FACTORIZATION-GUIDED TRAINING AND INFERENCE FOR DIFFUSION-DRAFTED SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding accelerates large language model inference through parallel verification of tokens proposed by a drafter. Diffusion drafters further parallelize proposal generation, but greater proposal capacity or higher mean accepted tokens (MAT) need not yield higher throughput. Because verification accepts only a consecutive prefix, early rejections can render later correct predictions ineffective, limiting MAT gains, while even higher MAT may not improve throughput if per-round latency increases. We introduce **DScout**, which uses two complementary factorizations to guide diffusion drafter training and inference for higher throughput. For training, DScout factorizes position-wise loss into inclusion, weighting, and prediction loss. This guides the design and combination of mixed-horizon training, intra-block unmasking, and acceptance-weighted loss to align supervision with prefix acceptance and improve MAT. For inference, DScout factorizes throughput as the ratio of MAT to per-round latency, guiding target-conditioning depth, drafter depth, and inference block size selection to identify a high-throughput configuration for each workload. Across six benchmarks, DScout achieves $3.84\times$ and $4.14\times$ mean speedups over autoregressive decoding on Qwen3-4B and Qwen3-8B, respectively, improving mean speedup over the state-of-the-art baseline by **9.1%** and **7.1%**. Across models and concurrency settings, our inference configuration policy provides up to **9.75%** additional mean speedup over DScout's fixed configuration.

1 INTRODUCTION

Autoregressive (AR) large language models generate tokens sequentially, requiring a model forward pass for each new token. Speculative decoding amortizes this cost by using a lightweight drafter to propose multiple tokens for parallel verification in a single target-model pass, while preserving the target model's output distribution (Leviathan et al., 2023; Chen et al., 2023). Diffusion drafters further parallelize proposal generation by predicting multiple draft tokens simultaneously. For example, DFlash conditions a lightweight block-diffusion model on target-model features to generate an entire proposal in a single draft pass (Chen et al., 2026).

However, improving parallel draft predictions does not necessarily translate into higher throughput. Verification accepts draft tokens only up to the first rejection, so later correct predictions contribute to mean accepted tokens (MAT) only if the preceding prefix is accepted. Training should therefore account for this prefix dependence when allocating supervision to improve MAT. Higher MAT, in turn, improves throughput only when the additional accepted tokens compensate for increased draft–verify latency. These challenges motivate jointly considering prefix dependence in training and the MAT–latency trade-off at inference.

We introduce **DScout**, which uses two complementary factorizations to guide diffusion drafter training and inference (Figure 1). A position-wise loss factorization guides supervision allocation to improve MAT, while a throughput factorization guides inference configuration selection through the MAT–latency trade-off.

For training, a correctly predicted token contributes to MAT only if the preceding prefix is accepted. This motivates examining not only prediction loss, but also which positions receive supervision and how their losses are weighted. We therefore factorize each position's loss contribution into inclusion, weighting, and prediction loss: whether the position receives supervision, how strongly it is weighted, and how its prediction loss is computed. Training interventions affecting different factors can be

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.

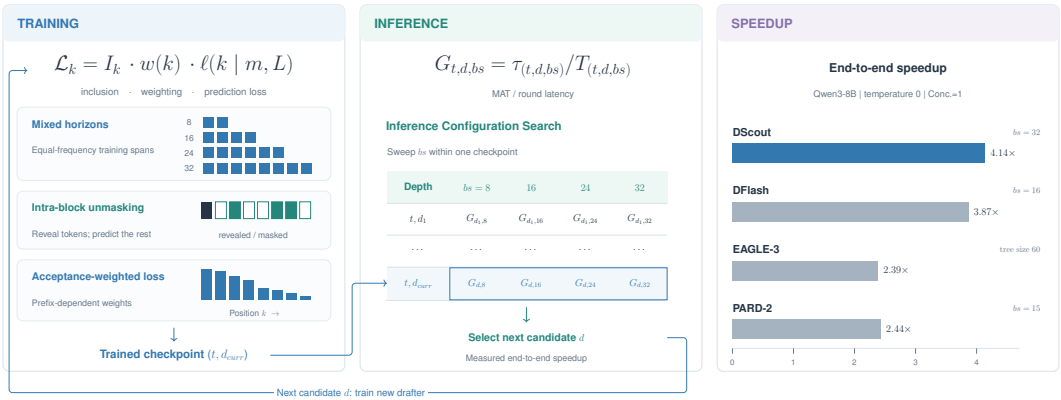

Figure 1: **Factorization-guided training and inference.** Training shapes position-wise supervision; inference selection balances MAT against latency. Right: six-task mean speedup for the displayed configurations from Table 3; each method uses its matched AR baseline.

studied separately, while those affecting the same factor should be considered jointly to account for potential interactions. Guided by this view, we design a recipe combining mixed-horizon training, intra-block unmasking, and acceptance-weighted loss to align supervision with prefix acceptance and improve MAT. Section 3.1 details the motivation and design of each component.

For inference, we factorize throughput into MAT and latency contributions as $G = \mathrm{MAT}/T$, where T is the latency of one draft–verify round. Guided by this factorization, we identify three broadly applicable configuration dimensions for diffusion drafters: target-conditioning depth, drafter depth, and inference block size. We assess their joint effects on MAT and latency to guide configuration selection. Our cost analysis and empirical measurements motivate full target conditioning with negligible incremental overhead, drafter depth selection that balances MAT gains against latency, and workload-aware inference block size selection without retraining through mixed-horizon training. Together, these policies guide the selection of high-throughput configurations for different models and concurrency settings. Section 3.2 details the analysis and selection procedure.

Our key contributions are:

- We formulate a position-wise loss factorization to guide diffusion drafter training, yielding a recipe that combines mixed-horizon training, intra-block unmasking, and acceptance-weighted loss to improve MAT without additional inference-time computation.
- We factorize throughput as the ratio of MAT to latency, guiding the joint selection of target-conditioning depth, drafter depth, and inference block size for high-throughput configurations across workloads.
- We validate the training recipe and inference configuration selection across target models, benchmarks, and concurrency settings through end-to-end comparisons and ablation studies.

2 Preliminaries

2.1 Speculative Decoding

In each speculative decoding round, a lightweight drafter proposes $bs - 1$ tokens conditioned on the verified context, where bs is the inference block size. The target verifies them in one parallel forward pass, accepts the consecutive prefix before the first rejection, and emits a replacement token at that position or a bonus token after full acceptance. Appropriate acceptance and replacement rules preserve the target model's output distribution (Leviathan et al., 2023; Chen et al., 2023).

Let N denote the number of draft tokens accepted before the first rejection, with $0 \le N \le bs - 1$. The conditional acceptance probability a_k and the prefix-survival probability $S(k)$ are

$$a_k = \Pr(N \ge k \mid N \ge k-1), \qquad S(k) = \Pr(N \ge k) = \prod_{j=1}^{k} a_j. \tag{2.1}$$

Mean accepted tokens (MAT) per round is

$$\tau = 1 + \mathbb{E}[N] = 1 + \sum_{k=1}^{bs-1} S(k), \tag{2.2}$$

where the additional one counts the target replacement or bonus token. The product in $S(k)$ captures the prefix dependence: a draft token can contribute to accepted length only if all preceding draft tokens are accepted.

2.2 Block-Diffusion Drafting

Block-diffusion language models generate text by unmasking tokens within a block using the available context. When used as speculative drafters, they predict multiple future tokens jointly rather than generating a proposal autoregressively. The target model then verifies the proposal as described in Section 2.1.

We adopt DFlash's target-conditioned block-diffusion architecture (Chen et al., 2026), which predicts all masked positions in a single forward pass. Its bs-position draft block contains the latest target-generated token followed by $bs - 1$ mask tokens.

To condition these predictions on the verified context, hidden states from t selected layers of the frozen target model are concatenated and fused through a lightweight linear projection:

$$\mathbf{h}_{\text{ctx}} = W_f[h^{(\ell_1)}; \ldots; h^{(\ell_t)}] + b_f, \qquad W_f \in \mathbb{R}^{H_D \times tH_T}. \tag{2.3}$$

Here $h^{(\ell_i)} \in \mathbb{R}^{H_T}$ represents the same context token at target layer ℓ_i; H_T and H_D are the target and drafter hidden widths, $\mathbf{h}_{\text{ctx}} \in \mathbb{R}^{H_D}$ is the fused feature, and $b_f \in \mathbb{R}^{H_D}$ is the projection bias. The target remains frozen. Each drafter layer uses its own projections to map the fused features into cached keys and values reused across rounds. Block positions attend to this context and bidirectionally to one another.

3 Method

DScout combines training recipe design with inference configuration selection. Two complementary factorizations guide these stages: a position-wise loss factorization for improving MAT, and a throughput factorization for balancing MAT against draft–verify latency.

3.1 Training Recipe Design

The prefix-survival relation in Eq. (2.1) shows that a correctly predicted token contributes to MAT only if the preceding prefix is accepted. Improvements in token-level predictions therefore need not yield corresponding gains in accepted length. This motivates examining how supervision is allocated across positions to better align training with the MAT objective.

3.1.1 Position-Wise Loss Factorization

We formalize this supervision allocation through a position-wise loss factorization.

During training, each block consists of a visible anchor token and $B - 1$ subsequent target tokens, where B denotes the training horizon. A mask pattern m specifies which future tokens within the block are masked for prediction and which are revealed as context.

For each future position k relative to the anchor, let I_k indicate whether it is included in the training loss: the position must lie within the training horizon and be masked for prediction. We factorize its loss contribution as

$$\mathcal{L}_k = \underbrace{I_k}_{\text{inclusion}} \underbrace{w(k)}_{\text{weighting}} \underbrace{\ell(k \mid m, L)}_{\text{prediction loss}}. \tag{3.1}$$

Here I_k determines whether position k receives supervision, $w(k)$ controls how strongly that position's loss is weighted, and $\ell(k \mid m, L)$ specifies the prediction loss under mask pattern m and

loss function L. The factorization accommodates different loss functions L. In this work, we use token-level cross-entropy throughout and do not explore alternative loss functions.

The training horizon B and mask pattern m may vary across updates. For a fixed position k, $\pi(k) = \mathbb{E}_{B,m}[I_k]$ measures how frequently it receives supervision. This exposes the position-dependent allocation of training signal induced by horizon and mask choices.

3.1.2 Factorization-Guided Training Strategies

We use the factorization to analyze training strategies in terms of the factors they affect. For each strategy, we explain its motivation and its effects on supervision, then examine interactions among strategies to guide their joint design.

Mixed-horizon training. Figure 2 shows that DFlash, with training and inference block sizes both set to 16, accepts the entire draft proposal in 10.9% of verification rounds, suggesting potential for longer continuations. Since extending inference beyond the training horizon can reduce acceptance length (Chen et al., 2026), we increase both training and inference block sizes to 32. As shown in Figure 2, this extends accepted lengths beyond 16 tokens, but also increases the proportion of rounds accepting only 1–4 tokens, indicating more early rejections.

This trade-off motivates mixed-horizon training, in which we use $B \in \{8, 16, 24, 32\}$ with equal frequency. By varying the inclusion frequency $\pi(k) = \mathbb{E}[I_k]$, this strategy emphasizes early positions while retaining long-horizon coverage. Varying B also changes the block context and thus the prediction-loss term $\ell(k \mid m, L)$. As shown in Figure 2, mixed-horizon training reduces early rejections relative to fixed-32 training while retaining continuations beyond 16 tokens at the same inference block size.

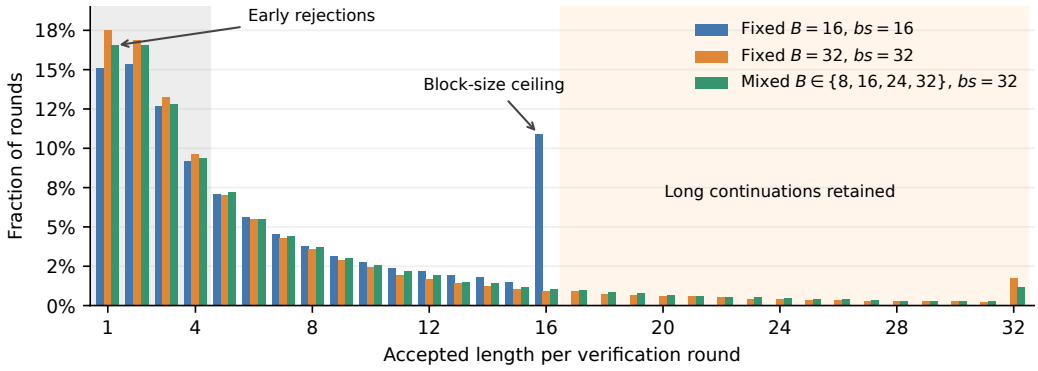

Figure 2: **Mixed horizons reduce early rejections relative to fixed-32 training.** Accepted-length distributions for Qwen3-8B hidden-9t9d checkpoints trained for six epochs. Probabilities average dataset-normalized histograms equally over the six evaluation tasks. The legend specifies training horizon B and inference block size bs; shading marks lengths 1–4 and beyond 16.

Intra-block unmasking. Random masking patterns provide data augmentation for diffusion language models (Ni et al., 2025). Revealing different subsets of tokens exposes the model to diverse prediction contexts beyond fully masked blocks. We apply intra-block unmasking by mixing fully masked and partially revealed training blocks.

During training, we retain fully masked future blocks with a fixed probability. For the remaining blocks, we sample a masking ratio from a Beta distribution and randomly reveal the complementary subset of future positions. Revealed positions contain the corresponding target tokens and provide context without contributing to the prediction loss. Unmasking changes I_k by excluding revealed positions from the loss, and $\ell(k \mid m, L)$ by providing context for the remaining predictions.

Acceptance-weighted loss. Position-dependent loss decay assigns larger weights to positions near the beginning of each draft block (Chen et al., 2026), but assigns the same weight to a given position across training examples, regardless of how likely the preceding draft prefix is to be accepted. We introduce acceptance-weighted loss (AWL), which dynamically weights each position's loss based

on the likelihood that verification reaches it. AWL acts on the weighting factor $w(k)$ in Eq. (3.1), replacing fixed position-dependent weights with prefix-dependent acceptance weights.

Let Q_j denote the drafter's output distribution at position j given the training block under mask pattern m and the target context features, and P_j the target distribution conditioned on its own generated prefix. For the draft prediction $\widehat{y}_j = \arg\max_v Q_j(v)$, we define

$$\alpha_j = \min\left(1, \frac{P_j(\widehat{y}_j)}{Q_j(\widehat{y}_j)}\right), \qquad w_k^{\mathrm{AWL}} = \mathrm{sg}\left[\prod_{j<k} \alpha_j\right],$$

where sg denotes stop-gradient and $w_1^{\mathrm{AWL}} = 1$. The target probability $P_j(\widehat{y}_j)$ is evaluated under the target-generated prefix, while $Q_j(\widehat{y}_j)$ comes from the drafter's prediction distribution. Both probabilities refer to the same draft token $\widehat{y}_j$. We compute α_j using the standard speculative-sampling acceptance rule (Leviathan et al., 2023; Chen et al., 2023).

The cumulative product of these acceptance scores reflects the sequential nature of verification: the token at position k can be accepted only if all preceding draft tokens are accepted. An earlier prediction with a low acceptance score therefore reduces the weights of subsequent positions. Unlike fixed position decay, AWL dynamically allocates supervision according to the acceptance scores of the drafter's predictions, linking the training weights directly to the verification rule.

We optimize the weighted mean cross-entropy over supervised positions:

$$\mathcal{L}_{\mathrm{train}} = \frac{\sum_{r,k} I_{r,k} w_{r,k}^{\mathrm{AWL}} \mathrm{CE}(Q_{r,k}, y_{r,k}^{\star})}{\sum_{r,k} I_{r,k} w_{r,k}^{\mathrm{AWL}} + \epsilon},$$

where r indexes training blocks in the minibatch, $y_{r,k}^{\star}$ is the corresponding target-generated token, and $\epsilon = 10^{-6}$ ensures numerical stability.

In this objective, prefix acceptance weights scale each position's contribution to the gradient. Early in training, the drafter's limited predictive ability yields low acceptance scores, suppressing later-position gradients and concentrating supervision near the start of each block. As the drafter improves its predictions at earlier positions, the cumulative acceptance weights of later positions increase, progressively extending supervision to longer prefixes. This progression aligns optimization with the verification process, suppressing gradient contributions from later positions when their preceding prefixes are unlikely to be accepted.

Joint design. The factorization guides not only the design of individual strategies, but also their combination. Mixed-horizon training and intra-block unmasking share the inclusion and prediction-loss factors, motivating joint evaluation. Although each improves MAT alone, combining intra-block unmasking with mixed horizons $\{4, 8, 16, 32\}$ degrades performance (Table 1). Unmasking short blocks may leave too few masked tokens, providing limited learning signal. We therefore use $\{8, 16, 24, 32\}$ to reduce short-block exposure. The combination then outperforms either strategy alone, though its gains are not additive, consistent with overlapping effects on shared factors. AWL targets the weighting factor $w(k)$ and is designed separately before being combined with the jointly designed horizon and masking strategies. The cumulative ablation in Figure 4 shows their incremental contributions and combined gains.

Table 1: Mean MAT with and without intra-block unmasking ($bs = 32$).

Training horizon	w/o unmasking	w/ unmasking
Fixed 32	5.942	6.248
Mixed $\{4, 8, 16, 32\}$	6.228	6.197
Mixed $\{8, 16, 24, 32\}$	6.208	**6.263**

Table 2: MAT across training horizons and inference block sizes. Settings in Section A.3.

Training horizon	$bs = 8$	16	24	32
Fixed 16	4.61	**6.01**	5.87	5.71
Fixed 32	4.31	5.29	5.65	6.18
Mixed $\{8, 16, 24, 32\}$	**4.85**	5.95	**6.24**	**6.38**

3.2 Inference Configuration Selection

The training recipe in Section 3.1 improves MAT without introducing additional inference-time computation. For inference, we factorize throughput into MAT and latency contributions, highlighting

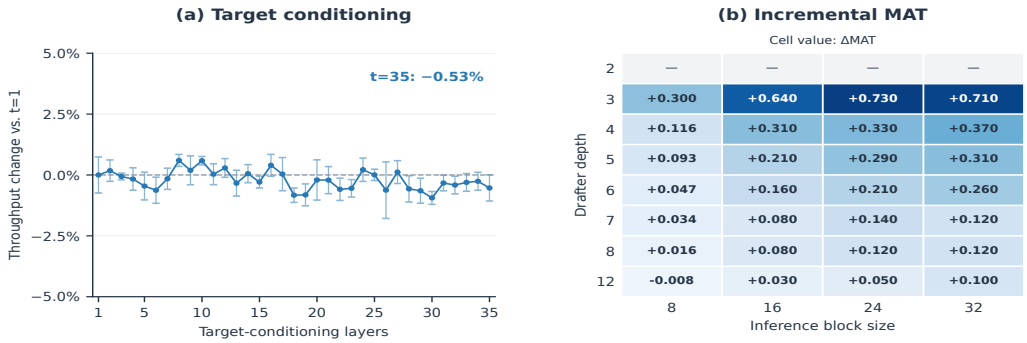

Figure 3: **Configuration design on Qwen3-8B.** (a) Throughput change with target-conditioning depth at fixed acceptance. (b) Marginal MAT gains with drafter depth; the last row spans depths 8–12. Protocols are given in Appendix A.3 and A.4.

the need to jointly assess their changes. This view guides our analysis of three configuration dimensions in diffusion drafters: target-conditioning depth t, drafter depth d, and inference block size bs. Our configuration-selection objective is

$$(t^\star, d^\star, bs^\star) = \arg\max_{t,d,bs} \frac{\tau(t,d,bs)}{T(t,d,bs)},$$

where τ is MAT (Eq. (2.2)), T is the latency of one draft–verify round (Liu et al., 2024), and the starred variables denote a throughput-maximizing configuration. We analyze these three dimensions to guide inference configuration selection.

Full target conditioning. Since the target model already computes hidden features at all layers, increasing t only enlarges the input to the fusion projection in Eq. (2.3), without changing subsequent drafter computation. Our FLOPs analysis shows that, for Qwen3-8B at $bs = 16$ and MAT ≈ 7, increasing t from 9 to 35 adds only 2.75% of the target projection and MLP FLOPs (Appendix A.4). Including attention matrix products and drafter computation in the total would further reduce this fraction. Figure 3(a) complements this analysis with controlled profiling at a fixed MAT of 16: throughput varies by less than 1% across the tested conditioning depths, with a 0.53% decrease from $t = 1$ to $t = 35$. This small measured cost motivates using all available target features under the tested execution conditions.

Unimodality-guided drafter depth selection. Each candidate drafter depth requires training a separate drafter, making an exhaustive depth sweep costly. We therefore seek to locate the throughput-maximizing depth efficiently. Our depth study reveals diminishing returns in MAT: additional drafter layers yield progressively smaller acceptance gains (Figure 3(b)). In contrast, each additional layer introduces the same computation. At a fixed inference block size, round latency grows approximately linearly with depth, as supported by controlled profiling (Appendix A.5). Together, these observations suggest a unimodal throughput curve under the MAT–latency factorization.

To formalize this, fix target conditioning and inference block size, and let τ_d and T_d denote MAT and round latency at depth d, with throughput $G_d = \tau_d/T_d$ and MAT increment $\Delta_d = \tau_{d+1} - \tau_d$. Assume diminishing MAT increments, $\Delta_{d+1} \le \Delta_d$, and linear latency, $T_d = T_{\text{fix}} + d\,T_{\text{layer}}$, where T_{fix} comprises target verification and the depth-independent parts of drafting, and $T_{\text{layer}} > 0$ is the latency added by each drafter layer. Adding one layer then changes throughput by

$$G_{d+1} - G_d = \frac{R_d}{T_d(T_d + T_{\text{layer}})}, \qquad R_d = \Delta_d T_d - T_{\text{layer}}\tau_d.$$

Since $R_{d+1} - R_d = (\Delta_{d+1} - \Delta_d)T_{d+1} \le 0$, once an additional layer no longer increases throughput, further layers cannot increase it either: G_d is unimodal in d (Appendix A.6). Unimodality thus allows the throughput peak at each fixed block size to be located efficiently, e.g., by binary search.

Figure 5 shows the corresponding empirical trend: throughput rises and then falls with drafter depth, with small local fluctuations. The highest measured throughput occurs at $d = 6$ for $bs = 16, 24, 32$, and at $d = 4$ for $bs = 8$.

Block-size selection without retraining. Mismatch between training and inference block sizes can reduce MAT, making configuration search costly if a separate drafter is trained for each candidate inference block size. Mixed-horizon training (Section 3.1.2) addresses this limitation. Table 2 shows that a drafter trained with mixed horizons achieves higher MAT than the fixed-32 model at every tested inference block size and remains close to the fixed-16 model at $bs = 16$. For each candidate drafter depth, the same mixed-horizon checkpoint supports evaluation across candidate inference block sizes, enabling joint selection of depth and block size without retraining for each block size.

4 Experimental Evaluation

We evaluate DScout from two complementary perspectives: how much it accelerates inference, and how its training recipe and inference configuration selection contribute to these improvements.

Benchmarks and baselines. We evaluate on GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021; Lightman et al., 2023) (Math), HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), and LiveCodeBench (Jain et al., 2024) (Code), and MT-Bench (Zheng et al., 2023) (Chat). We compare with EAGLE-3 (Li et al., 2025b), DFlash (Chen et al., 2026), and PARD-2 (An et al., 2026). Each method uses an autoregressive (AR) baseline matched in target model, hardware, runtime, sampling, and concurrency.

Training and implementation. We disable thinking mode for Qwen3-4B and Qwen3-8B. We train DScout, DFlash, EAGLE-3, and PARD-2 on the same 1.4M Open-PerfectBlend (Labonne, 2024) examples with target-generated responses, following each baseline's official training recipe. DScout follows DFlash's parameters (Chen et al., 2026), except Qwen3-8B uses our selected 35t6d configuration (Section 4.2). Training GPUs have 80 GB HBM and 3.35 TB/s bandwidth; main and serving benchmark GPUs have 192 GB HBM3 and 8 TB/s bandwidth. DScout, DFlash, and EAGLE-3 use SGLang (Zheng et al., 2024) with FlashInfer (Ye et al., 2025); PARD-2 uses its official Transformers runtime with a separate matched AR baseline. Training settings and study protocols appear in Table 6 and Section A.

Metrics and configuration selection. All evaluations are repeated five times. Speedup is the ratio of mean output throughput to matched mean AR throughput, computed per dataset and then averaged equally across datasets. Study-specific aggregation procedures are detailed in Appendix A. MAT (τ) is defined in Eq. (2.2). At concurrency above one, throughput measures aggregate service output. We report both $bs = 16$ and $bs = 32$ for DScout.

4.1 End-to-End Performance.

Benchmark performance. DScout improves mean speedup over DFlash on both Qwen3 targets and at both temperatures, reaching up to $4.14\times$ over AR and about 9% over DFlash (Table 3). Longer proposals help when their additional accepted tokens offset the extra draft–verify cost. On Qwen3-8B at temperature 0, this trade-off favors $bs = 32$ on the Math tasks, but $bs = 16$ on MBPP, LiveCodeBench, and MT-Bench, where acceptance gains are smaller. The preferred block size also changes with temperature on Qwen3-4B. Thus, drafting quality varies by workload, and higher MAT alone does not determine the fastest configuration.

Serving across models and concurrency. Table 4 extends the comparison to Llama-3.1-8B (Grattafiori et al., 2024) and Qwen3-Coder-30B-A3B (Qwen Team, 2025), with concurrency $c \in \{1, 8, 16, 32\}$. DScout (16) uses $bs = 16$, while DScout (best) selects one shared block size per target and concurrency by mean speedup across the six datasets; candidate sizes and selection details are given in Appendix A.2. DScout (best) achieves 2.45–$4.14\times$ mean speedup over AR and exceeds native-block-size DFlash in all tested settings. At fixed checkpoints, block-size selection provides up to 9.75% additional mean speedup over DScout (16), quantifying the benefit of adapting inference configuration to the serving load without retraining. DScout (16) already improves on DFlash except for Llama-3.1-8B at $c = 32$, where the selection gain is largest. There, selecting a shorter block recovers the advantage despite reducing MAT; Qwen3-8B at the same concurrency also becomes faster with lower MAT. Fewer accepted tokens per round can therefore yield higher throughput when rounds finish sufficiently faster. This explains why block-size selection must account for both target model and serving load.

Table 3: **Benchmark speedup and MAT on Qwen3 models.** All results use concurrency $c = 1$. Parentheses denote EAGLE-3 tree size or inference block size; † marks the separate PARD-2 runtime.

Model	Method	MATH				CODE						CHAT		Avg.	
		GSM8K		MATH-500		HumanEval		MBPP		LCB		MT-Bench			
Temperature = 0		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Q3-4B	EAGLE-3 (16)	2.07×	6.49	2.07×	6.21	1.88×	5.73	1.83×	5.65	1.75×	5.21	1.34×	4.01	1.82×	5.55
	EAGLE-3 (60)	2.12×	7.12	2.16×	6.95	1.99×	6.51	1.95×	6.46	1.87×	6.00	1.46×	4.70	1.92×	6.29
	DFlash (16)	4.51×	7.99	4.43×	7.54	3.48×	5.98	3.39×	5.87	3.15×	5.33	2.14×	3.62	3.52×	6.06
	PARD-2 (15)†	3.46×	8.36	3.49×	8.11	2.77×	6.47	2.27×	6.02	1.85×	5.21	1.52×	3.78	2.56×	6.32
	DScout (16)	4.75×	8.62	4.72×	8.20	**3.76**×	6.60	**3.61**×	6.40	**3.42**×	5.91	**2.28**×	3.94	3.76×	6.61
	DScout (32)	**5.19**×	10.06	**4.96**×	9.16	3.75×	6.99	3.57×	6.72	3.33×	6.11	2.23×	4.09	**3.84**×	7.19
Q3-8B	EAGLE-3 (16)	2.58×	6.87	2.66×	6.76	2.38×	6.15	2.26×	5.93	2.28×	5.79	1.77×	4.51	2.32×	6.00
	EAGLE-3 (60)	2.56×	7.39	2.67×	7.38	2.43×	6.84	2.38×	6.78	2.40×	6.63	1.90×	5.27	2.39×	6.71
	DFlash (16)	5.01×	8.19	4.86×	7.67	3.87×	6.15	3.55×	5.67	3.56×	5.60	2.36×	3.71	3.87×	6.17
	PARD-2 (15)†	3.61×	8.53	1.75×	7.99	2.77×	6.49	2.77×	5.81	1.96×	5.46	1.75×	3.68	2.44×	6.33
	DScout (16)	5.17×	8.71	5.01×	8.15	4.03×	6.61	**3.61**×	5.97	**3.71**×	6.02	**2.40**×	3.89	3.99×	6.56
	DScout (32)	**5.65**×	10.08	**5.44**×	9.29	**4.25**×	7.30	3.56×	6.15	3.62×	6.15	2.34×	3.96	**4.14**×	7.16
Temperature = 1		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Q3-4B	EAGLE-3 (16)	1.99×	6.09	1.82×	5.36	1.80×	5.37	1.74×	5.26	1.54×	4.51	1.24×	3.65	1.69×	5.04
	EAGLE-3 (60)	2.01×	6.84	1.89×	6.18	1.86×	6.18	1.81×	6.13	1.64×	5.36	1.32×	4.33	1.75×	5.84
	DFlash (16)	4.11×	7.05	3.38×	5.57	3.19×	5.31	2.98×	5.01	2.53×	4.15	1.98×	3.25	3.03×	5.06
	DScout (16)	4.15×	7.21	**3.58**×	5.99	**3.45**×	5.86	**3.26**×	5.59	**2.74**×	4.57	**2.10**×	3.50	**3.21**×	5.45
	DScout (32)	**4.46**×	8.30	3.56×	6.34	3.39×	6.10	3.15×	5.73	2.62×	4.64	2.00×	3.54	3.20×	5.77
Q3-8B	EAGLE-3 (16)	2.34×	6.35	2.20×	5.75	2.07×	5.47	2.01×	5.40	1.85×	4.82	1.49×	3.90	1.99×	5.28
	EAGLE-3 (60)	2.40×	7.06	2.31×	6.53	2.19×	6.27	2.16×	6.29	2.02×	5.69	1.64×	4.65	2.12×	6.08
	DFlash (16)	4.08×	6.67	3.44×	5.47	3.13×	5.01	2.99×	4.82	2.54×	4.03	**1.94**×	3.08	3.02×	4.85
	DScout (16)	4.13×	7.11	3.49×	5.81	3.23×	5.44	**3.02**×	5.11	**2.60**×	4.33	1.92×	3.20	3.06×	5.17
	DScout (32)	**4.41**×	7.84	**3.61**×	6.21	**3.29**×	5.70	2.99×	5.22	2.54×	4.36	1.90×	3.26	**3.12**×	5.43

Table 4: **Serving speedup and MAT across target models and concurrency levels** at temp = 0.

Model	Method	Con.1		Con.8		Con.16		Con.32	
		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Qwen3-4B	DFlash (16)	3.51×	6.06	2.87×	6.06	2.57×	6.08	2.30×	6.06
	DScout (16)	3.75×	6.61	**3.08**×	6.59	**2.70**×	6.60	**2.45**×	6.63
	DScout (best)	**3.83**×	7.19	**3.08**×	6.59	**2.70**×	6.60	**2.45**×	6.63
Qwen3-8B	DFlash (16)	3.87×	6.17	3.16×	6.13	2.74×	6.16	2.38×	6.17
	DScout (16)	3.99×	6.56	**3.21**×	6.59	**2.79**×	6.59	2.42×	6.58
	DScout (best)	**4.14**×	7.16	**3.21**×	6.59	**2.79**×	6.59	**2.46**×	6.07
Qwen3-Coder-30B-A3B-Instruct	DFlash (16)	2.58×	6.41	2.59×	6.41	2.76×	6.43	2.91×	6.43
	DScout (16)	2.65×	6.62	2.62×	6.58	2.82×	6.63	2.95×	6.67
	DScout (best)	**2.70**×	7.05	**2.68**×	7.10	**2.88**×	6.46	**2.98**×	6.10
Llama-3.1-8B-Instruct	DFlash (10)	3.37×	5.50	2.99×	5.47	2.80×	5.47	2.52×	5.48
	DScout (16)	3.75×	6.20	**3.17**×	6.14	**2.99**×	6.17	2.36×	6.18
	DScout (best)	**3.76**×	6.36	**3.17**×	6.14	**2.99**×	6.17	**2.59**×	5.56

4.2 DESIGN ANALYSIS

Training-recipe ablation. Following Section 3.1, we progressively add mixed-horizon training, intra-block unmasking, and AWL to the training recipe. Each configuration is trained independently to produce a separate checkpoint. At a common inference block size of $bs = 16$, the cumulative training ablation improves MAT by 16.1% (Figure 4). Every added strategy contributes, with AWL producing the largest increment after mixed horizons and intra-block unmasking. These gains require no additional inference-time operations; the increments are conditional on the preceding recipe rather than independent effects. The system-level comparison in Appendix Figure 7 shows higher acceptance at every position, with larger gains later in the block.

Drafter-depth selection. Figure 5 shows throughput increasing and then declining with drafter depth, with small local fluctuations. The highest measured throughput occurs at $d = 6$ for $bs = 16, 24, 32$, but at $d = 4$ for $bs = 8$, showing that the preferred depth depends on inference block size. The unimodality result applies under its stated assumptions; local measurement fluctuations need not



Figure 4: **Cumulative recipe ablation (9t9d).** Labels show MAT increments from the preceding step.

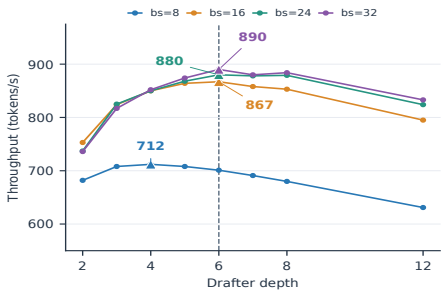

Figure 5: **Throughput versus drafter depth** on Qwen3-8B. Markers highlight observed maxima.

satisfy it exactly. Together with the target-conditioning analysis in Section 3.2, these results motivate our 35t6d configuration for Qwen3-8B, with inference block size selected for each concurrency setting.

5 Related Work

Autoregressive and parallel drafters. Early speculative decoding methods generate proposals autoregressively (Leviathan et al., 2023; Chen et al., 2023). The EAGLE family uses target-model features for drafting, with EAGLE-3 combining multi-layer features and direct token prediction (Li et al., 2024a;b; 2025b). Other approaches reduce sequential drafting through multiple prediction heads, as in Medusa and Hydra (Cai et al., 2024; Ankner et al., 2024), or parallel masked-token prediction, as in PARD (An et al., 2025).

Block-diffusion drafters. Diffusion language models support parallel token prediction (Arriola et al., 2025; Nie et al., 2025) and have been explored for speculative drafting in DiffuSpec, SpecDiff-2, and DART (Li et al., 2025a; Sandler et al., 2026; Liu et al., 2026). DFlash generates proposals in a single pass using a target-conditioned block-diffusion drafter (Chen et al., 2026). Building on this architecture, DScout studies how training supervision and inference configuration together determine decoding efficiency. Its two factorizations guide supervision toward accepted prefixes and configuration selection according to the MAT–latency trade-off.

Serving efficiency and inference configuration. vLLM and SGLang improve serving efficiency through memory management and scheduling (Kwon et al., 2023; Zheng et al., 2024), while SpecInfer supports efficient token-tree verification (Miao et al., 2023). Complementary to these systems, DScout selects target-conditioning depth, drafter depth, and inference block size based on their MAT–latency trade-offs. Mixed-horizon training further allows inference block size to be selected for different workloads and concurrency settings without retraining the checkpoint.

6 Discussion and Conclusion

Our current inference selection uses a shared block size for each target model and concurrency setting. A natural next step is to adapt block size online as acceptance and serving load change. Mixed-horizon checkpoints support this flexibility without retraining. An online policy could use recent acceptance statistics and measured round latency to adjust block size within a generation according to the MAT–latency trade-off. Evaluating such a policy under changing workloads, while accounting for adaptation overhead, would establish whether it yields sustained gains over fixed configurations.

Overall, DScout provides a systematic approach to translating diffusion-drafter prediction quality into practical decoding acceleration. For training, a position-wise loss factorization guides strategy design and combination. For inference, the MAT–latency factorization guides configuration selection. The resulting training recipe and inference configurations improve end-to-end performance across models, tasks, and serving loads. These factorizations can also inform future diffusion-drafter research by providing a framework for training strategy design and inference configuration selection.

Ethical Statement

This work proposes DScout to improve speculative decoding efficiency through diffusion-drafter training and inference configuration selection. Following the ICLR Code of Ethics[1], we acknowledge that faster generation may also lower the cost of producing harmful content at scale. DScout does not address biases, factual errors, or unsafe behaviors of the underlying target models. Our evaluation uses existing datasets and target-generated training responses, and focuses on decoding efficiency rather than safety or downstream deployment.

Reproducibility statement

Model, dataset, baseline, and runtime configurations are described in Section 4. All evaluations are repeated five times; aggregation procedures, serving-speedup variability, and study-specific protocols are reported in Appendix A. Training pseudocode and hyperparameters are provided in Algorithm 1 and Table 6. Inference block-size candidates and the selection criterion are detailed in Appendix A.2. The assumptions and proof of conditional unimodality over drafter depth appear in Appendix A.6.

AI use statement

We used generative AI tools to assist with literature searches, code editing and execution, preparation of tables and figures, manuscript drafting and language polishing, and discussions of methodology. Target-model-generated responses were also used to construct the target-aligned training data described in Section 4. The authors reviewed the AI-assisted material and take responsibility for the final text, code, analyses, and claims.

References

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

Zihao An, Taichi Liu, Ziqiong Liu, Dong Li, Ruofeng Liu, and Emad Barsoum. PARD-2: Target-aligned parallel draft model for dual-mode speculative decoding. *arXiv preprint arXiv:2605.08632*, 2026. URL https://arxiv.org/abs/2605.08632.

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for Medusa decoding. *arXiv preprint arXiv:2402.05109*, 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. In *International Conference on Learning Representations*, 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. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

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. *arXiv preprint arXiv:2401.10774*, 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. *arXiv preprint arXiv:2302.01318*, 2023. URL https://arxiv.org/abs/2302.01318.

[1] https://iclr.cc/public/CodeOfEthics

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.06036. arXiv:2602.06036.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

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. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024. URL https://arxiv.org/abs/2403.07974.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023. URL https://arxiv.org/abs/2309.06180.

Maxime Labonne. Open-PerfectBlend. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/mlabonne/open-perfectblend. Accessed 2026-09-17.

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

Guanghao Li, Zhihui Fu, Min Fang, Qibin Zhao, Ming Tang, Chun Yuan, and Jun Wang. DiffuSpec: Unlocking diffusion language models for speculative decoding. *arXiv preprint arXiv:2510.02358*, 2025a. URL https://arxiv.org/abs/2510.02358.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 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. *arXiv preprint arXiv:2406.16858*, 2024b. URL https://arxiv.org/abs/2406.16858.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025b. URL https://arxiv.org/abs/2503.01840.

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. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Fuliang Liu, Xue Li, Ketai Zhao, Yinxi Gao, Ziyan Zhou, Zhonghui Zhang, Zhibin Wang, Wanchun Dou, Sheng Zhong, and Chen Tian. DART: Diffusion-inspired speculative decoding for fast LLM inference. *arXiv preprint arXiv:2601.19278*, 2026. URL https://arxiv.org/abs/2601.19278.

Xiaoxuan Liu, Cade Daniel, Langxiang Hu, Woosuk Kwon, Zhuohan Li, Xiangxi Mo, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Optimizing speculative decoding for serving large language models using goodput. *arXiv preprint arXiv:2406.14066v2*, 2024. URL https://arxiv.org/abs/2406.14066v2.

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 generative large language model serving with tree-based speculative inference and verification. *arXiv preprint arXiv:2305.09781*, 2023. URL https://arxiv.org/abs/2305.09781.

Jinjie Ni, Qian Liu, Longxu Dou, Chao Du, Zili Wang, Hang Yan, Tianyu Pang, and Michael Qizhe Shieh. Diffusion language models are super data learners. *arXiv preprint arXiv:2511.03276*, 2025. URL https://arxiv.org/abs/2511.03276.

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. *arXiv preprint arXiv:2502.09992*, 2025. URL https://arxiv.org/abs/2502.09992.

Qwen Team. Qwen3-Coder-30B-A3B-Instruct. Hugging Face model card, 2025. URL https://huggingface.co/Qwen/Qwen3-Coder-30B-A3B-Instruct. Accessed 2026-09-17.

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://arxiv.org/abs/2511.00606.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. FlashInfer: Efficient and customizable attention engine for LLM inference serving. *arXiv preprint arXiv:2501.01005*, 2025. URL https://arxiv.org/abs/2501.01005.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685*, 2023. URL https://arxiv.org/abs/2306.05685.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2312.07104.

Supplementary Material

A Evaluation Details and Supplementary Results **13**
A.1 Additional Setup and Aggregation 13
A.2 Serving Configuration Grid 13
A.3 Acceptance and Depth Study Protocols 13
A.4 Target-Conditioning Cost 15
A.5 Controlled Depth–Latency Profiling 15
A.6 Conditional Unimodality over Drafter Depth 16

B DScout algorithm **17**

C Training hyperparameters **17**

A Evaluation Details and Supplementary Results

A.1 Additional Setup and Aggregation

Evaluation protocol. We use the six evaluation tasks defined in Section 4, with 128 prompts per task except MT-Bench (80), dataset seed 0, an 8192-token context limit, and a 2048-token generation cap. Sampling uses top-$p = 1$, with top-$k = 1$ at temperature 0 and no top-k truncation at temperature 1; the corresponding sampling seeds are 1000 and 2000. SGLang runs use 32 warm-up prompts; PARD-2 uses one. DScout, DFlash and EAGLE-3 are trained for six epochs, and PARD-2 for two. These are complete-system comparisons with unequal training budgets. Each comparison uses its matched AR baseline; speedups across the SGLang and PARD-2 runtimes do not establish an absolute throughput ranking. Training-control studies and system benchmarks use separate platforms, and their latencies are not pooled.

Run variability. Point estimates follow the task-level ratio-of-means definition in Section 4. For repeat variability, we instead compute one macro speedup per repeat by pairing candidate and AR runs with the same repeat index, then report the sample SD over five repeats. This measures variability on fixed prompts, not uncertainty over unseen tasks. Mechanism studies with different aggregation are specified below.

A.2 Serving Configuration Grid

We evaluate $bs \in \{8, 10, 12, 14, 16, 24, 32\}$ for each target and concurrency setting.

Figure 6 records every block-size candidate underlying DScout (best) in Table 4. Each target–concurrency setting keeps its checkpoint fixed and selects one block size by maximizing equal-weight mean speedup across all six tasks. Thus, these results evaluate block-size selection at fixed (t, d), not a joint optimum over all three dimensions. Selection uses the evaluation tasks themselves, without combining per-task optima. Table 5 reports repeat variability; entries are DFlash / DScout (best).

Table 5: Serving speedup: sample SD over five repeats.

Target	$c = 1$	$c = 8$	$c = 16$	$c = 32$
Qwen3-4B	.004 / .011	.043 / .045	.116 / .127	.052 / .028
Qwen3-8B	.004 / .017	.013 / .010	.072 / .064	.035 / .036
Qwen3-Coder-30B-A3B	.002 / .004	.023 / .013	.025 / .022	.042 / .033
Llama-3.1-8B	.004 / .003	.023 / .038	.112 / .095	.208 / .161

A.3 Acceptance and Depth Study Protocols

Training controls. The training-horizon controls use Qwen3-8B hidden-9t9d checkpoints trained for six epochs. Fixed-16 uses positional decay $\gamma = 7$; fixed-32 and mixed horizons use $\gamma = 15$.

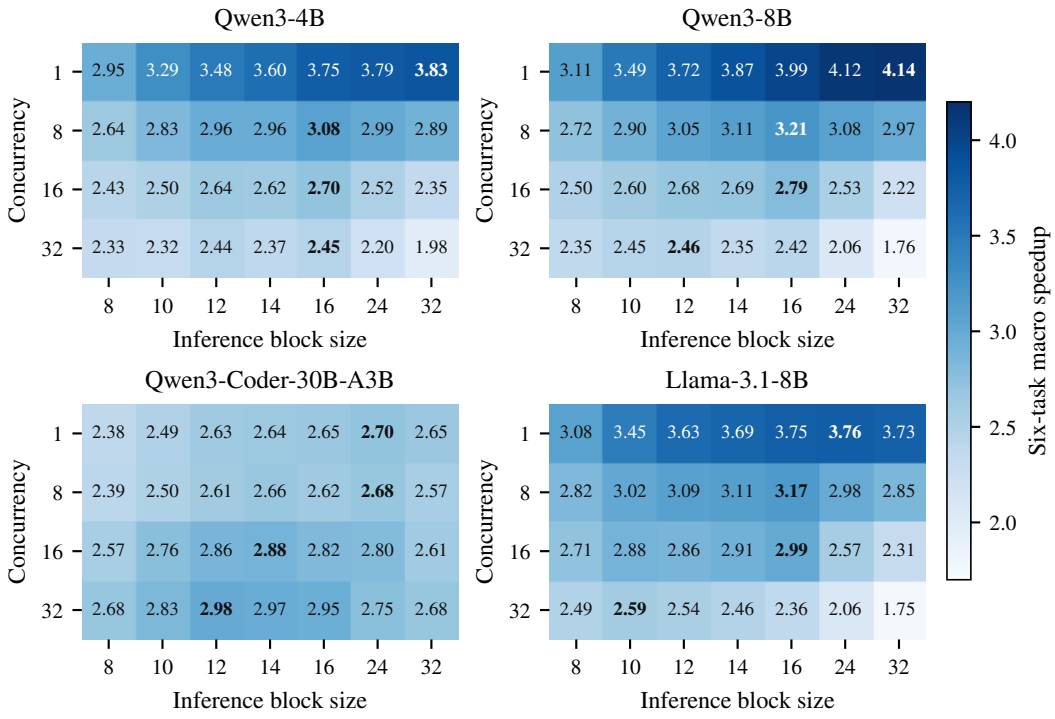

Figure 6: **Full serving grid.** Cells show six-task mean speedup; bold entries mark the selected block size in each row.

Mixed horizons sample $B \in \{8, 16, 24, 32\}$ equally. The distribution in Figure 2 averages dataset-normalized histograms over the same six tasks as the system benchmarks.

Position-wise acceptance. Figure 7 compares final DScout (35t6d) with DFlash on Qwen3-8B. Let $S(k)$ be the probability that the first k draft tokens survive verification, with $S(0) = 1$. We compute conditional acceptance as $S(k)/S(k-1)$ from each dataset's accepted-length histogram in each run, then average equally over five repeats and datasets within each domain. Conditioning on surviving prefixes means these probabilities need not decrease with position. The comparison evaluates complete systems and does not isolate architecture, training-recipe, or training-budget effects.

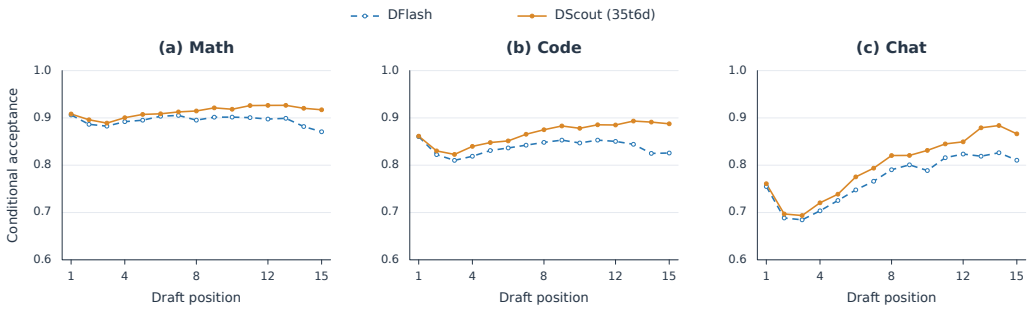

Figure 7: **Position-wise conditional acceptance** of DScout (35t6d) and DFlash at $bs = 16$, $c = 8$, and temperature 0.

Drafter depth. Figure 3(b) and Figure 5 use the Qwen3-8B 35t hidden-conditioned family trained for six epochs, with $c = 1$, $d \in \{2, 3, 4, 5, 6, 7, 8, 12\}$, and $bs \in \{8, 16, 24, 32\}$. For each dataset, we discard the minimum and maximum of five repeats and average the middle three, then average

equally across the six datasets. Figure 5 reports throughput in tokens per second (TPS); repeat-level dispersion is unavailable for these aggregates, so no error bars are shown. Highlighted maxima identify the best measured candidates, without implying statistically significant differences.

A.4 Target-Conditioning Cost

Measured throughput. Figure 3(a) sweeps $t = 1, \ldots, 35$ at fixed $d = 6$, $bs = 16$, and $c = 1$, with five repeats per setting. Acceptance is held fixed at 16 tokens per round to isolate execution cost from prediction quality. Let $q(t)$ be trimmed-mean throughput and $s(t)$ the across-repeat SD. Points show $100[q(t)/q(1) - 1]$, and bars show $\pm 100 s(t)/q(1)$ percentage points. The reference mean is fixed, so the bars represent run variability rather than confidence intervals for the ratio. These measurements capture end-to-end throughput, not isolated feature-fusion latency.

Illustrative arithmetic cost. For concatenation followed by a linear projection, increasing conditioning from 9 to 35 layers adds

$$\Delta F_{\text{fusion}} = 2(35 - 9)H^2 A, \qquad F_{\text{target,dense}} = 2L_T\, bs\, (2H^2 + 2HK + 3HI),$$

where A is the number of newly fused context tokens and a multiply-add counts as two FLOPs. The target term counts target projection and MLP FLOPs. Here L_T is the target layer count, I is the MLP intermediate width, and K is the total key/value projection width. We use $H_T = H_D = H$ for equal target and drafter hidden widths. Using Qwen3-8B's $L_T = 36$, $H = 4096$, $I = 12288$, and $K = 1024$,[2] with $bs = 16$ and $A \approx 7$ corresponding to MAT near seven when only newly accepted context tokens are fused, these costs are 6.11 and 222.26 GFLOPs, respectively (2.75%); fusing all 16 positions gives 6.28%. This cached-context calculation excludes prefill, attention matrix products, the output head, drafter operations, and memory traffic. It illustrates arithmetic scale, not a measured fusion profile or a bound on latency; feature transfer and kernel behavior remain relevant.

A.5 Controlled Depth–Latency Profiling

To examine execution cost independently of prediction quality, we profile drafter depths $d = 1, \ldots, 35$ with a fixed acceptance length. Let q_d denote the measured throughput. Under this fixed-acceptance protocol, relative round latency is $T_d/T_1 = q_1/q_d$. Figure 8 plots this ratio with an unconstrained least-squares affine fit, $T_d/T_1 \approx 0.9777 + 0.0190d$, yielding $R^2 = 0.9975$. The maximum relative deviation from the fitted line is 1.86%. These measurements support approximately linear latency over the profiled depth range, complementing the MAT increments and end-to-end throughput in Figure 3(b) and Figure 5.

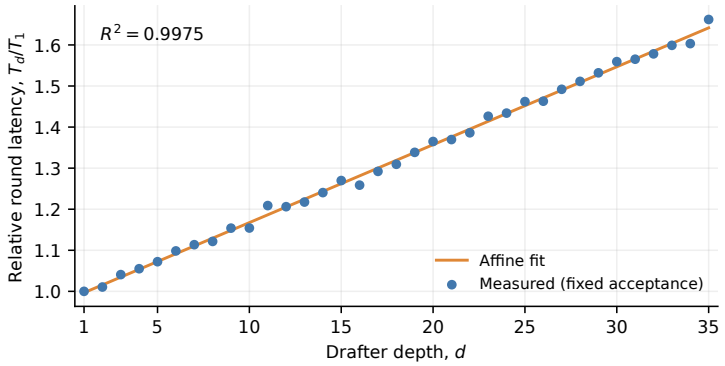

Figure 8: **Depth–latency profile.** Measured relative round latency and affine fit.

[2]Model dimensions: Qwen3-8B configuration.

A.6 Conditional Unimodality over Drafter Depth

Fix target conditioning, inference block size, workload, and execution conditions. On a finite interval of consecutive drafter depths, let $\tau_d > 0$ be MAT, $T_d > 0$ be mean round latency, and $\Delta_d = \tau_{d+1} - \tau_d$. Write the per-layer latency increment as $b_d = T_{d+1} - T_d > 0$. Assume diminishing MAT increments, $\Delta_{d+1} \leq \Delta_d$, and non-decreasing latency increments, $b_{d+1} \geq b_d$. Then throughput $G_d = \tau_d / T_d$ is unimodal in the sense that it is non-decreasing up to some depth and non-increasing thereafter (equivalently, quasi-concave in d): its increments cannot become positive after becoming nonpositive. Plateaus and boundary optima are permitted; MAT need not be monotonically increasing. The linear latency model in the main text is the special case $b_d = T_{\text{layer}}$.

Proof. Since $T_d > 0$ and $T_{d+1} > 0$,

$$G_{d+1} - G_d = \frac{R_d}{T_d T_{d+1}}, \qquad R_d = \Delta_d T_d - b_d \tau_d,$$

so the sign of each throughput increment is the sign of R_d. Using $\tau_{d+1} = \tau_d + \Delta_d$ and $T_{d+1} = T_d + b_d$ gives

$$R_{d+1} - R_d = (\Delta_{d+1} - \Delta_d) T_{d+1} - (b_{d+1} - b_d) \tau_{d+1} \leq 0,$$

where the first term is nonpositive by diminishing MAT increments and the second is nonpositive because latency increments do not decrease and $\tau_{d+1} > 0$. Thus R_d is non-increasing, and once $R_d \leq 0$, every subsequent throughput increment is nonpositive. This proves the claim. Under linear latency, the second term vanishes, so the sign change is driven by the diminishing MAT increments alone.

Relation to the measurements. Figure 3(b) motivates diminishing returns over the consecutive measured depths 2–8. Its final row reports the total gain from depth 8 to 12, not a one-layer increment, and does not resolve the unmeasured intermediate depths. Figure 5 shows throughput rising and then falling with small local fluctuations; it does not by itself establish exact unimodality. The separate fixed-acceptance profile in Appendix A.5 supports approximately linear latency. The proposition provides a search rationale under explicit assumptions, rather than an unconditional guarantee for the measured serving curves. It applies separately at each fixed block size: it does not establish unimodality of the envelope $\max_{bs} G(d, bs)$ or a global optimum over all configurations.

B DSCOUT ALGORITHM

Algorithm 1 DScout training

Require: Target-aligned dataset $\mathcal{D}$ and frozen target p_T
Require: Drafter q_θ and context adapter g_ϕ
for each minibatch $(x, y) \sim \mathcal{D}$ **do**
  Choose $B \in \{8, 16, 24, 32\}$ with equal frequency across updates and valid anchors $\mathcal{A}$
  $(H_{\mathcal{S}}, P) \leftarrow p_T(x, y)$ *target frozen*
  $C \leftarrow g_\phi(H_{\mathcal{S}})$
  $U \leftarrow 0; \quad Z \leftarrow 0$ *weighted-loss and weight accumulators*
  for each anchor $a \in \mathcal{A}$ **do**
    $Y \leftarrow (y_a, \ldots, y_{a+B-1})$
    $M \leftarrow \mathrm{SampleMask}(B)$
    $\widetilde{Y} \leftarrow \mathrm{Corrupt}(Y, M)$
    $Q_{1:B-1} \leftarrow q_\theta(\widetilde{Y}, C_a)$
    for $k = 1, \ldots, B-1$ **do**
      $\widehat{y}_k \leftarrow \arg\max_v Q_k(v)$
      $\alpha_k \leftarrow \min\left\{1, \dfrac{P_k(\widehat{y}_k)}{Q_k(\widehat{y}_k)}\right\}$
      $w_k \leftarrow \mathrm{sg}\left(\prod_{j<k} \alpha_j\right)$
    end for
    $U \leftarrow U + \sum_{k \in M} w_k \, \mathrm{CE}(Q_k, y_{a+k})$
    $Z \leftarrow Z + \sum_{k \in M} w_k$
  end for
  $\mathcal{L}_{\mathrm{train}} \leftarrow U/(Z + \epsilon)$
  Update θ and ϕ using $\nabla \mathcal{L}_{\mathrm{train}}$
end for

C TRAINING HYPERPARAMETERS

Table 6: DScout training configurations.

Configuration	Value
Target-aligned data	
Dataset	Open-PerfectBlend (Labonne, 2024)
Number of samples	1.4M (`1m4`)
Rollout temperature	0
Coverage-oriented training	
Anchors per sequence	512
Training horizon	$B \in \{8, 16, 24, 32\}$, equal frequency
Full-mask probability	0.4
Diffusion mask distribution	$\rho \sim \mathrm{Beta}(2, 2)$
Minimum mask fraction	0.25
Position weighting and normalization	
Position weighting	AWL
Weight normalization	Weighted mean
Diffusion-loss normalization	Weighted mean
Optimization	
Training epochs	6
Optimizer	AdamW
Learning rate	6×10^{-4}
Warmup ratio	0.04
Gradient clipping	1.0
Global batch size	64