PROSPECT: EFFICIENT AND CONSISTENT SELF-SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

In single-user and on-device settings, large language model (LLM) decoding typically runs at batch size one and is memory-bandwidth bound. Every forward pass streams the full model weights to emit a single token, leaving the GPU's arithmetic idle. Self-speculative decoding turns that idle arithmetic into tokens without adding any weights, by drafting the next block of tokens in parallel and verifying the previous block autoregressively in a single pass. There is no auxiliary drafter to train, keep aligned with the target, or hold in memory beside it, and no ceiling on draft quality set by a smaller model's capacity. We address two shortcomings of self-speculation. First, parallel drafts are sampled independently and can be individually plausible yet jointly incoherent. By front-loading the randomness of sampling, we generate consistent drafts without sacrificing parallelism. Second, a decode step cannot know how many of its drafts will be accepted, so a draft branch has to be appended for every possible acceptance count, which makes the query size quadratic in the draft length. By drafting only for the most likely acceptance counts, we reduce the query size to linear. The resulting decoder, *ProSpect*, decodes $3.9\times$ faster than autoregressive Qwen3-8B on SpeedBench, outperforming speculative decoding with dedicated drafters DSpark and DFlash by 28% and 43%, and $3.1\times$ faster than autoregressive Gemma-4-31B, ahead of DFlash by 12%.

1 INTRODUCTION

Autoregressive (AR) decoding in local and latency-critical applications, such as on-device assistants and single-user serving, is memory-bandwidth bound. Latency is dominated by streaming the model weights and KV cache from memory. The arithmetic units therefore sit idle, and speculative decoding (Leviathan et al., 2023; Chen et al., 2023) converts that idle capacity into lower latency. Several candidate tokens are drafted, verified against the target model in a single forward pass, and the longest correct prefix is committed, leaving what the model would have generated unchanged.

Classically the candidates come from a separate draft model (Li et al., 2024; Chen et al., 2026; Cheng et al., 2026b), a second set of weights that must be trained, kept aligned with the target and served beside it, and whose own forward passes add latency to every step. All of this competes for the memory bandwidth that binds at batch size one. In *multi-token prediction* (MTP) the model is its own drafter. Most MTP methods attach additional heads that each predict a token further ahead (Stern et al., 2018; Cai et al., 2024; Gloeckle et al., 2024; Cai et al., 2025). *Self-speculation* (Samragh et al., 2025; TiDAR, Liu et al., 2026; I-DLM, Yu et al., 2026) instead extends the input with a block of mask tokens, placeholders for the next few positions, and one forward pass drafts a token at each of them while verifying the previous step's drafts at the committed ones. Nothing beyond the model is trained or served, and draft quality is not capped by the capacity of a small head.

Self-speculation as practiced, however, has not been competitive with methods that use a separate draft model. We identify two shortcomings that hinder generation speed. First, parallel drafts sample from per-position marginals rather than the joint conditional, which sometimes results in mutually incompatible draft tokens. We front-load the randomness of sampling by sharing a per-position Gumbel-noise vector across drafting and verification. Sampling becomes a deterministic function of the logits, and reusing the same noise at verification lets the draft approximate the joint conditional

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.

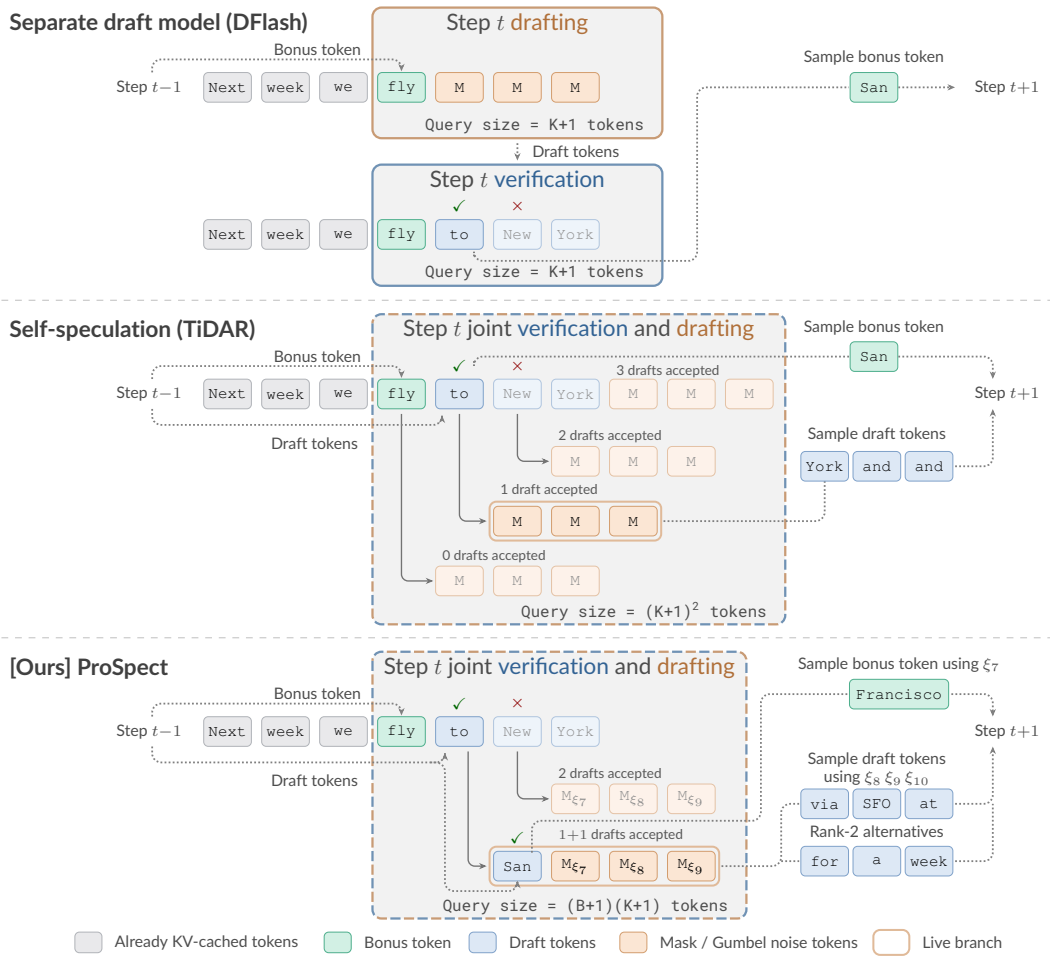


Figure 1: **Decoding with speculation.** Every method commits the longest prefix of drafts the base model would itself have produced, plus one bonus token. Each shaded panel is one forward pass, outlined by what the pass does: drafting (orange), verification (blue), or, with a border alternating the two, both together. *Top: a separate drafter* (Chen et al., 2026) spends two forward passes per step, one over a second set of weights to draft a block of K tokens and one over the target to verify them. *Middle: self-speculation* (Samragh et al., 2025; Liu et al., 2026) does both in a single pass, and since the acceptance count is unknown when the pass is built, it appends a branch of mask tokens `M` for every possible count and discards all but the live one. *Bottom: ProSpect* uses dedicated mask tokens $\texttt{M}_{\xi_i}$ to front-load the randomness of sampling, and places its B draft branches only where verification is most likely to land, which significantly reduces the query size. A branch may be headed by the position's rank-2 draft, here `San`, adding a potential extra accepted token.

rather than the product of marginals. Second, because self-speculation drafts and verifies in the same forward pass, it must draft before knowing how many of the current drafts verification will accept. Existing layouts therefore append a draft branch for every possible acceptance count, which makes the query size grow quadratically with the draft length (Samragh et al., 2025; Liu et al., 2026). Verification commits only the drafts up to the first rejection, so most of these branches are discarded. Instead of one draft branch per acceptance count, we allocate the branches to the most probable outcomes, re-ranking them before every forward pass by how likely verification is to accept them. Moreover, where the drafter is uncertain, we verify several candidates at the same position to boost the acceptance rate. The query size stays linear in the draft length.

Training touches only LoRA adapters and the mask-token embedding rows. On the serving side, drafting and verifying in one pass needs a non-causal mask, and the FlexAttention route taken by

prior work loses throughput as context grows. We implement the query instead as a cascade of FlashAttention calls inside vLLM, which closely matches the speed of the causal kernel.

On SpeedBench, ProSpect is the fastest method we measure on both Qwen3-8B and Gemma-4-31B, ahead of MTP and of speculative decoding with the dedicated drafters DFlash and DSpark. A model that drafts for itself can therefore take the place of these methods, with no additional weights to train.

Concretely, our main contributions are:

- A training recipe that front-loads the randomness of sampling, so that a parallel draft approximates the joint conditional rather than the product of its per-position marginals.
- A dynamic draft tree for parallel drafting: several candidates at the same position are verified where the draft is least confident, and which continuations to hedge is re-chosen every step by landing probability, so the query size stays linear in the draft length instead of quadratic.
- An inference engine built on vLLM, in which the drafting and verification layout is served by FlashAttention.
- A SpeedBench evaluation against both self-speculative decoders and standalone drafters, in which ProSpect outperforms DFlash and DSpark, and a study over eight reasoning benchmarks empirically confirming that ProSpect closely tracks base model accuracy.

2 Preliminaries: self-speculation

In self-speculation (Samragh et al., 2025; Liu et al., 2026; Yu et al., 2026) the target model is its own drafter. A single forward pass both verifies the drafts from the previous pass and produces the drafts for the next one (Figure 1 middle). The *query block* of that pass holds two kinds of positions, *verify* positions that carry the drafts from the previous pass as ordinary positions of the sequence, and *masked* positions appended after them for a continuation not yet committed to. Masked positions are invisible to the verify positions, so the autoregressive path through the sequence is exactly the base model's. The step commits the longest prefix of drafts that survives verification, plus a *bonus* token. Because a verify position predicts the next token, the position after the last accepted draft yields one token beyond the drafts, so a forward pass never commits fewer than one. Since the number of accepted tokens is unknown, the pass appends a mask block for every possible landing position, each conditioned on the drafts up to that point. Only the block that matches the accepted prefix is submitted for verification in the next step.

3 ProSpect

Self-speculation as described above carries two shortcomings. Parallel drafting fills a block in one pass, predicting only each position's marginal, so it cannot produce a jointly coherent draft. A step cannot know how many of its drafts will survive, so hedging every acceptance count makes the query grow quadratically in the draft length. We solve the first by front-loading the randomness of sampling with Gumbel-noise conditioning, and the second with dynamic draft trees that spend the query only on the outcomes the step is most likely to land on.

3.1 Gumbel-noise conditioning

Parallel drafts can be incoherent because each position of the block is predicted from the committed prefix alone, so the block is a product of per-position marginals. Suppose the committed text ends with *we fly to*. The next position may be split between *New* and *San*, and the position after it between *York* and *Francisco*. Independent sampling can return *New Francisco*. The verifier rejects drafts like this, and the step is wasted.

Ideally each position would condition on the tokens drafted before it, but that would make drafting sequential. Gumbel-noise conditioning, which builds on the Gumbel distillation proposed by Zhang et al. (2026), keeps drafting parallel by front-loading the randomness of sampling. Sampling a token from the logits ℓ, the vector of scores the model produces over the vocabulary, is the same as computing $y = \arg\max_k(\ell_k + \xi_k)$, where ξ is a vector of independent $\mathrm{Gumbel}(0, 1)$ draws, one

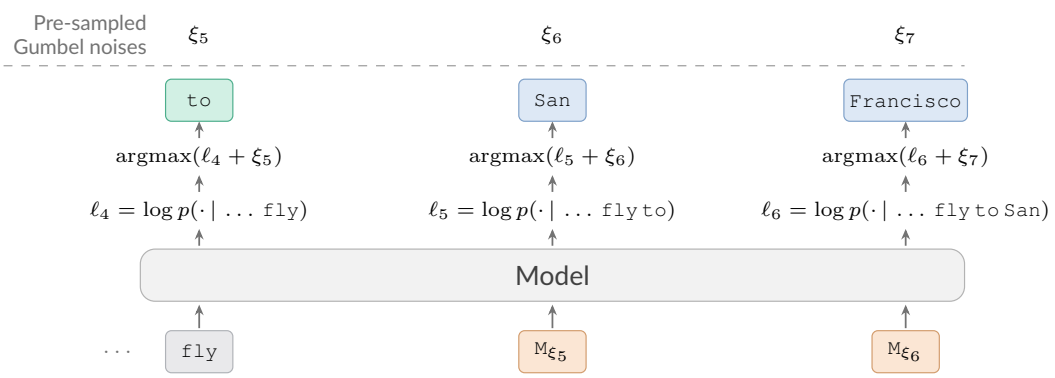


Figure 2: **Gumbel-noise conditioning.** ℓ_i is the log-probability vector the model predicts at position i, and the token at position $i{+}1$ is read off in a second step as $\text{argmax}(\ell_i + \xi_{i+1})$. The noise ξ_i of every position is drawn before generation starts, and its noise token $\texttt{M}_{\xi_i}$ is placed at position i, so that position can recompute its own token and condition on it. Once the noise is drawn, every token is a deterministic function of the logits.

per vocabulary entry. All of the randomness sits in ξ, and once ξ is fixed, the token is a deterministic function of the logits. The distribution of y is unchanged, but ξ can now be sampled for every position before generation starts and shown to all positions. A later position can then predict the token at an earlier one.

Figure 2 illustrates this process. The first block position sees the prefix *fly*, so it can compute the logits for the token that follows. The noise vector ξ_5 of the next position turns those logits into *to*, the draft at that position. The second position, with access to token $\texttt{M}_{\xi_5}$, can repeat that computation to recover *to*, and then apply ξ_6 to the logits to get *San*. Because each noise vector enters the input sequence as a token, a position can attend to the noise tokens of the positions before it. Each position in turn rebuilds the draft before it and predicts from the conditional distribution given those tokens, so the block is drafted from a chain of conditionals. Figure 3 (left) shows the effect on the drafter. At every offset the branching factor e^H (where H denotes the entropy) of Gumbel-conditioned ProSpect stays below that of both marginal-only ProSpect and DFlash.

Zhang et al. (2026) condition on the noise with a learned encoder that maps a noise vector into the embedding space at each draft position. We instead restrict the noise to a fixed set of L vectors drawn once from a seed, and give each one its own token $\texttt{M}_\xi$ in the vocabulary ($L = 10$ in our models). While restricting to a finite set introduces a slight bias, we found that it had no impact on generation quality. The noise at a draft position is then an ordinary token id, and the embedding table replaces the encoder. No separate noise-to-embedding model is trained or served. We also reuse the same noise at verification, so verification is an equality check rather than the probabilistic acceptance of Zhang et al. (2026). The verifier recomputes the position's logits, applies the request's temperature and top-k/top-p filter, adds the tied noise vector, and accepts the drafted token if it is the argmax. The accepted token is then the one the base model would have drawn under that noise.

3.2 Dynamic draft trees

The second problem is the unknown acceptance length. A step commits its drafts up to the first rejection, and how many will be accepted is not known when the query block is built. Drafts beyond the rejection point are discarded, so the query block has to carry fresh drafts that start from the rejection point, otherwise the next step has nothing to verify. What the block holds therefore decides how much of the step can be used. We view the drafts under verification as a tree rooted at the last committed token (i.e. the bonus token), in which each draft token is a node. Laying out the block then comes down to two questions. The first is how many drafts to attempt to verify for the same token position. The second is which nodes receive a *branch*, a mask block appended after the node, which the next step verifies if verification stops at that node.

Submitting several tokens and expanding each of them at every depth makes the query size grow exponentially with the draft length. Quadratic decoding (Samragh et al., 2025), also used by TiDAR

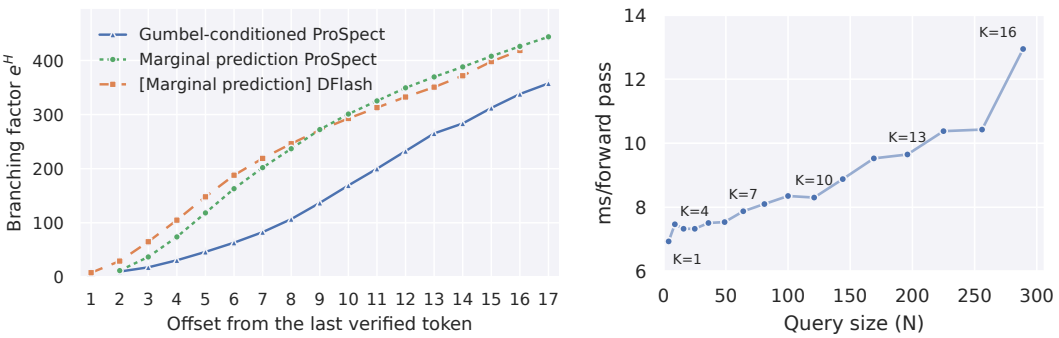


Figure 3: **Left: Draft branching factor by offset.** Both DFlash and marginal prediction ProSpect sample each draft token independently, spreading probability mass across a larger set of continuations. Gumbel-conditioning captures dependencies across draft positions and concentrates probability mass on coherent continuations. **Right: Runtime for the quadratic $\mathbf{N = (K+1)^2}$ verification tree.** Qwen3-8B self-speculation forward pass (H100, vLLM, FA4).

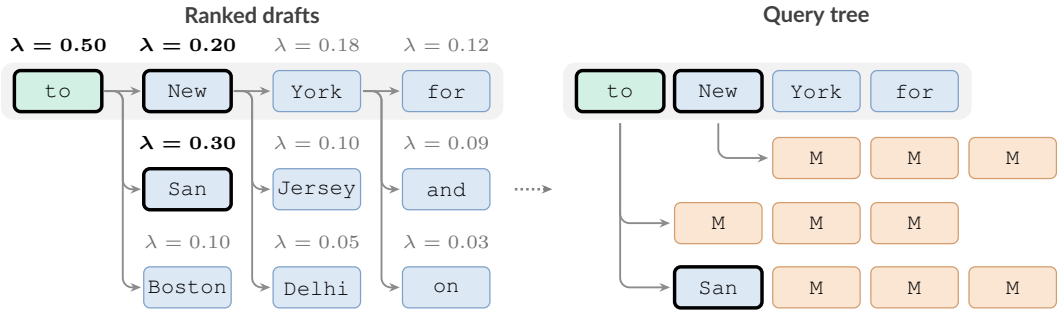


Figure 4: **Building one step's tree** (K=3 backbone drafts, B=3 branches and R=3 candidates per depth) after the prefix `Next week we fly`. Equation (1) scores every candidate landing, and the B highest, outlined, receive a branch. Here `San` outranks the backbone node `New` at its own depth, because a step that accepts `New` usually continues past it rather than stopping there.

(Liu et al., 2026), answers the two questions at the two extremes instead. It submits only the first-ranked token at each depth, so the verified part of the block is a single chain of K drafts, and it attaches a branch to every node of that chain, one per acceptance length. No step is ever left without drafts, and the cost is $K + 1$ branches, or $N = 1 + K + (K + 1)K = (K + 1)^2$ query tokens for K drafts. We refer to this layout as the static draft tree. It spends everything on the second question and nothing on the first.

Two observations indicate that this is a suboptimal division. First, the outcomes are far from equiprobable, and they concentrate at the shallow end. 40% of steps accept no draft token and 20% accept exactly one (ProSpect Qwen3-8B over SpeedBench). A branch at every depth therefore pays the same price for outcomes whose probabilities differ by orders of magnitude. Second, the rejections are near misses. When the draft token is rejected, the token the verifier selects is among the drafter's five most probable candidates 62% of the time. Submitting the first-ranked token alone thus discards cheap acceptances exactly where most steps stop.

We therefore invert the division. Each depth submits its R most probable tokens, and only B nodes receive a branch. The widening is almost free: an alternative is submitted for verification by sitting at the head of a draft branch, so raising R does not enlarge the block and only changes how the B branches are used. The forward pass slows substantially as the query grows (Figure 3, right), so we keep the query small by using a fixed number B of draft branches, each a head followed by K masked draft positions, which leaves at most $N = 1 + K + B\,(K + 1)$ query tokens, linear in the draft length. Unlike the static tree, the layout can leave a step cold: if verification stops at a node that received no branch, the next step has nothing to verify and commits only its bonus token before the tree is rebuilt.

It remains to choose which B nodes receive a branch. Verification walks the tree from the root, descending to the child that matches the verifier's next token until no child matches. We call the node where it stops the step's *landing*. A step is *cold* exactly when its landing has no branch, so we place the B branches at the most likely landings to minimize the chance of a cold step. We compute the landing probability of each node from the drafter's confidences.

We index the candidates by depth. Depth 0 is the root t_0, the token committed at the end of the previous step. At each depth $d \in \{1, \dots, K\}$ the drafter has a distribution q_d, read before temperature is applied, and a drafted token t_d. The chain $t_1, \dots, t_K$ is the backbone, and $p_d = q_d(t_d)$ is the drafter's confidence in it. The alternatives C_d at depth d are the $R-1$ most probable tokens under q_d other than t_d. Treating the acceptances at different depths as independent, with p_d as the probability that t_d is accepted, $r_d = \prod_{i=1}^{d} p_i$ (with $r_0 = 1$) is the probability that the first d backbone tokens are all accepted, and the landing probability of a candidate at depth d is

$$\lambda_{d,t_d} = r_d\,(1 - p_{d+1}), \qquad \lambda_{d,c} = r_{d-1}\,q_d(c) \quad (c \in C_d), \tag{1}$$

where $p_{K+1} = 0$, so that λ_{K,t_K} is the probability that every draft is accepted. The B branches go to the highest scores in a single ranking over all $(K+1) + K(R-1)$ candidates, so that backbone tokens and alternatives compete against one another. A cold step needs no separate treatment, since all draft confidences are then zero, so $\lambda_{0,t_0} = 1$ dominates and the root takes a branch. The ranking is accurate enough to act on: the depth verification reaches is among the five most likely landings about 95% of the time.

We refer to this layout as the dynamic draft tree. Figure 4 shows one step with $K = 3$, $R = 3$ and $B = 3$. The root holds the bonus token *to*, the backbone is *New York for*, and each backbone depth carries two alternatives. Of the ten candidates, the three nodes with highest landing probability λ take the branches: the root at 0.50, the alternative *San* at 0.30, and the backbone token *New* at 0.20. That *San* outranks *New* at the same depth is the ranking working as intended. A step that accepts *New* usually continues past it rather than stopping on it, while *San* is often where the step stops whenever *New* is wrong and *San* is right.

The B winning candidates determine the query block, drawn on the right of Figure 4. Because each branch continues from a different outcome, and at most one of them will be used, no branch may see another. Each attends to the committed prefix and to the backbone tokens on the path above it, and to nothing else (Figure 5, right). An alternative changes only which token sits at a branch head, never which keys any position attends to, so the pattern is fixed by K and B alone and the FlashAttention cascade applies unchanged. A branch carries at most one verification token, which makes the layout a restricted form of an EAGLE-style speculation tree (Li et al., 2025) designed for parallel drafting. Building the tree is a scan, a top-k and a scatter into the query block, over tensors the drafting pass has already produced.

3.3 Engineering details

Training follows the recipe of I-DLM (Yu et al., 2026) and SBD (Gat et al., 2025). Each training sequence is duplicated, with the second copy's tokens replaced by mask tokens at the same position indices, and a single forward pass sees both. A masked position attends to the unmasked prefix before its block and causally to the masked positions before it within the block (Appendix B). Each masked position is trained with next-token cross-entropy on the label of its unmasked counterpart, so drafting is the base model's own next-token problem under an incomplete context. Unmasked positions use a distillation loss against the frozen base model. We train only LoRA adapters and the mask-token embeddings rather than fine-tuning the whole model.

Gumbel-noise conditioning adds one step: each masked position must be given the noise token that actually draws the token it stands for. Noise vector ξ is *consistent* at position p when $\operatorname{argmax}(\ell_{p-1} + \xi_p)$ is the ground-truth token at p, ℓ_{p-1} being the log-probabilities of a frozen base-model pass under the temperature and top-k/top-p the model will be served with. We place a uniformly chosen consistent vector $\mathtt{M}_\xi$ at each masked position. If none of the L noise vector is consistent, the position drops out of the loss with the rest of its block.

At decode time we run attention as a cascade of two FlashAttention calls, an ordinary causal call over the committed prefix and a small masked call within the query block, rather than the slower FlexAttention used by TiDAR and SBD (Appendix B).

4 Related work

Standalone drafters. Speculative decoding usually takes its drafts from a separate draft model. EAGLE (Li et al., 2024; 2025) drafts autoregressively from the target's features, one token per drafter pass, and verifies a tree of candidates. DFlash (Chen et al., 2026) drafts a block in one pass with a block-diffusion drafter, and DSpark (Cheng et al., 2026b) adds a sequential head so that later positions can condition on earlier ones. All three need a second model beside the target, whereas in ProSpect the target proposes its own candidates.

Multi-token prediction. In multi-token prediction (MTP) the target model drafts several future tokens for itself, and verification against its own next-token predictions keeps the output unchanged. Most MTP methods add extra output heads that must be trained and served (Stern et al., 2018; Cai et al., 2024; Gloeckle et al., 2024; Cai et al., 2025). Others append mask tokens to the input and let the model's own weights fill them in. Kirchenbauer et al. (2026) self-distill the model into one that emits several tokens per pass without a verifier, trading exactness for speed. Concurrently, Uno (Sahoo et al., 2026) drafts a block through a gated LoRA pathway and verifies it against the frozen base weights in a separate pass.

Self-speculation. Among mask-token MTP methods, self-speculation drafts and verifies in the same forward pass. Samragh et al. (2025) and TiDAR (Liu et al., 2026) fill the next block with a masked-diffusion objective while verifying the previous one, and append one draft branch per acceptance count, so the query grows quadratically in the draft length. I-DLM (Yu et al., 2026) makes draft positions attend causally within their block, so each draft is the base model's own next-token problem under a truncated context. Nemotron-Labs-Diffusion (Fu et al., 2026) trains one model for autoregressive, diffusion and self-speculative decoding. All of them draft each position from its own marginal and hedge with a layout fixed in advance. Our Gumbel-noise conditioning and dynamic draft trees address both limitations.

Diffusion language models. Masked and block diffusion LMs (Sahoo et al., 2024; Arriola et al., 2025; Nie et al., 2025) fill several positions per pass by sampling from per-position marginals, without verification. SDAR (Cheng et al., 2026a), DiffusionGemma (DiffusionGemma Team et al., 2026) and the diffusion mode of Nemotron-Labs-Diffusion convert pretrained autoregressive models into blockwise diffusion models, and Set Block Decoding (Gat et al., 2025) adds parallel sampling of arbitrary future positions to a next-token model. Without verification the speedup costs generation quality, by an amount set by a denoising schedule chosen in advance. ProSpect verifies every token it emits, so its output matches the base model's.

5 Experiments

We measure generation speed on SpeedBench, accuracy on eight reasoning benchmarks, and ablate Gumbel-conditioning and dynamic draft trees.

5.1 Setup

Models and data. We adapt Qwen3-8B (Yang et al., 2025) and Gemma-4-31B (Gemma Team et al., 2026) with the LoRA recipe above. We take prompts from Nemotron-Post-Training-Dataset-v2, over its math, code, STEM, chat and multilingual splits, plus CodeAlpaca-20K for additional code coverage and the science arms of Nemotron-SFT-Science-v2. Only the prompts are used: the responses are generated synthetically by the model being adapted, so each model is trained on its own outputs and the adapter learns to draft the distribution it will later have to verify against.

SpeedBench. Throughput is measured on SpeedBench (Abramovich et al., 2026) over its qualitative split, at concurrency one, on a single H100 with FlashAttention-4, serving through vLLM. Baselines are DFlash (Chen et al., 2026), DSpark (Cheng et al., 2026b), I-DLM (Yu et al., 2026) and autoregressive decoding. We also run Nemotron-Labs-Diffusion-8B (Fu et al., 2026), a natively trained diffusion model of the same size, in each of the two decoding modes it ships with. No quantization is applied, and every model runs in bfloat16.

Table 1: SpeedBench results at batch size 1. The reported numbers are averages across the 11 SpeedBench categories. All models are served with vLLM and FlashAttention-4, except I-DLM and Nemotron-Labs-Diffusion, which run on their own SGLang forks with flashinfer. † marks methods whose adapter is also active at verification, so their generations are not bit-identical to the base model. § the DFlash and DSpark drafters released for Qwen3-8B are not trained on thinking-mode data. The temperature, top-p, top-k settings are the official model recommendations. Further generation speed benchmarks are available in Appendix C

		Thinking disabled				Thinking enabled			
Method	Config	tok/step	ms/step	tok/sec	Speedup	tok/step	ms/step	tok/sec	Speedup
Qwen3-8B — H100, temp. 0.7 / top-p 0.8 / top-k 20									
Base model	—	1.000	6.8	146.4	1.00×	1.000	7.0	143.5	1.00×
DFlash (Chen et al., 2026)	K=15	3.591	8.9	397.0	2.71×	2.738§	9.1	302.0§	2.10×
DSpark (Cheng et al., 2026b)	K=7	3.787	9.1	442.1	3.02×	3.301§	9.0	392.3§	2.73×
I-DLM ISD† (Yu et al., 2026)	N=4	2.346	8.8	275.3	1.88×	2.361	8.9	278.8	1.94×
[Ours] Gated ProSpect	K=15, B=7, R=4	3.957	11.4	363.5	2.48×	3.981	11.1	402.8	2.81×
[Ours] ProSpect†	K=15, B=7, R=4	**4.134**	8.2	**567.7**	**3.88**×	**4.073**	8.1	**601.7**	**4.19**×
Nemotron-Labs-Diffusion-8B (Fu et al., 2026) — H100, argmax sampling									
Diffusion	b=32	3.42	17.3	188.5	—	5.00	21.6	226.4	—
Linear self-speculation	b=32	3.73	13.0	281.4	—	4.78	16.1	295.2	—
Gemma-4-31B — H100, temp. 1.0 / top-p 0.95 / top-k 64									
Base model	—	1.000	25.1	39.8	1.00×	1.000	25.0	40.0	1.00×
DFlash (Chen et al., 2026)	K=15	3.569	29.5	109.3	2.75×	**4.198**	28.0	139.6	3.49×
Gemma-4 MTP (Gemma Team et al., 2026)	K=6	**3.826**	30.2	121.1	3.04×	4.195	29.4	141.8	3.54×
[Ours] ProSpect†	K=8, B=6, R=4	3.599	28.0	**122.2**	**3.07**×	4.102	27.0	**151.6**	**3.79**×

Ideally, ProSpect would be trained by the model developer as part of post-training, so that the released model is itself the drafter and there is no separate base model to match. The post-training data of Qwen3 and Gemma-4 are proprietary so we report a gated variant instead. Gated ProSpect (Samragh et al., 2025; Yu et al., 2026) keeps the adapter unmerged and applies it only at draft positions. Verification then reads the original weights, so the generation is bit-identical to base autoregressive decoding. The price is extra compute at inference time.

Every method is filtered for repetition before aggregation. A turn is dropped, together with the rest of its request, when some 20-gram occurs at least 50 times in it. Repetition loops are accepted at a near-perfect rate and would inflate both metrics. Filtering affects the Nemotron rows most and Qwen3-8B to a smaller degree. The unfiltered numbers and a per-category breakdown are given in Appendix E. We retain the math and coding benchmarks of Chen et al. (2026) in Appendix C for comparability. Acceptance rates there are all higher than on SpeedBench, which covers general LLM use more broadly.

ProSpect is the fastest method in Table 1 on both models and generation modes. On Qwen3-8B, ProSpect is 28% faster than DSpark and 43% faster than DFlash with thinking disabled. The two factors behind throughput, acceptance length and forward time, are the best among the baselines. A separate drafter pays a second forward pass over a separate set of weights every step, whereas the extra query tokens of ProSpect are only arithmetic inside a pass that is memory-bandwidth bound. On Gemma-4-31B, the margin over Gemma-4's sequential MTP heads is entirely step cost. Where the base distribution must be preserved exactly, gated ProSpect keeps the adapter off the verification path and is bit-identical to base decoding with 2.48× speedup.

Reasoning benchmarks. Table 2 compares ProSpect against base autoregressive decoding and against the I-DLM checkpoints (Yu et al., 2026), the closest published diffusion-adapted conversions of the same weights. Accuracy is measured on eight reasoning benchmarks over five runs per model with independent engine seeds. Because the verifier reads the adapted weights, ProSpect is not bit-identical to the base model.

Measured against the larger of the two standard deviations across seeds, ProSpect stays within one standard deviation of base decoding on five of the eight benchmarks for both Qwen3-8B and Gemma-4-31B. The average accuracy drop is minimal: 1.26 percentage points lower than base decoding on Qwen3-8B and 0.32 points lower on Gemma-4-31B.

Table 2: Accuracy (%) on eight reasoning benchmarks. Each cell is the mean and sample std over five runs with independent engine seeds. Sampling is each model's own recommendation: temperature 0.7 / top-p 0.8 / top-k 20 for the Qwen3-derived rows, 1.0 / 0.95 / 64 for Gemma-4-31B.

Model	ARC-C	MMLU-CoT	GPQA-D	GPQA-M	GSM8K	MATH500	HumanEval	MBPP	Avg
Qwen3-8B									
Base model (AR)	95.64 ± 0.29	82.69 ± 0.13	58.18 ± 1.61	56.88 ± 1.40	94.83 ± 0.44	96.56 ± 0.22	91.59 ± 0.67	88.76 ± 0.64	83.14
I-DLM-8B (Yu et al., 2026)	95.24 ± 0.44	81.89 ± 0.12	56.26 ± 2.10	52.54 ± 1.10	94.40 ± 0.30	95.60 ± 0.86	90.49 ± 1.40	85.24 ± 0.55	81.46
[Ours] ProSpect	95.35 ± 0.36	81.74 ± 0.14	56.87 ± 2.46	53.12 ± 1.00	94.97 ± 0.38	96.00 ± 0.66	90.98 ± 1.64	86.00 ± 0.68	81.88
Gemma-4-31B									
Base model (AR)	98.11 ± 0.15	90.81 ± 0.05	74.95 ± 1.32	72.10 ± 1.29	94.78 ± 0.10	91.44 ± 0.91	97.80 ± 0.70	88.44 ± 0.33	88.55
[Ours] ProSpect	97.96 ± 0.15	90.31 ± 0.16	74.95 ± 1.94	70.80 ± 1.24	94.33 ± 0.44	91.72 ± 0.73	97.56 ± 0.61	88.20 ± 0.45	88.23
Nemotron-Labs-Diffusion-8B (quoted from Fu et al. (2026))									
Block-diffusion decoding	—	78.71	43.94	—	93.03	85.80	78.66	83.86	—
Linear self-speculation	—	79.01	40.40	—	93.78	87.60	81.71	84.92	—

Table 3: Draft trees and Gumbel-noise conditioning on Qwen3-8B, SpeedBench at batch size 1. Δ tok/sec is throughput relative to the full ProSpect configuration.

			Thinking disabled					Thinking enabled				
Tree	Drafter	N	tok/step	ms/step	tok/sec	Speedup	Δ tok/sec	tok/step	ms/step	tok/sec	Speedup	Δ tok/sec
Static, K=6	Marginals only	49	2.844	7.8	395.4	2.70×	−30.4%	2.823	7.8	411.1	2.86×	−31.7%
	Gumbel-conditioned	49	3.166	7.8	445.2	3.04×	−21.6%	3.208	7.8	477.1	3.32×	−20.7%
Dynamic, K=7, B=7, R=4	Marginals only	64	3.264	8.1	450.9	3.08×	−20.6%	3.290	8.1	468.9	3.27×	−22.1%
	Gumbel-conditioned	64	3.659	8.0	552.7	3.78×	−2.6%	3.741	8.0	535.5	3.73×	−11.0%
Dynamic, K=15, B=7, R=4	Marginals only	128	3.490	8.3	470.8	3.22×	−17.1%	3.440	8.5	473.3	3.30×	−21.3%
	Gumbel-conditioned	128	4.134	8.2	567.7	3.88×	0.0%	4.073	8.1	601.7	4.19×	0.0%

5.2 Ablations

Table 3 isolates our two contributions using Qwen3-8B. It crosses the static and dynamic draft layouts, with two adapters trained on the same data and recipe, one conditioned on Gumbel noise and one drafting from the per-position marginals alone. Appendix D gives the full sweep over K, B and R, from which the dynamic shapes here are the best at each row budget.

Both contributions are needed. Gumbel-noise conditioning raises the acceptance length in all layouts. Since the dynamic tree makes long drafts affordable, the advantage of using Gumbel conditioning grows. Measured on throughput, dropping conditioning from the full configuration costs 17% with thinking disabled and 21% with thinking enabled, while returning to the best static tree costs 22% and 21% respectively. Dropping both costs around 30%.

6 Limitations

Like other speculative decoding methods, ProSpect does not scale well beyond batch size one. Every request contributes its own N-token query block, so query length grows with the batch and the forward pass turns compute-bound well before a serving batch is reached. Drafting relies on idle arithmetic, which exists only at low concurrency.

7 Conclusion

We built a self-speculative decoder in which one model drafts and verifies in a single forward pass, with no auxiliary drafter to train or serve. Gumbel-noise conditioning lets parallel drafts approximate the joint conditional, using a small set of vocabulary tokens instead of a noise encoder. A dynamic draft tree verifies several candidates where the draft is least confident, in a layout whose query length stays linear in the draft length. On SpeedBench, ProSpect is the fastest method we measure on both Qwen3-8B and Gemma-4-31B in both thinking modes, and it stays close to base-model accuracy on the reasoning benchmarks.

AI USE STATEMENT

We used a generative AI tool in two roles in this work: as a *coding assistant* during implementation and experiment tooling, and as a *writing assistant* for drafting and revising this manuscript. All AI-assisted work was reviewed and verified by the authors. Generated code was read and tested before use, and the numbers and claims reported here were checked against the logged experimental results. The authors take responsibility for the final content of this work, including all text, claims and artifacts produced with the aid of generative AI.

REFERENCES

Talor Abramovich, Maor Ashkenazi, Izzy Putterman, Benjamin Chislett, Tiyasa Mitra, Bita Darvish Rouhani, Ran Zilberstein, and Yonatan Geifman. SPEED-Bench: A unified and diverse benchmark for speculative decoding. In *Forty-third International Conference on Machine Learning*, 2026. doi: 10.48550/arXiv.2604.09557.

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. In *International Conference on Learning Representations*, volume 2025, pp. 50726–50753, 2025. doi: 10.48550/arXiv.2503.09573.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2401.10774.

Yuxuan Cai, Xiaozhuan Liang, Xinghua Wang, Jin Ma, Haijin Liang, Jinwen Luo, Xinyu Zuo, Lisheng Duan, Yuyang Yin, and Xi Chen. FastMTP: Accelerating LLM inference with enhanced multi-token prediction. *arXiv.org*, 2025. doi: 10.48550/arXiv.2509.18362.

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

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Forty-third International Conference on Machine Learning*, 2026. doi: 10.48550/arXiv.2602.06036.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, et al. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 22058–22075, 2026a. doi: 10.18653/v1/2026.findings-acl.1110.

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, et al. DSpark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv.org*, 2026b. doi: 10.48550/arXiv.2607.05147.

DiffusionGemma Team, Adrien Ali Taïga, James Assiene, Daniele Calandriello, Rahma Chaabouni, João Gante, Tamara von Glehn, Nate Keating, Chris Knutsen, Martin Kukla, et al. DiffusionGemma technical report. *arXiv*, 2026.

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, et al. Nemotron-Labs-Diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. *arXiv.org*, 2026. doi: 10.48550/arXiv.2607.05722.

Itai Gat, Heli Ben-Hamu, Marton Havasi, Daniel Haziza, Jeremy Reizenstein, Gabriel Synnaeve, David Lopez-Paz, Brian Karrer, and Yaron Lipman. Set block decoding is a language model inference accelerator. *arXiv.org*, 2025. doi: 10.48550/arXiv.2509.04185.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv.org*, 2026. doi: 10.48550/arXiv.2607.02770.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2404.19737.

John Kirchenbauer, Abhimanyu Hans, Brian Bartoldson, Micah Goldblum, Ashwinee Panda, and Tom Goldstein. Multi-token prediction via self-distillation. *AdaptFM: Resource-Adaptive Foundation Model Inference*, 2026. doi: 10.48550/arXiv.2602.06019.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International conference on machine learning*, pp. 19274–19286. PMLR, 2023. doi: 10.48550/arXiv.2211.17192.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948. PMLR, 21–27 Jul 2024. doi: 10.48550/arXiv.2401.15077.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. *Advances in Neural Information Processing Systems 38*, 38:151568–151587, 2025. doi: 10.52202/085713-4562.

Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. TiDAR: Think in diffusion, talk in autoregression. *Proceedings of Machine Learning and Systems*, 8:748–762, 2026. doi: 10.48550/arXiv.2511.08923.

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. *Advances in Neural Information Processing Systems 38*, 38:56354–56392, 2025. doi: 10.52202/085713-1689.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems 37*, 37:130136–130184, 2024. doi: 10.52202/079017-4135.

Subham Sekhar Sahoo, Lingjie Chen, Khiem Pham, Jonathan Geuter, Chaitanya Dwivedi, Varad Pimpalkhute, Yash Akhauri, Alexander Moreno, Mikhail Yurochkin, Zhenting Wang, et al. Unlocking lossless speedups in LLMs via discrete diffusion. *arXiv preprint arXiv:2609.04010*, 2026.

Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and Mehrdad Farajtabar. Your LLM knows the future: Uncovering its multi-token prediction potential. *arXiv.org*, 2025. doi: 10.48550/arXiv.2507.11851.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Neural Information Processing Systems*, volume 31, 2018.

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

Yifan Yu, Yuqing Jian, Junxiong Wang, Zhongzhu Zhou, Donglin Zhuang, Xinyu Fang, Sri Yanamandra, Xiaoxia Wu, Qingyang Wu, Shuaiwen Leon Song, et al. Introspective diffusion language models. *Conference on Language Modeling*, 2026. doi: 10.48550/arXiv.2604.11035.

Chi Zhang, Xixi Hu, Bo Liu, and Qiang Liu. Gumbel distillation for parallel text generation. In *The Fourteenth International Conference on Learning Representations*, 2026. doi: 10.48550/arXiv.2603.22216.

A TRAINING HYPERPARAMETERS

Table 4 gives the settings behind the adapted checkpoints. Training uses AdamW under a cosine schedule with linear warmup, in bfloat16 with FSDP and activation checkpointing. Only the LoRA adapters and the mask-token embedding rows receive gradients.

Table 4: Training hyperparameters. Optimizer steps are the checkpoint the reported results were measured at, and tokens seen counts supervised positions across all ranks up to it. The gated run did not log it.

	Qwen3-8B	Qwen3-8B (gated)	Gemma-4-31B
Adapter			
LoRA rank / α	128 / 256	128 / 256	64 / 128
LoRA dropout	0.05	0.05	0.05
LoRA modules	q, k, v, output projections and all MLP layers; LM head excluded	same	same
Adapter at inference	merged	draft rows only	merged
Optimization			
Learning rate	3×10^{-4}	3×10^{-4}	1×10^{-4}
Warmup ratio	0.03	0.03	0.03
Weight decay	0.01	0.01	0.01
Gradient-norm clip	1.0	1.0	1.0
Batch size	64	64	8
Optimizer steps	26,775	40,000	12,500
Tokens seen	4.3×10^{10}	—	2.6×10^{9}
Self-speculation			
Draft block size	16	16	16
Max sequence length	32,768	24,576	32,768
Gumbel noise tokens L	10	10	10
Loss			
Verify-half CE/KL blend α	0.5	—	0.5
Draft-half objective	CE only	CE/KL, α=0.5	CE only
Draft-half weight	0.2	1	0.2

B ATTENTION MASKS

Both copies of a training sequence lie over the same grid of fixed-size blocks. An unmasked position attends causally to the unmasked positions before it and never to a masked one. A masked position attends to the unmasked positions of strictly earlier blocks, which is the committed prefix at decode time, and causally to the masked positions before it in its own block (Figure 5, left). At decode time, once the query block carries a draft tree, separate branches must also not see one another (Figure 5, right).

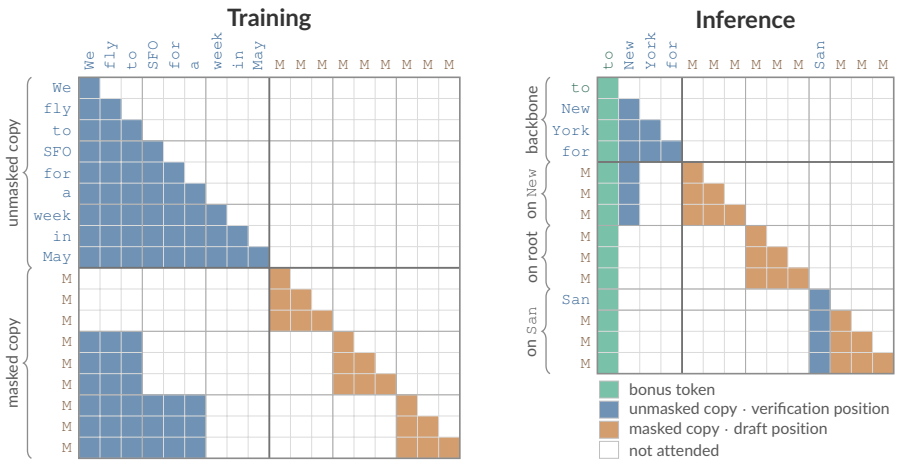


Figure 5: *Left:* training over the doubled sequence. A mask block attends causally within itself and to every unmasked token preceding the block. *Right:* the decode-time query block of Figure 4 (K=3, B=3), in which no branch may see another. Only attention within the block is drawn. Every position additionally attends to the whole committed prefix in the KV cache, which is the cascade's ordinary causal call.

TiDAR (Liu et al., 2026) and SBD (Gat et al., 2025) express these masks with FlexAttention, which builds the mask from a predicate but is much slower than FlashAttention at long context, where attention dominates the forward pass. Every position in the query block attends to the whole committed prefix, so attention against the KV cache needs no special mask. We therefore split attention into two FlashAttention calls. The first is causal over the prefix and carries all the work that scales with context. The second handles attention within the query block, where the layout's mask applies, and merges its output with the first.

C MATH AND CODING BENCHMARKS

SpeedBench is a general-assistant suite. To check that the ranking in Table 1 is not specific to it, we repeat the Qwen3-8B and Gemma-4-31B measurements on a second suite weighted toward math and coding: 510 prompts and 590 turns drawn from MATH500, GSM8K, AIME25, MBPP, HumanEval, LiveCodeBench and MT-Bench. The setup is identical to Table 1: one H100, concurrency one, each model's sampling settings, and the same repetition filter.

Table 5: Math and coding benchmarks (MATH500, GSM8K, AIME25, MBPP, HumanEval, LiveCodeBench and MT-Bench) at batch size 1, measured exactly as in Table 1. † marks methods whose adapter is also active at verification. ‡ I-DLM ISD runs on I-DLM's own SGLang fork, which forces `flashinfer` rather than FlashAttention-4.

		Thinking disabled				Thinking enabled			
Method	Config	tok/step	ms/step	tok/sec	Speedup	tok/step	ms/step	tok/sec	Speedup
Qwen3-8B — H100, temp. 0.7 / top-p 0.8 / top-k 20									
Base model	—	1.000	6.8	146.6	1.00×	1.000	7.0	142.8	1.00×
DFlash (Chen et al., 2026)	K=15	**6.712**	8.9	605.4	4.13×	3.316	9.0	316.8	2.22×
DSpark (Cheng et al., 2026b)	K=7	5.662	9.0	569.1	3.88×	3.761	9.1	411.3	2.88×
I-DLM ISD†‡ (Yu et al., 2026)	N=4	2.973	8.9	319.4	2.18×	2.672	9.0	291.0	2.04×
[Ours] ProSpect, static tree†	K=10	5.439	8.0	586.9	4.00×	4.369	8.5	540.3	3.78×
[Ours] ProSpect†	K=10, B=10, R=4	6.040	8.1	**731.7**	**4.99**×	**5.112**	8.5	**628.1**	**4.40**×
Gemma-4-31B — H100, temp. 1.0 / top-p 0.95 / top-k 64									
Base model	—	1.000	24.7	40.4	1.00×	1.000	24.9	40.2	1.00×
DFlash (Chen et al., 2026)	K=15	**6.526**	29.5	157.5	3.90×	**6.423**	28.9	183.0	4.55×
Gemma4 MTP (Gemma Team et al., 2026)	K=6	5.207	29.6	155.9	3.86×	5.145	29.5	165.6	4.12×
[Ours] ProSpect, static tree†	K=7	4.731	28.5	140.8	3.48×	4.774	28.5	156.1	3.89×
[Ours] ProSpect†	K=13, B=8, R=4	6.343	30.8	**163.0**	**4.03**×	6.346	30.8	**188.5**	**4.69**×

D THE SPEEDBENCH TREE SWEEP

Table 6 is the sweep the two ProSpect rows of Table 1 are drawn from: the static $(K{+}1)^2$ tree at $K \in \{6, 7, 8, 10\}$ and twelve dynamic shapes, in both thinking modes, on the same checkpoint, engine and day.

Table 6: The full SpeedBench tree sweep on Qwen3-8B, on the same checkpoint and engine as Table 1 and with the same repetition filter, so the two are directly comparable. N is the query rows per decode step and is the cost axis: the static tree spends $(K{+}1)^2$ rows on a K-token draft, the dynamic tree $1 + K + B(K{+}1)$. Speedups are against the no-speculation baseline of the same mode (146.4 and 143.5 tokens/second). The bold row is the dynamic cell Table 1 reports; its static counterpart is $K{=}10$, and Table 3 extracts the static-versus-dynamic comparison from this sweep.

		Thinking disabled				Thinking enabled			
Tree	N	tok/step	ms/step	tok/sec	Speedup	tok/step	ms/step	tok/sec	Speedup
Static, $K{=}6$	49	3.166	7.8	445.2	3.04×	3.208	7.8	477.1	3.32×
Static, $K{=}7$	64	3.277	7.8	437.5	2.99×	3.306	7.8	489.4	3.41×
Static, $K{=}8$	81	3.403	8.2	473.4	3.23×	3.373	8.1	482.2	3.36×
Static, $K{=}10$	121	3.518	8.1	471.6	3.22×	3.476	8.3	493.3	3.44×
Dynamic, $K{=}8$, $B{=}6$, $R{=}1$	63	3.369	7.8	480.5	3.28×	3.342	7.9	495.7	3.45×
Dynamic, $K{=}8$, $B{=}6$, $R{=}2$	63	3.604	7.8	507.4	3.47×	3.621	7.9	532.7	3.71×
Dynamic, $K{=}8$, $B{=}6$, $R{=}4$	63	3.673	7.8	500.2	3.42×	3.727	7.9	549.1	3.83×
Dynamic, $K{=}7$, $B{=}7$, $R{=}4$	64	3.659	8.0	552.7	3.78×	3.741	8.0	535.5	3.73×
Dynamic, $K{=}9$, $B{=}5$, $R{=}4$	60	3.650	7.9	511.8	3.50×	3.627	7.8	539.7	3.76×
Dynamic, $K{=}15$, $B{=}3$, $R{=}3$	64	3.527	7.8	486.1	3.32×	3.403	7.9	496.6	3.46×
Dynamic, $K{=}13$, $B{=}8$, $R{=}1$	126	3.632	8.1	474.1	3.24×	3.541	8.4	501.3	3.49×
Dynamic, $K{=}13$, $B{=}8$, $R{=}2$	126	3.993	8.2	543.3	3.71×	3.967	8.4	560.0	3.90×
Dynamic, $K{=}13$, $B{=}8$, $R{=}4$	126	4.077	8.1	540.1	3.69×	4.066	8.4	566.7	3.95×
Dynamic, $K{=}15$, $B{=}7$, $R{=}4$	**128**	**4.134**	**8.2**	**567.7**	**3.88×**	**4.073**	**8.1**	**601.7**	**4.19×**
Dynamic, $K{=}10$, $B{=}10$, $R{=}4$	121	4.090	8.1	535.1	3.66×	4.142	8.4	574.9	4.01×
Dynamic, $K{=}17$, $B{=}6$, $R{=}4$	126	4.049	8.2	566.0	3.87×	3.911	8.2	557.7	3.89×

E PER-CATEGORY BREAKDOWN

The tables below break the results of Table 1 and Table 5 down by prompt category on SpeedBench and by source benchmark on the math-and-coding suite. Each metric is reported both unfiltered and filtered. The filter is the degeneration filter used throughout the paper. It drops a turn, together with every other turn of its request, when some word 20-gram occurs at least 50 times in it.

The SpeedBench tables also carry the two Nemotron-Labs-Diffusion-8B rows of Table 1, FastDiffuser (*Diffusion*) and LinearSpec with the released LoRA drafter (*Linear SS*), both at block 32. Their setup differs from that of the Qwen3-8B columns. Both modes decode greedily, because neither algorithm reads the request's sampling parameters, while the Qwen3-8B methods sample at the settings of Table 1. They run on the Nemotron SGLang fork with flashinfer rather than on vLLM.

These runs use a 12,288-token context window and a 10,240-token generation cap, against a 32,768-token cap for Qwen3-8B.

E.1 SPEEDBENCH, THINKING ENABLED

Table 7: Accepted tokens per forward pass, SpeedBench, thinking enabled. The *Unfiltered* block is over all turns, the *Filtered* block excludes the turns the repetition filter drops (Table 10). For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}15$, $B{=}7$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}8$, $B{=}6$, $R{=}4$ tree. *Diffusion* is FastDiffuser and *Linear SS* LinearSpec, both at block 32.

		Qwen3-8B						Nemotron-Diffusion-8B		Gemma-4-31B				
Category	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Diffusion	Linear SS	Base	DFlash	MTP	Static	Dynamic
Unfiltered														
Coding	89	1.000	2.672	3.377	3.737	4.816	5.894	5.71	7.63	1.000	5.962	4.973	4.423	4.942
Humanities	88	1.000	2.424	2.685	3.316	4.216	4.632	10.10	11.98	1.000	3.524	3.760	3.248	3.776
QA	80	1.000	2.362	2.526	3.139	3.419	4.391	9.27	11.39	1.000	3.519	3.740	2.990	3.432
RAG	105	1.000	2.443	2.908	3.491	3.467	3.928	6.98	4.84	1.000	5.092	4.739	4.348	4.854
STEM	86	1.000	2.520	2.772	3.507	4.671	5.154	8.81	11.31	1.000	3.975	4.168	3.783	4.133
Multilingual	80	1.000	2.636	3.131	3.514	4.028	4.780	9.53	10.54	1.000	4.691	4.942	4.355	4.876
Roleplay	158	1.000	1.879	2.039	2.456	2.157	2.678	7.15	8.55	1.000	2.849	3.161	2.556	2.949
Math	98	1.000	2.637	2.894	3.728	4.861	5.714	8.28	11.60	1.000	5.232	4.891	4.450	5.028
Reasoning	190	1.000	2.448	2.994	3.503	3.931	4.685	7.37	10.45	1.000	4.623	4.319	3.762	4.248
Writing	84	1.000	2.130	2.466	3.029	2.642	3.095	5.61	6.81	1.000	2.975	3.348	2.712	3.150
Summarization	80	1.000	2.207	2.368	2.973	2.725	3.113	7.95	7.72	1.000	3.736	4.099	3.380	3.854
Filtered														
Coding	89	1.000	2.600	3.378	3.724	4.390	5.272	3.35	4.49	1.000	5.962	4.973	4.413	4.942
Humanities	88	1.000	2.377	2.685	3.316	3.661	4.186	8.95	6.89	1.000	3.524	3.760	3.213	3.642
QA	80	1.000	2.238	2.526	3.094	2.960	3.438	6.41	4.78	1.000	3.519	3.740	2.990	3.432
RAG	105	1.000	2.443	2.908	3.491	3.467	3.928	4.03	3.17	1.000	5.092	4.739	4.348	4.854
STEM	86	1.000	2.442	2.767	3.487	4.057	4.779	7.11	6.64	1.000	3.975	4.168	3.708	4.133
Multilingual	80	1.000	2.620	3.131	3.514	4.028	4.659	5.77	4.25	1.000	4.691	4.942	4.321	4.876
Roleplay	158	1.000	1.879	2.039	2.456	2.157	2.678	3.39	2.31	1.000	2.849	3.161	2.556	2.949
Math	98	1.000	2.625	2.867	3.728	4.672	5.518	4.96	7.90	1.000	5.232	4.891	4.450	5.028
Reasoning	190	1.000	2.434	2.994	3.503	3.709	4.376	4.56	3.41	1.000	4.623	4.319	3.762	4.248
Writing	84	1.000	2.130	2.466	3.029	2.642	3.095	3.30	3.43	1.000	2.975	3.348	2.712	3.150
Summarization	80	1.000	2.207	2.368	2.973	2.725	3.113	5.64	4.16	1.000	3.736	4.099	3.380	3.854

Table 8: Throughput (tokens/second), SpeedBench, thinking enabled. Pooled $\sum$ tokens$/\sum$ wall within the split; the *Filtered* block is the quotable one. For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}15$, $B{=}7$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}8$, $B{=}6$, $R{=}4$ tree. *Diffusion* is FastDiffuser and *Linear SS* LinearSpec, both at block 32.

		Qwen3-8B						Nemotron-Diffusion-8B		Gemma-4-31B				
Category	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Diffusion	Linear SS	Base	DFlash	MTP	Static	Dynamic
Unfiltered														
Coding	89	142.8	310.7	349.2	418.9	632.3	799.8	210.7	353.3	40.1	183.8	164.3	152.8	175.5
Humanities	88	143.2	298.9	300.1	391.2	634.1	707.6	259.0	405.8	40.0	128.6	134.0	125.9	153.4
QA	80	142.6	343.9	315.5	405.9	679.9	941.9	226.1	421.6	40.2	113.1	120.5	99.5	119.9
RAG	105	143.8	280.6	339.9	398.5	519.0	511.3	200.8	292.8	39.8	165.9	153.7	146.6	170.0
STEM	86	141.5	299.5	292.5	406.2	626.0	702.8	243.4	406.2	40.0	143.1	148.6	141.6	160.6
Multilingual	80	147.1	308.6	330.4	408.3	532.4	686.9	304.1	414.0	40.2	158.0	166.4	159.6	182.1
Roleplay	158	148.2	221.0	242.0	285.1	281.2	429.0	211.3	312.4	40.1	99.3	106.1	90.2	107.5
Math	98	140.0	287.6	283.3	397.7	576.2	698.8	237.8	400.0	40.0	165.7	161.1	151.8	178.6
Reasoning	190	146.0	289.0	353.2	405.8	542.9	683.2	191.5	313.7	40.1	138.8	136.5	120.5	143.1
Writing	84	147.6	244.0	278.8	343.9	329.2	397.5	193.1	321.4	40.1	102.7	112.0	95.2	114.2
Summarization	80	148.2	250.6	269.7	335.4	342.1	398.4	273.1	345.7	40.1	129.7	136.1	116.9	139.4
Filtered														
Coding	89	143.9	290.0	349.5	416.5	549.4	686.5	194.6	307.5	40.1	183.8	164.3	151.4	175.5
Humanities	88	144.1	282.7	300.1	391.2	501.0	601.0	315.6	328.0	40.0	128.6	134.0	124.1	146.2
QA	80	148.7	262.6	315.5	356.0	451.9	536.8	239.6	326.9	40.3	113.1	120.5	99.5	119.9
RAG	105	147.5	280.6	339.9	398.5	519.0	511.3	178.8	237.0	39.8	165.9	153.7	146.6	170.0
STEM	86	141.8	282.2	291.9	402.2	531.8	644.1	264.6	346.7	40.0	143.1	148.6	138.6	160.6
Multilingual	80	147.1	297.5	330.4	408.3	532.4	631.2	261.2	353.4	40.2	158.0	166.4	153.7	182.1
Roleplay	158	148.2	221.0	242.0	285.1	281.2	429.0	209.0	201.4	40.1	99.3	106.1	90.2	107.5
Math	98	140.2	285.2	279.0	397.7	549.0	671.6	215.2	352.3	40.0	165.7	161.1	151.8	178.6
Reasoning	190	147.3	282.7	353.2	405.8	478.9	604.1	206.6	276.4	40.1	138.8	136.5	120.5	143.1
Writing	84	147.6	244.0	278.8	343.9	329.2	397.5	172.3	225.0	40.1	102.7	112.0	95.2	114.2
Summarization	80	148.2	250.6	269.7	335.4	342.1	398.4	232.5	278.4	40.1	129.7	136.1	116.9	139.4

Table 9: Generation length (mean output tokens per turn), SpeedBench, thinking enabled. Every method is prompted with the same turns, so the *Unfiltered* spread across columns is the methods' own verbosity rather than a difference in workload; each *Filtered* value is over that method's own surviving turns. For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}15$, $B{=}7$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}8$, $B{=}6$, $R{=}4$ tree. *Diffusion* is FastDiffuser and *Linear SS* LinearSpec, both at block 32.

		Qwen3-8B						Nemotron-Diffusion-8B		Gemma-4-31B				
Category	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Diffusion	Linear SS	Base	DFlash	MTP	Static	Dynamic
Unfiltered														
Coding	89	6910	7331	6353	6341	8601	8897	6957	5934	2029	2112	1998	2022	1976
Humanities	88	4761	5114	4462	4377	6866	5685	21483	18983	3171	3410	3231	3289	3330
QA	80	2128	3657	1388	1635	3540	4315	9094	9640	791	801	780	653	673
RAG	105	1282	937	984	966	1229	896	3767	1413	443	467	460	425	437
STEM	86	9297	8842	8998	9172	11004	9782	20754	18820	4398	4593	4644	4705	4509
Multilingual	80	2618	3108	2766	2801	2562	2953	4949	5220	1419	1330	1314	1339	1263
Roleplay	158	650	623	658	623	655	879	4237	3006	609	615	597	600	605
Math	98	11096	9293	10693	10177	11044	10874	15733	16707	5084	5204	5077	4944	4954
Reasoning	190	2139	2006	1897	1926	2289	2518	6861	7459	1122	1069	1125	1051	1026
Writing	84	2490	2516	2567	2595	2445	2510	14761	10842	2334	2306	2310	2309	2226
Summarization	80	537	527	527	546	504	489	5947	2434	684	730	713	683	694
Filtered														
Coding	89	6316	5375	6053	6231	6473	6480	2816	2475	2029	2112	1998	1929	1976
Humanities	88	4439	4138	4462	4377	4276	4262	12215	6482	3171	3410	3231	3209	3086
QA	80	1342	1296	1388	1241	1591	1587	2181	1930	671	801	780	653	673
RAG	105	979	937	984	966	1229	896	1300	808	443	467	460	425	437
STEM	86	9021	7058	8432	8894	7812	8363	10084	6424	4398	4593	4644	4574	4509
Multilingual	80	2618	2732	2766	2801	2562	2576	1858	829	1307	1330	1314	1226	1263
Roleplay	158	650	623	658	623	655	879	871	430	609	615	597	600	605
Math	98	10872	9051	10465	10177	10120	9892	2957	5155	5084	5204	5077	4944	4954
Reasoning	190	1954	1843	1897	1926	1818	2028	2331	952	1122	1069	1125	1051	1026
Writing	84	2490	2516	2567	2595	2445	2510	7512	4486	2334	2306	2310	2309	2226
Summarization	80	537	527	527	546	504	489	2967	838	684	730	713	683	694

Table 10: Turns dropped by the repetition filter, SpeedBench, thinking enabled. The filter acts per request, so one flagged turn drops every turn of that question. For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}15$, $B{=}7$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}8$, $B{=}6$, $R{=}4$ tree. *Diffusion* is FastDiffuser and *Linear SS* LinearSpec, both at block 32.

		Qwen3-8B						Nemotron-Diffusion-8B		Gemma-4-31B				
Category	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Diffusion	Linear SS	Base	DFlash	MTP	Static	Dynamic
Coding	89	2	8	1	1	8	9	15	13	0	0	0	1	0
Humanities	88	1	3	0	0	8	5	45	43	0	0	0	1	3
QA	80	2	6	0	1	5	7	19	20	1	0	0	0	0
RAG	105	1	0	0	0	0	0	14	3	0	0	0	0	0
STEM	86	1	6	2	1	11	5	42	42	0	0	0	2	0
Multilingual	80	0	1	0	0	0	1	8	11	1	0	0	1	0
Roleplay	158	0	0	0	0	0	0	23	27	0	0	0	0	0
Math	98	1	1	1	0	4	5	42	41	0	0	0	0	0
Reasoning	190	2	2	0	0	6	9	58	74	0	0	0	0	0
Writing	84	0	0	0	0	0	0	28	20	0	0	0	0	0
Summarization	80	0	0	0	0	0	0	8	4	0	0	0	0	0

E.2 SPEEDBENCH, THINKING DISABLED

Table 11: Accepted tokens per forward pass, SpeedBench, thinking disabled. The *Unfiltered* block is over all turns, the *Filtered* block excludes the turns the repetition filter drops (Table 14). For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}15$, $B{=}7$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}8$, $B{=}6$, $R{=}4$ tree. *Diffusion* is FastDiffuser and *Linear SS* LinearSpec, both at block 32.

		Qwen3-8B						Nemotron-Diffusion-8B		Gemma-4-31B				
Category	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Diffusion	Linear SS	Base	DFlash	MTP	Static	Dynamic
Unfiltered														
Coding	89	1.000	2.757	5.750	5.361	4.754	5.380	5.50	7.15	1.000	6.097	5.256	4.616	5.199
Humanities	88	1.000	2.280	3.217	3.567	3.404	4.540	5.43	8.91	1.000	2.792	3.228	2.565	3.000
QA	80	1.000	2.077	2.479	3.149	2.631	2.962	4.54	5.37	1.000	2.767	3.144	2.430	2.779
RAG	105	1.000	2.549	3.874	4.134	3.656	4.235	2.12	4.16	1.000	4.496	4.565	3.896	4.323
STEM	86	1.000	2.517	3.971	4.021	4.329	5.583	6.03	9.34	1.000	3.423	3.762	3.160	3.625
Multilingual	80	1.000	2.670	3.520	3.458	5.080	5.884	5.34	9.59	1.000	3.583	4.564	3.657	4.176
Roleplay	158	1.000	1.719	2.016	2.402	1.960	2.338	7.85	10.96	1.000	2.062	2.377	1.854	2.224
Math	98	1.000	2.751	5.056	4.905	4.874	6.221	7.37	10.18	1.000	4.537	4.499	3.865	4.353
Reasoning	190	1.000	2.582	4.289	4.374	4.304	4.729	4.66	8.10	1.000	3.991	4.006	3.266	3.749
Writing	84	1.000	2.096	2.748	3.386	2.696	3.236	4.79	5.17	1.000	2.688	3.135	2.496	2.953
Summarization	80	1.000	2.193	2.649	3.187	2.828	3.209	1.95	2.53	1.000	2.837	3.554	2.786	3.292
Filtered														
Coding	89	1.000	2.757	5.750	5.361	4.535	5.261	3.70	4.23	1.000	6.097	5.256	4.616	5.199
Humanities	88	1.000	2.195	3.208	3.480	3.215	3.703	3.55	4.59	1.000	2.792	3.228	2.565	3.000
QA	80	1.000	2.053	2.479	3.149	2.423	2.799	1.84	2.20	1.000	2.744	3.144	2.430	2.779
RAG	105	1.000	2.549	3.843	4.090	3.656	4.235	2.12	3.98	1.000	4.496	4.565	3.896	4.323
STEM	86	1.000	2.349	3.940	3.925	3.650	4.602	4.03	4.16	1.000	3.423	3.762	3.046	3.543
Multilingual	80	1.000	2.638	3.533	3.450	4.944	5.828	4.41	4.04	1.000	3.583	4.564	3.657	4.176
Roleplay	158	1.000	1.719	2.016	2.402	1.960	2.338	3.15	2.05	1.000	2.062	2.377	1.854	2.224
Math	98	1.000	2.709	5.056	4.870	4.729	5.659	4.65	6.50	1.000	4.537	4.499	3.865	4.353
Reasoning	190	1.000	2.566	4.289	4.374	4.159	4.729	3.56	3.32	1.000	3.991	4.006	3.266	3.749
Writing	84	1.000	2.096	2.748	3.386	2.696	3.236	2.88	2.76	1.000	2.688	3.135	2.496	2.953
Summarization	80	1.000	2.193	2.649	3.187	2.828	3.209	1.95	2.53	1.000	2.837	3.554	2.786	3.292

Table 12: Throughput (tokens/second), SpeedBench, thinking disabled. Pooled $\sum$ tokens$/\sum$ wall within the split; the *Filtered* block is the quotable one. For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}15$, $B{=}7$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}8$, $B{=}6$, $R{=}4$ tree. *Diffusion* is FastDiffuser and *Linear SS* LinearSpec, both at block 32.

		Qwen3-8B						Nemotron-Diffusion-8B		Gemma-4-31B				
Category	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Diffusion	Linear SS	Base	DFlash	MTP	Static	Dynamic
Unfiltered														
Coding	89	148.3	308.5	609.6	568.0	820.9	821.2	251.8	398.3	39.9	168.3	159.5	150.3	164.1
Humanities	88	147.8	350.3	396.1	544.6	696.6	1071.1	222.9	363.4	39.9	101.6	115.7	98.4	114.1
QA	80	147.8	310.9	293.3	504.0	672.4	666.6	165.5	337.7	39.9	105.7	103.5	85.9	101.8
RAG	105	145.6	276.6	497.7	562.7	424.4	495.6	132.8	245.9	37.8	121.5	126.8	118.8	134.7
STEM	86	138.5	365.8	481.6	542.6	834.5	1146.4	198.6	404.3	40.0	116.8	131.6	132.7	146.1
Multilingual	80	140.8	358.5	323.6	384.0	844.3	757.5	305.0	418.6	39.9	117.0	149.8	126.8	145.0
Roleplay	158	146.4	201.9	236.5	277.3	257.6	301.2	223.7	362.1	39.5	72.8	82.7	68.6	82.7
Math	98	144.6	350.6	536.7	589.7	785.8	1133.3	269.4	399.0	39.9	142.6	146.0	138.4	150.4
Reasoning	190	147.7	305.3	430.0	463.5	664.7	630.3	126.0	327.8	39.8	118.8	125.5	108.2	126.6
Writing	84	147.5	241.0	303.9	378.6	339.3	397.6	172.2	296.6	39.8	89.2	102.8	86.6	103.5
Summarization	80	145.6	239.9	292.0	340.9	335.3	375.2	112.8	229.2	39.2	92.0	109.1	93.7	109.9
Filtered														
Coding	89	148.3	308.5	609.6	568.0	551.5	626.1	222.5	391.6	39.9	168.3	159.5	150.3	164.1
Humanities	88	147.8	275.5	396.9	454.1	550.2	643.4	213.3	314.3	39.9	101.6	115.7	98.4	114.1
QA	80	147.8	240.8	293.3	504.0	318.4	363.9	124.4	217.5	39.9	90.2	103.5	85.9	101.8
RAG	105	145.6	276.6	400.4	437.7	424.4	495.6	132.8	234.6	37.8	121.5	126.8	118.8	134.7
STEM	86	144.1	271.6	494.2	465.2	467.1	810.1	229.5	327.8	40.0	116.8	131.6	114.0	132.2
Multilingual	80	148.1	289.9	359.8	358.8	696.1	594.0	287.5	398.1	39.9	117.0	149.8	126.8	145.0
Roleplay	158	146.4	201.9	236.5	277.3	257.6	301.2	144.3	188.3	39.5	72.8	82.7	68.6	82.7
Math	98	144.6	325.9	536.7	555.7	687.3	846.7	243.8	347.7	39.9	142.6	146.0	138.4	150.4
Reasoning	190	147.7	286.8	430.0	463.5	562.8	630.3	157.8	292.4	39.8	118.8	125.5	108.2	126.6
Writing	84	147.5	241.0	303.9	378.6	339.3	397.6	167.8	211.3	39.8	89.2	102.8	86.6	103.5
Summarization	80	145.6	239.9	292.0	340.9	335.3	375.2	112.8	229.2	39.2	92.0	109.1	93.7	109.9

Table 13: Generation length (mean output tokens per turn), SpeedBench, thinking disabled. Every method is prompted with the same turns, so the *Unfiltered* spread across columns is the methods' own verbosity rather than a difference in workload; each *Filtered* value is over that method's own surviving turns. For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}15$, $B{=}7$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}8$, $B{=}6$, $R{=}4$ tree. *Diffusion* is FastDiffuser and *Linear SS* LinearSpec, both at block 32.

		Qwen3-8B						Nemotron-Diffusion-8B		Gemma-4-31B				
Category	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Diffusion	Linear SS	Base	DFlash	MTP	Static	Dynamic
Unfiltered														
Coding	89	604	696	585	577	1606	908	3733	2399	609	614	644	589	651
Humanities	88	898	2224	1132	1600	1788	3227	4548	5921	522	522	513	530	482
QA	80	285	745	284	714	1108	698	2870	1183	233	367	234	246	237
RAG	105	301	342	606	611	315	313	611	909	72	69	76	72	74
STEM	86	2933	4262	2334	2241	3902	4720	8036	6753	811	836	830	1183	1095
Multilingual	80	850	1205	1202	855	1626	810	1729	2544	334	332	332	326	327
Roleplay	158	227	210	217	232	227	229	6308	5630	185	187	188	178	180
Math	98	1421	2584	1397	1702	1961	2897	7358	5916	779	776	728	779	737
Reasoning	190	504	722	504	499	1060	692	4139	2604	332	331	331	330	332
Writing	84	1578	1436	1627	1600	1560	1538	9488	4600	1297	1292	1290	1272	1283
Summarization	80	165	187	166	166	164	167	278	282	126	126	122	124	124
Filtered														
Coding	89	604	696	585	577	519	546	2005	966	609	614	644	589	651
Humanities	88	898	770	768	876	1068	1065	2109	1682	522	522	513	530	482
QA	80	285	340	284	714	297	292	446	373	233	242	234	246	237
RAG	105	301	342	297	302	315	313	611	854	72	69	76	72	74
STEM	86	1478	930	1234	1137	941	1844	2811	1695	811	836	830	856	877
Multilingual	80	446	396	445	451	827	405	1336	529	334	332	332	326	327
Roleplay	158	227	210	217	232	227	229	916	359	185	187	188	178	180
Math	98	1421	1631	1397	1382	1319	1291	2782	2169	779	776	728	779	737
Reasoning	190	504	515	504	499	714	692	1566	633	332	331	331	330	332
Writing	84	1578	1436	1627	1600	1560	1538	3725	2039	1297	1292	1290	1272	1283
Summarization	80	165	187	166	166	164	167	278	282	126	126	122	124	124

Table 14: Turns dropped by the repetition filter, SpeedBench, thinking disabled. The filter acts per request, so one flagged turn drops every turn of that question. For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}15$, $B{=}7$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}8$, $B{=}6$, $R{=}4$ tree. *Diffusion* is FastDiffuser and *Linear SS* LinearSpec, both at block 32.

		Qwen3-8B						Nemotron-Diffusion-8B		Gemma-4-31B				
Category	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Diffusion	Linear SS	Base	DFlash	MTP	Static	Dynamic
Coding	89	0	0	0	0	3	1	5	5	0	0	0	0	0
Humanities	88	0	4	1	2	2	6	7	12	0	0	0	0	0
QA	80	0	1	0	0	2	1	6	2	0	1	0	0	0
RAG	105	0	0	1	1	0	0	0	1	0	0	0	0	0
STEM	86	4	9	3	3	8	8	15	14	0	0	0	3	2
Multilingual	80	1	2	2	1	2	1	1	5	0	0	0	0	0
Roleplay	158	0	0	0	0	0	0	45	36	0	0	0	0	0
Math	98	0	3	0	1	2	5	16	12	0	0	0	0	0
Reasoning	190	0	3	0	0	4	0	25	22	0	0	0	0	0
Writing	84	0	0	0	0	0	0	17	7	0	0	0	0	0
Summarization	80	0	0	0	0	0	0	0	0	0	0	0	0	0

E.3 MATH AND CODING BENCHMARKS, THINKING ENABLED

Table 15: Accepted tokens per forward pass, math and coding benchmarks, thinking enabled. The *Unfiltered* block is over all turns, the *Filtered* block excludes the turns the repetition filter drops (Table 18). For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}10$, $B{=}10$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}13$, $B{=}8$, $R{=}4$ tree.

		Qwen3-8B						Gemma-4-31B				
Benchmark	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Base	DFlash	MTP	Static	Dynamic
Unfiltered												
MATH500	80	1.000	2.826	3.795	4.108	4.866	5.665	1.000	7.399	5.500	5.305	7.164
MBPP	80	1.000	2.730	3.499	3.808	4.312	5.137	1.000	6.880	5.258	4.870	6.350
MT-Bench	160	1.000	2.424	2.927	3.362	3.604	4.259	1.000	4.541	4.251	3.622	4.699
GSM8K	80	1.000	2.815	3.521	4.022	4.515	5.240	1.000	6.931	5.477	5.047	6.738
AIME25	30	1.000	2.803	3.219	4.008	5.263	6.132	1.000	6.584	5.425	5.338	7.220
HumanEval	80	1.000	2.757	3.503	3.679	4.482	5.405	1.000	7.411	5.378	5.090	6.955
LiveCodeBench	80	1.000	2.570	2.763	3.632	4.793	5.672	1.000	5.317	4.903	4.546	5.843
Filtered												
MATH500	80	1.000	2.826	3.795	4.091	4.866	5.641	1.000	7.399	5.500	5.305	7.164
MBPP	80	1.000	2.715	3.470	3.761	4.312	4.977	1.000	6.880	5.258	4.870	6.350
MT-Bench	160	1.000	2.388	2.927	3.362	3.456	4.115	1.000	4.541	4.251	3.622	4.699
GSM8K	80	1.000	2.815	3.467	4.022	4.515	5.240	1.000	6.931	5.477	5.047	6.738
AIME25	30	1.000	2.749	3.219	4.008	5.222	6.100	1.000	6.584	5.425	5.318	7.220
HumanEval	80	1.000	2.731	3.503	3.679	4.328	5.124	1.000	7.411	5.378	5.058	6.955
LiveCodeBench	80	1.000	2.512	2.763	3.548	4.412	5.207	1.000	5.317	4.903	4.546	5.843

Table 16: Throughput (tokens/second), math and coding benchmarks, thinking enabled. Pooled $\sum$ tokens/$\sum$ wall within the split; the *Filtered* block is the quotable one. For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}10$, $B{=}10$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}13$, $B{=}8$, $R{=}4$ tree.

		Qwen3-8B						Gemma-4-31B				
Benchmark	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Base	DFlash	MTP	Static	Dynamic
Unfiltered												
MATH500	80	145.5	313.7	386.7	462.2	602.7	702.0	40.2	233.5	183.6	183.3	233.3
MBPP	80	145.7	318.0	408.1	453.8	550.8	704.8	40.3	203.2	169.9	164.4	198.2
MT-Bench	160	147.5	292.6	345.8	390.6	500.1	580.3	40.3	131.7	134.7	114.4	136.8
GSM8K	80	146.1	317.0	428.2	455.6	569.3	667.3	40.4	235.2	186.4	179.0	223.7
AIME25	30	139.1	299.7	310.8	413.5	569.0	661.7	40.1	214.3	182.0	181.9	225.5
HumanEval	80	145.8	326.3	404.9	423.0	642.4	786.9	40.3	241.7	176.0	180.5	222.9
LiveCodeBench	80	139.4	282.9	260.2	402.4	575.0	670.4	40.1	170.0	163.8	155.7	182.9
Filtered												
MATH500	80	145.5	313.7	386.7	454.1	602.7	694.2	40.2	233.5	183.6	183.3	233.3
MBPP	80	145.7	310.7	395.0	430.3	550.8	640.1	40.3	203.2	169.9	164.4	198.2
MT-Bench	160	147.5	275.6	345.8	390.6	448.4	528.0	40.3	131.7	134.7	114.4	136.8
GSM8K	80	148.5	317.0	394.8	455.6	569.3	667.3	40.4	235.2	186.4	179.0	223.7
AIME25	30	139.1	292.4	310.8	413.5	563.4	657.2	40.1	214.3	182.0	180.8	225.5
HumanEval	80	148.0	309.1	404.9	423.0	564.6	670.5	40.3	241.7	176.0	172.9	222.9
LiveCodeBench	80	139.7	271.1	259.2	388.8	523.4	604.8	40.1	170.0	163.8	155.7	182.9

Table 17: Generation length (mean output tokens per turn), math and coding benchmarks, thinking enabled. Every method is prompted with the same turns, so the *Unfiltered* spread across columns is the methods' own verbosity rather than a difference in workload; each *Filtered* value is over that method's own surviving turns. For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}10$, $B{=}10$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}13$, $B{=}8$, $R{=}4$ tree.

		Qwen3-8B						Gemma-4-31B				
Benchmark	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Base	DFlash	MTP	Static	Dynamic
Unfiltered												
MATH500	80	4463	4431	4708	4778	4758	4810	2103	2199	2260	1964	1975
MBPP	80	3322	3677	3903	3577	3213	4522	1466	1590	1474	1344	1460
MT-Bench	160	2085	2249	1956	1980	2224	2262	1154	1158	1201	1157	1227
GSM8K	80	2762	2412	2657	2209	2295	2277	532	524	531	511	513
AIME25	30	17233	17766	15373	17837	18625	19334	7239	7507	7441	7108	7260
HumanEval	80	2877	3157	2751	2595	3573	4462	928	855	930	1054	860
LiveCodeBench	80	12783	12054	13045	13907	14774	15104	6236	6163	6249	6064	5965
Filtered												
MATH500	80	4463	4431	4708	4423	4758	4456	2103	2199	2260	1964	1975
MBPP	80	3322	3206	3163	3207	3213	3422	1466	1590	1474	1344	1460
MT-Bench	160	2085	1875	1956	1980	1851	1883	1154	1158	1201	1157	1227
GSM8K	80	2382	2412	2276	2209	2295	2277	532	524	531	511	513
AIME25	30	17233	16103	15373	17837	18137	18871	7239	7507	7441	7000	7260
HumanEval	80	2499	2303	2751	2595	2436	2575	928	855	930	937	860
LiveCodeBench	80	12530	10068	12540	12378	11282	11677	6236	6163	6249	6064	5965

Table 18: Turns dropped by the repetition filter, math and coding benchmarks, thinking enabled. The filter acts per request, so one flagged turn drops every turn of that question. For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}10$, $B{=}10$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}13$, $B{=}8$, $R{=}4$ tree.

		Qwen3-8B						Gemma-4-31B				
Benchmark	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Base	DFlash	MTP	Static	Dynamic
MATH500	80	0	0	0	1	0	1	0	0	0	0	0
MBPP	80	0	2	2	1	0	3	0	0	0	0	0
MT-Bench	160	0	4	0	0	4	4	0	0	0	0	0
GSM8K	80	1	0	1	0	0	0	0	0	0	0	0
AIME25	30	0	3	0	0	1	1	0	0	0	1	0
HumanEval	80	1	3	0	0	3	5	0	0	0	1	0
LiveCodeBench	80	1	7	2	6	13	13	0	0	0	0	0

E.4 Math and coding benchmarks, thinking disabled

Table 19: Accepted tokens per forward pass, math and coding benchmarks, thinking disabled. The *Unfiltered* block is over all turns, the *Filtered* block excludes the turns the repetition filter drops (Table 22). For Qwen3-8B, *Static* is the $K{=}10$ tree and *Dynamic* the $K{=}10$, $B{=}10$, $R{=}4$ tree. For Gemma-4-31B, *Static* is the $K{=}7$ tree and *Dynamic* the $K{=}13$, $B{=}8$, $R{=}4$ tree.

		Qwen3-8B						Gemma-4-31B				
Benchmark	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Base	DFlash	MTP	Static	Dynamic
Unfiltered												
MATH500	80	1.000	3.112	7.533	6.243	6.217	6.895	1.000	6.780	5.398	4.934	6.536
MBPP	80	1.000	2.862	6.080	5.379	4.930	5.559	1.000	7.499	5.551	4.842	6.474
MT-Bench	160	1.000	2.453	3.994	4.229	3.627	4.166	1.000	3.850	3.949	3.325	4.250
GSM8K	80	1.000	3.054	5.861	6.145	5.194	5.896	1.000	5.880	5.270	4.496	5.726
AIME25	30	1.000	3.122	6.528	5.462	6.199	6.889	1.000	5.983	5.113	4.814	6.307
HumanEval	80	1.000	3.657	10.693	6.899	7.681	7.936	1.000	9.938	6.318	6.534	9.683
LiveCodeBench	80	1.000	2.907	6.123	5.267	5.508	6.026	1.000	5.413	4.790	4.227	5.403
Filtered												
MATH500	80	1.000	3.101	7.533	6.243	6.106	6.853	1.000	6.780	5.398	4.934	6.536
MBPP	80	1.000	2.847	6.080	5.379	4.930	5.559	1.000	7.499	5.551	4.842	6.474
MT-Bench	160	1.000	2.453	3.994	4.200	3.627	4.166	1.000	3.850	3.949	3.325	4.250
GSM8K	80	1.000	3.054	5.861	6.145	5.194	5.896	1.000	5.880	5.270	4.496	5.726
AIME25	30	1.000	2.863	6.591	5.462	5.738	6.768	1.000	5.983	5.113	4.814	6.307
HumanEval	80	1.000	3.657	10.693	6.899	7.681	7.936	1.000	9.938	6.318	6.534	9.683
LiveCodeBench	80	1.000	2.713	6.132	5.137	4.966	5.524	1.000	5.413	4.790	4.227	5.403

Table 20: Throughput (tokens/second), math and coding benchmarks, thinking disabled. Pooled $\sum$ tokens$/\sum$ wall within the split; the *Filtered* block is the quotable one. For Qwen3-8B, *Static* is the K=10 tree and *Dynamic* the K=10, B=10, R=4 tree. For Gemma-4-31B, *Static* is the K=7 tree and *Dynamic* the K=13, B=8, R=4 tree.

		Qwen3-8B						Gemma-4-31B				
Benchmark	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Base	DFlash	MTP	Static	Dynamic
Unfiltered												
MATH500	80	148.8	381.3	832.7	688.5	930.0	978.8	40.7	218.2	179.9	177.9	214.7
MBPP	80	140.2	360.4	657.6	582.3	619.6	693.7	40.6	220.3	181.3	159.6	197.9
MT-Bench	160	143.6	260.7	368.0	524.7	383.4	469.7	40.5	110.4	121.0	101.2	117.5
GSM8K	80	147.9	338.1	638.3	639.2	639.4	737.5	40.5	193.0	175.4	155.4	185.1
AIME25	30	139.3	387.5	652.0	610.9	924.6	942.9	40.2	175.1	170.5	164.7	192.1
HumanEval	80	146.5	391.7	1043.8	714.2	887.9	936.9	40.5	307.4	206.3	222.7	301.7
LiveCodeBench	80	135.0	394.1	605.4	739.4	995.1	1044.2	40.3	148.1	153.2	134.7	152.6
Filtered												
MATH500	80	148.8	370.7	832.7	688.5	766.3	952.6	40.7	218.2	179.9	177.9	214.7
MBPP	80	148.4	319.5	657.6	582.3	619.6	693.7	40.6	220.3	181.3	159.6	197.9
MT-Bench	160	148.3	260.7	368.0	490.6	383.4	469.7	40.5	110.4	121.0	101.2	117.5
GSM8K	80	147.9	338.1	638.3	639.2	639.4	737.5	40.5	193.0	175.4	155.4	185.1
AIME25	30	143.2	353.0	770.2	610.9	801.4	915.3	40.2	175.1	170.5	164.7	192.1
HumanEval	80	146.5	391.7	1043.8	714.2	887.9	936.9	40.5	307.4	206.3	222.7	301.7
LiveCodeBench	80	147.5	289.3	637.9	544.4	574.2	670.0	40.3	148.1	153.2	134.7	152.6

Table 21: Generation length (mean output tokens per turn), math and coding benchmarks, thinking disabled. Every method is prompted with the same turns, so the *Unfiltered* spread across columns is the methods' own verbosity rather than a difference in workload; each *Filtered* value is over that method's own surviving turns. For Qwen3-8B, *Static* is the K=10 tree and *Dynamic* the K=10, B=10, R=4 tree. For Gemma-4-31B, *Static* is the K=7 tree and *Dynamic* the K=13, B=8, R=4 tree.

		Qwen3-8B						Gemma-4-31B				
Benchmark	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Base	DFlash	MTP	Static	Dynamic
Unfiltered												
MATH500	80	696	1411	1142	620	1391	1407	582	569	625	684	571
MBPP	80	761	810	380	360	388	369	323	329	306	349	335
MT-Bench	160	697	522	497	910	488	493	464	474	467	472	465
GSM8K	80	262	283	265	270	276	275	184	180	184	183	183
AIME25	30	5558	11248	3626	2318	6200	5438	2335	2767	3116	2946	2589
HumanEval	80	163	162	162	162	160	161	184	185	183	181	181
LiveCodeBench	80	2770	5697	1520	1949	3577	3550	1260	1185	1244	1334	1288
Filtered												
MATH500	80	696	1014	1142	620	586	1011	582	569	625	684	571
MBPP	80	356	405	380	360	388	369	323	329	306	349	335
MT-Bench	160	494	522	497	708	488	493	464	474	467	472	465
GSM8K	80	262	283	265	270	276	275	184	180	184	183	183
AIME25	30	3614	3422	2621	2318	3248	4495	2335	2767	3116	2946	2589
HumanEval	80	163	162	162	162	160	161	184	185	183	181	181
LiveCodeBench	80	338	445	302	327	334	304	1260	1185	1244	1334	1288

Table 22: Turns dropped by the repetition filter, math and coding benchmarks, thinking disabled. The filter acts per request, so one flagged turn drops every turn of that question. For Qwen3-8B, *Static* is the K=10 tree and *Dynamic* the K=10, B=10, R=4 tree. For Gemma-4-31B, *Static* is the K=7 tree and *Dynamic* the K=13, B=8, R=4 tree.

		Qwen3-8B						Gemma-4-31B				
Benchmark	Turns	Base	I-DLM ISD	DFlash	DSpark	Static	Dynamic	Base	DFlash	MTP	Static	Dynamic
MATH500	80	0	1	0	0	2	1	0	0	0	0	0
MBPP	80	1	1	0	0	0	0	0	0	0	0	0
MT-Bench	160	2	0	0	2	0	0	0	0	0	0	0
GSM8K	80	0	0	0	0	0	0	0	0	0	0	0
AIME25	30	2	8	1	0	3	1	0	0	0	0	0
HumanEval	80	0	0	0	0	0	0	0	0	0	0	0
LiveCodeBench	80	6	13	3	4	8	8	0	0	0	0	0