SPECPF: SPECULATIVE PERMUTE-FLIP DECODING WITH SIGNATURE-FILTERED WATERMARKING

Anonymous authors
Paper under double-blind review

ABSTRACT

We study speculative decoding for the permute-and-flip (PF) decoder, proposed by Zhao et al. (ICLR 2025) while preserving its native watermark detectability. Our method, SpecPF, speculatively accelerates the recently proposed permute-and-flip decoder while preserving the correct output distribution, and hence all of its desirable properties, including optimal stability, natural watermarkability, and Pareto-optimality with respect to the stability–perplexity trade-off. We design a novel watermark scheme, namely a cross-fitted bigram filter followed by the exact PF Fisher/Gamma test. Our watermark detection method is tailored to the permute-and-flip distribution and is model-agnostic, consistently outperforming the original watermarking scheme. While keeping the target distribution exactly unchanged and hence inheriting all desirable properties of the original PF decoder, our speculative decoding method achieves up to $1.32\times$ wall-clock speedup, while longer lookahead increases the number of emitted tokens per target batch to 2.745. We also investigate the recently discovered sampling-watermark trade-off frontier for the SpecPF decoder. Experimental results across multiple model pairs and datasets validate both the improved sampling efficiency and stronger watermark detectability of our method.

1 INTRODUCTION

Large language models are now cornerstones of modern generative AI, with autoregressive sampling as a fundamental mechanism and strong capabilities across a wide range of tasks (Yang et al., 2025; Gemini Team, Google, 2025; Anthropic, 2026a; OpenAI, 2026). Standard decoding schemes include top-k sampling (Radford et al., 2019; Fan et al., 2018), nucleus sampling (Holtzman et al., 2020), locally typical sampling (Meister et al., 2023) and min-p sampling (Nguyen et al., 2025). Recently, a decoding method called the permute-and-flip (PF) decoder (Zhao et al., 2025), which is based on the permute-and-flip scheme (McKenna & Sheldon, 2020) from differential privacy, was proposed. It is Pareto-optimal in the stability–utility tradeoff, and more importantly, is inherently watermarkable. As contents generated by LLMs become difficult to distinguish from human-written content, their provenance become increasingly crucial (Dathathri et al., 2024; Sadasivan et al., 2025; Wu et al., 2025). Therefore, embedding traceable watermarks into generated content is becoming increasingly important, and may even become necessary; for example, Anthropic recently announced that future Claude models will generate text with embedded watermarks (Anthropic, 2026b).

Beyond provenance, the efficiency of autoregressive generation remains another key bottleneck in modern LLM deployment. It is crucial to generate watermarked content efficiently without compromising content quality. Speculative decoding has become an effective paradigm for accelerating autoregressive generation while preserving the target model's original distribution (Chen et al., 2023; Leviathan et al., 2023). However, speculative decoding and watermarking are not independent of each other. Recent studies (Hu & Huang, 2024; He et al., 2026; Liu et al., 2026a) have shown that integrating the two is inherently challenging and there exists a fundamental trade-off between them.

Our contributions. In this paper, we design an exact speculative decoding pipeline for the PF decoder, thereby preserving its target distribution and inheriting all of its desirable properties, including natural watermarkability, while providing up to $1.32\times$ wall-clock acceleration. Our construction is also drafter-GLS: the draft model affects only speculative efficiency and not the realized

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.

target output. We further develop a model-agnostic watermark detection scheme that improves upon the original PF detector, and rigorously analyze and compare different PF-specific watermark detection scores. Finally, we investigate the sampling-watermark tradeoff for SpecPF and characterize its efficiency-detectability operating points.

2 Related work

Speculative Decoding. Speculative decoding accelerates autoregressive generation without changing the target distribution (Sun et al., 2023; Chen et al., 2023; Leviathan et al., 2023). Instead of sampling every token directly from a large target model, a smaller draft model proposes candidate continuations, which are verified by the target model in parallel and then either accepted or corrected. Recent works further extend this paradigm through multi-draft decoding (Rowan et al., 2025; Khisti et al., 2025; Hu et al., 2025), online learning (Liu et al., 2023; Qian et al., 2026), and tree-based inference and verification (Miao et al., 2024). Alternative couplings based on shared randomness have also been studied, including exponential races (Kobus & Gündüz, 2025) and list-level distribution coupling (Rowan et al., 2025), with the latter additionally establishing a degree of drafter invariance. These works primarily consider speculative decoding for the standard target distribution. In contrast, our target decoder is the PF decoder.

LLM Watermarking. As LLM-generated text becomes increasingly fluent, it is becoming more difficult to distinguish from human-written content, making output watermarking important and potentially necessary (Anthropic, 2026b). Early methods typically modify token probabilities, for example by biasing sampling toward key-dependent token subsets (Kirchenbauer et al., 2023). In contrast, unbiased or distortion-free watermarking schemes use pseudorandomness (Christ & Gunn, 2024) to correlate generated tokens with a watermark while preserving the original target distribution (Aaronson & Kirchner, 2023; Kuditipudi et al., 2024). SynthID (Dathathri et al., 2024) further developed a scalable generative watermark that can be combined with speculative decoding. Beyond watermark construction, recent works have studied watermark detection and score design more systematically. In particular, Li et al. (2025) developed a general statistical framework based on pivots and characterized efficient and optimal detection rules, while Lattimore (2026) proposed a refined detection score for Gumbel watermarking. For PF watermarking, Zhao et al. (2025) introduced a native detector based on the PF randomness; in this work, we compare this original PF score with PF analogues of the Li and Lattimore scores and develop our signature-filtered detector. Finally, Hu & Huang (2024) identified a fundamental tradeoff between speculative decoding efficiency and watermark detectability, which was further studied by He et al. (2026); Liu et al. (2026a).

Permute-Flip, Differential Privacy and Report-Noisy-Max. The permute-and-flip mechanism was originally introduced by McKenna & Sheldon (2020) for differentially private selection, where its expected utility is always at least that of the exponential mechanism and can improve the expected suboptimality by up to a factor of two. Ding et al. (2021) subsequently showed that permute-and-flip is distributionally identical to report-noisy-max with independent exponential noise. This equivalence gives the particularly simple representation in which independent exponential random variables are added to the candidate scores and the largest perturbed score is selected. Zhao et al. (2025) brought this mechanism to LLM decoding, establishing its stability–utility properties and Pareto-optimality, and further showed that the exponential randomness naturally induces a keyed, distribution-preserving watermark analogous to the Gumbel watermark. Recently this scheme has been used for LLM alignment (Liu et al., 2026b).

3 Speculative Permute-Flip Decoder

3.1 Vanilla Permute-Flip Decoder

We first briefly review the constructions and properties of the permute-and-flip (PF) decoder of Zhao et al. (2025), which adapts the permute-and-flip mechanism originally developed for differentially private selection (McKenna & Sheldon, 2020). At an autoregressive context c, let $u_c(y)$ denote the processed target logit of token y and let $\mathcal{S}_P(c) \subseteq \mathcal{V}$ be the resulting support. For temperature $T > 0$, define $u_c^\star \triangleq \max_{y \in \mathcal{S}_P(c)} u_c(y)$. As shown in Algorithm 1, PF decoder randomly permutes

the candidate tokens and visits them in this order, accepting token y with probability

$$a_c^P(y) \triangleq \exp\left(\frac{u_c(y) - u_c^\star}{T}\right). \tag{1}$$

Algorithm 1: Permute-and-flip form.

Given: context c, processed logits u_c, support $\mathcal{S}_P(c)$, temperature T

$u_c^\star \leftarrow \max_{y \in \mathcal{S}_P(c)} u_c(y)$
sample uniformly random permutation π of $\mathcal{S}_P(c)$
foreach y *in the order* π **do**
  Draw $Z \sim \text{Bernoulli}\left(\exp\left(\frac{u_c(y)-u_c^\star}{T}\right)\right)$
  if $Z = 1$ **then**
    return y

Algorithm 2: Report-noisy-max form.

Given: context c, processed logits u_c, support $\mathcal{S}_P(c)$, temperature T

foreach $y \in \mathcal{S}_P(c)$ **do**
  Draw $E_{\Lambda(c)}(y) \sim \text{Exp}(1)$
return

$$\arg\max_{y \in \mathcal{S}_P(c)} \left\{\frac{u_c(y)}{T} + E_{\Lambda(c)}(y)\right\}$$

Interestingly, the PF decoder admits an equivalent report-noisy-max form with exponential noise, which is shown in Algorithm 2. The two forms have exactly the same distribution (Ding et al., 2021; Zhao et al., 2025), i.e., with $E_{\Lambda(c)}(y) \overset{\text{i.i.d.}}{\sim} \text{Exp}(1)$, PF decoder admits the following representation

$$Y(c) = \arg\max_{y \in \mathcal{S}_P(c)} \left\{\frac{u_c(y)}{T} + E_{\Lambda(c)}(y)\right\}. \tag{2}$$

This report-noisy-max representation naturally makes the decoder watermarkable (Zhao et al., 2025), and will be particularly useful for our speculative construction later. Moreover, PF inherits several desirable properties from its use for differential-privacy (McKenna & Sheldon, 2020):

- **Stability.** On a fixed support, PF with temperature T is $(2/T)$-stable, the same stability guarantee as softmax sampling at temperature T.
- **Utility.** PF satisfies
$$\mathbb{E}[u_c(Y(c))] \geq u_c^\star - T \log |\mathcal{S}_P(c)|, \tag{3}$$
and, at the same temperature, its expected logit utility is never worse than that of softmax sampling. Gor some logits, PF achieves a factor-2 smaller expected suboptimality than softmax sampling.
- **Pareto optimality.** PF is Pareto-optimal for the stability–utility tradeoff: no decoder with the same $(2/T)$-stability can uniformly improve its expected utility over all logits.

PF watermarking. The report-noisy-max form (2) gives PF a natural distortion-free watermark (Zhao et al., 2025), using the idea similar to tbhe celebrated Gumbel watermark Aaronson & Kirchner (2023). Since an exponential random variable can be written as $E = -\log R$ for $R \sim \text{Unif}(0, 1)$, let an ideal random field assign

$$R_{\Lambda(c)}(y) \overset{\text{i.i.d.}}{\sim} \text{Unif}(0, 1), \qquad E_{\Lambda(c)}(y) = -\log R_{\Lambda(c)}(y). \tag{4}$$

The keyed watermark replaces these uniforms by pseudorandom values

$$R_{\Lambda(c)}(y) = \text{PRF}_{\mathsf{k}}(\Lambda(c), y)\,, \qquad E_{\Lambda(c)}(y) = -\log R_{\Lambda(c)}(y), \tag{5}$$

where k is the secret key and $\Lambda(c)$ is a deterministic context label. The next token is then still selected according to (2). Under the standard PRF assumption, the keyed decoder is computationally indistinguishable from ordinary randomized PF decoding, while the key makes the underlying randomness reproducible for verification.

Given an emitted token w_t at context c_t, the detector reconstructs the corresponding PF pivot $V_t \triangleq R_{\Lambda(c_t)}(w_t)$ and the original PF detector uses the Fisher statistic

$$S_{\text{PF}} = \sum_{t=1}^{M} -\log V_t. \tag{6}$$

Under the null hypothesis that the text is independent of the secret key, and assuming the scored context–token labels are distinct, $V_t \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}(0,1)$ and $S_{\mathrm{PF}} \sim \Gamma(M,1)$, so detection can be performed using the corresponding upper-tail Gamma p-value (Zhao et al., 2025).

This detector requires only the generated text, the context-labeling rule, and the secret key, without target-model replay. In the following, we exploit the same shared exponential random field to accelerate PF decoding while leaving its target law and native watermark unchanged.

3.2 Speculative Permute-Flip Decoder

We now introduce SpecPF, which uses a lightweight draft model to predict the realized PF random map defined in (2). Let Q denote the draft model. At context c, let $v_c(y)$ denote its processed logit for token y and $\mathcal{S}_Q(c) \subseteq \mathcal{V}$ its resulting support. Crucially, the target and draft use the same exponential random field $E_{\Lambda(c)}(y)$. We define the shared-noise draft score

$$S_c^Q(y) \triangleq \frac{v_c(y)}{T_q} + E_{\Lambda(c)}(y), \qquad y \in \mathcal{S}_Q(c), \tag{7}$$

where T_q is the draft temperature. At a block root h, the draft selects the B highest-scoring distinct tokens,

$$D_B(h) \triangleq \mathrm{TopB}_{y \in \mathcal{S}_Q(h)}\, S_h^Q(y), \tag{8}$$

with ties resolved by a fixed deterministic order. Thus, $D_B(h)$ is not formed by B independent samples; rather, it consists of the top-B ranks of one shared-noise PF realization.

The basic idea is simple. The target-side logits and random field already determine a unique PF winner $Y(c)$ at every context through (2). The draft model does not participate in determining this output token; it only tries to predict which target-PF path will be realized. At the root, we keep up to B alternatives in (8). Each alternative is then extended for up to L steps by following the rank-one draft PF prediction

$$\widetilde{Y}(c) \triangleq \arg\max_{y \in \mathcal{S}_Q(c)} S_c^Q(y). \tag{9}$$

This yields a bounded-width speculative tree with at most B leaves, rather than branching by a factor of B at every depth.

See Algorithm 3 for a detailed construction. Importantly, the draft model is used only to construct the speculative tree. At every reached context c, the emitted token is always the target PF winner $Y(c)$ defined in (2). If the corresponding edge is present in $\mathcal{T}_h$, the next target context has already been evaluated and decoding continues within the same block; otherwise, the current speculative block terminates after emitting the correct target token. Hence, speculation changes only how many target-PF tokens can be obtained from one batched verification, rather than the target decoding rule itself. For $L = 1$, Algorithm 3 reduces to the shallow width-B version: the draft proposes B root candidates and a successful match provides one additional precomputed target PF token.

Inherited PF properties. On a fixed full support, PF with temperature T is $(2/T)$-stable, is never worse than softmax sampling in expected logit utility, can attain a factor-two smaller expected suboptimality, and is Pareto-optimal for the associated stability–utility tradeoff (McKenna & Sheldon, 2020; Zhao et al., 2025). These are properties of the PF output law, not of the speculative executor. Dynamic hard top-k can change support under an arbitrarily small perturbation, so the full-support stability theorem must not be applied across support changes; we evaluate fixed-support stability and operational support flips separately.

4 Theoretic Guarantees

We prove the exactness of our decoding method for an arbitrary choice of lookahead; hence, our speculative PF decoder directly inherits all the desirable properties of the vanilla PF decoder, as discussed in Section 3.1.

Theorem 1 (Drafter invariance)**.** *For fixed target-side randomness and logits, changing Q, T_q, K_q, B, or the candidate-selection policy does change the fixed-length output sequence. It can change the hit events, block boundaries, target batch shapes, cache trace, and wall-clock cost.*

Algorithm 3: Bounded-width longer-lookahead SpecPF. The draft keeps up to B alternatives at the root and follows one shared-noise PF continuation from each branch. The target then traverses the precomputed tree using its own PF map.

Given: target P, draft Q, output length N, prompt $w_{1:n}$, root width B, lookahead $L \geq 1$, target and draft processors, random field E, and deterministic tie-breaking

while $n < N$ **do**
- $h \leftarrow w_{1:n}$; $\ell \leftarrow \min\{L, N-n\}$; initialize prefix tree $\mathcal{T}_h$ with root h
- compute processed draft logits v_h and $\mathcal{S}_Q(h)$
- $D_B(h) \leftarrow \mathrm{TopB}_{y \in \mathcal{S}_Q(h)} \left\{ \frac{v_h(y)}{T_q} + E_{\Lambda(h)}(y) \right\}$
- **foreach** $y \in D_B(h)$ **do**
  - add edge $h \to h \,\|\, y$ to $\mathcal{T}_h$
  - $c \leftarrow h \,\|\, y$
  - **for** $d \leftarrow 2$ **to** ℓ **do**
    - compute processed draft logits v_c and $\mathcal{S}_Q(c)$
    - $\widetilde{y} \leftarrow \arg\max_{z \in \mathcal{S}_Q(c)} \left\{ \frac{v_c(z)}{T_q} + E_{\Lambda(c)}(z) \right\}$
    - add edge $c \to c \,\|\, \widetilde{y}$ to $\mathcal{T}_h$
    - $c \leftarrow c \,\|\, \widetilde{y}$
- evaluate all leaf paths of $\mathcal{T}_h$ in one target batch, retaining processed target logits u_c and $\mathcal{S}_P(c)$ for every represented context c
- $c \leftarrow h$
- **while** $n < N$ **do**
  - $y \leftarrow \arg\max_{z \in \mathcal{S}_P(c)} \left\{ \frac{u_c(z)}{T} + E_{\Lambda(c)}(z) \right\}$
  - emit y; append y to the realized prefix; $n \leftarrow n+1$
  - **if** y *is EOS* **then**
    - **return**
  - **if** $n = N$ *or* $c \to c \,\|\, y \notin \mathcal{T}_h$ **then**
    - **break**
  - $c \leftarrow c \,\|\, y$
- retain the cache prefix compatible with the realized path; discard unchosen branches

The draft acts only as an accelerator for a target-side random map. This is also different from conventional rejection-based speculative decoding, where the draft distribution enters both the acceptance probability and the correction law. See (Rowan et al., 2025; Daliri et al., 2025) for related drafter-GLS constructions. It has been pointed out by Liu et al. (2026a) that drafter invariance becomes important for watermarking, especially when considering the tradeoff between sampling efficiency and watermark strength; see also (Hu & Huang, 2024) the way to maintain watermark strength, which shares a similar spirit.

Theorem 2 (Exactness of SpecPF)**.** *Fix the target model P, its processed logits u_c, supports $\mathcal{S}_P(c)$, temperature T, the context labeler Λ, and deterministic tie-breaking at every context c. For any draft model Q, root width $B \geq 1$, and lookahead $L \geq 1$, let W^{PF} denote the continuation obtained by repeatedly applying the target PF map*

$$Y(c) = \arg \max_{y \in \mathcal{S}_P(c)} \left\{ \frac{u_c(y)}{T} + E_{\Lambda(c)}(y) \right\}, \tag{10}$$

and let W^{SpecPF} denote the continuation by Algorithm 3 using the same random field $\{E_{\Lambda(c)}(y)\}$. Then, for every realization of the random field,

$$W^{\mathrm{SpecPF}} = W^{\mathrm{PF}}. \tag{11}$$

Hence, Algorithm 3 has the same output distribution as the vanilla PF decoder, for any B and L.

5 Signature-Filtered PF Watermark

We now study the detection side of the PF watermark. Recall from Section 3.1 that, for an emitted token w_t at context c_t, the detector reconstructs the PF pivot $V_t \triangleq R_{\Lambda(c_t)}(w_t)$ and the

original PF detector of Zhao et al. (2025) uses the per-token score $-\log V_t$ and aggregates it $S_{\mathrm{PF}} = \sum_{t=1}^{M} -\log V_t$. Under the null hypothesis H_0, when the candidate text is independent of the secret key and the scored context–token labels are distinct, we have $V_t \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}(0,1)$ and $-\log V_t \overset{\text{i.i.d.}}{\sim} \mathrm{Exp}(1)$, and hence $S_{\mathrm{PF}} \sim \mathrm{Gamma}(M,1)$.

5.1 Optimality of the PF Watermark

We first show that the original PF score is already optimal at the single-token level, which motivates us to improve watermark detection from another dimension, as will be elaborated later.

Fix a context c and write

$$a_c^P(y) = \exp\left(\frac{u_c(y) - u_c^\star}{T}\right), \qquad y \in \mathcal{S}_P(c), \tag{12}$$

as in (1). Using $E_{\Lambda(c)}(y) = -\log R_{\Lambda(c)}(y)$, the PF rule (2) can equivalently be written as

$$Y(c) = \arg\min_{y \in \mathcal{S}_P(c)} \frac{R_{\Lambda(c)}(y)}{a_c^P(y)}. \tag{13}$$

Theorem 3 (Single-token Optimality)**.** *Fix a context c and its processed target logits. Let $R_{\Lambda(c)}(y) \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}(0,1)$ and $V = R_{\Lambda(c)}(Y(c))$ where $Y(c)$ is generated by* (13)*. Under the unwatermarked null, $V \sim \mathrm{Unif}(0,1)$. Under the PF-watermarked alternative, V has density*

$$g_c(v) = \sum_{y \in \mathcal{S}_P(c)} \prod_{\substack{z \in \mathcal{S}_P(c) \\ z \neq y}} \left[1 - v\frac{a_c^P(z)}{a_c^P(y)}\right]_+, \qquad 0 < v < 1, \tag{14}$$

where $[x]_+ \triangleq \max\{x, 0\}$. Moreover, $g_c(v)$ is non-increasing in v. Consequently, for every significance level $\alpha \in (0,1)$, $\phi_\alpha(V) = \mathbf{1}\{V \le \alpha\}$ is uniformly most powerful among all level-α tests based only on the single recovered pivot V, simultaneously for every fixed target-logit vector. Thus the original PF score $-\log V$ is single-token optimal*.*

The proof is in Appendix A.2. Theorem 3 shows that changing the single-token function of V_t cannot uniformly improve PF detection.

Nevertheless, it does not mean that all token positions carry equally strong watermark evidence. Indeed, the alternative density (14) depends on the local weight vector $\{a_c^P(y)\}$. When one token dominates and the remaining weights approach zero, $g_c(v)$ approaches the uniform density and the watermark signal becomes weak; when several tokens are competitive, selection biases the winning pivot more strongly toward zero. Thus the PF watermark naturally exhibits heterogeneous detection strength across contexts.

5.2 Signature-filtered PF detection

The above discussion motivates improving detection through *position selection* rather than replacing the native PF score. Recent work by Hong et al. (2026) introduced *signature filtering*, a detection-time method that learns observable token signatures associated with unreliable watermark evidence and removes such positions before applying the original watermark test. This method is model-agnostic at detection time and was particularly motivated by weak-signal and low-entropy regimes.

This principle is especially well matched to PF. First, Theorem 3 showed that the native score $-\log V_t$ is already optimal for a single pivot. Second, PF signal strength varies with the local competition among logits, but directly estimating this competition would require target-model replay. Observable n-gram signatures provide a simple proxy for these recurring local contexts. Finally and importantly, because the PF null pivots are exactly uniform, filtering positions using only observable text preserves the exact Gamma null distribution of the retained Fisher statistic.

We therefore adapt signature filtering to the PF score. Rather than changing the embedding mechanism or learning a new pivot score, we learn which observable bigram signatures tend to contribute weak PF evidence and exclude them before applying the exact detection test.

Algorithm 4: Signature-filtered PF detection.

Given: candidate continuation $w_{1:M}$, initial prompt or saved context labels, tokenizer, secret key k, frozen drop table $\mathcal{D}$, and $M_{\min} = 30$

reconstruct $V_t = R_{\Lambda(c_t)}(w_t)$ for every scored position t
form bigram signatures s_t by (15)
$A \leftarrow \{t : s_t \notin \mathcal{D}\}$
if $|A| < M_{\min}$ **then**
  $A \leftarrow \{1, \dots, M\}$
$S_A \leftarrow \sum_{t \in A} -\log V_t$
$p_A \leftarrow \mathbb{P}\{G \geq S_A\}$ for $G \sim \mathrm{Gamma}(|A|, 1)$
return p_A and $-\log(p_A)/M$

At position t, define the bigram signature

$$s_t = \begin{cases} (\mathrm{BOS}, w_1), & t = 1, \\ (w_{t-1}, w_t), & t \geq 2. \end{cases} \tag{15}$$

We use five-fold cross-fitting. For a held-out fold f, let $\mathcal{I}_{-f}$ denote all scored training positions in the other folds. For each signature s, define $n_{s,-f} = |\{i \in \mathcal{I}_{-f} : s_i = s\}|$ and the shrunken mean PF contribution

$$\widehat{\mu}_{s,-f} = \frac{\sum_{i \in \mathcal{I}_{-f} : s_i = s} (-\log V_i) + \lambda_0}{n_{s,-f} + \lambda_0} \tag{16}$$

for $\lambda_0 = 50$. The shrinkage target is the exact null mean $\mathbb{E}_0[-\log V] = 1$. Only signatures satisfying $n_{s,-f} \geq 30$ are eligible for filtering.

We sort the eligible signatures by increasing $\widehat{\mu}_{s,-f}$, so signatures with the weakest empirical PF evidence are considered first for removal. For any candidate drop set $\mathcal{D}$, let $A^{\mathrm{train}}_{-f}(\mathcal{D}) = \{i \in \mathcal{I}_{-f} : s_i \notin \mathcal{D}\}$, we use the training-only criterion

$$Z_{-f}(\mathcal{D}) = \frac{\sum_{i \in A^{\mathrm{train}}_{-f}(\mathcal{D})} \big(-\log V_i - 1\big)}{\sqrt{|A^{\mathrm{train}}_{-f}(\mathcal{D})|}}. \tag{17}$$

Among prefixes of the ordered signature list, we choose the one maximizing $Z_{-f}(\mathcal{D})$, subject to retaining at least 40% of the training positions and dropping at most 60%. The resulting table $\mathcal{D}_{-f}$ is then frozen before evaluating fold f. For a held-out sequence $w_{1:M}$, define $A_{-f}(w_{1:M} = \{t : s_t \notin \mathcal{D}_{-f}\}$. If $|A_{-f}| < M_{\min} = 30$, we fall back to $A_{-f} = \{1, \dots, M\}$. The signature-filtered PF statistic is

$$S_{-f} = \sum_{t \in A_{-f}(w_{1:M})} -\log V_t, \quad \text{and} \quad p_{-f} = \mathbb{P}\{G \geq S_{-f}\}, \tag{18}$$

where $G \sim \mathrm{Gamma}(|A_{-f}|, 1)$. For comparisons across a fixed original evidence budget M, we report $operatornameANLPPT = -\frac{1}{M} \log p_{-f}$ as the standard choice. See Algorithm 4.

6 Experiments

6.1 Setup

Models, data and decoding. We use two target–draft pairs, Qwen2.5-14B-Instruct with Qwen2.5-0.5B-Instruct (Qwen Team, 2024) and Vicuna-7B-v1.5 with Vicuna-68M (Zheng et al., 2023), and the first 1000 eligible prompts of CNN/DailyMail (Nallapati et al., 2016), ELI5 (Fan et al., 2019) and GSM8K (Cobbe et al., 2021) (six settings); the detection study and one speed comparison add Qwen2.5-7B-Instruct with the same draft (nine settings); each prompt has its own key. The target uses $T = 1$, top-$k = 50$ and at most 128 new tokens in float16, with batch size one on an NVIDIA RTX A6000. SpecPF uses $B = 3$ (widths 2–4 give similar token rates; Table 19), draft temperature $T_q = 0.5$, draft top-k $K_q = 100$, and $L = 1$ unless stated otherwise (Appendix C).

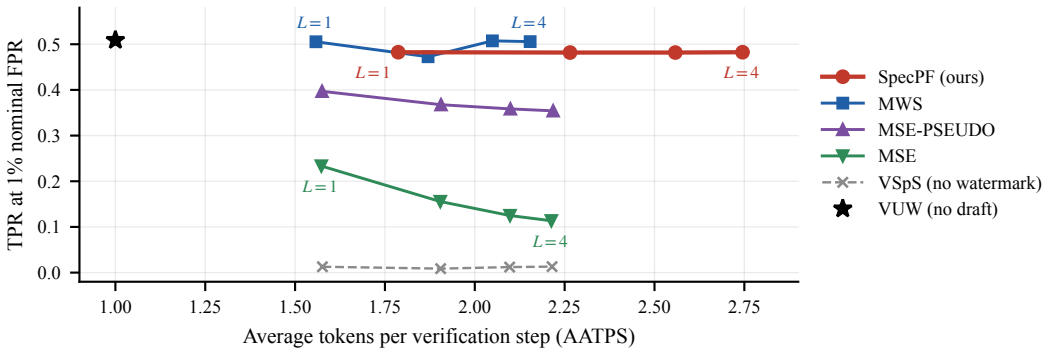
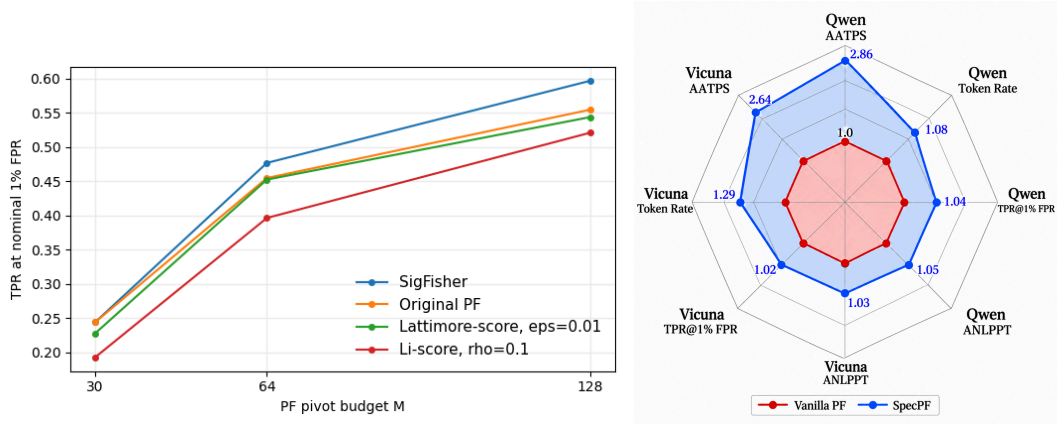


Figure 1: **Efficiency and detection together: at every lookahead, SpecPF has the highest AATPS, at a detection rate close to that of MWS.** TPR at 1% nominal FPR with $M = 64$ scored tokens against AATPS, pooled over the six Qwen2.5-14B and Vicuna-7B settings; each line joins lookaheads $L = 1, \ldots, 4$ from left to right. SpecPF is detected with the signature-filtered PF test and the others with the U score; for VSpS (no watermark) the vertical axis is the empirical FPR, and VUW (no draft) is a single point. SpecPF samples from the PF distribution and checks a tree of width $B = 3$; the others sample from the softmax distribution with one draft sequence (Table 7).



(a) SigFisher improves watermark.

(b) Improvement.

Figure 2: PF watermark detection and overall SpecPF gains. (a) TPR at nominal 1% FPR versus PF pivot budget M for SigFisher and PF-score baselines. (b) SpecPF relative to vanilla PF in AATPS, token rate, TPR@1% FPR, and ANLPPT on Qwen and Vicuna, with vanilla PF normalized to 1.

Baselines. For speed, the baseline is PF decoding without a draft (vanilla PF), timed in the same run as SpecPF. For efficiency and detection, we compare with the watermarked speculative sampling methods Maintain Watermark Strength (MWS) and Maintain Sampling Efficiency (MSE) of Hu & Huang (2024), MSE with the pseudorandom acceptance of He et al. (2026) (MSE-PSEUDO), Vanilla Unbiased Watermark (VUW, no draft), and Vanilla Speculative Sampling (VSpS, no watermark). They use the same drafts and lookaheads with one draft sequence per step (SpecPF: a tree of width $B = 3$), and are detected with the likelihood-agnostic U score of Hu & Huang (2024); we do not evaluate the detectors of He et al. (2026) that also use the acceptance randomness. These methods watermark softmax sampling and SpecPF watermarks PF, so each is evaluated under its own output distribution. Appendix D also compares SpecPF with GLS (Rowan et al., 2025).

Metrics. Following Hu & Huang (2024), we measure efficiency by the Average Accepted Tokens Per Step (AATPS). The average number of tokens emitted per verification step, including the target's own token at the first mismatch. We also report the token rate (TR, tokens per second) and the speedup over vanilla PF in the same run; we compare methods by AATPS, which does not depend on the implementation, and report the wall-clock speed of SpecPF by TR. Following He et al. (2026),

Table 1: Speed of SpecPF ($L = 1$, $B = 3$) and vanilla PF, measured in the same run with 1000 prompts per setting. TR: token rate (tokens/s); speedup: ratio of the two token rates; AATPS: tokens per verification step; memory: peak GPU memory, vanilla PF / SpecPF.

Target / draft	Dataset	Vanilla PF TR	SpecPF TR	Speedup	AATPS	Memory (GiB)
Qwen2.5-14B / 0.5B	CNN/DM	19.577	20.287	1.036×	1.826	28.73 / 29.41
	ELI5	19.569	20.717	1.059×	1.792	28.49 / 28.73
	GSM8K	16.313	18.636	1.142×	1.968	28.54 / 28.82
Vicuna-7B / 68M	CNN/DM	33.289	42.856	1.287×	1.762	13.03 / 14.71
	ELI5	34.378	43.809	1.274×	1.660	12.79 / 13.30
	GSM8K	36.474	48.162	1.320×	1.736	12.85 / 13.65

Table 2: Detection of PF watermarks over nine settings: TPR at 1% nominal FPR with the first M scored tokens, and empirical FPR and retained positions at $M = 64$. Scores after Zhao et al. (2025), Lattimore (2026) and Li et al. (2025) (Appendix B); all variants in Table 8.

Detector	TPR@30	TPR@64	TPR@128	FPR@64	Retained@64
Original PF score	0.244	0.455	0.554	0.009	1.000
PF power-law, $\epsilon = 0.01$	0.228	0.452	0.544	0.015	1.000
PF likelihood ratio, $\rho = 0.1$	0.193	0.396	0.521	0.014	1.000
Signature-filtered PF (ours)	0.244	0.481	0.600	0.013	0.834

we measure detection by the true positive rate (TPR) at a nominal false positive rate (FPR) of 1%, on the first $M \in \{30, 64, 128\}$ scored tokens of responses with at least M tokens. Thresholds are calibrated on half of the unwatermarked VSpS responses; the rest give the empirical FPR.

6.2 Results

SpecPF has the highest AATPS at a comparable detection rate. In Figure 1, SpecPF has the highest AATPS at every lookahead (1.787–2.745), and its TPR (0.4818–0.4825, unchanged across L, consistent with Corollary 1) is close to that of MWS (0.47–0.51 at AATPS 1.56–2.15) and VUW (0.509, no draft). MSE and MSE-PSEUDO reach an AATPS similar to that of MWS, but their TPR is lower and falls as L grows (from 0.233 to 0.114 and from 0.397 to 0.354). At $L = 4$, SpecPF emits 27% more tokens per step than MWS (2.745 vs. 2.154) at a TPR lower by 0.023; the methods have different output distributions, and SpecPF checks more target contexts per step, so this is not a comparison at equal target cost. As a multi-draft reference, GLS (Rowan et al., 2025) with four drafts at $L = 1$ (no watermark) has a lower AATPS than SpecPF with three (1.59–1.95 vs. 1.66–1.97), and SpecPF is 1.04–1.09× faster (Qwen2.5-7B and Vicuna-7B; Table 15).

SpecPF accelerates PF decoding. Table 1 shows that SpecPF emits 1.66–1.97 tokens per verification step and is 1.04–1.32× faster than vanilla PF, with at most 1.7 GiB of additional memory. The speedup decreases as the draft grows relative to the target: 1.27–1.32× for Vicuna (68M draft, 7B target), 1.04–1.14× for Qwen2.5-14B (0.5B draft), and 0.87–0.95× for Qwen2.5-7B with the same draft, where the cost of the draft and the tree is not recovered (Table 17). Consistent with Theorem 2, the responses of SpecPF and vanilla PF have similar perplexity, logit regret, ROUGE-L and GSM8K exact match (Table 20) and similar detection rates at every M (Table 10).

Signature filtering improves PF detection. In Table 2, signature filtering raises the TPR of the original PF score from 0.455 to 0.481 at $M = 64$ and from 0.554 to 0.600 at $M = 128$, while keeping 83% of the positions at $M = 64$, at a slightly higher empirical FPR (0.013 vs. 0.009). The PF analogues of the scores of Lattimore (2026) and Li et al. (2025), which replace $-\log V_t$ by another function of the pivot, have lower TPR, consistent with Theorem 3.

Concludion. We introduced SpecPF, an exact speculative decoding method for the PF decoder, which accelerates generation while preserving the output distribution. We further showed that the PF score is single-token optimal, motivating a signature-filtered detector that improves watermark detection. Experiments demonstrate improvements across multiple models and datasets.

DISCLOSURE OF USING AI

Generative AI tools were used to assist with manuscript writing and revision, numerical checks against experimental results, and discussion of result interpretation. All AI-assisted content was reviewed by the authors, who take full responsibility for the accuracy and integrity of the final manuscript.

REFERENCES

Scott Aaronson and H Kirchner. Watermarking of large language models. In *Large Language Models and Transformers Workshop at Simons Institute for the Theory of Computing*, 2023.

Anthropic. Claude Opus 5 System Card. `https://www.anthropic.com/claude-opus-5-system-card`, July 2026a.

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

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.

Miranda Christ and Sam Gunn. Pseudorandom error-correcting codes. In *Annual International Cryptology Conference*, pp. 325–347. Springer, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Majid Daliri, Christopher Musco, and Ananda Theertha Suresh. Coupling without communication and drafter-invariant speculative decoding. In *2025 IEEE International Symposium on Information Theory (ISIT)*, pp. 1–6. IEEE, 2025.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, 2024.

Zeyu Ding, Daniel Kifer, Sayed M. Saghaian N. E., Thomas Steinke, Yuxin Wang, Yingtai Xiao, and Danfeng Zhang. The permute-and-flip mechanism is identical to report-noisy-max with exponential noise. *arXiv preprint arXiv:2105.07260*, 2021.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 889–898, 2018.

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.

Gemini Team, Google. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. Technical report, Google DeepMind, 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 (ICLR)*, 2026.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*, 2020.

Chih-Duo Hong, Yen-Pang Chen, and Fang Yu. Signature filtering: A lightweight enhancement for statistical watermark detection in large language models. *arXiv preprint arXiv:2606.18430*, 2026.

Zhengmian Hu and Heng Huang. Inevitable trade-off between watermark strength and speculative sampling efficiency for language models. *Advances in Neural Information Processing Systems*, 37:55370–55402, 2024.

Zhengmian Hu, Tong Zheng, Vignesh Viswanathan, Ziyi Chen, Ryan A. Rossi, Yihan Wu, Dinesh Manocha, and Heng Huang. Towards optimal multi-draft speculative decoding. In *International Conference on Learning Representations (ICLR)*, 2025.

Ashish Khisti, M Reza Ebrahimi, Hassan Dbouk, Arash Behboodi, Roland Memisevic, and Christos Louizos. Multi-draft speculative sampling: Canonical decomposition and theoretical limits. In *International Conference on Learning Representations (ICLR)*, 2025.

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

Szymon Kobus and Deniz Gündüz. Speculative sampling via exponential races. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18189–18204, 2025.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2307.15593.

Tor Lattimore. Refined detection for gumbel watermarking. *arXiv preprint arXiv:2603.30017*, 2026.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, 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.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Online speculative decoding. *arXiv preprint arXiv:2310.07177*, 2023.

Yanxiao Liu, Sicheng Wan, Zhan Gao, and Deniz Gündüz. Watermarkable multi-draft speculative sampling via poisson processes. *arXiv preprint arXiv:2609.21858*, 2026a.

Yanxiao Liu, Sicheng Wan, and Deniz Gündüz. Expbon: Exponential-noise best-of-n for efficient test-time LLM alignment. *arXiv preprint arXiv:2609.21899*, 2026b.

Ryan McKenna and Daniel R. Sheldon. Permute-and-flip: A new mechanism for differentially private selection. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/01e00f2f4bfcbb7505cb641066f2859b-Abstract.html.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121, 2023.

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.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 280–290, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1028.

Minh Nguyen, Andrew Baker, Clement Neo, Allen Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent LLM outputs. In *International Conference on Learning Representations*, volume 2025, pp. 70333–70366, 2025.

OpenAI. GPT-6 Astra System Card. https://deploymentsafety.openai.com/gpt-6-astra, September 2026. Published September 3, 2026.

Yu-Yang Qian, Hao-Cong Wu, Yichao Fu, Hao Zhang, and Peng Zhao. When drafts evolve: Speculative decoding meets online learning. *arXiv preprint arXiv:2603.12617*, 2026.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Joseph Rowan, Buu Phan, and Ashish Khisti. List-level distribution coupling with applications to speculative decoding and lossy compression. *Advances in Neural Information Processing Systems*, 38, 2025.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, Soheil Feizi, et al. Can ai-generated text be reliably detected? stress testing ai text detectors under various attacks. *Transactions on Machine Learning Research*, 1, 2025.

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.

Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Lidia Sam Chao, and Derek Fai Wong. A survey on llm-generated text detection: Necessity, methods, and future directions. *Computational Linguistics*, 51(1):275–338, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Xuandong Zhao, Lei Li, and Yu-Xiang Wang. Permute-and-flip: An optimally stable and watermarkable decoder for LLMs. In *International Conference on Learning Representations, (ICLR)*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

A Theoretical Details

A.1 Proof of Theorem 2

Proof. We prove the stronger pathwise statement (11). Suppose that direct PF and Algorithm 3 have generated the same prefix c. By construction, whenever Algorithm 3 reaches c, it emits

$$y = \arg\max_{z \in \mathcal{S}_P(c)} \left\{ \frac{u_c(z)}{T} + E_{\Lambda(c)}(z) \right\} = Y(c). \tag{19}$$

This is exactly the next token emitted by direct PF after the same prefix c.

The speculative tree $\mathcal{T}_h$ affects only whether decoding can continue within the current target batch. If the edge $c \to c \,\|\, Y(c)$ is present in $\mathcal{T}_h$, then the target logits at the next context $c \,\|\, Y(c)$ have already been evaluated, and Algorithm 3 continues from that context. If the edge is absent, the current speculative block terminates after emitting $Y(c)$; the next block starts from the same updated prefix $c \,\|\, Y(c)$. In either case, the next emitted token is unchanged.

Therefore, starting from the common prompt, the two decoders emit the same token at every step. Induction over the emitted tokens proves $W^{\mathrm{SpecPF}} = W^{\mathrm{PF}}$ up to EOS or the output-length limit.

Finally, under the ideal random field $E_{\Lambda(c)}(y) \overset{\text{i.i.d.}}{\sim} \mathrm{Exp}(1)$, the sequence W^{PF} has exactly the vanilla PF distribution. Since Algorithm 3 is pathwise identical to it under the same field, it has exactly the same output distribution. □

A.2 Proof of Theorem 3

Proof. For any $y \in \mathcal{S}_P(c)$ and $v \in (0,1)$, the event that y is selected with pivot in $[v, v+dv]$ requires $R_{\Lambda(c)}(y) \in [v, v+dv]$ and

$$\frac{R_{\Lambda(c)}(z)}{a_c^P(z)} > \frac{v}{a_c^P(y)} \qquad \text{for every } z \neq y.$$

Since the random variables $R_{\Lambda(c)}(z)$ are independent uniforms,

$$\mathbb{P}\{Y(c) = y, V \in dv\} = \prod_{z \neq y} \left[1 - v\frac{a_c^P(z)}{a_c^P(y)}\right]_+ dv. \tag{20}$$

Summing over y gives (14). Every factor in (14) is non-increasing in v, and therefore so is $g_c(v)$.

Under H_0, the density of V is one, so $g_c(v)$ is exactly the likelihood ratio between the PF-watermarked alternative and the null. By the Neyman–Pearson lemma, the most powerful level-α test rejects for the largest values of $g_c(V)$. Since g_c is non-increasing, this is the lower-tail region $V \leq \alpha$, equivalently $-\log V \geq -\log \alpha$. The same rejection region applies to every fixed target-logit vector, which proves the uniform statement. □

B PF detector baselines

Let $V \in (0,1)$ be a recovered PF pivot. Every detector below is calibrated on a held-out unwatermarked null. The formulas explain the score; the fixed-FPR calibration makes their empirical powers comparable.

B.1 Original PF/Fisher score

The original score is $h_{\text{orig}}(v) = -\log v$. For M distinct ideal-null labels,

$$\sum_{t=1}^{M} h_{\text{orig}}(V_t) \sim \text{Gamma}(M, 1). \tag{21}$$

This exact p-value is also the base test applied after SigFisher filtering.

B.2 Li-style binary-PF likelihood-ratio score

For the binary normalized PF weight vector $(1, \rho)$, the selected-pivot density relative to the uniform null is

$$f_\rho(v) = 1 - \rho v + \left(1 - \frac{v}{\rho}\right) \mathbf{1}\{v \leq \rho\}. \tag{22}$$

The exact binary log-likelihood-ratio score is

$$h_\rho(v) = \log f_\rho(v). \tag{23}$$

It is a structured benchmark when a common reference ρ is plausible, but one fixed value is not naturally adapted to heterogeneous prompt- and position-dependent gaps. We report the prespecified grid

$$\rho \in \{0.05, 0.10, 0.20, 0.50\}. \tag{24}$$

B.3 Lattimore-style PF power-law score

The Gumbel detector of Lattimore (2026) emphasizes the watermark-favored boundary of its pivot. Since PF selects unusually small uniforms, the corresponding PF analogue is singular near zero:

$$h_\epsilon^{\text{PL}}(v) = \min\{v^{-1/2}, \epsilon^{-1/2}\} - (2 - \sqrt{\epsilon}). \tag{25}$$

For $U \sim \text{Unif}(0,1)$, $\mathbb{E}[h_\epsilon^{\text{PL}}(U)] = 0$. We report

$$\epsilon \in \{0.01, 0.02, 0.05, 0.10\}. \tag{26}$$

This is a PF-specific adaptation of the Lattimore principle, not a claim that Lattimore (2026) directly analyzed PF decoding.

B.4 Internal text-filter ablations

We additionally report: (i) retaining token types that occur at most once in a candidate continuation; (ii) retaining token types that occur at most twice; (iii) a five-fold cross-fitted unigram signature filter; and (iv) the selected five-fold cross-fitted bigram filter. The same minimum retained length and observable fallback are used for all filtered Fisher variants.

B.5 Confirmatory selection rule

The SigFisher hyperparameters are frozen before the final nine-cell comparison. Table 8 reports every detector, not only the best row. We report the selected detector alongside every prespecified comparator and the realized held-out FPR. A numerical ranking at a nominal operating point is not a proof of dominance at an exactly matched realized FPR or of global statistical optimality.

C Detailed experimental setup

C.1 Checkpoint and data versions

Table 3: Model checkpoints used in the formal experiments. The hashes are the local Hugging Face snapshot identifiers recorded by the benchmark configs.

Role	Checkpoint	Snapshot prefix
Qwen target	Qwen2.5-7B-Instruct	`a09a35458c70`
Qwen target	Qwen2.5-14B-Instruct	`cf98f3b3bbb4`
Qwen draft	Qwen2.5-0.5B-Instruct	`7ae557604adf`
Vicuna target	lmsys/vicuna-7b-v1.5	`3321f76e3f52`
Vicuna draft	double7/vicuna-68m	`f35c45e54830`

CNN/DailyMail uses configuration 3.0.0 and the test Arrow split. ELI5 uses the cached `sentence-transformers/eli5` pair configuration; the benchmark reads the first nonempty question/title field after filtering. GSM8K uses the `main` test Arrow split. Every dataset is deterministically filtered for a nonempty required field and then truncated to the first n eligible examples.

C.2 Prompt construction

CNN/DailyMail. Articles with character length in $(0, 3000)$ are eligible. The article is truncated to 1500 characters and inserted into

```
Summarize the following article in 3-5 sentences:
<article>
```

ELI5. The selected question is inserted into

```
Please explain like I'm five:
<question>
```

GSM8K. The problem is inserted into

```
Solve the following math problem step by step.  Give the
final answer clearly:
<question>
```

The resulting strings are passed through the model chat template. The prompt-token count and SHA-256 digest are saved for every measured row.

C.3 RANDOMNESS AND PAIRING

The base experiment seed is 7. Prompt i uses a deterministically derived per-prompt key from the base key and seed $7+i$. The original paired benchmark resets PyTorch and CUDA seeds per method and prompt. New sweep provenance records its own session and code settings. This design gives paired prompts and reproducible outputs while preventing one prompt's random stream from being reused as another prompt's stream. The generated token IDs, output hash, elapsed time, block count, AATPS, and peak memory are stored per method and prompt.

C.4 TIMING PROTOCOL

Each run first completes one full-generation warm-up on prompt zero for every measured method; the warm-up is excluded. For measured prompt i, the method order is cyclically shifted by i to reduce bias from clock or temperature drift. CUDA is synchronized immediately before and after the timed generation call. Peak allocated memory is reset before each call. Detector reconstruction and quality-model forwards are post hoc and excluded from the original paired generation timing. The new PF sweep records its own timing boundaries and pivot diagnostics, which are retained in its code provenance.

A physical GPU is intended to be valid for formal timing only when no unrelated process shares it during the run; an excluded warm-up or detached session name alone does not establish that condition. The paper reports hardware-dependent token rates together with AATPS and block latency, and it uses paired within-cell ratios as the main systems statistic.

C.5 FORMAL EXPERIMENT INVENTORY

- **Same-law acceleration and tradeoff methods:** nine cells, each with $n = 1000$, containing DIRECT-PF, Fused Shared-Noise Top-3 PF, VSpS, VUW, MSE, MWS, and MSE-PSEUDO at $L = 1$.
- **PF versus GLS:** six Qwen7/Vicuna cells, with $n = 2000$ on CNN/DailyMail and ELI5 and $n = 2000$ on GSM8K, comparing Fused PF $(L = 1, B = 3)$ and GLS $(L = 1, B = 4)$.
- **Longer-lookahead generation:** six main cells, 1000 prompts per method, PF $L \in \{1, 2, 3, 4\}$ and the four speculative baselines at $L \in \{2, 3, 4\}$; VUW is reused only as a non-speculative point.
- **Saved-output quality:** Direct/Fused $n = 1000$ per cell; seven-method $L = 1$ audit $n = 250$ per method and main cell; PF–GLS $n = 2000/1319$ as above.
- **Detector audit:** five-fold PF-positive cross-fitting; held-out VSpS null calibration; token budgets 30, 64, and 128; nominal FPR 1%.
- **Stability audit:** 64 sequences per cell, four positions per sequence, and fixed support size 50. The temperature grid is $0.4, 0.6, 0.8, 1.0, 1.2, 1.5$, and the perturbation-radius grid is $0.01, 0.02, 0.05, 0.1, 0.2$.

D COMPLETED SUPPLEMENTARY RESULTS

The supplementary results distinguish independently timed campaigns, model–dataset populations, detector variants, and actual sample counts.

Table 4: Per-cell completed PF lookahead sweep. $L = 1$ uses the shallow fused executor; $L > 1$ uses the bounded-width extension. TR values are not compared across model families or unrelated timing campaigns.

Setting	L	AATPS	TR	TR/$L1$	ms/block	Contexts	Exact/$L1$
Qwen14/0.5B–CNN/DM	1	1.826	20.607	1.000	88.61	4.00	1.000
	2	2.351	20.198	0.980	116.40	7.00	0.897
	3	2.687	18.870	0.916	142.41	9.99	0.900
	4	2.914	17.412	0.845	167.38	12.98	0.887
Qwen14/0.5B–ELI5	1	1.792	22.311	1.000	80.34	4.00	1.000
	2	2.249	21.246	0.952	105.86	6.98	0.844
	3	2.513	19.364	0.868	129.77	9.93	0.833
	4	2.668	17.392	0.780	153.41	12.87	0.834
Qwen14/0.5B–GSM8K	1	1.968	24.270	1.000	81.10	4.00	1.000
	2	2.791	26.044	1.073	107.17	6.98	0.913
	3	3.502	26.628	1.097	131.53	9.93	0.903
	4	4.108	26.459	1.090	155.26	12.87	0.916
Vicuna7/68M–CNN/DM	1	1.762	19.389	1.000	90.88	4.00	1.000
	2	2.223	19.073	0.984	116.56	6.99	0.950
	3	2.511	19.548	1.008	128.43	9.97	0.953
	4	2.701	19.159	0.988	140.98	12.94	0.949
Vicuna7/68M–ELI5	1	1.660	23.916	1.000	69.42	4.00	1.000
	2	1.954	21.209	0.887	92.14	6.98	0.904
	3	2.096	20.965	0.877	99.99	9.94	0.891
	4	2.162	19.689	0.823	109.78	12.89	0.894
Vicuna7/68M–GSM8K	1	1.736	24.093	1.000	72.05	4.00	1.000
	2	2.162	22.623	0.939	95.55	6.98	0.944
	3	2.393	23.255	0.965	102.89	9.94	0.936
	4	2.531	22.287	0.925	113.56	12.87	0.938

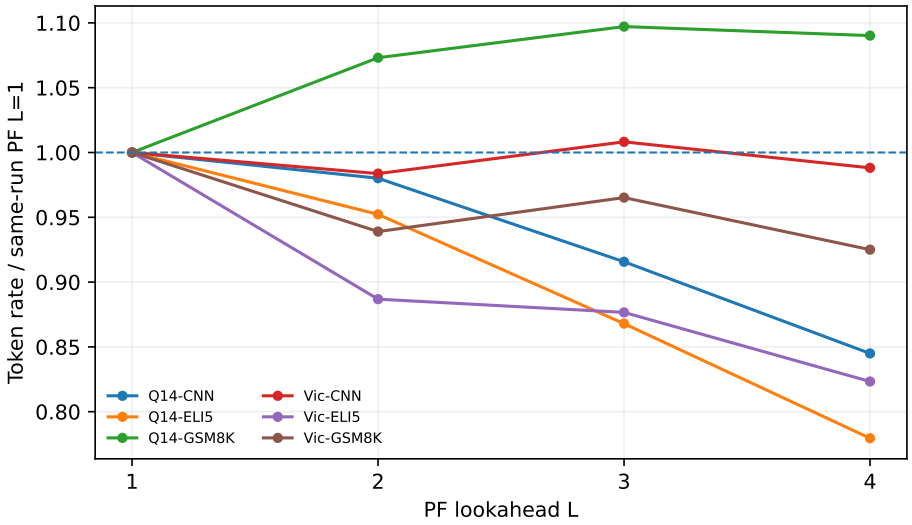


Figure 3: Per-cell PF throughput normalized to its own new $L = 1$ run. These within-sweep ratios are separate from Table 1. Larger lookahead increases token yield but is not uniformly faster.

Table 5: PF detection at all evidence budgets, using frozen $L = 1$ tables and a common eligible PF cohort within each budget. The cohorts differ across budgets.

M	Common n	Null FPR	TPR $L1$	TPR $L2$	TPR $L3$	TPR $L4$
30	2989	0.0191	0.2703	0.2710	0.2703	0.2707
64	2941	0.0119	0.4825	0.4818	0.4818	0.4825
128	1681	0.0107	0.6026	0.6032	0.6038	0.6038

The minimum retention of 30 makes the $M = 30$ filtered detector fall back to Original Fisher whenever any position is removed. At $M = 128$, early EOS substantially reduces the cohort; these values cannot be mixed with a baseline aggregate over different cells.

Table 6: Differences in PF TPR relative to $L = 1$ at $M = 64$. Intervals use 2000 stratified paired prompt-bootstrap replicates, conditional on the fitted tables, calibrated thresholds, and common eligible prompt sets. They do not include training/calibration uncertainty.

Lookahead	TPR difference	Conditional 95% interval
2	-0.0007	-0.0051 / 0.0034
3	-0.0007	-0.0054 / 0.0041
4	0.0000	-0.0044 / 0.0044

D.1 Per-cell PF lookahead and numerical agreement

D.2 Full measured baseline operating points

D.3 Detector and FPR sensitivity

D.4 Quality, stability, and cross-law comparisons

D.5 Scaling and historical implementation ablations

D.6 Longer-lookahead token yield and watermark operating points

Table 21 reports all four PF lookaheads. Pooled AATPS rises from 1.787 to 2.745. On the common 2941-sequence PF test cohort at $M = 64$, TPR is 0.4825, 0.4818, 0.4818, and 0.4825 for $L = 1, 2, 3, 4$, with realized null FPR 0.0119. The conditional paired intervals for changes from $L = 1$ include zero (Table 6); thus the measured detector power is similar, but these data do not prove pathwise watermark preservation.

Table 7: All measured operating points at $M = 64$ and nominal 1% FPR. PF uses frozen SigFisher; the competing family uses U-score. VSpS rates are null FPR, not watermark TPR. Each row covers all six main cells, but eligible prompt sets need not match across different methods.

Method	L	AATPS	TPR/FPR	n_+	Null FPR	n_0
Shared-noise PF	1	1.787	0.4825	2941	0.0119	2941
	2	2.265	0.4818	2941	0.0119	2941
	3	2.558	0.4818	2941	0.0119	2941
	4	2.745	0.4825	2941	0.0119	2941
MWS	1	1.558	0.5054	2974	0.0128	2971
	2	1.870	0.4726	2975	0.0087	2974
	3	2.049	0.5074	2974	0.0121	2969
	4	2.154	0.5057	2976	0.0131	2980
MSE	1	1.573	0.2334	2969	0.0128	2971
	2	1.904	0.1556	2969	0.0087	2974
	3	2.098	0.1249	2970	0.0121	2969
	4	2.213	0.1135	2977	0.0131	2980
MSE-PSEUDO (U)	1	1.575	0.3969	2973	0.0128	2971
	2	1.906	0.3676	2971	0.0087	2974
	3	2.099	0.3584	2969	0.0121	2969
	4	2.218	0.3542	2976	0.0131	2980
VSpS	1	1.576	0.0128	2971	0.0128	2971
	2	1.905	0.0087	2974	0.0087	2974
	3	2.097	0.0121	2969	0.0121	2969
	4	2.215	0.0131	2980	0.0131	2980
VUW	–	1.000	0.5089	2975	0.0128	2971

For PF, M counts emitted pivots; for U-score it counts non-skipped pivots, which can span more generated text. MSE-PSEUDO (U) names the installed pseudorandom-acceptance variant under U-score; an acceptance-aware native detector has not been validated here.

Table 8: All prespecified detector rows over nine cells. $M = 30, 64, 128$ have different eligible cohorts; the FPR, ANLPPT, and retention columns refer to $M = 64$.

Detector	TPR30	TPR64	TPR128	FPR64	ANLPPT64	Retain64
Lattimore-style ($\epsilon = 0.01$)	0.228	0.452	0.544	0.015	0.05899	1.000
Lattimore-style $\epsilon = 0.02$	0.220	0.441	0.543	0.014	0.05908	1.000
Lattimore-style $\epsilon = 0.05$	0.198	0.396	0.510	0.008	0.05719	1.000
Lattimore-style $\epsilon = 0.1$	0.194	0.341	0.498	0.007	0.05474	1.000
Li-style $\rho = 0.05$	0.189	0.373	0.490	0.015	0.05401	1.000
Li-style ($\rho = 0.1$)	0.193	0.396	0.521	0.014	0.05669	1.000
Li-style $\rho = 0.2$	0.186	0.390	0.504	0.009	0.05681	1.000
Li-style $\rho = 0.5$	0.202	0.300	0.482	0.005	0.05164	1.000
Original PF/Fisher	0.244	0.455	0.554	0.009	0.07903	1.000
SelfRare ≤ 1	0.244	0.400	0.532	0.011	0.07302	0.705
SelfRare ≤ 2	0.244	0.432	0.591	0.011	0.07650	0.808
SigFisher	0.244	0.481	0.600	0.013	0.08260	0.834

Table 9: Recorded matched-$L = 1$, $M = 64$ sensitivity over six cells. The table distinguishes the target nominal FPR from the actual held-out null rates; detection power should not be interpreted as measured at exactly equal realized FPR.

Nominal FPR	PF null FPR	U null FPR	PF TPR	MWS(U) TPR
0.005	0.0085	0.0074	0.4235	0.4122
0.010	0.0119	0.0128	0.4823	0.5054
0.020	0.0207	0.0256	0.5578	0.5938
0.050	0.0496	0.0522	0.6676	0.6866
0.100	0.1003	0.1091	0.7519	0.7764

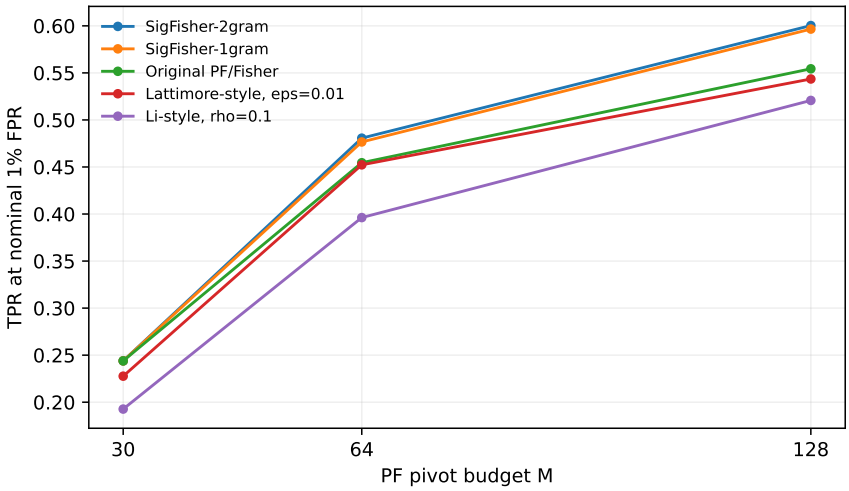


Figure 4: Completed nine-cell PF detector audit at nominal 1% FPR. The $M = 30, 64, 128$ cohorts have 4489, 4442, and 2677 positive sequences. This is not a same-cohort curve across budgets; the retained-count fallback explains equality of the three Fisher variants at $M = 30$.

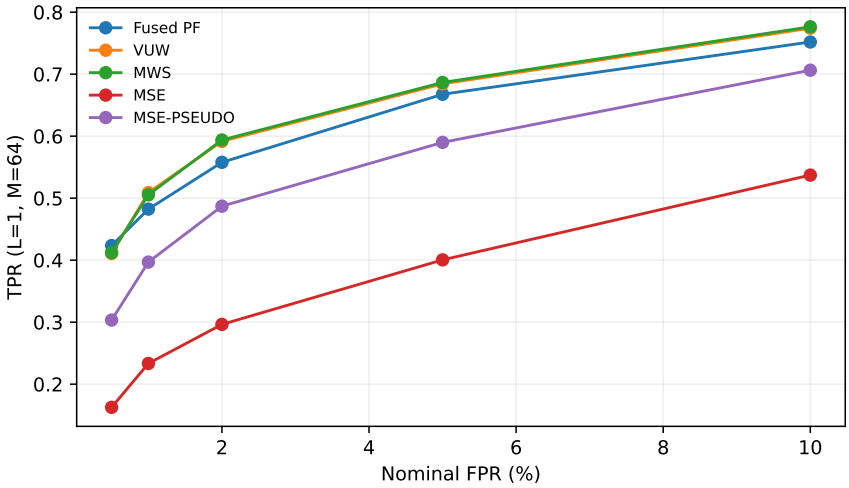


Figure 5: Historical matched-$L = 1$, $M = 64$ FPR sensitivity on six main cells. The horizontal axis is nominal, not realized, FPR. The plotted results do not constitute a full lookahead-by-FPR grid; realized calibration values remain in the reproducibility data.

Table 10: Corrected-label Direct/Fused PF recovery, evaluated using unfiltered Original PF/Fisher on matched eligible cohorts. This table does not use the filtered raw-sum audit as canonical SigFisher.

M	Common n	Null FPR	Direct TPR	Shallow PF TPR
30	2989	0.019	0.266	0.270
64	2942	0.011	0.458	0.467
128	1688	0.013	0.579	0.555

Each method uses its actual generation-side context-label domain. The domains differ, so this is a statistical control rather than a same-field token-equality experiment. The $M = 128$ cohort differs from the other budgets.

Table 11: Additional Direct/Fused quality metrics ($n = 1000$ per method and cell), shown as Direct PF/SpecPF.

Setting	Selected rank	Top-1 rate	Repeat-5	Distinct-2
Qwen14/0.5B–CNN/DM	1.352 / 1.351	0.856 / 0.858	0.0007 / 0.0009	0.964 / 0.964
Qwen14/0.5B–ELI5	1.448 / 1.452	0.810 / 0.807	0.0030 / 0.0033	0.927 / 0.927
Qwen14/0.5B–GSM8K	1.143 / 1.145	0.915 / 0.915	0.0452 / 0.0476	0.751 / 0.749
Vicuna7/68M–CNN/DM	1.127 / 1.126	0.931 / 0.930	0.0024 / 0.0028	0.938 / 0.938
Vicuna7/68M–ELI5	1.277 / 1.288	0.872 / 0.871	0.0161 / 0.0149	0.869 / 0.868
Vicuna7/68M–GSM8K	1.120 / 1.116	0.930 / 0.933	0.0837 / 0.0865	0.672 / 0.669

Table 12: Completed fixed-support stability–utility audit at $T = 1$, with 64 sequences and four positions per sequence per cell. Each support is the base target top-50 set. PF probabilities are evaluated by finite-support numerical quadrature.

Setting	Contexts	PF regret	Softmax regret	Reduction	Top-1 PF/SM
Qwen14/0.5B–CNN/DM	256	0.152	0.223	31.9%	0.894 / 0.861
Qwen14/0.5B–ELI5	256	0.333	0.466	28.4%	0.761 / 0.701
Qwen14/0.5B–GSM8K	256	0.082	0.135	38.7%	0.951 / 0.929
Vicuna7/68M–CNN/DM	256	0.135	0.203	33.4%	0.902 / 0.872
Vicuna7/68M–ELI5	256	0.254	0.362	29.6%	0.823 / 0.778
Vicuna7/68M–GSM8K	256	0.150	0.213	29.8%	0.874 / 0.842

The common theoretical budget is $2/T$. This compares expected utility of two decoders on the same selected logits, not task accuracy or global stability across changing hard top-k supports.

Table 13: Qwen14/0.5B: quality of all seven methods at $L = 1$, with 250 saved sequences per method and dataset. This cross-law comparison is distinct from the 1000-sequence Direct/Fused audit.

Dataset	Method	Raw PPL	Proc. LPPL	Regret	Distinct-2	Task
CNN/DM	Direct PF	1.580	0.450	0.203	0.966	0.173
	Shallow PF	1.574	0.445	0.202	0.965	0.169
	VSpS	1.718	0.532	0.286	0.966	0.171
	VUW	1.730	0.537	0.289	0.965	0.169
	MSE	1.730	0.535	0.288	0.967	0.170
	MWS	1.731	0.537	0.290	0.966	0.170
	MSE-PSEUDO	1.733	0.540	0.294	0.967	0.171
ELI5	Direct PF	1.820	0.592	0.260	0.923	0.113
	Shallow PF	1.835	0.599	0.265	0.923	0.114
	VSpS	2.066	0.716	0.379	0.929	0.113
	VUW	2.095	0.730	0.388	0.930	0.114
	MSE	2.100	0.733	0.392	0.929	0.113
	MWS	2.084	0.724	0.385	0.930	0.113
	MSE-PSEUDO	2.068	0.718	0.380	0.928	0.113
GSM8K	Direct PF	1.301	0.258	0.110	0.757	0.048
	Shallow PF	1.306	0.261	0.112	0.755	0.052
	VSpS	1.379	0.314	0.164	0.754	0.028
	VUW	1.377	0.314	0.165	0.762	0.028
	MSE	1.372	0.309	0.160	0.756	0.036
	MWS	1.369	0.309	0.159	0.762	0.016
	MSE-PSEUDO	1.391	0.322	0.172	0.764	0.040

Task is ROUGE-L for CNN/DM and ELI5 and exact match for GSM8K. The short generation cap and prompt/extraction configuration should be considered when interpreting the low GSM8K exact-match values.

Table 14: Vicuna7/68M: quality of all seven methods at $L = 1$, with 250 saved sequences per method and dataset. This cross-law comparison is distinct from the 1000-sequence Direct/Fused audit.

Dataset	Method	Raw PPL	Proc. LPPL	Regret	Distinct-2	Task
CNN/DM	Direct PF	1.224	0.198	0.086	0.943	0.198
	Shallow PF	1.212	0.188	0.079	0.939	0.196
	VSpS	1.289	0.246	0.130	0.939	0.197
	VUW	1.293	0.250	0.133	0.942	0.191
	MSE	1.288	0.246	0.132	0.943	0.198
	MWS	1.293	0.250	0.134	0.943	0.192
	MSE-PSEUDO	1.291	0.249	0.134	0.941	0.193
ELI5	Direct PF	1.502	0.399	0.177	0.860	0.125
	Shallow PF	1.529	0.416	0.187	0.865	0.126
	VSpS	1.662	0.494	0.265	0.873	0.124
	VUW	1.675	0.504	0.270	0.872	0.123
	MSE	1.675	0.503	0.275	0.861	0.124
	MWS	1.675	0.505	0.272	0.873	0.125
	MSE-PSEUDO	1.681	0.506	0.274	0.873	0.123
GSM8K	Direct PF	1.235	0.207	0.089	0.677	0.044
	Shallow PF	1.231	0.204	0.086	0.675	0.048
	VSpS	1.325	0.272	0.150	0.678	0.076
	VUW	1.313	0.264	0.142	0.679	0.084
	MSE	1.311	0.265	0.139	0.687	0.088
	MWS	1.312	0.264	0.140	0.684	0.072
	MSE-PSEUDO	1.315	0.266	0.140	0.686	0.056

Task is ROUGE-L for CNN/DM and ELI5 and exact match for GSM8K. The short generation cap and prompt/extraction configuration should be considered when interpreting the low GSM8K exact-match values.

Table 15: PF versus GLS in the separate original systems campaign. The entries compare $L = 1, B = 3$ PF with $L = 1, B = 4$ GLS. These are not same-law acceleration ratios.

Setting	n	AATPS PF/INV	TR PF/INV	TR ratio	GiB PF/INV
Qwen7/0.5B–CNN/DM	2000	1.836 / 1.772	30.640 / 28.758	1.065×	15.85 / 16.20
Qwen7/0.5B–ELI5	2000	1.816 / 1.764	31.753 / 29.921	1.061×	15.25 / 15.56
Qwen7/0.5B–GSM8K	1319	1.973 / 1.947	34.160 / 32.635	1.047×	15.42 / 15.74
Vicuna7/68M–CNN/DM	2000	1.756 / 1.690	46.100 / 42.127	1.094×	14.71 / 14.19
Vicuna7/68M–ELI5	2000	1.661 / 1.592	49.628 / 47.437	1.046×	13.33 / 13.19
Vicuna7/68M–GSM8K	1319	1.737 / 1.665	51.262 / 49.119	1.044×	13.70 / 13.44

Table 16: Completed PF/GLS quality comparison. Entries are PF/GLS. CNN/DM and ELI5 have 2000 sequences per method, and GSM8K has 1319. The target decoding laws differ.

Setting	n	Raw PPL	Proc. LPPL	Regret	Task
Qwen7/0.5B–CNN/DM	2000	1.512 / 1.704	0.405 / 0.520	0.183 / 0.283	0.189 / 0.183
Qwen7/0.5B–ELI5	2000	1.892 / 2.168	0.629 / 0.763	0.282 / 0.410	0.113 / 0.113
Qwen7/0.5B–GSM8K	1319	1.159 / 1.207	0.146 / 0.185	0.060 / 0.095	0.028 / 0.033
Vicuna7/68M–CNN/DM	2000	1.253 / 1.362	0.220 / 0.295	0.096 / 0.162	0.214 / 0.209
Vicuna7/68M–ELI5	2000	1.530 / 1.729	0.406 / 0.531	0.182 / 0.295	0.125 / 0.126
Vicuna7/68M–GSM8K	1319	1.231 / 1.331	0.204 / 0.278	0.088 / 0.150	0.066 / 0.061

Task is ROUGE-L for CNN/DM and ELI5 and exact match for GSM8K. Likelihood/utility improvements are not uniformly accompanied by improved task scores.

Table 17: Qwen7/0.5B below-break-even regime, with 1000 prompts per dataset. Higher token yield per speculative block does not offset the runtime overhead in these paired runs.

Dataset	Direct TR	Shallow PF TR	Speedup	AATPS
CNN/DM	33.857	29.459	0.870×	1.840
ELI5	34.997	30.883	0.882×	1.817
GSM8K	34.542	32.695	0.947×	1.974

Table 18: Historical Qwen7/0.5B implementation ablation ($n = 100$, $L = 1, B = 3$). Generic and fused executors have identical recorded outputs and AATPS; fusion reduces execution overhead.

Dataset	AATPS	Generic TR	Fused TR	TR ratio	Output match
CNN/DM	1.844	19.652	20.967	1.067×	1.000
ELI5	1.816	20.953	22.511	1.074×	1.000
GSM8K	1.974	24.007	25.379	1.057×	1.000

Table 19: Historical Qwen7/0.5B width/adaptive-policy ablation ($n = 100$, $L = 1$). These timings come from a different campaign from Table 18 and should not be compared across the two tables.

Dataset	Policy	AATPS	TR	ms/block
CNN/DM	Fixed $B = 2$	1.792	24.908	71.96
	Fixed $B = 3$	1.844	25.004	73.77
	Fixed $B = 4$	1.873	24.606	76.10
	Adaptive $\tau = 0.5$	1.794	24.903	72.03
	Adaptive $\tau = 1$	1.797	25.023	71.81
	Adaptive $\tau = 1.5$	1.806	25.176	71.72
ELI5	Fixed $B = 2$	1.741	25.284	68.85
	Fixed $B = 3$	1.816	25.920	70.04
	Fixed $B = 4$	1.855	26.464	70.10
	Adaptive $\tau = 0.5$	1.745	24.884	70.13
	Adaptive $\tau = 1$	1.754	25.158	69.71
	Adaptive $\tau = 1.5$	1.767	25.519	69.24
GSM8K	Fixed $B = 2$	1.954	27.951	69.90
	Fixed $B = 3$	1.974	28.014	70.46
	Fixed $B = 4$	1.982	27.906	71.04
	Adaptive $\tau = 0.5$	1.954	27.835	70.21
	Adaptive $\tau = 1$	1.956	27.911	70.07
	Adaptive $\tau = 1.5$	1.957	28.227	69.33

The historical fixed-width comparison covers $B = 2, 3, 4$; it does not include $B = 1$ or a corresponding final-model sweep. The adaptive policy uses shared-noise score gaps. These historical results do not establish a universally best width.

Table 20: PF quality experiments (1000 sequences per method and cell). Entries are Direct PF/SpecPF. The independently generated streams test empirical marginal consistency, not pathwise equality.

Setting	Raw PPL	Proc. LPPL	Raw regret	Task score
Qwen14/0.5B–CNN/DM	1.589 / 1.589	0.455 / 0.455	0.205 / 0.205	0.180 / 0.178
Qwen14/0.5B–ELI5	1.835 / 1.844	0.599 / 0.604	0.263 / 0.269	0.113 / 0.112
Qwen14/0.5B–GSM8K	1.287 / 1.294	0.248 / 0.252	0.104 / 0.107	0.045 / 0.043
Vicuna7/68M–CNN/DM	1.247 / 1.241	0.212 / 0.211	0.092 / 0.091	0.205 / 0.210
Vicuna7/68M–ELI5	1.502 / 1.515	0.399 / 0.408	0.176 / 0.182	0.124 / 0.124
Vicuna7/68M–GSM8K	1.236 / 1.229	0.208 / 0.202	0.089 / 0.087	0.055 / 0.061

Task score is ROUGE-L on CNN/DM and ELI5 and exact match on GSM8K. Metrics are means of saved per-sequence values. Lower PPL and regret quantify target-model likelihood and utility; they do not by themselves imply better task performance.

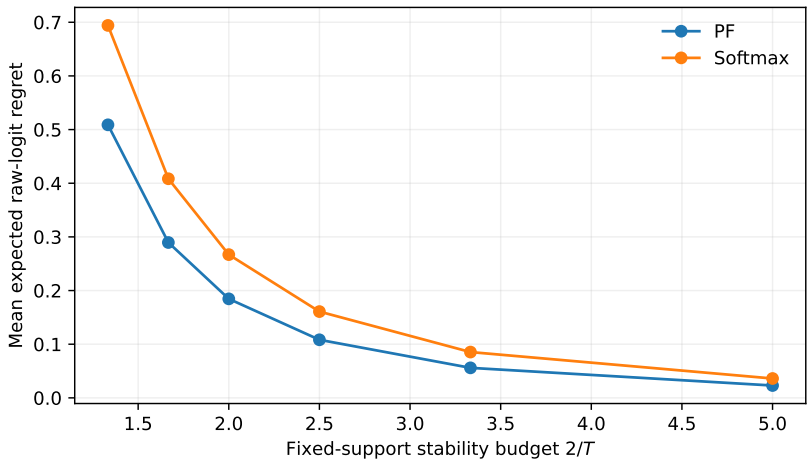


Figure 6: Completed six-cell fixed-support stability–utility audit. Each curve point is an equal-cell mean over the same 256 selected contexts per cell. Smaller $2/T$ is a stronger stability constraint. The figure compares expected raw-logit regret, not downstream task quality or dynamic-support stability.

Table 21: PF lookahead sweep on all six main cells (1000 prompts per cell and lookahead). AATPS pools emitted tokens and blocks. Detection uses the same frozen $L = 1$ SigFisher tables, thresholds, and 2941 common eligible held-out prompts at $M = 64$.

L	AATPS	TPR	Null FPR	Geo. TR/$L1$	Token-exact/$L1$
1	1.787	0.4825	0.0119	1.000×	1.000
2	2.265	0.4818	0.0119	0.968×	0.909
3	2.558	0.4818	0.0119	0.952×	0.903
4	2.745	0.4825	0.0119	0.903×	0.903

TR ratios are paired to the new $L = 1$ run within each cell, not to the earlier Direct-PF campaign. Token-exact is whole-sequence equality over all 6000 prompts; pivot-exact has the same reported values. A detector-rate comparison is not a proof of equality of output laws.