see the left panel of Fig. 1). We refer to this algorithm as standard speculative decoding, summarized in Alg. 1.

3 METHOD

At a high level, our method extends standard draft-and-verify speculative decoding (Chen et al., 2023; Leviathan et al., 2023) to a tree-based structure that facilitates watermark generation and detection, as illustrated in Fig. 1.

Draft tree generation. Instead of generating draft tokens along a single chain, we construct a draft tree to produce multiple consecutive tokens (left panel of Fig. 1). Starting from the root (i.e., the prompt or current generation history), the draft model recursively generates tokens in a tree-structured, autoregressive manner. The key idea is *vocabulary partition*: at each step, we partition the vocabulary into k disjoint subsets (branches) using a deterministic rule based on the current prefix, and sample a draft token from each subset to form a branch of the tree (middle panel of Fig. 1). Each sampled token becomes the parent for the next step. Repeating this procedure for L steps yields a tree of depth L and branching factor k. The target model then evaluates all candidate tokens in parallel.

Draft token verification. Given the draft tree, the target model verifies its candidate tokens sequentially across branches (right panel of Fig. 1). At each step, if a token is accepted, it is added to the generation history and verification proceeds along its child branch; otherwise, the algorithm moves to the next sibling branch at the same level and continues the verification process. If all sibling branches are rejected, the remaining probability mass lies in the residual, and a token is sampled directly from a particular residual distribution (which is derived from the target model to preserve unbiasedness). As a result, the algorithm can accept a sequence of tokens along a subtree, rather than a single chain as in standard speculative decoding, thereby improving sampling efficiency.

As we will show, this design simultaneously addresses all four requirements (**R1**–**R4**) in Section 1. In the following, we describe the algorithm in detail (Section 3.1), establish its theoretical guarantees (Section 3.2), and analyze the practical effects governed by the branching factor k (Section 3.3).

3.1 ALGORITHM DESCRIPTION

Main motivation: pseudorandom independence introduces detection ambiguity. Standard speculative decoding (Section 2) uses two independent pseudorandom variables—one for draft generation and one for residual sampling—to preserve unbiasedness. Independence is crucial: sharing a pseudorandom variable would couple these steps and break unbiasedness. Extending this to a k-branch tree requires assigning independent pseudorandom variables to each draft branch and the residual distribution. However, this introduces a major challenge for detection: *the detector does not know which part of the generation process produced each observed token*. In particular, a token

Pseudocode 1 Draft tree generation via vocabulary partition (brief form).

Require: Draft model $\boldsymbol{Q}$, branching factor k, lookahead depth L, root prefix $w_{<t}$, partition rule $\mathcal{A}$, pseudorandom variables $(\zeta^{(1)}, \ldots, \zeta^{(k)})$.

for $d = 1, \ldots, L$ **do**
  for each node with its prefix w_{pre} at depth $d-1$ ($w_{\text{pre}} = w_{<t}$ when $d = 1$) **do**
    $(\mathcal{W}_1, \ldots, \mathcal{W}_k) \leftarrow \mathcal{A}(w_{\text{pre}})$ ▷ vocabulary partition: $\mathcal{W}_i$ disjoint, $\bigcup_i \mathcal{W}_i = \mathcal{W}$
    For $i = 1, \ldots, k$, sample $\tilde{w}_{(w_{\text{pre}}, i)} \sim \boldsymbol{Q}^{(i)}_{\zeta^{(i)}}$, with $\boldsymbol{Q}^{(i)}$ from Eq. (1).
    For each $\tilde{w}_{(w_{\text{pre}}, i)}$ create a child node at depth d with prefix: $(w_{\text{pre}}, \tilde{w}_{(w_{\text{pre}}, i)})$.
  end for
end for
return the resulting k-ary draft tree rooted at $w_{<t}$.

may arise from any of the k branches or from the residual, resulting in $k+1$ possible sources. This ambiguity degrades detection power (He et al., 2026), as the number of possible sources grows with the factor k. To enable reliable detection, it is thus necessary to introduce additional structure that reduces this ambiguity. Our vocabulary partition strategy is designed for this purpose.

Vocabulary partition for draft tree generation. The draft tree is constructed recursively from the root prefix $w_{<t}$, where each node corresponds to a prefix and the prefix extends along each path. To grow the tree at a given node, we first partition the vocabulary into k disjoint subsets. Formally, let the current prefix be w_{pre}.[1] We first deterministically partition $\mathcal{W}$ as

$$(\mathcal{W}_1, \ldots, \mathcal{W}_k) \;=\; \mathcal{A}(w_{\text{pre}}), \qquad \mathcal{W}_i \cap \mathcal{W}_j = \emptyset \; (i \neq j), \quad \bigcup_{i=1}^{k} \mathcal{W}_i = \mathcal{W},$$

where $\mathcal{A}$ is a pseudorandom rule[2] known to both the model provider and the detector. Since $\mathcal{A}$ depends on the prefix, the partition can vary during generation, but remains deterministic for a given prefix and can be exactly reconstructed by the detector. For each subset $\mathcal{W}_i$, we then restrict and renormalize $\boldsymbol{Q}(\cdot \mid w_{\text{pre}})$ to obtain

$$\boldsymbol{Q}^{(i)}(\cdot \mid w_{\text{pre}}) = \{Q^{(i)}(w \mid w_{\text{pre}})\}_{w \in \mathcal{W}} \quad \text{with} \quad Q^{(i)}(w \mid w_{\text{pre}}) \;=\; \frac{Q(w \mid w_{\text{pre}})\, \mathbf{1}_{w \in \mathcal{W}_i}}{\sum_{v \in \mathcal{W}_i} Q(v \mid w_{\text{pre}})}. \tag{1}$$

For each subset $\mathcal{W}_i$, a watermarked child token $\tilde{w}_{(w_{\text{pre}}, i)}$ is then sampled using an independent pseudorandom variable $\zeta^{(i)}$ according to $\tilde{w}_{(w_{\text{pre}}, i)} \sim \boldsymbol{Q}^{(i)}_{\zeta^{(i)}} := \mathcal{S}(\boldsymbol{Q}^{(i)}, \zeta^{(i)})$, where $\mathcal{S}$ is any decoding function that introduces unbiased watermarking, i.e., $\mathbb{E}_{\zeta^{(i)}}[\boldsymbol{Q}^{(i)}_{\zeta^{(i)}}] = \boldsymbol{Q}^{(i)}$. Since the supports are disjoint, the k children at each node are distinct, enabling diverse tree construction (**R1**). Applying this procedure recursively, each child becomes the root of a subtree. Unrolling from the history $w_{<t}$ for L steps yields a k-ary draft tree of depth L. A brief description is given in Pseudocode 1, with the full implementation deferred to Alg. 3.

Draft tree verification. Given a draft tree, the target model first computes the logits for all draft candidates in parallel via tree-based attention (Miao et al., 2024). We then verify the k branches at each tree level sequentially (from 1 to k, as the branch indices are also deterministic and recoverable), following the same accept–reject mechanism as in Section 2, while updating the residual distribution across branches. More specifically, let w_{cur} denote the current verified prefix (initially $w_{<t}$) and initialize $\boldsymbol{P}^{(0)} = \boldsymbol{P}(\cdot \mid w_{\text{cur}})$. For each branch $i = 1, \ldots, k$, the candidate $\tilde{w}_{(w_{\text{cur}}, i)}$ (the draft token sampled from $\boldsymbol{Q}^{(i)}_{\zeta^{(i)}}$ at w_{cur}) is accepted with probability $\min\{1, P^{(i-1)}(\tilde{w})/Q^{(i)}(\tilde{w})\}$.[3] If accepted, the algorithm outputs $\tilde{w}_{(w_{\text{cur}}, i)}$ and proceeds to the next tree level along the i-th child. Otherwise, the residual distribution is updated as

$$\boldsymbol{P}^{(i)} \;=\; [\boldsymbol{P}^{(i-1)} - \boldsymbol{Q}^{(i)}]_+,$$

where $[\cdot]_+$ denotes the normalized positive part, and verification proceeds to branch $i+1$. If all k branches are rejected, a token is sampled from the residual $\boldsymbol{P}^{(k)}$ using $\zeta^{(0)}$, i.e., $w \sim \mathcal{S}(\boldsymbol{P}^{(k)}, \zeta^{(0)})$. The full verification procedure is summarized in Alg. 4.

[1] At the root, $w_{\text{pre}} = w_{<t}$; at deeper levels, w_{pre} extends $w_{<t}$ with the (tentative) draft tokens along the path.

[2] We set $\mathcal{A}$ as a deterministic hash function of the prefix in our implementation; see App. E.1 for details.

[3] Following He et al. (2026), we use independent pseudorandom acceptance thresholds across branches to preserve strong token–pseudorandom dependence; see App. C.4 for details.

Detection and the benefits of vocabulary partition. As discussed earlier, during detection, the pseudorandom variables associated with different branches can be recovered, but the generating source of each token is ambiguous. In general, a token may arise from any of the k branches or from the residual, resulting in $k+1$ possible sources. Our vocabulary partition strategy reduces this ambiguity. Because the supports of different branches are disjoint and exactly known from the prefix, observing $w_t \in \mathcal{W}_i$ implies that the token must have been generated either from branch i or from the residual. Consequently, the ambiguity is reduced from $k+1$ to 2 possible sources. This level of ambiguity matches that of standard speculative decoding (He et al., 2026) and satisfies the detectability requirement (**R3**). Since branch identification depends on the vocabulary partition, this reduction in source ambiguity is independent of the architecture producing $\boldsymbol{Q}$. It therefore applies to both standalone draft models and specially designed draft heads such as EAGLE (Li et al., 2024a), allowing watermarked generation to further benefit from advances in drafter design. The full decoding procedure is summarized in Alg. 2.

3.2 Theoretical Guarantees

In this subsection, we provide a theoretical analysis of our method, showing that it maintains unbiasedness (**R2**) and improves sampling efficiency over standard speculative sampling (**R4**). Due to space limitations, all proofs are deferred to App. C.

Setup and notation. Without loss of generality, we focus on a single generation step at one tree level, since the same procedure is applied at each level with a fresh verified prefix. Fix a prefix $w_{<t}$, and let $\boldsymbol{P} := \boldsymbol{P}(\cdot \mid w_{<t})$ and $\boldsymbol{Q} := \boldsymbol{Q}(\cdot \mid w_{<t})$ denote the target and draft next-token distributions over the vocabulary $\mathcal{W}$. Let $(\mathcal{W}_1, \ldots, \mathcal{W}_k) = \mathcal{A}(w_{<t})$ be the induced vocabulary partition, with branch distributions $\boldsymbol{Q}^{(i)}$ defined in Eq. (1). The pseudorandom variables $\zeta^{(0)}, \zeta^{(1)}, \ldots, \zeta^{(k)}$ are mutually independent. We denote by $\boldsymbol{P}'_\zeta$ the distribution of the final output token produced by the verification procedure, where $\zeta := (\zeta^{(0)}, \zeta^{(1)}, \ldots, \zeta^{(k)})$ collects all pseudorandom variables.

Unbiasedness. We first establish that our method preserves the target distribution, i.e., it is unbiased and does not degrade the output quality of the target language model (**R2**). A key condition for this result is the mutual independence of $\zeta^{(0)}, \ldots, \zeta^{(k)}$. Since the accept–reject decision at each branch depends on an independent pseudorandom variable, the probability of each verification path can be computed exactly, and expectations can be taken accordingly. As a result, the sequential verification procedure remains unbiased, in the same sense as standard speculative decoding (He et al., 2026).

Theorem 3.1 (Unbiasedness)**.** *Under the setup above, the verification procedure preserves the target distribution in expectation:* $\mathbb{E}_\zeta[\boldsymbol{P}'_\zeta] = \boldsymbol{P}$.

Improved sampling efficiency. We next analyze sampling efficiency, which measures how often draft tokens are accepted by the target model. In the presence of watermarking, the acceptance outcome also depends on pseudorandom variables ζ, so we measure efficiency in expectation over ζ. Following (He et al., 2026; Hu & Huang, 2024b), we define sampling efficiency for a single generation step as follows.

Definition 3.1 (Sampling efficiency)**.** *Under the setup above, let w_t denote the output token and let $\mathcal{D}_\zeta = \{\tilde{w}_{(w_{<t},\, i)}\}_{i=1}^{k}$ denote the k watermarked draft candidates. The sampling efficiency is*

$$A_k := \mathbb{P}(w_t \textit{ is accepted from one of the } k \textit{ draft candidates}) = \mathbb{E}_\zeta[\mathbb{P}(w_t \in \mathcal{D}_\zeta \mid w_{<t},\, \boldsymbol{P},\, \boldsymbol{Q})]\,.$$

In other words, the sampling efficiency is the probability that at least one of the k draft candidates is accepted by the target model. This definition is natural, as only accepted draft tokens contribute to acceleration: accepted tokens allow multiple tokens to be produced from a single target model evaluation, while rejection limits this gain. Thus, A_k directly quantifies the expected computational savings per step. In standard speculative decoding ($k = 1$), the sampling efficiency is (at most) $A_1 = \sum_w \min\{P(w), Q(w)\} = 1 - \mathrm{TV}(\boldsymbol{P}, \boldsymbol{Q})$ (Hu & Huang, 2024b; He et al., 2026; Dathathri et al., 2024), where TV denotes the total variation distance. Our second result shows that the vocabulary partition strategy improves this baseline for any $k \geq 2$, regardless of the partition.

Theorem 3.2 (Improved sampling efficiency)**.** *Under the setup above, let A_k denote the sampling efficiency. Then for any vocabulary partition $(\mathcal{W}_1, \ldots, \mathcal{W}_k)$ with $Q(\mathcal{W}_i) > 0$ for all i,*

$$A_k \;\geq\; 1 - \mathrm{TV}(\boldsymbol{P}, \boldsymbol{Q}) = A_1.$$

*In other words, the sampling efficiency of our method with $k \geq 2$ is at least as high as that of standard speculative decoding, thus satisfying (**R4**).*

So far, we have shown that our decoding method (in Alg. 2) with the vocabulary partition strategy addresses all four requirements (**R1**–**R4**). Moreover, Thm. 3.2 has a practical implication: our vocabulary partition strategy improves the acceptance rate over single-sequence speculation using only a partition of the vocabulary—no additional draft models are required. Notably, the sampling-efficiency advantage also applies to standard (non-watermarked) text generation, as the result relies on the partition structure and does not invoke watermark-specific properties (see App. C.2 for details and App. E.4 for results).

3.3 Effects of the Branching Factor

The branching factor k is an important hyperparameter that controls the diversity of the draft tree and influences sampling efficiency, computational cost, and watermark detectability. We now examine its impact on these three aspects.

Sampling efficiency improves with finer partitions. Theorem 3.2 shows that for any $k \geq 2$, the sampling efficiency A_k is at least $1 - \mathrm{TV}(\boldsymbol{P}, \boldsymbol{Q})$, the baseline of standard speculative decoding ($k = 1$). A natural question is whether increasing k further improves A_k. The answer is affirmative when finer partitions are obtained by subdividing coarser ones.

Theorem 3.3 (Improved sampling efficiency under partition refinement)**.** *Under the setup of Theorem 3.2, let $\Pi = (\mathcal{W}_1, \ldots, \mathcal{W}_k)$ be an ordered partition of $\mathcal{W}$ with sampling efficiency A_k (Def. 3.1), and let $\Pi' = (\mathcal{W}_{1,1}, \ldots, \mathcal{W}_{1,m_1},\ \mathcal{W}_{2,1}, \ldots, \mathcal{W}_{2,m_2},\ \ldots,\ \mathcal{W}_{k,1}, \ldots, \mathcal{W}_{k,m_k})$ be an ordered refinement of Π, meaning $\mathcal{W}_i = \bigcup_{j=1}^{m_i} \mathcal{W}_{i,j}$ for each i and the block order is preserved. Let $A_{k'}$ denote the sampling efficiency under Π', where $k' = \sum_{i=1}^{k} m_i$. Then*

$$A_{k'} \ \geq \ A_k.$$

Thm. 3.3 shows that sampling efficiency improves when a partition is refined, i.e., when any subset is subdivided into finer pieces while preserving the block ordering. As a direct consequence, Thm. 3.2 follows. Indeed, when $k = 1$, the partition is trivial, i.e., $\{\mathcal{W}\}$ which consists of the entire vocabulary. Any partition with $k \geq 2$ is therefore a refinement of this trivial partition, and the result $A_k \geq A_1$ follows immediately from Thm. 3.3.

Remark 3.1. The refinement condition is essential: for two arbitrary partitions of different sizes, sampling efficiency is not necessarily monotone in k. Empirically (Fig. 2), however, the average sampling efficiency over random partitions tends to increase with k, suggesting that monotonic improvement may still hold with high probability even without the refinement condition.

Entropy redistribution and its impact on detectability. The vocabulary partition strategy can affect watermark detectability by altering the entropy of the draft distribution across branches. Since watermarked tokens are sampled from the per-branch distributions $\boldsymbol{Q}^{(i)}$, any change in their entropy directly impacts the strength of the watermark signal. In particular, applying a partition redistributes the entropy of $\boldsymbol{Q}$ across branches. By the entropy chain rule,

$$\mathrm{Ent}(\boldsymbol{Q}) = -\sum_{i=1}^{k} \alpha_i \log \alpha_i + \sum_{i=1}^{k} \alpha_i \, \mathrm{Ent}(\boldsymbol{Q}^{(i)}), \qquad \alpha_i = \sum_{w \in \mathcal{W}_i} Q(w),$$

which separates the total entropy into a branch-selection term and the average entropy within each branch. As k increases, this redistribution becomes more pronounced, and some branch distributions $\boldsymbol{Q}^{(i)}$ may become more concentrated (i.e., lower entropy). For unbiased watermarks, lower entropy weakens the watermark signal. This is because when the distribution becomes more deterministic, the pseudorandomness has less influence on token selection, making the signal harder to detect (Li et al., 2025; He et al., 2025). Consequently, we expect a mild reduction in detectability as k increases.

Together, these analyses explain how partition refinement can improve sampling efficiency and how entropy redistribution may affect watermark detectability.

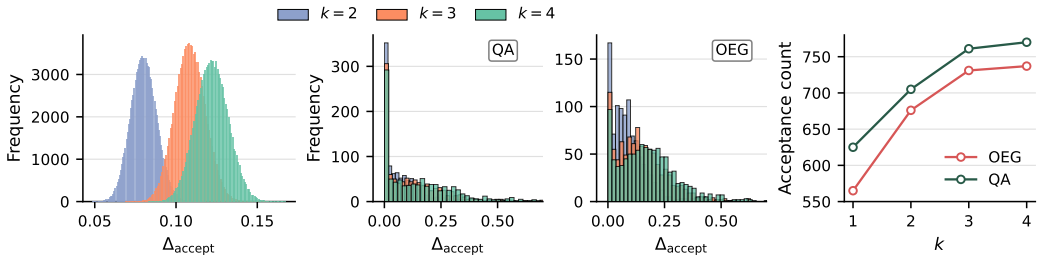


Figure 2: One-step sampling efficiency gain $\Delta_{\text{accept}} = A_k - A_1$ of our method over the baseline. **Left**: histogram over 10^6 randomly generated distribution pairs. **Middle**: histograms over 10^3 real next-token distributions from `Qwen3-32B/Qwen3-0.6B`. **Right**: total accepted tokens (out of 10^3) versus k for question answering (QA) and open-ended generation (OEG).

4 Experiments

We empirically evaluate our method (Alg. 2) in terms of sampling efficiency, generation throughput, and watermark detectability. We consider two unbiased watermarks: Gumbel-max (Aaronson, 2023) and SynthID (Dathathri et al., 2024),[4] and test three target–draft model pairs: `OPT-30B/OPT-125M` (Zhang et al., 2022), `Llama-13B/Llama-68M` (Touvron et al., 2023; Miao et al., 2024), and `Qwen3-32B/Qwen3-0.6B` (Yang et al., 2025a). We evaluate on two generation tasks: long-form question answering (QA) on `ELI5` (Fan et al., 2019) and open-ended generation (OEG) on `C4` (Raffel et al., 2020). Throughout, we use standard (single-sequence) speculative decoding for watermark generation (He et al., 2026) ($k = 1$) as the baseline (Section 2). Additional results, implementation details, and a detailed breakdown of decoding time are provided in App. E.

4.1 One-step Generation

To directly evaluate the effect of vocabulary partition on sampling efficiency, we consider a one-step generation setting, where each method generates exactly one token from the same next-token distribution. Specifically, given a prefix, the baseline ($k = 1$) and our method ($k = 2, 3, 4$) each generate one token under identical $\boldsymbol{P}$ and $\boldsymbol{Q}$ across methods. This allows us to directly measure the sampling efficiency gain $\Delta_{\text{accept}} := A_k - A_1$. We disable watermarking in this experiment, since the considered unbiased watermarks leave the expected sampling efficiency unchanged.

Simulation. We randomly sample two distributions over a vocabulary of size $|\mathcal{W}| = 100$ as $\boldsymbol{P}$ and $\boldsymbol{Q}$, and compute Δ_{accept} under a random partition for $k = 2, 3, 4$. This is repeated 10^6 times, and the resulting histograms are shown in the left panel of Fig. 2. The entire mass of each histogram lies strictly above zero, *confirming that the vocabulary partition strategy improves sampling efficiency in every trial, consistent with Thm. 3.2*. Moreover, the distributions are well-separated: the mean gain increases monotonically with k (approximately 0.08, 0.11, and 0.13 for $k = 2, 3, 4$), indicating that the average sampling efficiency tends to grow with k even for random (non-refinement) partitions.

Real model distributions. We validate these findings with the `Qwen3-32B/Qwen3-0.6B` model pair (results for the other model pairs appear in App. E.2), again in the one-step setting. The middle two panels of Fig. 2 show the histograms of Δ_{accept} over 10^3 prefixes. The result shows that Δ_{accept} *remains non-negative across all prefixes, confirming the theoretical guarantee in Thm. 3.2 on real next-token distributions*. The rightmost panel reports the total number of accepted tokens (out of 10^3) as a function of k. Both curves increase monotonically from $k = 1$ to $k = 4$, with the QA task showing consistently higher acceptance counts, likely because the draft model better approximates the target distribution on question-answering prompts.

4.2 Real Generation Tasks

We now evaluate full-sequence watermark generation on QA and OEG, measuring sampling efficiency, generation throughput, and watermark detectability.

[4]For SynthID, we set the number of tournament sampling rounds to the default of 30. See Appendix B for details.

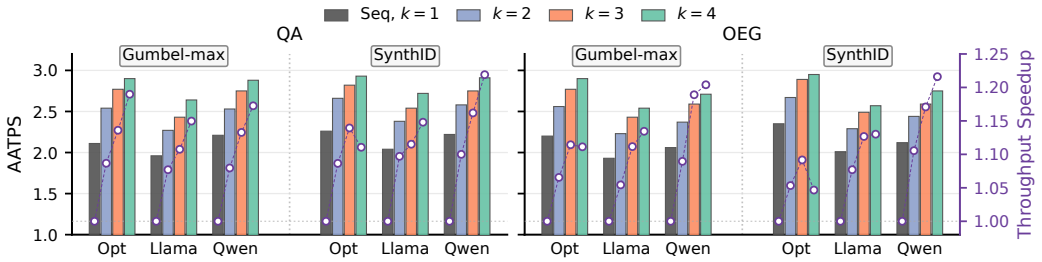


Figure 3: Average Accepted Tokens Per Step (AATPS; bars) for Seq ($k = 1$) and our method ($k = 2, 3, 4$) under Gumbel-max and SynthID watermarks. Purple lines show throughput speedup over Seq. **Left**: question answering (QA). **Right**: open-ended generation (OEG).

Table 1: Watermark detectability (TPR@FPR$= 1\%$) on the QA task at 200 tokens (mean $\pm$ 95% CI). The $k{+}1$ *PRs* setting has $(k{+}1)$-way pseudorandomness ambiguity, compared with two-way ambiguity with vocabulary partition in our method.

TPR@FPR1%	Gumbel-max			SynthID		
QA	OPT	Llama	Qwen	OPT	Llama	Qwen
Seq, $k = 1$	94.52±0.82	97.21±0.95	97.54±0.66	97.61±0.85	92.36±0.73	92.96±0.76
$k = 2$ (3 *PRs*)	87.06±2.66	88.30±0.89	93.19±0.75	79.39±2.76	76.87±3.23	78.97±1.75
$k = 2$	92.16±1.48	93.17±0.96	96.73±0.44	95.45±1.51	92.32±1.29	91.82±0.54
$k = 3$ (4 *PRs*)	82.90±2.26	81.57±1.13	89.67±1.04	62.35±5.38	56.34±3.57	66.63±4.03
$k = 3$	93.71±1.27	90.51±1.22	93.90±1.19	94.97±0.94	89.67±1.52	90.87±0.85
$k = 4$ (5 *PRs*)	77.66±3.98	77.45±1.60	87.38±1.23	42.54±4.66	39.89±4.74	49.85±4.92
$k = 4$	92.19±0.91	87.96±1.38	95.04±0.80	90.31±0.79	86.17±1.25	88.55±1.42

Sampling efficiency grows with branching factor k. Following prior work (Hu & Huang, 2024b; He et al., 2026), we measure the Average Accepted Tokens Per Step (AATPS), defined as the average number of tokens generated per decoding step (i.e., per target-model forward pass). Since we fix the lookahead depth to $L = 3$, AATPS lies in $[1, L + 1] = [1, 4]$: the lower bound corresponds to all draft tokens being rejected (fallback to a single residual sample), while the upper bound corresponds to all L drafts being accepted plus one additional token. Thus, higher AATPS directly indicates higher sampling efficiency. The result is shown in Figure 3. We find that across all settings, AATPS increases monotonically from the baseline (Seq, $k = 1$) to $k = 4$, showing that the per-step gains observed in Section 4.1 translate to improved efficiency in full generation. Results for varying lookahead depth are provided in App. E.3.1.

Sampling gains translate into higher throughput. We measure generation throughput in tokens per second. All tree configurations in Fig. 3 outperform Seq, with speedups of approximately 1.05–$1.22\times$. Throughput depends on both the tokens generated per step and the time spent per step: larger trees improve acceptance but also require more computation. Consequently, throughput does not always increase with k. For example, for `OPT` with SynthID, throughput peaks at $k = 3$ on both tasks despite higher AATPS at $k = 4$. App. E.5 provides a detailed breakdown of decoding time.

Detectability evaluation setup. We follow the detection protocol of He et al. (2026), adapted to tree-based generation (details in App. E.1.1). Table 1 reports TPR@FPR$=1\%$ on 200-token QA sequences for Seq ($k = 1$), our method ($k = 2, 3, 4$), and a detection ablation without partition-based branch identification ($k{+}1$ *PRs*). The ablation retains $k{+}1$ candidate pseudorandom sources, one per draft branch and one for residual sampling, whereas our method narrows the candidates to the identified draft branch and the residual source. Results on the OEG task are provided in App. E.3.2.

Vocabulary partition prevents severe detectability loss. As shown in Table 1, our method outperforms the $k{+}1$ *PRs* ablation for all three model pairs and both watermarks, with the gap widening as k increases. The benefit is especially pronounced for SynthID: at $k = 4$, TPR remains approximately 86–90%, compared with 40–50% for the ablation. These results show that partition-

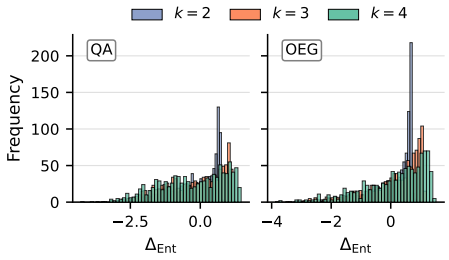


Figure 4: Per-prefix entropy gap: $\mathrm{Ent}(\boldsymbol{Q}) - \frac{1}{k}\sum_i \mathrm{Ent}(\boldsymbol{Q}^{(i)})$ on `Qwen3-0.6B`.

Table 2: Throughput (tokens/s) and detectability for watermark generation with a `Qwen3-0.6B` drafter or an EAGLE head (both with $k = 2$, $L = 3$). Parentheses show speedup over Seq.

Task	Watermark	Throughput		TPR@FPR1%	
		0.6B	EAGLE	0.6B	EAGLE
QA	Gumbel.	24.4,(1.08)	42.2,(1.87)	96.73	98.30
	SynthID	23.1,(1.10)	40.2,(1.92)	91.82	93.19
OEG	Gumbel.	21.9,(1.09)	32.5,(1.62)	98.95	99.51
	SynthID	22.0,(1.11)	29.5,(1.48)	98.46	99.15

based branch identification prevents the growing ambiguity among pseudorandom sources from severely weakening detection as more branches are added.

Entropy redistribution explains detectability variation. Relative to standard speculative decoding (k=1), our method largely preserves watermark detectability, with some degradation at larger branching factors. This is consistent with the entropy-based analysis in Section 3.3: more concentrated branch distributions make token selection less sensitive to watermark pseudorandomness, which can weaken detection. Figure 4 plots the per-prefix entropy gap $\mathrm{Ent}(\boldsymbol{Q}) - \frac{1}{k}\sum_i \mathrm{Ent}(\boldsymbol{Q}^{(i)})$ for `Qwen3-0.6B` in the one-step setting of Section 4.1 (results for other models are provided in App. E.3.2). The gap takes both positive and negative values, with larger shifts at higher k, showing that partitioning can raise or lower the per-branch entropy. In general, distribution entropy is tightly related to the corresponding watermark signal, which explains why larger entropy shifts lead to more pronounced detectability changes as k increases.

Compatibility with an EAGLE-style draft head. In our method, we replace the standalone `Qwen3-0.6B` drafter with an EAGLE-style draft head (Li et al., 2024a) trained for `Qwen3-32B`. With $k = 2$ and $L = 3$, Table 2 reports 1.48–1.92× throughput over Seq across both tasks and watermarks, while maintaining detectability comparable to that of the standalone drafter. As discussed in Section 3.1, vocabulary partitioning reduces source ambiguity independently of the drafter architecture. Our framework thus supports watermark embedding and effective detection with an EAGLE-style draft head, whose lower drafting cost enables further speedup. These results show that our framework is not limited to standalone draft models, allowing watermarked generation to benefit from advances in drafter design. Training details and additional results are provided in App. E.3.3.

5 Discussion

We introduced the first tree-based speculative decoding algorithm for watermarked generation, built on a simple vocabulary partition strategy that deterministically splits the draft vocabulary into disjoint subsets. This design simultaneously addresses all four requirements identified in Section 1: it enables diverse draft tree construction (**R1**), preserves the target distribution (**R2**), reduces detection ambiguity from k+1 to 2 (**R3**), and provably improves sampling efficiency over standard speculative decoding (**R4**). Experiments with Gumbel-max and SynthID show gains in sampling efficiency and generation throughput while largely preserving detectability. The EAGLE experiments further demonstrate that the same vocabulary partition strategy supports different drafters, enabling additional speedups.

Our results suggest several directions for future work. First, we treat the partition rule $\mathcal{A}$ as a fixed deterministic mapping based on the prefix. Designing distribution-aware or learned partitions may further improve efficiency. Second, designing sparse draft trees for faster verification while preserving vocabulary partitioning remains an open direction. Our exploratory pruning experiments reduce tree size but do not improve generation throughput over the dense $k = 4, L = 3$ tree (App. E.6), motivating joint optimization of tree expansion and execution cost. Finally, while our framework naturally extends to other unbiased watermarking schemes and we have demonstrated its compatibility with an EAGLE-style draft head, adapting it to biased watermarking schemes and integrating it with other draft architectures remain open research directions.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code development and to improve the clarity and readability of the manuscript. All AI-assisted code and text were reviewed by the authors. The authors take full responsibility for the final content of this work, including its code, results, and scientific claims.

REFERENCES

Scott Aaronson. Watermarking of large language models, August 2023. URL https://simons.berkeley.edu/talks/scott-aaronson-ut-austin-openai-2023-08-17.

Kwangjun Ahn, Ahmad Beirami, Ziteng Sun, and Ananda Theertha Suresh. Spectr++: Improved transport plans for speculative decoding of large language models. In *NeurIPS 2023 Workshop Optimal Transport and Machine Learning*, 2023.

Zihao An, Taichi Liu, Ziqiong Liu, Dong Li, Ruofeng Liu, and Emad Barsoum. Pard-2: Target-aligned parallel draft model for dual-mode speculative decoding. *arXiv preprint arXiv:2605.08632*, 2026.

Anthropic. Announcements: How claude's text watermark works, August 2026. URL https://www.anthropic.com/news/claude-text-watermark.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning*, pp. 5209–5235. PMLR, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. *Advances in Neural Information Processing Systems*, 37:129531–129563, 2024.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, Jamie Hayes, Nidhi Vyas, Majd Al Merey, Jonah Brown-Cohen, Rudy Bunel, Borja Balle, Taylan Cemgil, Zahra Ahmed, Kitty Stacpoole, Ilia Shumailov, Ciprian Baetu, Sven Gowal, Demis Hassabis, and Pushmeet Kohli. Scalable watermarking for identifying large language model outputs. *Nature*, 634 (8035):818–823, Oct 2024. URL https://doi.org/10.1038/s41586-024-08025-4.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3558–3567, 2019.

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. Three bricks to consolidate watermarks for large language models. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6. IEEE, 2023.

Jiayi Fu, Xuandong Zhao, Ruihan Yang, Yuansen Zhang, Jiangjie Chen, and Yanghua Xiao. GumbelSoft: Diversified language model watermarking via the GumbelMax-trick. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5791–5808, 2024.

Eva Giboulot and Teddy Furon. Watermax: breaking the llm watermark detectability-robustness-quality trade-off. *Advances in neural information processing systems*, 37:18848–18881, 2024.

Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: A series of lectures*, volume 33. US Government Printing Office, 1948.

Weiqing He, Xiang Li, Tianqi Shang, Li Shen, Weijie Su, and Qi Long. On the empirical power of goodness-of-fit tests in watermark detection. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Weiqing He, Xiang Li, Li Shen, Weijie Su, and Qi Long. Improving the trade-off between watermark strength and speculative sampling efficiency for language models. In *International Conference on Learning Representations*, 2026.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. Rest: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1582–1595, 2024.

Shijing Hu, Jingyang Li, Zhihui Lu, and Pan Zhou. Bridging draft policy misalignment: Group tree optimization for speculative decoding. In *International Conference on Learning Representations*, 2026.

Zhengmian Hu and Heng Huang. Accelerated speculative sampling based on tree monte carlo. In *Forty-first International Conference on Machine Learning*, 2024a.

Zhengmian Hu and Heng Huang. Inevitable trade-off between watermark strength and speculative sampling efficiency for language models. In *Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=6YKMBUiIsG.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. Unbiased watermark for large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=uWVC5FVidc.

Zhengmian Hu, Tong Zheng, Vignesh Viswanathan, Ziyi Chen, Ryan Rossi, Yihan Wu, Dinesh Manocha, and Heng Huang. Towards optimal multi-draft speculative decoding. In *International Conference on Learning Representations*, volume 2025, pp. 3181–3203, 2025.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017.

Wonseok Jeon, Mukul Gagrani, Raghavv Goel, Junyoung Park, Mingu Lee, and Christopher Lott. Recursive speculative decoding: Accelerating llm inference via sampling without replacement. *arXiv preprint arXiv:2402.14160*, 2024.

Ashish J Khisti, Arash Behravesh, Hassan Dbouk, Arash Behboodi, Roland Memisevic, and Christos Louizos. Importance weighted multi-draft speculative sampling. In *ICML 2024 Workshop on Theoretical Foundations of Foundation Models*, 2024.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. Speculative decoding with big little decoder. *Advances in Neural Information Processing Systems*, 36:39236–39256, 2023.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning*, volume 202, pp. 17061–17084, 2023.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=FpaCL1MO2C.

Tanishq Kumar, Tri Dao, and Avner May. Speculative speculative decoding. In *International Conference on Learning Representations*, 2026.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

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

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, pp. 28935–28948. PMLR, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *Advances in Neural Information Processing Systems*, 38:136737–136756, 2026.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Online speculative decoding. In *International Conference on Machine Learning*, pp. 31131–31146. PMLR, 2024.

Carol Xuan Long, Dor Tsur, Claudio Mayrink Verdun, Hsiang Hsu, Haim Permuter, and Flavio P Calmon. Optimized couplings for watermarking large language models. In *2025 IEEE International Symposium on Information Theory (ISIT)*, pp. 1–6. IEEE, 2025.

Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949, 2024.

Silvia Milano, Joshua A McGrane, and Sabina Leonelli. Large language models challenge the future of higher education. *Nature Machine Intelligence*, 5(4):333–334, 2023.

OpenAI. Understanding the source of what we see and hear online, May 2024. URL `https://openai.com/index/understanding-the-source-of-what-we-see-and-hear-online/`. Accessed: 2024-05-07.

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(1):5485–5551, 2020.

Joseph Rowan, Truong Buu Phan, and Ashish Khisti. List-level distribution coupling with applications to speculative decoding and lossy compression. *Advances in Neural Information Processing Systems*, 38:5576–5619, 2026.

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

Benjamin Spector and Chris Re. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*, 2023.

Kate Starbird. Disinformation's spread: Bots, trolls and all of us. *Nature*, 571(7766):449–450, 2019.

Qidong Su, Christina Giannoula, and Gennady Pekhimenko. The synergy of speculative decoding and batching in serving large language models. *arXiv preprint arXiv:2310.18813*, 2023.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36:30222–30242, 2023.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae H Ro, Ahmad Beirami, and Ananda Theertha Suresh. Block verification accelerates speculative decoding. In *International Conference on Learning Representations*, volume 2025, pp. 24965–24994, 2025.

Rahul Thomas and Arka Pal. Global resolution: Optimal multi-draft speculative sampling via convex optimization. In *International Conference on Learning Representations*, volume 2026, pp. 81274–81307, 2026.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Dor Tsur, Carol Long, Claudio Mayrink Verdun, Sajani Vithana, Hsiang Hsu, Richard Chen, Flavio Calmon, et al. Heavywater and simplexwater: Distortion-free llm watermarks for low-entropy distributions. *Advances in Neural Information Processing Systems*, 38:107204–107256, 2026.

Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, et al. Taxonomy of risks posed by language models. In *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pp. 214–229, 2022.

Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. A resilient and accessible distribution-preserving watermark for large language models. In *International Conference on Machine Learning*, pp. 53443–53470. PMLR, 2024.

Zilin Xiao, Hongming Zhang, Tao Ge, Siru Ouyang, Vicente Ordonez, and Dong Yu. Parallelspec: Parallel drafter for efficient speculative decoding. *arXiv preprint arXiv:2410.05589*, 2024.

Yangxinyu Xie, Xiang Li, Tanwi Mallick, Weijie Su, and Ruixun Zhang. Debiasing watermarks for large language models via maximal coupling. *Journal of the American Statistical Association*, 120 (551):1424–1436, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. Multi-candidate speculative decoding. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pp. 335–348. Springer, 2025b.

Ming Yin, Minshuo Chen, Kaixuan Huang, and Mengdi Wang. A theoretical perspective for speculative decoding algorithm. *Advances in Neural Information Processing Systems*, 37:128082–128117, 2024.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Xuandong Zhao, Yu-Xiang Wang, and Lei Li. Protecting language generation models via invisible watermarking. In *International Conference on Machine Learning*, volume 202, pp. 42187–42199, 2023.

Xuandong Zhao, Chenwen Liao, Yu-Xiang Wang, and Lei Li. Efficiently identifying watermarked segments in mixed-source texts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6304–6316, 2025.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-Francccois Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. In *International Conference on Learning Representations*, volume 2024, pp. 32011–32050, 2024.

A Related Work

Watermarking for LLMs. A growing body of work has developed watermarking schemes that embed a verifiable statistical signal into generated text (Kirchenbauer et al., 2023; Aaronson, 2023; Fernandez et al., 2023; Hu et al., 2024; Kuditipudi et al., 2024; Wu et al., 2024; Dathathri et al., 2024; Zhao et al., 2025; 2023; Giboulot & Furon, 2024; Fu et al., 2024; Xie et al., 2025; Long et al., 2025; Tsur et al., 2026). A central distinction within this literature is whether the induced distribution matches the underlying model: a watermark is called unbiased if the distribution of the output token recovers the original model distribution (Hu et al., 2024; Li et al., 2025). Unbiased schemes—most prominently the Gumbel-max watermark (Aaronson, 2023) and the SynthID watermark (Dathathri et al., 2024)—are particularly attractive because they leave the text quality intact in distribution. Most watermark schemes embed the verifiable signal by involving pseudorandomness into the next-token sampling process, thereby building the dependency between the output token and the pseudorandomness (Li et al., 2025). In parallel with designing the watermark scheme, He et al. (2026) define the watermark strength which quantifies how each token is coupled with the recoverable pseudorandom numbers. In our work, we focus on unbiased watermarks and adopt the watermark strength measure from He et al. (2026).

Speculative decoding. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) accelerates autoregressive generation by a draft-and-verify procedure: a cheap draft model proposes several candidate tokens, and the target model accepts or rejects them within a single forward pass, while provably preserving the target distribution. A large portion of recent work improves the drafting stage: training dedicated draft models via distillation (Zhou et al., 2024; Liu et al., 2024; Kim et al., 2023), retrieving drafts from a datastore (He et al., 2024), predicting the draft acceptance in advance (Kumar et al., 2026), or reducing drafting latency through parallel multi-token prediction (Xiao et al., 2024; An et al., 2026). A complementary line of work, most relevant to us, replaces single-sequence drafts with tree-structured drafts (Miao et al., 2024; Cai et al., 2024; Li et al., 2024a; Spector & Re, 2023; Yang et al., 2025b; Jeon et al., 2024; Li et al., 2026), so that multiple candidate continuations can be verified simultaneously under a shared prefix, leading to a higher speedup. Miao et al. (2024) introduced this idea in SpecInfer, using a topology-aware attention mask that lets the target model score all nodes of the draft tree in one forward pass. Related advances span several axes: exploring multi-draft coupling and verification (Sun et al., 2023; Yang et al., 2025b; Jeon et al., 2024; Hu & Huang, 2024a; Khisti et al., 2024; Hu et al., 2025; Ahn et al., 2023; Sun et al., 2025; Thomas & Pal, 2026; Rowan et al., 2026), refining training of draft models to better align with target models (Cai et al., 2024; Li et al., 2024a; Hu et al., 2026), staged or dynamic tree construction (Spector & Re, 2023; Su et al., 2023; Chen et al., 2024), and theoretical study (Yin et al., 2024). Collectively, these advances establish tree-based speculation as the dominant strategy for high-acceptance, high-throughput LLM inference. In this work, we focus on this tree-based regime and ask whether its efficiency gains can be transferred to watermarked generation.

Watermarking with speculative decoding. Only a limited number of works (Hu & Huang, 2024b; Dathathri et al., 2024; He et al., 2026) study the intersection of watermarking and speculative decoding. Dathathri et al. (2024) propose a fast decoding algorithm for the SynthID watermark, while Hu & Huang (2024b) identified a fundamental trade-off between watermark strength and speculative sampling efficiency. Subsequently, He et al. (2026) improved this trade-off by introducing pseudorandom acceptance. However, these approaches are restricted to standard (single-sequence) speculative decoding and do not explore tree-based extensions. In contrast, our work is the first to investigate tree-based speculative decoding for watermark generation.

B Considered Watermarks and Strength Measure

We instantiate our framework with two representative unbiased watermarks: the Gumbel-max watermark (Aaronson, 2023) and the SynthID watermark (Dathathri et al., 2024). Both are widely adopted in practice—the former has been deployed internally at OpenAI (OpenAI, 2024) and is a standard baseline across the watermarking literature, while the latter was introduced by Google and uses an alternative tournament-based construction. We describe each below.

Gumbel-max watermark. This scheme repurposes the Gumbel-max trick (Gumbel, 1948; Maddison et al., 2014; Jang et al., 2017), a classical tool for sampling from multinomial distributions. The pseudorandom variable takes the form $\zeta = (U_w)_{w\in\mathcal{W}}$, a collection of i.i.d. uniform variables indexed by the vocabulary. A standard identity states that,

$$\arg\max_{w\in\mathcal{W}} \frac{\log U_w}{P_w} \;\sim\; \boldsymbol{P},$$

so one can draw from $\boldsymbol{P}$ by evaluating this argmax. Aaronson (2023) turns this identity into a deterministic decoder by placing all mass on the maximizer: the watermarked distribution $\boldsymbol{P}_\zeta = \mathcal{S}^{\mathrm{gum}}(\boldsymbol{P}, \zeta)$ is given by

$$(\mathcal{S}^{\mathrm{gum}}(\boldsymbol{P}, \zeta))(w) = \begin{cases} 1, & w = \arg\max_{w'\in\mathcal{W}} \frac{\log U_{w'}}{P_{w'}}, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Averaging over ζ recovers $\boldsymbol{P}$, so the scheme is unbiased (Li et al., 2025).

SynthID watermark. SynthID takes a different route: rather than perturbing a single argmax, it embeds the signal through a sequence of pairwise "matches" between tokens, which Dathathri et al. (2024) call *tournament sampling*. The pseudorandom variable is $\zeta = (\boldsymbol{g}_i)_{i=1}^m$, where each $\boldsymbol{g}_i = (g_{i,w})_{w\in\mathcal{W}} \in \{0,1\}^{|\mathcal{W}|}$ has entries drawn independently from Bernoulli$(1/2)$, and $m \geq 1$ is the number of tournament rounds (we set $m = 30$ in our experiments). *Throughout the paper we work with the two-candidate variant*, in which a single round is modeled by the operator

$$(\mathcal{T}_{\boldsymbol{g}}(\boldsymbol{P}))(w) \;=\; P_w \cdot \Big(1 + g_w - \sum_{w':\, g_{w'}=1} P_{w'}\Big). \tag{3}$$

Operationally, $\mathcal{T}_{\boldsymbol{g}}(\boldsymbol{P})$ is the distribution of the winner of a one-versus-one contest: two candidates w_1, w_2 are sampled independently from $\boldsymbol{P}$, and the token with the larger bit g_w advances; ties (i.e., $g_{w_1} = g_{w_2}$) are broken by a fair coin flip. Composing m such rounds with independent bit-vectors produces the watermarked distribution

$$\boldsymbol{P}_\zeta \;=\; \mathcal{S}^{\mathrm{syn}}(\boldsymbol{P}, \zeta) \;=\; \mathcal{T}_{\boldsymbol{g}_m} \circ \cdots \circ \mathcal{T}_{\boldsymbol{g}_1}(\boldsymbol{P}), \tag{4}$$

whose sample corresponds to the overall winner after m bracket rounds. As shown in Dathathri et al. (2024), $\mathcal{S}^{\mathrm{syn}}$ is unbiased for every m.

B.1 Watermark Strength

The strength of a watermark quantifies how strongly each sampled token depends on the pseudorandom variable, and directly governs the difficulty of watermark detection (He et al., 2026). Following He et al. (2026), we measure it by the expected KL divergence between the reweighted distribution $\boldsymbol{P}_\zeta = \mathcal{S}(\boldsymbol{P}, \zeta)$ and the original distribution $\boldsymbol{P}$:

$$\mathsf{WS}(\boldsymbol{P}_\zeta) = \mathbb{E}_\zeta[D_{\mathrm{KL}}(\boldsymbol{P}_\zeta \,\|\, \boldsymbol{P})] = \mathbb{E}_\zeta\left[\sum_{w\in\mathcal{W}} P_\zeta(w) \log \frac{P_\zeta(w)}{P(w)}\right]. \tag{5}$$

where $D_{\mathrm{KL}}(\boldsymbol{P}_\zeta \| \boldsymbol{P})$ denotes the Kullback-Leibler (KL) divergence between the watermarked distribution $\boldsymbol{P}_\zeta$ and the original distribution $\boldsymbol{P}$. The watermark strength describes how strongly each token depends on pseudorandomness and directly governs the difficulty of watermark detection He et al. (2026).

Maximum watermark strength. For any unbiased watermark, $\mathsf{WS}(\boldsymbol{P}_\zeta) \leq \mathrm{Ent}(\boldsymbol{P})$, with equality if and only if $\boldsymbol{P}_\zeta$ is almost surely a point mass (i.e., $\mathrm{Ent}(\boldsymbol{P}_\zeta) = 0$ a.s.). Both watermarks above meet this condition—the Gumbel-max watermark directly, and the SynthID watermark asymptotically as $m \to \infty$. We give the intuition here and refer the reader to He et al. (2026) for the details.

Gumbel-max. The decoder in (2) is, by construction, a deterministic function of ζ: conditional on the uniform variables $(U_w)_{w\in\mathcal{W}}$, all probability concentrates on the single token $\arg\max_w \log U_w/P_w$. Hence $\boldsymbol{P}_\zeta$ is degenerate with probability one, so $\mathrm{Ent}(\boldsymbol{P}_\zeta) = 0$ and the maximum is attained.

SynthID as $m \to \infty$. The argument is less immediate because a finite tournament generally leaves $\boldsymbol{P}_\zeta$ nondegenerate. Write $\hat{\boldsymbol{P}}_t := \mathcal{T}_{\boldsymbol{g}_t} \circ \cdots \circ \mathcal{T}_{\boldsymbol{g}_1}(\boldsymbol{P})$ for the distribution after t rounds, and let $\mathcal{F}_t = \sigma(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_t)$. Since each round $\mathcal{T}_{\boldsymbol{g}}$ is unbiased, $\mathbb{E}[\hat{\boldsymbol{P}}_t \mid \mathcal{F}_{t-1}] = \hat{\boldsymbol{P}}_{t-1}$, so $(\hat{\boldsymbol{P}}_t)_{t \geq 0}$ is a bounded martingale and therefore converges almost surely to some limit $\hat{\boldsymbol{P}}_\infty$. This limit must be invariant under $\mathcal{T}_{\boldsymbol{g}}$ for every realization of $\boldsymbol{g}$. Inspecting (3) shows that a distribution is a fixed point of $\mathcal{T}_{\boldsymbol{g}}$ for every $\boldsymbol{g}$ only if it puts all mass on a single token. Thus $\hat{\boldsymbol{P}}_\infty$ is degenerate a.s., and $\mathcal{S}^{\mathrm{syn}}$ attains the maximum watermark strength in the limit $m \to \infty$.

C Proofs of Main Results

C.1 Proof of Theorem 3.1

Proof of Theorem 3.1. We proceed by induction on the number of branches k.

Base case ($k = 1$). The verification procedure reduces to standard speculative decoding with watermark: accept the draft token $\tilde{w}_1 \sim \mathcal{S}(\boldsymbol{Q}, \zeta^{(1)})$ with probability $\min\{1, P(\tilde{w}_1)/Q(\tilde{w}_1)\}$, and upon rejection sample from the watermarked residual $\mathcal{S}([\boldsymbol{P} - \boldsymbol{Q}]_+, \zeta^{(0)})$. By the unbiasedness of $\mathcal{S}$ and the independence of $\zeta^{(0)}$ and $\zeta^{(1)}$, this recovers $\boldsymbol{P}$ in expectation, as shown in He et al. (2026).

Inductive step. Assume the claim holds for any $(k-1)$-branch verification procedure (with any target distribution, any branch distributions with disjoint supports, and mutually independent pseudorandom variables). Consider the k-branch procedure. At branch 1, by the independence of $\zeta^{(1)}$ from the remaining pseudorandom variables and the unbiasedness of $\mathcal{S}$, the expected output distribution decomposes as:

$$\mathbb{E}_\zeta[P'_\zeta(w)] \;=\; \underbrace{\min\{P(w),\, Q^{(1)}(w)\}}_{\text{accepted from branch 1}} \;+\; Z_1 \cdot \mathbb{E}_{\zeta \setminus \zeta^{(1)}}[\tilde{P}'_\zeta(w)],$$

where $Z_1 = 1 - \sum_w \min\{P(w), Q^{(1)}(w)\}$ is the rejection probability of branch 1, and $\tilde{P}'_\zeta$ is the output distribution of the remaining $(k-1)$-branch procedure with target distribution $\boldsymbol{P}^{(1)} = [\boldsymbol{P} - \boldsymbol{Q}^{(1)}]_+$.

By the inductive hypothesis applied to the $(k-1)$-branch procedure (with target $\boldsymbol{P}^{(1)}$, branch distributions $\boldsymbol{Q}^{(2)}, \ldots, \boldsymbol{Q}^{(k)}$, residual pseudorandom variable $\zeta^{(0)}$, and branch pseudorandom variables $\zeta^{(2)}, \ldots, \zeta^{(k)}$, all mutually independent):

$$\mathbb{E}_{\zeta \setminus \zeta^{(1)}}[\tilde{P}'_\zeta(w)] = P^{(1)}(w) = \frac{(P(w) - Q^{(1)}(w))_+}{Z_1}.$$

Substituting back:

$$\mathbb{E}_\zeta[P'_\zeta(w)] = \min\{P(w), Q^{(1)}(w)\} + (P(w) - Q^{(1)}(w))_+ = P(w),$$

where the last equality is the identity $\min\{a, b\} + (a - b)_+ = a$. □

C.2 Proof of Theorem 3.2

Proof of Theorem 3.2. Suppress the conditioning on $w_{<t}$ and write $P(w)$, $Q(w)$ for the target and draft probabilities. Let $\alpha_i := Q(\mathcal{W}_i) = \sum_{w \in \mathcal{W}_i} Q(w) > 0$ and recall

$$Q^{(i)}(w) = Q(w)/\alpha_i \cdot \mathbf{1}\{w \in \mathcal{W}_i\}.$$

By unbiasedness $\mathbb{E}_\zeta[\boldsymbol{Q}^{(i)}_\zeta] = \boldsymbol{Q}^{(i)}$, the acceptance/rejection probabilities under watermarked proposals $\tilde{w}_i \sim \mathcal{S}(\boldsymbol{Q}^{(i)}, \zeta^{(i)})$ agree with those under $\tilde{w}_i \sim \boldsymbol{Q}^{(i)}$ after taking $\mathbb{E}_\zeta$. It therefore suffices to analyze the non-watermarked proposals.

Let Z_i denote the probability that the first i branches are all rejected, with $Z_0 := 1$. Branch i is reached only when branches $1, \ldots, i-1$ are all rejected, which occurs with probability Z_{i-1}. Conditional on

reaching branch i, the current target distribution is the residual $\boldsymbol{P}^{(i-1)}$ and the candidate $\tilde{w}_i$ is drawn from $\boldsymbol{Q}^{(i)}$, so the conditional acceptance probability is

$$\sum_{w\in\mathcal{W}_i} Q^{(i)}(w)\,\min\!\left\{1,\ \frac{P^{(i-1)}(w)}{Q^{(i)}(w)}\right\} \;=\; \sum_{w\in\mathcal{W}_i} \min\!\left\{Q^{(i)}(w),\ P^{(i-1)}(w)\right\}.$$

Since $\boldsymbol{Q}^{(1)},\dots,\boldsymbol{Q}^{(i-1)}$ are supported on $\mathcal{W}_1,\dots,\mathcal{W}_{i-1}$ respectively (all disjoint from $\mathcal{W}_i$), the residual mass on $\mathcal{W}_i$ is unchanged through the first $i-1$ updates: $P^{(i-1)}(w) = P(w)/Z_{i-1}$ for all $w \in \mathcal{W}_i$.[5] Substituting $Q^{(i)}(w) = Q(w)/\alpha_i$ and $P^{(i-1)}(w) = P(w)/Z_{i-1}$, the conditional acceptance probability becomes $\sum_{w\in\mathcal{W}_i} \min\{Q(w)/\alpha_i,\ P(w)/Z_{i-1}\}$. Multiplying by Z_{i-1} gives the unconditional acceptance probability at branch i:

$$\begin{aligned} a_i \;&:=\; \Pr(\text{accept at branch } i) \\ &=\; Z_{i-1}\sum_{w\in\mathcal{W}_i} \min\!\left\{\frac{Q(w)}{\alpha_i},\ \frac{P(w)}{Z_{i-1}}\right\} \;=\; \sum_{w\in\mathcal{W}_i} \min\!\left\{\frac{Z_{i-1}}{\alpha_i}\,Q(w),\ P(w)\right\}, \end{aligned}$$

with $Z_i = Z_{i-1} - a_i$ for $i = 1,\dots,k$. Therefore

$$A_k = 1 - Z_k \;=\; \sum_{i=1}^{k} a_i. \tag{6}$$

Define the baseline overlap on each block:

$$b_i \;:=\; \sum_{w\in\mathcal{W}_i} \min\{P(w),\,Q(w)\}, \qquad \text{so that} \quad \sum_{i=1}^{k} b_i = \sum_{w\in\mathcal{W}} \min\{P(w),\,Q(w)\}.$$

We show $\sum_{i=1}^{k} a_i \ge \sum_{i=1}^{k} b_i$. For each branch i, define $c_i := Z_{i-1}/\alpha_i$. If $c_i \ge 1$, then $c_i Q(w) \ge Q(w)$ for all $w \in \mathcal{W}_i$, so pointwise $\min\{P(w), c_i Q(w)\} \ge \min\{P(w), Q(w)\}$, hence $a_i \ge b_i$.

It remains to handle $c_i < 1$. Define the deficit $D_i := b_i - a_i \ge 0$. For all $w \in \mathcal{W}_i$, one has the pointwise bound

$$\min\{P(w), Q(w)\} - \min\{P(w), c_i Q(w)\} \;\le\; (1 - c_i)\,Q(w),$$

which follows by checking three cases: (i) $P(w) \le c_i Q(w)$: both minima equal $P(w)$, difference is 0; (ii) $c_i Q(w) < P(w) \le Q(w)$: difference is $P(w) - c_i Q(w) \le (1-c_i)Q(w)$; (iii) $P(w) > Q(w)$: difference is $(1 - c_i)Q(w)$. Summing over $\mathcal{W}_i$ gives

$$D_i \;\le\; (1 - c_i)\,\alpha_i \;=\; \alpha_i - Z_{i-1}. \tag{7}$$

Define the cumulative advantage after m branches: $\Delta_m := \sum_{j=1}^{m}(a_j - b_j)$. Using $\sum_{j=1}^{m} a_j = 1 - Z_m$ and $\sum_{j=1}^{m} b_j \le \sum_{w\in\bigcup_{j=1}^{m}\mathcal{W}_j} Q(w) = \sum_{j=1}^{m}\alpha_j$, we obtain

$$\Delta_m \;\ge\; (1 - Z_m) - \sum_{j=1}^{m}\alpha_j \;=\; \sum_{j=m+1}^{k}\alpha_j - Z_m. \tag{8}$$

For branch i, applying (8) with $m = i - 1$ gives $\Delta_{i-1} \ge \sum_{j=i}^{k}\alpha_j - Z_{i-1} \ge \alpha_i - Z_{i-1}$. If $c_i \ge 1$, then $a_i \ge b_i$ and $\Delta_i \ge \Delta_{i-1} \ge 0$. If $c_i < 1$, then by (7),

$$\Delta_i = \Delta_{i-1} - D_i \;\ge\; (\alpha_i - Z_{i-1}) - (\alpha_i - Z_{i-1}) = 0.$$

By induction, $\Delta_m \ge 0$ for all m, and in particular $\sum_{i=1}^{k} a_i \ge \sum_{i=1}^{k} b_i$.

By (6) and the definition of sampling efficiency (Def. 3.1),

$$A_k \;=\; 1 - Z_k \;=\; \sum_{i=1}^{k} a_i \;\ge\; \sum_{w\in\mathcal{W}} \min\{P(w),\,Q(w)\} \;=\; 1 - \mathrm{TV}(\boldsymbol{P},\boldsymbol{Q}). \qquad \square$$

[5] This step assumes $Z_{i-1} > 0$. When $Z_{i-1} = 0$ the branch is never reached and $a_i = 0$, which agrees with the closed-form expression on the right-hand side of (6) evaluated at $Z_{i-1} = 0$; the remaining argument is unaffected.

C.3 Proof of Theorem 3.3

Proof of Theorem 3.3. Suppress the conditioning on $w_{<t}$ and write $P(w)$, $Q(w)$ for the target and draft probabilities, following the notation established in the proof of Theorem 3.2.

For any block $U \subseteq \mathcal{W}$ with $Q(U) := \sum_{w \in U} Q(w) > 0$, define the *block map* $\phi_U : [0,1] \to [0,1]$ by

$$\phi_U(z) \;:=\; z \;-\; \sum_{w \in U} \min\!\left\{ P(w),\; \frac{z}{Q(U)}\, Q(w) \right\}, \qquad z \in [0,1]. \tag{9}$$

This map has the following interpretation: if the incoming reject mass (i.e., the probability that all previous branches were rejected) is z, then $\phi_U(z)$ is the reject mass after processing block U. Indeed, when block U is reached with reject mass z, the residual target distribution restricted to U assigns mass $P(w)/z$ to each $w \in U$, while the branch distribution is $Q(w)/Q(U)$. The unconditional acceptance probability at this block is $\sum_{w \in U} \min\{P(w),\, (z/Q(U))\, Q(w)\}$, so the remaining reject mass is z minus this quantity, which is exactly $\phi_U(z)$.

Under the coarse partition $\Pi = (\mathcal{W}_1, \ldots, \mathcal{W}_k)$, the overall reject probability after all k branches is

$$Z_k^{\Pi} \;=\; \phi_{\mathcal{W}_k} \circ \cdots \circ \phi_{\mathcal{W}_1}(1).$$

Under the refined partition Π', the reject probability is

$$Z_{k'}^{\Pi'} \;=\; \phi_{\mathcal{W}_{k,m_k}} \circ \cdots \circ \phi_{\mathcal{W}_{k,1}} \circ \cdots \circ \phi_{\mathcal{W}_{1,m_1}} \circ \cdots \circ \phi_{\mathcal{W}_{1,1}}(1),$$

where $k' = \sum_{i=1}^{k} m_i$ is the total number of blocks in Π'. The theorem is equivalent to showing $Z_{k'}^{\Pi'} \le Z_k^{\Pi}$.

We establish two properties of the block maps.

Property 1: Each ϕ_U is nondecreasing. For $z_2 \ge z_1 \ge 0$ and any $w \in U$,

$$0 \;\le\; \min\!\left\{ P(w),\; \frac{z_2}{Q(U)}\, Q(w) \right\} - \min\!\left\{ P(w),\; \frac{z_1}{Q(U)}\, Q(w) \right\} \;\le\; \frac{Q(w)}{Q(U)}\,(z_2 - z_1),$$

where the upper bound follows by checking three cases: (i) if $P(w) \le \frac{z_1}{Q(U)} Q(w)$, both minima equal $P(w)$ and the difference is 0; (ii) if $\frac{z_1}{Q(U)} Q(w) < P(w) \le \frac{z_2}{Q(U)} Q(w)$, the difference is at most $\frac{Q(w)}{Q(U)}(z_2 - z_1)$; (iii) if $P(w) > \frac{z_2}{Q(U)} Q(w)$, the difference equals $\frac{Q(w)}{Q(U)}(z_2 - z_1)$. Summing over $w \in U$ gives

$$\phi_U(z_2) - \phi_U(z_1) \;\ge\; (z_2 - z_1) - \sum_{w \in U} \frac{Q(w)}{Q(U)}\,(z_2 - z_1) \;=\; 0.$$

Property 2: Refining a block does not increase the reject mass. Let $U = \bigcup_{j=1}^{m} U_j$ be a partition of U into m sub-blocks with $Q(U_j) > 0$ for all j. We claim that for every $z \in [0,1]$,

$$\phi_{U_m} \circ \cdots \circ \phi_{U_1}(z) \;\le\; \phi_U(z). \tag{10}$$

To see this, observe that upon reaching block U with reject mass z, the residual target distribution restricted to U is $P(w)/z$ for $w \in U$. The sub-blocks $(U_1, \ldots, U_m)$ form a partition of U, and processing them sequentially is exactly the m-branch verification procedure of Theorem 3.2 applied to the restricted problem with target $P(\cdot)/z$ on U and draft $Q(\cdot)/Q(U)$ on U. By Theorem 3.2, the sampling efficiency of this m-branch procedure is at least $\sum_{w \in U} \min\{P(w)/z,\, Q(w)/Q(U)\}$, which is the acceptance probability of the single-block procedure on U. Multiplying by z gives that the total acceptance from the refined sub-blocks is at least that of the single block, establishing (10).

Combining the two properties. For each coarse block $\mathcal{W}_i$, Property 2 gives $\phi_{\mathcal{W}_{i,m_i}} \circ \cdots \circ \phi_{\mathcal{W}_{i,1}}(z) \le \phi_{\mathcal{W}_i}(z)$ for all z. Since all block maps are nondecreasing (Property 1), we can compose these inequalities across the k coarse blocks:

$$Z_{k'}^{\Pi'} \;=\; \left(\prod_{i=k}^{1} \phi_{\mathcal{W}_{i,m_i}} \circ \cdots \circ \phi_{\mathcal{W}_{i,1}} \right)(1) \;\le\; \left(\prod_{i=k}^{1} \phi_{\mathcal{W}_i} \right)(1) \;=\; Z_k^{\Pi},$$

where $\prod_{i=k}^{1}$ denotes right-to-left composition. Since $A_k = 1 - Z_k^{\Pi}$ and $A_{k'} = 1 - Z_{k'}^{\Pi'}$, the result follows. □

C.4 MAXIMUM WATERMARK STRENGTH VIA PSEUDORANDOM ACCEPTANCE

Beyond unbiasedness and improved sampling efficiency, we also want each generated token to carry a strong watermark signal to facilitate detection. We now formalize the maximum-watermark-strength guarantee of our verification procedure. Recall from Appendix B.1 that watermark strength is measured by $\mathsf{WS}(\boldsymbol{P}'_\zeta)$ (Eq. (5)),[6] and that an unbiased watermark attains the maximum $\mathsf{WS}(\boldsymbol{P}'_\zeta) = \mathrm{Ent}(\boldsymbol{P})$ iff $\boldsymbol{P}'_\zeta$ is almost surely degenerate. The Gumbel-max and SynthID decoders are already deterministic functions of ζ, but the accept–reject step in our verification procedure injects extra randomness through the uniform threshold $u \sim \mathrm{Unif}[0,1]$ (Line 6 in Alg. 4), which would otherwise prevent $\boldsymbol{P}'_\zeta$ from being degenerate. Following He et al. (2026), we replace the random threshold with a pseudorandom one: the acceptance coin for each branch is derived deterministically from the pseudorandom variables. This makes the entire verification procedure a deterministic function of the pseudorandom variables, achieving maximum watermark strength without affecting unbiasedness and sampling efficiency.

Theorem C.1 (Maximum watermark strength)**.** *Under the setup of Section 3.2, further assume that the watermark scheme $\mathcal{S}$ achieves maximum watermark strength (both Gumbel-max and SynthID ($m \to \infty$) watermarks satisfy this requirement; see Appendix B.1). Then, with pseudorandom acceptance thresholds, the verification procedure achieves maximum watermark strength: $\mathsf{WS}(\boldsymbol{P}'_\zeta) = \mathrm{Ent}(\boldsymbol{P})$, where $\mathrm{Ent}(\boldsymbol{P}) = -\sum_w P(w)\log P(w)$ is the entropy of the target distribution.*

Proof of Theorem C.1. To achieve maximum watermark strength, we replace the random acceptance threshold $u \sim \mathrm{Unif}[0,1]$ at each branch i (Line 6 in Alg. 4) with a pseudorandom threshold ξ_i, derived from a separate group of pseudorandom variables. The variables $\xi_1, \ldots, \xi_k$ are mutually independent and independent of $\zeta^{(0)}, \zeta^{(1)}, \ldots, \zeta^{(k)}$. Each controls a uniform random generator G (i.e., $G(\xi_i) \sim \mathrm{Unif}[0,1]$).

We show that the output distribution $\boldsymbol{P}'_\zeta$ is degenerate (i.e., places all mass on a single token) for every realization of $(\zeta^{(0)}, \ldots, \zeta^{(k)}, \xi_1, \ldots, \xi_k)$.

Consider the verification procedure. At branch $i = 1, \ldots, k$, the candidate token $\tilde{w}_{(w_{<t},\, i)} \sim \mathcal{S}(\boldsymbol{Q}^{(i)}, \zeta^{(i)})$ is deterministic given $(w_{<t}, \zeta^{(i)})$, since $\mathcal{S}$ achieves maximum watermark strength (i.e., $\mathcal{S}(\boldsymbol{Q}^{(i)}, \zeta^{(i)})$ is degenerate). The acceptance ratio $a_i = \min\{1, P^{(i-1)}(\tilde{w}_{(w_{<t},\, i)})/Q^{(i)}(\tilde{w}_{(w_{<t},\, i)})\}$ is therefore a deterministic scalar. The acceptance decision $G(\xi_i) < a_i$ is also deterministic given ξ_i. Thus:

- If $G(\xi_i) < a_i$, the output at this level is the deterministic token $\tilde{w}_{(w_{<t},\, i)}$.
- If $G(\xi_i) \geq a_i$, the residual $\boldsymbol{P}^{(i)} = [\boldsymbol{P}^{(i-1)} - \boldsymbol{Q}^{(i)}]_+$ is deterministic, and verification proceeds to branch $i{+}1$.

If all k branches are rejected, the output is sampled from $\mathcal{S}(\boldsymbol{P}^{(k)}, \zeta^{(0)})$, which is degenerate since $\boldsymbol{P}^{(k)}$ is deterministic (given all preceding decisions) and $\mathcal{S}$ produces degenerate outputs.

In every case, the output token is a deterministic function of $(\zeta^{(0)}, \ldots, \zeta^{(k)}, \xi_1, \ldots, \xi_k)$ and the prefix $w_{<t}$. Hence $\boldsymbol{P}'_\zeta$ (here we abuse notation by extending ζ to also include $(\xi_1, \ldots, \xi_k)$) is degenerate for every realization, and

$$\mathsf{WS}(\boldsymbol{P}'_\zeta) = \mathrm{Ent}(\boldsymbol{P}) - \mathbb{E}_\zeta[\mathrm{Ent}(\boldsymbol{P}'_\zeta)] = \mathrm{Ent}(\boldsymbol{P}).$$

□

Compatibility with unbiasedness. Since $\xi_1, \ldots, \xi_k$ are mutually independent and independent of $\zeta^{(0)}, \ldots, \zeta^{(k)}$, the acceptance decision at branch i remains an unbiased coin with the correct acceptance probability when marginalized over ξ_i. The proof of Theorem 3.1 relies only on the mutual independence of $\zeta^{(0)}, \ldots, \zeta^{(k)}$ and the marginal acceptance probabilities; neither is affected by the introduction of ξ_i. Therefore, unbiasedness $\mathbb{E}_\zeta[\boldsymbol{P}'_\zeta] = \boldsymbol{P}$ continues to hold.

[6] Treating $\boldsymbol{P}'_\zeta$ as a reweighted distribution of $\boldsymbol{P}$, we can equivalently write $\mathsf{WS}(\boldsymbol{P}'_\zeta) = \mathbb{E}_\zeta\big[D_{\mathrm{KL}}(\boldsymbol{P}'_\zeta \,\|\, \boldsymbol{P})\big]$.

Compatibility with improved sampling efficiency. Since $G(\xi_i) \sim \mathrm{Unif}[0,1]$ marginally and $\xi_1, \ldots, \xi_k$ are mutually independent and independent of $\zeta^{(0)}, \ldots, \zeta^{(k)}$, the marginal acceptance probability at each branch i—and hence the rejection probabilities Z_i driving the proof of Theorem 3.2—are unchanged when the uniform threshold u is replaced by $G(\xi_i)$. The proof only uses these marginal acceptance probabilities and the independence structure of ζ; neither is affected by the introduction of ξ_i. Therefore, the sampling efficiency bound $A_k \geq 1 - \mathrm{TV}(\boldsymbol{P}, \boldsymbol{Q})$ continues to hold under pseudorandom acceptance.

Remark C.1. *Watermark strength, defined in Appendix B.1, is conceptually distinct from the watermark detectability evaluated in Section 4. Watermark strength quantifies the dependence between tokens and pseudorandomness, reflecting detection performance under an idealized setting where the true token distributions are known. In contrast, watermark detectability is an empirical measure of detection performance in practical settings where the true distributions are unknown. Therefore, achieving maximum watermark strength does not guarantee strong detectability in practice.*

D ALGORITHMS

Algorithm 1 Standard speculative decoding for watermarked generation (Dathathri et al., 2024; He et al., 2026).

Require: Target model $\boldsymbol{P}$, draft model $\boldsymbol{Q}$, watermark decoder $\mathcal{S}$, lookahead depth L, maximum generation length T, initial prefix $w_{1:n}$, independent pseudorandom variables $(\zeta^{(0)}, \zeta^{(1)})$.

$w_{\mathrm{cur}} \leftarrow w_{1:n}$
while $|w_{\mathrm{cur}}| < T$ **do**
  $L' \leftarrow \min\big(L,\ T - |w_{\mathrm{cur}}|\big)$; $w_{\mathrm{pre}} \leftarrow w_{\mathrm{cur}}$; BONUS $\leftarrow$ True
  ——— ◇ *Draft generation and evaluation* ———
  for $s = 1$ **to** L' **do**
    $\tilde{w}_s \leftarrow \textsc{Sample}\Big(\boldsymbol{Q}_{\zeta^{(1)}}(\cdot \mid w_{\mathrm{pre}})\Big)$; $w_{\mathrm{pre}} \leftarrow (w_{\mathrm{pre}}, \tilde{w}_s)$
  end for
  In parallel: compute $\boldsymbol{P}(\cdot \mid w_{\mathrm{cur}}, \tilde{w}_{1:s-1})$ for $s = 1, \ldots, L'+1$
  ——— ◇ *Draft token verification* ———
  for $s = 1$ **to** L' **do**
    $a \leftarrow \min\left\{1,\ \dfrac{P(\tilde{w}_s \mid w_{\mathrm{cur}})}{Q(\tilde{w}_s \mid w_{\mathrm{cur}})}\right\}$; $u \sim \mathrm{Unif}[0,1]$ ▷ u can be made pseudorandom
    if $u < a$ **then**
      $w_{\mathrm{cur}} \leftarrow (w_{\mathrm{cur}}, \tilde{w}_s)$
    else
      $w \leftarrow \textsc{Sample}\Big([\boldsymbol{P} - \boldsymbol{Q}]_{+,\zeta^{(0)}}\Big)$; $w_{\mathrm{cur}} \leftarrow (w_{\mathrm{cur}}, w)$; BONUS $\leftarrow$ False; **break**
    end if
  end for
  if BONUS $=$ True **then** ▷ all L' tokens accepted; sample bonus token
    $w \leftarrow \textsc{Sample}\Big(\boldsymbol{P}_{\zeta^{(0)}}(\cdot \mid w_{\mathrm{cur}})\Big)$; $w_{\mathrm{cur}} \leftarrow (w_{\mathrm{cur}}, w)$
  end if
end while
return w_{cur}

Algorithm 2 TREESPECWM: Tree speculative decoding for watermark generation.

Require: Target model $\boldsymbol{P}$, draft model $\boldsymbol{Q}$, branching factor k, lookahead depth L, maximum generation length T, initial prefix $w_{1:n}$, watermark decoder $\mathcal{S}$, pseudorandom variables $(\zeta^{(0)}, \zeta^{(1)}, \ldots, \zeta^{(k)})$.

$w_{\text{cur}} \leftarrow w_{1:n}$
while $|w_{\text{cur}}| < T$ **do**
  $L' \leftarrow \min\big(L,\ T - |w_{\text{cur}}|\big)$
  ——— ◇ *Draft Tree Generation* ———
  NODES $\leftarrow$ TREEGENERATION$\big(\boldsymbol{Q},\ k,\ L',\ w_{\text{cur}},\ \mathcal{S},\ (\zeta^{(1)}, \ldots, \zeta^{(k)})\big)$ ▷ Alg. 3
  In parallel: compute $\boldsymbol{P}(\cdot \mid \mathbf{x})$ for every prefix $\mathbf{x}$ in the draft tree: root w_{cur} and all draft-extended prefixes in NODES.
  ——— ◇ *Draft Tree Verification* ———
  $w_{\text{cur}} \leftarrow$ TREEVERIFICATION$\big(\boldsymbol{P},\ k,\ L',\ w_{\text{cur}},\ \mathcal{S},\ (\zeta^{(0)}, \ldots, \zeta^{(k)}),\ \text{NODES}\big)$ ▷ Alg. 4
end while
return w_{cur}

Algorithm 3 TREEGENERATION: Draft tree generation with vocabulary partition strategy.

Require: Draft model $\boldsymbol{Q}$, branching factor k, lookahead depth L, root prefix $w_{<t}$, partition rule $\mathcal{A}$, watermark decoder $\mathcal{S}$, pseudorandom variables $(\zeta^{(1)}, \ldots, \zeta^{(k)})$.

FRONTIER $\leftarrow \{w_{<t}\}$, NODES $\leftarrow$ MAP()
for $d = 1$ to L **do**
  NEXTFRONTIER $\leftarrow \{\}$
  for each $w_{\text{pre}} \in$ FRONTIER **do**
    $(\mathcal{W}_1, \ldots, \mathcal{W}_k) \leftarrow \mathcal{A}(w_{\text{pre}})$ ▷ vocabulary partition; $\mathcal{W}_i$ disjoint, $\bigcup_i \mathcal{W}_i = \mathcal{W}$
    for $i = 1$ to k **do**
      $\boldsymbol{Q}^{(i)}(\cdot \mid w_{\text{pre}}) \leftarrow$ RESTRICTANDRENORM$(\boldsymbol{Q}(\cdot \mid w_{\text{pre}}),\ \mathcal{W}_i)$ ▷ Eq. 1
      $\tilde{w}_{(w_{\text{pre}},\, i)} \leftarrow$ SAMPLE$\Big(\boldsymbol{Q}^{(i)}_{\zeta^{(i)}}(\cdot \mid w_{\text{pre}})\Big)$
      NODES$[w_{\text{pre}},\ i] \leftarrow \big(\tilde{w}_{(w_{\text{pre}},\, i)},\ \boldsymbol{Q}^{(i)}(\cdot \mid w_{\text{pre}})\big)$ ▷ Store the token and its probability.
      NEXTFRONTIER $\leftarrow$ NEXTFRONTIER $\cup\ \{(w_{\text{pre}},\ \tilde{w}_{(w_{\text{pre}},\, i)})\}$
    end for
  end for
  FRONTIER $\leftarrow$ NEXTFRONTIER
end for
return NODES: a dictionary storing all draft tokens and their probability distributions under the draft model, keyed by the tuple (token prefix, branch index).

Algorithm 4 TREEVERIFICATION: Draft tree verification.

Require: Target model $\boldsymbol{P}$, branching factor k, lookahead depth L, root prefix $w_{<t}$, watermark decoder $\mathcal{S}$, pseudorandom variable $\zeta^{(0)}$ (independent of $(\zeta^{(1)}, \ldots, \zeta^{(k)})$), draft token tree NODES (from Alg. 3).

$w_{\text{cur}} \leftarrow w_{<t}$; BONUS $\leftarrow$ True
for $d = 1$ to L **do**
  $\boldsymbol{P}^{(0)}(\cdot \mid w_{\text{cur}}) \leftarrow \boldsymbol{P}(\cdot \mid w_{\text{cur}})$; ACCEPTED $\leftarrow$ False
  for $i = 1$ to k **do**
    retrieve $\big(\tilde{w}_{(w_{\text{cur}},\, i)},\ \boldsymbol{Q}^{(i)}(\cdot \mid w_{\text{cur}})\big)$ from NODES$[w_{\text{cur}},\, i]$
    $a_i \leftarrow \min\Big\{1,\ \frac{P^{(i-1)}(\tilde{w}_{(w_{\text{cur}},\, i)} \mid w_{\text{cur}})}{Q^{(i)}(\tilde{w}_{(w_{\text{cur}},\, i)} \mid w_{\text{cur}})}\Big\}$; $u \sim \text{Unif}[0,1]$ ▷ u can be made pseudorandom
    if $u < a_i$ **then**
      $w_{\text{cur}} \leftarrow (w_{\text{cur}},\ \tilde{w}_{(w_{\text{cur}},\, i)})$; ACCEPTED $\leftarrow$ True ; **break**.
    else
      $\boldsymbol{P}^{(i)}(\cdot \mid w_{\text{cur}}) \leftarrow \big[\boldsymbol{P}^{(i-1)}(\cdot \mid w_{\text{cur}}) - \boldsymbol{Q}^{(i)}(\cdot \mid w_{\text{cur}})\big]_+$
    end if
  end for
  if ACCEPTED $=$ False **then**
    $w \leftarrow \text{SAMPLE}\Big(\boldsymbol{P}^{(k)}_{\zeta^{(0)}}(\cdot \mid w_{\text{cur}})\Big)$; $w_{\text{cur}} \leftarrow (w_{\text{cur}},\ w)$; BONUS $\leftarrow$ False ; **break**
  end if
end for
if BONUS $=$ True **then** ▷ all L levels accepted; sample bonus token
  $w \leftarrow \text{SAMPLE}\big(\boldsymbol{P}_{\zeta^{(0)}}(\cdot \mid w_{\text{cur}})\big)$; $w_{\text{cur}} \leftarrow (w_{\text{cur}},\ w)$
end if
return w_{cur}

E Additional Experiment Results

E.1 Implementation details

Setup. We load all models from Hugging Face, using `FP16` precision (with Qwen in `BF16`). Both the target and draft models are deployed on a single NVIDIA H200 GPU. We evaluate sampling efficiency and wall-clock time using 100 prompts per configuration, with a maximum generation length of 200 tokens.

Vocabulary partition. We instantiate the partition rule $\mathcal{A}$ via a seeded random permutation of the vocabulary. Given a prefix $w_{<t}$, we hash it into a random seed, draw a permutation of the index set $\{1, \ldots, |\mathcal{W}|\}$ with that seed, and split the permuted indices into k contiguous blocks of equal size,[7] yielding $(\mathcal{W}_1, \ldots, \mathcal{W}_k)$. Notice that this is only one possible instantiation. The key requirement is that *the partition be recoverable during detection*, so any deterministic rule satisfying this property is applicable. In practice, one may use a truncated prefix (e.g., an n-gram) instead of the full context, and optionally incorporate a secret key into the hash function to enhance security. Importantly, in our experiments, when increasing the branching factor k from 2 to 4, we do not enforce the refinement condition in Thm. 3.3. Instead, we always partition the permuted vocabulary into k equal-sized contiguous blocks as described above.

E.1.1 Details of watermark detection.

Temperature. In Section 4, for evaluating watermark detectability, we set the temperature to 0.7 for both `OPT` and `LLaMA` model pairs, and to 1.0 for `Qwen`, due to its lower entropy compared to the other models.

Watermark detection methods. As shown in Section 3, the vocabulary partition strategy reduces pseudorandomness ambiguity from $k+1$ to 2: the two candidate sources for each token are the draft branch identified by its vocabulary subset and the source $\zeta^{(0)}$ used for residual or bonus sampling. This matches the setting of standard speculative sampling for watermark generation (Dathathri et al., 2024; He et al., 2026). Moreover, following He et al. (2026), we incorporate pseudorandom acceptance during draft tree verification (see Line 6 in Alg. 4). We therefore adapt the detection protocol of Seq He et al. (2026), using the pseudorandom acceptance variable associated with the identified draft branch to help distinguish between the two candidate sources. We briefly outline this detection method below.

Gumbel-max watermark. We adapt the Ars-τ detector of He et al. (2026) from the single-branch to our k-branch setting. The single design change is how the per-token test statistic is selected: We select the per-token test statistic by comparing the pseudorandom acceptance variable of the identified draft branch against a calibrated threshold τ.

Recall that the Gumbel-max pseudorandom variable at branch i has the form $\zeta^{(i)} = (U_w^{(i)})_{w \in \mathcal{W}}$, a collection of i.i.d. $\mathrm{Unif}[0,1]$ variables. The classical Aaronson test (Aaronson, 2023) uses the component at the generated token as the per-step statistic: under watermarking it concentrates near 1, and under non-watermarked text it remains uniform. The detector flags watermarking when $\sum_t -\log(1 - y_t)$ exceeds a null quantile.

In Alg. 2, with the vocabulary partition strategy, generated token w_t may originate from one of the k draft branches (accept) or from the residual distribution $\boldsymbol{P}^{(k)}_{\zeta^{(0)}}$ (reject). This corresponds to two candidate statistics: one for accept ($y_t^{(i^\star)}$) and another one for reject ($y_t^{(0)}$). The pseudorandom acceptance variables $u_t^{(i)} \sim \mathrm{Unif}[0,1]$ for $i = 1, \ldots, k$ (Line 6 in Alg. 4) further help distinguish between these two cases: a token is accepted if any branch passes its threshold test. Unlike the baseline, which has a single acceptance variable, we have k (one per branch). For an observed token w_t, let $i^\star$ be the unique index such that $w_t \in \mathcal{W}_{i^\star}$. The two candidate statistics are $y_t^{(i^\star)} = U_{w_t}^{(i^\star)}$ for the identified draft branch and $y_t^{(0)} = U_{w_t}^{(0)}$ for the source used in residual or bonus sampling. We use

[7] If $k \nmid |\mathcal{W}|$, the first $|\mathcal{W}| \bmod k$ blocks receive one extra element.

the corresponding pseudorandom acceptance variable $u_t^{(i^\star)}$ to select between them via the Ars-τ rule:

$$y_t \;=\; y_t^{(i^\star)}\,\mathbf{1}_{u_t^{(i^\star)}<\tau} \;+\; y_t^{(0)}\,\mathbf{1}_{u_t^{(i^\star)}\geq\tau}. \tag{11}$$

The threshold τ is calibrated on the training split via grid search, selecting the value that maximizes the TPR at FPR$= 1\%$. The final watermark test applies the Aaronson statistic $\sum_t -\log(1-y_t)$ to the selected $\{y_t\}$.

In our experiments, we generate 500 text sequences with the Gumbel-max watermark and evaluate detection over 10 random train/test splits of the usable sequences. We recalibrate τ on each training split and report mean test TPR with 95% confidence intervals. These intervals reflect variability across splits and threshold calibration on fixed generated texts.

SynthID watermark. As in the Gumbel-max case, we identify the candidate draft branch via the vocabulary partition and use its corresponding pseudorandom acceptance variable for detection.

Recall that the SynthID pseudorandom variable at branch i takes the form $\zeta^{(i)} = (\boldsymbol{g}_j^{(i)})_{j=1}^m$ with each $\boldsymbol{g}_j^{(i)} \in \{0,1\}^{|\mathcal{W}|}$. For a generated token w_t, the candidate statistic at branch i is the m-dimensional bit vector

$$y_t^{(i)} \;:=\; \left(g_{1,w_t}^{(i)},\; g_{2,w_t}^{(i)},\; \ldots,\; g_{m,w_t}^{(i)}\right) \;\in\; \{0,1\}^m,$$

which collects the w_t-indexed entry across all m tournament rounds. The key detection principle, established by the tournament construction (Dathathri et al., 2024), is that the entries of $y_t^{(i)}$ are biased toward 1 when $\zeta^{(i)}$ is the pseudorandom variable that actually generated w_t, and remain near-uniform in $\{0,1\}$ otherwise. Based on this, Dathathri et al. (2024) introduce a Bayesian detector, and He et al. (2026) extend it to incorporate the pseudorandom acceptance variable.

For detection of Alg. 2, as in Gumbel-max, each token yields two candidate statistics: $y_t^{(i^\star)}$ for the draft branch identified by the vocabulary partition and $y_t^{(0)}$ for the source used in residual or bonus sampling. We use the corresponding pseudorandom acceptance variable $u_t^{(i^\star)}$ to help distinguish between these two sources. We then use the Bayesian detector from He et al. (2026), which takes the two candidate statistics together with $u_t^{(i^\star)}$ and outputs the probability that the text is watermarked. Since this detector is more involved than in the Gumbel-max case, we refer to Dathathri et al. (2024); He et al. (2026) for the full construction.

Since the SynthID watermark detector requires more parameters, we generate 2,000 text sequences in our experiments, split into 1,000 for training and 1,000 for testing. We repeat the random split and Bayesian detector fitting 10 times on the same generated texts and report mean test TPR with 95% confidence intervals.

The $k{+}1$ *PRs* ablation. As described in Section 4, we use this detection scheme as an ablation of partition-based branch identification. We keep the generated texts unchanged but do not use the vocabulary partition to identify the draft-source statistic for each token. Each token then has $k{+}1$ candidate statistics: k from the draft sources ($\{y_t^{(i)}\}_{i=1}^k$) and one from the source used for residual or bonus sampling ($y_t^{(0)}$). Without partition-based branch identification, the detector lacks a direct correspondence between the observed token and a particular draft statistic. We therefore select one of the k draft statistics uniformly at random, treating the candidate draft sources symmetrically. This provides a clear reference for evaluating the benefit of partition-based branch identification.

E.2 One-step generation

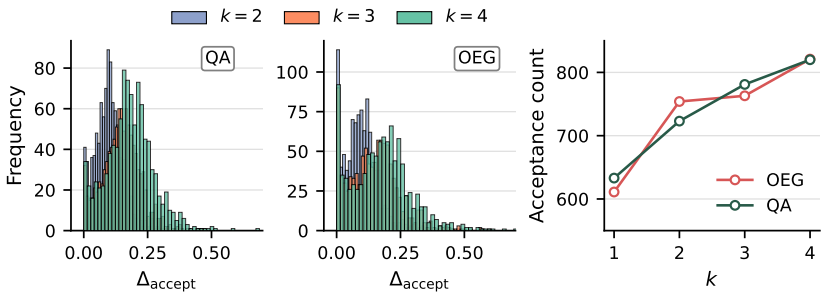


Figure 5: One-step sampling efficiency gain $\Delta_{\text{accept}} = A_k - A_1$ of Alg. 2 over the baseline. **Left**: histograms over 10^3 real next-token distributions from `OPT-30B/OPT-125M`. **Right**: total accepted tokens (out of 10^3) versus k for question answering (QA) and open-ended generation (OEG).

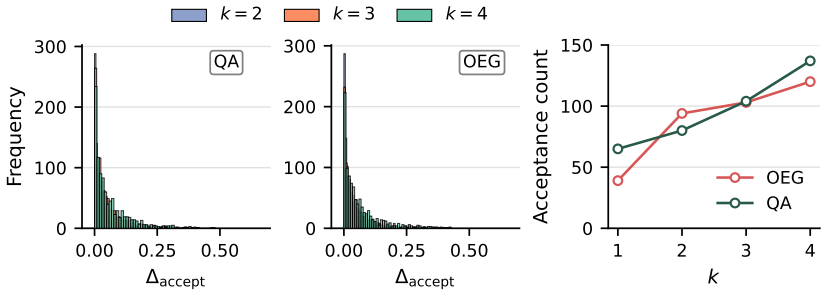


Figure 6: One-step sampling efficiency gain $\Delta_{\text{accept}} = A_k - A_1$ of Alg. 2 over the baseline. **Left**: histograms over 10^3 real next-token distributions from `Llama-13B/Llama-68M`. **Right**: total accepted tokens (out of 10^3) versus k for question answering (QA) and open-ended generation (OEG).

E.3 Real generation tasks

E.3.1 Sampling efficiency and throughput

Table 3 provides the numerical results underlying Fig. 3. We use the `OPT-30B/OPT-125M`, `Llama-13B/Llama-68M`, and `Qwen3-32B/Qwen3-0.6B` target–draft pairs on QA (`ELI5`) and OEG (`C4`). With $L = 3$, increasing k improves AATPS across all settings and yields higher throughput than Seq, although throughput does not always increase monotonically with k. App. E.5 analyzes the additional work per decoding step.

Table 4 extends the evaluation to lookahead depths $L = 2, 3, 4$, with the tree branching factor fixed at $k = 2$. Tree achieves higher AATPS than Seq at every evaluated depth on both tasks.

Table 3: AATPS and throughput (tokens/s) with varying branching factor k and fixed lookahead depth $L = 3$. Higher values are better; bold marks the best configuration in each metric row.

Model	**Watermark**	**Metric**	Seq	$k = 2$	$k = 3$	$k = 4$
Question answering (QA)						
OPT	Gumbel-max	AATPS	2.11	2.54	2.77	**2.90**
		Throughput	66.9	72.7	76.0	**79.6**
	SynthID	AATPS	2.26	2.66	2.82	**2.93**
		Throughput	62.4	67.8	**71.1**	69.3
Llama	Gumbel-max	AATPS	1.96	2.27	2.43	**2.64**
		Throughput	68.8	74.1	76.2	**79.1**
	SynthID	AATPS	2.04	2.38	2.54	**2.72**
		Throughput	60.8	66.7	67.8	**69.8**
Qwen	Gumbel-max	AATPS	2.21	2.53	2.75	**2.88**
		Throughput	22.6	24.4	25.6	**26.5**
	SynthID	AATPS	2.22	2.58	2.75	**2.91**
		Throughput	21.0	23.1	24.4	**25.6**
Open-ended generation (OEG)						
OPT	Gumbel-max	AATPS	2.20	2.56	2.77	**2.90**
		Throughput	65.6	69.9	**73.1**	72.9
	SynthID	AATPS	2.35	2.67	2.89	**2.95**
		Throughput	60.0	63.2	**65.5**	62.8
Llama	Gumbel-max	AATPS	1.93	2.23	2.43	**2.54**
		Throughput	69.9	73.7	77.7	**79.3**
	SynthID	AATPS	2.01	2.29	2.49	**2.57**
		Throughput	58.4	62.9	65.8	**66.0**
Qwen	Gumbel-max	AATPS	2.06	2.37	2.59	**2.71**
		Throughput	20.1	21.9	23.9	**24.2**
	SynthID	AATPS	2.12	2.44	2.59	**2.75**
		Throughput	19.9	22.0	23.3	**24.2**

Table 4: AATPS and throughput (tokens/s) with varying lookahead depth L and fixed branching factor $k = 2$. Higher values are better; bold marks the higher value within each Seq/Tree pair for each metric.

Model	Watermark	Metric	$L = 2$		$L = 3$		$L = 4$	
			Seq	Tree	Seq	Tree	Seq	Tree
Question answering (QA)								
OPT	Gumbel-max	AATPS	1.90	**2.23**	2.11	**2.54**	2.26	**2.78**
		Throughput	68.3	**76.4**	66.9	**72.7**	65.2	**71.5**
	SynthID	AATPS	2.02	**2.29**	2.26	**2.66**	2.43	**3.00**
		Throughput	65.7	**68.5**	62.4	**67.8**	60.7	**67.9**
Llama	Gumbel-max	AATPS	1.82	**2.04**	1.96	**2.27**	2.05	**2.43**
		Throughput	70.6	**72.9**	68.8	**74.1**	68.3	**70.8**
	SynthID	AATPS	1.86	**2.07**	2.04	**2.38**	2.10	**2.57**
		Throughput	58.9	**60.8**	60.8	**66.7**	56.1	**62.0**
Qwen	Gumbel-max	AATPS	2.00	**2.20**	2.21	**2.53**	2.34	**2.81**
		Throughput	23.4	**25.9**	22.6	**24.4**	18.8	**21.5**
	SynthID	AATPS	2.02	**2.23**	2.22	**2.58**	2.40	**2.85**
		Throughput	22.9	**24.6**	21.0	**23.1**	18.6	**21.2**
Open-ended generation (OEG)								
OPT	Gumbel-max	AATPS	1.98	**2.18**	2.20	**2.56**	2.35	**2.85**
		Throughput	68.0	**69.3**	65.6	**69.9**	62.1	**68.8**
	SynthID	AATPS	2.07	**2.25**	2.35	**2.67**	2.54	**2.99**
		Throughput	61.2	**62.6**	60.0	**63.2**	57.3	**61.4**
Llama	Gumbel-max	AATPS	1.80	**2.00**	1.93	**2.23**	2.00	**2.40**
		Throughput	**70.1**	65.4	69.9	**73.7**	63.4	**71.6**
	SynthID	AATPS	1.86	**2.04**	2.01	**2.29**	2.10	**2.47**
		Throughput	**61.4**	60.4	58.4	**62.9**	54.2	**61.8**
Qwen	Gumbel-max	AATPS	1.88	**2.12**	2.06	**2.37**	2.18	**2.60**
		Throughput	22.6	**25.0**	20.1	**21.9**	18.4	**20.6**
	SynthID	AATPS	1.92	**2.13**	2.12	**2.44**	2.20	**2.68**
		Throughput	21.2	**23.9**	19.9	**22.0**	17.3	**19.8**

E.3.2 WATERMARK DETECTABILITY

Table 5: Watermark detectability (TPR@FPR$=1\%$) on the OEG task at 200 tokens (mean $\pm$ 95% CI). The $k{+}1$ *PRs* setting has $(k{+}1)$-way pseudorandomness ambiguity, compared with two-way ambiguity in our method.

TPR@FPR1%	Gumbel-max			SynthID		
OEG	OPT	Llama	Qwen	OPT	Llama	Qwen
Seq, $k=1$	98.54±0.71	97.62±0.92	99.23±0.21	95.82±0.42	95.66±0.56	98.41±0.23
$k=2$ (3 *PRs*)	91.36±1.57	92.90±1.61	98.79±0.19	79.58±1.60	79.36±1.51	94.92±0.74
$k=2$	96.90±0.60	96.28±0.76	98.95±0.37	95.35±0.43	93.12±0.78	98.46±0.21
$k=3$ (4 *PRs*)	84.98±1.31	89.71±0.97	96.33±0.72	60.84±1.78	64.15±4.01	87.40±1.05
$k=3$	96.05±0.80	94.19±0.78	99.07±0.71	94.22±0.78	92.56±1.40	98.68±0.15
$k=4$ (5 *PRs*)	83.56±1.51	83.15±1.67	94.09±0.43	47.23±4.55	50.34±3.72	82.27±2.64
$k=4$	95.94±0.76	93.47±1.11	98.09±0.41	93.87±0.37	91.48±0.84	97.71±0.28

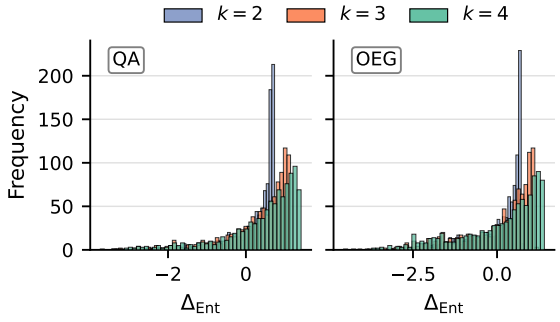


Figure 7: Per-prefix entropy gap: $\mathrm{Ent}(\boldsymbol{Q}) - \frac{1}{k}\sum_i \mathrm{Ent}(\boldsymbol{Q}^{(i)})$ on `OPT-125M`.

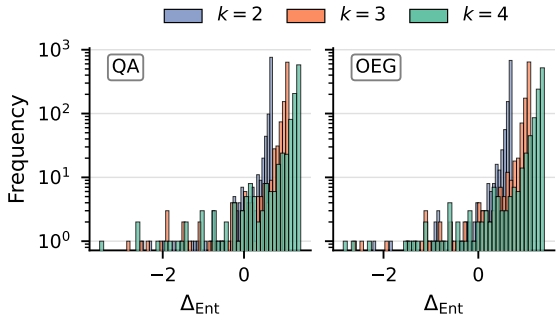


Figure 8: Per-prefix entropy gap: $\mathrm{Ent}(\boldsymbol{Q}) - \frac{1}{k}\sum_i \mathrm{Ent}(\boldsymbol{Q}^{(i)})$ on `Llama-68M`.

E.3.3 COMPATIBILITY WITH AN EAGLE-STYLE DRAFT HEAD

Setup. We train an EAGLE-style draft head for `Qwen3-32B` following Li et al. (2024a), using 100,000 prompts from `ELI5` (QA) datasets. The head consists of a single decoder layer trained on hidden states from continuations generated by the frozen target. We compare it with the standalone `Qwen3-0.6B` drafter on QA and OEG under Gumbel-max and SynthID, using watermarked partition trees with $k = 2, 3, 4$ and $L = 3$. Efficiency is evaluated on 100 prompts per task, with a maximum output length of 200 tokens and temperature 1.0.

Sampling efficiency and throughput. `Qwen3-0.6B` is a strong drafter whose distribution closely matches `Qwen3-32B`, yielding consistently higher AATPS than EAGLE (Table 6). Its forward passes are more expensive than those of the single-layer EAGLE head, which achieves higher throughput despite lower draft acceptance. The AATPS gap is small on QA but widens on OEG, consistent with the head being trained on `ELI5`. EAGLE nevertheless remains faster on OEG, showing that its speed advantage extends beyond the training domain. For both drafters, AATPS increases with k, whereas EAGLE throughput peaks at $k = 3$.

Table 6: AATPS and throughput (tokens/s) for `Qwen3-32B` with a standalone `Qwen3-0.6B` drafter or an EAGLE head.

Task	Watermark	k	AATPS		Throughput	
			0.6B	EAGLE	0.6B	EAGLE
QA	Gumbel-max	2	**2.53**	2.49	24.4	**42.2**
		3	**2.75**	2.62	25.6	**44.4**
		4	**2.88**	2.72	26.5	**42.7**
	SynthID	2	**2.58**	2.54	23.1	**40.2**
		3	**2.75**	2.72	24.4	**41.6**
		4	**2.91**	2.86	25.6	**40.7**
OEG	Gumbel-max	2	**2.37**	1.88	21.9	**32.5**
		3	**2.59**	2.02	23.9	**33.0**
		4	**2.71**	2.07	24.2	**32.9**
	SynthID	2	**2.44**	1.92	22.0	**29.5**
		3	**2.59**	2.09	23.3	**30.8**
		4	**2.75**	2.23	24.2	**29.8**

Watermark detectability. We evaluate detectability following the setup described in App. E.1.1. Table 7 shows that EAGLE maintains detection rates comparable to those of the standalone drafter. The small differences are consistent with the role of draft-distribution entropy discussed in Section 3.3: changing the drafter changes the proposal distribution, whose entropy influences the watermark signal.

Table 7: Watermark detectability (TPR@FPR$= 1\%$) with a standalone `Qwen3-0.6B` drafter or an EAGLE head.

Configuration	Gumbel-max		SynthID	
	QA	OEG	QA	OEG
$k = 2$ (0.6B)	96.73	98.95	91.82	98.46
$k = 2$ (EAGLE)	98.30	99.51	93.19	99.15
$k = 3$ (0.6B)	93.90	99.07	90.87	98.68
$k = 3$ (EAGLE)	96.27	99.71	92.88	98.83
$k = 4$ (0.6B)	95.04	98.09	88.55	97.71
$k = 4$ (EAGLE)	95.47	99.43	89.53	98.91

More broadly, these results show that our vocabulary partition strategy enables tree-based watermark generation without being tied to a particular drafter. By operating on the draft distribution, our framework supports both conventional standalone models and more advanced draft heads such as EAGLE. This flexibility allows watermarked generation to benefit from advances in drafter design, achieving further throughput gains while maintaining detectability.

E.4 Vocabulary Partition Accelerates Text Generation Without Watermarking

We evaluate vocabulary partition without watermarking using `Qwen3-32B`/`Qwen3-0.6B` as the target–draft pair on QA (`ELI5`) and OEG (`C4`). We compare single-sequence speculative decoding (Seq) with partition-based trees ($k = 2, 3, 4$), using $L = 3$, temperature $T = 1.0$, and 100 prompts per task with a maximum generation length of 200 tokens. As shown in Table 8, all tested partition trees improve both AATPS and throughput over Seq. At $k = 4$, AATPS improves by 28.5% and 31.6% over Seq on QA and OEG, respectively, with corresponding throughput speedups of $1.26\times$ and $1.23\times$. These results show that vocabulary partition improves both sampling efficiency and generation speed in these non-watermarked settings.

Table 8: Throughput of non-watermarked generation with `Qwen3-32B`/`Qwen3-0.6B`.

Task	**Metric**	Seq	$k = 2$	$k = 3$	$k = 4$
QA (ELI5)	AATPS	2.237	2.561	2.737	2.874
	Throughput (tokens/s)	21.39	25.18	26.75	26.85
OEG (C4)	AATPS	2.074	2.373	2.583	2.729
	Throughput (tokens/s)	20.84	22.84	25.42	25.57

E.5 Wall-clock time analysis

Measurement setup. To explain the throughput trends in App. E.3.1, we profile the same three target–draft model pairs on a single NVIDIA H200 GPU, using Gumbel-max on OEG with $L = 3$, temperature $T = 1.0$, and a maximum generation length of 200 tokens. Each configuration uses 100 prompts for timing and 30 for component profiling. Table 9 reports these costs in milliseconds per decoding step.

Cost attribution. Draft and Target denote model forward passes. Sample includes vocabulary-partition hashing, branch-distribution normalization, and watermark sampling; Verify covers acceptance decisions, residual updates, and fallback or bonus-token sampling. Tree includes tree construction and attention-mask preparation, while KV covers tensor assembly and cache operations. Other accounts for time outside these components. Alongside Seq and partition trees with $k = 2, 3, 4$, we include a single-branch Control using the tree implementation. Its additional 1.7–2.2 ms per step over Seq shows that some overhead is incurred before adding branches.

Sources of additional cost. Batched execution limits the growth of model-forward time for these small trees. Each decoding step still uses L draft forward passes and one target forward pass, while processing more token positions per call. For `OPT`, moving from Seq to $k = 4$ increases the target's input from 4 to 85 positions, but draft and target forward times rise only from 9.08 to 9.67 ms and from 19.57 to 20.84 ms, respectively. Total time per step increases from 34.18 to 41.44 ms, with much of the additional cost coming from sampling, verification, and tree and cache operations.

Implications for throughput. The throughput results in App. E.3.1 illustrate the balance between producing more tokens per step and spending more time on each step. In Table 3, for `OPT` with SynthID on OEG, increasing k from 3 to 4 raises AATPS from 2.89 to 2.95, but lowers throughput from 65.5 to 62.8 tokens/s. Table 4 also shows that, for `Llama` on OEG at $L = 2$, Tree has higher AATPS but lower throughput than Seq under both watermarks. Thus, higher sampling efficiency does not guarantee higher throughput for every configuration. The choice of k and L should be guided by measured throughput, which reflects both the sampling gain and the work required per step.

E.6 Sparse Draft Trees

In non-watermarked generation, methods such as EAGLE-2 (Li et al., 2024b) use dynamic draft trees to allocate computation to more promising continuations and improve decoding speed. We investigate whether this idea can further accelerate watermarked generation with our framework.

To preserve source identification during watermark detection, we require pruning to maintain the correspondence between vocabulary subsets and branch-specific pseudorandom sources. Every

Table 9: Per-step cost breakdown (ms) with Gumbel-max on OEG, $L = 3$. Control denotes a single-branch run through the tree implementation. Component shares are rescaled to the total measured without instrumentation; small differences in row sums are due to rounding.

Config.	Draft	Sample	Target	Verify	Tree	KV	Other	Total
`OPT-30B/OPT-125M`								
Seq	9.08	2.66	19.57	1.50	0.27	0.73	0.37	34.18
Control	9.53	2.70	19.05	1.69	1.00	1.49	0.80	36.27
$k = 2$	9.56	2.92	19.32	2.12	1.16	1.49	0.82	37.39
$k = 3$	9.62	3.21	20.02	2.48	1.43	1.49	0.87	39.13
$k = 4$	9.67	3.67	20.84	2.86	1.97	1.48	0.94	41.44
`Llama-13B/Llama-68M`								
Seq	3.77	2.52	19.53	1.41	0.22	0.48	0.24	28.18
Control	3.94	2.53	19.23	1.58	0.92	1.01	0.68	29.89
$k = 2$	4.07	2.77	18.82	2.00	1.09	1.01	0.71	30.47
$k = 3$	4.27	3.09	18.92	2.45	1.36	1.04	0.79	31.91
$k = 4$	4.30	3.54	19.34	2.85	1.86	1.05	0.88	33.83
`Qwen3-32B/Qwen3-0.6B`								
Seq	55.49	3.09	44.83	1.83	0.27	0.84	0.31	106.65
Control	56.31	2.82	44.07	1.74	1.03	2.02	0.90	108.88
$k = 2$	57.95	3.35	45.36	2.48	1.36	2.15	1.10	113.74
$k = 3$	58.15	3.85	45.55	2.94	1.64	2.17	1.19	115.48
$k = 4$	57.24	4.44	44.55	3.01	2.03	2.02	1.07	114.35

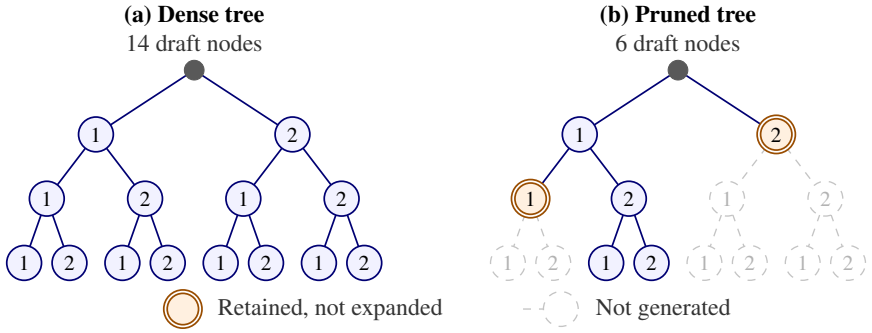


Figure 9: Dense and pruned draft trees for $k = 2, L = 3$. Every expanded node retains both partition children; pruning stops further expansion at selected nodes. Gray dashed subtrees indicate descendants that are not generated.

expanded node therefore retains all k partition children; pruning only determines which children receive further descendants (Fig. 9). This keeps candidate branch identities recoverable from each token and its prefix, while leaving branch sampling and verification unchanged. Under this constraint, pruning can accelerate larger trees, but none of the tested configurations improves latency over the dense $k = 4, L = 3$ reference.

Setup and pruning strategies. We evaluate the same three target–draft model pairs as in Section 4, using Gumbel-max on QA (`ELI5`), temperature $T = 1.0$, and 200 prompts with a maximum generation length of 200 tokens. Every expanded node generates and retains all k partition children; pruning only determines which children receive further descendants. Thus, branch sampling and verification remain unchanged.

We consider two pruning strategies.

- *Budget pruning* expands two nodes at depth 1 and four at depth 2, yielding 28 draft nodes at $k = 4, L = 3$. RANDOM selects nodes uniformly, whereas DYNAMIC ranks them by the product of draft partition masses along their paths.

Table 10: Sparse-tree results on QA (ELI5) with Gumbel-max. Nodes exclude the root; bold marks the lowest latency per model across both panels.

Configuration	Nodes/step	AATPS ↑	ms/token ↓
(a) Pruning a small tree: $k = 4, L = 3$			
OPT-30B/OPT-125M			
Dense	83.5	2.858	**13.29**
Random $(2, 4)$	28.0	2.288	16.13
Dynamic $(2, 4)$	28.0	2.529	14.66
Threshold $\tau = 0.02$	74.7	2.871	13.93
Llama-13B/Llama-68M			
Dense	84.0	2.621	**13.87**
Random $(2, 4)$	28.0	2.050	17.71
Dynamic $(2, 4)$	28.0	2.393	15.33
Threshold $\tau = 0.02$	73.8	2.619	15.38
Qwen3-32B/Qwen3-0.6B			
Dense	84.0	2.867	**42.94**
Random $(2, 4)$	28.0	2.203	55.70
Dynamic $(2, 4)$	28.0	2.691	46.05
Threshold $\tau = 0.02$	56.0	2.862	43.55
(b) Pruning a larger tree: $k = 8, L = 3$			
OPT-30B/OPT-125M			
Dense	577.5	3.184	29.04
Threshold $\tau = 0.05$	283.3	3.037	21.35
Llama-13B/Llama-68M			
Dense	584.0	2.896	24.50
Threshold $\tau = 0.05$	300.2	2.817	20.87
Qwen3-32B/Qwen3-0.6B			
Dense	584.0	3.170	56.98
Threshold $\tau = 0.05$	127.6	2.991	43.22

- *Threshold pruning* expands a node only if its incoming branch has draft mass $\alpha_i = \boldsymbol{Q}(\mathcal{W}_i \mid w_{\text{pre}}) \geq \tau$. We test $\tau \in \{0.02, 0.05, 0.10\}$ at $k = 4, L = 3$, and $\tau = 0.05$ for larger trees with $(k, L) \in \{(6, 3), (8, 3), (4, 4), (4, 5)\}$, each with a dense control.

Table 10 presents representative configurations.

Fewer nodes do not guarantee lower latency. At $k = 4, L = 3$, Dynamic achieves higher AATPS than Random at the same node budget, but still loses 6–12% AATPS relative to Dense and increases latency by 7–11%. Threshold pruning can retain nearly all AATPS without improving latency: for Qwen, $\tau = 0.02$ removes one third of the nodes, with AATPS changing from 2.867 to 2.862, yet latency changes from 42.94 to 43.55 ms/token.

For larger trees, pruning does reduce latency relative to the corresponding dense control. At $k = 8, L = 3$, all three model pairs benefit, but the resulting latency remains above that of the small dense reference; Qwen comes closest at 43.22 versus 42.94 ms/token.

These results are consistent with the cost analysis in App. E.5: batched model forward passes are relatively insensitive to the width of small trees, and reducing width alone does not remove sequential drafting across levels. Pruning therefore needs to save enough time per step to offset any loss in tokens generated per step. The tested strategies do not achieve this balance relative to the small dense tree, motivating future work that jointly optimizes tree expansion and execution cost.

F Broader Impacts

Our work proposes a tree-based speculative decoding algorithm that accelerates watermarked generation for LLMs. The most direct societal benefit is practical: watermarking has long been viewed as a

promising tool for establishing content provenance and curbing the misuse of generative AI, but its inference-time overhead has been a recurring obstacle to large-scale deployment. By narrowing this efficiency gap, our method makes it more feasible for model providers to ship watermarking on by default, which in turn strengthens infrastructure for tracing AI-generated content, attributing outputs, and supporting downstream tools that rely on reliable watermark signals.

We also note several risks worth acknowledging. First, our theoretical analysis shows that the vocabulary partition strategy improves sampling efficiency even in the absence of watermarking, meaning the algorithm itself is a general-purpose acceleration technique. Faster LLM inference is broadly beneficial, but it also reduces the marginal cost of generating high-volume text content, which can be exploited for spam, automated disinformation campaigns, or scaled influence operations. Second, broader deployment of watermarking itself raises concerns around user awareness and consent, particularly when end users are not told that their generated outputs carry recoverable signals tied to a specific provider.

To mitigate these risks, it is important that watermarking schemes built on our method are deployed transparently, with clear disclosure to end users about whether their outputs are watermarked and how the resulting signals may be used. Our work focuses on the algorithmic side—preserving unbiasedness and detectability while improving efficiency—and does not change the underlying watermarking semantics, so existing best practices for responsible watermark deployment continue to apply. In conclusion, while accelerating watermark generation can make content attribution more practical and support responsible AI use, it is important to consider and address potential societal risks to ensure ethical and transparent deployment.