# STEP: SEPARATING TRAJECTORY EVOLUTION AND PREDICTION FOR SPECULATIVE DECODING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Speculative decoding accelerates large language model inference, but its efficiency depends on sustaining accurate draft predictions over multiple future positions. Autoregressive drafters such as EAGLE-3 use a shared hidden state for token prediction and recurrent state propagation. We observe that high token prediction accuracy can coexist with substantial recurrent-state deviation from target-model features, motivating separate representations for these two roles. We introduce STEP, a feature–logit disentangled drafter that separates trajectory evolution from token prediction. To stabilize recurrent propagation, we employ a staged feature-alignment curriculum that uses target-model fused features as an early training reference before gradually removing the auxiliary constraint. We further exploit the resulting long-horizon drafting capability with confidence-guided progressive tree expansion, selectively extending promising branches to obtain longer accepted continuations. Across four target models, five benchmarks, and two temperatures, STEP outperforms EAGLE-3 in all 40 settings, with average relative gains of 10.1% in decoding speed and 15.1% in acceptance length. Ablations support complementary roles for disentanglement and alignment, while progressive tree expansion further improves decoding efficiency. In SGLang, STEP also achieves 2.9–11.9% higher throughput than EAGLE-3 across all tested batch sizes under the same speculative token budget.

## 1 INTRODUCTION

Autoregressive decoding limits the efficiency of large language models (LLMs), particularly as reasoning models generate long reasoning traces (Guo et al., 2025) and agentic workflows involve repeated model interactions (Yao et al., 2023). Speculative decoding reduces this latency by using a lightweight drafter to propose tokens that the target model verifies in parallel (Leviathan et al., 2023; Chen et al., 2023). Its practical benefit depends on balancing drafting cost against prediction quality across multiple future positions.

Two common approaches to this trade-off are parallel and autoregressive drafting. Parallel drafters, including Medusa (Cai et al., 2024) and DFlash (Chen et al., 2026), reduce sequential drafting cost by proposing multiple future tokens in one forward pass. Because later proposals are not explicitly conditioned on tokens newly sampled at earlier draft positions in the same forward pass (Ankner et al., 2024; Chen et al., 2026), parallel drafting achieves low drafting latency while potentially sacrificing acceptance at deeper positions. Autoregressive drafters (Zhou et al., 2024; Du et al., 2024; Cheng et al., 2024; Miao et al., 2024) such as EAGLE-3 (Li et al., 2025) instead condition each draft step on the sampled prefix through lightweight recurrent computation. This sequential conditioning incurs additional drafting cost, but supports more accurate later-position predictions and longer accepted continuations. It also naturally supports incremental tree expansion, allowing candidate branches to be extended according to their confidence (Li et al., 2024b).

The stronger proposal quality of autoregressive drafting comes from recursively using each draft state to generate subsequent predictions. This raises a natural question: can the same representation support both token prediction and stable recurrent state propagation? In EAGLE-3, the hidden output of the draft Transformer at each step serves both token prediction and recurrent state propagation (Li et al., 2025). Because the drafter is conditioned on fused target-model features and recursively feeds its own hidden states into subsequent draft steps, deviation from the corresponding target fused



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.

features provides a natural diagnostic of recurrent state drift. We therefore track token prediction accuracy together with the cosine similarity between the recurrent state and the corresponding target fused feature. On LLaMA-3.1-8B-Instruct (Dubey et al., 2024), these two quantities exhibit markedly different behaviors. Figure 1(b) shows that token prediction accuracy remains relatively high at early draft positions before declining sharply at later steps. In contrast, recurrent-state similarity is already only about 0.5 at the first step (Figure 1(a)) and drops to 0.26 by step seven, while token prediction accuracy remains 69%. These results suggest that the shared recurrent state does not simultaneously preserve both properties desired for extended autoregressive drafting: a representation effective for token prediction and a stable recurrent state propagation that remains close to the target fused feature manifold. As drafting proceeds recursively, the recurrent state deviates further from the target features, accompanied by pronounced degradation in token prediction at later positions. This motivates decoupling the two roles, allowing the drafter to maintain stable recurrent state propagation while using a separate representation specialized for token prediction.

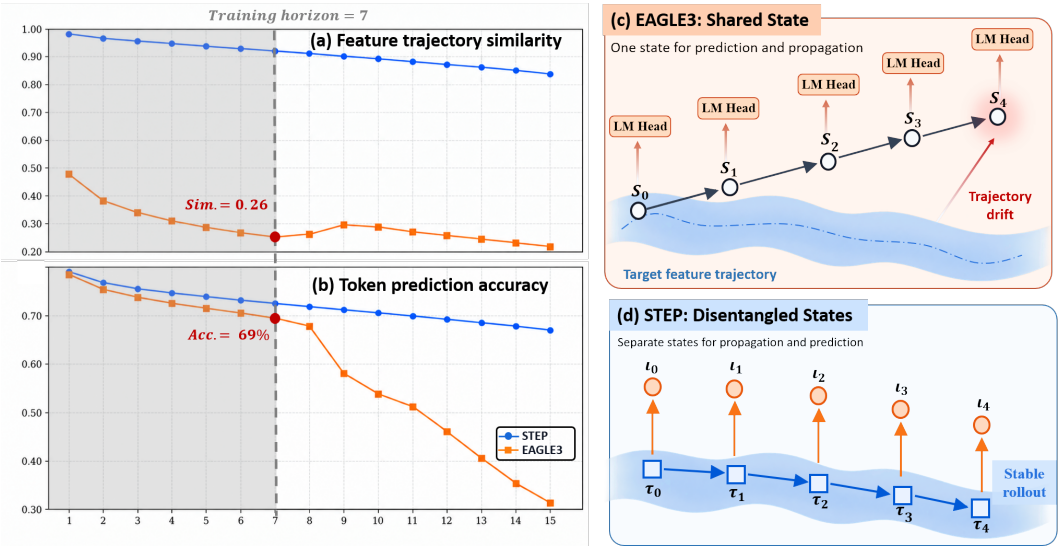


Figure 1: **Feature alignment and token accuracy across drafting horizons.** (a–b) On LLaMA-3.1-8B-Instruct, EAGLE-3 exhibits low cosine similarity to target fused features at early draft positions and a pronounced accuracy decline at later positions. Our drafter maintains higher alignment and accuracy. Both drafters use a seven-step training rollout; the dashed line marks this length. (c–d) EAGLE-3 shares one state for recurrent state propagation and token prediction, whereas our design separates the trajectory and logit states.

We introduce **STEP**, a feature–logit disentangled drafter that **s**eparates **t**rajectory **e**volution from token **p**rediction. STEP combines two complementary designs: architectural separation of the representations used for token prediction and recurrent state propagation, and training-time stabilization of the recurrent trajectory through a staged feature-alignment curriculum. Architecturally, the drafter maintains a trajectory state dedicated to recurrent state propagation and derives a separate logit state for token prediction through a lightweight nonlinear readout (Figure 1(d)). This allows the logit state to specialize for token prediction without requiring the trajectory state to directly serve the same prediction-oriented objective. To promote stable recurrent propagation over long drafting horizons, we further introduce a weak feature-alignment loss that uses the target model's fused features as an early training reference, encouraging the trajectory state to remain near the target feature manifold. We thus gradually remove the alignment term after it provides an initial geometric anchor, avoiding over-regularization toward exact feature reconstruction while allowing the trajectory state to further adapt under the prediction objective. Together, architectural disentanglement and staged alignment enable more stable recurrent state propagation and improve prediction at later draft positions. These design choices translate into substantially stronger long-horizon drafting behavior. On LLaMA-3.1-8B-Instruct, STEP achieves 67.03% accuracy at step fifteen, compared with 31.30% for EAGLE-3 (Figure 1(b)). Consistent with this improvement, our trajectory state retains a cosine similarity of 0.84 to the target fused features at the same step (Figure 1(a)).

These more accurate later predictions create an opportunity for deeper drafting, but realizing this benefit requires balancing tree depth against drafting and verification costs (Li et al., 2024b; Brown et al., 2024). Shallow expansion can miss useful continuations, whereas uniformly increasing depth spends computation on branches with little chance of acceptance. We therefore introduce **confidence-guided progressive tree expansion**, which first explores shallow branches and then selectively deepens those exceeding a cumulative-confidence threshold. Combined with the proposed drafter, this strategy produces longer accepted drafts and approximately 10% higher end-to-end decoding speed than EAGLE-3 across the evaluated models and benchmarks.

Our contributions are threefold:

- We identify **representation entanglement** in autoregressive drafting: EAGLE-3 uses a shared recurrent state for both token prediction and recurrent state propagation, making it difficult to simultaneously preserve high token prediction accuracy and a target-like recurrent state over extended rollouts.
- We propose a **feature–logit disentangled drafter** with staged feature alignment, separating recurrent state propagation from token prediction while using early feature alignment to guide the trajectory state toward a stable target-like representation.
- We develop **confidence-guided progressive tree expansion** to translate improved long-horizon prediction into longer accepted drafts and faster end-to-end decoding.

## 2 Related Work

**Parallel and semi-autoregressive multi-token drafting.** Medusa (Cai et al., 2024) augments target model with multiple decoding heads that predict future positions in parallel, substantially reducing sequential drafting cost. However, its heads predict future positions without explicitly conditioning on tokens sampled by preceding heads. Hydra (Ankner et al., 2024) addresses this limitation by introducing sequentially dependent heads that condition on preceding draft tokens, improving draft quality while retaining the lightweight head-based architecture. DFlash (Chen et al., 2026) takes a different parallel approach, using a lightweight block diffusion drafter conditioned on target-model features to generate an entire draft block in a single forward pass (Stern et al., 2018). While efficient, its parallel positions lack explicit conditioning on the realized intra-block prefix, which can lead to acceptance degradation toward later draft positions. DSpark (Cheng et al., 2026) builds on this parallel backbone with a sequential module that injects intra-block token dependencies and mitigates such suffix decay. Collectively, these methods highlight the importance of modeling dependencies among draft tokens, even when substantial drafting computation is performed in parallel.

**Autoregressive feature drafting and tree construction.** EAGLE (Li et al., 2024a) introduces feature-level autoregression, incorporating sampled tokens to resolve uncertainty in predicting subsequent features. EAGLE-2 (Li et al., 2024b) extends this framework with context-aware dynamic draft trees (Chen et al., 2024), using draft confidence to guide expansion and candidate selection. EAGLE-3 (Li et al., 2025) further introduces multi-layer feature fusion and training-time test, removing explicit feature-regression supervision. Unlike parallel and semi-autoregressive approaches, EAGLE-style drafters explicitly condition every draft step on the realized sampled prefix. Our study builds on this autoregressive framework and focuses on a different limitation that emerges under repeated recursion: the coupling between local token prediction and recurrent state propagation.

## 3 Preliminaries

### 3.1 Speculative Decoding

Let $p_\theta$ and $q_\phi$ denote the target and draft distributions. Given a verified prefix $x_{1:t}$, the drafter autoregressively proposes up to $K$ tokens, which the target model evaluates in parallel. Greedy verification accepts consecutive proposals matching the target's next-token predictions. For stochastic decoding, let $p_k$ and $q_k$ denote the target and draft distributions at draft position $k$, both conditioned on the same realized token prefix. Each proposal $\hat{x}_k$ is accepted with probability $\min\{1, p_k(\hat{x}_k)/q_k(\hat{x}_k)\}$. At the first rejection, a correction token is sampled from the residual distribution

$$r_k(x) = \frac{[p_k(x) - q_k(x)]_+}{\sum_v [p_k(v) - q_k(v)]_+}, \qquad [z]_+ = \max(z, 0).$$

If all proposals are accepted, the target may sample one additional token.

Let $M$ denote the number of tokens committed in one decoding cycle, including any correction or additional target token. Ignoring scheduling and bookkeeping overhead, an idealized single-request latency speedup is

$$S \approx \frac{\mathbb{E}[M] T_{\mathrm{AR}}}{\mathbb{E}[T_{\mathrm{draft}} + T_{\mathrm{verify}}]}, \tag{1}$$

where $T_{\mathrm{AR}}$ is the average target-only decoding time per token, while $T_{\mathrm{draft}}$ and $T_{\mathrm{verify}}$ denote the drafting and verification time per cycle, respectively. These costs depend on the draft depth and, for tree-based drafting, the number of candidate nodes. Longer drafts therefore improve speed only when the additional accepted tokens offset their computational cost.

### 3.2 EAGLE-3 and Drafting Horizons

EAGLE-3 conditions a lightweight autoregressive drafter on fused features from target model. At draft step $k$, let $\mathcal{C}_k$ denote the available causal context, including target-prefix features, preceding draft states, and token embeddings. Its shared-state computation can be summarized as

$$s_k = D_\phi(\mathcal{C}_k), \qquad q_k(\cdot) = \mathrm{softmax}(\mathrm{LMHead}(s_k)). \tag{2}$$

The state $s_k$ also enters the context for subsequent draft steps, together with the newly sampled token's embedding. Thus, the same representation supports both token prediction and recurrent propagation. Equation (2) abstracts the cached computation.

EAGLE-3 removes explicit feature-prediction supervision and uses training-time test to train on recursively generated states. We denote the number of recurrent draft steps unrolled during training by $H$, and the drafting horizon used at inference by $K$. These two quantities are conceptually distinct; throughout our analysis, we study how the recurrent state propagation and token prediction quality evolve as $K$ increases, including extended rollouts with $K > H$.

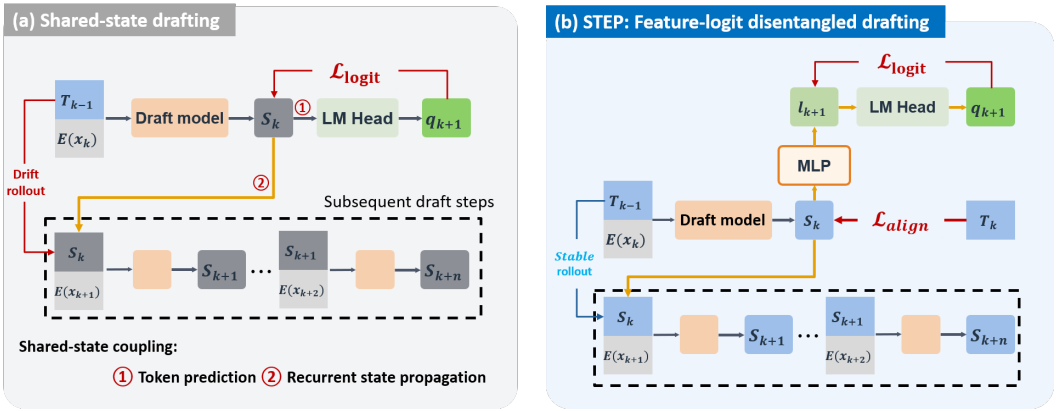


Figure 2: **Feature–logit disentanglement in STEP. (a)** Shared-state drafting uses the same state $s_k$ for token prediction and recurrent state propagation, coupling the two roles within a single representation. **(b)** STEP retains $s_k$ as the trajectory state and derives a separate logit state $\ell_k$ through a lightweight MLP for token prediction. Only $s_k$ is propagated across draft steps, while an auxiliary alignment objective $\mathcal{L}_{\mathrm{align}}$ guides its trajectory toward the target fused features early in training, with its weight gradually reduced to zero in the later stage.

## 4 Method

STEP separates recurrent state propagation from token prediction in an autoregressive drafter. Figure 2 illustrates the feature–logit disentangled architecture. During training, we employ a

staged feature-alignment curriculum to encourage stable recurrent state propagation. We then use confidence-guided progressive tree expansion to allocate deeper drafting to selected continuations.

### 4.1 Representation Entanglement in Autoregressive Drafting

In EAGLE-3, the recurrent state is simultaneously optimized to serve as a logit state for token prediction and as a trajectory state for recurrent state propagation. Token prediction requires features that discriminate among candidate tokens, whereas recurrent state propagation requires a state that remains useful when repeatedly consumed by the drafter. Accurate token predictions within a finite training horizon $H$ do not establish that the same representation supports effective propagation over longer inference horizons $K > H$. The shared-state architecture assigns both requirements to one representation without a separate learned prediction branch dedicated to token prediction (Figure 2 a). We therefore separate the state used for recurrent state propagation from the representation used for token prediction.

### 4.2 Feature–Logit Disentangled Drafter

Our drafter maintains a *trajectory state* $s_k \in \mathbb{R}^{d_s}$ and derives a *logit state* $\ell_k \in \mathbb{R}^{d_\ell}$ through a lightweight mapping $g_\psi$:

$$s_k = D_\phi(\mathcal{C}_k), \qquad \ell_k = g_\psi(s_k), \qquad q_k = \mathrm{softmax}(W_{\mathrm{LM}}\ell_k), \tag{3}$$

where $W_{\mathrm{LM}} \in \mathbb{R}^{V_d \times d_\ell}$ is the draft LM head and $V_d$ is the draft vocabulary size. The trajectory state $s_k$ enters the context update together with the proposed token embedding, whereas the logit state $\ell_k$ is used only for token prediction and is not fed back as the recurrent feature input. Both states are recomputed at each draft step (Figure 2b).

We implement the draft model $D_\phi$ with a single Transformer layer, while $g_\psi$ is instantiated as the feed-forward (MLP) sublayer of a Transformer block. Unlike an additional Transformer layer, $g_\psi$ performs only nonlinear transformation: it introduces no self-attention computation and requires no additional KV cache during autoregressive drafting. The added prediction branch therefore incurs substantially less sequential inference overhead than introducing another Transformer layer.

To further keep the disentangled drafter lightweight, we reduce the width of the prediction branch from the trajectory dimension $d_s$ to $d_\ell < d_s$. Accordingly, $g_\psi$ produces the reduced-width logit state $\ell_k$, and the draft LM head $W_{\mathrm{LM}}$ operates on this narrower representation. Compared with directly applying a wide LM head to the trajectory state, this design replaces the original wide vocabulary projection with a lightweight nonlinear prediction transformation followed by a reduced-width LM head. The parameter and computation savings from the narrower prediction branch largely offset the additional cost introduced by $g_\psi$, allowing the drafter to introduce feature–logit disentanglement while preserving essentially the same parameter budget and drafting-time inference cost.

This architecture assigns recurrent propagation primarily to $s_k$ and vocabulary discrimination to $\ell_k$. The separation is architectural: prediction gradients still propagate through $g_\psi$ into the trajectory state. Its purpose is to give vocabulary prediction a dedicated transformation while preserving a separate representation for future state generation.

### 4.3 Staged Feature-Alignment Curriculum

We use feature alignment as a training curriculum for the trajectory state, providing an early reference for learning recurrent geometry. Following the training-time-test principle (Li et al., 2025), training unrolls the drafter for $H$ steps using recursively generated trajectory states and the aligned token embeddings from the training sequence. Let $\widetilde{p}_k$ denote the target token distribution mapped to the draft vocabulary, and let $f_k$ denote the corresponding target fused feature in the same space as $s_k$. The prediction loss is a soft-target cross-entropy objective (Hinton et al., 2015). We combine it with an auxiliary trajectory-alignment objective:

$$\mathcal{L}_{\mathrm{logit}} = \sum_{k=1}^{H} \omega_k \, \mathbb{E}_{\mathrm{train}} \left[ - \sum_{v \in \mathcal{V}_d} \widetilde{p}_k(v) \log q_k(v) \right], \tag{4}$$

$$\mathcal{L}_{\mathrm{align}} = \sum_{k=1}^{H} \omega_k \, \mathbb{E}_{\mathrm{train}} \left[ \frac{1}{d_s} \left\| s_k - \mathrm{sg}(f_k) \right\|_2^2 \right], \tag{5}$$

where $\mathcal{V}_d$ is the draft vocabulary, $\omega_k$ denotes the rollout weight, and $\mathrm{sg}$ stops gradients through the alignment target. The expectations denote the batch reductions with the corresponding supervision masks. Both objectives use the same rollout weights.

The training objective combines token prediction with trajectory alignment:

$$\mathcal{L} = \mathcal{L}_{\text{logit}} + \lambda \mathcal{L}_{\text{align}}. \tag{6}$$

We initially set $\lambda = 0.1$, using feature alignment as a weak auxiliary constraint on the trajectory state. We subsequently reduce $\lambda$ to zero in late training, allowing the trajectory state to further adapt under the prediction objective while avoiding persistent feature-matching constraints.

### 4.4 Confidence-Guided Progressive Tree Expansion

Deeper drafting is useful only when additional accepted tokens compensate for the drafting and verification costs (Chen et al., 2023). Uniformly extending every branch spends computation on continuations with low proposal confidence. Prior work adapts draft depth using aggregate confidence over the current beam (Brown et al., 2024). We use a two-stage expansion strategy that combines shallow exploration with selective deepening of individual high-confidence branches to exploit the drafter's longer prediction horizon.

The first stage explores candidate continuations to a shallow depth $d_0$. Following confidence-based tree construction (Li et al., 2024b), we score a branch $b = (\hat{x}_1^{(b)}, \ldots, \hat{x}_d^{(b)})$ by its cumulative draft confidence:

$$C(b) = \prod_{j=1}^{d} q\Big(\hat{x}_j^{(b)} \mid x_{1:t}, \hat{x}_{1:j-1}^{(b)}\Big). \tag{7}$$

The implementation accumulates these scores in log space. Let $\mathcal{F}_0$ denote the frontier at the end of shallow exploration. Only branches exceeding a threshold $\gamma$ are eligible for the second stage:

$$\mathcal{F}_\gamma = \{b \in \mathcal{F}_0 : C(b) > \gamma\}. \tag{8}$$

The second stage continues drafting from these branches using their trajectory states and cached contexts, subject to the prescribed expansion widths, depth limits and node budgets. Candidate continuations are ranked by cumulative confidence. Branches that fail the threshold are not expanded further, and the second stage is skipped if $\mathcal{F}_\gamma$ is empty. The retained candidates and their ancestors form the tree passed to the target model for verification. Here, $C(b)$ is a proposal-likelihood score for allocating computation; acceptance remains determined by target verification. Algorithm 1 in Appendix A provides the complete construction procedure.

Both stages reuse the same trained drafter. For each model, we keep all tree-construction hyperparameters fixed across datasets, including the two stages' expansion settings and the confidence threshold. The selected branches adapt to the input through their draft probabilities, while the configuration remains unchanged. This scheduling gives the model access to deeper continuations where its long-horizon predictions are most promising, while retaining shallow exploration elsewhere.

## 5 Experiments

### 5.1 Experimental Setup

**Models and tasks.** We follow the model and benchmark settings of DFlash (Chen et al., 2026) and EAGLE-3 (Li et al., 2025). Our target models are Vicuna-13B (Chiang et al., 2023), LLaMA-3.1-8B-Instruct, LLaMA-3.3-70B-Instruct, and DeepSeek-R1-Distill-LLaMA-8B. We evaluate conversation, code generation, mathematical reasoning, instruction following, and summarization on MT-Bench (Zheng et al., 2023), HumanEval(Chen et al., 2021), GSM8K (Cobbe et al., 2021), Alpaca (Taori et al., 2023), and CNN/DailyMail (Nallapati et al., 2016), respectively. For each target model, a single draft checkpoint is evaluated across all five benchmarks without task-specific fine-tuning. Results are reported at temperatures 0 and 1.

**Training and implementation.** We follow EAGLE-3's training data setup, using prompts from ShareGPT and UltraChat-200K (Ding et al., 2023) with target-generated responses. Training uses

Table 1: Speedup and average acceptance length $\tau$ across different models and benchmarks.

| | | MT-bench | | HumanEval | | GSM8K | | Alpaca | | CNN/DM | | Mean | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | Method | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ |
| Temperature=0 | | | | | | | | | | | | | |
| V 13B | SpS | 1.93x | 2.27 | 2.23x | 2.57 | 1.77x | 2.01 | 1.76x | 2.03 | 1.93x | 2.33 | 1.92x | 2.24 |
| | PLD | 1.58x | 1.63 | 1.85x | 1.93 | 1.68x | 1.73 | 1.16x | 1.19 | 2.42x | 2.50 | 1.74x | 1.80 |
| | Medusa | 2.07x | 2.59 | 2.50x | 2.78 | 2.23x | 2.64 | 2.08x | 2.45 | 1.71x | 2.09 | 2.12x | 2.51 |
| | Lookahead | 1.65x | 1.69 | 1.71x | 1.75 | 1.81x | 1.90 | 1.46x | 1.51 | 1.46x | 1.50 | 1.62x | 1.67 |
| | Hydra | 2.88x | 3.65 | 3.28x | 3.87 | 2.93x | 3.66 | 2.86x | 3.53 | 2.05x | 2.81 | 2.80x | 3.50 |
| | EAGLE-3 | 4.53x | 6.68 | 5.45x | 7.79 | 4.48x | 6.55 | 4.37x | 6.61 | 4.04x | 6.63 | 4.57x | 6.85 |
| | **STEP** | **4.96x** | **8.07** | **6.20x** | **9.81** | **4.97x** | **7.53** | **4.83x** | **7.64** | **4.27x** | **8.27** | **5.05x** | **8.26** |
| L31 8B | DFlash | 2.60x | 4.11 | 2.91x | 4.25 | 3.70x | 5.36 | 2.56x | 3.78 | 2.43x | 3.53 | 2.84x | 4.21 |
| | EAGLE-3 | 4.02x | 6.20 | 4.52x | 6.82 | 4.10x | 6.34 | 4.26x | 6.83 | 3.38x | 5.39 | 4.06x | 6.32 |
| | **STEP** | **4.51x** | **7.01** | **5.00x** | **7.99** | **4.53x** | **7.06** | **4.62x** | **7.35** | **3.71x** | **5.85** | **4.47x** | **7.05** |
| L33 70B | EAGLE-3 | 4.10x | 5.82 | 4.99x | 6.75 | 4.71x | 6.38 | 4.66x | 6.68 | 3.37x | 5.14 | 4.37x | 6.15 |
| | **STEP** | **4.45x** | **6.71** | **6.00x** | **8.53** | **5.13x** | **7.32** | **5.02x** | **7.65** | **3.52x** | **5.65** | **4.82x** | **7.17** |
| DSL 8B | EAGLE-3 | 3.98x | 5.89 | 4.41x | 6.57 | 4.77x | 7.10 | 3.84x | 5.71 | 3.34x | 4.96 | 4.07x | 6.05 |
| | **STEP** | **4.43x** | **6.74** | **4.86x** | **7.31** | **5.62x** | **9.24** | **4.24x** | **6.23** | **3.57x** | **5.38** | **4.54x** | **6.98** |
| Temperature=1 | | | | | | | | | | | | | |
| V 13B | SpS | 1.62x | 1.84 | 1.72x | 1.97 | 1.46x | 1.73 | 1.52x | 1.78 | 1.66x | 1.89 | 1.60x | 1.84 |
| | EAGLE-3 | 3.87x | 5.74 | 4.55x | 6.65 | 3.81x | 5.79 | 3.72x | 5.67 | 3.56x | 5.83 | 3.90x | 5.94 |
| | **STEP** | **4.14x** | **6.66** | **4.95x** | **7.85** | **4.15x** | **6.65** | **4.00x** | **6.09** | **3.90x** | **6.39** | **4.23x** | **6.73** |
| L31 8B | DFlash | 2.06x | 3.10 | 3.04x | 4.45 | 2.32x | 3.42 | 2.08x | 3.08 | 2.94x | 2.01 | 2.49x | 3.21 |
| | EAGLE-3 | 2.87x | 4.33 | 3.90x | 6.13 | 3.32x | 5.01 | 3.40x | 5.68 | 2.75x | 4.50 | 3.25x | 5.13 |
| | **STEP** | **3.18x** | **5.33** | **4.34x** | **7.14** | **3.77x** | **6.17** | **3.73x** | **6.08** | **3.06x** | **4.83** | **3.62x** | **5.91** |
| L33 70B | EAGLE-3 | 3.93x | 5.62 | 4.72x | 6.43 | 4.48x | 6.14 | 4.47x | 6.45 | 3.25x | 5.02 | 4.17x | 5.93 |
| | **STEP** | **4.20x** | **6.35** | **5.39x** | **7.73** | **4.85x** | **6.96** | **4.91x** | **7.47** | **3.42x** | **5.56** | **4.55x** | **6.81** |
| DSL 8B | EAGLE-3 | 3.21x | 4.93 | 3.46x | 5.28 | 4.19x | 6.53 | 2.92x | 4.52 | 2.73x | 4.27 | 3.30x | 5.11 |
| | **STEP** | **3.56x** | **5.64** | **3.71x** | **5.71** | **4.83x** | **8.67** | **3.18x** | **5.01** | **2.97x** | **4.62** | **3.65x** | **5.93** |

AdamW (Loshchilov & Hutter, 2019) with a learning rate of $5 \times 10^{-5}$ and gradient clipping at 0.5. Training and runtime evaluation are conducted on NVIDIA A800 GPUs with 80GB memory per GPU. DFlash, EAGLE-3 and STEP are re-evaluated under the same environment. Other baseline (Fu et al., 2024; Saxena, 2023) results in Table 1 are taken from the literature and included for reference.

**Metrics.** Speedup ratio measures wall-clock decoding speed relative to vanilla autoregressive decoding of the same target model. Average acceptance length $\tau$ measures output tokens committed per drafting–verification cycle. Following EAGLE-3, we also measure acceptance along draft chains conditional on all preceding proposals being accepted. Because STEP uses the unchanged target model and standard speculative verification, it preserves the target model's output distribution; we therefore focus on decoding efficiency.

### 5.2 End-to-End Decoding Efficiency

Table 1 shows consistent improvements over EAGLE-3 across all four target models, five benchmarks, and both temperatures. STEP increases both speedup and $\tau$ in all 40 configurations, yielding average relative gains of 10.1% in decoding speed and 15.1% in $\tau$. The mean relative speed gain is 10.4% at temperature 0 and 9.7% at temperature 1. STEP also substantially outperforms DFlash in the settings where it is evaluated. On LLaMA-3.1-8B at temperature 0, for example, STEP achieves an average speedup of $4.47\times$ and $\tau = 7.05$, compared with $2.84\times$ and $4.21$ for DFlash; the same trend persists at temperature 1. These results show that STEP consistently converts longer accepted continuations into measurable end-to-end runtime gains.

### 5.3 Disentanglement and Alignment Beyond the Training Horizon

We compare EAGLE-3, disentanglement without feature alignment, and STEP on LLaMA-3.1-8B-Instruct using MT-Bench at temperature 0 (Figure 3). All three use a seven-step training horizon ($H = 7$) and are evaluated with 15-step draft chains. At position $k$, we measure $\alpha_k = \Pr(A_k = 1 \mid$

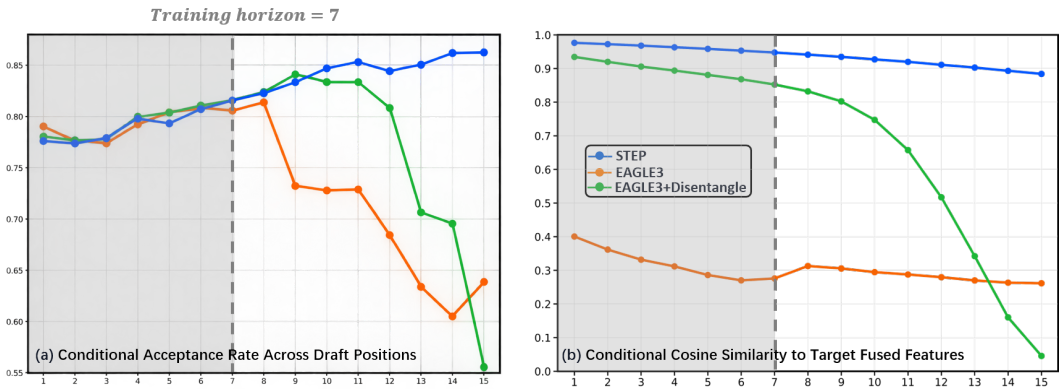


Figure 3: **Conditional long-horizon drafting on MT-Bench with LLaMA-3.1-8B-Instruct.** We compare EAGLE-3, disentanglement only, and STEP in (a) conditional acceptance rate and (b) similarity across draft positions. Both metrics condition on all preceding draft tokens being accepted.

Table 2: **Effect of progressive tree expansion on LLaMA-3.1-8B-Instruct.** Each entry reports speedup ($\times$) / $\tau$. Speedup is relative to target-only autoregressive decoding; $\tau$ is the average number of output tokens committed per drafting–verification cycle. The w/o and w/ settings use original tree construction and progressive expansion, respectively.

| Benchmark | EAGLE-3 w/o | EAGLE-3 w/ | STEP w/o | STEP w/ |
|---|---|---|---|---|
| MT-Bench | 4.02 / 6.20 | 4.19 / 6.76 | 4.19 / 6.39 | **4.51 / 7.01** |
| GSM8K | 4.10 / 6.34 | 4.15 / 6.69 | 4.37 / 6.70 | **4.53 / 7.06** |

$A_1 = \cdots = A_{k-1} = 1$), where $A_k$ denotes a match to the target's greedy prediction. We average $\cos(s_k, f_k)$ over the same eligible prefixes, without requiring acceptance at position $k$. This conditioning excludes prefixes containing an earlier rejected proposal, allowing recurrent-state behavior to be examined without that source of token-prefix divergence.

**Disentanglement improves extended drafting without feature supervision.** Despite having no alignment loss, the disentanglement-only variant maintains substantially higher cosine similarity than EAGLE-3 through the early and intermediate rollout positions (Figure 3b). More importantly, this representation improvement translates into consistently higher conditional acceptance at positions 9–12, with an average gain of about 11 percentage points over EAGLE-3 (Figure 3a). This joint improvement suggests that introducing a dedicated nonlinear vocabulary readout alone already yields recurrent representations that are better suited for extended drafting.

**Adding staged alignment sustains the benefit at later positions.** Disentanglement alone does not maintain feature alignment throughout the evaluated rollout. Its cosine similarity falls from 0.832 at position 8 to 0.045 at position 15, alongside a late decline in conditional acceptance. Disentanglement alone improves the recurrent representation, but it does not explicitly constrain where the trajectory state should evolve in feature space. Adding staged feature alignment yields the full STEP model, which maintains substantially higher similarity and conditional acceptance through position 15 (Figure 3). This ablation shows that the alignment objective complements architectural disentanglement by stabilizing recurrent state propagation and preserving high token-prediction accuracy at later draft positions. Intuitively, the weak alignment loss guides the trajectory state toward the target feature manifold during early training, providing a favorable region for stable recurrent rollout while leaving token prediction as the dominant optimization objective. To verify that this behavior is not specific to conversational data, we further evaluate on GSM8K, a mathematical reasoning benchmark. Appendix Figure 4 shows the same overall trend, with disentanglement improving extended-horizon behavior and staged alignment further stabilizing later draft positions.

Table 3: **Serving throughput in SGLang on MT-Bench and HumanEval with LLaMA-3.1-8B-Instruct.** All measurements use an NVIDIA A800. The baseline reports throughput without speculative decoding; EAGLE-3 and STEP report speedups relative to the baseline at the same batch size.

| Method | 1 | 2 | 4 | 8 | 16 | Mean speedup |
|---|---|---|---|---|---|---|
| *MT-Bench* | | | | | | |
| Baseline (tokens/s) | 89 | 178 | 342 | 630 | 1065 | – |
| EAGLE-3 (60) | 2.78× | 2.42× | 1.93× | 1.30× | 0.85× | 1.86× |
| **STEP (60)** | **2.96×** | **2.49×** | **2.05×** | **1.37×** | **0.91×** | **1.96×** |
| *HumanEval* | | | | | | |
| EAGLE-3 (60) | 3.27× | 2.82× | 2.25× | 1.47× | 0.91× | 2.14× |
| **STEP (60)** | **3.66×** | **3.02×** | **2.50×** | **1.59×** | **1.01×** | **2.36×** |

### 5.4 Drafter Design and Progressive Tree Expansion

Table 2 compares EAGLE-3 and STEP with and without confidence-guided progressive tree expansion on MT-Bench and GSM8K using LLaMA-3.1-8B-Instruct. Without progressive expansion, STEP already outperforms EAGLE-3 on both benchmarks, supporting a runtime benefit from its drafter design. Progressive expansion yields larger speed gains for STEP, further widening its advantage over EAGLE-3. Together with the draft-chain ablations in Section 5.3, these results support the complementarity of STEP's long-horizon drafting capability and progressive tree expansion.

### 5.5 Serving Throughput in SGLang

To assess whether STEP's gains carry over to a production-grade inference framework, we evaluate LLaMA-3.1-8B-Instruct on MT-Bench and HumanEval using SGLang (Zheng et al., 2024) on an NVIDIA A800. We compare target-only autoregressive decoding (baseline), EAGLE-3, and STEP at batch sizes of 1, 2, 4, 8, and 16. Both speculative methods use the same token budget of 60. Table 3 reports baseline throughput and the speedups of EAGLE-3 and STEP relative to the baseline at each batch size. STEP consistently outperforms EAGLE-3 across both benchmarks and all tested batch sizes. On MT-Bench, STEP improves throughput over EAGLE-3 by 2.9%–7.1%, achieving a mean speedup of $1.96\times$ compared with $1.86\times$ for EAGLE-3. On HumanEval, the corresponding improvement is 7.1%–11.9%, with the mean speedup increasing from $2.14\times$ to $2.36\times$.

For both methods, speedup generally decreases as batch size increases, as target-model parallelism reduces the relative benefit of speculative decoding. On MT-Bench, both methods fall below target-only decoding at batch size 16, although STEP remains 7.1% faster than EAGLE-3 ($0.91\times$ versus $0.85\times$). On HumanEval, STEP still slightly exceeds the baseline at the same batch size ($1.01\times$), while EAGLE-3 reaches $0.91\times$. Overall, STEP preserves its throughput advantage over EAGLE-3 across workloads and serving batch sizes.

## 6 Conclusion

This work highlights a broader issue in autoregressive speculative decoding: a learned drafter is not only a local token predictor, but also a recurrent system whose internal states must remain useful under repeated self-conditioning. Our results suggest that strong token prediction within the training horizon does not necessarily imply reliable state evolution beyond it, motivating greater attention to the representations propagated across draft steps. STEP explores this perspective by separating recurrent state propagation from token prediction and using feature alignment as a temporary training guide rather than a permanent reconstruction objective. More broadly, this design suggests that prediction quality and trajectory stability can be treated as distinct objectives when building learned recursive drafters. Our results also indicate that long-horizon modeling and tree construction are complementary: improving the reliability of deeper predictions creates useful drafting capacity, while adaptive expansion determines how efficiently that capacity is exploited. We hope this perspective encourages future work to study state-level behavior in learned drafters, develop objectives that generalize beyond finite training rollouts, and jointly adapt drafting depth and computation budgets to serving conditions.

### AI USE STATEMENT

In this work, generative AI tools were used for language polishing, including improvements to grammar, clarity, conciseness, and presentation, as well as for coding assistance during implementation and debugging. All technical content, analyses, experimental design, and conclusions were developed and verified by authors. All AI-assisted text and code were reviewed and validated by authors, who take full responsibility for the final content of this work.

## REFERENCES

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for Medusa decoding. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=FbhjirzvJG.

Oscar Brown, Zhengjie Wang, Andrea Do, Nikhil Mathew, and Cheng Yu. Dynamic depth decoding: Faster speculative decoding for LLMs, 2024. URL https://arxiv.org/abs/2409.00142.

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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235. PMLR, 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023. URL https://arxiv.org/abs/2302.01318.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding, 2026. URL https://arxiv.org/abs/2602.06036. Accepted at ICML 2026; arXiv camera-ready version.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde 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, Josh 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, 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 *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4116. URL https://proceedings.neurips.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, 2026. URL https://arxiv.org/abs/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. URL https://arxiv.org/abs/2403.09919.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality. LMSYS Org blog, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, 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*, pp. 3029–3051. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.183. URL https://aclanthology.org/2023.emnlp-main.183/.

Cunxiao Du, Jing Jiang, Xu Yuanchen, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, and Yang You. GliDe with a CaPE: A low-hassle method to accelerate speculative decoding. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11704–11720. PMLR, 2024. URL https://proceedings.mlr.press/v235/du24c.html.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783v1. Version 1.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of LLM inference using lookahead decoding. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 14060–14079. PMLR, 2024. URL https://proceedings.mlr.press/v235/fu24a.html.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948. PMLR, 2024a. URL https://proceedings.mlr.press/v235/li24bt.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.emnlp-main.422. URL https://aclanthology.org/2024.emnlp-main.422/.

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*, volume 38, 2025. doi: 10.52202/085713-4562. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c7b5a35ea98b62512a869c19ea7b03cb-Abstract-Conference.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 932–949, 2024. doi: 10.1145/3620666.3651335.

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

Apoorv Saxena. Prompt lookup decoding. GitHub repository, November 2023. URL https://github.com/apoorvumang/prompt-lookup-decoding.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://papers.nips.cc/paper_files/paper/2018/hash/c4127b9194fe8562c64dc0f5bf2c93bc-Abstract.html.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpaca: A strong, replicable instruction-following model. Stanford Center for Research on Foundation Models project report, March 2023. URL https://crfm.stanford.edu/2023/03/13/alpaca.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

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*, volume 36, 2023. doi: 10.52202/075280-2020. URL https://papers.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

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*, volume 37, 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. DistillSpec: Improving speculative decoding via knowledge distillation. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rsY6J3ZaTF.

# A APPENDIX

**Tree-expansion hyperparameters and baseline evaluation.** For the LLaMA-3.1-8B-Instruct results reported in Table 1, we use a shallow exploration depth $d_0 = 7$, a cumulative-confidence threshold $\gamma = 0.4$, an additional deepening depth $d_1 = 5$, a total speculative-token budget $N = 60$, and an expansion width $W = 10$ at each draft step. The same configuration is used across all benchmarks reported for this model, without dataset-specific tuning. Because our experiments are conducted on NVIDIA A800 GPUs, which differ from the hardware setup used in the original EAGLE-3 evaluation, we re-evaluate EAGLE-3 on the same hardware and software environment as STEP rather than directly using its reported runtime results. For EAGLE-3, we follow the corresponding tree-construction and decoding hyperparameters reported in the original work. This provides a matched runtime comparison while avoiding hardware-dependent differences in decoding speed.

**Progressive tree construction.** Algorithm 1 gives the complete procedure for confidence-guided progressive tree expansion. For each branch $b$, we maintain its cumulative confidence in log space, $s(b) = \log C(b)$, together with the cached trajectory state required to continue drafting from that branch. Stage I performs shallow exploration to depth $d_0$. At each level, every active branch proposes its top-$W$ next tokens, and the resulting candidates are ranked by cumulative confidence; the globally highest-confidence branches form the frontier for the next draft step. Importantly, the complete terminal layer of Stage I is retained before frontier pruning. We then select branches whose cumulative confidence exceeds $\gamma$ and use them as the initial frontier of Stage II. Only these branches are recursively extended for up to $d_1$ additional steps, reusing their cached trajectory states and contexts. Finally, all generated candidates are globally ranked under the token budget $N$, and the selected nodes together with their required ancestors are packed into the retrieval paths, tree attention mask, and position indices used for target-model verification.

**Long-horizon behavior on GSM8K.** We further evaluate EAGLE-3, disentanglement only, and STEP on GSM8K using the same conditional long-horizon analysis as in Section 5.3. As shown in Figure 4, the overall trend is consistent with that observed on MT-Bench: disentanglement improves recurrent-state alignment and extended-horizon acceptance over EAGLE-3, while staged feature alignment further stabilizes the trajectory state and preserves prediction quality at later draft positions. These results indicate that the observed long-horizon behavior extends from conversational data to mathematical reasoning.

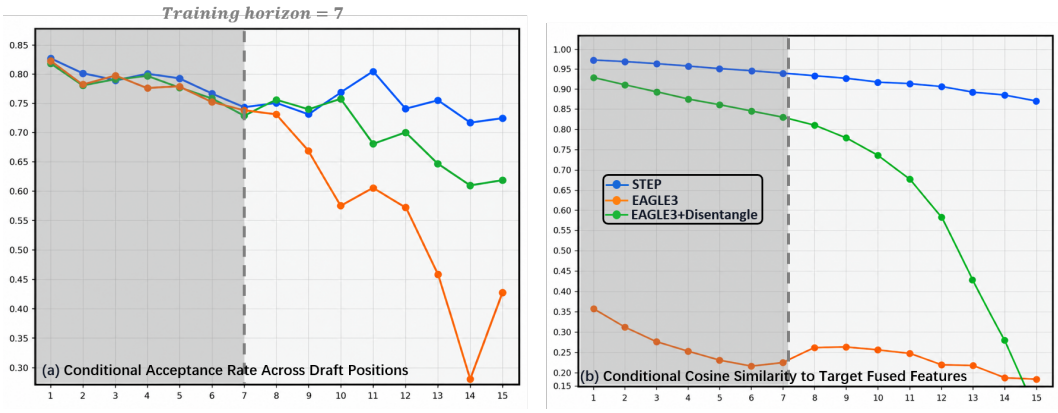


Figure 4: **Conditional long-horizon drafting on GSM8K with LLaMA-3.1-8B-Instruct.** We compare EAGLE-3, disentanglement only, and STEP in (a) conditional acceptance rate and (b) similarity across draft positions. Both metrics condition on all preceding draft tokens being accepted.

**Algorithm 1** Confidence-Guided Progressive Tree Expansion

**Require:** Accepted prefix $x_{1:t}$ and drafter $q_\theta$
**Require:** Shallow token depth $d_0 \geq 2$, additional depth $d_1 \geq 1$
**Require:** Proposal width $W$, threshold $\gamma$, total tree budget $N$
**Ensure:** Draft tokens $\mathbf{y}$, retrieval paths $\mathbf{R}$, tree mask $\mathbf{M}$, and positions $\mathbf{p}$

**function** EXPAND($\mathcal{F}, W$)
  $\{(\pi_b, \xi_b^+)\}_{b\in\mathcal{F}} \leftarrow$ BATCHEDDRAFTSTEP($q_\theta, \mathcal{F}$) ▷ reuse cached branch states
  $\mathcal{G} \leftarrow \varnothing$
  **for all** $b \in \mathcal{F}$ **do**
    **for all** $v \in$ TOKENTOPK($\pi_b, W$) **do**
      $c \leftarrow b \oplus v$
      $s(c) \leftarrow s(b) + \log \pi_b(v)$
      $\mathrm{par}(c) \leftarrow b$ and $\xi_c \leftarrow \xi_b^+$
      $\mathcal{G} \leftarrow \mathcal{G} \cup \{c\}$
  **return** $\mathcal{G}$

**procedure** BUILDTREE($x_{1:t}, q_\theta, d_0, d_1, W, \gamma, N$)
  $(\pi_r, \xi_r) \leftarrow$ PREFILL($q_\theta, x_{1:t}$)
  $s(r) \leftarrow 0$, $\mathcal{V} \leftarrow \varnothing$, and $\mathcal{G} \leftarrow \varnothing$
  **Stage I: shallow exploration**
  **for all** $v \in$ TOKENTOPK($\pi_r, W$) **do**
    Create $c = r \oplus v$ with $s(c) \leftarrow \log \pi_r(v)$
    $\mathrm{par}(c) \leftarrow r$ and $\xi_c \leftarrow \xi_r$
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{c\}$
  $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{G}$
  $\mathcal{F} \leftarrow \mathcal{G}$ ▷ the seed contains exactly $W$ branches
  **for** $\ell = 2, \ldots, d_0$ **do**
    $\mathcal{G} \leftarrow$ EXPAND($\mathcal{F}, W$)
    $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{G}$
    $\mathcal{F} \leftarrow$ GLOBALTOPK($\mathcal{G}, W, s$)
  $\mathcal{F}_0 \leftarrow \mathcal{G}$ ▷ complete, unpruned terminal layer
  $\mathcal{F}_\gamma \leftarrow \{b \in \mathcal{F}_0 : \exp(s(b)) > \gamma\}$
  **Stage II: confidence-guided deepening**
  **if** $\mathcal{F}_\gamma \neq \varnothing$ **then**
    $\mathcal{F} \leftarrow \mathcal{F}_\gamma$
    **for** $j = 1, \ldots, d_1$ **do**
      $\mathcal{G} \leftarrow$ EXPAND($\mathcal{F}, W$)
      $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{G}$
      $\mathcal{F} \leftarrow$ GLOBALTOPK($\mathcal{G}, W, s$)
  **Final budgeted tree construction**
  $m \leftarrow \min\{N - 1, |\mathcal{V}|\}$
  $\mathcal{U} \leftarrow$ GLOBALTOP($\mathcal{V}, m, s$) ▷ ties favor shallower ancestors
  $\mathcal{U} \leftarrow$ SORTBYCREATIONORDER($\mathcal{U}$)
  $(\mathbf{y}, \mathbf{R}, \mathbf{M}, \mathbf{p}) \leftarrow$ PACKTREE($r, \mathcal{U}, \mathrm{par}$)
  **return** $(\mathbf{y}, \mathbf{R}, \mathbf{M}, \mathbf{p})$ ▷ target verification follows