ON THE SPOT: ENHANCING SPECULATIVE DECODING FOR DIFFUSION LANGUAGE MODELS VIA COMMIT-ORDER PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models (DLMs) promise parallel text generation, but in practice face an unresolved speed–quality trade-off. Exact decoding fills in only one masked position per model pass, while faster multi-token decoding schemes are lossy. Speculative decoding is the standard lossless acceleration for autoregressive models. However, it does not transfer directly to DLMs, which must choose both what to generate and where. We show that this order is the missing piece: the model's own predictions from the previous pass already give the right token at 92–100% of positions, but not which position comes next. Based on this observation, we introduce ***S**peculating **P**ositions **O**rder **T**rajectory* (**SPOT**), which learns to predict this order. A 0.62M-parameter model reads only the target's output distributions and proposes a tree of candidate continuations, and the target verifies all of them in a single pass by checking whether each equals its own next state. On the verification side, our analysis shows that prior block-diffusion verification approaches either assume $T{=}0$ or a sequential proposal interface. To support tree-based speculation at any temperature, we propose using coupled-noise match verification for diffusion blocks, following shared-Gumbel-noise coupling approaches. Our comprehensive experiments show that SPOT cuts target passes by up to $6.8\times$ and reaches up to $3.2\times$ the throughput of standard decoding, beating the calibrated and training-free drafting baselines in every task and budget at $T{=}0$ and $T{=}1$. In addition, we show that the learned drafter transfers across models, demonstrating successful transfer across six different DLMs. Code is available at: Anonymous repository.

1 INTRODUCTION

Diffusion language models (DLMs) (Nie et al., 2025; Arriola et al., 2025) generate text by iteratively refining masked token sequences rather than predicting tokens left-to-right. Block-diffusion models make this particularly practical by combining bidirectional denoising within each block with causal generation across blocks (Arriola et al., 2025; Cheng et al., 2026). The diffusion paradigm enables new capabilities such as bidirectional attention, iterative refinement, enhanced safety robustness, and the potential to substantially accelerate language generation via parallel decoding (Ye et al., 2025; Rahimi et al., 2026; Wu et al., 2026). Yet this parallelism comes with a fundamental speed–quality trade-off. High-quality diffusion decoding typically uses conservative unmasking and therefore requires many denoising steps, while faster decoding schemes reveal multiple positions per step using confidence-based, top-k, or dynamic policies, potentially changing the decoding trajectory and degrading generation quality (Xu et al., 2026; Pan et al., 2025; Sun et al., 2026).

For autoregressive language models (AR-LMs), speculative decoding provides an elegant lossless solution. A cheap drafter proposes several future tokens, and the target verifies them in parallel while preserving exactly the distribution of draft-free decoding (Leviathan et al., 2023; Chen et al., 2023). Recent work has begun bringing this idea to DLMs through self-drafting, draft graphs, learned residual predictors, and auxiliary draft models (Gao et al., 2025; Agrawal et al., 2026; Pan et al., 2025; Cui et al., 2026; Xu et al., 2026). However, speculation changes fundamentally once attention inside a block is bidirectional. In an autoregressive model, the next position is fixed, so the drafter only needs to predict *what token comes next*. In a DLM, committing one position changes

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.

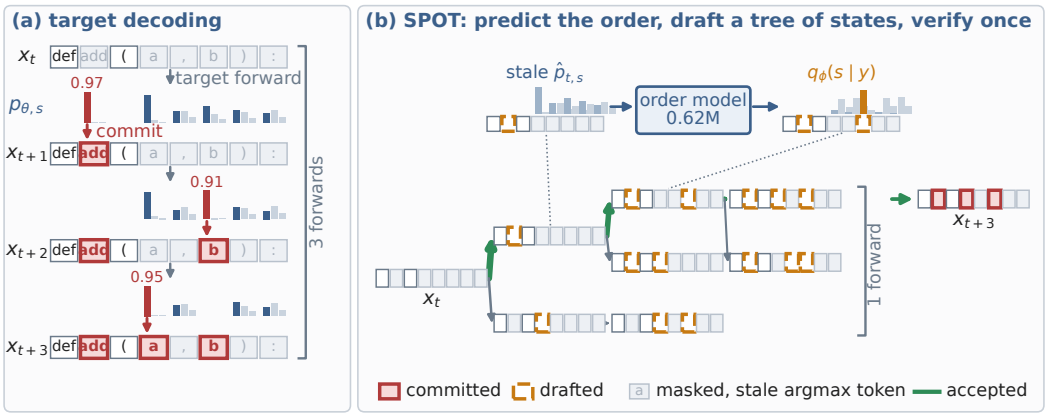


Figure 1: SPOT overview (red: committed; amber dashed: drafted; grey: masked, with its stale argmax token). **(a)** The target commits its most confident position per forward: the tokens are largely known in advance, the *order* is not. **(b)** SPOT reads a hypothetical prefix's stale distributions, predicts the next position with a 0.62M order model, grows a tree of states by prefix mass (edge width), and verifies all of them in one batched forward, committing three positions.

the representations and output distributions of all remaining masked positions in the block. The target must therefore decide both *what token to write* and *where to write it next*. Its future is not a fixed sequence of token positions, but a trajectory through partially filled block states. Several recent DLM speculative-decoding methods exploit this state-based view by proposing hypothetical future states and checking whether the target follows the same transitions (Gao et al., 2025; Agrawal et al., 2026; Pan et al., 2025; Xu et al., 2026). This makes verification possible, but leaves two central questions unresolved: **(Q1)** what information should a drafter predict to speculate several transitions ahead? and **(Q2)** how can a tree of speculative states be verified exactly when the target samples rather than decodes greedily?

Our key finding is that the main difficulty is not token prediction, but *commit-order prediction*. Over the speculative horizons relevant for acceleration, the target's stale distributions already provide strong token hypotheses: their top prediction matches the eventual committed token at 92–100% of measured positions. Predicting which position commits next is considerably harder. This suggests reusing the target's stale token predictions while focusing the drafter on commit order, retaining additional high-probability token hypotheses when confidence is low. Based on this observation, we introduce ***S**peculating **P**ositions **O**rder **T**rajectory* (**SPOT**), a lightweight 0.62M-parameter drafter that predicts future commit order from the target's stale output distributions (Figure 1). SPOT is prefix-conditional: after each hypothetical commitment, it predicts the next position again and organizes these predictions into a tree of candidate block states, prioritizing high-probability branches. The target then evaluates the current and drafted states in a single batched forward and accepts the longest prefix consistent with its trajectory. The second challenge is verifying speculative trees exactly when the target samples. Under greedy decoding, a drafted transition is verified by comparing it with the target's next state. Under sampling, independent draws may disagree even when they come from the same distribution, causing state-equality verification to reject valid proposals. We address this with *coupled-noise verification*: the drafter and target share per-position Gumbel randomness, making the target transition deterministic conditioned on the shared noise. State equality can therefore verify an entire speculative tree, while marginalizing over the noise exactly recovers the target's sampling distribution. This enables exact tree-based verification at arbitrary temperatures without the sequential proposal interface required by classical rejection sampling.

The resulting method substantially reduces both model evaluations and wall-clock decoding time. On SDAR-8B, SPOT uses 15–35% fewer target forwards than Spiffy's calibrated draft graph and achieves up to 1.5× its throughput, with the adaptive-budget configuration reaching up to 2.67×. Relative to standard draft-free decoding, SPOT reduces target passes by up to 6.8× and reaches up to 3.2× throughput. The gains hold across tasks at both greedy decoding and $T{=}1$ sampling. We further evaluate on SDAR-4B, SDAR-1.7B and LLaDA-2.0-mini, and test drafter transfer to two additional targets, LLaDA-8B and Dream-7B, covering six DLMs in total.

Our main contributions are:

- We analyze speculative decoding for diffusion targets along a drafting and a verification axis, show that commit-order prediction is the drafting bottleneck while the target's stale distributions already supply the tokens, show why independent sampling breaks state-equality verification at $T > 0$, and motivate coupling the drafter and verifier through shared randomness.
- Based on this analysis, we introduce SPOT, a 0.62M-parameter prefix-conditional commit-order drafter that builds multi-step trees of block states from stale distributions, paired with coupled-noise verification, which provably preserves the target's distribution at any sampling temperature.
- We evaluate SPOT across tasks, temperatures, block widths and six diffusion language models: it beats calibrated and training-free baselines at $T{=}0$ and $T{=}1$, cuts target passes by up to $6.8\times$, and its checkpoints transfer between models, including LLaDA-8B and Dream-7B, without retraining.

2 Related Work

Speculative decoding and DLMs. Speculative decoding was introduced for autoregressive (AR) language models, where a lightweight drafter proposes tokens that the target verifies in parallel while preserving its distribution (Leviathan et al., 2023; Chen et al., 2023). Later work improved the drafter with auxiliary decoding heads, learned drafters and dynamically built draft trees (Cai et al., 2024; Li et al., 2024b;a; 2025), and a complementary line uses diffusion models as drafters for AR targets (Christopher et al., 2025; Li et al., 2026; Chen et al., 2026; Ringel & Romano, 2026). DLMs instead generate text by denoising (Austin et al., 2021a; Nie et al., 2025; Ye et al., 2025); block-causal models such as SDAR make this practical block by block (Arriola et al., 2025; Cheng et al., 2026), while approximate caching and parallel unmasking accelerate decoding at a cost in fidelity (Wu et al., 2026; Sun et al., 2026). Because attention inside a block is bidirectional, the target decides not only what to write but where, so the commit order is an extra dimension of decoding and speculative decoding does not transfer directly. For verification we build on communication-free coupling: Daliri et al. (2025) show that two parties sampling from P and Q under shared Gumbel noise agree with probability at least $(1 - D_{TV})/(1 + D_{TV})$, and apply this to *drafter-invariant* speculative decoding. We carry the construction to diffusion decoding, where per-*position* shared noise makes the target's sampled commitment at any hypothetical block state a deterministic function of that state (Section 4.3).

Speculative decoding for diffusion targets. SSD (Gao et al., 2025) and Spiffy (Agrawal et al., 2026) study speculative decoding with DLMs targets: SSD orders positions by confidence into a self-drafted chain and verifies it greedily, whereas Spiffy drafts states along an offline-calibrated graph and verifies them blockwise. Both are training-free and greedy. MRP (Xu et al., 2026) learns a residual head that extrapolates the target's logits a few denoising steps ahead and accepts drafted tokens in one pass, greedy only (Section 5.4 sets its reported gains beside ours). BlockSpec (Pan et al., 2025) extends training-free self-speculation to trees on decoding trajectories, and SimSD (Cui et al., 2026) drafts with a same-family sibling model but verifies approximately, so it is not lossless and is not added to our comparison. In contrast, our drafter learns the commit order, builds multi-step trees, verifies exactly under sampling, and transfers across targets without retraining.

3 Analysis of Block-Diffusion Speculative Decoding

Setting. The target decodes a block of L masked positions one commit per forward. A draft is a hypothetical block state with a few more positions filled by the drafter, and D drafts are verified at once by appending each behind the cached prefix as its own copy of the block, so a single forward over $(1{+}D)L$ positions returns the target's next commit at the root and at every draft. A draft is accepted iff it equals that commit at its parent, so the accepted set is one path and the output cannot change. A draft therefore costs a whole block rather than a token: budgets are small, and target forwards (NFE) and wall clock diverge (Appendix D.2). Two measurements on SDAR-8B shape the method: *what* a drafter must predict, and *how* its proposals must be verified once the target samples.

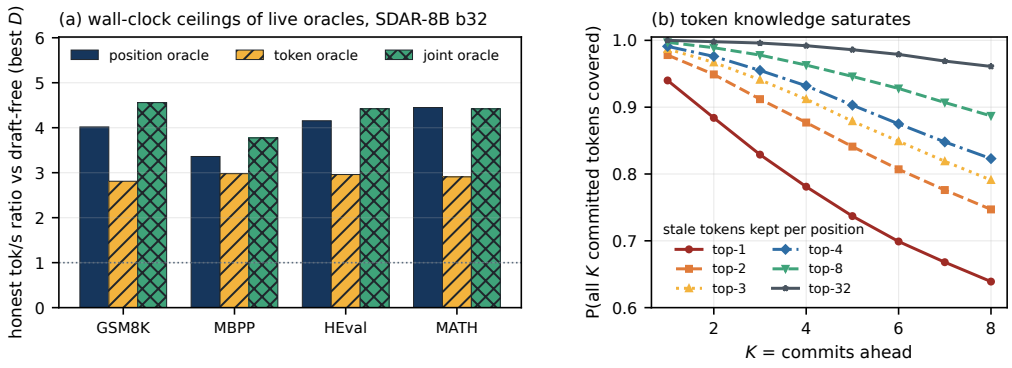


Figure 2: **(a) Live oracle ceilings** on SDAR-8B (b32): peak throughput ratios over draft-free decoding; positions are worth more than tokens, and a perfect drafter reaches 3.8–4.6× (full grid in Table 7). **(b) Stale coverage**: probability that all committed tokens over the next K steps lie within the stale top-t candidates.

3.1 Draft-Trajectory Dynamics

Key Takeaway Knowing the future position order is almost as effective as knowing the full draft trajectory: the commit order carries most of the useful drafting information.

A draft must predict which position the target commits next and what token it writes. To isolate their value we evaluate *live oracles* that peek at the target continuation and propose (i) true *positions* filled with stale-argmax tokens, (ii) true *tokens* ordered by stale confidence, or (iii) both; each bounds the gain obtainable from one information source under standard verification.

Figure 2(a) and Table 7 (Appendix F.2) show that position knowledge dominates on every task and keeps paying across the tested budgets (NFE reduction $5.4\times \rightarrow 7.1\times$ from $D8$ to $D20$ on GSM8K, $5.7\times \rightarrow 8.1\times$ on MBPP), whereas true tokens yield only 3.3–$6.5\times$ at $D20$. The stale distributions already carry token identity: the stale argmax matches the true token in 92% of first-block commits and $\geq$ 99% thereafter, and eight stale candidates per position cover 89% of entire eight-commit continuations (Figure 2(b)). They fail to predict order: the probability that the previous round's top-confidence position commits next drops to 3% by block 9 (Appendix F.9). *What* the model writes is nearly known; *which position* it writes next is not.

NFE does not map directly to wall clock, since every drafted round forwards $(1+D)L$ positions, so we charge each oracle its measured verification cost (lookahead subtracted); under this accounting the joint oracle peaks at 3.8–$4.6\times$ (at an interior budget, $D12$) against an NFE ceiling of 8.0–$9.7\times$. By learning the missing order, SPOT predicts the next commit with 75% accuracy on held-out trajectories (Appendix D.1) against 22% for the stale-confidence order, recovering 70–85% of the position oracle's NFE reduction and 54–81% of the joint wall-clock ceiling (Table 2). Commit-order prediction is therefore the primary information bottleneck for drafting on this target, token knowledge a complementary second axis, and verification cost a separate limit on realized speed.

3.2 Verification Dynamics Across Sampling Temperatures

Key Takeaways

- **Exactness.** Coupled and uncoupled matching preserve the target sampling distribution, whereas greedy matching increasingly drifts as temperature rises.
- **Acceptance.** Coupling becomes increasingly important at higher temperatures, where independent matching rapidly loses acceptance.
- **High-temperature regime.** Target quality deteriorates sharply beyond $T\approx1.5$ and becomes the dominant limitation.

Table 1: Temperature sweep on SDAR-8B MBPP. Draft-free columns: target accuracy, argmax fraction, mean committed-token probability. SPOT $D8$: NFE and throughput against the T-matched draft-free control. Greedy-match columns show distribution drift. The $D8$ NFE standard deviation over seeds grows from 0.3 at T=0.5 to 8.5 at T=2.5. †Greedy match is not distribution-preserving, so its NFE is not comparable.

	draft-free ($D0$)			SPOT $D8$ NFE (tok/s ratio)			greedy SPOT		Spiffy $D8$ NFE	
T	acc	argmax	mean p	coupled	uncoupled	greedy†	acc	argmax	coupled	uncoupled
0.5	0.35	0.986	0.993	58.2 (2.56)	58.2 (2.57)	57.6 (2.57)	0.36	0.978	84.3	84.0
1.0	0.34	0.976	0.965	60.1 (2.45)	60.2 (2.43)	56.8 (2.55)	0.38	0.973	86.3	86.6
1.5	0.08	0.866	0.828	74.0 (1.97)	82.6 (1.77)	57.4 (2.55)	0.28	0.958	94.2	108.9
2.0	0.00	0.057	0.052	187.4 (0.78)	256.7 (0.54)	94.7 (1.52)	0.01	0.750	114.6	258.5
2.5	0.00	0.001	0.000	200.8 (0.71)	269.1 (0.54)	128.1 (1.13)	0.00	0.582	110.5	269.2

At T=0 the acceptance test above is *state equality* against a deterministic commit. At T>0 the commitment is a random draw, and there are three ways to keep verifying by equality: a *greedy match* accepts a drafted state iff it equals the target's *argmax* commitment, as the T=0 verifier would if simply run at temperature; an *uncoupled match* lets the target sample independently and accepts only if the sample happens to equal the proposal; a *coupled-noise match* shares the sampler's Gumbel noise between drafter and verifier, so that the target's draw is again a deterministic function of the state (Section 4.3). We ran all three on SPOT, Spiffy and the BlockSpec-style tree at $T \in \{0.5, 1, 1.5, 2, 2.5\}$ (Table 1, Appendix F.7).

(i) Coupled and uncoupled match are exact; greedy match is not. In both exact regimes the emitted tokens track the draft-free statistics within sampling noise: over all three drafters at $D8$ and $D20$ and $T \leq 1.5$ (Table 22), the emitted argmax fraction stays within 0.016 of the draft-free value, the mean committed-token probability within 0.015, and accuracy within 0.04. Greedy match drifts, and the drift grows with T: at T=1.5 it emits accuracy 0.28 where the model samples 0.08 and picks the argmax at 95.8% of commits against the model's 86.6%; the drift is proportional to acceptance, so it makes the *best* drafter the *most* biased. **(ii) Coupling keeps acceptance alive as the target's samples move away from its argmax.** At $T \leq 1$ the sampled token is the argmax 98–99% of the time and the two verifiers agree within 1% in NFE; at T=1.5 uncoupled costs 12–16% more NFE for SPOT and Spiffy (74.0 → 82.6 and 94.2 → 108.9); at T=2 uncoupled accepts almost nothing (256.7 NFE against 269.2 draft-free) while coupled still drafts. The mechanism is the coupling bound of Daliri et al. (2025): a stale-informed proposal survives with probability at least $(1-\delta)/(1+\delta)$ in the total variation δ between stale and fresh conditionals, whereas an independent sample equals the argmax only with probability $p_{\max}$. **(iii) At high temperatures, target quality becomes the limiting factor.** Draft-free accuracy already drops sharply to 0.08 at T=1.5 and reaches 0.00 at $T \geq 2$, leaving little useful generation quality to preserve.

4 METHOD

4.1 SETUP

Let p_θ be a block-causal diffusion language model (Cheng et al., 2026; Bie et al., 2025). Within an active block of length L, a state $x \in \mathcal{X} = (\mathcal{V} \cup \{\text{M}\})^L$ assigns each position either a token in vocabulary $\mathcal{V}$ or the mask symbol M, and $\mathcal{M}(x) = \{s : x_s = \text{M}\}$ denotes its masked positions. Given the completed causal context C, one target forward returns $p_{\theta,s}(\cdot \mid x, C)$ for every $s \in \mathcal{M}(x)$. Let $s^\star(x) = \arg\max_{s \in \mathcal{M}(x)} \max_{v \in \mathcal{V}} p_{\theta,s}(v \mid x, C)$ be the highest-confidence masked position and $x \oplus (s, v)$ be the state obtained by filling position s with token v. At $T = 0$, the target commits $v_0^\star(x) = \arg\max_v p_{\theta,s^\star(x)}(v \mid x, C)$, giving the deterministic transition

$$\Phi_0(x) = x \oplus \big(s^\star(x), v_0^\star(x)\big). \tag{1}$$

At $T > 0$, the committed token is instead sampled from the temperature-adjusted conditional

$$\pi_{T,s}(v \mid x, C) = \frac{p_{\theta,s}(v \mid x, C)^{1/T}}{\sum_{u \in \mathcal{V}} p_{\theta,s}(u \mid x, C)^{1/T}}. \tag{2}$$

A *drafter* $\mathcal{D}$ proposes a finite tree of hypothetical future states rooted at the current state, with each edge representing one slot–token commitment. The target evaluates the root and all drafted states in one batched forward. The verifier accepts an edge (y, z) only when z equals the target's own next state from y, which we call *state-equality verification*; because the target has a unique next commitment, the accepted subset of a draft tree is always a single path. Let X_T^{DF} and $X_{T,\mathcal{D}}^{\mathrm{Spec}}$ denote the draft-free and speculative committed trajectories. We call a verifier *distribution preserving* when $X_{T,\mathcal{D}}^{\mathrm{Spec}} \overset{d}{=} X_T^{\mathrm{DF}}$.

4.2 The drafter: learned commit order

Order model. At speculative round t, let $\widehat{p}_{t,s}(\cdot)$ denote the most recently available target conditional at masked position s. These conditionals are *stale*: hypothetical future commitments change the true target distribution, but no additional target forward is performed while drafting. Our drafter is a three-layer transformer encoder ($d_{\mathrm{model}} = 128$, 8 heads, 0.62M parameters). Each masked position is represented using its stale top-32 probabilities (raw and log), eight summary statistics, and two path features encoding the positions already committed along the hypothetical prefix and its depth. Given a hypothetical state y, the model predicts $q_\phi(s \mid y)$, where $s \in \mathcal{M}(y)$, a distribution over the target's next committed position. The same prefix-conditional predictor is learned in Algorithm 1 and recursively queried during Algorithm 2. The drafter uses no target hidden states, token embeddings, or text, and can therefore be easily evaluated across different DLMs.

Training. We train q_ϕ entirely from draft-free target trajectories (Algorithm 1). Let $(s_t^\star, s_{t+1}^\star, \ldots)$ denote the target commit sequence following decoding state x_t. For each prefix depth d, the first d commitments define a hypothetical state $y_{t,d}$ while the stale conditionals $\widehat{p}_t$ remain fixed; the next target commitment $s_{t+d}^\star$ is used as supervision. This produces prefix-conditioned examples $(\widehat{p}_t, y_{t,d}, s_{t+d}^\star)$ that match the states encountered during recursive drafting. We optimize $\mathcal{L}_{\mathrm{order}}(\phi) = -\mathbb{E}_{(\widehat{p},y,s^\star)\sim\mathcal{E}} \log q_\phi(s^\star \mid y)$, where $\mathcal{E}$ is the resulting training set. Full data-generation and optimization details are given in Appendix D.1.

Inference. At inference (Algorithm 2), drafting starts from the current target state x_t and stale conditionals $\widehat{p}_t$. For every candidate prefix y, the order model scores the remaining positions using $q_\phi(\cdot \mid y)$, and each position is paired with token hypotheses from the stale target conditional. Let $\mathcal{H}_t(s)$ denote the token-hypothesis set for position s. At $T = 0$, its primary hypothesis is $\widetilde{v}_0(s) = \arg\max_v \widehat{p}_{t,s}(v)$, with the runner-up additionally included when the stale top-1 probability is below 0.6. At $T > 0$, $\mathcal{H}_t(s)$ uses the coupled stale draw of Eq. 4, introduced formally in Section 4.3.

For a candidate child $z = y \oplus (s, v)$, define its *prefix mass* recursively as

$$m(z) = m(y)\, q_\phi(s \mid y)\, \widehat{p}_{t,s}(v), \qquad m(x_t) = 1. \tag{3}$$

Algorithm 2 uses DDTree, a level-batched best-first search adapted from Ringel & Romano (2026), to allocate budget D to the highest-mass candidate states. Since every factor in Eq. 3 lies in $[0, 1]$, a child's mass cannot exceed its parent's, allowing selected nodes to be expanded level-by-level while recovering the same top-D fixpoint as sequential best-first search. A calibration temperature τ is applied to q_ϕ before search; the final model uses $\tau = 1$.

Adaptive budget and padding. Algorithm 2 treats D as a hard upper bound. We additionally apply a prefix-mass floor γ, discarding candidates with $m(z) < \gamma$ and thereby adapting the realized verification budget to drafter confidence. Independently, the target batch may be padded with dummy replicas to maintain a fixed $(1 + D)L$ shape and reduce batch-shape-dependent numerical variation. We report the padded $\gamma = 0$ configuration as SPOT and the unpadded configuration with a floor calibrated once per target ($\gamma = 0.1$ on SDAR-8B; Appendix D.5) as SPOT-γ. The complete budget, mass-floor, and padding ablations are given in Section 5.5.

4.3 Diffusion-block verification by coupled-noise match

Greedy verification. At $T = 0$, the target transition Φ_0 in Eq. 1 is deterministic. Therefore, a drafted edge (y, z) is accepted exactly when $z = \Phi_0(y)$. Proposition 1 is elementary and included both for completeness and as a basis for the proof of Proposition 2.

Proposition 1 (Greedy exactness)**.** *For any drafter $\mathcal{D}$ and any finite chain or tree draft, state-equality verification at $T = 0$ produces exactly the draft-free target trajectory: $X^{\mathrm{Spec}}_{0,\mathcal{D}} = X^{\mathrm{DF}}_0$.*

Sampling with shared Gumbel noise. At $T > 0$, state equality cannot use independent drafter and verifier samples: even identical distributions would then produce different tokens with nonzero probability. Instead, we couple their sampling through shared randomness. For every absolute output position a and vocabulary item v, draw $G_{a,v} \overset{\text{iid}}{\sim} \mathrm{Gumbel}(0, 1)$, and let $G = \{G_{a,v}\}_{a,v}$ denote the resulting table. The table is shared by drafter and verifier and indexed by absolute position, so the same position uses the same Gumbel row in both.

For target state x, the target draw is $v^\star_G(x) = \arg\max_{v\in\mathcal{V}} \left[\frac{1}{T}\log p_{\theta,s^\star(x)}(v \mid x, C) + G_{a(s^\star(x)),v}\right]$, which defines the transition $\Phi_G(x) = x \oplus \big(s^\star(x), v^\star_G(x)\big)$. By the Gumbel–argmax identity, $v^\star_G(x) \sim \pi_{T,s^\star(x)}(\cdot \mid x, C)$. The drafter applies the *same* noise to its stale conditional:

$$\widetilde{v}_G(s) = \arg\max_{v\in\mathcal{V}} \left[\frac{1}{T}\log \widehat{p}_{t,s}(v) + G_{a(s),v}\right]. \tag{4}$$

Thus, whenever the stale and fresh conditionals have the same Gumbel-perturbed argmax, drafter and target propose exactly the same token. Conditioned on G, Φ_G is deterministic, so verification again reduces to state equality: $z = \Phi_G(y)$.

Proposition 2 (Coupled-noise exactness)**.** *For any drafter $\mathcal{D}$ and $T > 0$, the Gumbel-driven draft-free trajectory satisfies $X^{\mathrm{DF}}_T(G) \overset{d}{=} X^{\mathrm{DF}}_T$, and for every realization g of G, $X^{\mathrm{Spec}}_{T,\mathcal{D}}(g) = X^{\mathrm{DF}}_T(g)$. Consequently, $X^{\mathrm{Spec}}_{T,\mathcal{D}} \overset{d}{=} X^{\mathrm{DF}}_T$.*

Propositions 1 and 2 therefore separate *correctness* from *draft quality*: the verifier always preserves the target trajectory or distribution, while the drafter determines only how many future target transitions are already present in the tree and can be accepted in one target forward. The complete formal setup and proofs are given in Appendix A. For a drafted position s, let $\widehat{\pi}_{T,s}$ and $\pi_{T,s}$ denote the stale and fresh temperature-adjusted conditionals. The shared-Gumbel coupling of Daliri et al. (2025) satisfies $\Pr[\widetilde{v}_G(s) = v^\star_G(s)] \geq \frac{1-\delta_T}{1+\delta_T}$, where $\delta_T = D_{\mathrm{TV}}(\widehat{\pi}_{T,s}, \pi_{T,s})$. Hence coupled verification becomes more effective as the stale and fresh sampling distributions become closer.

Relation to rejection sampling. When the drafter exposes an explicit sequential proposal distribution, exact verification can alternatively be performed by rejection sampling. Appendix B formalizes this alternative, proves its exactness and maximal per-link acceptance probability $1 - D_{\mathrm{TV}}(\pi_k, q_k)$, and validates its implementation. Classical per-link RS assumes a sequential proposal interface, whereas coupled-noise match requires only state equality and therefore applies directly to our batched draft trees.

5 Experiments

We establish that SPOT takes fewer target forwards with higher throughput than every baseline at T=0 and T=1 (Section 5.2 and Section 5.3), then a checkpoint transfers across targets without additional training (Section 5.4), and finally each component of the drafter earns its place (Section 5.5).

5.1 Experimental setup

Our headline target is **SDAR-8B** (Cheng et al., 2026), a dense block-causal diffusion LM at block width L=32 with its exact prefix KV cache. The same recipe gives drafters for **SDAR-4B** and **SDAR-1.7B** and for the mixture-of-experts **LLaDA2.0-mini** (Bie et al., 2025), and transfer is measured on two further targets, **LLaDA-8B** (Nie et al., 2025) and **Dream-7B** (Ye et al., 2025): six diffusion language models in all. We evaluate GSM8K (Cobbe et al., 2021), MBPP (Austin et al., 2021b), HumanEval (Chen et al., 2021) and MATH500 (Hendrycks et al., 2021) on n=350 samples (all 164 for HumanEval) over three seeds, except where a caption says otherwise, at generation length 256 and budgets $D \in \{8, 12, 20\}$ and report NFE per sample, accuracy, and a *throughput ratio* against the draft-free run of the same task, model and temperature. SPOT is one checkpoint

Table 2: Main results, SDAR-8B b32 at $T{=}0$ and $T{=}1$: NFE per sample (throughput ratio over the shared draft-free control), best in bold, second best underlined. Shaded rows give each block's accuracy range. SPOT-γ: unpadded, $\gamma{=}0.1$. At $T{=}1$ all methods use coupled-noise match except the free chain (RS). Over seeds the ratio's standard deviation is at most 0.15 and NFE's at most 1.3.

NFE↓ (tok/s↑)	GSM8K		MBPP		HumanEval		MATH500	
method	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$
$T{=}0$	acc. 0.89–0.90		acc. 0.41–0.43		acc. 0.74–0.74		acc. 0.11–0.11	
no draft ($D0$)	270.8↓ (1.00↑)		269.2↓ (1.00↑)		271.4↓ (1.00↑)		271.8↓ (1.00↑)	
SPOT (ours)	**69.6** (2.50)	**52.6** (1.56)	**56.7** (2.61)	**39.6** (1.72)	**62.2** (2.74)	**44.6** (1.85)	**84.5** (2.03)	**66.0** (1.20)
SPOT-γ (ours)	75.1 (**2.76**)	67.8 (**2.78**)	60.7 (**2.98**)	47.8 (**2.99**)	66.5 (**3.09**)	56.4 (**3.13**)	92.2 (**2.27**)	89.5 (**2.27**)
Spiffy graph	91.3 (1.96)	71.9 (1.14)	83.4 (1.86)	61.3 (1.17)	86.8 (2.11)	65.9 (1.28)	99.4 (1.80)	82.7 (0.99)
SSD free chain	92.2 (1.95)	84.9 (1.00)	84.7 (1.80)	76.1 (0.91)	87.0 (2.08)	80.2 (1.03)	100.5 (1.80)	93.6 (0.88)
BlockSpec tree	136.4 (1.33)	114.6 (0.71)	135.0 (1.10)	111.5 (0.58)	136.0 (1.34)	112.7 (0.73)	136.7 (1.29)	116.1 (0.68)
$T{=}1$	acc. 0.87–0.90		acc. 0.34–0.38		acc. 0.66–0.72		acc. 0.10–0.12	
SPOT (ours)	**72.7** (2.46)	**54.8** (1.55)	**60.1** (2.45)	**42.6** (1.59)	**65.1** (2.73)	**47.4** (1.74)	**87.6** (1.97)	**68.9** (1.18)
SPOT-γ (ours)	78.4 (**2.67**)	71.0 (**2.66**)	64.9 (**2.80**)	52.1 (**2.86**)	69.5 (**3.04**)	59.3 (**3.09**)	95.4 (**2.16**)	92.7 (**2.16**)
Spiffy graph	94.3 (2.03)	75.5 (1.20)	86.3 (1.77)	64.8 (1.07)	89.4 (2.16)	68.9 (1.31)	103.5 (1.72)	87.7 (0.93)
free chain, RS	90.0 (1.54)	84.4 (0.85)	64.5 (1.86)	52.4 (1.12)	78.0 (1.84)	70.4 (1.05)	116.9 (1.14)	115.8 (0.60)
BlockSpec tree	139.3 (1.27)	118.2 (0.68)	138.3 (1.07)	115.2 (0.57)	138.4 (1.34)	115.5 (0.72)	140.7 (1.28)	121.0 (0.68)

and one calibration per target, and SPOT-γ runs it unpadded with that floor. Baselines are Spiffy's calibrated draft graph (Agrawal et al., 2026), an SSD-style confidence-ordered free chain (Gao et al., 2025) and a BlockSpec-style confidence tree (Pan et al., 2025), all sharing our decoder and verifier so that the comparison is about drafting alone. Other block widths, temperatures, and generation lengths, and the full protocol, are in Appendices D, E and F.

5.2 Main results

In the upper block of Table 2, SPOT delivers 3.2–4.7× fewer target forwards than draft-free decoding at $D8$ and 4.1–6.8× at $D20$, at 2.0–2.7× single-device wall at $D8$; against Spiffy's calibrated graph it has 15–35% fewer NFE and 1.1–1.5× its throughput in every task-and-budget setting, with larger margins against the chain and the tree. The baselines' throughput collapses with budget where their NFE does not: the tree is *slower than draft-free* at $D20$ on every task and the chain on three of four, since a wider draft costs $(1{+}D)L$ positions whether or not it is accepted, while they accept only 1.1–4.0 tokens per round against SPOT's 4.5–7.7 (Appendix F.9). SPOT-γ removes the budget dependence: its throughput ratio is flat in the budget (2.3–3.2× at $D8$, $D12$ and $D20$ alike) and at or above the padded recipe in every SDAR-8B setting at both temperatures. The floor pays 4–24 NFE per setting, so SPOT stays NFE-optimal while SPOT-γ still has fewer forwards than every baseline in 21 of 24 settings. MRP (Xu et al., 2026), the closest learned drafter, reports 1.17–1.40× over one-token-per-step decoding on these same targets in a different implementation, against SPOT's 2.0–3.6× at $D8$, a margin wider than that difference can account for, and its argmax-equality acceptance has no distribution-preserving extension to sampling (Table 5).

5.3 Sampling temperature: coupled-noise match against the baselines

The lower block of Table 2 shows that nothing changes when the target samples at $T{=}1$: under the same verifier, SPOT keeps 15–34% fewer NFE than Spiffy and 1.15–1.48× its throughput. Free chain under rejection sampling is the strongest baseline on MBPP NFE, but stays 0.47–0.92 below SPOT in throughput, since every accepted link pays an acceptance test, and collapses on MATH500 (116 NFE, 0.60× at $D20$). Every $T{=}1$ result is distribution-preserving, with accuracies within 0.06 of the control over seeds; tree + coupled-noise match is the deployment configuration at any temperature (Appendix F.8). The ratios grow with generation length: at 512 SPOT reaches 2.79× draft-free at $D8$ and SPOT-γ 3.23×, against 2.45× and 2.86× at 256 (Appendix F.5). SPOT is also exact under whichever commit rule the target runs. Swept against the lossy threshold decoder of Wu et al. (2026) across its confidence bar, it composes with the rule without changing its accuracy, is 1.2–1.4× faster than that decoder tuned to the same quality on GSM8K, MBPP and HumanEval, and leads Spiffy at every bar (Figure 3, Appendix F.6).

Table 3: Checkpoint transfer (T=0, b32, $D8$, padded): each checkpoint pointed at a target with no retraining or recalibration, NFE per sample, the better drafter in bold. Both drafters run the same target weights over the same $(1+D)L$ positions per forward, so throughput tracks NFE (ratios in Table 14). †approximate prefix cache of Wu et al. (2026), not distribution-preserving.

NFE↓ at $D8$	GSM8K		MBPP		HumanEval		MATH500	
target \ drafter	SDAR-8B	mini	SDAR-8B	mini	SDAR-8B	mini	SDAR-8B	mini
SDAR-8B	**69.6**	72.8	**56.7**	58.6	**62.2**	66.8	**84.5**	87.9
LLaDA2.0-mini	85.9	**83.0**	105.2	**94.9**	97.8	**88.2**	87.1	**84.8**
LLaDA-8B†	**79.8**	83.1	**54.5**	56.7	**83.3**	87.6	**83.6**	86.3
Dream-7B†	**66.0**	67.8	**51.2**	52.0	**57.3**	58.3	**77.2**	79.1

5.4 Transferability within and across model families

Transfer takes two forms. Re-running identical training recipe on **SDAR-4B** reproduces the SDAR-8B picture, lowest NFE and highest throughput in every comparison at T=0 and T=1, and on **LLaDA2.0-mini**, a mixture-of-experts target with a different tokenizer, it gives the lowest NFE across 36 comparisons and the higher throughput in 34. On **SDAR-1.7B** SPOT keeps the lowest NFE everywhere but cedes three throughput columns at T=0 and four at T=1 to Spiffy, where a cheap target forward makes the padded draft much of the step. The method carries to block width 16 untuned (Appendices F.3 and F.4). Table 3 swaps the trained checkpoints between targets, tokenizers included, with no retraining or recalibration, and Table 14 gives the full six-target, four-drafter matrix: the transferred drafter costs 3–7% NFE on SDAR-8B and *still beats every baseline calibrated on SDAR-8B* (Table 2), and 3–11% on mini, where it stays ahead of every baseline on GSM8K and MATH500. Beyond the block-causal family, **LLaDA-8B** (Nie et al., 2025) and **Dream-7B** (Ye et al., 2025) attend over the sequence, so drafted runs and control both use the approximate cache of Wu et al. (2026) (Appendix D) and these rows measure reach rather than losslessness: checkpoints that never saw either model cut LLaDA-8B's 256 forwards to 55–88 and Dream-7B's to 51–79: the order model learns *where a diffusion target commits*, and much of that carries.

5.5 Ablations

Table 18 (Appendix F.7) varies one component on SDAR-8B MBPP. The largest effect comes from replacing the tree with a chain: NFE increases by 7.3 at $D8$ and by 11.9 at $D20$, while ratio drops from 1.72 to 1.34. Budget therefore pays through branching, not a longer chain. The token hedge and the scaled corpus are each worth under one NFE, and padding moves throughput by at most 0.04. The mass floor trades NFE for wall-clock speed, but only without padding: at $D20$, $\gamma = 0/0.05/0.1$ gives $39.6/44.3/47.8$ NFE at $1.76/2.84/2.99\times$ wall, by computing fewer verification positions.

6 Discussion

DLM decoding remains constrained by a fundamental speed–quality trade-off. SPOT addresses this challenge with a two-fold solution. We show that stale target distributions already contain highly reliable token information, making commit order the key missing signal for effective speculation. A lightweight 0.62M-parameter model predicts this order and constructs multi-step draft trees, while coupled-noise verification preserves the target distribution under sampling. SPOT recovers 54–81% of the cost-aware oracle ceiling, uses 15–35% fewer target forwards than an offline-calibrated draft graph at up to $1.5\times$ its throughput, and transfers across six DLMs without retraining. More broadly, our results show that substantially faster DLM generation can be achieved without sacrificing the quality or exactness of the underlying decoding process, establishing commit-order prediction as a promising foundation for the next generation of speculative decoding methods for DLMs.

Limitations. The main limitation of SPOT is that its gains diminish when verification is costly or acceptance is low. Verification becomes the bottleneck at wide blocks, while exact tree packing remains unresolved under bidirectional attention. Gains also narrow on low-acceptance tasks, and at $T \geq 2$ the T=0-trained order model is outperformed by noise-aware ranking (Appendix F.7).

AI USE STATEMENT

Generative AI tools were used during manuscript preparation to support language refinement and editing, and during development to assist with implementation of the experimental codebase. They were not used to formulate the research questions, develop the proposed method, define the evaluation procedure, or draw conclusions from the experimental findings. The quantitative results reported in this paper originate from the experiments described in Section 5.1 and the appendix. The authors inspected and verified all AI-assisted contributions, including code incorporated into the experimental workflow, and remain fully responsible for the manuscript, implementation, experimental results, and conclusions.

ETHICS STATEMENT

This work focuses on accelerating inference in diffusion large language models and does not involve human subjects or the collection of personal data. We do not anticipate any ethical concerns beyond those associated with the underlying language models and their downstream applications.

REPRODUCIBILITY STATEMENT

Code, and the 0.62M drafter checkpoint for SDAR-8B as an example, are all available in anonymized repository along with detailed explanations on how to run what's needed to reproduce. Every number in this paper comes from one benchmark run on one machine (eight identical NVIDIA RTX 6000 Ada cards; one job per card), with the protocol of Section 5.1 and Appendix D. https://anonymous.4open.science/r/spot-speculative-decoding-E354

REFERENCES

Sudhanshu Agrawal, Risheek Garrepalli, Raghavv Goel, Christopher Lott, Fatih Porikli, and Mingu Lee. Structuring The Future: Diffusion LLM Speculative Decoding via Calibrated Draft Graphs, June 2026. URL http://arxiv.org/abs/2509.18085. arXiv:2509.18085 [cs.LG].

Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models. *International Conference on Learning Representations*, 2025:50726–50753, May 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/7ede97c3e082c6df10a8d6103a2eebd2-Abstract-Conference.html.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured Denoising Diffusion Models in Discrete State-Spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993. Curran Associates, Inc., 2021a. URL https://proceedings.neurips.cc/paper/2021/hash/958c530554f78bcd8e97125b70e6973d-Abstract.html.

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, August 2021b. URL http://arxiv.org/abs/2108.07732. arXiv:2108.07732 [cs.PL].

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, Gerald Shen, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Yoshi Suhara, Olivier Delalleau, Zijia Chen, Zhilin Wang, David Mosallanezhad, Adi Renduchintala, Haifeng Qian, Dima Rekesh, Fei Jia, Somshubra Majumdar, Vahid Noroozi, Wasi Uddin Ahmad, Sean Narenthiran, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Igor Gitman, Ivan Moshkov, Wei Du, Shubham Toshniwal, George Armstrong, Branislav Kisacanin, Matvei Novikov, Daria Gitman, Evelina Bakhturina, Prasoon Varshney, Makesh Narsimhan, Jane Polak Scowcroft, John Kamalu, Dan Su, Kezhi Kong, Markus Kliegl, Rabeeh Karimi Mahabadi, Ying Lin, Sanjeev Satheesh, Jupinder Parmar, Pritam Gundecha, Brandon Norick, Joseph Jennings, Shrimai Prabhumoye, Syeda Nahida

Akter, Mostofa Patwary, Abhinav Khattar, Deepak Narayanan, Roger Waleffe, Jimmy Zhang, Bor-Yiing Su, Guyue Huang, Terry Kong, Parth Chadha, Sahil Jain, Christine Harvey, Elad Segal, Jining Huang, Sergey Kashirsky, Robert McQueen, Izzy Putterman, George Lam, Arun Venkatesan, Sherry Wu, Vinh Nguyen, Manoj Kilaru, Andrew Wang, Anna Warno, Abhilash Somasamudramath, Sandip Bhaskar, Maka Dong, Nave Assaf, Shahar Mor, Omer Ullman Argov, Scot Junkin, Oleksandr Romanenko, Pedro Larroy, Monika Katariya, Marco Rovinelli, Viji Balas, Nicholas Edelman, Anahita Bhiwandiwalla, Muthu Subramaniam, Smita Ithape, Karthik Ramamoorthy, Yuting Wu, Suguna Varshini Velury, Omri Almog, Joyjit Daw, Denys Fridman, Erick Galinkin, Michael Evans, Shaona Ghosh, Katherine Luna, Leon Derczynski, Nikki Pope, Eileen Long, Seth Schneider, Guillermo Siman, Tomasz Grzegorzek, Pablo Ribalta, Monika Katariya, Chris Alexiuk, Joey Conway, Trisha Saar, Ann Guan, Krzysztof Pawelec, Shyamala Prayaga, Oleksii Kuchaiev, Boris Ginsburg, Oluwatobi Olabiyi, Kari Briski, Jonathan Cohen, Bryan Catanzaro, Jonah Alben, Yonatan Geifman, and Eric Chung. Llama-Nemotron: Efficient Reasoning Models, September 2025. URL http://arxiv.org/abs/2505.00949. arXiv:2505.00949 [cs.CL].

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. LLaDA2.0: Scaling Up Diffusion Language Models to 100B, December 2025. URL http://arxiv.org/abs/2512.15745. arXiv:2512.15745 [cs.LG].

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, June 2024. URL http://arxiv.org/abs/2401.10774. arXiv:2401.10774 [cs.LG].

Sahil Chaudhary. Code Alpaca: An Instruction-following LLaMA model for code generation, 2023. URL https://github.com/sahil280114/codealpaca. Publication Title: GitHub repository.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating Large Language Model Decoding with Speculative Sampling, February 2023. URL https://arxiv.org/abs/2302.01318v1.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block Diffusion for Flash Speculative Decoding, May 2026. URL http://arxiv.org/abs/2602.06036. arXiv:2602.06036 [cs.CL].

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, July 2021. URL http://arxiv.org/abs/2107.03374. arXiv:2107.03374 [cs.LG].

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A Synergistic Diffusion-AutoRegression Paradigm for Scalable Sequence Generation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 22058–22075, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1110. URL https://aclanthology.org/2026.findings-acl.1110/.

Jacob K Christopher, Brian R. Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. Speculative Diffusion Decoding: Accelerating Language Generation through Diffusion. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 12042–12059, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.601. URL https://aclanthology.org/2025.naacl-long.601/.

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, November 2021. URL http://arxiv.org/abs/2110.14168. arXiv:2110.14168 [cs.LG].

Junxia Cui, Haotian Ye, Runchu Tian, Hongcan Guo, Jinya Jiang, Haoru Li, Chaojie Ren, Yiming Huang, Kaijie Zhu, Zhongkai Yu, Kun Zhou, and Jingbo Shang. SimSD: Simple Speculative Decoding in Diffusion Language Models, June 2026. URL http://arxiv.org/abs/2606.02544. arXiv:2606.02544 [cs.CL].

Majid Daliri, Christopher Musco, and Ananda Theertha Suresh. Coupling Without Communication and Drafter-Invariant Speculative Decoding. In *2025 IEEE International Symposium on Information Theory (ISIT)*, pp. 1–6, June 2025. doi: 10.1109/ISIT63088.2025.11195663. URL https://ieeexplore.ieee.org/abstract/document/11195663. ISSN: 2157-8117.

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 Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.183. URL https://aclanthology.org/2023.emnlp-main.183/.

Yifeng Gao, Ziang Ji, Yuxuan Wang, Biqing Qi, Hanlin Xu, and Linfeng Zhang. Self Speculative Decoding for Diffusion Large Language Models, October 2025. URL http://arxiv.org/abs/2510.04147. arXiv:2510.04147 [cs.CL].

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset, March 2021. URL https://arxiv.org/abs/2103.03874v2.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast Inference from Transformers via Speculative Decoding. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 19274–19286. PMLR, July 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Guanghao Li, Zhihui Fu, Min Fang, Qibin Zhao, Ming Tang, Chun Yuan, and Jun Wang. DiffuSpec: Unlocking Diffusion Language Models for Speculative Decoding. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 20896–20910, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1048. URL https://aclanthology.org/2026.findings-acl.1048/.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster Inference of Language Models with Dynamic Draft Trees. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. 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: Speculative Sampling Requires Rethinking Feature Uncertainty. In *Proceedings of the 41st International Conference on*

Machine Learning, pp. 28935–28948. PMLR, July 2024b. URL https://proceedings.mlr.press/v235/li24bt.html.

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, pp. 136737–136756. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c7b5a35ea98b62512a869c19ea7b03cb-Abstract-Conference.html.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan LI. Large Language Diffusion Models. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 50608–50646. Curran Associates, Inc., 2025. doi: 10.52202/085713-1689. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/48b383b24230e0e6e649d9c98dae4d8c-Abstract-Conference.html.

Tianxiang Pan, Baitao Gong, Mo Guang, Hongwei Yong, Tianpeng Jiang, Yaqian Li, Zheng Cao, and Kaiwen Long. BlockSpec: Blockwise Speculative Decoding for Diffusion LLMs. October 2025. URL https://openreview.net/forum?id=hmAviop5rm.

Eliron Rahimi, Elad Hirshel, Rom Himelstein, Amit LeVi, Avi Mendelson, and Chaim Baskin. Step-Wise Refusal Dynamics in Autoregressive and Diffusion Language Models, February 2026. URL https://arxiv.org/abs/2602.02600v4.

Liran Ringel and Yaniv Romano. Accelerating Speculative Decoding with Block Diffusion Draft Trees, April 2026. URL http://arxiv.org/abs/2604.12989. arXiv:2604.12989 [cs.CL].

Shengyin Sun, Yiming Li, Renxi Liu, Xinqi Li, Hui-Ling Zhen, Weizhe Lin, Chen Chen, Xianzhi Yu, Mingxuan Yuan, and Chen Ma. PSD: Pushing the Pareto Frontier of Diffusion LLMs via Parallel Speculative Decoding, May 2026. URL http://arxiv.org/abs/2605.15609. arXiv:2605.15609 [cs.CL].

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free Acceleration of Diffusion LLM by Enabling KV Cache and Parallel Decoding. *International Conference on Learning Representations*, 2026:57027–57051, April 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5d8d4e6061c3ba96c240b7fa1ae3471d-Abstract-Conference.html.

Yufeng Xu, Zishuo Bao, Qian Wang, Zeshen Zhang, Haoqi Zhang, Bowen Peng, Ang Li, Rahul Chalamala, and Yucheng Lu. Multi-Token Residual Prediction, June 2026. URL http://arxiv.org/abs/2605.18817. arXiv:2605.18817 [cs.LG].

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion Large Language Models, August 2025. URL http://arxiv.org/abs/2508.15487. arXiv:2508.15487 [cs.CL].

A Formal Setup and Exactness Proofs

This section makes the decoding semantics used in Section 4 explicit and proves Propositions 1 and 2.

A.1 Formal decoding model

States and terminal states. Recall

$$\mathcal{X} = (\mathcal{V} \cup \{\text{M}\})^L, \qquad \mathcal{M}(x) = \{s : x_s = \text{M}\}.$$

Define the nonterminal state space

$$\mathcal{X}_{\text{nt}} = \{x \in \mathcal{X} : \mathcal{M}(x) \neq \varnothing\}.$$

All target transition maps below are defined on $\mathcal{X}_{\text{nt}}$.

The completed causal context C is fixed while an active block is decoded. A target forward at state x returns

$$p_{\theta,s}(\cdot \mid x, C)$$

for every $s \in \mathcal{M}(x)$.

Idealized target evaluation. The exactness results concern the mathematical target function: evaluating $p_\theta(\cdot \mid x, C)$ as one replica of a batch is assumed to return the same mathematical conditional as evaluating the same (x, C) alone. This separates algorithmic exactness from finite-precision effects caused by physical batch shape; the latter are quantified in Appendix G.

Position selection. The target chooses

$$s^\star(x) = \arg\max_{s \in \mathcal{M}(x)} \max_{v \in \mathcal{V}} p_{\theta,s}(v \mid x, C).$$

All argmax operations use a fixed deterministic tie-breaking rule. In particular, $s^\star(x)$ depends only on the current target state and not on sampling randomness that has not yet been consumed.

For $s \in \mathcal{M}(x)$ and $v \in \mathcal{V}$, write

$$x \oplus (s, v)$$

for the state obtained by replacing $x_s = \text{M}$ by v.

Greedy transition. At $T = 0$,

$$v_0^\star(x) = \arg\max_v p_{\theta,s^\star(x)}(v \mid x, C),$$

and therefore

$$\Phi_0 : \mathcal{X}_{\text{nt}} \to \mathcal{X}, \qquad \Phi_0(x) = x \oplus \big(s^\star(x), v_0^\star(x)\big).$$

Starting from initial block state x_0, draft-free greedy decoding is the deterministic orbit

$$x_0^{\text{DF}} = x_0, \qquad x_{j+1}^{\text{DF}} = \Phi_0(x_j^{\text{DF}})$$

until a terminal state is reached.

Sampling transition. At $T > 0$, the target token law at position s is

$$\pi_{T,s}(v \mid x, C) = \frac{p_{\theta,s}(v \mid x, C)^{1/T}}{\sum_{u \in \mathcal{V}} p_{\theta,s}(u \mid x, C)^{1/T}}.$$

Thus standard draft-free target sampling at state x first chooses the deterministic position $s^\star(x)$ and then draws

$$V \sim \pi_{T,s^\star(x)}(\cdot \mid x, C).$$

Shared Gumbel representation. Let $a(s)$ denote the absolute output position corresponding to local block position s. Draw

$$G_{a,v} \overset{\text{iid}}{\sim} \text{Gumbel}(0,1)$$

for every absolute position a and token v, and let

$$G = \{G_{a,v}\}_{a,v}.$$

For fixed G, define

$$v_G^\star(x) = \arg\max_v \left[\frac{1}{T}\log p_{\theta,s^\star(x)}(v \mid x, C) + G_{a(s^\star(x)),v}\right]$$

and

$$\Phi_G : \mathcal{X}_{\text{nt}} \to \mathcal{X}, \qquad \Phi_G(x) = x \oplus \big(s^\star(x), v_G^\star(x)\big).$$

For any fixed realization $G = g$, Φ_g is deterministic.

Draft trees. At speculative round t, a drafter $\mathcal{D}$ returns a finite rooted tree

$$\mathcal{T}_t = (\mathcal{N}_t, \mathcal{E}_t)$$

whose root is the current committed state x_t. Every edge $(y, z) \in \mathcal{E}_t$ corresponds to exactly one hypothetical commitment:

$$z = y \oplus (s, v)$$

for some $s \in \mathcal{M}(y)$ and $v \in \mathcal{V}$.

The exactness arguments make no assumption about how this tree is constructed. In particular, $\mathcal{D}$ may be learned or hand-designed, may return a chain or a tree, may depend on all previous target outputs, and at $T > 0$ may depend on the shared randomness G.

State-equality verifier. Let Φ denote Φ_0 for $T = 0$ or Φ_G after fixing the shared randomness for $T > 0$. Beginning at the root $y_0 = x_t$, the verifier proceeds recursively.

If y_j has a drafted child z such that

$$z = \Phi(y_j),$$

the verifier accepts that child and continues from it. If no such child exists, it rejects all remaining descendants, commits $\Phi(y_j)$ directly, and terminates the speculative round. If y_j is terminal, no further transition is made.

Since $\Phi(y_j)$ is a unique one-step extension, at most one child of a node can be accepted. Hence the accepted portion of an arbitrary draft tree is a single path.

Let X_T^{DF} denote the complete draft-free trajectory and $X_{T,\mathcal{D}}^{\text{Spec}}$ the complete committed speculative trajectory. A verifier is distribution preserving when

$$X_{T,\mathcal{D}}^{\text{Spec}} \overset{d}{=} X_T^{\text{DF}}$$

for every admissible drafter $\mathcal{D}$.

A.2 Proof of Proposition 1

Proof. Fix target p_θ, causal context C, initial block state x_0, and an arbitrary drafter $\mathcal{D}$. By the fixed deterministic tie-breaking rule, at $T = 0$ the target transition

$$\Phi_0 : \mathcal{X}_{\text{nt}} \to \mathcal{X}$$

is deterministic and gives a unique successor for every nonterminal state.

Let

$$x_0^{\text{DF}} = x_0, \qquad x_{k+1}^{\text{DF}} = \Phi_0(x_k^{\text{DF}})$$

be the draft-free trajectory, and let x_k^{Spec} denote the state of the speculative decoder after k committed transitions.

We prove by induction on k that

$$x_k^{\text{Spec}} = x_k^{\text{DF}}$$

for every commitment before termination.

Base case. For $k = 0$, both procedures start from the same initial state:

$$x_0^{\mathrm{Spec}} = x_0 = x_0^{\mathrm{DF}}.$$

Inductive step. Assume

$$x_k^{\mathrm{Spec}} = x_k^{\mathrm{DF}}$$

for some nonterminal state. From x_k^{Spec}, the target has the unique successor

$$\Phi_0(x_k^{\mathrm{Spec}}).$$

If the draft tree contains this state as a child of x_k^{Spec}, state-equality verification accepts that child. Otherwise, the verifier rejects the speculative continuation and commits the same state as the target fallback. Hence, in either case,

$$\begin{aligned} x_{k+1}^{\mathrm{Spec}} &= \Phi_0(x_k^{\mathrm{Spec}}) \\ &= \Phi_0(x_k^{\mathrm{DF}}) \\ &= x_{k+1}^{\mathrm{DF}}. \end{aligned}$$

Thus the induction hypothesis holds for $k + 1$.

Since $\Phi_0(x)$ is unique, at most one child of any tree node can be accepted. Therefore, even for an arbitrary finite draft tree, the accepted states form a single path, and the argument above applies to every committed transition.

By induction,

$$X_{0,\mathcal{D}}^{\mathrm{Spec}} = X_0^{\mathrm{DF}}.$$

□

The argument is block-local only for notation. Since the completed block is identical to draft-free decoding, the causal context of the next block is also identical. Induction over blocks therefore gives equality of the complete generated sequence at $T = 0$.

A.3 Proof of Proposition 2

Proof. Fix $T > 0$, target p_θ, causal context C, initial state x_0, and an arbitrary drafter $\mathcal{D}$.

The proof has three steps. First, we show that the shared-Gumbel construction induces the same draft-free trajectory distribution as ordinary temperature-T sampling. Second, conditioned on a realization of the Gumbel table, verification reduces to the deterministic case of Proposition 1. Third, we marginalize over the shared randomness.

Step 1: Gumbel-driven draft-free law. Fix a nonterminal state x and let

$$s = s^\star(x).$$

Since $\{G_{a(s),v}\}_{v\in\mathcal{V}}$ are independent standard Gumbels, the Gumbel–argmax identity gives

$$\Pr[v_G^\star(x) = u] = \frac{\exp\big(T^{-1}\log p_{\theta,s}(u \mid x, C)\big)}{\sum_{v\in\mathcal{V}} \exp(T^{-1}\log p_{\theta,s}(v \mid x, C))} \tag{5}$$

$$= \frac{p_{\theta,s}(u \mid x, C)^{1/T}}{\sum_{v\in\mathcal{V}} p_{\theta,s}(v \mid x, C)^{1/T}} \tag{6}$$

$$= \pi_{T,s}(u \mid x, C). \tag{7}$$

Thus, for any fixed state, the Gumbel construction induces exactly the temperature-T target token law.

It remains to show that using a single Gumbel table indexed by absolute position preserves this law along the full adaptive trajectory. Consider commitment step j of Gumbel-driven draft-free decoding, with current state x_j and selected position

$$s_j = s^\star(x_j).$$

Let $\mathcal{F}_j$ denote the history generated by the states and Gumbel rows used during the first j commitments. Since each commitment fills a previously masked position, s_j has not been committed before, and therefore the row

$$G_{a(s_j),\cdot}$$

has not previously been used.

Moreover, s_j is determined by x_j and hence by $\mathcal{F}_j$; its selection does not depend on any unused Gumbel row. Since the Gumbel rows are independent across absolute positions, $G_{a(s_j),\cdot}$ remains an independent standard-Gumbel vector conditional on $\mathcal{F}_j$. Therefore,

$$\Pr[v_G^\star(x_j) = u \mid \mathcal{F}_j] = \pi_{T,s_j}(u \mid x_j, C).$$

Hence the Gumbel-driven decoder has the same conditional transition law as ordinary draft-free sampling at every commitment. Since both start from x_0, the chain rule gives

$$X_T^{\mathrm{DF}}(G) \overset{d}{=} X_T^{\mathrm{DF}}. \tag{8}$$

Step 2: conditional pathwise exactness. Fix any realization $G = g$. Then

$$\Phi_g : \mathcal{X}_{\mathrm{nt}} \to \mathcal{X}$$

is a deterministic transition map.

The proof of Proposition 1 requires only that the target have a unique deterministic successor at each state and that a drafted child be accepted exactly when it equals that successor. Both properties hold with Φ_g in place of Φ_0. Therefore, for every realization g,

$$X_{T,\mathcal{D}}^{\mathrm{Spec}}(g) = X_T^{\mathrm{DF}}(g). \tag{9}$$

This argument is independent of how $\mathcal{D}$ constructs its draft. In particular, the draft may depend on stale target distributions, learned parameters, previous decoding history, or the shared table G itself; a drafted edge is committed only when it equals the target successor under the same realization g.

Step 3: marginalization. By Step 2, for every realization g of the shared Gumbel table,

$$X_{T,\mathcal{D}}^{\mathrm{Spec}}(g) = X_T^{\mathrm{DF}}(g).$$

Therefore, when G is drawn according to its Gumbel distribution,

$$X_{T,\mathcal{D}}^{\mathrm{Spec}}(G) = X_T^{\mathrm{DF}}(G).$$

By definition, $X_{T,\mathcal{D}}^{\mathrm{Spec}}$ denotes this same random speculative trajectory with its dependence on G left implicit.

Hence, for any set A of complete decoding trajectories,

$$\Pr\left[X_{T,\mathcal{D}}^{\mathrm{Spec}} \in A\right] = \Pr\left[X_T^{\mathrm{DF}}(G) \in A\right].$$

Using equation 8,

$$\Pr\left[X_{T,\mathcal{D}}^{\mathrm{Spec}} \in A\right] = \Pr\left[X_T^{\mathrm{DF}} \in A\right].$$

Since this holds for every A,

$$X_{T,\mathcal{D}}^{\mathrm{Spec}} \overset{d}{=} X_T^{\mathrm{DF}}.$$

Finally, for fixed G, the committed trajectory is the deterministic orbit of Φ_G and is therefore invariant to the choice of drafter. This establishes drafter-invariance.

□

B Rejection Sampling as an Alternative Verifier

Coupled-noise match is the verifier used by SPOT because it requires only state equality and applies directly to DDTree. If a drafter instead exposes an explicit *sequential proposal distribution*, exact verification can alternatively be performed by rejection sampling (RS). We formalize this alternative here, prove its correctness and maximal per-link acceptance probability, and describe the implementation-level validation used in our experiments.

B.1 SEQUENTIAL PROPOSAL SETUP

Consider a verified predecessor state

$$y_k \in \mathcal{X}_{\mathrm{nt}}.$$

Let

$$\mathcal{Z}(y_k) = \mathcal{M}(y_k) \times \mathcal{V}$$

be the set of possible one-step position–token commitments.

Target commit law. Let

$$s_k^\star = s^\star(y_k).$$

At $T > 0$, the target distribution over one-step commitments is

$$\pi_k(s, v) = \mathbf{1}\{s = s_k^\star\}\, \pi_{T,s_k^\star}(v \mid y_k, C). \tag{10}$$

Therefore

$$\sum_{z \in \mathcal{Z}(y_k)} \pi_k(z) = 1.$$

Proposal law. Suppose the drafter exposes a normalized proposal distribution

$$q_k \in \Delta(\mathcal{Z}(y_k))$$

and generates one proposed commitment

$$\widetilde{Z}_k = (\widetilde{S}_k, \widetilde{V}_k) \sim q_k. \tag{11}$$

For the chain proposal used in our experiments,

$$q_k(s, v) = q_k^{\mathrm{pos}}(s)\, \widehat{\pi}_{T,k,s}(v), \tag{12}$$

where q_k^{pos} is the drafter's next-position distribution and $\widehat{\pi}_{T,k,s}$ is its stale temperature-adjusted token conditional.

The factorization in equation 12 is not required by the proof; exactness needs only that the proposal actually be sampled from the explicit distribution q_k supplied to the verifier.

Accept–residual rule. Given proposal $\widetilde{Z}_k = z$ with $q_k(z) > 0$, accept it with probability

$$\alpha_k(z) = \min\left\{1, \frac{\pi_k(z)}{q_k(z)}\right\}. \tag{13}$$

Upon rejection, draw a replacement from

$$r_k(z) = \frac{(\pi_k(z) - q_k(z))_+}{\sum_{w \in \mathcal{Z}(y_k)} (\pi_k(w) - q_k(w))_+}. \tag{14}$$

If the denominator is zero, then $\pi_k = q_k$, rejection has probability zero, and r_k is never queried.

B.2 EXACTNESS AND OPTIMAL ACCEPTANCE

Proposition 3 (Rejection-sampling exactness and optimal acceptance)**.** *Fix a verified predecessor state y_k. Let $\widetilde{Z}_k \sim q_k$ be accepted according to equation 13, with rejected proposals replaced using equation 14. Then the committed pair Z_k satisfies*

$$Z_k \sim \pi_k.$$

Moreover,

$$\Pr[\mathrm{accept} \mid y_k] = 1 - D_{\mathrm{TV}}(\pi_k, q_k),$$

which is the maximal possible agreement probability among all couplings with marginals π_k and q_k. Applying the rule sequentially at every verified predecessor therefore reproduces the complete draft-free target distribution.

Proof. Fix y_k and abbreviate

$$\pi = \pi_k, \qquad q = q_k, \qquad r = r_k.$$

Step 1: accepted mass. For arbitrary $z \in \mathcal{Z}(y_k)$ with $q(z) > 0$,

$$\Pr[\widetilde{Z}_k = z, \text{accept}] = q(z) \min\left\{1, \frac{\pi(z)}{q(z)}\right\} \tag{15}$$

$$= \min\{q(z), \pi(z)\}. \tag{16}$$

If $q(z) = 0$, both sides of equation 16 are zero, so the identity holds for all z.

Summing over z,

$$\Pr[\text{accept}] = \sum_z \min\{q(z), \pi(z)\}. \tag{17}$$

Hence

$$\Pr[\text{reject}] = 1 - \sum_z \min\{q(z), \pi(z)\}. \tag{18}$$

Step 2: rejected mass. Because π is normalized,

$$1 - \sum_z \min\{q(z), \pi(z)\} = \sum_z [\pi(z) - \min\{q(z), \pi(z)\}] \tag{19}$$

$$= \sum_z (\pi(z) - q(z))_+. \tag{20}$$

Therefore

$$\Pr[\text{reject}] = \sum_z (\pi(z) - q(z))_+.$$

If this quantity is positive, then

$$r(z) = \frac{(\pi(z) - q(z))_+}{\Pr[\text{reject}]},$$

so

$$\Pr[\text{reject}, Z_k = z] = (\pi(z) - q(z))_+. \tag{21}$$

Step 3: one-step exactness. Combining the accepted and rejected branches,

$$\Pr[Z_k = z] = \min\{q(z), \pi(z)\} + (\pi(z) - q(z))_+ \tag{22}$$

$$= \pi(z). \tag{23}$$

Thus

$$Z_k \sim \pi_k.$$

If the denominator of equation 14 is zero, then $(\pi(z) - q(z))_+ = 0$ for every z. Hence $\pi(z) \le q(z)$ everywhere, and because both distributions have total mass one, $\pi = q$. In that case equation 17 gives $\Pr[\text{accept}] = 1$, so the residual branch is never used.

Step 4: maximal acceptance. For discrete distributions,

$$D_{\mathrm{TV}}(\pi, q) = 1 - \sum_z \min\{\pi(z), q(z)\}.$$

Combining this identity with equation 17,

$$\Pr[\text{accept}] = 1 - D_{\mathrm{TV}}(\pi_k, q_k).$$

The quantity

$$\sum_z \min\{\pi_k(z), q_k(z)\}$$

is the maximal possible agreement probability of two random variables with marginals π_k and q_k. The accept–residual construction therefore realizes a maximal coupling.

Step 5: trajectory exactness. Let $Y_0, Y_1, \ldots$ denote the states produced by sequential RS. Conditioned on any realized history $Y_{0:k} = y_{0:k}$, one-step exactness gives

$$Z_k \mid (Y_{0:k} = y_{0:k}) \sim \pi_k(\cdot \mid y_k),$$

which is exactly the target's one-step commit law at y_k. Hence, for every successor y',

$$\Pr[Y_{k+1} = y' \mid Y_{0:k} = y_{0:k}] = \Pr_{\mathrm{DF}}[X_{k+1} = y' \mid X_k = y_k].$$

Since RS and draft-free decoding start from the same initial state, the chain rule implies that every finite trajectory $y_0, \ldots, y_m$ has the same probability under both procedures:

$$\Pr[Y_0 = y_0, \ldots, Y_m = y_m] = \Pr_{\mathrm{DF}}[X_0 = y_0, \ldots, X_m = y_m].$$

Therefore, the complete RS trajectory has the draft-free target distribution. □

B.3 Relation to batched tree verification

Proposition 3 assumes the classical per-link proposal interface: at each verified state y_k, a single proposal is sampled from $q_k(\cdot \mid y_k)$ and corrected by the accept–residual rule. Only after this step determines the realized successor is the proposal at the next verified state generated.

This interface does not directly provide a verification rule for a preconstructed draft tree containing multiple candidate children of the same predecessor. Applying the per-link RS rule unchanged selects only one sampled proposal at each visited state; moreover, after a rejection, the residual sample need not belong to the preconstructed tree. Thus, although target conditionals at multiple hypothetical states could still be evaluated in parallel, classical per-link RS does not directly exploit the full batched tree structure used by our drafter.

Coupled-noise verification avoids this limitation. Once the shared randomness G is fixed, the target defines a unique successor $\Phi_G(y)$ for every hypothetical state y. Each drafted edge (y, z) can therefore be verified independently by the state-equality test

$$z = \Phi_G(y),$$

allowing all hypothetical states in the tree to be evaluated in the same target batch and the unique target-consistent path to be recovered afterward.

More general multi-proposal or tree-aware rejection-sampling constructions may also be possible, but are outside the scope of this work.

B.4 Empirical validation

Proposition 3 establishes the correctness of the idealized accept–residual procedure. We additionally verify that our implementation matches this behavior on live target states.

For each test state, we record the target and proposal distributions π_k and q_k, repeatedly execute the RS verifier, and compare the empirical distribution of committed position–token pairs with π_k using the χ^2 goodness-of-fit test employed in our benchmark. We include deliberately miscalibrated proposal distributions so that the validation also covers regimes in which q_k differs substantially from the target.

We separately verify the acceptance-rate identity from Proposition 3. For each tested state, the theoretical RS acceptance probability is

$$\Pr[\mathrm{accept}] = 1 - D_{\mathrm{TV}}(\pi_k, q_k).$$

The acceptance rate measured from repeated executions of our implementation differs from this theoretical value by at most 0.002 across the tested states.

These checks are run before the RS benchmark experiments and serve only to validate the implementation; the distributional guarantee itself follows from Proposition 3.

Algorithm 1 Training the prefix-conditional commit-order predictor from draft-free target trajectories.

Require: Target p_θ, prompt set $\mathcal{S}$, maximum replay depth $d_{\max}$
Ensure: Trained order model $q_{\phi^\star}$

$\mathcal{R}_\theta \leftarrow \textsc{Decode}(p_\theta, \mathcal{S})$ ▷ Collect draft-free target trajectories
$\mathcal{E} \leftarrow \varnothing$
for each recorded target state $(x_t, \widehat{p}_t, \mathbf{s}_t^\star) \in \mathcal{R}_\theta$ **do**
  for $d = 0, \ldots, d_{\max}$ **do**
    $y_{t,d} \leftarrow \textsc{Prefix}(x_t, \mathbf{s}_t^\star, d)$ ▷ Replay the first d target commitments
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\widehat{p}_t, y_{t,d}, s_{t+d}^\star)\}$ ▷ Next position is the supervision target
  end for
end for
$\phi^\star \leftarrow \arg\min_\phi \mathbb{E}_{(\widehat{p},y,s^\star)\sim\mathcal{E}} [-\log q_\phi(s^\star \mid y)]$
return $q_{\phi^\star}$

C Drafter Training and Draft-Tree Construction

This appendix gives the complete procedures used to train the prefix-conditional commit-order predictor and to construct the speculative draft tree at inference time. The notation follows Section 4.2. In particular, $\widehat{p}_t$ denotes the stale target conditionals available at speculative round t, and $q_\phi(s \mid y)$ denotes the predicted distribution over the target's next committed position conditioned on hypothetical prefix state y.

C.1 Training the commit-order predictor

Training uses only draft-free trajectories generated by the target. For each target state x_t, we retain the stale conditionals $\widehat{p}_t$ available at that state together with the subsequent target commit sequence

$$\mathbf{s}_t^\star = (s_t^\star, s_{t+1}^\star, \ldots).$$

For every prefix depth d, we replay the first d target commitments to construct the hypothetical state $y_{t,d}$ while keeping $\widehat{p}_t$ fixed. The next target commitment $s_{t+d}^\star$ is then used as the supervision target. Hence each training example has the form

$$(\widehat{p}_t, y_{t,d}, s_{t+d}^\star).$$

The important feature of this construction is that the stale conditionals $\widehat{p}_t$ remain unchanged while the hypothetical prefix is extended. The resulting examples therefore match the information available to the drafter during recursive inference rather than giving it access to future target evaluations.

C.2 Prefix-conditional draft-tree construction

At speculative round t, drafting begins from the current state x_t and the stale target conditionals $\widehat{p}_t$. For each active hypothetical state y, the order model produces

$$q_\phi(s \mid y), \qquad s \in \mathcal{M}(y).$$

Each candidate position is paired with token hypotheses from $\mathcal{H}_t(s)$. At $T = 0$, the primary hypothesis is

$$\widetilde{v}_0(s) = \arg\max_v \widehat{p}_{t,s}(v),$$

with the stale runner-up additionally retained when the top-1 probability is below 0.6. At $T > 0$, $\mathcal{H}_t(s)$ is determined by the coupled stale draw in Eq. 4.

For every extension

$$z = y \oplus (s, v),$$

we use the same prefix mass defined in the main text,

$$m(z) = m(y)\, q_\phi(s \mid y)\, \widehat{p}_{t,s}(v), \qquad m(x_t) = 1.$$

Algorithm 2 Prefix-conditional construction of the speculative draft tree under budget D.

Require: Current target state x_t, stale conditionals $\widehat{p}_t$, order model q_ϕ, draft budget D
Ensure: Draft tree $\mathcal{T}_t$

$\mathcal{T}_t \leftarrow \{x_t\}$
$m(x_t) \leftarrow 1$
$\mathcal{F} \leftarrow \{x_t\}$ ▷ Current expansion frontier
while $|\mathcal{T}_t| - 1 < D$ **do**
  $\mathcal{C} \leftarrow \varnothing$
  for each active prefix $y \in \mathcal{F}$ **do**
    $q_y(\cdot) \leftarrow q_\phi(\cdot \mid y)$ ▷ Predict next commit position
    for each $s \in \mathcal{M}(y)$ **do**
      construct $\mathcal{H}_t(s)$ from $\widehat{p}_{t,s}$ ▷ Token hypotheses
      for each $v \in \mathcal{H}_t(s)$ **do**
        $z \leftarrow y \oplus (s, v)$
        $m(z) \leftarrow m(y)\, q_y(s)\, \widehat{p}_{t,s}(v)$
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{z\}$
      end for
    end for
  end for
  $(\mathcal{T}_t, \mathcal{F}) \leftarrow \text{DDTREETOPD}(\mathcal{T}_t, \mathcal{C}, D)$ ▷ Retain highest-mass states within the remaining budget
end while
return $\mathcal{T}_t$

DDTree then allocates the available draft budget to the highest-mass states.

Because every multiplicative factor in the prefix mass lies in $[0, 1]$, a child's mass cannot exceed that of its parent. This monotonicity permits level-wise expansion while recovering the same top-D fixpoint as sequential best-first search. The calibration temperature τ is applied to q_ϕ before the search, with $\tau = 1$ in the final model.

D Experimental setup

D.1 Drafter training details

Architecture: three-layer transformer encoder over the masked positions, d_{model}=128, 8 heads, top-32 stale probabilities per position (m_{top}=32), 4 token candidates per position for the hedge; 0.62M parameters. Inputs per masked position: stale top-32 probabilities (raw and log), eight summary statistics, two path features (taken-position set, depth). Training data: decoding trajectories of the target itself at T=0, with prefix-conditional examples extracted at every (round, depth). The prompt pools are recorded per shard and differ by target. The SDAR-8B b32 and LLaDA2.0-mini b32 drafters are trained on CodeAlpaca (Chaudhary, 2023) alone, ≈21k and ≈14k trajectories; the SDAR-4B drafter on a four-pool mixture of CodeAlpaca, UltraChat (Ding et al., 2023) and the chat and code shards of the Llama-Nemotron post-training corpus (Bercovich et al., 2025), 15,027 trajectories. No pool contains MATH-500 or any evaluation sample. Optimization: Adam, learning rate 3×10^{-4}, 10 epochs; the released SDAR-8B b32 checkpoint reaches 0.75 held-out next-commit accuracy (0.77 at depth 0, 0.72 at depth 7). Training a drafter takes under two GPU-hours on one card. The identical procedure (data regenerated per target and width) produces the checkpoints for SDAR-8B b32/b16, SDAR-4B and LLaDA2.0-mini b32/b16; no per-setting tuning.

What the drafters have seen. Two consequences follow, disclosed rather than controlled. The SDAR-8B and mini drafters have seen only code, while two of the four evaluation tasks are mathematics: their GSM8K and MATH500 runs are out of domain for the drafter, which makes those results evidence that the learned order carries across domains rather than memorizing one. And because the SDAR-4B drafter was trained on a different mixture, any difference between the SDAR-8B and SDAR-4B tables confounds target scale with corpus composition; we report both and attribute

neither. Regenerating the SDAR-8B corpus would invalidate every downstream run of the benchmark, so the confound is left in place and stated here.

D.2 Draft layout and verification cost

A target forward at a block state yields a distribution at every masked position; the most confident position is filled and the block is re-run. To verify D drafts in one forward, each draft is appended behind the cached prefix as a separate copy of the block with its own mask layout, and a copy attends to the prefix and to itself only, so its logits are exactly what the target would compute at that state. One forward over $(1{+}D)L$ positions therefore returns the target's own next commit at the root and at every draft, a rejected draft costs positions but never changes the output, and packing a tree into a single sequence is not exact here, since a copy's logits depend on its complete mask layout.

D.3 Decoding configuration

Strict single-commit decoding, generation length 256, block width 32 (16 where stated), exact prefix KV cache on (for LLaDA2.0-mini: an exact prefix-KV path we added to the released model code, verified to reproduce the uncached decoder's commits), draft layout "concat" (draft replicas appended behind a blockwise mask), padding of drafted forwards to $(1{+}D)L$ positions unless stated, distribution-preserving verdict derived from the configuration and recorded per run. Prompt templates and stopping rules per task are those of the released harness (Appendix E).

D.4 Baseline configuration

Spiffy's draft graph is calibrated with the released procedure of Agrawal et al. (2026) (offline collection on held-out prompts: MBPP train and MATH500 problems disjoint from the evaluation samples, 25 each, lookahead 6, graph size 24) separately for SDAR-8B and SDAR-4B, and truncated to the budget at decode time. The SSD-style free chain uses candidate rank 1, the sampler's ordering and depth 4. The BlockSpec-style confidence tree is a best-first tree over the target's renormalized stale confidences. All baselines run through the same decoder and verifier as SPOT; their draft-free control is shared with ours per task.

D.5 Mass-floor calibration

The floor γ of SPOT-γ is calibrated per target on 50 held-out samples, MBPP test samples 100–149 (every evaluated run uses samples 0–99): $\gamma \in \{0, 0.02, 0.05, 0.1, 0.2\}$, unpadded, $D \in \{8, 12, 20\}$, $T{=}0$, seed 0, the five runs back to back on one card so that the grid shares machine state, with the $\gamma{=}0$ run also providing the draft-free control every ratio is taken against. The selected value is the argmax of the mean throughput ratio over the three budgets, the smaller floor being kept when two are within 3% (Table 4). On SDAR-8B the choice is clear (0.1 leads 0.05 by 10%); on SDAR-4B 0.05 leads 0.02 by 2.5%, inside the tie band, so the smaller floor is kept; on LLaDA2.0-mini no floor beats $\gamma{=}0$, whose throughput does not fall with the budget on that target, and the floor is switched off. On the evaluated samples the calibrated floor is at or above the padded $\gamma{=}0$ recipe in 47 of the 48 SDAR settings (four tasks, three budgets, two temperatures, two models); the exception is SDAR-4B MATH500 $D12$ at $T{=}0$, 4% below at the same forwards. No single value achieves this on both models: $\gamma{=}0.05$, the best fixed choice, is at or above the padded recipe in all 32 settings at $D12$ and $D20$ but in only 10 of 16 at $D8$, where the fixed budget is already near its throughput optimum and the floor has little to recover; four of its six misses are within 3% and two, SDAR-4B MATH500 at $D8$, are 8% and 13%. One test run (SDAR-4B, GSM8K, $T{=}1$) was repeated after its first pass showed a 15% wall drop at $D8$ alone with identical forwards; the second pass is reported and the first archived in place.

D.6 Methods outside the comparison

SimSD. SimSD (Cui et al., 2026) drafts with a same-family sibling model and verifies the drafted tokens in one forward under a plug-in attention mask that lends them temporally valid contexts. That test approximates the sequential denoising distribution instead of reproducing it, and its reported speedups come with changed (on average improved) generation quality. It is therefore a

Table 4: Mass-floor calibration grids: mean throughput ratio over $D8/D12/D20$ on the 50 held-out MBPP samples, unpadded, against the same card's draft-free run. Bold: selected.

target	γ=0	0.02	0.05	0.1	0.2
SDAR-8B b32	2.58	3.01	3.21	**3.53**	3.38
SDAR-4B b32	2.99	**3.54**	3.63	3.62	3.46
LLaDA2.0-mini b32	**2.10**	2.09	2.04	1.94	1.94

lossy accelerator, not a lossless speculative decoder, and its throughput is not commensurable with that of the exact methods compared here; we do not compare against it.

MRP, from reported numbers. MRP (Xu et al., 2026) trains a three-layer residual head to extrapolate the backbone's logits up to K=3 denoising steps ahead and keeps a drafted token where the backbone's argmax agrees with it. It reports results on the same SDAR-Chat 1.7B, 4B and 8B targets and the same four tasks as this paper (its Table 2: block width 16, greedy decoding, one H100), against a one-token-per-step baseline, which is the draft-free decoder of Section 3. We did not run its code; Table 5 sets its reported speedups beside SPOT's $D8$ throughput ratio over our draft-free control (block width 32, length 256, one RTX 6000 Ada). The two are measured in different implementations, so the comparison is between relative gains: SPOT is ahead in all twelve settings, by 1.5–3.1×, and uses 3.2–4.9× fewer target forwards than draft-free decoding, a quantity MRP does not report. Its acceptance test is argmax equality, which has no distribution-preserving extension to sampling.

Table 5: MRP's reported speedups next to SPOT. MRP: speculative mode (K=3) over its one-token-per-step baseline, Table 2 of Xu et al. (2026) (one H100, block width 16, greedy). SPOT: $D8$ throughput ratio over our draft-free control (block width 32, length 256, one RTX 6000 Ada), NFE reduction in parentheses. MRP generates up to 4096 tokens. Different implementations: relative gains only.

target	method	GSM8K	MATH500	HumanEval	MBPP
SDAR-1.7B	MRP (reported)	1.25×	1.23×	1.23×	1.17×
	SPOT (ours)	**2.79**× (3.6)	**2.49**× (3.2)	**3.15**× (4.2)	**3.61**× (4.9)
SDAR-4B	MRP (reported)	1.36×	1.26×	1.35×	1.27×
	SPOT (ours)	**3.17**× (4.0)	**2.63**× (3.3)	**3.48**× (4.4)	**3.48**× (4.9)
SDAR-8B	MRP (reported)	1.40×	1.39×	1.34×	1.34×
	SPOT (ours)	**2.50**× (3.9)	**2.03**× (3.2)	**2.74**× (4.4)	**2.61**× (4.7)

D.7 Full-attention targets (LLaDA-8B, Dream-7B)

LLaDA-8B-Instruct and Dream-7B-Instruct attend over the whole sequence, so the keys and values of the prefix depend on the block being decoded and no exact prefix cache exists for them. For the transfer rows of Table 3 both run under an approximate prefix cache of the Fast-dLLM kind (Wu et al., 2026): the prefix is encoded once when a block opens and reused, unrefreshed, for every forward of that block; the draft-free control of each task runs under the same cache, and every such run is recorded as not distribution-preserving. The draft replicas use the same concat layout as on SDAR, rotated at the positions of the block they draft for; for LLaDA-8B this required threading explicit position ids through the released model code (Dream's already accepts them), and the patched model reproduces the released one exactly when positions are left implicit. Two checks were run in fp32 before the benchmark: an appended, attention-isolated replica changes the decoded block's logits by at most 10^{-4} (LLaDA) and 4×10^{-4} (Dream), and a replica of the true block reproduces that block's own logits to 10^{-4} and 2×10^{-4}, so verification on these targets is exact relative to the cached decoder; the approximation lives entirely in the cache, which is why these rows are reported against the cached control and kept out of every lossless claim.

D.8 ACCOUNTING DETAILS

NFE counts target forwards only; oracle lookahead and drafter forwards are excluded from it, and the drafter's own cost is charged to wall clock and therefore to the throughput ratio. Throughput is measured per sample as generated tokens over net wall time of the generation loop, summed over the 100 samples, and reported as the ratio to the draft-free run of the same task, model and temperature. That control is measured once per task, inside the SPOT run, and copied into every other method's run, so every method of a task is normalized by the same number; all eight cards are the same model, each run ran alone on its card, and a control that came out a wall outlier was re-run and the original archived. Four independent draft-free measurements of SDAR-8B b32 MBPP on four cards span 18.61–19.13 tok/s, a cross-invocation term of about 3% for the control. Drafted runs vary more: three passes of one transfer run (SDAR-8B GSM8K under the mini-trained drafter) on three cards gave identical NFE and throughput ratios of 2.45, 2.17 and 2.49 at $D8$; the outlying pass was archived and the two agreeing passes' value is reported. The comparisons this paper draws are an order of magnitude larger than either term, and margins under about 5% are reported but not called wins. Honest oracle throughput subtracts the oracle's lookahead forwards from wall time. Accuracy uses the task's standard scorer on the generated text.

E DATASETS

GSM8K: the `main` test split (1319 problems), first 100, zero-shot chat prompt, answer extracted after "####" or as the last number. MBPP: the `full` test split, first 100 problems (task ids 11–110), 3-shot with the exemplars of the `prompt` split as in the MBPP paper and lm-eval, answers wrapped in `[BEGIN]`/`[DONE]` and executed against the reference tests. HumanEval: the 164-problem test set, first 100, completion executed against the reference tests. MATH500: the standard 500-problem subset of MATH, first 100, exact-match on the boxed answer after normalization. For every task the first-100 samples are the exact prefix of the full-set order the released harness uses, so runs at larger n extend these rather than replace them; the draft-free accuracies on the 100-prompt subsets (Table 2) differ from full-set numbers by subset composition (HumanEval's first 100 problems are easier than its full 164). The prompt templates and stopping rules are those of the released harness and ship with it.

F EXTENDED RESULTS

F.1 EXTENDED COMPARISON AGAINST RELATED METHODS

Table 6 compares the published diffusion drafters on the axes this paper argues are the relevant ones.

Table 6: Speculative decoding for diffusion targets, on the axes this paper argues are the relevant ones. "Learns order" = the commit order is predicted by a trained model rather than read off stale confidence or fixed offline; "multi-step" = one drafted proposal spans more than one commit; "exact $T{>}0$" = the committed distribution is provably the target's at sampling temperature.

method	drafter	size	learns order	multi-step	exact $T{>}0$	single device
Spiffy (2026)	offline graph	—	✗	✓	✗	✓
SSD (2025)	self (confidence)	—	✗	✓	✗	✓
BlockSpec (2025)	self (conf. tree)	—	✗	✓	✗	✓
SimSD (2026)	sibling model	1.7B	✓	✓	approx.	✗
MRP (2026)	residual head	3 layers	✗	✓	✗	✓
SPOT (ours)	order model	0.62M	✓	✓	✓	✓

F.2 ORACLE GRID

Table 7 is the full live-oracle grid behind Figure 2(a): NFE (reduction over the setting's draft-free NFE) and honest throughput ratio for the position, token and joint oracles at $D \in \{8, 12, 20\}$ on the eight SDAR-8B settings and on LLaDA2.0-mini MBPP. Honest throughput subtracts the oracle's lookahead forwards from wall time (the correction removes 85–96% of the oracle's raw wall) and

divides by the draft-free pass of the *same invocation*: the grid of one setting can be assembled from per-mode runs and crash resumes, each of which re-runs that pass, so only its own pass shares the machine state of its oracle rows. On the SDAR-8B b32 MATH500 setting the position and token rows come from a pass that ran at 14.1 tok/s and the joint rows from one at 18.4; pairing each with its own pass is what makes the three oracles comparable (Appendix G). NFE is monotone in the budget for every oracle. At b32 the throughput optimum is interior, $D12$ for the joint oracle on every setting; at b16 the narrower block makes each replica cheaper and honest throughput is still rising at $D20$, the largest budget in the grid, so that optimum lies outside the measured range.

Table 7: Live-oracle grid, $T{=}0$, generation length 256, $n{=}100$, seed 0. Per setting: NFE (reduction $\times$ over the setting's draft-free NFE) and honest tok/s ratio over the setting's draft-free throughput, for the position, token and joint oracles at $D8$/$D12$/$D20$; the draft-free column gives the baseline NFE and tok/s. Bold: best NFE and best throughput per setting.

			position oracle			token oracle			joint oracle		
setting	metric	draft-free	$D8$	$D12$	$D20$	$D8$	$D12$	$D20$	$D8$	$D12$	$D20$
SDAR-8B b32 GSM8K	NFE	271.1	50.1 (5.4×)	43.0 (6.3×)	38.2 (7.1×)	68.6 (3.9×)	64.8 (4.2×)	62.1 (4.4×)	46.8 (5.8×)	38.2 (7.1×)	**31.3 (8.7×)**
	tok/s ratio	18.4 tok/s	3.89×	4.02×	3.25×	2.81×	2.60×	1.89×	4.19×	**4.56×**	4.02×
SDAR-8B b32 MBPP	NFE	267.1	46.8 (5.7×)	39.7 (6.7×)	33.0 (8.1×)	53.7 (5.0×)	47.2 (5.7×)	40.9 (6.5×)	43.7 (6.1×)	35.4 (7.5×)	**27.5 (9.7×)**
	tok/s ratio	18.7 tok/s	3.25×	3.36×	3.05×	2.82×	2.77×	2.35×	3.50×	**3.77×**	3.53×
SDAR-8B b32 HumanEval	NFE	272.6	47.5 (5.7×)	40.0 (6.8×)	33.4 (8.2×)	62.3 (4.4×)	57.6 (4.7×)	53.6 (5.1×)	46.1 (5.9×)	37.9 (7.2×)	**30.3 (9.0×)**
	tok/s ratio	18.9 tok/s	3.91×	4.15×	3.58×	2.96×	2.81×	2.13×	4.05×	**4.42×**	3.97×
SDAR-8B b32 MATH500	NFE	272.9	54.2 (5.0×)	48.5 (5.6×)	45.8 (6.0×)	82.7 (3.3×)	81.8 (3.3×)	81.8 (3.3×)	48.9 (5.6×)	40.3 (6.8×)	**34.1 (8.0×)**
	tok/s ratio	16.3 tok/s	4.42×	**4.45×**	3.40×	2.91×	2.62×	1.87×	4.12×	4.42×	3.71×
SDAR-8B b16 GSM8K	NFE	263.7	58.2 (4.5×)	55.7 (4.7×)	47.5 (5.6×)	74.5 (3.5×)	74.1 (3.6×)	72.0 (3.7×)	54.3 (4.9×)	51.8 (5.1×)	**39.8 (6.6×)**
	tok/s ratio	18.6 tok/s	4.21×	4.26×	4.82×	3.33×	3.20×	3.27×	4.52×	4.58×	**5.57×**
SDAR-8B b16 MBPP	NFE	262.0	54.4 (4.8×)	53.6 (4.9×)	40.9 (6.4×)	60.9 (4.3×)	59.7 (4.4×)	59.4 (4.4×)	52.0 (5.0×)	50.9 (5.2×)	**36.6 (7.2×)**
	tok/s ratio	18.7 tok/s	3.84×	3.83×	4.51×	3.37×	3.28×	3.20×	4.10×	4.17×	**5.08×**
SDAR-8B b16 HumanEval	NFE	262.8	53.5 (4.9×)	52.2 (5.0×)	40.1 (6.6×)	65.2 (4.0×)	64.5 (4.1×)	59.1 (4.4×)	52.6 (5.0×)	51.0 (5.2×)	**37.4 (7.0×)**
	tok/s ratio	19.0 tok/s	4.38×	4.36×	5.28×	3.59×	3.50×	3.67×	4.45×	4.48×	**5.67×**
SDAR-8B b16 MATH500	NFE	263.3	61.8 (4.3×)	58.8 (4.5×)	54.2 (4.9×)	86.8 (3.0×)	86.4 (3.0×)	86.3 (3.1×)	55.8 (4.7×)	52.5 (5.0×)	**42.3 (6.2×)**
	tok/s ratio	19.4 tok/s	3.90×	3.93×	4.14×	2.70×	2.58×	2.54×	4.35×	4.43×	**5.21×**
LLaDA2.0-mini b32 MBPP	NFE	270.4	63.1 (4.3×)	58.1 (4.7×)	56.3 (4.8×)	92.2 (2.9×)	92.0 (2.9×)	91.5 (3.0×)	59.6 (4.5×)	53.0 (5.1×)	**48.6 (5.6×)**
	tok/s ratio	6.3 tok/s	3.31×	3.36×	3.21×	2.33×	2.15×	2.02×	3.65×	3.74×	**3.90×**

F.3 More models

The main comparison repeated on the three non-headline targets the recipe was trained for: **SDAR-4B** at $T{=}0$ and $T{=}1$ (Tables 8 and 9), **SDAR-1.7B** at both (Tables 10 and 11), and **LLaDA2.0-mini** at $T{=}0$ (Table 12) with its $T{=}1$ pair in Table 13. The recipe is unchanged in each case: one checkpoint, one calibration, the arms and budgets of Table 2.

Table 8: Main results, SDAR-4B b32, $T{=}0$ (format as Table 2). SPOT has the lowest NFE and the highest throughput ratio in every ours-vs-baseline comparison. The ratio's standard deviation over seeds is at most 0.24.

NFE↓ (tok/s↑)	GSM8K acc. 0.86–0.87		MBPP acc. 0.25–0.26		HumanEval acc. 0.59–0.60		MATH500 acc. 0.13–0.14	
method	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$
no draft ($D0$)	270.8↓ (1.00↑)		269.2↓ (1.00↑)		271.4↓ (1.00↑)		271.8↓ (1.00↑)	
SPOT (ours)	**68.4** (**3.17**)	**51.2** (2.25)	**54.6** (3.48)	**37.9** (2.26)	**61.5** (**3.48**)	**44.9** (2.51)	**82.2** (2.63)	**64.7** (1.73)
SPOT-γ (ours)	68.5 (3.12)	53.1 (**3.04**)	54.9 (**3.69**)	39.9 (**3.34**)	61.8 (3.34)	46.1 (**3.28**)	82.4 (**2.72**)	68.3 (**2.38**)
Spiffy graph	91.1 (2.71)	71.3 (1.77)	82.5 (2.52)	59.8 (1.54)	88.6 (2.69)	66.6 (1.81)	97.6 (2.50)	80.5 (1.54)
SSD free chain	91.5 (2.69)	83.7 (1.49)	83.2 (2.41)	74.5 (1.16)	89.4 (2.71)	78.5 (1.55)	98.9 (2.47)	91.4 (1.34)
BlockSpec tree	136.2 (1.74)	114.3 (1.04)	135.1 (1.56)	111.1 (0.79)	135.9 (1.74)	113.3 (1.00)	136.1 (1.76)	115.5 (1.01)

The smallest target is where padding costs most. SDAR-1.7B is the only setting in the benchmark where a baseline takes any throughput column from SPOT: Spiffy is ahead by 0.01–0.14 on GSM8K at $D20$ and on MATH500 at both budgets, the three settings with the lowest acceptance. The NFE ordering is untouched – SPOT drafts 14–23% fewer forwards than Spiffy in those same cells – so the gap is verification cost, not drafting: a 1.7B forward is cheap enough that the $(1{+}D)L$ padded positions are a large share of it. SPOT-γ, which drops the padding, recovers all but one of those columns at $T{=}0$ and all of them at $T{=}1$.

Table 9: SDAR-4B b32 at T=1 (coupled-noise match; free chain also under rejection sampling). Every comparison won on both columns. Over seeds the ratio's standard deviation is at most 0.16 and NFE's at most 0.6.

NFE↓ (tok/s↑)	GSM8K acc. 0.86–0.88		MBPP acc. 0.19–0.23		HumanEval acc. 0.56–0.60		MATH500 acc. 0.14–0.16	
method	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$
no draft ($D0$)	270.8↓ (1.00↑)		269.2↓ (1.00↑)		271.4↓ (1.00↑)		271.8↓ (1.00↑)	
SPOT (ours)	**71.0** (3.01)	**53.4** (2.26)	**57.0** (3.35)	**40.2** (2.17)	**63.0** (3.47)	**46.2** (2.45)	**86.0** (2.53)	**68.2** (1.71)
SPOT-γ (ours)	71.5 (**3.14**)	56.0 (**3.06**)	57.9 (**3.44**)	42.9 (**3.24**)	63.1 (**3.56**)	48.1 (**3.31**)	86.7 (**2.60**)	72.6 (**2.27**)
Spiffy graph	94.0 (2.60)	74.4 (1.81)	85.0 (2.47)	62.9 (1.50)	90.3 (2.75)	68.3 (1.85)	102.5 (2.40)	86.4 (1.48)
free chain, match	94.4 (2.61)	86.5 (1.51)	85.9 (2.46)	77.3 (1.21)	90.9 (2.77)	79.9 (1.60)	103.6 (2.34)	96.6 (1.31)
free chain, RS	88.5 (1.89)	84.2 (1.17)	61.1 (2.55)	50.5 (1.53)	76.6 (2.15)	68.2 (1.43)	111.0 (1.50)	109.9 (0.88)
BlockSpec tree	139.0 (1.75)	117.8 (1.05)	137.9 (1.48)	114.4 (0.75)	137.8 (1.76)	115.5 (1.05)	141.0 (1.72)	121.3 (1.01)

Table 10: Main results, SDAR-1.7B b32, T=0 (format as Table 2). SPOT has the lowest NFE in every ours-vs-baseline comparison and the higher throughput ratio in 21 of 24; SPOT-γ, which drops the padding, leads in 23. The ratio's standard deviation over seeds is at most 0.17.

NFE↓ (tok/s↑)	GSM8K acc. 0.70–0.72		MBPP acc. 0.27–0.31		HumanEval acc. 0.45–0.47		MATH500 acc. 0.09–0.10	
method	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$
no draft ($D0$)	270.8↓ (1.00↑)		269.2↓ (1.00↑)		271.4↓ (1.00↑)		271.8↓ (1.00↑)	
SPOT (ours)	**75.5** (2.79)	**58.8** (2.87)	**55.4** (3.61)	**38.9** (3.34)	**64.6** (3.15)	**46.4** (3.26)	**85.1** (2.49)	**67.7** (2.51)
SPOT-γ (ours)	75.9 (**2.95**)	62.2 (**3.40**)	55.7 (**3.74**)	40.5 (**4.48**)	64.7 (**3.24**)	48.2 (**4.04**)	85.3 (2.50)	71.7 (**2.82**)
Spiffy graph	94.0 (2.63)	76.8 (2.93)	84.5 (2.87)	64.4 (2.58)	89.2 (2.69)	69.4 (3.09)	98.6 (**2.51**)	82.8 (2.64)
SSD free chain	94.6 (2.55)	88.0 (2.52)	91.9 (2.61)	76.5 (2.19)	90.0 (2.62)	82.0 (2.54)	99.1 (2.44)	92.7 (2.36)
BlockSpec tree	136.6 (1.71)	115.6 (1.68)	135.0 (1.73)	111.5 (1.37)	135.4 (1.73)	112.8 (1.67)	136.1 (1.69)	115.7 (1.64)

Table 11: SDAR-1.7B b32 at T=1 (coupled-noise match; free chain also under rejection sampling). Lowest NFE in every comparison and the higher throughput ratio in 28 of 32; SPOT-γ wins every comparison on both columns. The ratio's standard deviation over seeds is at most 0.12.

NFE↓ (tok/s↑)	GSM8K acc. 0.69–0.71		MBPP acc. 0.23–0.28		HumanEval acc. 0.37–0.40		MATH500 acc. 0.10–0.11	
method	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$
no draft ($D0$)	270.8↓ (1.00↑)		269.2↓ (1.00↑)		271.4↓ (1.00↑)		271.8↓ (1.00↑)	
SPOT (ours)	**79.1** (2.58)	**62.0** (2.57)	**57.5** (3.49)	**40.7** (3.26)	**67.1** (3.10)	**48.9** (3.16)	**88.9** (2.30)	**72.0** (2.29)
SPOT-γ (ours)	79.8 (**2.60**)	66.2 (**2.93**)	58.4 (**3.56**)	43.4 (**4.23**)	67.5 (**3.28**)	51.7 (**3.95**)	90.1 (**2.40**)	76.7 (**2.67**)
Spiffy graph	98.7 (2.31)	82.3 (2.57)	86.5 (2.78)	66.6 (2.62)	91.7 (2.70)	72.4 (3.07)	104.5 (**2.40**)	90.0 (2.52)
free chain, match	99.0 (2.32)	92.6 (2.29)	93.5 (2.56)	78.7 (2.14)	92.6 (2.69)	84.6 (2.64)	104.8 (2.36)	99.0 (2.25)
free chain, RS	99.0 (1.43)	96.2 (1.23)	71.4 (2.05)	67.6 (1.60)	87.8 (1.72)	83.5 (1.49)	115.8 (1.26)	115.4 (1.06)
BlockSpec tree	141.4 (1.55)	121.2 (1.53)	137.5 (1.68)	114.3 (1.36)	138.2 (1.73)	116.0 (1.72)	141.5 (1.68)	122.4 (1.62)

Table 12: Main results, LLaDA2.0-mini b32 (MoE, exact prefix-KV cache), T=0, n=100; MBPP and GSM8K with three seeds, MATH500 seed 0, HumanEval seed 0 except two $D8$ baseline entries that are two-seed means. SPOT has the lowest NFE in all 36 comparisons and the higher throughput ratio in 34; the two exceptions are MBPP and HumanEval at $D12$, where Spiffy is ahead by 0.7% and 0.1%; mini's per-round acceptance is low for every drafter (1.05–3.77 accepted tokens per drafted round across the four tasks), so the wall-clock margins are thin on the code tasks.

NFE↓ (tok/s↑)	GSM8K acc. 0.59–0.65		MBPP acc. 0.28–0.35		HumanEval acc. 0.66–0.68		MATH500 acc. 0.10–0.12	
method	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$
no draft ($D0$)	271.6↓ (1.00↑)		270.4↓ (1.00↑)		269.3↓ (1.00↑)		270.2↓ (1.00↑)	
SPOT (ours)	**83.6** (**2.32**)	**64.0** (**2.62**)	**96.0** (**2.04**)	**75.6** (**2.16**)	**88.5** (**2.86**)	**65.8** (**3.27**)	**85.1** (**2.33**)	**66.5** (**2.59**)
Spiffy graph	98.0 (1.59)	80.8 (2.22)	103.3 (2.01)	86.6 (2.03)	99.8 (2.69)	78.7 (3.05)	98.7 (2.11)	80.6 (2.28)
SSD free chain	95.9 (2.14)	90.1 (2.08)	103.0 (1.99)	96.3 (1.91)	100.5 (2.73)	88.8 (2.77)	97.7 (1.58)	91.4 (1.59)
BlockSpec tree	136.8 (1.44)	117.2 (1.36)	136.2 (1.49)	116.7 (1.49)	134.9 (1.91)	115.3 (1.96)	135.5 (1.46)	116.3 (1.51)

Table 13: LLaDA2.0-mini b32 at $T{=}1$ (MBPP, $n{=}100$, seed 0), the setting's only $T{=}1$ evidence: SPOT against a drafter tuned for this setting alone.

	NFE↓ (tok/s↑)		
drafter	$D8$	$D12$	$D20$
SPOT	102.3 (2.00)	94.1 (2.04)	82.1 (2.00)
setting-specific drafter	108.5 (1.80)	99.0 (1.87)	85.1 (1.93)

The full transfer matrix. Table 14 points every trained checkpoint at every target it can drive, at $D8$. The native drafter's advantage shrinks with target size: on SDAR-8B it has the lowest NFE on all four tasks and the best transferred checkpoint costs 3.4–7.4%, on SDAR-4B it leads three of four at 1.5–2.8%, and on SDAR-1.7B the four checkpoints are interchangeable within 0.4–1.9%, with the SDAR-4B drafter taking GSM8K and the mini drafter MATH500 from the native one. LLaDA2.0-mini, the one target with a different tokenizer, keeps the widest native margin at 3.4–11.4%. On LLaDA-8B and Dream-7B, which have no native drafter at all, all four checkpoints land within about 4% of each other on every task. What the drafter learns is largely a property of block-diffusion decoding rather than of one target.

Table 14: Every checkpoint against every target it can drive, $T{=}0$, b32, $D8$, padded. *marks the target's native drafter, the one trained on it; all other rows are transferred with no retraining or recalibration. Ratio is against that target's own draft-free control, best per column within a target in bold. The LLaDA2.0-mini rows are one seed. †no exact prefix cache, as in Table 3.

		NFE↓ (tok/s↑) at $D8$			
target	drafter	GSM8K	MBPP	HumanEval	MATH500
SDAR-8B	SDAR-8B*	**69.6** (**2.50**)	**56.7** (**2.61**)	**62.2** (2.74)	**84.5** (**2.03**)
	SDAR-4B	71.4 (2.37)	57.6 (2.44)	64.1 (**2.75**)	85.9 (1.98)
	SDAR-1.7B	72.0 (2.36)	57.4 (2.46)	65.3 (2.69)	88.9 (1.86)
	mini	72.8 (2.33)	58.6 (2.34)	66.8 (2.48)	87.9 (1.91)
SDAR-4B	SDAR-8B	69.3 (3.11)	**54.6** (3.42)	63.3 (3.29)	83.7 (2.60)
	SDAR-4B*	**68.4** (**3.17**)	**54.6** (3.48)	**61.5** (**3.48**)	**82.2** (**2.63**)
	SDAR-1.7B	69.4 (3.09)	54.8 (**3.52**)	62.9 (3.40)	83.9 (2.57)
	mini	69.0 (**3.16**)	55.4 (3.43)	62.0 (3.39)	82.6 (2.60)
SDAR-1.7B	SDAR-8B	75.8 (2.65)	55.4 (3.51)	66.3 (3.08)	86.2 (2.39)
	SDAR-4B	**75.3** (2.77)	55.9 (3.57)	**64.3** (3.04)	85.2 (2.42)
	SDAR-1.7B*	75.5 (2.79)	**55.4** (**3.61**)	64.6 (**3.15**)	85.1 (**2.49**)
	mini	75.6 (**2.81**)	56.4 (3.54)	64.8 (3.08)	**84.4** (2.46)
LLaDA2.0-mini	SDAR-8B	85.9 (1.86)	105.2 (**2.68**)	97.8 (1.95)	87.1 (2.25)
	SDAR-4B	84.7 (**2.41**)	100.9 (2.52)	92.9 (2.08)	85.4 (2.32)
	SDAR-1.7B	85.3 (2.30)	98.5 (2.47)	98.2 (1.92)	88.1 (2.27)
	mini*	**83.0** (2.34)	**94.9** (2.23)	**88.2** (**2.23**)	**84.8** (**2.35**)
LLaDA-8B†	SDAR-8B	**79.8** (1.61)	**54.5** (**2.13**)	83.3 (**1.52**)	83.6 (1.55)
	SDAR-4B	80.0 (**1.66**)	54.7 (2.09)	**82.8** (1.50)	**83.3** (**1.57**)
	SDAR-1.7B	83.2 (1.64)	55.4 (2.09)	86.4 (1.44)	86.1 (1.54)
	mini	83.1 (1.57)	56.7 (1.93)	87.6 (1.42)	86.3 (1.48)
Dream-7B†	SDAR-8B	**66.0** (1.89)	**51.2** (2.41)	57.3 (2.15)	77.2 (1.58)
	SDAR-4B	66.2 (**1.92**)	**51.3** (**2.42**)	**56.9** (2.22)	**76.8** (**1.64**)
	SDAR-1.7B	67.0 (1.91)	51.6 (2.39)	58.2 (**2.23**)	78.4 (1.59)
	mini	67.8 (1.84)	52.0 (2.33)	58.3 (2.16)	79.1 (1.48)

F.4 More block sizes

The b16 rows of Table 7 give the oracle grid at block width 16 (joint ceiling 6.2–7.2× NFE, 5.1–5.7× honest wall), and Table 15 the four-setting recipe transfer on MBPP: with the recipe frozen (same architecture, training procedure and calibration), the single SPOT drafter is at or ahead of a drafter tuned separately for each setting, on every setting and budget, by 0.4–1.0 NFE on SDAR b32, 0.9–1.3 NFE and 0.14–0.21 in throughput ratio on SDAR b16, 1.1–3.3 NFE on mini b32 and 4.6–6.5 NFE on mini b16.

Table 15: Recipe transfer across widths and targets (MBPP, T=0, n=100): the single SPOT recipe vs. a drafter tuned separately for each setting.

	NFE↓ (tok/s↑)		
setting	$D8$	$D12$	$D20$
SDAR-8B b32	55.6 (2.64)	47.1 (2.42)	38.7 (1.75)
SDAR-8B b16	60.8 (3.46)	58.7 (3.29)	44.4 (3.23)
mini b32	96.0 (2.04)	86.9 (2.12)	75.6 (2.16)
mini b16	90.5 (1.98)	82.1 (2.08)	72.6 (2.19)

F.5 Generation length

Table 16 repeats the T=1 MBPP comparison at generation lengths 64 and 512 beside the 256 of the main text, each against its own draft-free control. **The gains grow with generation length.** SPOT's throughput ratio over draft-free decoding at $D8$ rises from 1.56× at length 64 to 2.45× at 256 and 2.79× at 512, and SPOT-γ's best ratio from 1.87× to 2.86× to 3.23×. The margin over the strongest baseline grows with it: against Spiffy at $D8$, SPOT takes 11% fewer forwards at 1.08× its throughput at length 64, 30% fewer at 1.38× at 256, and 37% fewer at 1.50× at 512. The reason is amortisation. A block costs $(1+D)L$ positions to verify whether or not the draft is accepted, and a longer generation spreads that fixed cost over more accepted tokens, which also lifts the NFE reduction over draft-free decoding from 2.8× at length 64 to 4.5× at 256 and 5.2× at 512.

The same accounting sets the limit at the short end. At length 64 the generation is two blocks, there is almost nothing to amortise against, and at $D20$ the padded recipe falls below draft-free decoding (0.89×) exactly as the baselines do (0.54–0.76×). SPOT still has the lowest NFE in every column, and SPOT-γ, which drops the padding, is flat in the budget at 1.87× for both $D8$ and $D20$: at short generations the unpadded recipe is the one to deploy. At length 512 no such caveat applies, and SPOT has both the lowest NFE and the highest throughput ratio against every baseline at both budgets.

Table 16: Generation length, SDAR-8B b32 MBPP at T=1: NFE per sample (throughput ratio over that length's own draft-free control), best in bold. The 256 columns are the T=1 MBPP block of Table 2.

	length 64		length 256		length 512	
method	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$
no draft ($D0$)	77.2↓ (1.00↑)		269.2↓ (1.00↑)		525.2↓ (1.00↑)	
SPOT (ours)	**27.5** (1.56)	**22.2** (0.89)	**60.1** (2.45)	**42.6** (1.59)	**100.2** (2.79)	**66.7** (1.98)
SPOT-γ (ours)	31.8 (**1.87**)	30.3 (**1.87**)	64.9 (**2.80**)	52.1 (**2.86**)	105.1 (**3.10**)	76.3 (**3.23**)
Spiffy graph	30.9 (1.45)	27.0 (0.76)	86.3 (1.77)	64.8 (1.07)	158.4 (1.86)	112.9 (1.21)
free chain, match	31.0 (1.38)	29.6 (0.65)	87.4 (1.76)	79.1 (0.88)	159.5 (1.82)	143.1 (0.92)
free chain, RS	31.0 (1.10)	29.9 (0.56)	64.5 (1.86)	52.4 (1.12)	104.6 (2.34)	76.6 (1.60)
BlockSpec tree	42.0 (1.06)	36.5 (0.54)	138.3 (1.07)	115.2 (0.57)	266.3 (1.05)	219.2 (0.57)

F.6 Composition with multi-token decoding

The draft-free control of the main text commits one position per forward: exact, but not the fastest decoder. The threshold rule of Wu et al. (2026) commits every masked position whose confidence

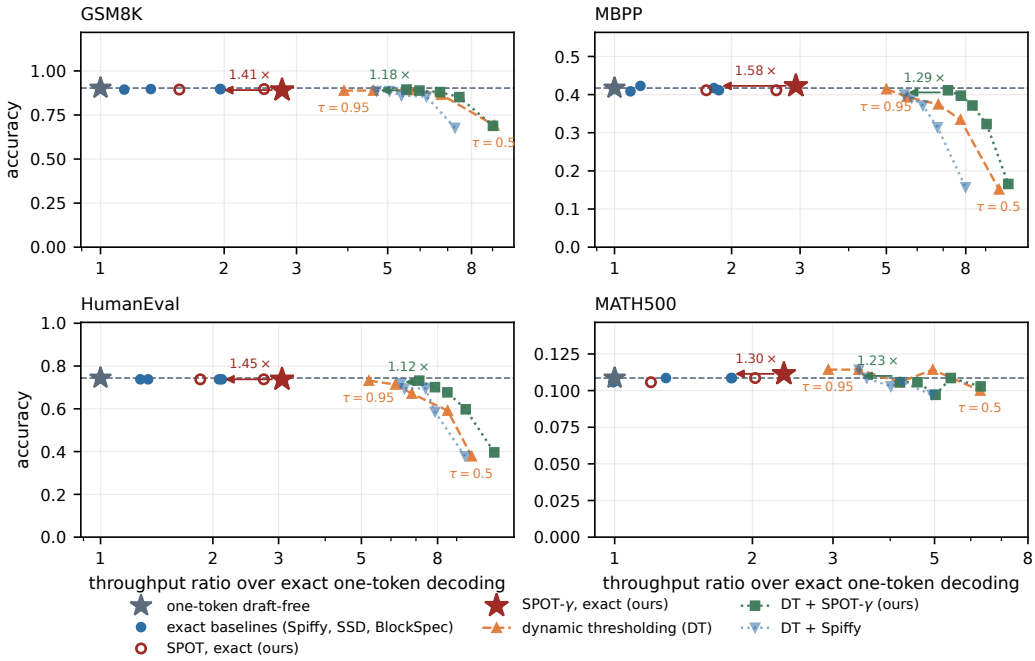


Figure 3: The speed–quality plane, SDAR-8B b32 at T=0, n=350 over three seeds. Throughput is the ratio over the exact one-token draft-free control of the same seed (dashed line: that control's accuracy). Exact methods keep the target's commit law and hold that line; the red arrow spans the best exact baseline at any budget to SPOT-γ. Threshold decoding leaves the line, tracing the orange curve as τ falls from 0.95 to 0.5; adding a drafter moves a curve right at unchanged accuracy. The green arrow is SPOT-γ over Spiffy's graph at τ=0.95 (1.12–1.34× across all bars). Over seeds the ratio's relative standard deviation is at most 10%, median 2%.

exceeds τ in the same forward, from a factorised approximation of the joint. SPOT is exact with respect to whichever commit rule the target runs, since verification is state equality and the tree is keyed by the *set* of committed pairs, so it composes with this rule unchanged. Table 17 runs both decoders in one lane at τ=0.9, every ratio against the exact control. The lossy rule alone is 3.4–6.2× that control at up to 0.03 accuracy cost; lossless SPOT-γ is 2.3–3.1× at no cost; on top of the lossy rule SPOT still removes 36–42% of the forwards and SPOT-γ adds 1.28–1.38× throughput. The margin is smaller than in exact mode because a threshold forward already commits several positions, and because the order model predicts the walk but not where a multi-position step ends, so only a drafted state equal to the whole committed set is accepted.

Sweeping the bar. Figure 3 varies τ from 0.95 to 0.5. The lossy rule's speed is bought with quality once the bar drops: 2.9–5.2× at τ=0.95 for at most 0.01 accuracy, but at τ=0.5 its 6.3–9.9× costs 23% of GSM8K's accuracy, 49% of HumanEval's and 64% of MBPP's. Adding SPOT-γ leaves accuracy at matched τ within 0.018 and raises throughput by 1.37–1.44× at τ=0.95, shrinking to 0.99–1.15× at 0.5. At fixed accuracy, SPOT-γ is 1.18–1.44× faster than the lossy decoder tuned to the same quality on GSM8K, HumanEval and MBPP (MATH500's 0.11 accuracy does not separate the bars), and it leads Spiffy's graph in all 20 task-and-bar settings.

F.7 More ablations and curves

Full budget grid. Tables 19 and 20 repeat Table 2 with the $D12$ column.

Budget curves. Table 21 and Figure 4 give SPOT's (padded, γ=0) response to the budget on the four SDAR-8B tasks and on mini MBPP. NFE is monotone to $D20$ on every setting; the throughput optimum sits at $D6$ on every SDAR-8B task (3.04× MBPP, 3.13× HumanEval, 2.80× GSM8K, 2.38× MATH500) and falls to 1.25–1.87× at $D20$, while on cached mini the curve rises slowly to

Table 17: Composition with threshold decoding, SDAR-8B b32 at T=0: NFE per sample (throughput ratio over the exact draft-free control of the same seed), best in the threshold block in bold, second best underlined. Shaded rows give each block's accuracy range. Threshold rows commit every position above τ=0.9 per forward (Wu et al., 2026); SPOT-γ is unpadded with γ=0.1. NFE is seed-independent at T=0; over seeds the ratio's standard deviation is at most 0.29, under 6% of the ratio.

NFE↓ (tok/s↑)	GSM8K	MBPP	HumanEval	MATH500
exact decoding, one commit per forward	acc. 0.89–0.90	acc. 0.42–0.42	acc. 0.74–0.74	acc. 0.11–0.11
no draft ($D0$)	270.8 (1.00)	269.2 (1.00)	271.4 (1.00)	271.8 (1.00)
SPOT-γ $D20$ (ours)	67.8 (2.77)	47.8 (2.93)	56.4 (3.06)	89.5 (2.34)
threshold decoding, τ=0.9 (lossy)	acc. 0.89–0.89	acc. 0.39–0.40	acc. 0.70–0.71	acc. 0.11–0.11
no draft ($D0$)	58.7 (4.61)	44.3 (5.65)	43.4 (6.17)	79.8 (3.41)
Spiffy graph $D8$	38.0 (5.05)	26.7 (5.81)	28.5 (6.53)	51.8 (3.56)
SPOT $D8$ (ours)	**36.5** (5.04)	**25.6** (6.13)	**27.9** (6.74)	**49.3** (3.72)
SPOT-γ $D8$ (ours)	40.5 (**5.98**)	29.0 (**7.77**)	30.8 (**7.87**)	54.9 (**4.59**)

Table 18: Ablations on SDAR-8B MBPP b32 at T=0 (ratio standard deviation over seeds at most 0.11). Top: recipe components. Bottom: padding and the mass floor γ, the two knobs that trade forwards for wall clock. The full γ grid and the τ ablation are in the released records.

	NFE↓ (tok/s↑)	
configuration	$D8$	$D20$
final recipe (tree, hedge 0.6, τ=1, γ=0, padded)	56.7 (2.61)	39.6 (1.72)
− token hedge (gate off)	57.0 (2.61)	40.3 (1.68)
chain instead of tree	64.0 (2.32)	51.5 (1.34)
− scaled training corpus	57.3 (2.60)	40.4 (1.72)
no padding, γ=0	56.7 (2.59)	39.6 (1.76)
no padding, γ=0.05	58.2 (2.83)	44.3 (2.84)
no padding, γ=0.1 (calibrated; SPOT-γ)	60.7 (2.98)	47.8 (2.99)
padded, γ=0.1	60.8 (2.43)	48.1 (1.45)

Table 19: SDAR-8B b32, T=0, all three budgets: GSM8K and MBPP. NFE is identical across seeds at T=0; the throughput ratio has standard deviation at most 0.11, median 0.04.

NFE↓ (tok/s↑)	GSM8K acc. 0.89–0.90			MBPP acc. 0.41–0.43		
method	$D8$	$D12$	$D20$	$D8$	$D12$	$D20$
no draft ($D0$)	270.8↓ (1.00↑)			269.2↓ (1.00↑)		
SPOT (ours)	**69.6** (2.50)	**60.9** (2.31)	**52.6** (1.56)	**56.7** (2.61)	**48.2** (2.38)	**39.6** (1.72)
SPOT-γ (ours)	75.1 (**2.76**)	70.9 (**2.83**)	67.8 (**2.78**)	60.7 (**2.98**)	54.1 (**3.06**)	47.8 (**2.99**)
Spiffy graph	91.3 (1.96)	82.8 (1.71)	71.9 (1.14)	83.4 (1.86)	74.9 (1.59)	61.3 (1.17)
SSD free chain	92.2 (1.95)	89.9 (1.63)	84.9 (1.00)	84.7 (1.80)	83.8 (1.40)	76.1 (0.91)
BlockSpec tree	136.4 (1.33)	128.4 (1.12)	114.6 (0.71)	135.0 (1.10)	126.9 (0.85)	111.5 (0.58)

Table 20: SDAR-8B b32, T=0, all three budgets: HumanEval and MATH500. Seed spread as in Table 19.

NFE↓ (tok/s↑)	HumanEval acc. 0.74–0.74			MATH500 acc. 0.10–0.11		
method	$D8$	$D12$	$D20$	$D8$	$D12$	$D20$
no draft ($D0$)	271.4↓ (1.00↑)			271.8↓ (1.00↑)		
SPOT (ours)	**62.2** (2.74)	**53.3** (2.67)	**44.6** (1.85)	**84.5** (2.03)	**75.1** (1.80)	**66.0** (1.20)
SPOT-γ (ours)	66.5 (**3.09**)	61.0 (**3.19**)	56.4 (**3.13**)	92.2 (**2.27**)	89.9 (**2.27**)	89.5 (**2.27**)
Spiffy graph	86.8 (2.11)	78.2 (1.87)	65.9 (1.28)	99.4 (1.80)	90.5 (1.58)	82.7 (0.99)
SSD free chain	87.0 (2.08)	85.3 (1.70)	80.2 (1.03)	100.5 (1.80)	96.7 (1.49)	93.6 (0.88)
BlockSpec tree	136.0 (1.34)	127.5 (1.14)	112.7 (0.73)	136.7 (1.29)	128.4 (1.11)	116.1 (0.68)

a plateau at $D12$–$D20$ (2.22× at $D16$): the optimum's location is a property of the target's forward economics. With the mass floor of SPOT-γ the SDAR curves flatten (Table 18).

Table 21: Budget response, T=0, n=100 (three seeds at every budget shown on the SDAR-8B settings; mini seed 0 except $D0/6/8/12/20$). Bold: throughput optimum.

		$D0$	$D3$	$D5$	$D6$	$D8$	$D12$	$D16$	$D20$
SPOT, GSM8K	NFE↓	271.1	102.4	84.2	78.0	69.5	60.8	54.7	52.6
	tok/s↑	1.00	2.38	2.69	**2.80**	2.60	2.43	1.85	1.63
SPOT, MBPP	NFE↓	267.1	87.7	70.9	63.9	55.6	47.1	39.7	38.7
	tok/s↑	1.00	2.25	2.80	**3.04**	2.70	2.47	2.15	1.79
SPOT, HumanEval	NFE↓	272.6	96.5	78.0	71.1	62.2	53.4	46.7	44.4
	tok/s↑	1.00	2.51	2.88	**3.13**	2.82	2.68	2.27	1.87
SPOT, MATH500	NFE↓	272.9	116.5	98.6	92.0	84.2	74.0	69.2	65.2
	tok/s↑	1.00	2.10	2.31	**2.38**	2.08	1.91	1.48	1.25
SPOT, mini MBPP	NFE↓	270.4	128.0	109.4	104.6	96.0	86.9	80.7	75.6
	tok/s↑	1.00	1.32	1.58	1.98	2.04	2.12	**2.22**	2.16

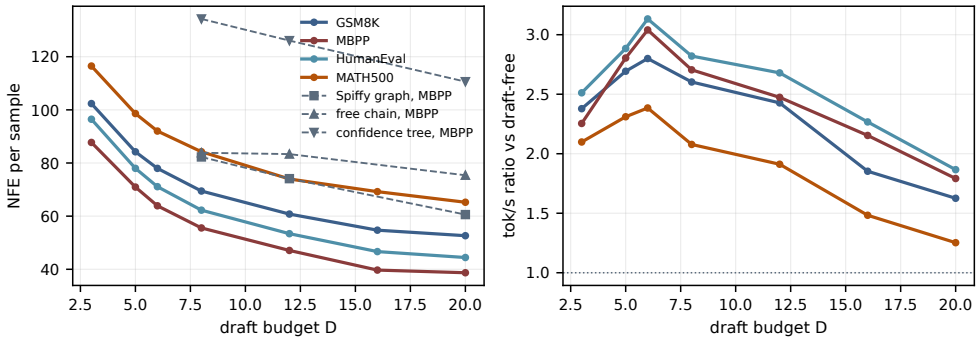


Figure 4: Budget curves (SDAR-8B, n=100, T=0). Left: NFE vs. draft budget for SPOT (solid) and, on MBPP, the fixed baselines at $D8/12/20$ (markers). Right: throughput ratio over the shared draft-free control for SPOT, non-monotone with an interior optimum at $D6$ on every task.

Temperature sweep. Table 1, discussed in Section 3.2, condenses the sweep to $D8$ and to SPOT and Spiffy; Table 22 is the complete grid: five temperatures, three drafters, three verifiers, $D8$ and $D20$, with the draft-free row of each temperature carrying the target's own accuracy, argmax fraction and mean committed-token probability. Two rows worth quoting: at T=1.5, coupled vs uncoupled

at $D20$ is 54.1 vs 62.7 NFE for SPOT and 73.4 vs 83.6 for Spiffy; the confidence tree, which ranks positions by stale confidence without the noise, gains little from coupling (121.6 vs 125.5 at the same budget). At T=2, where the target's samples are near-uniform, uncoupled acceptance all but disappears for every drafter (all three sit at ∼264 NFE against the 267.1 of draft-free decoding in this grid) while coupling still drafts, and Spiffy's noise-aware position ranking (stale confidence of the *drawn* token) overtakes the T=0-trained order model under coupling (114.6 vs 187.4 NFE at $D8$), an ordering that holds nowhere the target produces usable text.

Table 22: Complete temperature sweep, SDAR-8B MBPP b32, n=100, seed 0: NFE (tok/s ratio over the T-matched draft-free control) for every drafter under the three equality verifiers at $D8$ and $D20$. †greedy match is not distribution-preserving at $T > 0$ (Proposition 2 does not apply); its NFE is not comparable to the exact regimes' and its emitted text drifts from the target's distribution (Table 1).

		coupled-noise match		uncoupled match		greedy match†	
T	drafter	$D8$	$D20$	$D8$	$D20$	$D8$	$D20$
0.5	no draft ($D0$)	267.1 NFE, accuracy 0.35, argmax fraction 0.988, mean committed p 0.993					
	SPOT	56.8 (2.56)	39.2 (1.74)	56.7 (2.53)	39.1 (1.68)	55.9 (2.62)	38.9 (1.76)
	Spiffy graph	83.0 (1.83)	60.4 (1.16)	82.7 (1.87)	60.3 (1.19)	82.3 (1.83)	59.9 (1.16)
	BlockSpec-style tree	135.6 (1.12)	112.0 (0.60)	135.1 (1.10)	111.4 (0.59)	134.8 (1.12)	111.3 (0.60)
1.0	no draft ($D0$)	267.1 NFE, accuracy 0.38, argmax fraction 0.978, mean committed p 0.967					
	SPOT	58.3 (2.90)	40.8 (1.90)	58.2 (2.93)	40.9 (1.97)	55.6 (2.97)	38.6 (1.98)
	Spiffy graph	85.0 (2.15)	63.2 (1.33)	85.6 (2.13)	63.3 (1.33)	82.0 (2.08)	59.2 (1.33)
	BlockSpec-style tree	137.1 (1.23)	114.2 (0.66)	135.5 (1.22)	111.9 (0.66)	134.0 (1.32)	110.4 (0.71)
1.5	no draft ($D0$)	267.1 NFE, accuracy 0.11, argmax fraction 0.874, mean committed p 0.840					
	SPOT	71.6 (2.12)	54.1 (1.29)	81.6 (1.79)	62.7 (1.07)	55.4 (2.71)	39.7 (1.79)
	Spiffy graph	93.0 (1.61)	73.4 (0.92)	107.2 (1.47)	83.6 (0.85)	82.2 (1.92)	60.2 (1.22)
	BlockSpec-style tree	145.3 (1.00)	121.6 (0.52)	149.1 (1.03)	125.5 (0.54)	132.6 (1.08)	109.7 (0.57)
2.0	no draft ($D0$)	267.1 NFE, accuracy 0.00, argmax fraction 0.017, mean committed p 0.012					
	SPOT	192.3 (0.77)	150.3 (0.43)	264.0 (0.54)	263.5 (0.23)	103.9 (1.45)	71.3 (0.96)
	Spiffy graph	112.4 (1.39)	101.2 (0.69)	264.5 (0.54)	264.1 (0.23)	94.7 (1.66)	73.5 (0.98)
	BlockSpec-style tree	186.8 (0.77)	148.7 (0.42)	264.7 (0.57)	264.2 (0.25)	134.7 (1.08)	115.1 (0.55)
2.5	no draft ($D0$)	267.1 NFE, accuracy 0.00, argmax fraction 0.001, mean committed p 0.000					
	SPOT	196.9 (0.71)	154.2 (0.40)	267.0 (0.55)	267.0 (0.24)	133.6 (1.05)	101.0 (0.62)
	Spiffy graph	108.2 (1.38)	95.6 (0.69)	267.0 (0.55)	267.0 (0.24)	100.9 (1.44)	83.6 (0.79)
	BlockSpec-style tree	182.7 (0.81)	144.3 (0.44)	267.0 (0.56)	267.0 (0.24)	135.0 (1.11)	115.3 (0.57)

F.8 Additional verifier material

Table 23 isolates verifier and structure on SPOT's own proposals at T=1: the tree is worth 12–17 NFE over the same drafter's chain under the same verifier, and on chains RS and match are within 3 NFE of each other, with match faster at $D8$ and $D12$ and the two level at $D20$, where RS's lower NFE finally pays for the per-link acceptance test.

Table 23: Verifier and structure at T=1, SDAR-8B MBPP: SPOT's proposals under each verifier (match: coupled-noise match; RS: rejection sampling, chains only). Over seeds the ratio's standard deviation is at most 0.09.

		NFE↓ (tok/s↑)		
drafter	verifier	$D8$	$D12$	$D20$
SPOT	tree + match	60.1 (2.45)	51.5 (2.18)	42.6 (1.59)
SPOT	chain + match	71.6 (2.01)	65.6 (1.70)	59.3 (1.12)
SPOT	chain + RS	68.7 (1.91)	62.7 (1.64)	56.3 (1.12)

On chains, where both exact verifiers are defined, the earlier development record established a *regime law*: per link, match transfers the drafter's argmax skill through the shared noise, while RS degrades with $D_{TV}(\pi, q)$ rather than with argmax stability, so RS wins where the stale conditional is flat

(stale confidence $\lesssim 0.6$) and match where it is sharp; temperature moves the mix of regimes (the low-confidence share of rounds grows from $\sim$6% at T=0.5 to $\sim$46% at T=1.5) without changing the mechanism, and a sharper drafter raises match specifically. The final benchmark puts SPOT's chains in the sharp regime: match is 0.09 ahead in throughput ratio at $D8$, where RS pays an acceptance test per link, and the two are level at $D20$ (0.003 apart), where RS's 3.0 fewer forwards finally cover that cost. A confidence-gated hybrid (match above $\sim$ 0.6, RS below) is the natural chain-only refinement and remains unimplemented, since trees under match dominate every chain configuration.

F.9 Mechanism: what happens in deep blocks

Two measurements explain the gap structure of Table 2 (SDAR-8B MBPP b32, $D20$, seed 0). First, stale-order knowledge decays across blocks: the probability that the previous round's top-confidence position still commits falls $0.74/0.68/0.34/0.16/0.11/0.08/0.07/0.04/0.03$ across blocks 0–8, while the stale argmax token stays right ($0.92/0.94/0.99/1.00\ldots$)—deep blocks are near-deterministic in *what* and unpredictable in *which position first*, from stale confidence alone. Second, accepted tokens per drafted round by block:

block →	0	1	2	3	4	5	6	7	8
SPOT	2.29	2.67	5.83	9.54	12.30	13.76	14.50	14.50	14.50
Spiffy graph	1.88	2.23	3.45	4.48	4.74	5.05	5.18	5.14	5.15
SSD-style free chain	1.69	1.89	2.57	3.11	3.22	3.39	3.41	3.41	3.40
BlockSpec-style tree	1.55	1.39	1.50	1.55	1.57	1.58	1.58	1.58	1.58

Every drafter accepts more as the blocks become deterministic, but the amount is set by how deep a *correctly ordered* path each can propose: the learned, prefix-conditional order model rides the transition law to 14.5 accepted tokens per round, the fixed 18-node graph is capped by its depth-6 chains at $\sim$5, the free chain by its depth at $\sim$3.4, and the confidence tree spends its budget on mutually exclusive first-level siblings ($\sim$15 of 20 nodes) and reaches depth 2 or less in 90% of drafted rounds. Figure 5 shows the same pair graphically.

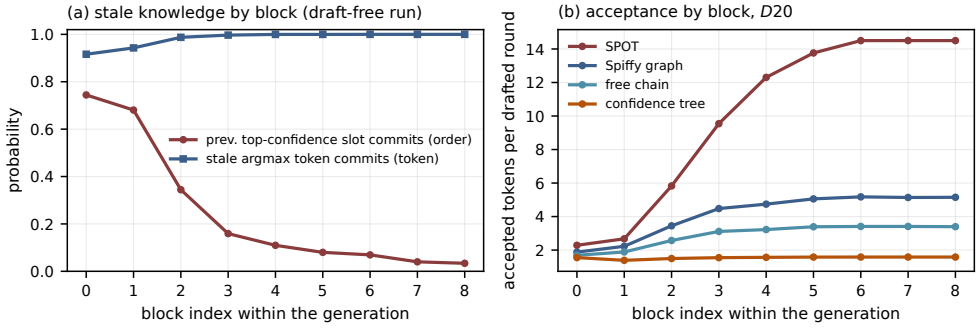


Figure 5: Left: stale-order knowledge decays across blocks (previous round's top-confidence position still commits) while the stale argmax token stays right. Right: accepted tokens per drafted round by block, $D20$, SDAR-8B MBPP.

F.10 Error analysis of the drafter

Every rejection is attributable: at the first rejected link of a drafted round, the fallback commit records the rank of the *true* position under the order model's proposal ordering and of the true token under the stale conditional. On the development full-set runs of the final drafter (the analysis predates the fixed-hardware benchmark; the drafter is unchanged), position misses account for 62–81% of first rejections and token misses for 3–6%; a growing share of rejections at $D20$ (32% on MBPP, 90% in the last block) occur with the truth ranked *first* by the order model—the tree ran out of nodes—which is why NFE keeps improving with budget and why the mass floor γ trades NFE for wall clock rather than being free. The drafter's prefix-mass score is nearly calibrated against realized

acceptance (96% acceptance in the top mass bin, 3% in the bottom; Figure 6). These proportions are reported as a diagnostic of the drafter and no claim in the main text rests on them.

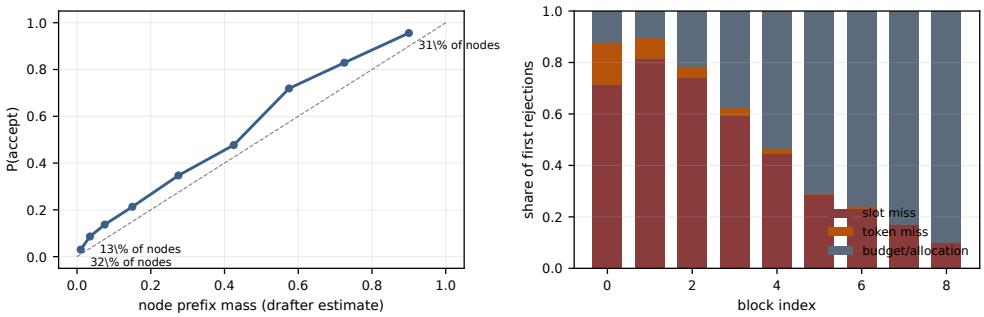


Figure 6: Left: acceptance calibration of the drafter's own node score (realized $P(\text{accept})$ per prefix-mass bin against the diagonal). Right: first-rejection attribution by block ($D20$): position misses dominate early, deep blocks are budget-bound. Development runs of the final drafter.

F.11 Drafter architecture

The capacity probe of the development record found the order model feature-limited rather than capacity-limited: at fixed inputs, $8\times$ the parameters changed held-out next-commit accuracy by $+0.004$, whereas broadening the training data moved decoding. The released drafter is three layers, d_{model}=128, 8 heads. The probe is offline: we did not sweep depth or width at decode time on the final benchmark, so the released size is a working choice rather than a tuned optimum.

G Reproducibility

Code and checkpoints are available at the following anonymized repoistory:

https://anonymous.4open.science/r/spot-speculative-decoding-E354

Seeds and wall clock. At T=0 the decoder is deterministic, so forwards and accuracy do not depend on the seed and only wall clock does. Every T=0 setting is run three times to measure that spread: across the 106 three-seed settings the max–min spread of the throughput ratio has median 0.09 and is at or below 0.15 in 89 and at or below 0.20 in 99. The one outlier is SDAR-4B HumanEval at $D8$ (0.57), the shortest task in the benchmark at 164 samples. At T=1 the target's own sampling moves the forwards as well: across the 120 three-seed T=1 settings the NFE standard deviation has median 0.24 and maximum 1.76, and accuracy stays within 0.06 of the control. A run whose throughput fell more than 10% from its sibling seeds for no configuration reason – machine state rather than method, since at T=0 the forwards are identical – was re-run and the original archived; this happened to 26 of ∼1500 cells, and to 14 of the 132 cells of the threshold lanes (Appendix F.6); two HumanEval cells reproduced their wall clock on a second card and are reported as measured. The benchmark completed 1197 jobs, and the threshold lanes 228 more, with no failures.

Oracle pairing. Oracle runs are unpadded. NFE is identical across invocations of a setting and wall clock is not, so every oracle row is divided by the draft-free pass of the invocation that produced it. On SDAR-8B b32 MATH500, whose position and token rows come from a pass at 14.1 tok/s and whose joint rows come from one at 18.4, dividing every row by the faster pass would understate the position oracle by 24%.

Hardware. One node, eight identical NVIDIA RTX 6000 Ada cards (48 GB), PyTorch 2.6.0, CUDA 12.4, Transformers 4.55.4; one run per card, never co-located. Runs disturbed by other users' jobs or by two host reboots were re-run from their last finished task.