GIVE THE DRAFTER A SECOND CHANCE TO GUESS: PARALLEL RECONSIDERATION FOR SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding accelerates large language model inference with a small drafter, whose per-guess acceptance rate determines the achievable speedup. Feature-level drafters share a design choice: every token, easy or hard, is drafted under a *single* forward pass. Many of the tokens the target rejects are ones the drafter is *itself unsure about* yet the target answers decisively. To address this, we introduce **Parallel Reconsideration**: guided by its own uncertainty, the drafter *reconsiders* its hesitant guesses with extra passes that run *in parallel* with the one-pass draft, adding negligible latency. A co-designed training recipe and serving-time drafting algorithm append these reconsidered guesses to the original draft tree as refined candidates. Across models and benchmarks, on average, Parallel Reconsideration improves over the baselines by **5.3%** to **10.2%** on acceptance length and by **3.5%** to **9.2%** on speedup.

1 INTRODUCTION

Large language models (LLMs) have become a pivotal component of modern computing, powering assistants, search, and the fast-growing class of *agentic* systems that plan, call tools, and write code over long horizons (Anthropic, 2025; OpenAI, 2025; Google, 2025; Moonshot AI, 2026). Yet this shift exposes a stubborn cost: in the agentic era even a simple coding task can unfold into generated tokens across many chained model calls, and each of those tokens is produced *one at a time*. This is not an implementation artifact but the nature of **autoregressive decoding**: every token is conditioned on all previous tokens, so a single forward pass of a multi-billion-parameter model yields exactly one token, and end-to-end latency scales with the length of the generation (Pope et al., 2023). As outputs grow longer, this sequential bottleneck, rather than raw compute, increasingly governs how fast a model can respond.

Speculative decoding directly mitigates this bottleneck (Leviathan et al., 2023; Xia et al., 2023). A cheap *drafter* proposes several candidate tokens, and the target model *verifies* them in parallel in a *single* forward pass. A *modified rejection sampling* step keeps the longest run of drafted tokens the target agrees with, provably preserving its output distribution exactly (Leviathan et al., 2023); the saving comes from *parallelism*, as k accepted tokens cost one target pass instead of k expensive ones. The drafter thus decides everything: the more tokens it gets right per round, the greater the speedup. One notable class is the EAGLE family (Li et al., 2024a;b; 2025), which trains a lightweight drafter that predicts at the *feature* level rather than the token level, reusing the target's hidden states to draft with a single small decoder layer and setting the standard for modern speculative decoding.

This efficiency, however, comes with a design choice shared by feature-level drafters: **every token, easy or hard, is drafted with the same fixed budget of a single forward pass through a small decoder.**[1] Some proposed tokens the target rejects are not ones the draft gets confidently wrong; they are tokens the draft is *itself unsure about* even though the target has a clear answer there. We hypothesize that part of this loss stems from how much the draft can *compute*: a single pass of one small layer may not suffice to settle these hard tokens, yet the draft must commit and move on regardless.

[1] For ease of discussion we assume a single-layer, EAGLE-style drafter throughout; the ideas extend directly to deeper or otherwise structured drafters, where the fixed budget is one pass through the whole drafter.

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.

The obvious fix is to give the drafter more capacity, e.g. more layers. Yet a bigger drafter slows the entire round and eats into the speedup that is the whole point of speculative decoding. This exposes the real tension: **how can the drafter spend more compute on the tokens that need it, without paying for that compute on the tokens that do not?** A viable answer must be *selective*, spending the extra compute only at the uncertain positions; *(near-)free*, adding negligible wall-clock latency over the ordinary draft; and *lossless*, leaving the verification untouched.

We answer this with **Parallel Reconsideration**, which grants the drafter a *second chance to guess* on the tokens it is unsure about. The drafter's own uncertainty is the signal: where it is already decisive, the guess is committed as usual; where it hesitates, the drafter is given more attempts to revise its guess into a better one before committing. Moreover, this reconsideration is nearly *free*: the additional attempts are batched into the same forward pass as the ordinary draft, adding only negligible parallel overhead at inference time. Our contributions are as follows:

- **Parallel Reconsideration.** We introduce uncertainty-guided *draft-time scaling* to feature-level drafters: instead of a fixed one-pass-per-token budget, the drafter scales its computation per position according to its own uncertainty. Parallel Reconsideration reconsiders the uncertain guesses with additional passes that run *in parallel* with the ordinary draft, so the added computation is nearly free (Section 3).
- **A co-design of training and serving.** We design a training recipe under which the reconsideration branch deliberately learns to correct the drafter's mistakes without degrading its ordinary drafting branch (Section 4). In serving, we design a drafting algorithm that grafts the reconsidered guesses onto the ordinary draft tree as *additional* candidates (Section 5).
- **Extensive empirical evaluation.** Across models and benchmarks, Parallel Reconsideration improves acceptance length over the baselines by **5.3–10.2%** on average across models, draft budgets, and sampling temperatures, at a nearly identical per-round drafting cost. We further show that spending the same compute on a wider draft or a second decoder layer does not match it, and ablate each component of the design (Section 6).

2 Background and Motivation

2.1 Preliminaries and the cost model

We consider an EAGLE-style drafter (Li et al., 2024a) that predicts at the feature level. Let f_p^0 denote the draft feature at speculative position p produced by a single forward pass of the drafter's single decoder layer M, let $e(t)$ be the embedding of token t, let $\mathrm{fc}(g, e)$ be EAGLE's fusion of a feature g with a token embedding e (concatenate, then project), and let $\mathrm{head}(\cdot)$ be the LM head. To advance one position, the drafter samples a token $t_{p+1} \sim \mathrm{head}(f_p^0)$ from the current feature and predicts the next feature by $f_{p+1}^0 = M(\mathrm{fc}(f_p^0, e(t_{p+1})))$, one ***draft step***. Each round, the drafter generates candidate tokens in γ steps sequentially, incurring a drafting cost

$$T_{\text{draft}} = \gamma \cdot t_{\text{step}}, \tag{1}$$

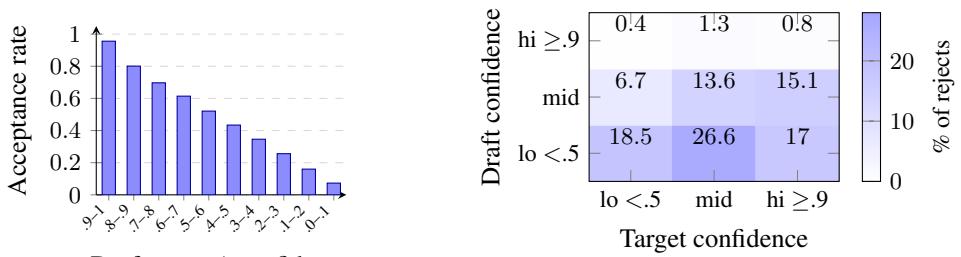
where t_{step} is the latency of a single forward pass of M; the drafting cost therefore grows linearly with γ. The target model then verifies the candidates in one forward pass, costing T_{verify}, and accepts the longest prefix it agrees with. The expected number of tokens accepted per round is the *acceptance length* τ, and the amortized cost per accepted token is

$$\frac{T_{\text{draft}} + T_{\text{verify}}}{\tau} = \frac{\gamma\, t_{\text{step}} + T_{\text{verify}}}{\tau}. \tag{2}$$

The goal is therefore to **raise the acceptance length τ while holding $T_{\text{draft}} + T_{\text{verify}}$ on par with prior methods**: a drafter that raises τ by inflating $\gamma\, t_{\text{step}}$ can easily lose more in the numerator than it gains in the denominator. Underlying this formulation is an implicit assumption that prior work has left unquestioned: each step runs M *exactly once*, so every token, easy or hard, is drafted from a single forward pass with no opportunity to revise.

2.2 Where does the drafter lose acceptance?

To understand where acceptance is lost, we analyze a trained EAGLE-3.1 drafter for Llama-3.1-8B-Instruct on a held-out split. For each drafted position p, we read off two quantities: the



(a) Acceptance vs. drafter confidence.

(b) Draft vs. target confidence (% of rejects).

Figure 1: **Two observations on where an EAGLE-style drafter loses acceptance. (a)** Acceptance rate versus the drafter's own top-1 confidence, which falls from 95.6% on confident positions to 7.3% on the least confident. **(b)** Drafter confidence against target confidence over the rejected tokens (% of the reject pool): the drafter is unsure on 62.1% of them, and 17.0% of the pool is unsure at a decisive target ($c_p^{\mathrm{t}} \geq 0.9$).

drafter confidence $c_p^{\mathrm{d}} = \max_t \mathrm{head}(f_p^0)_t$, its own top-1 probability, and the *target confidence* $c_p^{\mathrm{t}} = \max_t \pi_{\mathrm{tgt}}(t \mid x_{<p})$, the top-1 probability of the target's next-token distribution at the same position. The drafter confidence measures how sure the drafter is; the target confidence measures how *resolvable* the position is, i.e. when c_p^{t} is high, a single correct token exists and can, in principle, be predicted by the drafter. Figure 1 summarizes two observations relating these quantities to acceptance.

Observation 1: drafter's confidence closely correlates with the acceptance probability. Acceptance tracks c_p^{d} closely (Figure 1a): positions the drafter is confident about ($c_p^{\mathrm{d}} \geq 0.9$) are accepted 95.6% of the time, and acceptance falls steadily as its confidence drops: to 0.52 at 0.5–0.6 and 0.07 in the lowest bucket. The drafter is thus seldom confidently wrong; where it loses acceptance, it was already unsure. The drafter's own confidence, available at draft time, therefore indicates how likely the target is to reject the token it drafts.

Observation 2: a fair share of rejections is recoverable, but confidence does not say which. Figure 1b breaks down the rejected tokens based on the drafter's and target's model confidence. Among rejected tokens, the drafter is unsure on 62.1% of them, and 43.6% of them are the ones the target model still favors one token ($c_p^{\mathrm{t}} \geq 0.5$). Target model is decisive on 17.0% of rejected tokens ($c_p^{\mathrm{t}} \geq 0.9$), against a comparable 18.5% where the target model is unsure ($c_p^{\mathrm{t}} < 0.5$). A fair share of the acceptance the drafter loses is thus recoverable: at those positions a token the target would have accepted was there to be drafted. But the drafter cannot tell which: its confidence is equally low at a decisive target and at an ambiguous one.

2.3 Spend compute where the drafter is unsure

The two observations compose into a single reading: the drafter loses acceptance on tokens it is itself unsure about (Observation 1), on a fair share of which the target nonetheless has a decisive answer (Observation 2). It is not confidently wrong at these positions, and a correct token is there to be drafted, yet it is **committed prematurely**: a single draft step is not enough to resolve the uncertain position p before the drafter must commit its guess and move to the next position $p+1$. What the drafter is missing, then, is not a better objective but more computation:

Give the drafter more compute where it is unsure, and none extra where it is not.

Within feature-level drafters, prior work mostly refines *what* the drafter computes in a single step rather than *how much* it computes, which stays fixed at one forward pass of M per position. A naive bigger drafter with more compute is self-defeating under Eq. (2), since it raises t_{step} on *every* position, including the confident majority that one step already resolves, inflating T_{draft} faster than it lifts τ (see Section 6.4 for details). We therefore scale draft-time compute *selectively*, spending extra draft steps only on the uncertain positions where the acceptance lost is recoverable, and leaving the confident

majority at one forward pass, as **Parallel Reconsideration** (PR). Section 6 revisits these observations and analyzes whether Parallel Reconsideration indeed helps on the raised challenges.

3 PARALLEL RECONSIDERATION

We introduce **Parallel Reconsideration** with the reconsideration step (Section 3.1), the learned self-query that lets the drafter refine the single-pass draft rather than growing the next one (Section 3.2), the learned gate that decides which positions have a chance of drafting a better token through reconsideration (Section 3.3), and the parallel execution that keeps it cheap (Section 3.4). We describe the method with a single reconsideration step for clarity, though in practice, the drafter may take more than one.

3.1 THE RECONSIDERATION STEP

Recall that the ordinary draft step at position p reads the one-pass feature f_p^0, samples the next token $t_{p+1} \sim \mathrm{head}(f_p^0)$, and advances to position $p+1$ via $f_{p+1}^0 = M(\mathrm{fc}(f_p^0, e(t_{p+1})))$. Given an uncertain feature f_p^0 and Q, a learned self-query (defined below), ***Reconsideration*** spends one additional forward pass to refine it into f_p^1, from which the token is re-sampled:

$$f_p^1 = M\big(\mathrm{fc}(f_p^0, Q(f_p^0))\big), \qquad \tilde{t}_{p+1} \sim \mathrm{head}(f_p^1), \tag{3}$$

3.2 THE SELF-QUERY

In an EAGLE-style drafter, current feature f_p^0 does not determine the next feature on its own, since different sampled tokens t_{p+1} continue into different f_{p+1}^0 (Li et al., 2024a), so the drafter fuses $e(t_{p+1})$ to tell M which branch to advance along. Reconsideration, however, stays at p without advancing to re-decide the very token t_{p+1} the drafter is unsure about. We therefore give reconsideration a *steering embedding*: a learned *self-query* $Q(f_p^0)$, with $Q : \mathbb{R}^d \to \mathbb{R}^d$, encoding what a second look at an uncertain state should attend to in order to refine f_p^0 instead of the naive one-pass draft of next token's feature.

3.3 THE RECONSIDERATION GATE

Not every uncertain position is worth another pass, and c_p^{d} does not say which are: it reads the same where the target is decisive and where it is ambiguous (Observation 2). Recoverability has to be learned from the feature itself. Writing $f_p^0, \ldots, f_p^R$ for the features at each reconsideration level, up to at most R, a small gate at level r reads each and emits a halt probability:

$$\lambda_r = \sigma\big(\mathrm{gate}(f_p^r)\big), \qquad q(r) = \lambda_r \prod_{s<r} (1 - \lambda_s). \tag{4}$$

Every level asks the same question, whether another pass still has a chance of drafting a better token, so $q(r)$ is the chance a position reaches level r and stops there.

3.4 RECONSIDERING IN PARALLEL

A naive Reconsideration adds forward passes dedicated to the reconsideration, and run naively in series each one stalls the draft at p until f_p^1 is computed: a draft of γ positions with N reconsidered ones costs $(\gamma + N)\, t_{\mathrm{step}}$ instead of $\gamma\, t_{\mathrm{step}}$ (Figure 3, top). However, the one-pass draft step and its reconsideration,

$$f_{p+1}^0 = M\big(\mathrm{fc}(f_p^0, e(t_{p+1}))\big), \qquad f_p^1 = M\big(\mathrm{fc}(f_p^0, Q(f_p^0))\big), \tag{5}$$

are disjoint, so the two can be batched into a single forward of M: *co-forwarding* rather than serializing (Figure 3, bottom). The decoding stage of LM serving is memory bandwidth-bound, so the added rows cost a negligible ϵ overhead (Appendix A.1). Drafting thus becomes *interleaved*: the ordinary steps grow the draft tree as before, and in parallel, a reconsidered position adds its $\tilde{t}_{p+1}$ as a reconsidered child beside the one-pass ones. Both kinds are draft candidates, pruned and verified together under the standard speculative-decoding verification in serving time.

4 TRAINING PARALLEL RECONSIDERATION

The single decoder layer M must serve two heads at once: the one-pass $\mathrm{head}(f_p^0)$ that guesses and advances the draft to position $p+1$, and the reconsidered $\mathrm{head}(f_p^1)$ that revises a premature guess. Neither can be trained merely as an intermediate step toward the other, because at serving *both* are read out and committed to the tree: the one-pass head guesses the token at most positions and the reconsidered head revises it at the uncertain ones. Training must therefore keep the one-pass head as accurate as an ordinary drafter while pushing the reconsidered head to win on the recoverable tokens single pass misses, and doing so is delicate because the two share the feature f_p^0. Supervise only the reconsidered head and the one-pass head degrades, overloading all tokens to be reconsidered and losing the ordinary-drafter quality on the easy tokens; supervise both to match the target with equal weight and gradient descent finds the trivial solution $f_p^1 \approx f_p^0$, so the reconsidered token merely echoes the same one-pass token. Either failure would undercut speculative-decoding quality (Appendix J). We therefore hold the one-pass head to ordinary-drafter quality with an unconditional loss, and supervise each reconsidered feature only at the positions and depths where it is actually used.

One loss, every depth. Thus during training, we predict token at every depth r and score it with $\ell_p^r = \mathrm{CE}(\mathrm{head}(f_p^r), t^\star_{p+1})$, then weight each depth's loss by the mass $q(r)$ the gate routes to it, while leaving the one-pass depth unweighted. The gate has already judged the chance each depth is the one that settles the token (Eq. (4)), so $q(r)$ is the natural weight: a depth earns gradient in proportion to how often it is the one asked to answer. Each depth's loss also shapes where the gate halts, pulling its mass toward whichever depth reads out the token best. Weighting alone, however, leaves the gate free to abandon a depth entirely (a depth it stops routing to receives no gradient), so we add an *entropy* bonus that keeps every depth in use. With $\mathrm{sg}[\cdot]$ the stop-gradient, we have the training objective:

$$\mathcal{L} = \sum_p \Big(\underbrace{\ell_p^0}_{\text{one-pass}} + \underbrace{\sum_{r\geq 1} \mathrm{sg}[q(r)]\, \ell_p^r}_{\text{reconsideration heads}} + \underbrace{\sum_{r\geq 0} q(r)\, \mathrm{sg}[\ell_p^r]}_{\text{gate}} - \underbrace{\beta\, H\big(\mathbb{E}_p[\,q\,]\big)}_{\text{balance}} \Big), \tag{6}$$

The mixed cache. A reconsidered child commits the key/value of its parent's halt level rather than of parent's one-pass feature, so a position deeper in the draft attends over a *mixture*: one-pass keys and values at the confident ancestors that were never reconsidered, reconsidered ones at the uncertain ancestors that were. To adapt to this serving phenomenon, each position j we sample a halt depth $d_j \sim q$ and commit the key/value that depth produces,

$$\kappa_j = \mathrm{kv}\Big(f_j^{\,d_j}\Big), \qquad d_j \sim q, \tag{7}$$

so position p attends over the prefix $\kappa_{<p}$, resampled afresh each step.

Appendix B gives the full training recipe, including the two on-policy rollouts the two heads are trained on, the per-depth weighting, and the analysis of the balance term.

5 SERVING A RECONSIDERED TREE

EAGLE-3.1 serves by building a *dynamic draft tree*, grown over every round's γ draft steps. Each step forwards a batch of B_w frontier nodes; every forwarded node offers children sampled from its feature, and each node v carries a score equal to its cumulative path probability from the root, $s(v) = \sum_{u\in \mathrm{path}(v)} \log \pi_{\mathrm{one}}(u \mid \mathrm{parent}(u))$, where π_{one} is the one-pass distribution of the node's parent. After γ steps, a single global cut keeps the B highest-scoring nodes and prunes the rest.

Parallel Reconsideration keeps that procedure and has a second one in parallel. Alongside the frontier-growing nodes the drafter holds a *reconsideration pool* $\mathcal{C}$ of nodes that have not yet halted. A draft step now has two things to do, advance the frontier and reconsider the pool, and it does them at once: the B_w highest-scoring frontier nodes move the draft to the next position, and the m highest-bidding nodes of $\mathcal{C}$ are reconsidered, within the same forward batch of M (Section 3.4), so the γ-step growth of Eq. (1) is untouched.

Which m nodes are reconsidered follows from what reconsideration changes: on the tree, reconsidering a node v leaves v itself fixed and revises the distribution it offers its children (Section 3.1). A

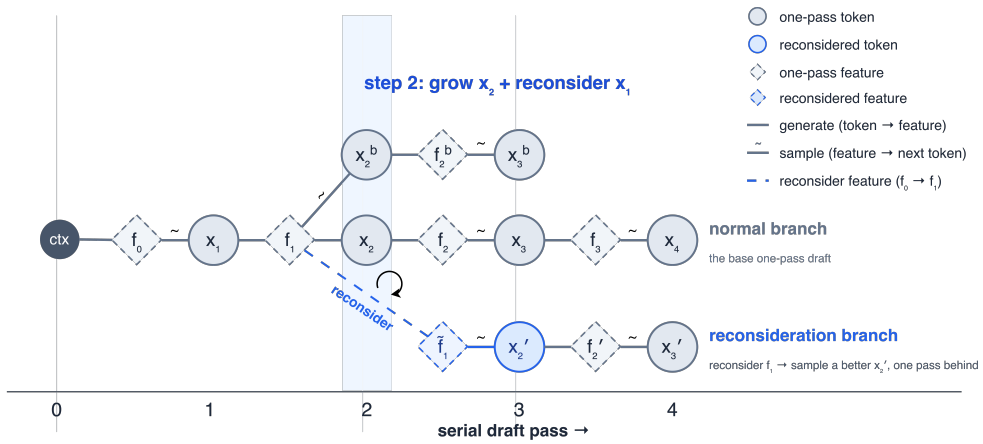


Figure 2: **Co-forwarding reconsideration on the draft tree.** Each token produces a feature (diamond) from which the next token is sampled ($\sim$); the one-pass draft (gray, level 0) forms the main branches, and tree width comes free because a node's top-k children are all sampled from its single feature. The reconsideration pass (blue, level 1) deepens a node's feature within the same batched forward as the next draft step so it proposes a corrected token (here x_2') with no extra sequential draft step. The one-pass draft still reaches x_4 while the reconsidered branch grows in parallel.

node is therefore worth reconsidering when the draft is likely to reach it and its one-pass continuation is likely to be wrong, which together are the probability that the draft breaks immediately after v. We rank the pool by

$$\mathrm{bid}(v) = \underbrace{s(v)}_{\log \Pr[\text{reach } v]} + \underbrace{\log\big(1 - \max_t \pi_{\mathrm{one}}(t \mid v)\big)}_{\log \Pr[v\text{'s one-pass child is wrong}]} = \log \Pr[\text{the draft breaks at } v\text{'s children}], \tag{8}$$

and reconsider the top m. We note that the gate and the bid are two complementary mechanisms: The gate asks whether another pass has a *chance* of drafting a better token, which confidence cannot answer (Observation 2); a node it declines to halt, meaning it has a chance of drafting a better child. The bid then asks, among the nodes that have that chance, which are the most valuable to reconsider this step, and confidence is what answers it: the less confident v's one-pass guess, the likelier the first draft guesses it wrong, and the more there is to gain by reconsidering it (Observation 1).

A reconsidered node re-draws its children from the reconsidered feature, and the token $\tilde{t}_v \sim \mathrm{head}(f_v^{d_v})$ it now proposes enters the tree as an *additional* child beside the ones its one-pass feature offered. Once node v halts at d_v with $\pi_{d_v} = \mathrm{head}(f_v^{d_v})$, the draft has a new unified scoring over its original vs. reconsidered children,

$$\pi_{\mathrm{mix}}(t \mid v) \;=\; \pi_{\mathrm{one}}(t \mid v) \;+\; \underbrace{q(d_v)\,\big(\pi_{d_v}(t \mid v) - \pi_{\mathrm{one}}(t \mid v)\big)}_{\text{what reconsideration adds}}, \tag{9}$$

where the gate's own $q(d_v)$ sets how far the reconsidered distribution is trusted over the one-pass one, mimicking the weighting the training objective of Eq. (6) optimizes. Reconsideration thus corrects the probability of an edge, upward where a further pass favors the token and downward where it does not; since $s(\cdot)$ sums edge log-probabilities along the path, the change $\log \pi_{\mathrm{mix}} - \log \pi_{\mathrm{one}}$ on an edge is added to the score of every node beneath it. A node the bid never selects has no π_{d_v}, so Eq. (9) loses its second term and returns π_{one}. The global cut then ranks reconsidered and one-pass nodes alike, and the target verifies the final tree. Appendix C gives the full discussion on serving algorithm of Parallel Reconsideration.

6 Experiments

We describe the setup (Section 6.1), then report acceptance length and speedup against EAGLE-3.1 at a matched budget (Section 6.2), the gain under batched serving (Section 6.3), an ablation of each

Table 1: **Acceptance length τ and decoding speedup over autoregressive decoding.** Parenthesized values give the total draft budget B (number of draft nodes verified per round), held equal for both methods along with the offer width, grow width, and number of draft steps. Δ rows give the relative gain of PR over EAGLE-3.1 at the same budget.

		Math						**Code**						**Chat**			
		GSM8K		MATH-500		AIME25		HumanEval		MBPP		LCB		MT-Bench		**Avg.**	
Model	**Method**	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup
Temperature $= 0$																	
Llama-3.1-8B	EAGLE-3.1 (24)	4.41	1.90×	4.70	2.14×	**5.41**	**2.29**×	5.15	2.25×	4.97	2.24×	4.31	1.78×	4.20	1.69×	4.74	2.04×
	PR (24)	**4.73**	**2.03**×	**4.77**	**2.16**×	5.36	2.28×	**5.45**	**2.38**×	**5.28**	**2.37**×	**4.78**	**1.95**×	**4.65**	**1.85**×	**5.00**	**2.15**×
	Δ	+7.3%	+6.8%	+1.5%	+0.9%	−0.9%	−0.4%	+5.8%	+5.8%	+6.2%	+5.8%	+10.9%	+9.6%	+10.7%	+9.5%	**+5.5%**	**+5.4%**
	EAGLE-3.1 (48)	4.84	2.05×	4.99	2.13×	5.28	2.27×	5.62	2.49×	5.32	2.33×	4.70	1.95×	4.60	2.03×	5.05	2.18×
	PR (48)	**5.40**	**2.26**×	**5.31**	**2.23**×	**5.53**	**2.37**×	**5.87**	**2.56**×	**5.57**	**2.41**×	**5.11**	**2.11**×	**4.77**	**2.05**×	**5.37**	**2.28**×
	Δ	+11.6%	+10.2%	+6.4%	+4.7%	+4.7%	+4.4%	+4.4%	+2.8%	+4.7%	+3.4%	+8.7%	+8.2%	+3.7%	+1.0%	**+6.3%**	**+4.6%**
Qwen3-8B	EAGLE-3.1 (24)	4.39	1.94×	3.96	2.04×	3.81	1.86×	4.46	2.11×	4.34	2.11×	3.66	1.78×	3.36	1.57×	4.00	1.92×
	PR (24)	**4.83**	**2.11**×	**4.26**	**2.16**×	**4.20**	**2.00**×	**4.79**	**2.26**×	**4.84**	**2.32**×	**3.99**	**1.91**×	**3.59**	**1.66**×	**4.36**	**2.06**×
	Δ	+10.0%	+8.8%	+7.6%	+5.9%	+10.2%	+7.5%	+7.4%	+7.1%	+11.5%	+10.0%	+9.0%	+7.3%	+6.8%	+5.7%	**+9.0%**	**+7.3%**
	EAGLE-3.1 (48)	4.84	2.18×	4.30	2.24×	4.23	1.99×	4.85	2.41×	4.75	2.22×	4.00	1.86×	3.66	1.71×	4.38	2.09×
	PR (48)	**5.13**	**2.29**×	**4.50**	**2.32**×	**4.60**	**2.10**×	**5.03**	**2.50**×	**5.07**	**2.33**×	**4.18**	**1.93**×	**3.82**	**1.77**×	**4.62**	**2.18**×
	Δ	+6.0%	+5.0%	+4.7%	+3.6%	+8.7%	+5.5%	+3.7%	+3.7%	+6.7%	+5.0%	+4.5%	+3.8%	+4.4%	+3.5%	**+5.5%**	**+4.3%**
Gemma-3-12B	EAGLE-3.1 (24)	4.92	3.08×	4.21	2.54×	4.29	2.68×	6.37	3.40×	5.29	3.08×	5.28	2.78×	3.25	1.79×	4.80	2.76×
	PR (24)	**5.28**	**3.16**×	**4.65**	**2.78**×	**4.60**	**2.86**×	**7.07**	**3.78**×	**5.65**	**3.30**×	**5.90**	**3.05**×	**3.75**	**1.98**×	**5.27**	**2.99**×
	Δ	+7.3%	+2.6%	+10.5%	+9.4%	+7.2%	+6.7%	+11.0%	+11.2%	+6.8%	+7.1%	+11.7%	+9.7%	+15.4%	+10.6%	**+9.8%**	**+8.3%**
	EAGLE-3.1 (48)	5.43	3.23×	4.53	2.60×	4.67	2.93×	6.73	3.68×	5.68	3.20×	5.81	3.27×	3.58	1.94×	5.20	2.98×
	PR (48)	**5.96**	**3.53**×	**4.89**	**2.80**×	**4.93**	**3.07**×	**7.10**	**3.84**×	**6.08**	**3.33**×	**6.46**	**3.60**×	**3.94**	**2.09**×	**5.62**	**3.18**×
	Δ	+9.8%	+9.3%	+7.9%	+7.7%	+5.6%	+4.8%	+5.5%	+4.3%	+7.0%	+4.1%	+11.2%	+10.1%	+10.1%	+7.7%	**+8.1%**	**+6.7%**
Temperature $= 1$																	
Llama-3.1-8B	EAGLE-3.1 (24)	3.47	1.45×	4.13	2.02×	4.49	2.06×	4.61	2.16×	4.40	1.99×	2.82	1.21×	2.07	1.17×	3.71	1.72×
	PR (24)	**4.01**	**1.61**×	**4.45**	**2.12**×	**4.51**	**2.08**×	**4.80**	**2.18**×	**4.55**	**2.03**×	**3.15**	**1.32**×	**2.29**	**1.24**×	**3.97**	**1.80**×
	Δ	+15.6%	+11.0%	+7.7%	+5.0%	+0.4%	+1.0%	+4.1%	+0.9%	+3.4%	+2.0%	+11.7%	+9.1%	+10.6%	+6.0%	**+7.0%**	**+4.7%**
	EAGLE-3.1 (48)	3.99	1.67×	4.44	2.06×	4.51	2.13×	5.13	2.31×	4.79	2.16×	3.02	1.29×	2.72	1.45×	4.09	1.87×
	PR (48)	**4.56**	**1.84**×	**4.79**	**2.19**×	**4.58**	**2.16**×	**5.42**	**2.36**×	**5.08**	**2.23**×	**3.49**	**1.47**×	**2.83**	**1.49**×	**4.39**	**1.96**×
	Δ	+14.3%	+10.2%	+7.9%	+6.3%	+1.6%	+1.4%	+5.7%	+2.2%	+6.1%	+3.2%	+15.6%	+14.0%	+4.0%	+2.8%	**+7.3%**	**+4.8%**
Qwen3-8B	EAGLE-3.1 (24)	4.47	2.17×	3.62	1.74×	3.25	1.52×	4.32	1.94×	4.08	1.78×	3.38	1.52×	3.12	1.47×	3.75	1.73×
	PR (24)	**4.76**	**2.27**×	**3.90**	**1.84**×	**3.43**	**1.57**×	**4.50**	**1.96**×	**4.30**	**1.84**×	**3.53**	**1.57**×	**3.25**	**1.50**×	**3.95**	**1.79**×
	Δ	+6.5%	+4.6%	+7.7%	+5.7%	+5.5%	+3.3%	+4.2%	+1.0%	+5.4%	+3.4%	+4.4%	+3.3%	+4.2%	+2.0%	**+5.3%**	**+3.5%**
	EAGLE-3.1 (48)	4.84	2.20×	3.94	1.94×	3.59	1.70×	4.73	2.26×	4.48	2.15×	3.67	1.70×	3.37	1.57×	4.09	1.93×
	PR (48)	**5.21**	**2.36**×	**4.19**	**2.00**×	**3.70**	**1.72**×	**4.95**	**2.40**×	**4.69**	**2.24**×	**3.92**	**1.78**×	**3.63**	**1.65**×	**4.33**	**2.02**×
	Δ	+7.6%	+7.3%	+6.3%	+3.1%	+3.1%	+1.2%	+4.7%	+6.2%	+4.7%	+4.2%	+6.8%	+4.7%	+7.7%	+5.1%	**+5.9%**	**+4.7%**
Gemma-3-12B	EAGLE-3.1 (24)	5.00	2.79×	4.12	2.41×	3.79	2.00×	6.41	3.33×	5.44	3.17×	5.17	3.17×	3.12	1.48×	4.72	2.62×
	PR (24)	**5.38**	**2.90**×	**4.38**	**2.54**×	**4.29**	**2.24**×	**7.51**	**3.86**×	**5.83**	**3.32**×	**5.59**	**3.41**×	**3.28**	**1.54**×	**5.18**	**2.83**×
	Δ	+7.6%	+3.9%	+6.3%	+5.4%	+13.2%	+12.0%	+17.2%	+15.9%	+7.2%	+4.7%	+8.1%	+7.6%	+5.1%	+4.1%	**+9.7%**	**+8.0%**
	EAGLE-3.1 (48)	5.44	2.81×	4.47	2.36×	4.35	2.14×	6.70	3.89×	5.83	2.98×	5.54	3.13×	3.40	1.64×	5.10	2.71×
	PR (48)	**6.22**	**3.19**×	**4.82**	**2.53**×	**4.86**	**2.33**×	**7.25**	**4.20**×	**6.66**	**3.39**×	**5.99**	**3.37**×	**3.55**	**1.70**×	**5.62**	**2.96**×
	Δ	+14.3%	+13.5%	+7.8%	+7.2%	+11.7%	+8.9%	+8.2%	+8.0%	+14.2%	+13.8%	+8.1%	+7.7%	+4.4%	+3.7%	**+10.2%**	**+9.2%**

design choice and of a larger drafter (Section 6.4), and a comparison against spending the same batch on a wider draft (Section 6.5). Appendix D reports more experimental details and studies.

6.1 Experimental setup

We compare **Parallel Reconsideration** against **EAGLE-3.1**, the state-of-the-art feature-level speculative decoder, on three target models, **Llama-3.1-8B-Instruct**, **Qwen3-8B**, and **Gemma-3-12B**, and across seven benchmarks spanning math (GSM8K, MATH-500, AIME25), code (HumanEval, MBPP, LiveCodeBench), and chat (MT-Bench). Both drafters use a single decoder layer and share an identical target model, both are trained in SpecForge (Li et al., 2026) under EAGLE-3's released recipe, and both are served through SGLang (Zheng et al., 2024) under the same budget configuration, so every number we report is measured end to end on a production serving engine. Because SGLang already removes the system bottlenecks that an under-optimized autoregressive baseline suffers from, the speedups we measure are lower than those reported in the original EAGLE-3 paper. As an external check on the baseline, our EAGLE-3.1 acceptance lengths agree with those reported independently for the same models and benchmarks by Chen et al. (2026). Our primary metrics are the *acceptance length* τ and wall-clock speedup over autoregressive decoding on one H200. Appendix D.1 gives the models, benchmarks, training and serving details, and the budget configuration in full.

6.2 Acceptance length and speedup

Table 1 reports acceptance length and speedup across all seven benchmarks. Parallel Reconsideration improves average acceptance length over EAGLE-3.1 at every budget, on all three models, and at both temperatures, by **5.3–10.2%**, and the gain carries through to wall-clock speedup (**3.5–9.2%**) at a nearly identical per-round drafting cost (Appendix H). The gain is consistent across benchmarks from different domains: LiveCodeBench reaches **+10.1%** speedup on Gemma-3, and **+10.2%** speedup on GSM8K for Llama3.1.

6.3 Serving at larger batch sizes

Table 3: **Removing each design choice, and spending the compute on depth instead.** Llama-3.1-8B at $B = 48$, temperature 0, reporting τ / speedup. The first two rows remove one component of Parallel Reconsideration and retrain; the third is EAGLE-3.1 with a second decoder layer and no reconsideration, spending its additional capacity on every position rather than on the uncertain ones. Δ is relative to Parallel Reconsideration.

	GSM8K		LiveCodeBench		MBPP		**Avg.**	
Drafter	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup
EAGLE-3.1	4.84	2.05×	4.70	1.95×	5.32	2.33×	4.95	2.11×
Parallel Reconsideration	**5.40**	**2.26**×	**5.11**	**2.11**×	**5.57**	**2.41**×	**5.36**	**2.26**×
no balance term ($\beta = 0$)	5.17	2.14×	5.03	2.04×	5.58	2.41×	5.26	2.20×
Δ	−4.3%	−5.3%	−1.6%	−3.3%	+0.2%	+0.0%	−1.9%	−2.8%
no self-query ($Q \to e(t_p)$)	5.03	2.10×	4.92	2.02×	5.47	2.35×	5.14	2.16×
Δ	−6.9%	−7.1%	−3.7%	−4.3%	−1.8%	−2.5%	−4.1%	−4.6%
EAGLE-3.1, two decoder layers	4.87	1.90×	4.91	1.85×	5.37	2.14×	5.05	1.96×
Δ	−9.8%	**−15.9%**	−3.9%	**−12.3%**	−3.6%	**−11.2%**	−5.8%	**−13.1%**

Speculative decoding is hardest to justify at large batch sizes, where the target's forward is already compute-bound and every drafted node competes for the same verification capacity. We therefore benchmark Parallel Reconsideration at batch sizes $n \in \{1, 2, 4, 8, 16\}$ against autoregressive decoding at the same n (Table 2).

Gemma-3-12B holds **2.85**× on LiveCodeBench at $n = 8$ and **2.14**× at $n = 16$, and Qwen3-8B is flat through $n = 4$ (**2.25**× on GSM8K, **1.95**× on LiveCodeBench). Reconsideration buys acceptance out of the draft budget rather than the verification budget (Section 5), so the target still verifies the same B nodes per round at every n, and the co-forwarded rows are absorbed by the drafter's own batch rather than the target's.

Table 2: **Speedup as requests are batched.** Wall-clock speedup of EAGLE-3.1 and Parallel Reconsideration (PR) over autoregressive decoding at a matched batch size n, measured on the same serving engine ($B = 48$, temperature 0).

			Batch size n				
			1	**2**	**4**	**8**	**16**
Llama	GSM8K	EAGLE-3.1	2.05×	1.68×	1.62×	1.26×	1.12×
		PR	2.26×	1.91×	1.91×	1.46×	1.22×
	MT-Bench	EAGLE-3.1	2.03×	1.71×	1.71×	1.50×	1.41×
		PR	2.05×	1.79×	1.77×	1.56×	1.51×
	LCB	EAGLE-3.1	1.95×	1.61×	1.60×	1.40×	1.31×
		PR	2.11×	1.84×	1.78×	1.33×	1.29×
Qwen	GSM8K	EAGLE-3.1	2.18×	2.16×	2.01×	1.68×	1.39×
		PR	2.29×	2.26×	2.25×	1.94×	1.64×
	MT-Bench	EAGLE-3.1	1.71×	1.72×	1.71×	1.35×	1.11×
		PR	1.77×	1.77×	1.69×	1.41×	1.24×
	LCB	EAGLE-3.1	1.86×	1.81×	1.80×	1.60×	1.31×
		PR	1.93×	1.87×	1.95×	1.66×	1.43×
Gemma	GSM8K	EAGLE-3.1	3.23×	2.79×	2.54×	2.50×	1.71×
		PR	3.53×	2.99×	2.72×	2.70×	1.94×
	MT-Bench	EAGLE-3.1	1.94×	1.92×	1.83×	1.71×	1.50×
		PR	2.09×	2.02×	1.91×	1.91×	1.40×
	LCB	EAGLE-3.1	3.27×	2.79×	2.72×	2.55×	2.04×
		PR	3.60×	3.02×	2.97×	2.85×	2.14×

6.4 Each component, and the alternative of a bigger drafter

Table 3 removes one design choice at a time and retrains. It also answers a question the cost model raises: if reconsideration is worth a second pass of M, why not simply give the drafter a second layer and spend that capacity on every position?

The self-query. Replacing $Q(f_p^0)$ with $e(t_p)$, the embedding of the token that produced f_p^0, costs 4.1% of average acceptance and 4.6% of speedup. Steering reconsideration with a committed token rather than the feature's own query recovers less, which is what Section 3.2 predicts: a token embedding directs M to advance along that branch, while reconsideration needs to stay in place.

The balance term. Setting $\beta = 0$ costs 1.9% of average acceptance. The term keeps a depth from collapse before the head's term can train it (Appendix I), so removing it degrades the drafter without disabling it.

A bigger drafter instead. Giving EAGLE-3.1 a second decoder layer raises its acceptance, by 2.0% on average over the one-layer EAGLE baseline, and still *loses* 7.0% of its speedup, because the extra layer raises t_{step} on every draft step. Against Parallel Reconsideration the gap is **13.1%** of speedup at 5.8% lower acceptance.

6.5 Reconsidering against drafting wider

A draft step of Parallel Reconsideration batches $B_w + m$ nodes where EAGLE-3.1 batches B_w. Table 4 grants EAGLE-3.1 the same $m = 3$ extra rows for one-pass breadth, either advancing 3 more

Table 4: **Reconsidering the additional nodes against drafting wider with them.** Gemma-3-12B at $B = 48$, temperature 1, reporting τ / speedup. Both EAGLE-3.1 controls and PR batch 9 nodes per draft step; PR with $m = 0$ batches 6. Δ is relative to EAGLE-3.1. Controls are detailed in Appendix D.2.

	GSM8K		LiveCodeBench		MT-Bench		**Avg.**	
Drafter	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup
EAGLE-3.1 ($B_w = 6$)	5.44	2.81×	5.54	3.13×	3.40	1.64×	4.79	2.53×
+ advance 3 more	5.51	2.83×	5.68	3.17×	3.43	1.64×	4.87	2.55×
Δ	+1.3%	+0.7%	+2.5%	+1.3%	+0.9%	+0.0%	+1.7%	+0.8%
widen to $B_w = 9$	5.49	2.82×	5.61	3.15×	3.42	1.64×	4.84	2.54×
Δ	+0.9%	+0.4%	+1.3%	+0.6%	+0.6%	+0.0%	+1.0%	+0.4%
PR without reconsideration ($m = 0$)	5.44	2.81×	5.55	3.13×	3.38	1.63×	4.79	2.52×
Δ	+0.0%	+0.0%	+0.2%	+0.0%	−0.6%	−0.6%	−0.1%	−0.4%
+ reconsider 3 (PR)	**6.22**	**3.19×**	**5.99**	**3.37×**	**3.55**	**1.70×**	**5.25**	**2.75×**
Δ	**+14.3%**	**+13.5%**	**+8.1%**	**+7.7%**	**+4.4%**	**+3.7%**	**+9.6%**	**+8.7%**

nodes from the previous step's batch or widening to $B_w = 9$. These lift average acceptance by only 1.7% and 1.0%, against **9.6%** for reconsidering, while PR with $m = 0$ matches EAGLE-3.1 within noise: the extra rows pay only when spent on a second look.

7 Related Work

Speculative decoding. Speculative decoding accelerates autoregressive generation by drafting several tokens cheaply and verifying them in one target forward pass, accepting the longest prefix the target agrees with so the output distribution is preserved exactly (Leviathan et al., 2023; Xia et al., 2023); since the acceptance rule guarantees that distribution, subsequent work buys speed either by raising the *acceptance length* or by lowering the *drafting cost*.

Drafters. Self-drafting methods derive the draft from the target's own feature, as in the EAGLE family, which autoregresses on target features with a single lightweight decoder layer (Li et al., 2024a;b; 2025; Eldenk et al., 2026), or in the prediction heads attached to the target's last hidden state (Cai et al., 2024; Ankner et al., 2024). Training-free drafters remove the auxiliary model altogether and verify n-gram continuations generated in parallel (Fu et al., 2024; Le et al., 2026), while parallel drafters attack the cost axis directly by emitting an entire block per forward pass (An et al., 2025; Chen et al., 2026; Cheng et al., 2026). A separate line changes what the drafter optimizes, recasting draft training as variational inference over draft paths so that acceptance, rather than token likelihood, is the objective (Zou et al., 2026). Parallel Reconsideration is orthogonal to most of these: it adds a second guess at the uncertain positions of an existing one-pass draft, and applies to any drafter that exposes a reusable draft feature.

Extra passes. Spending extra passes to sharpen a prediction is itself familiar, from looped and universal transformers to adaptive-compute methods that learn how many steps each token needs (Giannou et al., 2023; Dehghani et al., 2019; Banino et al., 2021; Schuster et al., 2022; Zhu et al., 2025); we loop not to make an autoregressive model reason better, but to revise the failure of autoregressive *drafting*, and we run the extra pass in parallel so it costs no additional step. Appendix E provides a more detailed discussion of related work.

8 Conclusion

We presented **Parallel Reconsideration**, which grants the drafter a *second chance to guess* on the tokens it is unsure about: where it is already decisive the guess is committed as usual, and where it hesitates the drafter is given a few extra attempts to revise it. The attempts are batched into the same forward pass as the ordinary draft, so they add no sequential draft step and cost only negligible parallel overhead. This brings *draft-time scaling* to feature-level drafters, letting the drafter vary how much it computes per position rather than how well it computes in one step. Across Llama-3.1-8B-Instruct, Qwen3-8B, and Gemma-3-12B, it raises average acceptance length over EAGLE-3.1 by **5.3–10.2%** and wall-clock speedup by **3.5–9.2%** on math, code, and chat benchmarks at a matched draft budget.

AI USE STATEMENT

Generative AI tools were used only to polish the writing of this paper; the authors are responsible for all content.

ETHICS STATEMENT

This work studies lossless acceleration of large language model inference and does not involve human subjects, personal data, or new datasets. All benchmarks used are publicly released.

REPRODUCIBILITY STATEMENT

We have made every effort to describe our method, training, and evaluation in enough detail to be reproduced: the method is specified in Sections 3–5, with the full training recipe in Appendix B and the serving algorithm in Appendix C (Algorithm 1), and the experimental setup, hyperparameters, and implementation details are given in Appendices D.1 and F. We are unable to release code at submission time because it is pending our organization's legal approval; we will release the code upon approval should the paper be made public.

REFERENCES

Zihao An, Huajun Bai, Ziqiong Liu, Dong Li, and Emad Barsoum. PARD: Accelerating LLM inference with low-cost PARallel draft model adaptation. *arXiv preprint arXiv:2504.18583*, 2025.

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for Medusa decoding. In *Conference on Language Modeling (COLM)*, 2024.

Anthropic. Claude code. `https://claude.com/product/claude-code`, 2025.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. In *ICML Workshop on Automated Machine Learning (AutoML)*, 2021.

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 (ICML)*. PMLR, 2024.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *International Conference on Machine Learning (ICML)*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL `https://arxiv.org/abs/2107.03374`.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. In Amir Globersons, Lester

Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/ea1f5f0878d43ff4fb8bf64ef4a2326c-Abstract-Conference.html.

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, Qinyu Chen, Chengqi Deng, Jiping Yu, Damai Dai, Zhengyan Zhang, Yixuan Wei, Yixuan Tan, Wenkai Yang, Runxin Xu, Yu Wu, Zhean Xu, Xuanyu Wang, Muyang Chen, Rui Tian, Xiao Bi, Zhewen Hao, Shaoyuan Chen, Huanqi Cao, Wentao Zhang, Anyi Xu, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. DSpark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *CoRR*, abs/2607.05147, 2026. doi: 10.48550/ARXIV.2607.05147. URL https://doi.org/10.48550/arXiv.2607.05147.

Yunfei Cheng, Aonan Zhang, Xuanyu Zhang, Chong Wang, and Yi Wang. Recurrent drafter for fast speculative decoding in large language models. *arXiv preprint arXiv:2403.09919*, 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.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations (ICLR)*, 2019.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Doğaç Eldenk, Payal Mohapatra, Yigitcan Comlek, Kaan Oktay, Hongyang Zhang, and Stephen Xia. Attention drift: What autoregressive speculative decoding models learn. *arXiv preprint arXiv:2605.09992*, 2026.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of LLM inference using lookahead decoding. In *International Conference on Machine Learning (ICML)*. PMLR, 2024.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning (ICML)*. PMLR, 2023.

Google. Google antigravity. https://antigravity.google/, 2025.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations (ICLR)*, 2025.

Hoang Anh Duy Le, Shaochen Zhong, Yifan Lu, Yingtong Dou, Jiayi Yuan, Yu-Neng Chuang, Xiran Fan, Guanchu Wang, Yuzhong Chen, and Xia Hu. FAFO: Lossy KV cache compression for lossless inference acceleration via draftless fumble decoding. In *International Conference on Machine Learning (ICML)*. PMLR, 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.

Shenggui Li, Chao Wang, Yikai Zhu, Yubo Wang, Fan Yin, Shuai Shi, Yefei Chen, Xiaomin Dong, Qiaoling Chen, Jin Pan, Ji Li, Laixin Xie, Yineng Zhang, Lei Yu, Yonggang Wen, Ivor Tsang, and Tianwei Zhang. SpecForge: A flexible and efficient open-source training framework for speculative decoding. *CoRR*, abs/2603.18567, 2026. doi: 10.48550/ARXIV.2603.18567. URL https://doi.org/10.48550/arXiv.2603.18567.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning (ICML)*, 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 (EMNLP)*, 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Duyu Tang, Kai Han, and Yunhe Wang. Kangaroo: Lossless self-speculative decoding for accelerating LLMs via double early exiting. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/16336d94a5ffca8de019087ab7fe403f-Abstract-Conference.html.

Moonshot AI. Kimi K3. https://www.kimi.ai/blog/kimi-k3, 2026.

OpenAI. Codex. https://openai.com/codex/, 2025.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In Dawn Song, Michael Carbin, and Tianqi Chen (eds.), *Proceedings of the Sixth Conference on Machine Learning and Systems, MLSys 2023, Miami, FL, USA, June 4-8, 2023*. mlsys.org, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/c4be71ab8d24cdfb45e3d06dbfca2780-Abstract-mlsys2023.html.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. SpecExec: Massively parallel speculative decoding for interactive LLM inference on consumer devices. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3909–3925, 2023.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning harmonized representations for speculative sampling. In *International Conference on Learning Representations (ICLR)*, 2025.

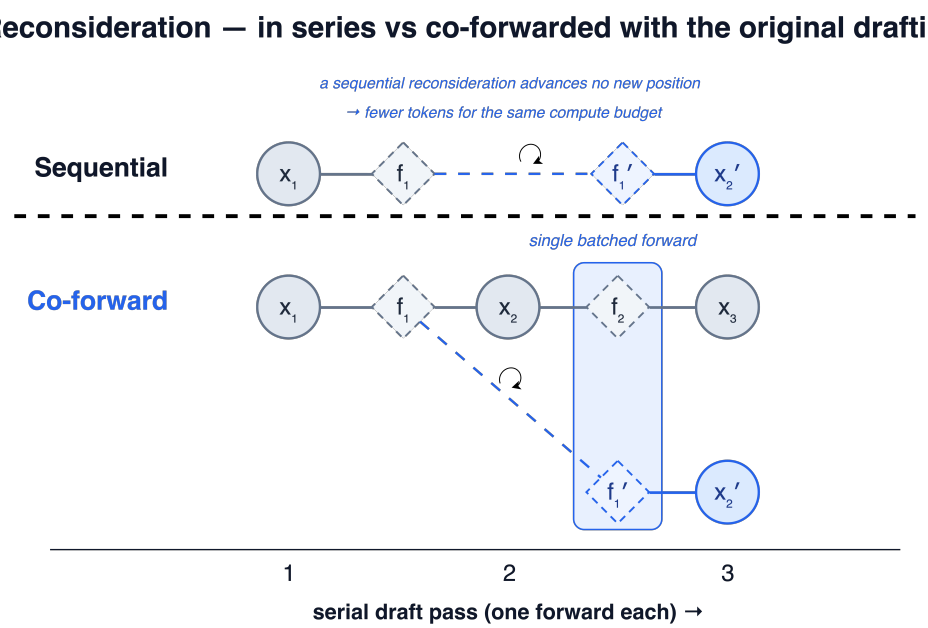


Figure 3: **Reconsidering in parallel.** Diamonds are features produced by a forward pass of the shared block M and circles are the tokens sampled from them; gray marks ordinary one-pass drafting (feature f_i, token x_i) and blue marks reconsideration (deepened feature f_i', reconsidered token x_i'). *Top (sequential):* running reconsideration in series spends a forward of M that advances no new position, so within a fixed draft budget the drafter reaches fewer tokens. *Bottom (co-forward):* reconsideration of a position depends only on its one-pass feature, which is already available, so it is co-forwarded with the ordinary draft step in a single batched pass of M. The one-pass chain still reaches x_3 and the reconsidered token x_2' is obtained for negligible additional latency.

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 (NeurIPS)*, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models. *CoRR*, abs/2510.25741, 2025. doi: 10.48550/ARXIV.2510.25741. URL `https://doi.org/10.48550/arXiv.2510.25741`.

Xiandong Zou, Jianshu Li, Jing Huang, and Pan Zhou. Variational speculative decoding: Rethinking draft training from token likelihood to sequence acceptance. In *International Conference on Machine Learning (ICML)*, 2026.

A Details on Parallel Reconsideration

A.1 Reconsidering in parallel

Reconsideration adds forward passes, and naively those passes are sequential. Running the second pass in series stalls the draft at p until f_p^1 is computed, so a draft of γ positions with N reconsidered

ones takes $(\gamma + N)\, t_{\text{step}}$ instead of $\gamma\, t_{\text{step}}$ (Eq. (1), Figure 3, top). Under a fixed drafting-time budget the drafter reaches N fewer positions, shortening the original total number of guesses.

The stall, however, is an artifact of ordering, not of dependency. Writing the draft step and its reconsideration side by side (Eq. (5)), both take the same feature f_p^0 as input, and their outputs go to separate places: f_{p+1}^0 advances the ongoing draft to $p{+}1$, whereas f_p^1 only re-samples $\tilde{t}_{p+1}$ and later seeds the reconsidered branch's own continuation, which the ongoing draft never reads. Concretely, the reconsideration row keeps position index p and attends to the committed prefix $\kappa_{<p}$ and its own key, not to $\mathrm{kv}(f_p^0)$; its key/value replaces slot p on the reconsidered branch only. The reconsideration is therefore independent of the draft step that advances past it, so the two can be batched together into a single forward of M — *co-forwarding* rather than serializing (Figure 3, bottom).

At the small batch sizes of latency-oriented decoding a forward of M is memory-bandwidth bound: its latency is set by streaming the drafter's weights from HBM, and the extra arithmetic from enlarging the batch is hidden beneath that load. A co-forwarded pass therefore costs $t_{\text{step}} + \epsilon$ for a negligible ϵ, so a draft of γ positions costs $\gamma\,(t_{\text{step}} + \epsilon) \approx \gamma\, t_{\text{step}}$, against $(\gamma + N)\, t_{\text{step}}$ for the serial version. The drafter reaches exactly as many positions as it would without reconsideration; we detail the batched execution in Section 5. Measured end to end, carrying the reconsideration rows costs 1.015–$1.022\times$ the per-cycle time of the one-pass drafter, i.e., roughly 2% of cycle time (Appendix H for details).

The result is an *interleaved* drafting strategy: the ordinary steps form the main branches of one-pass guesses, and each gated position might add a reconsidered candidate $\tilde{t}_{p+1}$ (Eq. (3)) beside its one-pass sibling. Verification scores both, so at that position the reconsidered token takes nothing from its one-pass sibling: it is an extra candidate rather than a substitute, and it can supply an accepted token only where the one-pass guess would have been rejected. The two share the verification budget, however, and Section 5 is where they are ranked against each other for it.

B Details on Training Parallel Reconsideration

This appendix gives a more detailed discussion on the training recipe of Parallel Reconsideration. Section 4 states the objective of Eq. (6) and the mixed cache of Eq. (7); we expand here on the two failure modes the recipe is designed against (Appendix B.1), on what a reconsidered feature is actually supervised to do (Appendix B.2), on the rollouts the losses are computed over, and on why the balance term regularizes the gate's marginal usage rather than its per-position halt distribution (Appendix B.3).

B.1 Why the one-pass head cannot be scaffolding

Loop transformers, which reuse a layer across depths much as we do, read out only at a single depth and are free to supervise the intermediate depths purely as scaffolding for that one output (Appendix E). Parallel Reconsideration cannot do this, because at serving *both* depths are read out and committed to the tree: the one-pass head guesses the token at most positions and the reconsidered head revises it at the uncertain ones. The difficulty is that the two share the feature f_p^0. On one hand, f_p^0 is the feature every later draft step conditions on, so it must retain enough of the target distribution to keep proposing acceptable tokens down the whole draft; on the other hand, reconsideration deepens f_p^0 into f_p^1, and for the reconsidered token to differ from the one-pass token the objective must let f_p^1 diverge from f_p^0 rather than reproduce it.

This is what makes the two failure modes of Section 4 opposite rather than two versions of the same error. Supervising only the reconsidered head leaves the one-pass head untrained, so the draft it proposes degrades and shortens, and since a reconsidered candidate is only reached once the tokens before it are accepted, the reconsidered branches are left with shorter and shorter prefixes to extend. Supervising both to match the target with equal weight instead removes any incentive for the two features to differ, and gradient descent finds the trivial optimum $f_p^1 \approx f_p^0$. Appendix J discusses both in more detail.

The recipe therefore holds the one-pass draft to ordinary EAGLE quality with an unconditional loss,

$$\mathcal{L}_p^{\text{one}} = \mathrm{CE}\big(\mathrm{head}(f_p^0),\, t_{p+1}^{\star}\big)\,, \tag{10}$$

against the teacher token $t^\star_{p+1}$, which appears as the unweighted first term of Eq. (6). Supervising it at every position frees reconsideration to specialize on the positions the one-pass guess hesitates over, without eroding the ones it already resolves.

B.2 What a reconsidered feature is supervised to do

Each reconsideration pass reapplies M to the previous depth's feature through the self-query, $f_p^r = M(\mathrm{fc}(f_p^{r-1}, Q(f_p^{r-1})))$ (Eq. (3)), so depth r is a strictly deeper reading of the same position p. Such a feature does two jobs at serving, and the two are supervised differently. *Every* depth is read out to propose a candidate token, so we supervise the head at all depths. Only *one* depth, the halt depth d_p where the gate stops, has its key/value written to the cache and drafts the next position, so the continuation is supervised at that depth alone. Which depth halts is therefore central: it decides what the deeper positions read and what the recurrence is trained on. We sample it, $d_p \sim q$ (Eq. (4)), by walking the depths and halting at the first pass the gate declines, the *same* way in training and at serving.

The mixed cache, concretely. Section 4 introduces the mixed cache of Eq. (7). Training naively on a fixed one-pass cache, reading f_j^0 at every ancestor, would leave a gap: that cache never appears at serving, where each ancestor commits at its own halt depth. For a position 5 whose ancestors halt at depths $d_{1:4} = (0, 1, 0, 2)$,

$$\begin{aligned} \text{one-pass cache:} \quad & \big(\mathrm{kv}(f_1^0),\, \mathrm{kv}(f_2^0),\, \mathrm{kv}(f_3^0),\, \mathrm{kv}(f_4^0)\big), \\ \text{mixed cache } \kappa_{<5}: \quad & \big(\mathrm{kv}(f_1^0),\, \mathrm{kv}(f_2^1),\, \mathrm{kv}(f_3^0),\, \mathrm{kv}(f_4^2)\big), \end{aligned} \tag{11}$$

the two agree at the confident ancestors $1, 3$ (halt depth 0) but differ at the uncertain $2, 4$, where the mixed cache carries the reconsidered key/value the drafter will actually commit. Sampling $d_j \sim q$ deepens the ancestors of a single rollout faithfully under the halting distribution, while a served branch under a budget deepens few; both are prefixes the same per-position commit rule produces, so a reconsidered branch never reads one the drafter was not trained for.

On the halt depths seen at training and at serving. The depths q favors at training are not the depths the drafter meets at serving. Training samples $d_p \sim q$ at every position, so any position may deepen; at serving the reconsideration budget m admits only a few eligible nodes per step, so most positions are committed from depth 0 however willing q is to go deeper. The weighting does not turn this into a train–serve gap, because it reallocates supervision rather than withholding it. Every depth is read out and scored at *every* position, and only the size of its gradient varies with $q(r)$, so no depth is trained solely on the positions q favors and each stays accurate wherever it is eventually read. A position served from depth 0 under a tight budget is one the one-pass term trains unconditionally; a position the budget does deepen is one the gated terms have trained at that depth. The same holds for the cache those losses are computed over, by the argument above.

Two on-policy rollouts. EAGLE-3 (Li et al., 2025) trains by unrolling an on-policy draft chain from every position of the training sequence: starting from a position's target feature, the drafter speculates several steps forward on its own predictions, and each drafted position is scored against the teacher token. We unroll two such chains from every starting position. The *one-pass chain* is EAGLE-3's own, advancing from each one-pass feature to the next, $f_p^0 \to f_{p+1}^0$. The *reconsideration chain* branches off at the starting position, then runs on its own features: at each of its positions it deepens the current feature to the halt feature $f_p^{d_p}$ and drafts the next position from it,

$$f_{p+1}^0 = M\big(\mathrm{fc}\big(f_p^{d_p},\, e(t_{p+1})\big) \;\big|\; \kappa_{<p+1}\big)\,, \tag{12}$$

reading the mixed cache assembled above and reconsidering afresh from there.

Both chains are scored, position by position, by the head losses of Eq. (6), so each learns its own predictions on its own rollout. They share parameters through M but couple in the backward pass only at the branch point f_p^0, which feeds both the one-pass step and the reconsideration chain. Its gradient is therefore the sum of two signals: the one-pass head loss at f_p^0 and the losses of the entire reconsideration rollout it seeds. Thus f_p^0 is optimized jointly to minimize the one-pass token error and to encode a state from which reconsideration can recover the tokens that error misses. The continuation of Eq. (12) needs no term of its own: keeping each chain accurate at every position already trains its features to draft the next one.

B.3 WHICH ENTROPY THE BALANCE TERM REWARDS

The balance term of Eq. (6) is not the obvious entropy. The halt distribution induces two: the per-position entropy averaged over positions, $\mathbb{E}_p[H(q)]$, and the entropy of the marginal usage $\bar{q} = \mathbb{E}_p[q]$. We reward the marginal $H(\bar{q})$, and the decomposition of Proposition 1 (Appendix K) makes the choice precise.

The point is what each term does *not* do. Rewarding per-position entropy $\mathbb{E}_p[H(q)]$ would push every position to split its mass across depths and actively destroy the specialization we are trying to build. Rewarding the marginal $H(\bar{q})$ leaves that specialization alone: it charges only for letting a depth fall out of use, and is neutral as to whether the surviving mass is realized by confident per-position decisions or by undecided ones. The specialization itself comes from the head term of Eq. (6), which trains each depth on the tokens routed to it; the balance term's job is only to keep a depth from going dark before that term can train it. It is also symmetric in the depths: a gate that always halts at depth 0 and one that always runs to depth R both give $H(\bar{q}) = 0$, so it resists collapse onto any single depth rather than pulling toward deeper ones (Zhu et al., 2025). Appendix I validates both claims empirically.

No price on depth. No term of Eq. (6) charges for reconsidering. With the losses detached, $\sum_r q(r)\,\mathrm{sg}[\ell_p^r]$ is minimized by routing each position to its lowest-loss depth, and nothing opposes going deeper. The objective therefore asks the gate only whether a further pass improves the token, which is what Section 3.3 wants it to judge; how many passes the drafter can afford is set where the resource binds, by the depth cap R and the m reconsideration rows a forward carries (Section 5).

C DETAILS ON SERVING A RECONSIDERED TREE

This appendix gives a more detailed discussion on the serving algorithm of Parallel Reconsideration, summarized in Algorithm 1. We expand on how the reconsideration pool is maintained across draft steps (Appendix C.1), on what the repricing rule of Eq. (9) does to a node's existing children as well as its new ones (Appendix C.2), on why the round degenerates exactly to EAGLE-3.1 when no node is eligible (Appendix C.3), and on why verification, and hence losslessness, is unaffected (Appendix C.4).

C.1 MAINTAINING THE RECONSIDERATION POOL

The gate is consulted once per level, always on the feature the last forward produced, so no evaluation of it costs a pass of its own. At level 0 this is what admits a node to the pool: the step that forwards u reads $\lambda_0 = \sigma(\mathrm{gate}(f_u^0))$ off the one-pass feature it has just produced, and u enters the pool with probability $1 - \lambda_0$, sampled as in training (Section 3.3). Eligibility is therefore settled before any reconsideration pass is spent, which is what lets the pool be filtered for free.

At level $r \geq 1$ the same test decides how much further a node already in the pool goes. Each step the bid selects the top m of the pool (Eq. (8)); those nodes are carried one level deeper in the same batched forward, and the gate then reads λ_{r_v} off the feature $f_v^{r_v}$ that forward produced. A node it halts leaves the pool with its halt depth d_v fixed and reprices its edges; a node it declines to halt has r_v incremented and remains for a later step, so a single position may be reconsidered across several steps up to the cap R, at which it halts regardless. Two consequences follow. First, the bid is re-evaluated every step over a changing pool, so a node that loses the competition early may still be selected once shallower competitors have halted. Second, the m rows a step carries are not tied to any one position: they are spent on whichever m pooled nodes currently bid highest, which is what makes m a per-step budget rather than a per-position one.

C.2 WHAT THE REPRICING RULE COVERS

Once v halts at d_v, Eq. (9) prices *every* edge out of it, not only the token the reconsidered feature newly proposes. The children its one-pass feature offered when it was advanced are repriced by the same rule, and because the path score $s(\cdot)$ is a sum of edge log-probabilities, that correction carries into the subtree those children have already grown: in the algorithm this is the increment $\delta_c = \log \pi_{\mathrm{mix}}(c \mid v) - \log \pi_{\mathrm{one}}(c \mid v)$ added to $s(c)$ and to every node beneath c. A node the bid

Algorithm 1: One draft round of Parallel Reconsideration. A forwarded node joins the reconsideration pool $\mathcal{C}$ only if the gate declines to halt it at depth 0. The bid then selects the m eligible nodes whose reconsideration is most worth the pass, and the gate sets how deep each goes. With $\mathcal{C}$ held empty the procedure is the EAGLE-3.1 loop.

input : budget B; grow width B_w; offer width k; reconsideration budget m; reconsideration cap R; draft steps γ
output : draft tree $\mathcal{T}$, $|\mathcal{T}| \le B$

$\mathcal{T} \leftarrow \{\text{root}\}$; $\mathcal{U} \leftarrow \{\text{root}\}$; $\mathcal{C} \leftarrow \emptyset$ `// tree; un-forwarded nodes; reconsideration pool`
for step $= 1, \dots, \gamma$ **do**
  $\mathcal{A} \leftarrow$ top-B_w of $\mathcal{U}$ by $s(\cdot)$; $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{A}$ $\mathcal{S} \leftarrow$ top-m of $\mathcal{C}$ by $\mathrm{bid}(\cdot)$ `// Eq. (8); the m most promising of the pool`
  $\{f_u^0\}_{u\in\mathcal{A}} \cup \{f_v^{r_v+1}\}_{v\in\mathcal{S}} \leftarrow M(\mathcal{A}\cup\mathcal{S})$ `// one level deeper per selected node`
  foreach $v \in \mathcal{S}$ **do**
    halt with probability $\lambda_{r_v} = \sigma\big(\mathrm{gate}(f_v^{r_v})\big)$, Eq. (4) `// sampled, as in training`
    if *halted or* $r_v = R$ **then**
      $d_v \leftarrow r_v$; remove v from $\mathcal{C}$
    else
      $r_v \leftarrow r_v + 1$; keep v in $\mathcal{C}$ for a later step
  `/* two chains: a reconsidered node adds to what it already offered */`
  foreach $u \in \mathcal{A}$ **do**
    offer the top k tokens of $\pi_{\text{one}}(\cdot \mid u)$ as children of u, scoring each by $s(c) = s(u) + \log \pi_{\text{one}}(c \mid u)$
  foreach v *halted this step* **do**
    `/* v now prices its edges by` π_{mix}`: reprice the children it offered when it was advanced, then admit the ones the halt distribution adds */`
    foreach *existing child* c *of* v **do**
      $\delta_c \leftarrow \log \pi_{\text{mix}}(c \mid v) - \log \pi_{\text{one}}(c \mid v)$, Eq. (9)
      add δ_c to $s(c)$ and to every node beneath c `// s(·) sums along the path`
    foreach t *with* $q(d_v)\big(\pi_{d_v}(t \mid v) - \pi_{\text{one}}(t \mid v)\big) > 0$ *and* t *not already a child of* v **do**
      offer t as a child c of v with $s(c) = s(v) + \log \pi_{\text{mix}}(t \mid v)$
  $\mathcal{U} \leftarrow \mathcal{U} \cup \{\text{children offered this step}\}$; $\mathcal{T} \leftarrow \mathcal{T} \cup \{\text{children offered this step}\}$
  foreach $u \in \mathcal{A}$ **do**
    $r_u \leftarrow 1$; add u to $\mathcal{C}$ with probability $1 - \lambda_0 = 1 - \sigma(\mathrm{gate}(f_u^0))$ `// gate eligibility at depth 0, free from` f_u^0
return top-B of $\mathcal{T}$ by $s(\cdot)$ `// one global cut, one-pass and reconsidered alike`

never selects has no π_{d_v}, so Eq. (9) loses its second term and returns π_{one}, leaving its edges exactly as the one-pass draft priced them.

This is where reconsideration does more than widen the tree. On a token v has not proposed, a positive correction admits a new candidate. On one it already holds, the correction moves that child instead, up where the deeper pass agrees with it and down where it does not, against its siblings and against the rest of the tree at the cut.

C.3 Degenerating to the one-pass baseline

The round reduces to EAGLE-3.1's exactly when the pool is empty. With $\mathcal{C} = \emptyset$ no node is ever selected by the bid, no forward carries reconsideration rows, every node keeps π_{one} by the degenerate case of Eq. (9), and the global cut ranks a tree grown entirely by one-pass steps. This is the sense in which reconsideration is additive at serving: the gate controls whether the second mechanism runs at all, and where it declines, the drafter behaves as the baseline it was built on.

C.4 Verification is unchanged, and the output is lossless

Reconsideration changes *which* tokens enter the draft tree, but not how the tree is verified. The verifier receives only the tree's tokens and its topology. It never receives a draft distribution: neither π_{one}, nor π_{d_v}, nor π_{mix}, nor any path score $s(\cdot)$. These quantities, together with the sampled halting decisions, are used only on the drafter side, to decide what to propose and what survives the global cut. We therefore serve Parallel Reconsideration through SGLang's tree verification for EAGLE, unmodified, the same path used for the EAGLE-3.1 and HASS baselines. At temperature 0 it accepts the longest path that matches the target's greedy tokens. At temperature $T > 0$ it applies target-only multi-candidate rejection sampling, which treats every drafted child as a deterministic proposal.

Losslessness. Fix a verified node with prefix x, and let $\pi = \pi_{\text{tgt}}(\cdot \mid x)$ be the target's distribution at temperature T. Let $c_1, \ldots, c_k$ be the node's children in the order the verifier visits them. The verifier sets $r_0 = \pi$. For $i = 1, \ldots, k$, it accepts c_i with probability $r_{i-1}(c_i)$. On rejection it sets $r_i = r_{i-1}(\cdot \mid \cdot \neq c_i)$, that is, r_{i-1} with c_i removed and the rest renormalized. If every child is rejected, it samples the emitted token from r_k. Sampling from r_{i-1} is the same as emitting c_i with probability $r_{i-1}(c_i)$ and otherwise sampling from r_{i-1} conditioned on $\neq c_i$, which is r_i. By induction on i, the emitted token is therefore distributed exactly as $r_0 = \pi$, for *every* choice of $c_1, \ldots, c_k$, duplicates included (a repeated child has zero residual mass and is never accepted twice). The only requirement is that the children are fixed before the verifier draws its coins, which holds because drafting finishes before verification begins. Applying the argument node by node along the accepted path shows that each emitted token is an exact sample from the target given its prefix. The output sequence is thus distributed as autoregressive sampling from the target.

Nothing in this argument refers to how the candidates were produced. Whether a child was drawn from π_{one} or from $\mathrm{head}(f_v^{d_v})$, whether its edge was repriced by Eq. (9), and whether the gate halted stochastically, all of this changes only the candidate set, and hence *how many* tokens are accepted per round, never *which distribution* they are drawn from. The same holds for EAGLE-3.1, whose tree is also chosen by drafter-side scores that the verifier never sees. Reconsideration therefore adds nothing to verification that a standard speculative decoder does not already rely on, and the temperature-1 results in Table 1 are lossless in the same sense as the baseline's.

D Details on Experiments

This appendix gives the full experimental detail behind Section 6. We state the setup in full (Appendix D.1), then report five further studies: whether spending the same compute on a wider draft reproduces the gain (Appendix D.2), how much the choice of which nodes to reconsider matters (Appendix D.3), whether the gain holds across tree configurations (Appendix D.4) and on top of a different base drafter (Appendix D.5), and which of the drafter's failures reconsideration actually recovers, returning to the diagnosis of Section 2.2 (Appendix D.6).

D.1 Experimental setup

Models and baseline. We evaluate on three target models, **Llama-3.1-8B-Instruct**, **Qwen3-8B**, and **Gemma-3-12B**, and compare **Parallel Reconsideration** against **EAGLE-3.1**, the state-of-the-art feature-level speculative decoder. We use EAGLE-3.1 to denote the current form of the EAGLE-3 drafter (Li et al., 2025) with the normalization fixes of Eldenk et al. (2026). Both drafters use a single decoder layer and share an identical target model, and both are trained on UltraChat (Ding et al., 2023) under EAGLE-3's released recipe, unchanged (Appendix F). Parallel Reconsideration takes at most two reconsideration steps per position (depth cap $R = 2$) on every model.

Implementation. We train both methods in SpecForge (Li et al., 2026), publicly available training recipe of EAGLE-3.1, and add reconsideration as a second head and gate within the same trainer. Both are then served through SGLang (Zheng et al., 2024), and every acceptance length and speedup we report is measured on SGLang serving engine, faithfully reporting realistic scenarios. As an external check on the baseline, our EAGLE-3.1 acceptance lengths agree with those reported independently for the same models and benchmarks by DFlash (Chen et al., 2026).

Benchmarks and metrics. We evaluate on seven benchmarks in three categories: *math* (**GSM8K** (Cobbe et al., 2021), **MATH-500** (Lightman et al., 2024), **AIME25**), *code* (**HumanEval** (Chen et al., 2021), **MBPP** (Austin et al., 2021), **LiveCodeBench** (Jain et al., 2025)), and *chat* (**MT-Bench** (Zheng et al., 2023)). Our primary metric is the *acceptance length* τ, the mean number of tokens the target accepts per verification round, which is the drafter's contribution to end-to-end speedup at fixed target and verifier cost (Eq. (2)); we also report the wall-clock speedup over autoregressive decoding, measured on one H200. We report both greedy decoding (temperature 0) and sampling at temperature 1, with max generation length of 2048 tokens. Benchmark sizes and sources are in Appendix F.

Budget configuration. Both drafters serve a dynamic draft tree bounded by a total budget B (the number of draft nodes verified per round), grown over γ draft steps with offer width k and grow width B_w (Section 5). We hold all of these equal across the two drafters: $B \in \{24, 48\}$, shown in parentheses in Table 1, with $k = 6$, $B_w = 6$, and $\gamma = 8$. Parallel Reconsideration additionally batches at most $m = 3$ reconsidered nodes per draft step, which share the co-forwarded batch; Appendix D.2 grants the baseline the same enlarged batch and spends it on one-pass breadth instead.

D.2 Reconsidering against drafting wider

Holding B, k, B_w and γ equal leaves one asymmetry: a draft step of Parallel Reconsideration batches $B_w + m$ nodes where EAGLE-3.1 batches B_w. If enlarging the batch is nearly free, then EAGLE-3.1 can enlarge its own by m and spend the additional nodes on one-pass breadth, and if that recovers the same acceptance then the gain belongs to the wider batch rather than to reconsideration.

We therefore grant the baseline exactly that, in two ways. The first control draws m nodes from the previous step's batch and, for each such node v, forwards with an ordinary draft step its best one-pass child not yet forwarded, i.e., a sibling of the child already advanced. These rows take new inputs and so compute new features, growing a second branch beneath v; the only difference is tree growing against reconsideration. Three properties are held fixed. The *source set* is the same, the previous step's batch rather than the current frontier, since reconsideration cannot reach further either. The *final pruning* is the same, in that the children they grow compete for the same places in the tree on the same score. And the *batch is the same size*, $B_w = 6$ advanced nodes plus $m = 3$ more.

The second control drops the first of those constraints and simply widens the grow width to $B_w = 9$. All nine nodes now extend the current frontier, which is more drafting breadth than Parallel Reconsideration has: our own m nodes are confined to positions already forwarded, whereas these are free to advance the best of the live frontier.

As a sanity check in the other direction, we also run Parallel Reconsideration with $m = 0$: the same drafter and serving path, but with no node ever reconsidered, so each step batches the same $B_w = 6$ nodes as EAGLE-3.1.

Table 4 (Section 6.5) reports the results. Without reconsideration, Parallel Reconsideration matches EAGLE-3.1 to within noise (-0.1% acceptance and -0.4% speedup on average), as Appendix C.3 predicts. The second head and gate therefore neither help nor hurt on their own, and the serving path adds no measurable overhead when it is idle. Neither control recovers the gain either. Advancing 3 more lifts average acceptance by 1.7% over EAGLE-3.1 and widening to $B_w = 9$ by 1.0%, against **9.6%** for reconsidering the same three, and speedup splits the same way ($+0.8\%$ and $+0.4\%$ against $+\mathbf{8.7}\%$). That the freer of the two controls rules out breadth as the explanation: the additional capacity pays only when it is spent on a second look at an uncertain feature. Appendix D.6 examines which of the one-pass drafter's misses reconsideration recovers.

Table 5: **Choosing which nodes to reconsider.** Acceptance length τ and speedup over autoregressive decoding on Llama-3.1-8B at temperature 0, with the same trained drafter and tree configuration ($B = 48$, $m = 3$), varying only the rule that selects the m reconsidered nodes. Δ is relative to the bid of Eq. (8).

	GSM8K		MBPP		LCB		**Avg.**	
Selection	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup
Bid (Eq. (8))	**5.40**	**2.26**×	**5.58**	**2.41**×	**5.12**	**2.11**×	**5.37**	**2.26**×
Highest confidence	5.29	2.20×	5.48	2.37×	5.01	2.06×	5.26	2.21×
Δ	−2.0%	−2.7%	−1.8%	−1.7%	−2.1%	−2.4%	−2.0%	−2.2%
Random	5.15	2.15×	5.44	2.33×	4.85	1.99×	5.15	2.16×
Δ	−4.6%	−4.9%	−2.5%	−3.3%	−5.3%	−5.7%	−4.1%	−4.6%

D.3 Choosing which nodes to reconsider

Each draft step reconsiders the m nodes that bid highest under Eq. (8). To isolate how much this choice matters, we keep the trained drafter, the tree configuration and $m = 3$ fixed, and replace only the selection rule with two alternatives: *highest confidence*, which reconsiders the nodes of the previous step whose one-pass guess is most confident, and *random*, which reconsiders nodes chosen uniformly at random. We evaluate on GSM8K, MBPP and LiveCodeBench with Llama-3.1-8B at temperature 0 (Table 5).

The bid gives the highest acceptance length and speedup on all three benchmarks. It leads highest confidence by 2.0% in average acceptance and 2.2% in speedup, and random by 4.1% and 4.6%.

D.4 Sensitivity to the tree configuration

Our main results fix one tree configuration ($B = 48$, $B_w = 6$, $k = 6$) for both drafters. To check that the gain is not an artifact of that choice, we vary the total budget B, the grow width B_w and the offer width k one at a time around it, again holding every knob equal across the two drafters (Table 6).

The acceptance gain holds across all nine configurations: Parallel Reconsideration adds $+0.26$ to $+0.30$ tokens per round over EAGLE-3.1, or 5.3–6.3% in relative terms, from $B = 32$ to $B = 64$ and whether the tree is narrow ($B_w = 4$) or wide ($B_w = 10$), and the gain grows to $+0.44$ tokens ($+10.0\%$) at the smallest budget ($B = 24$). Reconsideration therefore neither depends on a particular tree shape nor gets absorbed by a larger one. Growing the budget raises τ for both drafters, but past $B = 48$ the added verification cost outweighs the added acceptance, so speedup peaks at our default for Parallel Reconsideration (2.29×) and saturates for EAGLE-3.1 (2.18× to 2.20×). Widening B_w beyond 6 leaves τ unchanged for both and only costs speedup; narrowing it to 4 costs speedup most for Parallel Reconsideration, whose speedup margin nearly vanishes there (2.112× against 2.105×). Offer width behaves similarly: $k = 4$ and $k = 8$ keep the acceptance gain but lose speedup relative to $k = 6$. The default is thus the best configuration for Parallel Reconsideration, and it beats EAGLE-3.1 at every configuration tried, including the baseline's own best (2.29× against 2.20× at $B = 64$).

D.5 Reconsideration on top of a different drafter

Reconsideration adds a second head and gate to a one-pass drafter, so it is not tied to EAGLE-3.1. We test it on top of another speculative decoding method, HASS (Zhang et al., 2025). We train HASS and HASS with reconsideration on Llama-3.1-8B, compare them under the tree configuration of Table 1 ($B = 48$, $k = 6$, $B_w = 6$, $\gamma = 8$), and evaluate on all seven benchmarks at temperature 0 (Table 7).

Parallel Reconsideration lifts HASS on every benchmark, by $\mathbf{7.5}\%$ in average acceptance length ($5.02 \to 5.39$) and $\mathbf{4.4}\%$ in speedup ($2.15\times \to 2.24\times$), comparable to its gain over EAGLE-3.1 on the same model (Table 1). The gain is therefore not specific to one base drafter. The largest gain is on AIME25 ($+12.7\%$ acceptance, $+9.6\%$ speedup).

Table 6: **Sensitivity to the tree configuration.** Acceptance length τ and speedup over autoregressive decoding for EAGLE-3.1 and Parallel Reconsideration (PR) under identical budgets, varying one of the total budget B, grow width B_w and offer width k at a time around our default ($B = 48$, $B_w = 6$, $k = 6$, marked $*$ and repeated in each block); $\gamma = 8$ and $m = 3$ throughout, on Qwen3-8B, GSM8K, temperature 0. $\Delta\tau$ is PR minus EAGLE-3.1. Bold marks the best value per column.

			τ			Speedup	
B	B_w	k	EAGLE-3.1	PR	$\Delta\tau$	EAGLE-3.1	PR
Total budget							
24	6	6	4.39	4.83	**+0.44**	1.94×	2.11×
32	6	6	4.606	4.869	+0.263	2.077×	2.208×
48*	6	6	4.835	5.127	+0.292	2.184×	**2.291**×
64	6	6	**4.961**	**5.222**	+0.261	**2.195**×	2.280×
Grow width							
48	4	6	4.797	5.069	+0.272	2.105×	2.232×
48*	6	6	4.835	5.127	+0.292	2.184×	**2.291**×
48	8	6	4.826	5.126	+0.300	2.138×	2.266×
48	10	6	4.819	5.124	+0.304	2.096×	2.227×
Offer width							
48	6	4	4.767	5.061	+0.294	2.162×	2.286×
48*	6	6	4.835	5.127	+0.292	2.184×	**2.291**×
48	6	8	4.837	5.124	+0.287	2.080×	2.250×

Table 7: **Parallel Reconsideration on top of HASS.** Acceptance length τ and speedup over autoregressive decoding on Llama-3.1-8B at temperature 0, both drafters taken at the same training checkpoint and served under the tree configuration of Table 1 ($B = 48$). Δ is the relative gain of adding reconsideration to HASS.

	Math						**Code**						**Chat**			
	GSM8K		MATH-500		AIME25		HumanEval		MBPP		LCB		MT-Bench		**Avg.**	
Method	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup
HASS	4.81	2.03×	4.84	2.11×	5.21	2.27×	5.59	2.38×	5.34	2.38×	4.68	1.90×	4.64	1.97×	5.02	2.15×
HASS + PR	**5.17**	**2.12**×	**5.21**	**2.21**×	**5.88**	**2.49**×	**5.86**	**2.45**×	**5.67**	**2.44**×	**4.99**	**1.98**×	**4.97**	**2.02**×	**5.39**	**2.24**×
Δ	+7.6%	+4.6%	+7.7%	+4.6%	+12.7%	+9.6%	+4.7%	+2.9%	+6.0%	+2.4%	+6.6%	+3.9%	+7.2%	+2.6%	**+7.5%**	**+4.4%**

D.6 WHICH FAILURES DOES RECONSIDERATION RECOVER?

Figure 1 is correlational: a decisive target does not by itself show that a one-layer drafter could match it with more computation. As a partial check on the reading of Section 2.3, we measure *which* of the drafter's failures a trained second pass recovers. Taking every token the one-pass drafter rejects (30.2% of 3.1M scored positions on the held-out split) and bucketing them by the same $c_p^{\mathrm{d}} \times c_p^{\mathrm{t}}$ bands, we report the fraction of each bucket reconsideration, trained on the same model, turns from wrong to right (Table 8).

Reconsideration recovers **28.5%** of all rejected tokens, and where it recovers them supports the hypothesis. Recovery rises monotonically along the target axis ($16.4 \rightarrow 26.7 \rightarrow$ **39.9%**): it is governed by whether the target had a decisive answer, not by how unsure the drafter sounded. The premature commitments of Observation 2, drafted with $c_p^{\mathrm{d}} < 0.5$ where the target was decisive, are recovered **39.7%** of the time, while the both-uncertain tokens that Observation 2 identified as largely unrecoverable move least (17.7%). A mechanism that merely sharpened the existing distribution would help most where the drafter was already close; reconsideration instead spends itself on the determined-but-missed tokens, which is the band Section 2.3 set out to reach.

Table 8: **What reconsideration recovers, by bucket.** *Top:* percentage of each bucket's rejected tokens that Parallel Reconsideration gets right, with the bucket's share of the reject pool in parentheses. Buckets are the $c_p^{\mathrm{d}} \times c_p^{\mathrm{t}}$ bands of Section 2.2. *Bottom:* mean change in top-1 probability against the position's own one-pass distribution, on the tokens reconsideration recovers and on those it does not.

	target confidence c_p^{t}			
drafter c_p^{d}	< 0.5	0.5–0.9	≥ 0.9	**all**
recovered, % of bucket (bucket's share of pool)				
< 0.5	17.7% (18.5%)	29.5% (26.6%)	**39.7%** (17.0%)	28.8% (62.1%)
0.5–0.9	11.3% (6.7%)	23.4% (13.6%)	**42.0%** (15.1%)	29.0% (35.4%)
≥ 0.9	3.7% (0.4%)	6.9% (1.3%)	27.1% (0.8%)	12.9% (2.5%)
all	16.4% (25.6%)	26.7% (41.5%)	**39.9%** (32.9%)	**28.5%** (100%)
Δ *top*-1 *probability vs. own one-pass distribution*				
recovered	$+0.008$	$+0.024$	$\mathbf{+0.061}$	$\mathbf{+0.039}$
not recovered	$+0.000$	$+0.002$	$+0.004$	$+0.001$

The added confidence is selective in the same way (Table 8, bottom). On the tokens it recovers, reconsideration raises the top-1 probability by $\mathbf{+0.039}$, rising to $\mathbf{+0.061}$ where the target is decisive; on the tokens in the same pool it fails to recover, the same measurement gives $+0.001$. Each token is measured against its own one-pass distribution, which keeps c_p^{d}, the bucketing variable, out of the difference. This suggests the deeper pass does not simply sharpen indiscriminately.

E Details on Related Work

Speculative decoding. Speculative decoding accelerates autoregressive generation by drafting several tokens cheaply and verifying them in a single target forward pass, accepting the longest prefix the target agrees with so the output distribution is preserved exactly (Leviathan et al., 2023; Xia et al., 2023). Because the acceptance rule preserves that distribution by construction, subsequent work improves speed along two axes: the *acceptance length*, how many drafted tokens survive verification per round, and the *drafting cost*, how much latency the draft itself adds. Our cost model (Section 2) is the standard amortized form of this trade-off, and Parallel Reconsideration targets the first axis without paying on the second.

Feature-level drafting. The original formulation pairs the target with a smaller standalone language model as the drafter, which must be trained or found separately for each target. Self-drafting instead derives the draft from the target's own computation. The EAGLE family drafts at the feature level, autoregressing on a combination of hidden features from several target layers with a single lightweight decoder layer and verifying candidates with dynamic draft trees (Li et al., 2024a;b; 2025), most recently corrected for the attention drift that degrades the drafter as the speculation chain lengthens (Eldenk et al., 2026); a related line attaches lightweight prediction heads directly to the target's last hidden state, either independent (Medusa heads) (Cai et al., 2024) or made sequentially dependent to raise acceptance (Hydra heads) (Ankner et al., 2024). Parallel Reconsideration is built on the EAGLE-style feature-level drafter but is orthogonal to this design axis: it adds a second guess at uncertain positions on top of the existing one-pass draft, and applies to any drafter that exposes a reusable draft feature.

Adaptive drafter compute and draft budgets. Several lines of work already vary how much the drafter computes per token. Self-speculative decoding drafts with a subset of the target's own layers: Draft & Verify skips intermediate layers during drafting (Zhang et al., 2024), LayerSkip trains the target for early exit and drafts from its shallow layers (Elhoushi et al., 2024), and Kangaroo drafts from a shallow sub-network and exits drafting early once confidence drops (Liu et al., 2024). These methods adapt the *depth* of a sequential draft, so extra compute on a token is extra sequential latency; Parallel Reconsideration keeps the one-layer feature-level drafter and runs the extra pass beside the ordinary step. ReDrafter replaces the draft heads with a recurrent drafter conditioned on the target's

hidden state (Cheng et al., 2024), which still uses one recurrent step per drafted token. Sequoia and SpecExec allocate a fixed draft-tree budget unevenly across positions, placing more candidates where they are most likely to be accepted (Chen et al., 2024; Svirschevski et al., 2024), as EAGLE-2's dynamic trees do with drafter confidence (Li et al., 2024b). These methods vary how many candidates each position receives, not how each candidate is computed; reconsidered candidates enter the same tree and so compose with them. A direct empirical comparison with these methods is left to future work.

Training-free drafting. A complementary direction removes the trained drafter entirely. Lookahead decoding generates and verifies n-gram continuations in parallel through Jacobi iteration, requiring no auxiliary model or datastore (Fu et al., 2024), and FAFO extends this draftless n-gram candidate-pool paradigm by letting a compressed KV cache produce lossy guesses that the same forward pass verifies (Le et al., 2026). Such methods trade draft quality for zero training overhead; Parallel Reconsideration takes the opposite stance, spending a small amount of otherwise-idle draft compute to make each speculated position more accurate.

Parallel drafting. A drafter that autoregresses over γ positions is itself sequential, so a line of work replaces it with one that emits an entire block in a single forward pass. PARD trains a target-independent draft model to predict several future tokens at once (An et al., 2025), and diffusion-based drafters push this further: DFlash conditions a block diffusion model on target features to produce a draft block in one pass and holds the strongest reported speedups on this benchmark suite (Chen et al., 2026), while DSpark couples a parallel backbone with a lightweight sequential module to restore intra-block dependencies (Cheng et al., 2026). These methods reduce the *cost* axis by removing the drafter's sequential steps, whereas Parallel Reconsideration raises the *acceptance* axis by improving each drafted position; a block-parallel drafter still commits to every position from a single pass, so reconsideration composes with it.

Draft training objectives. Orthogonal to the drafter's architecture is what it is trained to optimize. Standard drafters minimize per-token likelihood against the target, which is misaligned with the sequence-level acceptance that verification actually rewards. Variational Speculative Decoding recasts draft training as variational inference over draft paths, maximizing the marginal probability of target acceptance rather than token likelihood (Zou et al., 2026). Our training (Section 4) shares this acceptance-first view but pursues it through a different mechanism: rather than reweighting a single draft distribution, we train a second reconsideration pass to recover the specific positions the one-pass draft gets wrong.

Looped and adaptive-compute transformers. The idea of spending extra passes to sharpen a prediction has a long history. Looped transformers reuse a fixed block to emulate iterative computation (Giannou et al., 2023), universal transformers add a recurrent depth with a per-token halting mechanism (Dehghani et al., 2019), and adaptive-compute methods such as PonderNet and CALM learn how many steps or layers each token needs (Banino et al., 2021; Schuster et al., 2022). A recent looped language model, Ouro, applies this recurrence to the target model during pre-training to improve its own generation quality (Zhu et al., 2025). Our use of recurrence differs in purpose: we loop not to make an autoregressive model reason better, but to repair the specific failure of autoregressive *drafting*, where the drafter must commit to a token before it has seen enough context to place it, and we run the extra pass in parallel so it costs no additional sequential step.

F Implementation details

Training. Both drafters follow EAGLE-3's released recipe on UltraChat (Ding et al., 2023), with no hyperparameter retuning for reconsideration. We use AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, no weight decay) at a peak learning rate of 10^{-4}, cosine-annealed to zero after linear warmup over the first 1.5% of steps, with gradients clipped to norm 0.5. Training runs in BF16 with FP32 master weights and gradient checkpointing, sequences capped at $4{,}096$ tokens, per-device batch size 1 at data-parallel degree 4 and no accumulation, giving an effective batch of four sequences (roughly 16k tokens) per step. The balance coefficient of Eq. (6) is $\beta = 0.01$.

Benchmarks. We evaluate each benchmark on its full set.

Table 9: **Evaluation benchmarks.** Each is evaluated on its full set.

Category	Benchmark	Prompts	Source
Chat	MT-Bench	80	FastChat `mt_bench`
Math	GSM8K	1,319	`openai/gsm8k` (main)
	MATH-500	500	`HuggingFaceH4/MATH-500`
	AIME25	30	`math-ai/aime25`
Code	HumanEval	164	`openai/openai_humaneval`
	MBPP	500	`google-research-datasets/mbpp` (full)
	LiveCodeBench	400	`livecodebench/code_generation`
Total		**2,993**	

G LIMITATIONS

Reconsideration, in principle, composes with any drafter that exposes a reusable draft feature, including the block-parallel drafters that are currently state of the art (Appendix E). DFlash (Chen et al., 2026) is the natural next target: it reduces the cost of drafting while reconsideration raises the acceptance of each drafted position, so the two act on different terms of Eq. (2) and there is no obvious reason they would conflict. Unlike EAGLE-style drafters, however, DFlash does not rely on a token embedding to guide its feature drafting, so there is no embedding input for the self-query to take the place of (Eq. (5)); applying reconsideration on top would require substantial redesign, which we leave for future work. Doing so also requires retraining a DFlash drafter with the extra head and gate, and its training recipe has not been released, so we cannot reproduce the baseline or train reconsideration on top of it. We therefore demonstrate on EAGLE-3.1, whose recipe is publicly released, and leave the block-parallel combination open.

H THE COST OF CO-FORWARDING, MEASURED

Section 3.4 argues that a reconsideration pass shares the batch that advances the draft, costing ϵ rather than a full draft step. We check that claim directly. For each cell we form the per-cycle time $t_{\text{cycle}} = \tau/\text{throughput}$, the wall-clock milliseconds per verification step, and compare Parallel Reconsideration against EAGLE-3.1 run for run at temperature 0. Table 10 reports the ratio alongside the serial reference of Section 3.4: run in series, the N reconsidered positions of a γ-step draft add N sequential draft steps, so drafting takes $(\gamma + N)\, t_{\text{step}}$ instead of $\gamma\, t_{\text{step}}$, a factor of $(\gamma + N)/\gamma$. That factor inflates only the drafting phase, so the serial version's whole-cycle cost would lie between 1 and $(\gamma + N)/\gamma$ depending on drafting's share of the cycle.

Table 10: **Per-cycle cost of co-forwarding on Qwen3-8B.** Measured ratio of $t_{\text{cycle}} = \tau/\text{throughput}$ (time per verification step), Parallel Reconsideration over EAGLE-3.1 run for run at temperature 0; this is a whole-cycle ratio (drafting plus verification). With $\gamma = 8$ draft steps and N reconsidered positions per draft, serializing would stretch drafting by $(\gamma + N)/\gamma$ (Section 3.4), an upper bound on the serial whole-cycle ratio. Co-forwarding keeps the measured ratio near 1.

B	B_w	N	**EAGLE-3.1** t_{cycle}	**PR** t_{cycle}	**measured** whole-cycle ratio	**serial drafting** $(\gamma{+}N)/\gamma$
48	6	3	26.03	26.60	1.022×	1.375×
24	6	3	25.55	25.94	1.015×	1.375×

With $\gamma = 8$ and $N = 3$ reconsidered positions, serializing would lengthen drafting by 37.5%; co-forwarded, the same reconsiderations lengthen the whole cycle by only 1.5–2.2% (ratios 1.015 and 1.022 at $B = 24$ and $B = 48$). The co-forwarded rows thus add roughly 2% to total cycle time (drafting plus verification), consistent with the memory-bandwidth argument of Section 3.4.

Table 11: **Where the gate halts, over training.** Percentage of positions halting at each depth, $R = 2$. Depth 0 is the one-pass feature, so those positions are never reconsidered. At 5k the allocation is tilted deep, with only 19–24% of positions halting at depth 0; mass shifts toward depth 0 over the next 35k steps, and from 40k onward the split holds steady at roughly 47–61% / 26–41% / 11–17% for depths 0 / 1 / 2.

Step	5k	20k	40k	60k	80k	100k	120k	140k	160k	180k	200k
Llama-3.1-8B											
$d_p = 0$	23.3	40.6	58.4	60.6	57.9	58.6	58.4	57.5	59.6	60.0	56.8
$d_p = 1$	37.9	44.2	29.0	26.9	30.5	30.2	29.3	30.8	29.7	27.5	30.2
$d_p = 2$	38.9	15.1	12.6	12.5	11.5	11.3	12.3	11.7	10.7	12.5	13.1
Qwen3-8B											
$d_p = 0$	24.2	46.8	51.5	48.8	47.6	51.4	46.9	46.9	47.8	47.7	47.6
$d_p = 1$	44.2	42.1	36.3	39.2	40.6	35.4	37.3	38.3	37.2	36.2	35.9
$d_p = 2$	31.6	11.0	12.2	12.0	11.8	13.2	15.7	14.8	15.0	16.1	16.6
Gemma-3-12B											
$d_p = 0$	19.0	41.6	49.0	52.2	57.1	58.1	56.1	54.9	55.7	56.6	55.8
$d_p = 1$	31.3	42.5	37.5	33.7	28.5	27.4	29.1	29.3	28.7	26.3	28.6
$d_p = 2$	49.7	15.9	13.5	14.1	14.4	14.4	14.8	15.8	15.6	17.1	15.6

I What the gate learns to allocate

Table 11 reads the halt depth d_p over the first 200k training steps, which separates the two jobs the balance term is claimed to do and not do in Section 4.

No depth goes dark. Across all three models and every checkpoint, no depth ever drops below 10.7% of positions, so every depth keeps receiving gradient and none is abandoned.

J Supervising both heads

This section expands on the two degenerate training objectives of Section 4 and why each collapses serving quality.

Supervising only the reconsidered head. Training only $\mathrm{head}(f_p^r)$ for $r \geq 1$ is the naive way to apply a looped transformer (Giannou et al., 2023; Dehghani et al., 2019) to drafting: run the layer to its final depth and supervise the last head, exactly as one would optimize a recurrent model for its own output. It fails here for a reason specific to speculative decoding. The one-pass head $\mathrm{head}(f_p^0)$ is never trained, so the one-pass draft is free to drift and nothing holds it to ordinary-drafter quality. Because the served tree is grown from the one-pass feature and the reconsidered candidates only extend prefixes the one-pass draft already reaches, a degraded one-pass front leaves the reconsidered branches with short or empty prefixes to build on. At serving, this can pull acceptance length below the one-pass-only baseline even if the reconsidered head looks accurate in isolation under teacher forcing. This is why adapting loop transformer idea to the draft is non-trivial: unlike a looped language model, whose only output is the final pass, the draft must keep *every* intermediate pass serving-ready, since the tree is grown from the one-pass feature and reconsideration only adds to it.

Supervising both heads symmetrically. If instead both $\mathrm{head}(f_p^0)$ and $\mathrm{head}(f_p^1)$ are trained to match the target with equal weight, gradient descent has no incentive to make the two features differ, and it is free to settle on the trivial optimum $f_p^1 \approx f_p^0$. The reconsidered token then echoes the one-pass token: the second pass is real compute that adds no new candidate, and acceptance length can do no better than the one-pass baseline.

The recipe of Section 4 avoids both failures by holding the one-pass draft to full quality with an unconditional loss while supervising the reconsidered feature only at the positions and depths where it is actually used, so f_p^1 is pushed away from f_p^0 exactly where a second guess can help.

K MARGINAL VS. PER-POSITION ENTROPY

The balance term of Eq. (6) is not the obvious entropy. The halt distribution induces two: the per-position entropy averaged over positions, $\mathbb{E}_p[H(q)]$, and the entropy of the marginal usage $\bar{q} = \mathbb{E}_p[q]$. We reward the marginal $H(\bar{q})$, and a clean decomposition says why.

Proposition 1 (Marginal vs. per-position entropy)**.** *For any distribution of per-position halt distributions q,*

$$H(\bar{q}) = \underbrace{I(d;p)}_{\textit{spread across depths}} + \underbrace{\mathbb{E}_p\big[H(q)\big]}_{\textit{per-position indecision}} \geq \mathbb{E}_p\big[H(q)\big], \tag{13}$$

where $d \sim q$ is the halt depth and $I(d;p) \geq 0$ is its mutual information with the position. Consequently $H(\bar{q})$ depends on the routing only through the marginal $\bar{q}$: it is indifferent *between a deterministic routing that sends different positions to different depths ($I(d;p)$ large, $\mathbb{E}_p[H(q)] = 0$) and an undecided one that splits every position across depths ($I(d;p) = 0$, $\mathbb{E}_p[H(q)]$ large), so long as the two induce the same $\bar{q}$.*

Proof of Proposition 1. Fix a batch of positions p, each with a halt distribution $q(\cdot \mid p)$ over depths $r \in \{0, 1, \ldots, R\}$, and let p be drawn uniformly over the batch. Define the joint distribution of position and halt depth d,

$$\Pr(p, d = r) = \Pr(p)\, q(r \mid p), \tag{14}$$

and the marginal usage $\bar{q}(r) = \mathbb{E}_p[\, q(r \mid p)\,] = \Pr(d = r)$.

Decomposition. The entropy of the marginal is the entropy of d, which by the chain rule of mutual information decomposes as

$$\begin{aligned} H(\bar{q}) = H(d) &= I(d;p) + H(d \mid p) \\ &= I(d;p) + \mathbb{E}_p\big[H(q(\cdot \mid p))\big], \end{aligned} \tag{15}$$

since $H(d \mid p) = \sum_p \Pr(p)\, H(q(\cdot \mid p)) = \mathbb{E}_p[H(q)]$. Because $I(d;p) \geq 0$, this gives $H(\bar{q}) \geq \mathbb{E}_p[H(q)]$, which is exactly the Jensen inequality $H(\mathbb{E}_p[q]) \geq \mathbb{E}_p[H(q)]$ applied to the concave entropy functional; (15) identifies the gap as the mutual information $I(d;p)$. This proves (13).

Indifference. Both terms in (15) are nonnegative and $H(\bar{q}) \leq \log(R{+}1)$, with equality iff $\bar{q}$ is uniform. Since $H(\bar{q})$ is a function of $\bar{q}$ alone, it cannot distinguish two routings that induce the same marginal, however differently they allocate mass per position. Two extremes make this concrete. The *deterministic* routings — $q(\cdot \mid p)$ a point mass at some depth $d(p)$, with $d(p)$ distributed uniformly across the batch — attain $H(\bar{q}) = \log(R{+}1)$ with $\mathbb{E}_p[H(q)] = 0$ and $I(d;p) = \log(R{+}1)$. The fully undecided routing — every $q(\cdot \mid p)$ uniform — attains the same $H(\bar{q}) = \log(R{+}1)$ with the decomposition reversed, $\mathbb{E}_p[H(q)] = \log(R{+}1)$ and $I(d;p) = 0$. The balance term scores both identically, so it neither rewards nor penalizes per-position indecision; it penalizes only a marginal that has collapsed onto a subset of the depths. By contrast, regularizing $\mathbb{E}_p[H(q)]$ directly is maximized only by the second extreme, forcing per-position indecision. □

Practical note. We estimate $\bar{q}$ per minibatch, so the balance term rewards spread across the depths *used within a batch* rather than a global marginal; with the small coefficient $\beta = 0.01$ this is enough to keep every depth populated without measurably softening the per-position halt decisions.