DIFFUSION DRAFTERS CAN PREDICT BEYOND THEIR TRAINING BLOCKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive large language models (LLMs) generate tokens sequentially, while speculative decoding reduces inference latency by drafting multiple candidate tokens for parallel verification. Recent diffusion draft models generate a fixed-length block of candidates in a single forward pass, and extending this block could allow more tokens to be accepted per round. However, naive extension can disrupt original predictions through bidirectional attention, mixing gains from added predictions with losses at the original positions. To assess the benefit of added predictions, we study whether frozen draft models can use their beyond-block predictive capability while preserving accuracy at the original positions. We introduce Native-Preserving Extrapolation (NPE), which prevents original positions from attending to added positions while allowing added positions to use representations from the original block. Across four frozen DSpark and DFlash models, NPE preserves original-position accuracy and provides useful additional predictions that increase the number of tokens accepted per round. With the original predictions preserved, we further analyze what supports or disrupts the added predictions, when these predictions accelerate generation, and what further training adds to prediction quality. Our findings show that a draft model's useful prediction range can extend beyond its training block without sacrificing accuracy at the original positions.

1 INTRODUCTION

Autoregressive large language models (LLMs) generate a single token per forward pass (OpenAI et al., 2023; Yang et al., 2025). This sequential process makes generating long responses costly (Cai et al., 2024). Speculative decoding reduces this cost by using a lightweight draft model to propose multiple candidate tokens, which a target model then verifies in parallel (Leviathan et al., 2022; Chen et al., 2023). Diffusion models enable parallel drafting through denoising (Christopher et al., 2025; Li et al., 2026a), and recent approaches train lightweight diffusion language models to predict a fixed-length block of tokens in a single forward pass (Chen et al., 2026; Cheng et al., 2026; Huang et al., 2026). However, the block length limits how many candidates can be accepted per round, even when every prediction is correct. Extending the prediction block beyond its training length could therefore allow more draft tokens to be accepted in each round.

However, naive extension may reduce accuracy at the original positions (Wu, 2026), because bidirectional attention allows added positions to disrupt the original predictions. In our experiments with draft models trained on shorter blocks, extension nevertheless increases the number of accepted tokens per round and accelerates generation, even as original-position accuracy declines. These gains show that added predictions can already be useful, but their contribution is partly offset by damage to the original predictions. We therefore ask: **Can a frozen draft model provide useful predictions beyond its training block while preserving accuracy at the original positions?**

Answering this question requires extending the prediction block without altering the computation at the original positions. We introduce Native-Preserving Extrapolation (NPE) to satisfy this requirement. NPE prevents original positions from attending to added positions while allowing added positions to use representations from the original block. This preserves the original predictions and allows us to assess whether the added predictions remain useful under this constraint. Across all four evaluated

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.

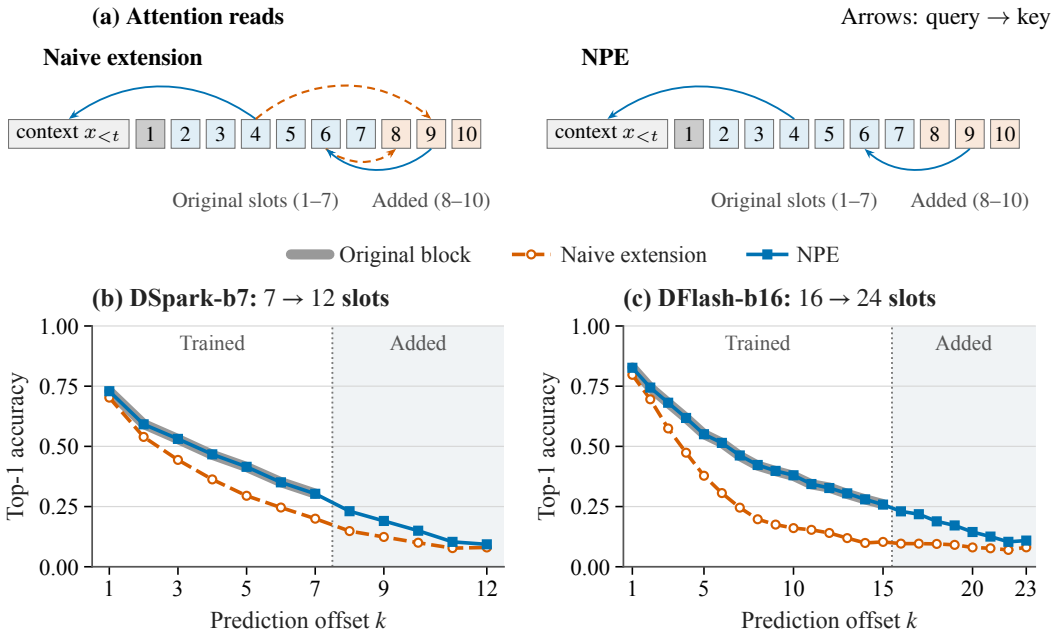

Figure 1: NPE preserves original predictions and extends beyond training. (a) Selected query-to-key reads for 7 → 10 slots; NPE removes dashed reads. (b, c) Qwen3-4B fixed-prefix top-1: 40 prompts, up to 64 anchors each. Shading marks added offsets; DFlash-b16 originally predicts 15 tokens.

draft models, NPE provides useful additional candidates that increase the number of draft tokens accepted per round without sacrificing original-position accuracy.

To understand how this existing capability can be used and improved, we study **what supports or disrupts the added predictions, when these predictions accelerate generation, and what further training improves beyond the predictions already available.** These analyses yield three findings:

1. Added predictions benefit from earlier representations but remain vulnerable to interference from later positions. Blocking attention to original-block representations reduces added-position accuracy. As the block grows, later added positions can also disrupt earlier added predictions. Restricting this interference helps maintain accuracy during further extension. Removing all added-position interactions reduces accuracy, confirming the value of earlier added representations.

2. Accepting more draft tokens per round does not always lead to faster generation. Verification can use an added prediction only after accepting the preceding candidates, whereas computing that prediction incurs a cost in every extended round. Extension accelerates generation when the additional accepted tokens compensate for this cost. When verification rarely reaches the added positions, generation becomes slower despite accepting more draft tokens per round. Extension therefore benefits generation when added predictions are used often enough to offset their cost.

3. Further training makes existing beyond-block predictions more accurate. In our DSpark training comparisons, protecting original predictions and restricting interference among added positions allow the short-block model to generate nearly as many tokens per round as the model trained on longer blocks. Yet, when both models use identical preceding text, the latter remains more accurate at distant positions. Further training therefore improves distant predictions beyond the accuracy recovered through prediction preservation.

In conclusion, our findings show that frozen draft models can make useful predictions beyond their training blocks while preserving accuracy at the original positions. The training block length therefore need not define the model's useful prediction range, and exploiting predictions beyond the training block need not come at the expense of predictions within the original block.

2 RELATED WORK

Parallel drafting uses multi-token heads (Cai et al., 2024; Gloeckle et al., 2024; Ankner et al., 2024), joint prediction (Monea et al., 2023; Lin et al., 2025; Bhendawade et al., 2024), or separate drafters (Xiao et al., 2024; An et al., 2025). Diffusion models predict masked tokens jointly (Nie et al., 2025; Arriola et al., 2025), enabling denoising-based proposals (Christopher et al., 2025; Li et al., 2026a) and lightweight block drafters (Chen et al., 2026; Cheng et al., 2026; Huang et al., 2026). Target features (Li et al., 2024a; 2025; Zhang et al., 2026b) and caching (Wu et al., 2025b;a) further improve drafting or inference. To build useful candidate sequences, methods use tree construction (Ringel & Romano, 2026; Lin & Jang, 2026; Li et al., 2026b), prefix-conditioned scoring (Wang et al., 2026b; Hu et al., 2026), or candidate correlation and refinement (Rusanovsky et al., 2026; Wang et al., 2026a). Extending the prediction block introduces another difficulty: added positions can disrupt original predictions. DFlash generalizes better to shorter blocks (Chen et al., 2026, §5.5.4), and DBloom separates added-position gains from losses at original positions and addresses extension-induced damage through post-training (Wu, 2026). Future Lens probes future-token information in existing representations (Pal et al., 2023), while gated adaptation and isolated attention add multi-token prediction without changing an autoregressive model's next-token computation (Samragh et al., 2025). We freeze an already-trained block drafter, preserve its original multi-position predictions, and measure predictions beyond its supervised range, then compare short- and long-block training. Using these predictions effectively requires both accurate candidates and appropriate computation costs. Acceptance-oriented training aligns learning with left-to-right verification through distillation, position weighting and prefix-based losses (Sandler et al., 2026; Wu et al., 2026; Whalen et al., 2026; Yang & Li, 2026); our training comparisons retain the DSpark objective, with additional attention and supervision controls in Appendix G. At inference, adaptive methods select tree structures (Li et al., 2024b; Chen et al., 2024; Wang et al., 2025), lengths (Huang et al., 2024), or configurations (Hou et al., 2025), while SpecExec targets parallel verification with offloaded models (Svirschevski et al., 2024). For diffusion drafting, BlockPilot selects length per prompt (Zhang et al., 2026a); DiffuSpec, FailFast and AdaFlash adapt length online (Li et al., 2026a; Pan et al., 2025; Qian et al., 2026). DSpark and CaDDTree incorporate verification costs (Cheng et al., 2026; Zhang et al., 2026c). Our analysis connects these costs to how often verification reaches the added positions.

3 PRELIMINARIES

This section introduces speculative decoding, diffusion drafting, and block extension, then describes how we measure prediction quality and generation efficiency.

3.1 SPECULATIVE DECODING

Speculative decoding uses a draft model q_θ to propose candidate tokens and a target model p to verify them (Leviathan et al., 2022; Chen et al., 2023; Xia et al., 2023). Given the generated prefix $x_{\leq t}$, the draft model proposes several candidates, which the target model evaluates in a single forward pass. Accepting several candidates advances generation in one round, as in blockwise parallel decoding and Lookahead Decoding (Stern et al., 2018; Fu et al., 2024).

The draft model can propose multiple alternatives at each position to increase the chance of matching the target model's output (Cai et al., 2024; Li et al., 2024b). These alternatives form a *draft tree*: each node represents a candidate token, and each path from the root represents a possible continuation. For a candidate sequence $y_1, \ldots, y_d$ along a path, the target model evaluates

$$p(y_k \mid x_{\leq t}, y_{<k}), \qquad k = 1, \ldots, d, \tag{1}$$

where $y_{<k}$ denotes the preceding candidates on that path. Tree attention computes these probabilities in one forward pass without exchanging information between branches (Miao et al., 2024).

Our experiments use DARTree's pruned draft trees (Li et al., 2026b) with sample-then-match verification (Miao et al., 2024). Verification follows one path, accepting a candidate at depth k only after accepting all preceding candidates. With L accepted draft tokens, the output is

$$(y_1, \ldots, y_L, z), \tag{2}$$

where the target model supplies z, giving $L + 1$ output tokens per round.

3.2 DIFFUSION DRAFTING AND BLOCK EXTENSION

Masked diffusion language models generate text by predicting missing tokens (Sahoo et al., 2024; Ye et al., 2025; Arriola et al., 2025). The diffusion draft models studied here predict several positions together (Chen et al., 2026; Cheng et al., 2026). They receive target hidden states for the preceding text $x_{<t}$ and a fixed-length block containing the last known token x_t and mask tokens for the remaining positions. The last known token serves as an *anchor*, while the mask tokens mark positions to be predicted. Bidirectional attention lets positions within the block exchange information, so a prediction can depend on representations at both earlier and later positions.

We call the input positions *slots* and denote the number used during training by B. The model produces predictions for D future tokens, indexed by their distance k from the anchor, called the *prediction offset*. In a draft tree, offset k corresponds to depth k. DeepSpec models use slot i to predict offset i, giving $D = B$; DFlash-b16 uses slot $i \geq 2$ for offset $i - 1$, giving $D = 15$. Correction heads, such as DSpark's Markov head or Domino's GRU, adjust candidate scores using the preceding tokens along each path (Cheng et al., 2026; Huang et al., 2026). Correlation scoring and parallel refinement also capture dependencies among candidate tokens (Rusanovsky et al., 2026; Wang et al., 2026a).

To extend prediction beyond the training block, we append K mask slots while keeping both models fixed. The extended block has $H = B + K$ slots and predicts K additional offsets, $D+1, \ldots, D+K$. We call the training-block slots *original* positions and the appended slots *added* positions. Naive extension retains bidirectional attention across all slots. Original positions can therefore attend to added positions, allowing extension to change existing predictions as well as produce new ones.

3.3 EVALUATION MEASURES

We measure prediction accuracy, use during verification, and generation speed.

Prediction quality. We give each configuration the same reference prefix and score the predictions from the language modeling head before Markov correction. Future input positions remain masked; reference tokens after the anchor are used for scoring only. We report top-1 accuracy, the fraction of argmax predictions matching the reference token, and negative log-likelihood (NLL), the negative logarithm of the probability assigned to that token. Higher accuracy and lower NLL indicate better predictions; Appendix A details the *fixed-prefix probe* protocol.

Acceptance length. Following Xia et al. (2024), we measure acceptance length as the average number of tokens generated per round,

$$\tau = \mathbb{E}[L + 1], \tag{3}$$

including the token supplied by the target model. Verification must first accept preceding candidates, so better distant predictions may barely increase acceptance length. Conditional acceptance is $a_k = \Pr(L \geq k \mid L \geq k - 1)$, the probability of accepting a candidate upon reaching depth k.

Throughput. We report mean per-prompt throughput: the number of generated tokens divided by decoding time for each request, averaged over prompts and timed runs. For MT-Bench, we first average the two turns of each conversation. Extension can increase both acceptance length and round time, so more tokens per round yield a speedup only when they offset the added computation time (Leviathan et al., 2022; Sadhukhan et al., 2024; Zhang et al., 2026c).

4 DRAFT MODELS CAN PREDICT BEYOND THEIR TRAINING BLOCKS

4.1 NATIVE-PRESERVING EXTRAPOLATION

Naive extension introduces new predictions while also changing predictions within the original block. Its overall performance therefore reflects both the quality of the added predictions and the disruption to existing predictions. To examine prediction beyond training while preserving the original predictions, we introduce Native-Preserving Extrapolation (NPE).

NPE prevents original positions from attending to added positions, while allowing added positions to attend to the preceding context, the original block, and one another. For a model trained with B slots

and extended to $H = B + K$ slots, we mask attention scores in every layer as follows:

$$M_{ij}^{\mathrm{NPE}} = \begin{cases} -\infty, & i \le B < j, \\ 0, & \text{otherwise}, \end{cases} \qquad 1 \le i, j \le H, \tag{4}$$

where i denotes the query position and j denotes the key position. As illustrated in Figure 1, this mask blocks attention from original queries to added keys. Attention to the preceding context and interactions within the original block remain unchanged. Added positions can thus use the original block's representations without affecting how those representations are computed.

For the same preceding text, original positions receive unchanged inputs and attend to the same keys and values as in the unextended block. This preserves their representations through successive layers and, in exact arithmetic, leaves their output logits unchanged. The applicable conditions and numerical checks under finite-precision computation are provided in Appendix B.

4.2 Prediction accuracy beyond the training block

We evaluate four frozen draft checkpoints: DSpark-b7 paired with Qwen3-4B or Qwen3-8B, and DFlash-b7 and DFlash-b16 paired with Qwen3-4B (Yang et al., 2025). We compare vocabulary-head predictions at original and added positions using identical reference prefixes: 40 prompts and up to 64 anchors per prompt (Appendix A). This removes differences in generated text from the comparison.

Figure 1 shows the position-wise accuracy of DSpark-b7 and DFlash-b16 on Qwen3-4B. Extending DSpark-b7 from 7 to 12 slots with bidirectional attention reduces top-1 accuracy at original offsets 4–7 by 22–34%. Under NPE, accuracy at all original offsets remains within 0.002 of the unextended model. Beyond training, accuracy declines gradually from 0.23 at offset 8 to 0.09 at offset 12.

DFlash-b16 exhibits the same pattern when extended from 16 to 24 slots. Naive extension reduces accuracy at original offsets 4–15 by 23–65%. NPE keeps original-position accuracy within 0.003 of the unextended model, while added-position accuracy declines from 0.23 at offset 16 to 0.11 at offset 23. The first added offset is 16 because the original block predicts 15 tokens.

These results show that prediction accuracy does not disappear at the training boundary. The frozen draft models can predict at added positions while retaining their original-position accuracy. Naive extension alone therefore cannot establish whether useful beyond-block prediction requires training.

4.3 Acceptance of the added predictions

Accurate predictions at added positions can increase the number of tokens produced per round only when verification reaches and accepts them. We therefore evaluate complete decoding on 2,320 prompts from GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021; Lightman et al., 2023), HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), and MT-Bench (Zheng et al., 2023). Checkpoints and evaluation protocols are detailed in Appendix A.

To evaluate whether extending the prediction block makes better use of a fixed verification budget, Table 1 keeps the number of verified tree nodes unchanged within each model group. Longer blocks use narrower trees, allocating more of this budget to distant positions. Naive extension and NPE use identical block lengths and tree widths, so their comparison measures the benefit of preserving the original computation under the same extended configuration.

Table 1 shows that naive extension has different effects across draft models. Extending DSpark-b7 from 7 to 10 slots increases acceptance length by 11.0% on Qwen3-4B and 10.0% on Qwen3-8B, despite reducing accuracy at original positions. For these models, the added predictions provide enough accepted tokens to outweigh the loss at original positions. In contrast, extending DFlash-b16 from 16 to 24 slots reduces acceptance length by 26.0%, consistent with Wu (2026).

NPE improves acceptance length over naive extension on all four checkpoints. Relative to the original block, DSpark-b7 gains 15.8% on Qwen3-4B and 13.8% on Qwen3-8B. For DFlash-b16, NPE turns the 26.0% decline under naive extension into a 1.1% increase using the same checkpoint. The added predictions therefore contribute to generation even in a model for which naive extension reduces overall acceptance. For the fourth checkpoint, DFlash-b7 on Qwen3-4B, extending to 10 slots with NPE raises τ from 6.782 under naive extension to 7.026, a 3.6% increase in the three-run comparison

Table 1: Acceptance and mean per-prompt throughput at a shared node budget. All rows within a group verify N tree nodes. Five benchmarks, three runs; all columns use the same full-test session; $\Delta\tau$ is relative to the original block. Shading denotes NPE and boldface denotes group maxima. Evaluation protocols appear in Appendices C, H.

Method	**Slots** H	**Width** w	**Nodes** N	$\tau\uparrow$	$\Delta\tau$ (%)	**tok/s**↑
DSpark-b7 / Qwen3-4B ($N = 160$)						
Original	7	23	160	6.426	—	184.0
Naive extension	10	16	160	7.133	+11.0	211.1
NPE	10	16	160	**7.441**	**+15.8**	**217.6**
DSpark-b7 / Qwen3-8B ($N = 80$)						
Original	7	12	80	6.232	—	170.6
Naive extension	10	8	80	6.857	+10.0	195.8
NPE	10	8	80	**7.090**	**+13.8**	**198.7**
DFlash-b16 / Qwen3-4B ($N = 160$)						
Original	16	11	160	7.431	—	**203.7**
Naive extension	24	8	160	5.500	−26.0	136.6
NPE	24	8	160	**7.514**	**+1.1**	195.3

Table 2: Original-state restoration accounts for most of NPE's gain. States are combined at $H = 10$; $\Delta\tau$ is relative to naive extension. Five benchmarks, 2,320 paired conversations (Appendix D).

State source		**DSpark-4B**		**DFlash-4B**	
Original	Added	$\tau\uparrow$	$\Delta\tau$ (%)	$\tau\uparrow$	$\Delta\tau$ (%)
Naive	Naive	7.133	—	6.782	—
NPE	Naive	7.380	+3.46	6.913	+1.93
Naive	NPE	7.230	+1.37	6.826	+0.64
NPE	**NPE**	**7.441**	**+4.32**	**7.026**	**+3.60**

on the same five benchmarks (Appendix C). Together, these results show that frozen draft models provide useful beyond-block predictions while preserving original-position accuracy.

5 Further Analysis

To fully understand how this existing capability can be used and improved, we also analyze what supports or disrupts beyond-block predictions, when these predictions accelerate generation, and how further training improves their quality.

5.1 Information and interference at added positions

NPE changes both the original representations and the information available to added positions. To determine which changes account for the acceptance gain over naive extension, we exchange original and added hidden states between the configurations before predicting tokens, following activation-patching interventions (Zhang & Nanda, 2024). This allows us to measure the contribution of restoring original representations while accounting for interactions with added representations. Restoring original states explains 74% of the acceptance-length gain on DSpark-4B, 68% on DFlash-4B, and 86% on DFlash-b16 (Table 2). Much of the difference between naive extension and NPE therefore comes from recovering predictions within the training block.

We next test what information supports the added predictions. We block added positions from attending to the original mask slots while retaining access to the preceding text and anchor. Averaged over offsets 8–10, this reduces top-1 accuracy by 11.8 percentage points on DSpark-4B and 28.0 points on DFlash-4B. Access to the preceding text and anchor alone therefore does not replace the

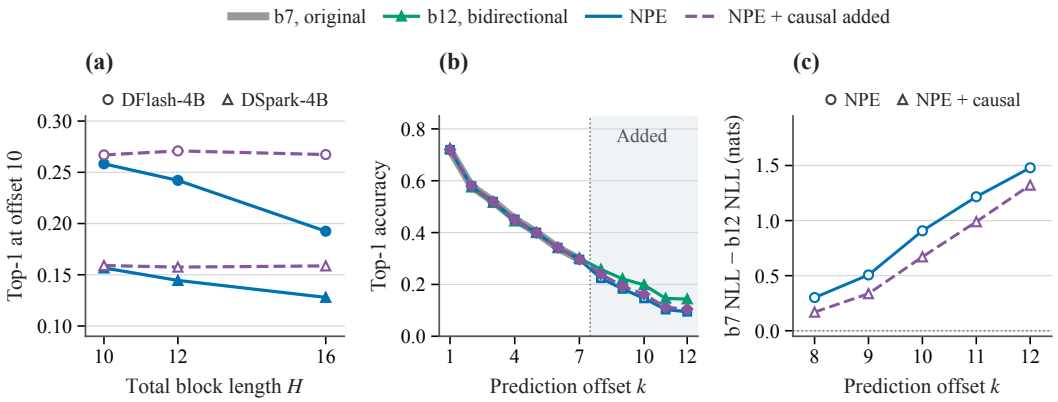

Figure 2: Added-position attention and longer-block training. (a) Offset-10 accuracy as block length grows. (b) DSpark-4B top-1 for the first training pair; shading marks added positions. (c) NLL differences for the same training pair: b7 with NPE (solid) or NPE + causal added (dashed), minus b12 with its original bidirectional attention. Positive values favor b12. Each panel uses 40 prompts with up to 64 fixed-prefix anchors each.

information computed within the original block. Replacing original-slot values in the final attention layer with values from another anchor also reduces DFlash-4B's added-position accuracy, even when keys and attention weights remain unchanged. Changing the position indices of added slots likewise reduces accuracy. These results show that added predictions use representations tied to the current context and the position being predicted (Appendix E).

Useful interactions must nevertheless be distinguished from harmful ones. NPE protects the original block, but added positions can still attend to one another in both directions. To test whether later positions disrupt earlier added predictions, we hold the prediction offset fixed and increase the total block length. At offset 10, DFlash-4B's accuracy falls from 0.258 at 10 slots to 0.193 at 16 slots. Preventing added positions from attending to later added positions keeps accuracy near 0.27 across these lengths (Figure 2(a)). We call this restriction *causal attention among added positions*: each added position attends to the context, the original block, and added positions up to itself.

Across four draft models, extending the block from 10 to 16 slots reduces top-1 accuracy at offset 10 by 1.6–6.6 percentage points under NPE, while accuracy changes by less than 0.4 points with causal attention among added positions (Appendix F). Removing all interactions among added positions instead lowers offset-10 accuracy at 16 slots from 0.267 to 0.149 on DFlash-4B and from 0.159 to 0.130 on DSpark-4B, despite preserving access to the original block. Earlier added representations therefore provide useful information, even though later positions can introduce interference. However, applying the causal restriction to the original block can be harmful: making all positions causal reduces acceptance on both DSpark sizes (Appendix C). The useful attention pattern preserves the original block's learned interactions while limiting interference from extension.

5.2 More tokens per round do not always lead to faster generation

We now ask when the additional predictions provided by NPE translate into faster generation. Table 3 uses the same three-run experiments as Table 1 and compares NPE extension with each model's original block. DSpark-4B and DSpark-8B are extended from 7 to 10 slots, while DFlash-b16 is extended from 16 to 24 slots. Model weights remain fixed, and throughput is measured as the mean of per-prompt generation rates.

The relevant distinction is between computing a prediction and using it. Following Wu (2026), let r_k be the probability that verification reaches depth k and a_k the probability of accepting the candidate conditional on reaching it. The probability that this depth contributes an accepted token is

$$\Pr(L \geq k) = r_k a_k, \tag{5}$$

Table 3: Selective extension: measured mean per-prompt throughput and an oracle estimate. The oracle applies added yield and cost only to full-acceptance rounds before averaging request rates. The last row subtracts unrounded changes. Five benchmarks, three runs from Table 1; all changes use the original block as reference. $D = 15$ for DFlash-b16 and 7 for DSpark; see Appendix H.

	DFlash-b16	**DSpark-4B**	**DSpark-8B**
Block extension	$16 \to 24$	$7 \to 10$	$7 \to 10$
Original block fully accepted	14.4%	58.1%	54.7%
Extra tokens when reaching $D + 1$	3.54	2.24	2.26
Extra time per round (ms)	+2.63 (6.6%)	+0.24 (0.6%)	+1.14 (2.8%)
Mean per-prompt throughput change			
Every round extended (measured)	-4.1%	$+18.3\%$	$+16.4\%$
Full-acceptance rounds only (oracle)	$+5.8\%$	$+21.6\%$	$+19.6\%$
Oracle minus measured (pp)	+9.9	+3.3	+3.1

where L is the number of accepted draft tokens. An extended block computes the added predictions in every round, whereas these predictions contribute only after the preceding candidates have been accepted.

Tables 1 and 3 show how this distinction affects speed. For DSpark-4B, NPE extension increases acceptance length by 15.8%, while adding 0.6% to the time per round. Mean per-prompt throughput therefore rises by 18.3%. DFlash-b16 gains only 1.1% in acceptance length while adding 6.6% to round time, and its throughput falls by 4.1%. The full-acceptance row in Table 3 helps explain the difference: the original block is fully accepted in 58.1% of DSpark-4B rounds but only 14.4% of DFlash-b16 rounds. DFlash-b16 therefore pays the extension cost in many rounds that end before the added predictions can contribute. DSpark-8B follows the DSpark-4B pattern, with 54.7% full acceptance and a 16.4% throughput gain.

The oracle rows in Table 3 estimate the opportunity to avoid this unused computation. The oracle extends only rounds in which the original block would be fully accepted, assuming that the measured added-token yield and extra cost transfer to those rounds. For DFlash-b16, the estimated throughput change moves from -4.1% under extension in every round to $+5.8\%$ under selective extension. This estimate requires advance knowledge of acceptance and is not an executable policy. It identifies the opportunity for adaptive extension: computing useful beyond-block predictions when verification is likely to use them.

The DSpark and DFlash-b16 results therefore connect the value of beyond-block predictions to when they are computed. Table 3 reports additional accepted tokens for all three models when verification reaches the added region, but extending every round pays for these predictions even when verification stops earlier. The oracle changes the timing of extension while retaining the measured prediction yield. Its estimated gain thus comes from avoiding unused computation, showing why both verification reach and extension cost matter when choosing a block length.

5.3 Further training makes existing beyond-block predictions more accurate

We next ask what training on a longer block adds beyond preserving existing predictions. Starting from the same DSpark-4B checkpoint (Cheng et al., 2026), we continue training on the same 40,000 prompts for three epochs with either 7 or 12 slots. We call the resulting models b7 and b12. Both undergo continuation training with the same data and number of epochs, so the comparison measures the effect of training at different block lengths. Both predict 12 positions at evaluation, so positions 8–12 are beyond b7's training block but within b12's. Table 4 separates the training length from the attention pattern used at inference; Appendix G provides training details and additional controls.

Table 4 supports two comparisons. The first three rows hold the trained b7 weights fixed and change only inference attention: naive extension, NPE, or NPE with causal attention among added positions. This comparison measures how much output can be recovered from the same model by reducing interference. NPE increases acceptance length, and the causal restriction improves it further. The fourth row provides the longer-block training reference: b12 using its original bidirectional attention

Table 4: DSpark-4B acceptance length at inference length 12 under different training lengths and inference attention patterns. Ranges span two matched training pairs, and τ includes the target token.

Model	Training length	Inference attention	$\tau \uparrow$
b7	7	Naive extension	7.003–7.004
b7	7	NPE	7.644–7.687
b7	7	NPE + causal added	7.762–7.801
b12	12	Bidirectional	7.875–7.877

across all 12 positions. Comparing this row with the b7 configurations measures the remaining acceptance-length gap after interference is reduced. The b12 model reaches $\tau = 7.875$–7.877, exceeding b7 with NPE and causal added attention by 0.9–1.5% across the matched training pairs. Reducing interference thus brings the short-block model close to the longer-block model in acceptance length.

Figure 2(b,c) compares prediction quality for the first training pair, with b12 retaining its original bidirectional attention throughout. In Figure 2(b), NPE and causal added attention improve b7's predictions, but its accuracy at positions 8–12 remains below b12's. Figure 2(c) plots the corresponding NLL differences: both b7 configurations have higher NLL than b12 at every added offset. For b7 with NPE and causal added attention, the top-1 accuracy gap to b12 increases from approximately 1.8 percentage points at offset 8 to 3.9 points at offset 12. The corresponding NLL gap increases from 0.17 to 1.32 nats, indicating a larger benefit from longer-block training at more distant positions.

The acceptance lengths in Table 4 and the prediction-quality curves in Figure 2(b,c) capture different effects. A more accurate prediction at an added position can increase acceptance length only when verification accepts the preceding candidates. Fixed-prefix evaluation instead compares predictions at each offset on identical preceding text, including offsets that decoding may rarely reach. Top-1 accuracy measures how often the predicted token is correct, while NLL is the negative log-probability of the reference token. Similar acceptance lengths therefore do not imply comparable prediction quality: b12 remains more accurate at the added positions, but this advantage can increase the number of accepted tokens only when verification reaches those positions. Reducing interference brings the short-block model close to the longer-block model in acceptance length, while longer-block training still yields more accurate predictions at the added positions.

6 Conclusion

Our results show that frozen draft models can make useful beyond-block predictions while preserving original-position accuracy. NPE blocks interference from added positions, whose predictions use earlier representations. Their decoding benefit depends on verification reach and computation cost. In our DSpark training comparisons, protecting existing predictions closes most of the acceptance-length gap to longer-block training, while longer-block training still improves distant prediction quality. Naive extension can underestimate existing capability; similar acceptance lengths can hide training gains.

7 Future work

A remaining challenge is translating beyond-block prediction into consistent acceleration. Our current approach computes added predictions even when verification stops within the original block. The oracle estimates gains from extending only fully accepted blocks, but requires advance knowledge of acceptance. Existing controllers adapt draft length using acceptance feedback, local difficulty or a learned length head (Li et al., 2026a; Pan et al., 2025; Qian et al., 2026). For NPE, controllers must identify profitable extensions at low overhead before computing added predictions. An adaptive policy must choose both whether to extend and how many positions to add. Recent acceptance history could guide this decision before drafting. Evaluation should include the controller's cost and compare mean per-prompt throughput against fixed-length extension on identical requests.

AI USE STATEMENT

AI tools (Claude and Codex) were used to assist with writing and language polishing, including revising wording and improving the clarity and readability of the manuscript. The authors are responsible for the manuscript and its final content.

REPRODUCIBILITY STATEMENT

The base drafter and target checkpoints used in this study are publicly available; the tree verifier is the public DARTree implementation with three additive changes (the NPE and control masks on the added slots, an all-zero mask for the backend-matched naive control, and a scorer without a correction head for the DFlash drafters); the evaluation protocols specify prompt sets, run counts, decoding parameters and statistical procedures, and Appendix H describes the timing exclusions and maps every main-text figure and table to its result files and script. The anonymous supplement contains the per-request result files behind Tables 1, 2 and 3, the fixed-prefix probe outputs behind Figure 2 and Tables 13, 14 and 20, and the analysis scripts used to compute the reported statistics, with their inputs and entry points listed in Appendix H. The remaining code and results will be released.

ETHICS STATEMENT

This work studies inference efficiency in existing open models and introduces no new data or model capability. Lossless sampling preserves the target model's output distribution.

REFERENCES

Zihao An, Huajun Bai, Ziqiong Liu, Dong Li, and Emad Barsoum. Pard: Accelerating llm inference with low-cost parallel draft model adaptation, 2025. URL https://arxiv.org/abs/2504.18583.

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding, 2024. URL https://arxiv.org/abs/2402.05109.

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models, 2025. URL https://arxiv.org/abs/2503.09573.

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, 2021. URL https://arxiv.org/abs/2108.07732.

Nikhil Bhendawade, Irina Belousova, Qichen Fu, Henry Mason, Mohammad Rastegari, and Mahyar Najibi. Speculative streaming: Fast llm inference without auxiliary models, 2024. URL https://arxiv.org/abs/2402.11131.

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, 2024. URL https://arxiv.org/abs/2401.10774.

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

Jian Chen, Yesheng Liang, and Zhijian Liu. Dflash: Block diffusion for flash speculative decoding, 2026. URL https://arxiv.org/abs/2602.06036.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis,

Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. In *Advances in Neural Information Processing Systems 37*, NeurIPS 2024, pp. 129531–129563. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-4116. URL http://dx.doi.org/10.52202/079017-4116.

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

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

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems 35*, NeurIPS 2022, pp. 16344–16359. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2022. doi: 10.52202/068431-1189. URL http://dx.doi.org/10.52202/068431-1189.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding, 2024. URL https://arxiv.org/abs/2402.02057.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction, 2024. URL https://arxiv.org/abs/2404.19737.

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, 2021. URL https://arxiv.org/abs/2103.03874.

Yunlong Hou, Fengzhuo Zhang, Cunxiao Du, Xuan Zhang, Jiachun Pan, Tianyu Pang, Chao Du, Vincent Y. F. Tan, and Zhuoran Yang. Banditspec: Adaptive speculative decoding via bandit algorithms, 2025. URL https://arxiv.org/abs/2505.15141.

Lanxiang Hu, Zhaoxiang Feng, Yulun Wu, Haoran Yuan, Yujie Zhao, Yu-Yang Qian, Bojun Wang, Peng Zhao, Daxin Jiang, Yibo Zhu, Tajana Rosing, and Hao Zhang. JetSpec: Breaking the Scaling Ceiling of Speculative Decoding with Parallel Tree Drafting. *arXiv preprint arXiv:2606.18394*, 2026. URL https://arxiv.org/abs/2606.18394.

Jianuo Huang, Yaojie Zhang, Qituan Zhang, Hao Lin, Hanlin Xu, and Linfeng Zhang. Domino: Decoupling causal modeling from autoregressive drafting in speculative decoding, 2026. URL https://arxiv.org/abs/2605.29707.

Kaixuan Huang, Xudong Guo, and Mengdi Wang. Specdec++: Boosting speculative decoding via adaptive candidate lengths, 2024. URL https://arxiv.org/abs/2405.19715.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2022. URL https://arxiv.org/abs/2211.17192.

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 2026a. 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/.

Tianyi Li, Yaxin Luo, Xinyi Shang, and Zhiqiang Shen. Dartree: Speculative diffusion decoding with autoregressive draft trees, 2026b. URL https://arxiv.org/abs/2608.13524.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty, 2024a. URL https://arxiv.org/abs/2401.15077.

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

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems 38*, NeurIPS 2025, pp. 151568–151587. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2025. doi: 10.52202/085713-4562. URL http://dx.doi.org/10.52202/085713-4562.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Feng Lin, Hanling Yi, Yifan Yang, Hongbin Li, Xiaotian Yu, Guangming Lu, and Rong Xiao. Bita: Bi-directional tuning for lossless acceleration in large language models. *Expert Systems with Applications*, 279:127305, 2025. ISSN 0957-4174. doi: 10.1016/j.eswa.2025.127305. URL http://dx.doi.org/10.1016/j.eswa.2025.127305.

Saw S. Lin and Jyh-Shing Roger Jang. Dominotree: Conditional tree-structured drafting with domino for speculative decoding, 2026. URL https://arxiv.org/abs/2607.08642.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017. URL https://arxiv.org/abs/1711.05101.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, pp. 932–949. ACM, April 2024. doi: 10.1145/3620666.3651335. URL http://dx.doi.org/10.1145/3620666.3651335.

Giovanni Monea, Armand Joulin, and Edouard Grave. Pass: Parallel speculative sampling, 2023. URL https://arxiv.org/abs/2311.13581.

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 38*, NeurIPS 2025, pp. 56354–56392. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2025. doi: 10.52202/085713-1689. URL http://dx.doi.org/10.52202/085713-1689.

OpenAI et al. Gpt-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774.

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron Wallace, and David Bau. Future lens: Anticipating subsequent tokens from a single hidden state. In Jing Jiang, David Reitter, and Shumin Deng (eds.), *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pp. 548–560, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.conll-1.37. URL https://aclanthology.org/2023.conll-1.37/.

Rui Pan, Zhuofu Chen, Hongyi Liu, Arvind Krishnamurthy, and Ravi Netravali. Fail Fast, Win Big: Rethinking the Drafting Strategy in Speculative Decoding via Diffusion LLMs. *arXiv preprint arXiv:2512.20573*, 2025. URL https://arxiv.org/abs/2512.20573.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models, 2023. URL https://arxiv.org/abs/2309.00071.

Ofir Press, Noah A. Smith, and Mike Lewis. Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2108.12409.

Yu-Yang Qian, Hao-Cong Wu, Chen Chen, Jiacheng Sun, Zhenhua Dong, Peng Zhao, and Zhi-Hua Zhou. AdaFlash: Adaptive Speculative Decoding via On-Policy Distilled Diffusion Drafters. *arXiv preprint arXiv:2607.19223*, 2026. URL https://arxiv.org/abs/2607.19223.

Liran Ringel and Yaniv Romano. Accelerating speculative decoding with block diffusion draft trees, 2026. URL https://arxiv.org/abs/2604.12989.

Matan Rusanovsky, Yoav Miron, Roy Uziel, Omer Belhasin, Hao Guo, Ran Zilberstein, Maor Ashkenazi, and Michael Elad. LiLiCorr: Lightweight Likelihood Correlation of Parallel Drafts for Speculative Decoding. *arXiv preprint arXiv:2608.20530*, 2026. URL https://arxiv.org/abs/2608.20530.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding, 2024. URL https://arxiv.org/abs/2408.11049.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems 37*, NeurIPS 2024, pp. 130136–130184. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-4135. URL http://dx.doi.org/10.52202/079017-4135.

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 preprint arXiv:2507.11851*, 2025. URL https://arxiv.org/abs/2507.11851.

Jameson Sandler, Jacob Christopher, Tom Hartvigsen, and Ferdinando Fioretto. SpecDiff-2: Scaling diffusion drafter alignment for faster speculative decoding. In *Proceedings of Machine Learning and Systems*, volume 8, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/hash/041dad5ed2191b44ba3ed0e00cdc3187-Abstract-Conference.html.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games*, volume 2, pp. 307–317. Princeton University Press, 1953.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models, 2018. URL https://arxiv.org/abs/1811.03115.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL http://dx.doi.org/10.1016/j.neucom.2023.127063.

Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. Specexec: Massively parallel speculative decoding for interactive llm inference on consumer devices. In *Advances in Neural Information Processing Systems 37*, NeurIPS 2024, pp. 16342–16368. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024. doi: 10.52202/079017-0522. URL http://dx.doi.org/10.52202/079017-0522.

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. Opt-tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Association for Computational Linguistics*, 13:188–199, 2025. ISSN 2307-387X. doi: 10.1162/tacl_a_00735. URL http://dx.doi.org/10.1162/tacl_a_00735.

Zheng Wang, Davis Wertheimer, Yu Chin Fabian Lim, Mudhakar Srivatsa, Raghu K. Ganti, Minjia Zhang, and Naigang Wang. xPress: Parallel Refinement for Diffusion Drafters in Speculative Decoding. *arXiv preprint arXiv:2608.02438*, 2026a. URL https://arxiv.org/abs/2608.02438.

Zheng Wang, Zhifan Ye, Qi Cheng, Yonggan Fu, Ziyan Wang, Feng Zhu, Haozhe Zhao, Jan Kautz, Pavlo Molchanov, Humphrey Shi, and Minjia Zhang. PRESTO: Prefix-Aligned Tree Drafting for Diffusion Speculative Decoding. *arXiv preprint arXiv:2607.22634*, 2026b. URL https://arxiv.org/abs/2607.22634.

Lexington Whalen, Yuki Ito, and Ryo Sakamoto. Teaching Diffusion to Speculate Left-to-Right. *arXiv preprint arXiv:2606.11552*, 2026. URL https://arxiv.org/abs/2606.11552.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion llm, 2025a. URL https://arxiv.org/abs/2509.26328.

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, 2025b. URL https://arxiv.org/abs/2505.22618.

Ephrem Wu. Ceiling-clipped acceptance histograms indicate stranded speed-up in block-diffusion speculative decoding, 2026. URL https://arxiv.org/abs/2608.30427.

Tianyu Wu, Yu Yao, Zhenting Qi, Han Zheng, Zhuohan Wang, Haoran Ma, Lawrence Liao, Himabindu Lakkaraju, Ju Li, and Yilun Du. D-PACE: Dynamic Position-Aware Cross-Entropy for Parallel Speculative Drafting. *arXiv preprint arXiv:2605.18810*, 2026. URL https://arxiv.org/abs/2605.18810.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3909–3925. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.257. URL http://dx.doi.org/10.18653/v1/2023.findings-emnlp.257.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 7655–7671. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.456. URL http://dx.doi.org/10.18653/v1/2024.findings-acl.456.

Zilin Xiao, Hongming Zhang, Tao Ge, Siru Ouyang, Vicente Ordonez, and Dong Yu. Parallelspec: Parallel drafter for efficient speculative decoding, 2024. URL https://arxiv.org/abs/2410.05589.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang

Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Tianjian Yang and Meng Li. Spec-auf: Accept-until-fail training under train-inference misalignment for masked block drafters, 2026. URL https://arxiv.org/abs/2607.01893.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Fred Zhang and Neel Nanda. Towards Best Practices of Activation Patching in Language Models: Metrics and Methods. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.16042.

Hao Zhang, Yiming Hu, Yong Wang, Mingqiao Mo, Xin Xiao, and Xiangxiang Chu. Blockpilot: Instance-adaptive policy learning for diffusion-based speculative decoding, 2026a. URL https://arxiv.org/abs/2606.31315.

Jiebin Zhang, Zhenghan Yu, Song Liu, Eugene J. Yu, Zheng Li, Dawei Zhu, Jiangshan Duo, Weimin Xiong, Yifan Song, Guanghua Yu, Jianchen Zhu, and Sujian Li. Dflare: Scaling up draft capacity for block diffusion speculative decoding, 2026b. URL https://arxiv.org/abs/2606.02091.

Shuai Zhang, Huachuan Qiu, Hongliang He, and Yong Dai. Cost-Aware Diffusion Draft Trees for Speculative Decoding. *arXiv preprint arXiv:2606.01813*, 2026c. URL https://arxiv.org/abs/2606.01813.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36*, NeurIPS 2023, pp. 46595–46623. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-2020. URL http://dx.doi.org/10.52202/075280-2020.

Appendix Contents

A Setup and evaluation protocol **16**

B NPE: definition, preservation and implementation checks **17**

C Complete decoding results **18**

D Matched prefixes and state-swap attribution **19**

E What the added predictions depend on **20**

F Interactions inside the added region **21**

G What continuation training adds **23**

H Runtime measurement and reproduction **24**

A SETUP AND EVALUATION PROTOCOL

Models and decoding. Table 5 lists the checkpoints and tree budgets. DFlash-b16 is the released z-lab base checkpoint evaluated by Wu (2026), without DBloom post-training. Decoding uses DARTree's pruned tree (Li et al., 2026b) and sample-then-match verification (Miao et al., 2024). The full evaluation comprises GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021; Lightman et al., 2023), HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021) and MT-Bench (Zheng et al., 2023): 2,320 prompts, up to 2,048 new tokens, $T = 1$ and batch size one. Timing uses three back-to-back runs per configuration on an exclusively held RTX 5880 Ada, alternating arm order across runs. Pilot experiments use 32 prompts per benchmark and 512 new tokens.

Table 5: Checkpoints and decoding budgets. B counts input slots and D predicted positions; DFlash-b16 reserves one slot for the anchor. Continuation training starts from the released DSpark-4B checkpoint.

drafter	target	B	D	head	layers	tree (w/N)	used in
DSpark-4B (DeepSpec)	Qwen3-4B	7	7	Markov	5	16/160	all sections
DSpark-8B (DeepSpec)	Qwen3-8B	7	7	Markov	5	8/80	§4, §5.1
DFlash-4B (DeepSpec)	Qwen3-4B	7	7	none	5	16/160	§4, §5.1
DFlash-b16 (z-lab, released base)	Qwen3-4B	16	15	none	5	8/160	§4, §5.2
Domino-b16	Qwen3-4B/8B	16	16	GRU	5	10/160	preservation check
DSpark-b7 (Gemma)	Gemma-4-12B	7	7	Markov	5	—	fixed prefixes
b7/b12 continuations	Qwen3-4B	7/12	7/12	Markov	5	16/160	§5.3

Aggregation and uncertainty. Acceptance length τ pools the per-round acceptance lengths, including the target token. Mean per-prompt throughput averages each request's generated-token count divided by its decoding time. MT-Bench's two turn-level rates are averaged within a conversation; conversations/prompts and timed runs then receive equal weight. Relative changes divide these mean rates, and the two MT-Bench turns stay together during resampling. We use 10,000 paired bootstrap draws: prompts for acceptance, and run→prompt hierarchical resampling for timing, with corresponding arms sampled together and prompts stratified by benchmark. Table 1 uses the hierarchical procedure for both metrics. The three timed runs share a generation seed, so repeated τ values do not measure sampling variation. Appendix H reports timing exclusions and variation across runs.

Fixed-prefix probes. The main probe set uses 40 validation-cache prompts and up to 64 anchors per prompt (2,540–2,560 total), uniformly spaced over valid loss-mask positions. Probes use bf16, with eager attention when attention weights are needed. Table 13 uses 40 prompts $\times$ 16 anchors (640 total). The DFlash-b16 set uses 40 prompts, 2,529 anchors spanning 24 positions, SDPA attention and the target's output head; causal target states before each anchor provide prefix features. The Gemma comparison uses 40 prompts $\times$ 16 anchors from its own cache. Top-1 compares the drafter's argmax with the true token; NLL averages the true token's negative log-probability. Resampling groups all anchors from each prompt into a single sampling unit.

Probe inputs and output heads. For an anchor x_t, the known prefix is $x_{\leq t}$. Each block contains x_t followed by $H-1$ mask tokens at positions $t, \ldots, t+H-1$. Its queries can access target features for $x_{<t}$ and slots within that block, subject to the chosen attention mask. Features at or after t and slots belonging to other anchors are masked out. The target features are causal, so computing or caching a complete reference sequence does not expose its future tokens to a block.

The probes call the backbone followed by the vocabulary head. DeepSpec models use `compute_logits`; DFlash-b16 uses the target's vocabulary head. DSpark's Markov correction head is not called: neither reference nor predicted tokens after the anchor condition the output scores. Let $s(k)$ map prediction offset k to its output slot: $s(k) = k$ for DeepSpec and $s(k) = k + 1$

for DFlash-b16, with slots numbered from one. We score each configuration at each anchor as

$$\begin{aligned} u &\leftarrow [x_t, \mathrm{MASK}, \ldots, \mathrm{MASK}], \\ h_{1:H} &\leftarrow \mathrm{Backbone}(u, \mathrm{TargetFeatures}(x_{<t}); M), \\ q_k &\leftarrow \mathrm{softmax}(\mathrm{VocabularyHead}(h_{s(k)})), \\ \mathrm{hit}_k &\leftarrow \mathbf{1}[\arg\max_v q_k(v) = x_{t+k}], \quad \mathrm{NLL}_k \leftarrow -\log q_k(x_{t+k}). \end{aligned}$$

Reference future tokens enter only the last line. These scores measure parallel prediction from a fixed prefix. Complete decoding instead uses DSpark's Markov correction with candidate predecessors along each tree path, followed by target verification.

Training and held-out checks. Continuation training uses 40k prompts disjoint from the benchmarks and a fixed three-epoch endpoint (step 231), without benchmark-based checkpoint selection (Appendix G). A stricter evaluation slice retains within-benchmark ranks ≥ 80 and excludes MT-Bench, leaving 1,920 prompts. On this slice, NPE improves τ over naive extension by 3.89% on DSpark-4B and 3.57% on DSpark-8B, consistent with the full-set results.

B NPE: DEFINITION, PRESERVATION AND IMPLEMENTATION CHECKS

Property 1 (Native preservation) *Assume that attention with a relative position encoding is the only operation that mixes slots, that all other operations act per position, and that any correction head reads only the prefix of a path. Under NPE the hidden states and logits at the original positions are identical, in exact arithmetic, to those of the original block-B forward pass at every layer.*

Proof of Property 1. We proceed by induction over layers. In the first layer, each native query attends to the same context and native keys, at the same relative positions, as in the native block-B forward pass; added keys are masked, so the native attention outputs are identical. Given identical native states at a layer, attention and per-position feed-forward operations preserve that equality at the next layer. A prefix-conditioned head also scores native paths without added states.

Here, *native* denotes the original block. The DSpark Markov head and Domino GRU read only path prefixes and satisfy the head assumption. Finite-precision deviations are reported in Table 6. Verification also depends on tree structure and round-start prefixes (Appendix D).

For block-level refinement heads such as xPress (Wang et al., 2026a), preservation also requires unchanged inputs and operations at original positions; unchanged backbone states alone are insufficient.

Backend-matched control. An all-zero mask routes naive extension through the same attention kernel as NPE. In one seven-arm pilot, naive extension with FlashAttention (Dao et al., 2022) gives $\tau = 6.629$ at 187.8 tok/s, the backend-matched naive control 6.722 at 189.1 tok/s, and NPE 6.964 at 199.2 tok/s. The mask's benefit therefore persists when the attention backend is matched.

Table 6: Original-state deviation $|\Delta h|$ / argmax agreement relative to the unextended forward pass. Deviations are absolute means over positions and hidden dimensions, except fp32 NPE cells, which report maxima (mean 2×10^{-6}). Each fp32 or Domino row uses 640 anchors. The bf16 DSpark and DFlash rows use all anchors from the main probe set.

drafter, extension	precision	NPE	naive	fully causal
DSpark-4B, $7 \to 10$	fp32 eager	$\leq 1.3 \times 10^{-4}$ / 100%	0.59 / 76.8%	0.54 / 79.8%
DSpark-4B, $7 \to 10$	bf16 SDPA	0.018 / 99.4%	0.65 / 74.0%	0.60 / 77.6%
DSpark-4B, $7 \to 12$	bf16 SDPA	0.019 / 99.2%	0.83 / 69.9%	0.60 / 77.6%
DSpark-8B, $7 \to 10$	bf16 SDPA	0.011 / 99.2%	0.35 / 77.3%	0.35 / 80.4%
DFlash-4B, $7 \to 10$	bf16 SDPA	0.020 / 99.3%	0.51 / 85.6%	0.70 / 81.1%
Domino-4B, $16 \to 24$	fp32 SDPA	$\leq 1.3 \times 10^{-4}$ / 100%	0.97 / 52.6%	0.66 / 70.8%
Domino-4B, $16 \to 24$	bf16 SDPA	0.014 / 98.6%	0.97 / 52.3%	0.66 / 70.4%

C COMPLETE DECODING RESULTS

Tables 7–10 supplement Table 1. Each comparison uses arms from the same session. Repeat-last recycles the b7 drafter's last original state for three additional depths, matching the extended arms' tree shape. Fully causal attention also restricts interactions within the original block. Table 1 uses the new matched-budget sessions, with every arm filling the shared node budget. Table 8 retains the earlier stock-width sessions as a separate comparison.

Additional generation seeds. On the five full sets, DSpark-4B at $H = 10$ gives naive / NPE τ of 7.111 / 7.466 with seed 20260916 and 7.118 / 7.439 with seed 31337. The paired gains are 4.99% and 4.52%, compared with 4.32% for the main seed. Each additional seed uses one run per arm.

Greedy decoding. At $T = 0$ on the five-benchmark pilot, NPE improves τ over naive extension by 5.5% on DSpark-4B and 3.0% on DSpark-8B. The corresponding throughput gains are 7.69% and 0.62%. The acceptance gain persists, while the speed gain remains model-dependent.

Table 7: Matched complete-decoding comparisons with repeat-last. Five benchmarks, three runs, $T = 1$, 2,048-token limit; $w = 16$, node budget 160 (4B), and $w = 8$, node budget 80 (8B). Throughput is mean per-prompt tokens per second; changes are relative to repeat-last.

Method	H	τ	$\Delta\tau$ (%)	tok/s	Δtok/s (%)
		DSpark-b7 / Qwen3-4B			
Repeat-last	7+3	7.053	—	196.9	—
Naive extension	10	7.133	+1.13	204.2	+3.71
NPE	10	7.441	+5.50	212.9	+8.15
Fully causal	10	6.931	−1.73	200.9	+2.05
		DSpark-b7 / Qwen3-8B			
Repeat-last	7+3	6.667	—	185.4	—
Naive extension	10	6.857	+2.85	195.5	+5.43
NPE	10	7.090	+6.35	198.7	+7.17
Fully causal	10	6.590	−1.16	187.3	+1.01

Table 8: Earlier stock-width sessions: mean per-prompt throughput on five benchmarks and three runs. Changes use the original narrow tree as reference; wider trees use 154 nodes (4B) and 77 (8B).

Configuration	tok/s	Δtok/s (%)
	DSpark-b7 / Qwen3-4B	
Original, stock width	183.9	—
Wider tree	183.7	−0.07
Naive extension	207.7	+12.98
NPE	213.6	+16.16
	DSpark-b7 / Qwen3-8B	
Original, stock width	171.3	—
Wider tree	166.0	−3.07
Naive extension	190.5	+11.23
NPE	196.9	+14.97

Table 9: DFlash-b7 without a correction head: five benchmarks, three runs, $w = 16$, node budget 160. τ averages verification rounds; throughput is the mean per-prompt rate. Repeat-last is the reference.

Configuration	τ	tok/s	$\Delta\tau$ (%)	Δtok/s (%)
Repeat-last	6.326	170.0	—	—
Naive extension	6.782	190.6	+7.21	+12.13
NPE	7.026	193.2	+11.07	+13.66

Table 10: Mean per-prompt throughput changes for the complete-test sessions of Table 1. Each group uses its original block as the reference for all reported changes.

Configuration	tok/s	Δtok/s (%)
DSpark-b7 / Qwen3-4B		
Original	184.0	—
Naive extension	211.1	+14.8
NPE	217.6	+18.3
DSpark-b7 / Qwen3-8B		
Original	170.6	—
Naive extension	195.8	+14.8
NPE	198.7	+16.4
DFlash-b16 / Qwen3-4B		
Original	203.7	—
Naive extension	136.6	−32.9
NPE	195.3	−4.1

D MATCHED PREFIXES AND STATE-SWAP ATTRIBUTION

Original acceptance on matched prefixes. The first round starts from the same benchmark prompt in every arm; MT-Bench second turns are excluded because their prefixes contain generated answers. With the original-depth node budget fixed, the DSpark-4B pilot gives original-position acceptance lengths of 5.944 for the b7 drafter and 5.925 for NPE: difference -0.019. This agrees with approximate preservation in bf16 (99.4% argmax agreement). Complete decoding can still differ at original positions because accepting added tokens changes later round boundaries.

State swaps. We use activation-patching interventions (Zhang & Nanda, 2024). For each round input, we run naive extension and NPE, restoring the drafter cache between passes, then combine the original and added hidden states before the prediction heads. Tree construction, verification and cache updates are unchanged. These arms require two forwards, so we report acceptance rather than speed. For DSpark, the Markov head subsequently scores candidate paths; swapping hidden states is therefore distinct from swapping final distributions.

Let τ_{00} denote naive extension, τ_{10} replacement of only original states with NPE states, τ_{01} replacement of only added states, and τ_{11} full NPE. We use a two-component Shapley attribution (Shapley, 1953), averaging each region's marginal contribution over the two replacement orders:

$$\phi_N = \tfrac{1}{2}[(\tau_{10} - \tau_{00}) + (\tau_{11} - \tau_{01})], \qquad \phi_E = \tfrac{1}{2}[(\tau_{01} - \tau_{00}) + (\tau_{11} - \tau_{10})].$$

These contributions sum to $\tau_{11} - \tau_{00}$ without assuming a zero interaction. On the full decoding sets (Table 2), restoring original states accounts for 74% of the gain on DSpark-4B and 68% on DFlash-4B. The respective ϕ_N/ϕ_E estimates are $0.229/0.080$ and $0.166/0.078$ tokens per round. The interaction terms are -0.036 for DSpark-4B and $+0.070$ for DFlash-4B.

For DFlash-b16 extended to 24 slots ($w = 8$, $N = 160$), the four arms give $\tau = 5.500/7.351/5.922/7.514$. The original-state contribution is 1.722, or 86% of the gain; the added-state contribution is 0.292. The interaction is -0.260. Attributions resample all four arms jointly.

First-round attribution. Table 11 repeats the comparison at shared initial prefixes. The b7 drafters gain less in the first round than over complete decoding; DFlash-b16 retains a large gain, with 98% attributed to original-state restoration. The first-round comparison isolates shared initial prefixes, while complete decoding includes the subsequent prefixes reached by each arm.

Table 11: State swaps on shared first-round prefixes across the five benchmarks, using only the first turn of MT-Bench. DSpark-4B and DFlash-4B use $w = 16$, $N = 160$; DFlash-b16 uses $w = 8$, $N = 160$. Contributions and interactions are in tokens per round.

first round, τ_1	naive	native←NPE	added←NPE	NPE	NPE − naive	ϕ_N	ϕ_E	interaction
DSpark-4B, H=10	8.504	8.549	8.441	8.597	+0.093	+0.100	−0.007	+0.112
DFlash-4B, H=10	7.149	7.122	6.993	7.154	+0.005	+0.067	−0.062	+0.188
DFlash-b16, 24 slots	5.210	7.018	5.076	7.243	+2.032	+1.987	+0.045	+0.359

E WHAT THE ADDED PREDICTIONS DEPEND ON

Interventions change only added-position computation, except the two marked isolated-anchor controls. We use the main probe set unless stated otherwise (Appendix A).

Position and attention controls. Table 12 compares four changes to the added slots: *pinned* gives all three the position id of slot 7; *shuffled* reverses their position ids (10, 9, 8); *no-native* removes their reads of the native slots; *context-only* additionally removes their reads of each other, retaining self-attention. Writing the block's key set as context $\cup\,\mathcal{N}\cup\mathcal{E}$, an added query reads context $\cup\,\mathcal{N}\cup\mathcal{E}$ under NPE, context $\cup\,\mathcal{E}$ under no-native, and context $\cup\,\{\text{itself}\}$ under context-only; a native query reads context $\cup\,\mathcal{N}$ in all four. The same four changes under naive attention are listed for comparison.

Table 12: Added-slot predictions on fixed prefixes when only the added slots are changed and the native side is held fixed by NPE. Native offsets 1–7 are identical across the NPE rows (DSpark-4B top-1 .725/.601/.534/.465/.413/.363/.297; DFlash-4B .869/.777/.724/.649/.599/.542/.482) and deviate by at most 0.002 from the corresponding unextended b7 predictions.

		top-1 at offsets 8 / 9 / 10			NLL at offsets 8 / 9 / 10		
drafter	added slots	8	9	10	8	9	10
DSpark-4B	naive extension	.167	.139	.111	6.12	6.71	7.01
	NPE	.240	.195	.147	5.19	5.95	6.49
	NPE + pinned	.088	.044	.038	7.53	9.90	10.97
	NPE + shuffled	.089	.195	.044	7.48	5.95	9.77
	NPE + no-native	.060	.050	.056	10.46	10.18	9.84
	NPE + context-only	.063	.040	.038	10.20	11.40	11.54
	naive + pinned	.082	.040	.036			
	naive + shuffled	.129	.139	.044			
	naive + no-native	.060	.050	.056			
	naive + context-only	.063	.040	.038			
DFlash-4B	naive extension	.378	.288	.230	3.09	3.68	4.04
	NPE	.422	.344	.255	2.90	3.43	3.80
	NPE + pinned	.067	.042	.035	7.41	9.38	10.23
	NPE + shuffled	.055	.344	.046	5.88	3.43	8.75
	NPE + no-native	.048	.043	.042	11.39	10.55	9.43
	NPE + context-only	.059	.046	.039	10.31	11.13	11.44

Position and read path at decode time. Our extrapolation extends the supervised prediction horizon, distinct from input-length extrapolation and context-window extension (Press et al., 2022; Peng et al., 2023). The beyond-block predictions depend on the added slots' rotary position offsets (Su et al., 2024) ($+7, +8, +9$ relative to the anchor, predicting offsets $+8, +9, +10$), which the

drafter has never predicted from but whose relative geometry lies inside its trained context range, and on their reads of the native block. On the five full sets, pinned costs -5.29% of pooled τ against NPE (7.047 vs. 7.441) and shuffled -4.75% (7.088), with added-slot $a_8..a_{10}$ of $0.81/0.58/0.42$ and $0.68/0.81/0.49$ against NPE's $0.88/0.86/0.82$ and native profiles equal to NPE's to two decimals.

Which native slots the added slots read. The block's first slot carries the anchor token (A), the remaining $B-1$ carry mask inputs (M). In the two "anchor in isolation" controls in Table 13, the anchor reads only the context and itself, preventing it from relaying information from the mask slots. The message swap applies activation patching (Zhang & Nanda, 2024) to attention values, replacing the mask slots' attention values with those computed at another anchor of the same prompt, matching scale and within-block positions. The last-layer swap leaves keys and attention weights unchanged.

Table 13: Attention to original positions, on 40 validation prompts $\times$ 16 anchors (640 total). Under bf16, full NPE differs from the unextended forward by at most 0.008 in original-position top-1 accuracy. The first-layer swap is a no-op because every mask slot initially has the same value vector.

	top-1 at offsets 8 / 9 / 10						NLL at offsets 8 / 9 / 10					
	DFlash-4B			DSpark-4B			DFlash-4B			DSpark-4B		
full NPE	.431	.323	.280	.252	.159	.136	2.75	3.81	3.73	4.88	6.33	6.62
added cannot read the mask slots	.052	.075	.067	.066	.055	.073	10.59	10.33	8.76	10.01	10.55	9.71
and the anchor in isolation	.047	.066	.075	.061	.058	.073	10.56	10.27	8.73	9.99	10.53	9.66
anchor in isolation, mask slots readable	.414	.311	.266	.250	.167	.142	2.83	3.87	3.76	4.93	6.36	6.63
mask-slot messages swapped, every layer	.087	.064	.105	.072	.058	.077	6.23	6.35	5.51	8.24	9.04	8.15
last layer only	.245	.231	.239	.211	.161	.138	3.76	4.38	4.07	5.20	6.52	6.59
first layer only	.431	.323	.280	.252	.159	.136	2.75	3.81	3.73	4.88	6.33	6.62
naive extension	.395	.295	.244	.169	.131	.094	2.93	4.06	3.93	5.79	7.25	7.17

F INTERACTIONS INSIDE THE ADDED REGION

The added slots read each other under NPE; this appendix changes only that graph, on fixed prefixes at three block lengths and at decode time on the full sets, and keeps the native branch untouched.

The added slots' own attention graph. NPE lets the added slots read each other. Table 14 compares that with a causal added block (added slot i reads added slots up to i) and with independent added slots (each reads only itself), all on the same prefixes and with the native branch untouched.

Table 14: Added-region attention at offsets 8–10 as block length H increases. Main probe set: 40 prompts $\times$ 64 anchors. Original predictions are unchanged. Causal attention preserves earlier added predictions; independent added slots lose useful interactions. Table 15 compares four drafters.

		DFlash-4B, offsets 8 / 9 / 10			DSpark-4B, offsets 8 / 9 / 10		
H	added slots read						
10	all added (NPE)	.404	.330	.258	.236	.184	.157
10	added up to themselves	.411	.341	.267	.242	.198	.159
10	themselves only	.411	.273	.146	.242	.171	.129
12	all added (NPE)	.396	.316	.242	.230	.179	.144
12	added up to themselves	.411	.339	.271	.243	.194	.157
12	themselves only	.411	.274	.144	.243	.170	.130
16	all added (NPE)	.369	.272	.193	.214	.153	.128
16	added up to themselves	.411	.341	.267	.242	.196	.159
16	themselves only	.411	.273	.149	.242	.169	.130

Across all four drafters, extending from $H = 10$ to 16 lowers offset-10 accuracy under NPE, while changes under causal added attention remain below .004 (Table 15).

The causal added region at decode time. Table 16 runs the two added-region graphs on the five full sets at the same H and node budget, with the native block bidirectional in both. The fully causal

Table 15: Top-1 at offset 10 on four drafters: the drop from $H{=}10$ to 16 under NPE, the change over the same range under the causal added graph, and the causal graph's margin over NPE at $H{=}16$. Point values at offset 10 (DFlash-4B and DSpark-4B in Table 14): DSpark-8B $.168/.166/.147$ under NPE against $.172/.171/.171$ causal at $H{=}10/12/16$; Gemma $.145/.139/.130$ against $.141/.139/.144$, with independent added slots giving top-1 accuracy .086 at each tested length.

drafter	prefix set	drop under NPE, $H{=}10 \to 16$	drop under causal graph, $H{=}10 \to 16$	causal − NPE at $H{=}16$
DFlash-4B	40×64	.066	.000	$+.075$
DSpark-4B	40×64	.029	.000	$+.031$
DSpark-8B	40×64 (own cache)	.021	.001	$+.024$
Gemma-4-12B	40×16 (own cache)	.016	$-.003$	$+.014$

control in Table 7 also restricts attention within the original block. Acceptance comparisons use a paired prompt-level bootstrap; timing follows Appendix H.

Table 16: Added-region attention during decoding: pooled τ and accepted draft tokens beyond the training block (depth ≥ 8 for b7, ≥ 16 for DFlash-b16). Five full sets, $T = 1$, 2,048 tokens, one run; $\Delta\tau$ is relative to NPE at the same length. Tree budgets follow Table 5.

family	H, added region	τ	beyond-block tokens per round	$\Delta\tau$ vs NPE
DSpark-4B	10, bidirectional (NPE)	7.441	1.207	—
	10, causal	7.444	1.222	$+0.04$
	12, bidirectional (NPE)	7.680	1.510	—
	12, causal	7.759	1.602	$+1.03$
	16, bidirectional (NPE)	7.505	1.363	—
	16, causal	7.902	1.794	$+5.29$
DFlash-4B	10, bidirectional (NPE)	7.026	0.999	—
	10, causal	7.028	1.010	$+0.03$
	12, bidirectional (NPE)	7.131	1.168	—
	12, causal	7.206	1.232	$+1.05$
	16, bidirectional (NPE)	7.071	1.063	—
	16, causal	7.279	1.307	$+2.94$
DSpark-8B	16, bidirectional (NPE)	7.258	1.378	—
	16, causal	7.540	1.687	$+3.89$
DFlash-b16	20, bidirectional (NPE)	7.417	0.344	—
	20, causal	7.465	0.351	$+0.65$
	24, bidirectional (NPE)	7.514	0.446	—
	24, causal	7.518	0.451	$+0.05$

At $H = 16$, causal attention among added positions improves mean per-prompt throughput over NPE by 7.75% on DSpark-4B and 2.65% on DFlash-4B; round time changes by -0.9% and $+0.1\%$, respectively. On DSpark-4B, extending from $H = 10$ to 16 lowers NPE throughput from 210.2 to 200.6 tok/s, whereas causal added attention raises it from 212.8 to 216.2 tok/s within that session.

Limited use of distant predictions. For DFlash-b16 at 24 slots, verification reaches the first added prediction in 12.6% of rounds, depth 20 in 6.0%, and the bonus-token stage in 1.4% (Table 17). Conditional acceptance differs by at most 0.02 between the two graphs. Low reach and similar conditional acceptance limit the gain in τ. Fixed-prefix accuracy is shown in Figure 1(c).

Table 17: DFlash-b16 extended to 24 slots: conditional acceptance $a_k = \Pr(L \geq k \mid L \geq k-1)$ and reach $r_k = \Pr(L \geq k-1)$, where L is the number of accepted draft tokens. Depth 24 is the bonus-token stage. Same full-set decodes as Table 16.

DFlash-b16, 24 slots	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	a_{21}	a_{22}	a_{23}
bidirectional (NPE)	0.86	0.84	0.82	0.80	0.77	0.70	0.67	0.65
causal	0.86	0.84	0.82	0.81	0.78	0.72	0.67	0.63
reach r_k (NPE)	k=16: 12.6%		k=20: 6.0%		k=24: 1.4%			

G WHAT CONTINUATION TRAINING ADDS

Matched training pairs. The released DSpark-4B b7 checkpoint is continued at block lengths 7 and 12 using the same 40k prompts and fixed three-epoch schedule, with two independent training seeds. Table 18 compares their full-set acceptance lengths. With b12 retaining its original bidirectional attention, its advantage over b7 with NPE + causal added is 0.94% and 1.48% for the two pairs. An additional attention control applies the same slot-7 boundary and causal added attention to b12, giving advantages of 0.57% and 1.46%.

Table 18: Acceptance length for the two continuation-training pairs on the five full sets (one decoding run per configuration). The first three columns use the b7 continuation; the last two use the b12 continuation under bidirectional attention or the additional causal slot-7 control.

	naive	NPE	NPE + causal added	b12 cont. bidirectional	b12 cont. causal slot-7
pair 1, 12 positions	7.004	7.687	7.801	7.875	7.846
pair 2, 12 positions	7.003	7.644	7.762	7.877	7.875
pair 1, 10 positions	7.077	7.421	7.452	7.456	—
pair 2, 10 positions	7.088	7.414	7.440	7.490	—

Continuation objective and configuration. All eight block-12 runs initialize from the same released DSpark-b7 checkpoint and use the same 39,619 cached training examples from the 40k-prompt corpus, 256 anchors per example and maximum sequence length 1,536. The model runs in bf16 with fp32 optimizer states; AdamW (Loshchilov & Hutter, 2017) uses learning rate 10^{-4}, no weight decay, 4% warmup and cosine decay, a global batch of 512 and three epochs (231 updates). Seeds are 42 and 1337. We retain DSpark's loss components and position weights while varying attention and supervision range; acceptance-oriented objectives instead change the learning signal to reflect verification (Sandler et al., 2026; Wu et al., 2026; Whalen et al., 2026; Yang & Li, 2026). The objective uses token cross-entropy, probability-distribution L1 and confidence-head binary cross-entropy, with coefficients 0.1, 0.9 and 1.0. Each term uses the same valid-position mask and offset weights $w_k = \exp(-(k-1)/4)$. Let $\mathcal{L}_N$ and $\mathcal{L}_E$ denote the resulting weighted mean losses within positions 1–7 and 8–12 of each example, respectively. The region control uses

$$\mathcal{L}_\lambda = \frac{\mathcal{L}_N + \lambda \mathcal{L}_E}{1+\lambda}, \qquad \lambda \in \{0, 1\}.$$

Thus $\lambda = 0$ supervises the original region, while $\lambda = 1$ gives each region half the total weight. This normalization is shared by both training attention graphs. The configuration is `config/dspark/dspark_qwen3_4b.py` at DeepSpec revision `005e03b`, with the overrides in `pair_train/train_2x2.sh`; `DSPARK_LOSS_REGION=7:0` or `7:1` sets the regional weights, and `DSPARK_TRAIN_GRAPH=npe_causal:7` selects causal slot-7 attention (unset for bidirectional). `train_2x2_config.json` records the resolved settings of all eight runs.

Separating training attention and supervision. The 2×2 comparison varies training attention (fully bidirectional or causal slot-7) and supervised positions (1–7 or 1–12), with two seeds per arm (Table 19). All arms are evaluated on the full decoding sets and on the same validation prefixes (40

prompts, up to 64 anchors, span 12). Under NPE + causal added at inference, added supervision improves τ by $0.95\%/0.86\%$ for bidirectional training and $1.87\%/1.76\%$ for causal training.

At added positions, supervision improves mean top-1 by .027 and .029 under bidirectional training, and .034 and .035 under causal training. Attention-only changes range from $-.008$ to $+.002$. Point estimates favor direct supervision, but individual comparisons remain uncertain.

Table 19: DSpark-4B continuation training at block 12: pooled τ under three inference graphs and fixed-prefix top-1 averaged over original positions 1–7 and added positions 8–12. Added-position accuracy uses NPE + causal added with the boundary after position 7.

training attention	supervised	seed	τ causal slot-7	τ NPE boundary	τ own graph	top-1 1–7	top-1 8–12
bidirectional	1–7	42	7.771	7.740	7.784	.474	.167
bidirectional	1–7	1337	7.781	7.744	7.798	.475	.168
bidirectional	1–12	42	7.845	7.851	7.900	.477	.194
bidirectional	1–12	1337	7.848	7.852	7.870	.477	.196
causal slot-7	1–7	42	7.747	7.643	—	.472	.162
causal slot-7	1–7	1337	7.767	7.653	—	.471	.160
causal slot-7	1–12	42	7.892	7.847	—	.474	.196
causal slot-7	1–12	1337	7.903	7.862	—	.474	.195

Table 20: Prediction quality at added offsets 8–12 for the same-data DSpark pair on fixed prefixes (40 prompts $\times$ 64 anchors). Top: top-1 accuracy. Bottom: NLL for the listed b7 and b12 attention configurations. Original-position accuracy is summarized in the accompanying text.

top-1 by offset	8	9	10	11	12
original b7 checkpoint + NPE to 12	.231	.190	.150	.103	.093
original b7 checkpoint, naive to 12	.148	.124	.100	.077	.080
b7 continuation + NPE to 12	.224	.183	.146	.103	.095
b7 continuation + NPE to 10	.235	.193	.158		
b7 continuation + NPE, causal added, to 12	.240	.193	.162	.112	.105
b7 continuation + NPE, causal added, to 10	.240	.192	.160		
b12 continuation, native	.258	.222	.198	.146	.143
b12 continuation, slot-7 boundary (NPE graph)	.260	.222	.202	.147	.142
b12 continuation, slot-7 boundary, causal added	.252	.216	.198	.150	.144
b12 continuation, run at 10	.257	.217	.194		
NLL, b7 continuation + NPE to 12	5.33	5.78	6.49	7.24	7.46
NLL, b7 continuation + NPE, causal added, to 12	5.19	5.61	6.25	7.01	7.30
NLL, b12 continuation, native	5.02	5.27	5.58	6.02	5.98
NLL, b12 continuation, slot-7 boundary, causal added	5.05	5.29	5.58	6.03	5.97

Additional attention controls. Applying the slot-7 boundary and causal added attention to both models also allows prediction quality to be compared under identical attention connections. Across both training pairs, b12 has higher top-1 accuracy and lower NLL at every added offset, while original-position accuracy differs by at most 0.013 (Table 20). For the first pair, causal added attention narrows b7's relative accuracy deficit from 13–34% against bidirectional b12 to 5–27% under this attention control. Under this control, the absolute top-1 gap grows from .013 at offset 8 to .039 at offset 12, and the NLL gap grows from 0.14 to 1.33 nats. Bootstrap draws resample the 40 prompts jointly across configurations (10,000 draws). Causal added attention improves b7 NLL at every added offset in both pairs. Added-position top-1 differs by less than .005 between training pairs under either matched attention pattern; the original checkpoint with NPE remains within 0.02 of the continued b7 model.

H RUNTIME MEASUREMENT AND REPRODUCTION

Table 21 decomposes per-round cost; below, we detail timing exclusions and run-level variation.

Table 21: Per-round cost on the full sets. Stage changes are in ms relative to the first row of each group; gain and cost are the emitted-token and round-time ratios. Emitted tokens per round differ from acceptance-trace τ by at most 0.01 because of the truncated final round. Stage differences within ± 0.5 ms are within single-session timing noise.

	arm	τ (tok/round)	T_{round} (ms)	Δdraft	Δtree	Δverify	gain	cost
4B, native session	b7 tree (w=16, 112 nodes)	6.367	37.13	—	—	—	—	—
	naive H=10 (160 nodes)	7.126	38.51	+0.00	+0.87	+0.48	1.119	1.037
	NPE H=10 (160 nodes)	7.434	38.90	+0.30	+0.87	+0.53	1.168	1.048
4B	repeat-last	7.048	39.57	—	—	—	—	—
	naive H=10	7.126	39.60	+0.01	+0.00	+0.02	1.011	1.001
	NPE H=10	7.434	39.17	+0.21	−0.08	−0.48	1.055	0.990
	fully causal	6.926	39.97	+0.40	+0.00	+0.05	0.983	1.010
8B	repeat-last	6.658	40.95	—	—	—	—	—
	naive H=10	6.850	40.94	+0.01	+0.01	−0.04	1.029	1.000
	NPE H=10	7.082	41.32	+0.32	+0.02	+0.01	1.064	1.009

Definitions behind Table 3. The three-run sessions of Table 1 provide each original-block request's generated tokens T_{0i}, decoding time C_{0i} and full-acceptance round count n_{fi}. The extended runs provide the mean added-token yield $\bar{e}$ conditional on reaching depth $D{+}1$; Δc is the difference between extended and original mean round times. For each request, the oracle estimates

$$v_{\text{oracle},i} = \frac{T_{0i} + n_{fi}\bar{e}}{C_{0i} + n_{fi}\Delta c}.$$

We average these estimated rates using the same turn, prompt and run weights as measured throughput. The estimate assumes that conditional yield and extra per-round cost transfer to the original run's full-acceptance rounds despite different round-start prefixes. It represents neither an implemented throughput nor a strict bound. The calculation is produced by `summarize_throughput.py` from the retained v3 request records; `audit_numbers.py` (section F) displays the estimates.

Timing exclusions and sensitivity. A timed unit is one benchmark/run pair with all arms run back-to-back. The scheduler checks for foreign compute processes on the session GPU; contaminated or interrupted units are rerun as a whole. A second rule reruns units when an arm's CPU overhead exceeds $1.5\times$, or verification time exceeds $1.15\times$, the unit minimum. Three units were flagged only by this timing rule: 4B main HumanEval run 0, and 8B native HumanEval run 2 and MBPP run 2. These timing exclusions concern the earlier comparison sessions. The three matched-budget sessions of Table 1 recorded no foreign compute process on a session GPU. The timing rule flagged the 4B HumanEval and MBPP units of all three runs and the 8B HumanEval unit of run 1 (verification time 1.16–$1.34\times$ the unit minimum): the slower arms verified in 33–39 ms per round against 26–29 ms in every other unit and in the earlier sessions, on three cards during the same hour, so the disturbance was host-wide rather than a property of the trees. Every unit of the affected benchmarks was re-timed, including the two 8B HumanEval units whose arms had slowed together and escaped the ratio rule. After re-timing, neither threshold flags any unit; no DFlash-b16 run needed replacement.

The exclusions affect throughput, while acceptance lengths are deterministic for a fixed generation seed. Discarded attempts of the earlier sessions were overwritten, preventing an all-attempts analysis there; the matched-budget sessions retain them, and keeping them in place of the re-timed units would move the NPE gains over the original block in Table 1 from $+18.3\%$ / $+16.4\%$ to $+18.49\%$ / $+17.04\%$ on Qwen3-4B / Qwen3-8B. Tightening the thresholds to $1.3\times$ / $1.10\times$ removes two 4B units and no 8B unit. Across the remaining three complete runs on 1,820 prompts, NPE improves mean per-prompt throughput over repeat-last by 7.86%.

Independent timing check. A separate DSpark-4B session compares naive extension and NPE at $H = 10$ on the full sets, retaining every attempt and excluding units only when a foreign GPU process is recorded. Four contaminated units in run 0 were rerun on another card. The three per-run throughput gains are 2.71%, 3.63% and 2.41%, with a gain of 2.92% after averaging across runs; the mean time per decoding round increases by 1.2%.

Run-level variation. Table 22 reports the main-session NPE versus naive contrasts. Rows marked budget use the sessions in Table 1; the remaining rows use earlier sessions.

Table 22: Mean per-prompt throughput changes for NPE versus naive extension. Each run uses its paired reference. *All runs* computes relative changes after averaging per-prompt rates over three runs.

Session	Run 0 (%)	Run 1 (%)	Run 2 (%)	All runs (%)
4B, repeat-last session	+7.97	+2.30	+2.79	+4.28
8B, repeat-last session	−0.24	+2.52	+2.70	+1.65
DFlash-b7	+0.65	+1.90	+1.57	+1.37
4B, budget	+2.56	+3.87	+2.74	+3.06
8B, budget	+0.71	+0.91	+2.73	+1.45
DFlash-b16, budget	+42.97	+43.53	+42.25	+42.92

Target-only comparisons. Table 23 reports matched-subset timing on the first 80 request rows per benchmark: 400 rows, including both turns of 40 MT-Bench conversations. Each session uses one exclusively held GPU, batch size one, $T = 1$ and up to 2,048 generated tokens; tree decoding and the harness's target-only loop process the same prompts. Speedup is the ratio of mean per-prompt throughputs within the same session. Each rate uses the same turn and prompt weighting as the complete-test results. Target-only time uses the measured autoregressive loop's wall time, while tree decoding uses its recorded decode time. These comparisons are reported separately from the complete-test sessions. Repeat-last and fully causal attention have no target-only comparison.

Target-only rates are similar for Qwen3-4B and Qwen3-8B, consistent with substantial per-step overhead in this implementation. Speedups are relative to this generation loop. The stock-width sessions had a higher host load than the extension sessions (load averages approximately 65 and 52); comparisons use the paired baseline within each session. All rates and speedups are produced by `summarize_throughput.py`; `compute_speedup3.py` displays subset results.

Reproduction index. Table 24 maps the main figures and tables to per-request results under `results/frozen-horizon-v1/` and their analysis scripts. `summarize_throughput.py -root results/frozen-horizon-v1` reads retained results; redirect its output to `data/throughput_summary.json`; `render_throughput.py` updates the throughput tables and shared numerical values. Figure 2(b,c) uses exported probe summaries in `data/`. Table-specific scripts produce fixed-prefix and attribution results.

Table 23: Target-only comparisons on matched subsets: 400 request rows, including both turns of 40 MT-Bench conversations (360 conversations/prompts). Each group is a separate session. Both rates average per-prompt token/time ratios; speedup divides tree throughput by the paired target-only throughput. $T = 1$, batch size one, 2,048-token limit.

Configuration	H	Target tok/s	Tree tok/s	Speedup
		DSpark-4B, stock		
Original, $w = 16$	7	42.2	171.0	4.06×
Wider, $w = 22$	7	42.2	172.3	4.09×
		DSpark-4B, extension		
Naive, $w = 16$	10	43.5	188.9	4.34×
NPE, $w = 16$	10	43.5	197.9	4.55×
		DSpark-4B, budget		
Original, $w = 23$	7	44.8	179.0	3.99×
		DSpark-8B, stock		
Original, $w = 8$	7	40.5	154.9	3.82×
Wider, $w = 11$	7	40.5	152.9	3.78×
		DSpark-8B, extension		
Naive, $w = 8$	10	42.7	174.7	4.10×
NPE, $w = 8$	10	42.7	178.8	4.19×
		DSpark-8B, budget		
Original, $w = 12$	7	44.0	163.1	3.71×
		DFlash-b16, extension		
Naive, $w = 8$	24	43.6	123.7	2.84×
Original, $w = 8$	16	43.6	184.5	4.23×
NPE, $w = 8$	24	43.6	174.1	4.00×
Wider, $w = 10$	16	43.6	189.5	4.35×
		DFlash-b16, budget		
Original, $w = 11$	16	45.2	200.3	4.44×

Table 24: Result directories and analysis scripts for the main figures and supporting tables.

main-text item	result directories	script
Figure 1(b)	`probe/offset_acc_dspark4b_b12.json`	`probe_paths.py -suite factorial`
Figure 1(c)	`probe/offset_acc_dflash16_b24.json`	`probe_dflash16_offsets.py`
Tables 1, 10	`full4b_v3`, `full8b_v3`, `full4b_dflash16_v3`	`summarize_throughput.py`
Table 23	`npe/ar_subset_4b`, `npe/ar_subset_8b`, `npe/ar_subset2_*`, `npe/ar_subset3_*`	`summarize_throughput.py`
Table 2	`fullset/swap_*`, `full4b`, `full4b_dflash`	`swap_attrib.py`
Table 11	`clean_timing`, `fullset/swap_*`, `full4b_dflash*`, `causal_added`	`first_round_swap.py`
Figure 2(a)	`graph_probe_pp`	`probe_paths.py -suite graph`
Table 3	`full4b_v3`, `full8b_v3`, `full4b_dflash16_v3` (original and NPE arms, three runs)	`summarize_throughput.py`
Figure 2(b,c)	`data/pair_offsets.json`, `data/pair_graph_probe`	`render_analysis.py`
Table 19	`train_2x2/eval`	`summarize_2x2.py`
Tables 7, 8, 9	`full4b`, `full8b`, `full*_native`, `full4b_dflash`	`summarize_throughput.py`
Tables 16, 18, 4	`causal_added`, `causal_added_8b`, `full4b_dflash16_graph`, `pair_eval`	`audit_numbers.py` (sections D, E), `causal_report.py`