Compass: REPAIRING WHEN TARGET-MODEL GUIDANCE HURTS IN PARALLEL SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Target-model features are increasingly used to improve parallel speculative drafting, but stronger guidance does not necessarily translate into faster decoding. We reveal that target-model guidance can hurt it in two ways: feature injection adds drafting overhead that can limit the speed benefits of improved acceptance, while target-guided adaptation can introduce new errors in prefixes already accepted by the original drafter, undermining the benefit of target-model guidance. We introduce *Compass*, a lightweight target-guided parallel speculative decoding framework that combines compact layerwise target guidance with reference-aware guidance optimization. The guidance module compresses multi-layer target states into a shared low-dimensional source and injects separate gate and up input shifts into each draft MLP, reusing native projections to limit drafting overhead. The optimization module uses the initial drafter as a frozen reference model to protect previously accepted prefixes and encourage recovery beyond its acceptance boundary. We construct a margin-based prefix objective aligned with greedy verification and derive a per-block lower bound linking the designed loss to acceptance gain over the reference model. Implemented in vLLM, *Compass* achieves mean AR-relative speedups of 5.27×, 4.25×, and 2.39× on Qwen3-8B, Qwen3-14B, and Qwen3-32B across six benchmarks, exceeding the strongest available baseline for each target by 7.0%, 8.1%, and 12.4%. Ablations show that compact guidance retains acceptance benefits at lower drafting cost, while reference-aware optimization extends accepted prefixes and accelerates decoding.

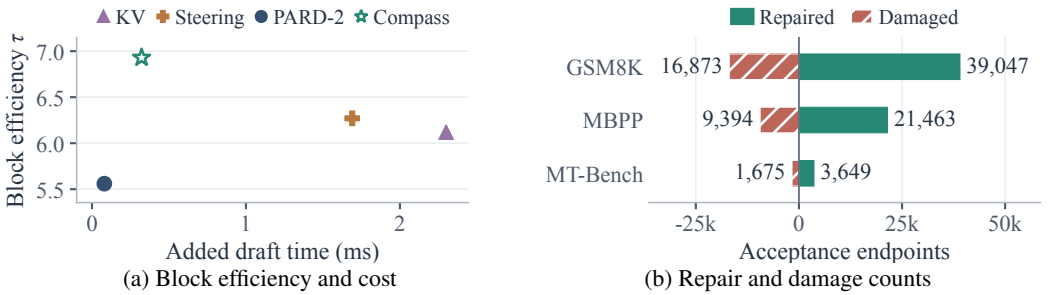
1 INTRODUCTION

Autoregressive generation remains a major source of latency in large language model (LLM) inference, as producing each new token requires another sequential model evaluation (DeepSeek-AI, 2025; Chen et al., 2021; Gemini Team, 2024; Pope et al., 2023). Speculative decoding reduces this overhead by using a lightweight drafter to propose multiple tokens that the target model verifies together (Leviathan et al., 2023; Chen et al., 2023; Miao et al., 2024; Li et al., 2024b). Its practical benefit depends on both the cost of each draft–verify cycle and the number of tokens committed (Chen et al., 2024). An effective speculative decoder must therefore keep proposal generation inexpensive while producing sufficiently long accepted prefixes.

Parallel drafting reduces proposal latency by predicting an entire token block in a single backbone pass rather than invoking the drafter repeatedly (An et al., 2026a; Chen et al., 2026; Stern et al., 2018; Cai et al., 2024). However, low drafting cost does not by itself ensure that the proposals align with the target model. To improve the quality of these parallel proposals, recent methods increasingly condition the drafter on hidden states from the target model (Li et al., 2025; Berdoz et al., 2025; An et al., 2026b). While such features can improve proposal alignment, stronger guidance does not necessarily yield faster decoding: *target guidance can hurt by increasing drafting latency and damaging the accepted prefix already achieved by the original drafter*.

First, target guidance introduces drafting overhead that can limit its speed benefits. Input-level fusion is inexpensive, but deeper draft layers access target guidance only indirectly through representations propagated from the input. KV-based attention conditioning and MLP steering provide direct layerwise access, but introduce additional computation. Figure 1(a) shows that input-level fusion incurs little overhead but achieves lower block efficiency, whereas KV-based and steering guidance achieve

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.



(a) Block efficiency and cost (b) Repair and damage counts

Figure 1: Guidance cost and acceptance regressions with Qwen3-14B / PARD-Qwen3-0.6B. (a) Mean block efficiency τ over GSM8K, MBPP, and MT-Bench versus added draft time relative to unguided PARD (2.03 ms). (b) Repaired and damaged prefix endpoints on the same datasets, comparing original PARD-TI with guidance-only adaptation of a frozen draft backbone.

higher block efficiency at substantially greater drafting cost. Thus, the additional computation required for layerwise guidance can undermine the speed benefits of improved acceptance.

Second, target-guided adaptation can destroy acceptance that the original drafter already achieves. Figure 1(b) shows 64,159 repaired but also 27,942 damaged prefix endpoints across GSM8K, MBPP, and MT-Bench, with damage offsetting 43.6% of the repair gains. This matters because an early new error invalidates all longer prefixes, while correcting a later token cannot extend acceptance if an earlier mismatch remains. Token-level distillation does not explicitly distinguish predictions that should be preserved from those that still need correction. Together, these challenges raise a core design question: *How can we provide low-cost target guidance across draft layers and train the drafter to extend accepted prefixes while preserving existing ones?*

To address this question, we introduce *Compass*, a lightweight target-guided parallel speculative decoding framework which combines Compact Layerwise Target Guidance with Reference-Aware Guidance Optimization. The guidance module compresses multi-layer target states into a shared low-dimensional source and injects gate and up shifts into each draft MLP while reusing the drafter's native projections, providing low-cost layerwise guidance without an additional target forward pass. The optimization module uses the initial drafter as a frozen reference model to protect previously accepted prefixes and encourage recovery beyond its acceptance boundary. Its margin-based prefix objective is aligned with greedy verification, and we derive a per-block lower bound connecting the designed loss to acceptance gain. The reference model is used only during training and adds no inference stage.

We implement *Compass* in vLLM and evaluate it on Qwen3-8B, Qwen3-14B, and Qwen3-32B across six benchmarks. *Compass* achieves the highest speedup and block efficiency across all 18 target–benchmark pairs, with mean AR-relative speedups of $5.27\times$, $4.25\times$, and $2.39\times$, exceeding the strongest baselines by 7.0%, 8.1%, and 12.4%, respectively. Its throughput advantage persists across batch sizes, while ablations confirm complementary gains from two modules.

Our contributions are as follows.

- We reveal that target-model guidance can hurt it in two ways: feature injection adds drafting overhead that can limit the speed benefits of improved acceptance, while target-guided adaptation can introduce new errors in prefixes already accepted by the original drafter, undermining the benefit of target-model guidance.
- We propose *Compass*, combining Compact Layerwise Target Guidance for low-cost layerwise guidance with Reference-Aware Guidance Optimization for reference-aware prefix optimization, and derive a lower bound linking the loss to acceptance gain.
- We implement *Compass* in vLLM and achieve mean AR-relative speedups of $5.27\times$, $4.25\times$, and $2.39\times$ on Qwen3-8B, Qwen3-14B, and Qwen3-32B across six benchmarks, with consistent gains across batch sizes and component ablations.

2 PRELIMINARIES

2.1 SPECULATIVE DECODING

Let T denote the target model and D_θ the draft model. Given a confirmed prefix $x_{\le a}$, the drafter proposes K tokens $\hat{y}_{1:K}$, which the target evaluates together in one causal forward pass (Leviathan et al., 2023; Chen et al., 2023). The verifier commits an accepted prefix and supplies a correction token at the first rejection, or a bonus token if the whole block is accepted.

For the acceptance analysis, let $g_i = \arg\max_v p_T(v \mid x_{\le a}, g_{<i})$ denote the target's greedy continuation, with a fixed rule for resolving ties. Define H_j to indicate whether the first j proposals match this continuation:

$$H_j = \prod_{i=1}^{j} \mathbf{1}[\hat{y}_i = g_i], \qquad A = \sum_{j=1}^{K} H_j. \tag{1}$$

Here A is the accepted-prefix length, and $H_j = 1$ means that acceptance reaches endpoint j. A correct token after an earlier mismatch cannot extend the accepted prefix. A full round that is not truncated by a stopping condition commits $A + 1$ tokens, including the correction or bonus. Its cost per committed token is therefore the draft–verify cycle cost divided by $A + 1$, linking decoding efficiency to both drafting cost and accepted-prefix length.

2.2 PRETRAINED PARALLEL DRAFTING

A parallel drafter predicts all K proposals in one backbone pass. We use the anchor–query block construction, retaining its pretrained initialization (An et al., 2026a). The last confirmed token x_a, called the anchor, is followed by $K - 1$ latent queries $q_{2:K}$. These queries are placeholder inputs for predicting later tokens, rather than previously generated proposals.

The draft backbone produces final hidden states $U_{1:K} = f_\theta(x_{\le a}, q_{2:K})$. The anchor state U_1 predicts token $a + 1$, and query state U_i predicts token $a + i$. A vocabulary projection shared across these positions gives $z_i = W_{\mathrm{LM}} U_i$ and $\hat{y}_i = \arg\max_v z_i(v)$. Each query can attend to the confirmed context and preceding query states, but not future target tokens. Thus, later proposal positions can use earlier query states without sequentially feeding generated tokens back into the drafter.

3 METHOD

Compass is designed to limit guidance overhead while preserving and extending accepted prefixes (Figure 2). The frozen target T provides multilayer features and training supervision, and verifies proposals at inference. In our framework, D_θ denotes the trainable guided drafter, with θ covering all draft-model weights and added guidance parameters. We retain a frozen copy of the initial parallel drafter, D_{θ_0}, as the reference model to identify prefixes accepted before target-guided adaptation.

Specifically, *Compact Layerwise Target Guidance* compresses multilayer target features into a shared low-dimensional source and supplies it to every draft MLP through layer-specific gate and up input shifts. *Reference-Aware Guidance Optimization* uses the reference's acceptance outcomes on the same block and target continuation to assign protection losses to endpoints reached by the reference model and repair losses to the remaining endpoints. These losses train the guided drafter to preserve existing acceptance and recover additional accepted prefixes. The reference model is used only during training; inference uses D_θ and T, retaining a single parallel draft pass.

3.1 COMPACT LAYERWISE TARGET GUIDANCE

We add target features to the gate and up inputs of every draft MLP. We compress cached target hidden states into one low-dimensional vector shared across draft layers. Each layer uses two separate linear maps to turn this vector into a gate input shift and an up input shift. Both shifts have the same dimension as the draft hidden states and are added before the existing gate and up projections. This lets the guidance module reuse the drafter's original weights for expansion, instead of learning additional target-feature projections to the larger MLP intermediate dimension.

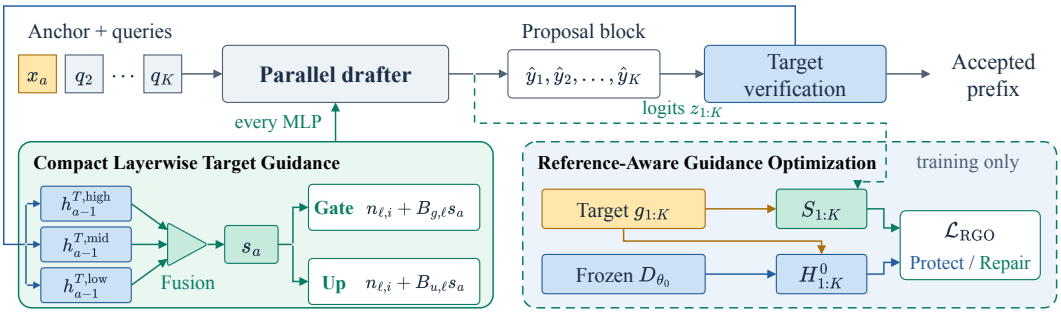


Figure 2: Overview of *Compass*. Compact Layerwise Target Guidance supplies gate and up shift with low additional cost. Reference-Aware Guidance Optimization combines guided scores $S_{1:K}$ and reference outcomes $H^0_{1:K}$ for the same $g_{1:K}$ to protect and repair prefixes.

For the confirmed anchor x_a, we collect hidden states from shallow, middle, and deep target layers, following the multilevel feature fusion of EAGLE-3 (Li et al., 2025). We RMS-normalize (Zhang & Sennrich, 2019) all three states separately that come from position $a-1$, concatenate the results, and apply a shared bias-free linear map to form an r-dimensional target features:

$$G_a = [\mathrm{RMS}(h_{a-1}^{T,\text{low}}); \mathrm{RMS}(h_{a-1}^{T,\text{mid}}); \mathrm{RMS}(h_{a-1}^{T,\text{high}})], \qquad s_a = PG_a. \tag{2}$$

Here $P \in \mathbb{R}^{r \times 3d_T}$, where d_T is the target hidden width. The anchor and all its latent queries share s_a across draft layers when predicting the next K tokens. The source therefore requires no additional target forward pass and is computed once for the block.

Conditional gated architectures allow external information to affect both channel activations and their multiplicative gates (van den Oord et al., 2016a;b). By adding a separate shift to each branch input, we let target features adjust both the channel values and the gates that scale them, without adding a new gating network. Each layer ℓ learns two bias-free maps $B_{g,\ell}, B_{u,\ell} \in \mathbb{R}^{d_D \times r}$, where d_D is the draft hidden width. Let $n_{\ell,i}$ denote the normalized MLP input after attention and its residual addition at block position i. We add the shifts before the existing gate and up projections:

$$\mathrm{MLP}_\ell(n_{\ell,i}, s_a) = W_{d,\ell}[\mathrm{SiLU}(W_{g,\ell}(n_{\ell,i} + B_{g,\ell} s_a)) \odot W_{u,\ell}(n_{\ell,i} + B_{u,\ell} s_a)]\,. \tag{3}$$

In Equation 3, the up shift changes channel activations, while the gate shift changes their multiplicative modulation. The native down projection $W_{d,\ell}$ and residual connection are retained. By generating both shifts in draft width d_D and reusing the native $W_{g,\ell}$ and $W_{u,\ell}$ for expansion, the guidance interface adds $2rd_D$ parameters per layer, rather than $2rm$ for separate mappings from the same r-dimensional source into both expanded branches of width m. Section 4.5 evaluates the effect of this compact parameterization on complete draft and cycle latency. Each layer's two shifts are computed once and reused across the block, while their interaction with different hidden states $n_{\ell,i}$ allows positions to respond differently to the shared source.

We initialize both branch maps to zero and P to a nonzero value, so the guided drafter starts from the original PARD. The guidance module determines how target information can modify the draft logits, but access to stronger features does not specify which modifications improve acceptance.

3.2 Reference-Aware Guidance Optimization

As Figure 1(b) shows, target-guided training can repair draft errors while introducing new errors that shorten previously accepted prefixes, reducing the acceptance gains. To address this, we introduce separate protection and repair losses relative to the reference model, encouraging the draft model to use target features to correct errors while preserving prefixes that were already accepted.

For a confirmed context $x_{\le a}$, the reference model D_{θ_0} and guided drafter D_θ propose $\hat{y}^0_{1:K}$ and $\hat{y}^\theta_{1:K}$. We compare both with the same target greedy continuation $g_{1:K}$. The indicator H^u_j records whether all of the first j proposals match the target:

$$H^u_j = \prod_{i=1}^{j} \mathbf{1}[\hat{y}^u_i = g_i], \qquad u \in \{0, \theta\}. \tag{4}$$

The reference's accepted length is $A^0 = \sum_{j=1}^{K} H_j^0$. We assign protection losses to prefixes of lengths $j \leq A^0$ and repair losses to longer prefixes. Because the reference model is frozen, this assignment remains fixed for each training example.

For $K = 4$, suppose the reference model matches positions 1, 2, and 4 but fails at position 3. Then $H_{1:4}^0 = (1, 1, 0, 0)$ and $A^0 = 2$: endpoints 1 and 2 receive protection, while 3 and 4 receive repair losses. Matching token 4 alone cannot restore its prefix; token 3 must be corrected while the other matches are preserved.

Considering that H_j^θ is discrete, we construct a continuous prefix score from the guided logits $z_i(v)$ to optimize prefix acceptance. Greedy verification depends on whether the target token ranks first. We therefore use the margin $m_i = z_i(g_i) - \max_{v \neq g_i} z_i(v)$ to compare the target token with its strongest alternative to measure whether it ranks above all other tokens. Following logistic score-difference models (Burges et al., 2005), we map each margin through the sigmoid σ and combine the resulting scores with a prefix product (van Krieken et al., 2022):

$$r_i = \sigma(m_i/T_s), \qquad S_j = \prod_{i=1}^{j} r_i, \qquad T_s > 0. \tag{5}$$

Here T_s controls smoothing, not decoding temperature. The product includes every token needed to accept the first j proposals, so one low token score limits all longer prefix scores that contain it. S_j is a training score, not a calibrated acceptance probability, and its product form does not assume token independence.

We use different losses for protection and repair. For a prefix accepted by the reference model, we use a negative logarithmic loss. The logarithm turns the prefix product into a sum of token losses, so a low score at one position does not weaken the protection signal at other positions. For a longer prefix, correcting one token is not enough to accept the whole prefix if another error remains. We therefore use a loss linear in S_j without taking its logarithm. This makes each token's repair signal depend on the scores of the other tokens in the prefix. The resulting loss is:

$$\ell_j = \begin{cases} -\log S_j / \log 2, & H_j^0 = 1 \quad \text{(protection)}, \\ 1 - 2S_j, & H_j^0 = 0 \quad \text{(repair)}. \end{cases} \tag{6}$$

Both losses encourage agreement with the target, not imitation of reference errors. Unlike reference-based token filtering (Hu et al., 2025), the objective includes all K prefix lengths:

$$\mathcal{L}_{\mathrm{RGO}}(a) = \frac{1}{K} \sum_{j=1}^{K} \ell_j. \tag{7}$$

The constants in Equation 6 connect the loss to accepted length. An incorrect guided prediction has $r_i \leq 1/2$, so any prefix containing it has $S_j \leq 1/2$. Dividing the protection loss by $\log 2$ therefore gives a penalty of at least one when a reference-accepted prefix is lost. The repair loss is negative only when $S_j > 1/2$, which requires every margin in the prefix to be positive, and is bounded below by -1. These properties give the acceptance-gain bound below.

3.3 Acceptance Analysis

Proposition (Acceptance gain). Consider the reference model and the guided drafters on the same confirmed context and K-token target greedy continuation. Let A^0 and A^θ be their accepted lengths. With finite guided logits, $T_s > 0$, and a common fixed argmax tie rule,

$$\underbrace{A^\theta - A^0}_{\text{accepted-token gain}} \quad \geq \quad \underbrace{-K\,\mathcal{L}_{\mathrm{RGO}}(a)}_{\text{lower bound from the loss}} \quad . \tag{8}$$

Proof. Accepted length is a sum of prefix indicators, so we compare the loss with $H_j^0 - H_j^\theta$ for each prefix. A lost reference-accepted prefix contributes 1 to this difference, and its protection loss is at least 1 because $S_j \leq 1/2$. A newly accepted prefix contributes -1, and its repair loss is bounded below by -1. When the acceptance outcome is unchanged, the difference is zero: protection is

always nonnegative, and an unrepaired prefix has $S_j \leq 1/2$ and hence a nonnegative repair loss. Thus $H_j^0 - H_j^\theta \leq \ell_j$ in every case. Summing over all j gives $A^0 - A^\theta \leq \sum_j \ell_j = K\mathcal{L}_{\mathrm{RGO}}(a)$, which proves Equation 8. □

The result connects the loss to the number of accepted tokens, not just to the prefix scores. If $\mathcal{L}_{\mathrm{RGO}}(a) < 0$, the guided drafter must accept at least one more token than the reference model on that block. If $0 \leq \mathcal{L}_{\mathrm{RGO}}(a) < 1/K$, its accepted length cannot decrease because accepted lengths are integers. These are sufficient conditions: a positive loss can still accompany an improvement, and a lower loss does not always mean longer acceptance. The bound does not guarantee improvement after every training update or higher decoding throughput. Appendix A gives further analysis of the bound and gradients.

3.4 Training

We initialize the guided drafter from pretrained PARD and retain a frozen copy as a reference model. Training blocks come from responses generated by the target model. At each anchor x_a, both drafters receive the same confirmed context $x_{\leq a}$ and latent queries $q_{2:K}$. The guided drafter additionally receives historical target features through the compact guidance module.

The reference model's proposals determine $H_{1:K}^0$ and hence which prefixes receive protection or repair. The guided logits provide the prefix scores for $\mathcal{L}_{\mathrm{RGO}}$ and the token distributions for the token-level losses. We jointly train all draft-model weights and guidance parameters with

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{KL}} + \lambda_{\mathrm{CE}}\mathcal{L}_{\mathrm{CE}} + \lambda_{\mathrm{RGO}}\mathcal{L}_{\mathrm{RGO}}. \tag{9}$$

Here $\mathcal{L}_{\mathrm{KL}}$ is forward KL from target model to the guided token distribution. Only this term uses PARD-2's Confidence-Adaptive Token (CAT) weights, based on target confidence in preceding continuation tokens (An et al., 2026b). Uniform token cross-entropy $\mathcal{L}_{\mathrm{CE}}$ provides direct supervision even when low prefix scores weaken the repair signal from the prefix loss.

The prefix loss averages block losses within each response before applying sequence weights. Appendix B.1 details the loss normalizations, and Section 4.1 gives the coefficients. The target and reference model remain frozen throughout training; inference uses only the trained drafter and target.

4 Experiments

4.1 Experimental Setup

Models and benchmarks. We evaluate Qwen3-8B, Qwen3-14B, and Qwen3-32B as target models (Qwen Team, 2025) and use PARD-Qwen3-0.6B as drafters. The six benchmarks cover **math** (GSM8K, MATH-500, AIME25), **code** (HumanEval, MBPP), and **chat** (MT-Bench) (Cobbe et al., 2021; Lightman et al., 2023; Mathematical Association of America, 2025; Chen et al., 2021; Austin et al., 2021; Zheng et al., 2023). All methods use the same prompts, stopping rules, and maximum output length of 2,048 tokens, with greedy decoding ($T = 0$) and thinking mode disabled.

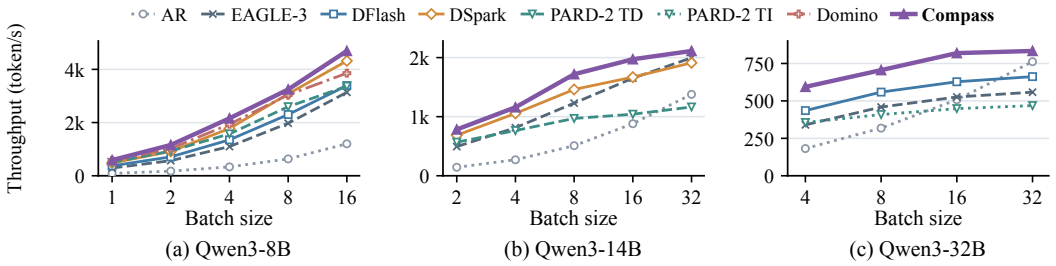
Baselines. We compare against target-only autoregressive decoding (AR), EAGLE-3 (Li et al., 2025), DFlash (Chen et al., 2026), PARD-2 in target-independent (TI) and target-dependent (TD) modes (An et al., 2026b), Domino (Huang et al., 2026), and DSpark (Cheng et al., 2026).

Metrics. The main table reports speedup α, the output-token throughput ratio to matched AR, and block efficiency τ, the mean number of committed tokens per sequence per verification cycle. Block efficiency includes any correction or bonus token actually committed.

Training and implementation. We sample 200K training prompts and 1,024 validation prompts from Open-PerfectBlend (Labonne, 2024) using seed 42, and regenerate responses with the corresponding target. The drafter and guidance module are jointly trained for five epochs; the training of drafters uses eight Ascend 910B NPUs. The default source dimension is $r = 256$, $\lambda_{\mathrm{CE}} = 0.1$, $\lambda_{\mathrm{RGO}} = 0.8$, and $T_s \approx 1.974$. All methods run in the same vLLM engine (Kwon et al., 2023) with chain-based verification. Tables 4 and 5 give the training details and main evaluation settings. Unless

Table 1: Greedy decoding on Qwen3 targets (Q3), with $B = \mathrm{TP} = 1, 2, 4$ for 8B, 14B, and 32B. Parentheses give proposal count K. Each pair reports AR-relative speedup α and block efficiency τ. Overall averages the six benchmarks equally; bold marks the best displayed value per target.

Target	Method		GSM8K		MATH-500		AIME25		HumanEval		MBPP		MT-Bench		Overall	
			α	τ	α	τ	α	τ	α	τ	α	τ	α	τ	α	τ
Q3 8B	AR		1.00	—	1.00	—	1.00	—	1.00	—	1.00	—	1.00	—	1.00	—
	EAGLE-3	(8)	3.30	5.89	3.19	5.71	2.81	5.14	2.75	5.33	2.68	4.82	1.67	3.00	2.73	4.98
	EAGLE-3	(16)	2.68	6.94	2.57	6.65	2.24	5.87	2.04	5.65	1.98	5.19	1.16	3.02	2.11	5.55
	DFlash	(8)	4.23	5.11	5.05	6.19	4.53	5.73	4.20	5.87	3.93	4.78	2.46	2.99	4.07	5.11
	DSpark	(8)	5.45	7.00	5.40	6.95	4.70	6.22	4.27	6.47	4.46	5.72	3.09	3.95	4.56	6.05
	PARD-2 (TI)	(16)	4.89	6.62	5.50	7.60	4.45	6.43	4.18	6.83	3.87	5.21	2.46	3.32	4.22	6.00
	PARD-2 (TD)	(16)	5.73	7.85	6.55	9.07	5.33	7.67	4.50	8.15	4.59	6.19	2.84	3.82	4.92	7.12
	Domino	(16)	6.46	9.27	6.28	8.84	5.10	7.33	4.30	7.24	4.48	6.28	2.74	3.80	4.89	7.13
	Compass	(16)	**6.58**	**9.69**	**6.69**	**9.56**	**5.57**	**8.23**	**4.60**	**9.33**	**5.07**	**7.19**	**3.11**	**4.35**	**5.27**	**8.06**
Q3 14B	AR		1.00	—	1.00	—	1.00	—	1.00	—	1.00	—	1.00	—	1.00	—
	EAGLE-3	(8)	3.44	5.99	3.21	5.58	2.70	4.79	2.80	5.68	2.69	4.57	1.67	2.86	2.75	4.91
	EAGLE-3	(16)	2.87	7.23	2.60	6.47	2.12	5.38	2.20	6.28	2.03	4.96	1.18	2.91	2.17	5.54
	DSpark	(8)	4.62	7.02	4.72	6.93	4.00	6.07	3.65	6.72	4.00	5.70	2.62	3.92	3.93	6.06
	PARD-2 (TI)	(16)	3.40	6.55	3.79	7.35	3.08	6.13	3.04	7.08	2.76	5.04	1.72	3.21	2.97	5.89
	PARD-2 (TD)	(16)	3.96	7.79	4.53	8.83	3.63	7.32	3.49	8.52	3.29	6.07	2.05	3.84	3.49	7.06
	Compass	(16)	**5.16**	**9.18**	**5.14**	**9.24**	**4.35**	**7.46**	**3.81**	**9.26**	**4.28**	**7.12**	**2.77**	**4.47**	**4.25**	**7.79**
Q3 32B	AR		1.00	—	1.00	—	1.00	—	1.00	—	1.00	—	1.00	—	1.00	—
	EAGLE-3	(8)	1.87	3.70	1.94	3.88	1.82	3.68	1.79	4.01	1.73	3.38	1.37	2.65	1.76	3.55
	EAGLE-3	(16)	1.31	3.80	1.37	4.00	1.29	3.78	1.34	4.10	1.20	3.46	0.94	2.66	1.24	3.63
	DFlash	(8)	2.40	4.77	2.42	4.82	2.22	4.57	2.18	5.22	1.98	3.83	1.57	3.00	2.13	4.37
	PARD-2 (TI)	(16)	1.96	6.41	2.21	7.29	1.87	6.18	2.06	8.13	1.50	4.76	0.99	3.07	1.76	5.97
	Compass	(16)	**2.97**	**9.08**	**2.92**	**7.83**	**2.46**	**6.90**	**2.19**	**9.19**	**2.20**	**6.41**	**1.61**	**3.90**	**2.39**	**7.22**



(a) Qwen3-8B (b) Qwen3-14B (c) Qwen3-32B

Figure 3: Output throughput on GSM8K. Qwen3-8B, 14B, and 32B use TP of 1, 2 and 4 respectively.

otherwise specified, ablations, hyperparameter analysis, and guidance comparisons use Qwen3-14B with $K = 16$ and $B = \mathrm{TP} = 2$, and evaluate on GSM8K, MBPP, and MT-Bench.

4.2 Main Results

Compass achieves the highest speedup and block efficiency across all 18 target–benchmark pairs across math, code, and chat (Table 1). Its mean AR-relative speedups are $5.27\times$, $4.25\times$, and $2.39\times$ on Qwen3-8B, 14B, and 32B, exceeding the strongest baseline means—PARD-2 (TD), DSpark, and DFlash—by 7.0%, 8.1%, and 12.4%, respectively. Across the three target configurations, the widening speed advantage accompanies mean τ gains of 13.1%, 28.5%, and 65.2% over these baselines: more committed tokens amortize each verification round. At matched $K = 16$, τ remains 10.3–20.8% higher than PARD-2 (TD on 8B/14B; TI on 32B). Compared with PARD-2 (TD)'s input fusion and CAT weighting, Compact Layerwise Target Guidance provides compact layerwise MLP guidance, and Reference-Aware Guidance Optimization trains for reference-relative prefix gains.

Batch scaling. Figure 3 shows that *Compass* achieves the highest throughput across all tested configurations. At the largest batch sizes ($B = 16$ for 8B and $B = 32$ for 14B/32B), it reaches 4,692, 2,112, and 834 token/s, outperforming the strongest alternatives by 8.6%, 5.8%, and 9.4%, respectively. This advantage persists even as larger batches improve AR hardware utilization and narrow the headroom for speculative decoding; notably, at $B = 32$ on Qwen3-32B, *Compass* is the only tested speculative method that remains faster than AR.

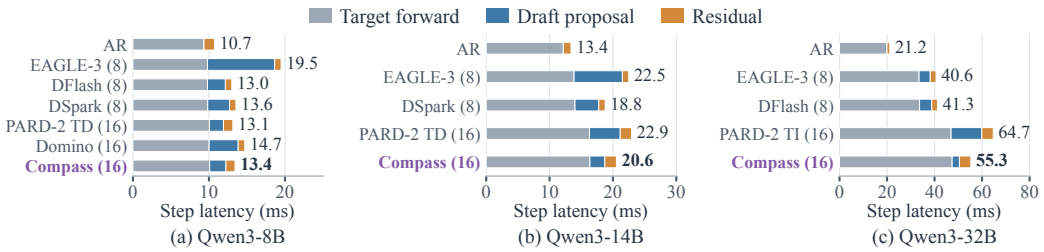


Figure 4: Per-step runtime breakdown with $B = \text{TP} = 1, 2, 4$ for Qwen3-8B, 14B, and 32B, respectively. Residual is total latency minus target forward and draft proposal time.

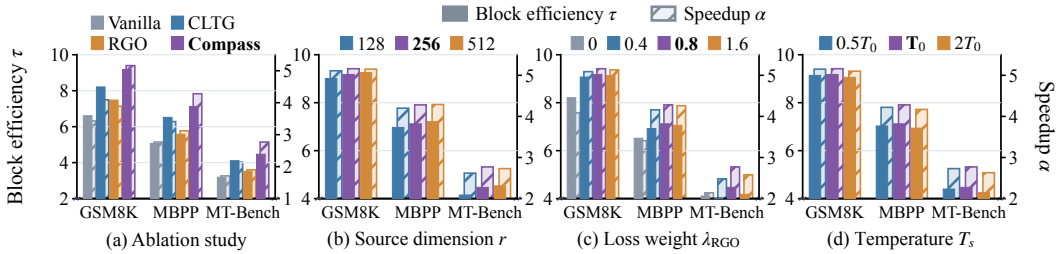


Figure 5: Component ablation and hyperparameter analysis on Qwen3-14B. Solid bars show τ (left); hatched bars show α (right). Vanilla uses KL+CE; *Compass* uses both CLTG and RGO.

Runtime breakdown. Figure 4 shows that *Compass* generates 16 proposals with 75.7%, 69.1%, and 35.1% less proposal time than EAGLE-3 at $K = 8$ on the three targets, respectively. It uses one backbone pass, avoiding EAGLE-3's repeated drafting calls and the sequential heads of Domino and DSpark. Proposal time accounts for only 16.0%, 11.4%, and 5.5% of total step latency.

4.3 Ablation Study

For this ablation, CLTG and RGO abbreviate Compact Layerwise Target Guidance and Reference-Aware Guidance Optimization, respectively. We compare Vanilla (KL+CE), CLTG only, RGO only, and *Compass*Figure 5(a) shows gains from both components on all three tasks: relative to Vanilla, CLTG and RGO individually raise mean τ from 4.96 to 6.28 and 5.53, and mean speedup by 21.9% and 12.6%, respectively. *Compass* achieves the best acceptance and speed throughout, improving speed over Vanilla by 50.4–62.0%. Adding RGO to CLTG gives a further 26.7% mean speed gain.

Table 2: Repair and damage per matched block, relative to the reference model.

Dataset	Method	$\bar{R}\uparrow$	$\bar{D}\downarrow$	$\Delta\bar{A}\uparrow$
GSM8K	CLTG-only	3.51	1.29	2.22
	Compass	4.14	0.95	3.19
MBPP	CLTG-only	1.89	0.68	1.21
	Compass	2.23	0.42	1.81
MT-Bench	CLTG-only	0.32	0.12	0.20
	Compass	0.60	0.04	0.56

On matched prefixes (Table 2), RGO increases repaired endpoints $\bar{R}$ by 0.63, 0.34, and 0.28 per block on GSM8K, MBPP, and MT-Bench, respectively, while reducing damage $\bar{D}$ by 26.4%, 38.2%, and 66.7%. The resulting gains in $\Delta\bar{A} = \bar{R} - \bar{D}$ are 0.97, 0.60, and 0.36 accepted draft tokens per block over CLTG-only. Reduced damage accounts for 22.2–43.3% of these gains, supporting the benefit of protecting reference-accepted prefixes alongside repairing failed ones.

4.4 Hyperparameter Analysis

Source dimension. Figure 5(b-d) evaluates the effects of source dimension r, reference-aware weight λ_{RGO}, and smoothing temperature T_0. The compact choice $r = 256$ gives the highest mean speedup ($4.07\times$) and stays within 0.3% of the best tested speedup on every dataset. Increasing r from 128 to 256 improves both τ and speed on all three tasks. Doubling it again to 512 adds only 0.06–0.09 committed tokens per cycle and gives no mean speedup gain ($4.06\times$). Thus, $r = 256$ delivers near-peak speed across domains with half the guidance projection parameters of $r = 512$.

Reference-aware weight. The default $\lambda_{\text{RGO}} = 0.8$ achieves the highest τ and speedup on GSM8K, MBPP, and MT-Bench, supporting a shared loss balance across domains. The speed benefit is also robust to the tested weights: all three nonzero settings (0.4, 0.8, and 1.6) outperform

$\lambda_{\mathrm{RGO}} = 0$ on every task, with gains of 15.9–29.4%. The common peak at 0.8 suggests that moderate prefix supervision complements KL/CE more effectively than further increasing its weight.

Smoothing temperature. The default $T_0 \approx 1.974$ gives the highest τ and speedup on all three tasks. Across $T_s \in \{0.5T_0, T_0, 2T_0\}$, mean speedup remains between $3.97\times$ and $4.07\times$, indicating limited average sensitivity to smoothing. At $2T_0$, τ decreases by 0.12, 0.18, and 0.22 on GSM8K, MBPP, and MT-Bench, respectively. The shared optimum is consistent with avoiding excessive smoothing, which weakens positive-margin evidence in the prefix products.

4.5 Guidance Design and Runtime Cost

With $\lambda_{\mathrm{RGO}} = 0$, Compact Layerwise Target Guidance with both branches achieves the highest τ (6.28) and speedup ($3.21\times$) in Table 3. Steering into the expanded up branch reaches similar τ (6.27), but costs 3.72 ms per draft versus 2.35 ms for our interface: a 36.8% latency reduction and 9.9% speed gain. Steering improves τ over input-only fusion (5.56 to 6.27), yet barely improves speed ($2.91\times$ to $2.92\times$). Reusing native expansion projections preserves the acceptance benefit while limiting guidance overhead. Dual-branch guidance raises τ to 6.28 and speed by 5.9% and 5.2% comparing with the Gate-only and up-only approaches, at only 0.05 ms additional draft time. This supports complementary control of channel activations and gates at little runtime cost.

Table 3: Guidance analysis on Qwen3-14B.

Variant	Params (M)	Draft (ms)	Cycle (ms)	τ	α
Input-only	19.92	2.11	20.62	5.56	2.91
Steering	519.06	3.72	22.86	6.27	2.92
Gate only	11.27	2.30	20.74	5.82	3.03
Up only	11.27	2.30	20.76	5.84	3.05
Dual branch	18.61	2.35	21.19	**6.28**	**3.21**

5 Related Work

Target-feature guidance. Target-model representations provide additional information for draft generation (Li et al., 2024a; Ankner et al., 2024). EAGLE-3 fuses shallow, middle, and deep target features at the drafter input (Li et al., 2025), while PARD-2 adds projected multilayer features to draft input embeddings (An et al., 2026b). For layerwise conditioning, DFlash injects target context into draft attention keys and values (Chen et al., 2026; Huang et al., 2026; Cheng et al., 2026), and SD^2 adds target-dependent biases to the expanded up branch of each draft MLP (Berdoz et al., 2025). *Compass* injects separate gate and up input shifts into every draft MLP, using a shared low-dimensional source and reusing native projections to limit guidance overhead.

Acceptance-aware training. Several methods adapt draft training to prefix-based verification. DFlash emphasizes earlier positions with exponentially decaying token weights (Chen et al., 2026), while PARD-2's Confidence-Adaptive Token (CAT) optimization uses cumulative target confidence in preceding tokens (An et al., 2026b). D-PACE derives adaptive position weights from a differentiable accepted-length surrogate based on draft confidence (Wu et al., 2026). SpecDiff-2's streak-distillation directly optimizes sums of prefix-probability products along target-generated continuations (Sandler et al., 2026). Spec-AUF restricts supervision to the current drafter's accepted prefix and first incorrect token (Yang & Li, 2026), and AdaSPEC uses a separately distilled reference model to select tokens for distillation (Hu et al., 2025). *Compass* instead uses the initial drafter's accepted prefixes to separate protection from repair, and connects the resulting loss to a lower bound on reference-relative acceptance gain.

6 Conclusion

We introduced *Compass*, a target-guided parallel speculative decoding framework that combines *Compact Layerwise Target Guidance* with *Reference-Aware Guidance Optimization*. Its guidance reused native MLP projections for a single parallel draft pass, while the prefix objective trained the drafter to preserve and extend acceptance relative to the reference model. We derived a per-block lower bound on acceptance gain from the prefix loss under greedy verification. Across six benchmarks, *Compass* achieved mean AR-relative speedups of $5.27\times$, $4.25\times$, and $2.39\times$ on Qwen3-8B, Qwen3-14B, and Qwen3-32B, respectively. Ablations and runtime analyses supported the complementary roles of limiting guidance overhead and optimizing prefix acceptance.

AI USE STATEMENT

Generative AI tools were used for language editing, structural revision, and literature cross-checking. No experimental values were generated by these tools. The authors reviewed the final text and take responsibility for its content.

REPRODUCIBILITY STATEMENT

We have made efforts to support the reproducibility of the proposed method and reported experimental results. Section 3 describes the *Compass* framework, including compact layerwise target guidance, reference-aware prefix optimization, and the training procedure. Section 3.3 and Appendix A provide the assumptions, proof, and further analysis of the acceptance-gain bound under greedy verification. Section 4 and Appendix B describe the datasets, training data preparation, target and draft models, baselines, evaluation metrics, and vLLM evaluation settings. Appendix B further specifies loss normalization, training hyperparameters, hardware configurations, and the settings used for component ablations, hyperparameter analysis, and batch-scaling experiments.

REFERENCES

Zihao An, Huajun Bai, Ziqiong Liu, Dong Li, and Emad Barsoum. PARD: Accelerating LLM inference with low-cost parallel draft model adaptation. In *International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=XbOyv7iVGL.

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, 2026b. 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. In *Conference on Language Modeling*, 2024. doi: 10.48550/arXiv.2402.05109. URL https://arxiv.org/abs/2402.05109.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Frédéric Berdoz, Peer Rheinboldt, and Roger Wattenhofer. Steering pretrained drafters during speculative decoding, 2025. URL https://arxiv.org/abs/2511.09844. Accepted at AAAI 2026.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005. doi: 10.1145/1102351.1102363. URL https://www.microsoft.com/en-us/research/publication/learning-to-rank-using-gradient-descent/.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning*, volume 235, pp. 5209–5235, 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*, pp. 129531–129563, 2024. URL https://arxiv.org/abs/2402.12374.

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. Preprint.

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.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. URL https://arxiv.org/abs/2403.05530.

Yuezhou Hu, Jiaxin Guo, Xinyu Feng, and Tuo Zhao. AdaSPEC: Selective knowledge distillation for efficient speculative decoders, 2025. URL https://arxiv.org/abs/2510.19779.

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.

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 *ACM Symposium on Operating Systems Principles*, 2023. doi: 10.1145/3600006.3613165. URL https://arxiv.org/abs/2309.06180.

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

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023. URL https://arxiv.org/abs/2211.17192. ICML 2023 Oral.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, volume 235, pp. 28935–28948, 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 *Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432, 2024b. doi: 10.18653/v1/2024.emnlp-main.422. URL https://aclanthology.org/2024.emnlp-main.422.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test, 2025. 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, 2023. URL https://arxiv.org/abs/2305.20050.

Mathematical Association of America. American invitational mathematics examination (AIME) 2025, 2025. URL https://maa.org/math-competitions/american-invitational-mathematics-examination-aime.

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*, pp. 932–949, 2024. doi: 10.1145/3620666.3651335. URL https://arxiv.org/abs/2305.09781.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems*, volume 5, pp. 606–624, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/c4be71ab8d24cdfb45e3d06dbfca2780-Abstract-mlsys2023.html.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Jameson Sandler, Jacob Christopher, Tom Hartvigsen, and Ferdinando Fioretto. Specdiff-2: Scaling diffusion drafter alignment for faster speculative decoding. *Proceedings of Machine Learning and Systems*, 8:1128–1147, 2026.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, 2018. URL https://arxiv.org/abs/1811.03115.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016a. URL https://arxiv.org/abs/1609.03499.

Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with PixelCNN decoders. *arXiv preprint arXiv:1606.05328*, 2016b. URL https://arxiv.org/abs/1606.05328.

Emile van Krieken, Erman Acar, and Frank van Harmelen. Analyzing differentiable fuzzy logic operators. *Artificial Intelligence*, 302:103602, 2022. doi: 10.1016/j.artint.2021.103602. URL https://arxiv.org/abs/2002.06100.

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.

Tengyu Xu, Eryk Helenowski, Karthik Abinav Sankararaman, Di Jin, Kaiyan Peng, Eric Han, Shaoliang Nie, Chen Zhu, Hejia Zhang, Wenxuan Zhou, Zhouhao Zeng, Yun He, Karishma Mandyam, Arya Talabzadeh, Madian Khabsa, Gabriel Cohen, Yuandong Tian, Hao Ma, Sinong Wang, and Han Fang. The perfect blend: Redefining RLHF with mixture of judges, 2024. URL https://arxiv.org/abs/2409.20370.

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. Version 1.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, 2019. URL https://arxiv.org/abs/1910.07467.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2306.05685.

A Analysis of Reference-Aware Guidance Optimization

This appendix expands the analysis in Section 3.3. The reference and guided drafters are evaluated on the same confirmed context and the same K target tokens generated by greedy decoding, with a common fixed argmax tie rule. We use $A^u = \sum_{j=1}^{K} H_j^u$ and $\mathcal{L}_{\mathrm{RGO}}(a) = K^{-1} \sum_{j=1}^{K} \ell_j$ as in the main text.

Acceptance bound. For finite guided logits and $T_s > 0$, the prefix scores satisfy $0 < S_j < 1$. If a guided prefix is rejected, at least one target-token margin is nonpositive, including a tie resolved against the target token, so $S_j \leq 1/2$. The protection loss is therefore at least one for a lost reference-accepted prefix and is otherwise nonnegative. The repair loss is nonnegative for a rejected guided prefix and is always bounded below by -1. The four cases are summarized below:

H_j^0	H_j^θ	$H_j^0 - H_j^\theta$	Bound on ℓ_j
1	0	1	$\ell_j \geq 1$
1	1	0	$\ell_j \geq 0$
0	0	0	$\ell_j \geq 0$
0	1	-1	$\ell_j \geq -1$

In every case, $H_j^0 - H_j^\theta \leq \ell_j$. Summing over $j = 1, \ldots, K$ gives

$$\frac{A^0 - A^\theta}{K} \leq \mathcal{L}_{\mathrm{RGO}}(a), \tag{10}$$

which is equivalent to the acceptance-gain bound in Equation 8.

Averaging over inputs. For a fixed K, let $\alpha_a \geq 0$ be fixed weights with $\sum_a \alpha_a = 1$ over the compared inputs. Applying the same inputs and weights to both drafters gives

$$\sum_a \alpha_a (A_a^\theta - A_a^0) \geq -K \sum_a \alpha_a \mathcal{L}_{\mathrm{RGO}}(a). \tag{11}$$

Thus, a negative weighted mean loss implies a positive weighted mean accepted-length gain. The weights and input set are shared between the two sides of the comparison.

Loss scaling. The construction uses logistic score comparisons (Burges et al., 2005) and product/log-product combinations (van Krieken et al., 2022). The identity $\sigma(0) = 1/2$, together with $S_j \leq \min_{i \leq j} r_i$, provides the score bound for rejected prefixes. Scaling the logarithmic protection loss to equal one at this boundary gives $-\log S_j / [-\log(1/2)] = -\log S_j / \log 2$. For a linear repair loss, requiring a value of zero at $S = 1/2$ and a limit of -1 as $S \to 1$ gives

$$\ell_{\mathrm{repair}}(S) = \frac{1/2 - S}{1 - 1/2} = 1 - 2S. \tag{12}$$

These constants connect loss values to unit changes in accepted-prefix length; they are not claimed to be empirically optimal. The additive constant in the repair term affects its value and the bound, but not its gradients.

Interpreting the bound. Since protection terms are nonnegative and each of the $K - A^0$ repair terms is bounded below by -1, the block loss satisfies $\mathcal{L}_{\mathrm{RGO}}(a) \geq -(K - A^0)/K$. A negative loss is sufficient, but not necessary, for an acceptance gain. To illustrate, take $K = 2$, $A^0 = 1$, and $r_1 = r_2 = r$. In both rows below, both guided proposals are accepted, so $A^\theta - A^0 = 1$:

r	Protection	Repair	$\mathcal{L}_{\mathrm{RGO}}(a)$	$-K\mathcal{L}_{\mathrm{RGO}}(a)$
0.9	0.1520	-0.6200	-0.2340	0.4680
0.6	0.7370	0.2800	0.5085	-1.0170

Only the first row has a negative loss, although the realized acceptance gain is the same. This illustrates why the bound can be conservative and why loss values do not fully determine accepted length. For an individual block, $0 \leq \mathcal{L}_{\mathrm{RGO}}(a) < 1/K$ rules out a decrease because accepted-length changes are integers; this conclusion does not apply to a weighted mean across inputs.

Gradient behavior. The prefix formulation gives reference-dependent emphasis to earlier positions. Expanding the logarithmic protection terms and exchanging the two sums yields

$$K\,\mathcal{L}_{\mathrm{RGO}}(a) = -\frac{1}{\log 2}\sum_{i=1}^{A^0}(A^0 - i + 1)\log r_i + (K - A^0) - 2\sum_{j=A^0+1}^{K} S_j. \tag{13}$$

Thus, protection is equivalent to a margin-logistic loss with triangular weights determined by the reference's accepted length: position $i \le A^0$ supports $A^0 - i + 1$ already accepted endpoints. Repair retains the joint prefix products, so the effect of an improvement at a later position is moderated by the scores of all preceding predictions.

For $k \le j$, differentiating Equation 6 with respect to the guided margin gives

$$\frac{\partial \ell_j}{\partial m_k} = \begin{cases} -\dfrac{1 - r_k}{T_s \log 2}, & H_j^0 = 1, \\ -\dfrac{2S_j(1 - r_k)}{T_s}, & H_j^0 = 0. \end{cases} \tag{14}$$

The derivatives are zero for $k > j$. The protection gradient does not contain the full prefix product and remains responsive when a protected margin is low. The repair gradient does contain S_j: a low early margin can therefore weaken the repair signal for every longer prefix that includes it. Uniform token cross-entropy supplies direct supervision even in this regime, while CAT-weighted KL provides complementary distribution-level alignment. Since margins share model parameters and training also includes these token-level losses, the derivatives do not guarantee that every margin increases after an update. We compute $\log S_j = \sum_{i=1}^{j} \log \sigma(m_i/T_s)$ in FP32 for numerical stability.

B Implementation and Experimental Details

B.1 Loss Computation

Let $\mathcal{A}_b$ denote the training anchors in response b, and $\mathcal{R}_b$ the supervised anchor–position pairs (a, i). The frozen target supplies distributions $p_{b,a,i}$ and tokens $g_{b,a,i}$, while the guided draft distribution is $q_{b,a,i} = \mathrm{softmax}(z_{b,a,i})$. CAT weights (An et al., 2026b) are applied only to KL:

$$\omega_{b,a,1} = 1, \qquad \omega_{b,a,i} = \mathrm{stopgrad}\left(\prod_{k<i} p_{b,a,k}(g_{b,a,k})\right). \tag{15}$$

With fixed sequence weights $w_b \ge 0$ and $\sum_b w_b > 0$, the two token losses retain their distinct normalizations:

$$\mathcal{L}_{\mathrm{KL}} = \frac{\sum_b w_b \sum_{(a,i)\in\mathcal{R}_b} \omega_{b,a,i} D_{\mathrm{KL}}(p_{b,a,i} \| q_{b,a,i})}{\sum_b w_b \sum_{(a,i)\in\mathcal{R}_b} \omega_{b,a,i}}, \tag{16}$$

$$\mathcal{L}_{\mathrm{CE}} = \frac{\sum_b w_b \sum_{(a,i)\in\mathcal{R}_b} -\log q_{b,a,i}(g_{b,a,i})}{\sum_b w_b |\mathcal{R}_b|}. \tag{17}$$

The designed loss first averages the block losses from Equation 7 within each response, then across responses:

$$\mathcal{L}_{\mathrm{RGO}} = \frac{1}{\sum_b w_b} \sum_b \frac{w_b}{N_b} \sum_{a\in\mathcal{A}_b} \mathcal{L}_{\mathrm{RGO}}(b, a), \qquad N_b = |\mathcal{A}_b|. \tag{18}$$

B.2 Implementation and Evaluation Notes

Data and guidance settings. Training and validation prompts are drawn from `mlabonne/open-perfectblend` (Labonne, 2024; Xu et al., 2024); the original responses are replaced with outputs from the corresponding frozen target.[1] For Qwen3-14B and Qwen3-32B, EAGLE-3's block-input taps are $[2, 20, 37]$ and $[2, 32, 61]$, corresponding to the completed-block outputs $[1, 19, 36]$ and

[1] Dataset release: `https://huggingface.co/datasets/mlabonne/open-perfectblend`.

Table 4: Training settings for *Compass* on the three target models. The target and original-PARD reference remain frozen.

Hyperparameter	Qwen3-8B	Qwen3-14B	Qwen3-32B
Training			
Training prompts	200,000	200,000	200,000
Validation prompts	1,024	1,024	1,024
Random seed	42	42	42
Response generation	Corresponding target		
Mask-token ID	151665	151665	151665
Guidance rank r	256	256	256
Trainable weights	Entire guided drafter and guidance maps		
Training draft length K_{train}	8	8	8
Maximum sequence length	1,024	1,024	1,024
Optimizer	AdamW	AdamW	AdamW
Initial / minimum learning rate	4×10^{-7}	4×10^{-7}	4×10^{-7}
Peak learning rate	4×10^{-6}	4×10^{-6}	4×10^{-6}
Learning-rate warm-up	First 5% of training steps		
Learning-rate schedule	Linear warm-up, then cosine decay		
Weight decay	0.01	0.01	0.01
Gradient clipping norm	1.0	1.0	1.0
Epochs	5	5	5
Per-device batch size	8	8	8
Gradient accumulation	1	1	1
Global batch size	64	64	64
KL coefficient	1	1	1
CE coefficient λ_{CE}	0.1	0.1	0.1
λ_{RGO} after warm-up	0.8	0.8	0.8
Prefix-loss warm-up	First 10% of one epoch		
Smoothing temperature T_s	$5.8125/\log 19 \approx 1.97406$		
Training precision	BF16	BF16	BF16
Training accelerators	8× Ascend 910B	8× Ascend 910B	8× Ascend 910B

Table 5: Target-specific configurations for the main comparison. Feature taps use zero-based completed-block indices; the 14B and 32B taps follow EAGLE-3 (Li et al., 2025). Settings for the additional analyses are specified below.

Setting	Qwen3-8B	Qwen3-14B	Qwen3-32B
Compass draft backbone	Qwen3-0.6B	Qwen3-0.6B	Qwen3-0.6B
Target feature taps	$[1, 17, 33]$	$[1, 19, 36]$	$[1, 31, 60]$
GPU configuration	1× RTX 5090	2× RTX 5090	4× RTX 5090
Batch size B	1	2	4
Tensor parallelism	1	2	4
Compass proposal count K	16	16	16
Precision / quantization	BF16 / none		
Decoding / thinking mode	Greedy ($T = 0$) / disabled		
Prompt format	Plain prompts		
Maximum output length	2,048 tokens		
Serving engine	vLLM		

$[1, 31, 60]$ in Table 5. The Qwen placeholder token follows PARD (An et al., 2026a). Target-feature RMS normalization uses FP32, $\epsilon = 10^{-6}$, and no learnable affine parameters. For the Qwen3-8B configuration, $d_T = 4096$, $d_D = 1024$, $m = 3072$, and $L_D = 28$, giving 17,825,792 added guidance parameters at $r = 256$. We ramp λ_{RGO} to 0.8 over the first 0.1 epoch and keep it fixed thereafter.

Component and hyperparameter analyses. These experiments use Qwen3-14B with $K = 16$, $B = \text{TP} = 2$, and GSM8K, MBPP, and MT-Bench. All four component variants share the PARD initialization, training data, five-epoch budget, and KL/CE losses; the draft backbone remains trainable. Hyperparameter experiments vary one setting at a time: $r \in \{128, 256, 512\}$,

$\lambda_{\mathrm{RGO}} \in \{0, 0.4, 0.8, 1.6\}$, or $T_s/T_0 \in \{0.5, 1, 2\}$, where $T_0 = 5.8125/\log 19$. Other settings retain their defaults.

Guidance comparisons. All five variants in Table 3 use $\lambda_{\mathrm{RGO}} = 0$ and the same 14B configuration. We report equally weighted means of τ and α over the three datasets, and profile draft and cycle latency at 2,048-token contexts. Steering projects the shared $r = 256$ source into the expanded up branch of each draft MLP, with no gate injection.

Batch scaling. Figure 3 evaluates GSM8K with $B \in \{1, 2, 4, 8, 16\}$ for 8B, $B \in \{2, 4, 8, 16, 32\}$ for 14B, and $B \in \{4, 8, 16, 32\}$ for 32B, keeping TP fixed at 1, 2, and 4, respectively. EAGLE-3, DFlash, and DSpark use $K = 8$; PARD-2, Domino, and *Compass* use $K = 16$. PARD-2 uses TD for 8B/14B and TI for 32B. The plotted metric is aggregate output-token throughput in tokens/s.

Evaluation protocol. Methods are compared within each target configuration using the same GPU count, tensor parallelism, precision, batch size, prompts, output cap, and vLLM runtime. Qwen3-8B, Qwen3-14B, and Qwen3-32B use their own matched AR denominators at $B = \mathrm{TP} = 1$, 2, and 4, respectively. These changes in batch size and deployment configuration prevent the comparison from isolating the effect of model size. Block efficiency averages committed tokens over individual sequence-blocks. The six-benchmark Overall is an unweighted mean of benchmark-level results, corresponding to math/code/chat weights of $3/6$, $2/6$, and $1/6$. With equal weight per domain instead, *Compass* retains the highest mean speedup for all three targets. For 14B, the domain-balanced means are $3.898\times$ for *Compass* and $3.631\times$ for DSpark, a 7.4% advantage, consistent with the six-benchmark ranking.

The available baselines vary by target: 14B lacks DFlash and Domino, while 32B lacks PARD-2 (TD), DSpark, and Domino. The reported main results provide benchmark-level speedups and block efficiencies, without repeated-run uncertainty. Thus, small differences such as the 0.4% lead on 32B HumanEval do not establish a statistically reliable improvement. DFlash uses the `zlab-b16` checkpoint label for 8B in the source results; the 32B checkpoint revision remains to be documented. The main table does not establish the separate contributions of the guidance interface and prefix objective or verify token-by-token equivalence to AR; these require component ablations and output checks, respectively.

For the acceptance diagnostics in Figure 1(b), we compare original PARD-TI with a model adapted using target guidance alone: the draft backbone is frozen and $\lambda_{\mathrm{RGO}} = 0$. Both models receive the same confirmed contexts and target-generated continuations on GSM8K, MBPP, and MT-Bench, with $K = 16$. Speedups use each method's complete draft–verify execution, with no evaluation warm-up.