Attention Drift: What Auto-Regressive Speculative Decoding Models Learn

Doğaç Eldenk[1] **Payal Mohapatra**[2] **Yigitcan Comlek**[3]
Kaan Oktay[1] **Hongyang Zhang**[4,5] **Stephen Xia**[6]

[1]fal [2]Arm Inc., USA [3]GE Aerospace
[4]University of Waterloo [5]UT Austin [6]Northwestern University

Abstract

Speculative decoding accelerates LLM inference by drafting future tokens with a small model, but drafter models degrade sharply under template perturbation and long-context inputs. We identify a previously-unreported phenomenon we call **attention drift**: as the drafter generates successive tokens within a speculation chain, attention progressively moves from the prompt onto its own recently-generated tokens. We observe this across both *EAGLE3* drafters and *MTP heads*, suggesting drift is a property of drafter designs. We trace this to the un-normalized residual path between chain steps: the drafter's hidden state magnitude grows monotonically with chain depth, which exhibits dynamics consistent with additional pre-norm transformer layers stacked on the target rather than as a standalone autoregressive predictor. In order to limit the growth, we propose two architectural changes: Post-norm on the drafter hidden states and per-hidden-state RMSNorm after capturing target hidden states. Our interventions improve acceptance length over the current leading model, pre-norm EAGLE3, by up to $2\times$ under template perturbation, $1.18\times$ on long-context tasks, and $1.10\times$ on seven standard benchmarks spanning multi-turn chat, math, and coding. Our changes also allow shorter train-time-test depths to generalize over longer drafting sequences.

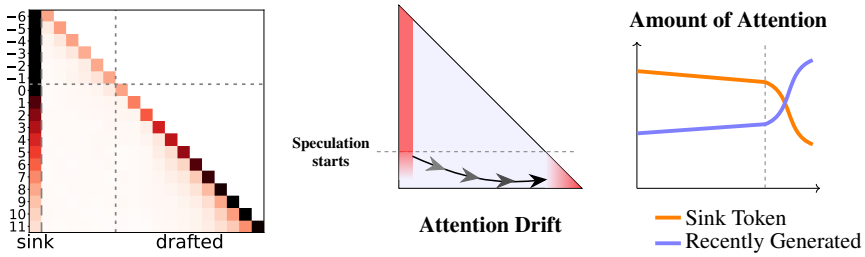


Figure 1: **Attention drift.** During speculation, the drafter's attention moves from the prompt's sink token onto its own recently-generated tokens. *Left:* Emergence of the attention sink demonstrated on drafter's attention heatmap, rows = query, columns = key; darker = higher attention. *Center*: Graphical visualization of attention drift on a drafter. *Right*: Attention per token position on x axis, with speculated tokens to the right of the dashed line.

1 Introduction

Speculative decoding [1, 2] is a lossless acceleration technique for large language model (LLM) inference in which a lightweight *drafter* predicts future tokens that are later verified by a larger

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

target or *verifier* model. In practice, speculative decoding systems must operate under a wide range of deployment conditions, including different inference engines, context lengths, system prompts, and chat templates. Robustness under varying scenarios is critical to maintaining high acceptance rates and consistent inference speedups. However, recent works have shown that drafters often degrade substantially under challenging deployment scenarios, such as template perturbations and long-context inputs [3, 4]. As a result, practitioners frequently retrain or specialize drafters for specific inference engines, prompts, or templates to maximize performance. Despite speculative decoding models being relatively cheap to train, this sensitivity points to an underlying issue. Our three main contributions are as follows,

(i) In this work, we identify a previously unreported phenomenon that helps explain this fragility, which we call **attention drift** (Figure 1). As a drafter generates more tokens during speculation, its attention progressively shifts away from the sink toward recently generated tokens. We observe this phenomenon consistently across EAGLE-3 drafters [5] and Multi-Token Prediction (MTP) heads [6], suggesting that attention drift reflects a broader property of auto-regressive drafter designs.

(ii) To understand this behavior, we analyze the hidden-state dynamics of speculative drafters and find that the unnormalized residual connection between speculation steps causes hidden-state magnitudes to grow monotonically with chain depth, resembling additional transformer layers stacked on top of the verifier. The drafter implicitly learns a depth-dependent refinement process instead of a stable autoregressive token predictor, making it sensitive to template changes and long contexts (Section 4.2).

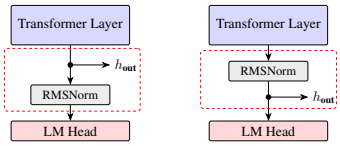


Figure 2: Overview of Pre-norm (Left) and proposed Post-norm (Right) speculative decoding.

(iii) Motivated by this observation, we introduce a simple architectural modification based on *post-normalization* (Figure 2) combined with a normalization before hidden-state fusion, that prevents hidden-state magnitude growth and stabilizes the drafting process. These changes substantially reduce attention drift and improve acceptance length over the current state-of-the-art pre-norm EAGLE3 architecture by up to $2\times$ under template perturbation, $1.18\times$ on long-context tasks, and $1.10\times$ across seven standard benchmarks spanning multi-turn chat, reasoning, coding, and mathematics. We further show that post-norm drafters generalize to inference depths beyond those seen during training, enabling shorter train-time-test depths and reducing training cost.

2 Preliminaries

Speculative Decoding alternates between two stages (Figure 3). In the drafting phase, the drafter auto-regressively generates a sequence of k candidate future tokens, where k is the *speculation depth*. In the verification phase, the target model evaluates the drafted sequence in a single forward pass and applies rejection sampling against its own token distribution; the longest valid prefix is accepted. A key efficiency metric is the *acceptance length* (τ), defined as the average number of drafted tokens accepted per verification round. Higher acceptance lengths generally translate directly into larger end-to-end inference speedups for a fixed drafter overhead.

EAGLE [7] and its successors EAGLE-2/EAGLE-3 [8, 5] feed the target's fused internal hidden states through a fully-connected projection (FC in Figure 9) into a single pre-norm transformer decoder layer [9] that serves as the drafter (Figure 2 Left). The drafter is trained with a token-level cross-entropy loss against the frozen target's predicted distribution, with a fixed number of speculation steps (the train-time-test depth, TTT). EAGLE-3, our focus, is the dominant auto-regressive drafter design in production engines such as vLLM [10] and SGLang [11].

Attention Sinks. An attention sink [12] is a token (typically near the start of a sequence) that absorbs disproportionately large attention during inference. Sinks are widely observed in modern LLMs and are believed to act as stable anchors that stabilize attention, particularly under long-context.

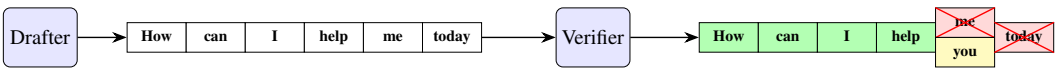
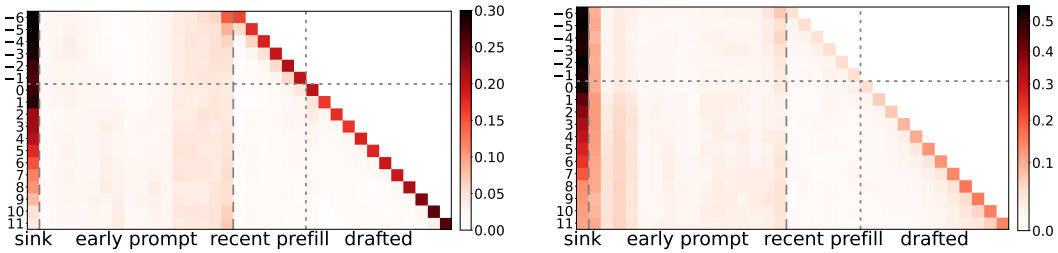


Figure 3: Verification phase: green tokens are accepted, yellow is resampled and red ones are rejected.

3 Attention Drift



(a) Qwen3.5 35B (b) Llama3 8B.

Figure 4: Attention heatmaps for visualizing attention drift on EAGLE-3 drafters. Aggregated over 200+ samples from varied prompts and sequence positions.

Looking at speculative decoding models' attention, a pattern emerges: **whenever the target model develops a sink, the drafter develops a sink at exactly the same place**. This observation is consistent across **MTP heads** (Appendix **??**) and **EAGLE-3** drafters. Upon close inspection of three model families with different attention patterns, we observed Llama's sink is on the first token, Qwen's on the second, and GPT-oss has no sink. In every case, the drafter matched its target.

The drafter attention resembles the verifier attention as the drafter consumes verifier hidden states. However, once drafting begins, the attention distribution progressively changes as the drafter transitions from consuming verifier hidden states to consuming its own generated hidden states. During this process, attention increasingly shifts away from the sink and toward recently generated tokens.

We refer to this phenomenon as **attention drift**, visualized in Figures 4 and 5 across multiple EAGLE-3 drafters. In models with strong sinks, the sink progressively weakens during drafting while attention mass concentrates on recently drafted tokens. The drafter therefore operates in two modes: attending to the verifier's hidden states at pre-fill and to its own hidden states during generation, with a gradual learned transition between them. We hypothesize that this transition makes the drafter **more fragile to out-of-distribution inputs**, most notably in long context and at deeper speculation steps.

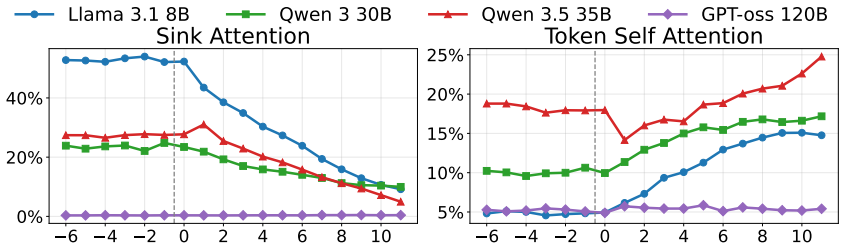


Figure 5: Percentage of attention concentrated on Sink token (Left) and Latest Token (Right) on EAGLE-3 drafters.

MTP Heads

Here, we examine MTP Heads [6] as a cross-architecture check on whether attention drift is specific to EAGLE-3 or a more general property of drafters that consume their own previous outputs. MTP Heads are auxiliary prediction heads jointly trained with the target model during pretraining, sharing the target's LM head and predicting multiple positions ahead. This differs from EAGLE-3, which is trained post-hoc as a separate drafter on top of a frozen target with its own LM head.

We inspect the MTP head of Qwen3.5 9B, a single transformer layer that follows the target model's attention architecture and reuses the same weights across consecutive speculation steps. During MTP speculation, we observe similar trends (Figure 6): attention to the sink token decreases substantially, while attention to the most recently drafted token increases.

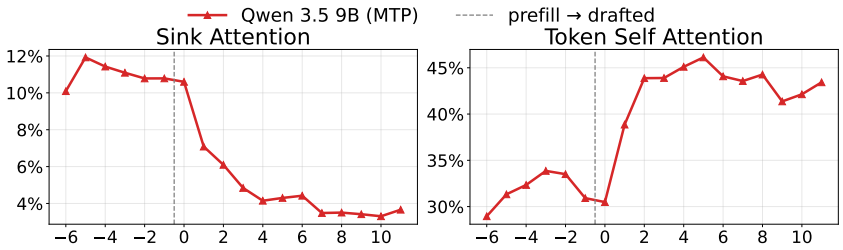


Figure 6: Attention drift on MTP heads. Sink and drafted token self attention on Qwen3.5 9B MTP heads (MT-Bench, 80 prompts).

4 Experiments: What Causes Attention Drift?

To understand why attention drift occurs and what it reveals about the underlying model, we trained drafters from three different verifier model families. We primarily evaluate on MT-Bench [13], eight categories of multi-turn instructions allows us to evaluate across diverse prompts. All architectural changes explored in this section required re-training of the drafter (details in Appendix D).

4.1 Hidden State Magnitudes

Attention drift occurs as we move away from the target's hidden states towards consuming the drafter's own hidden states during generation. Therefore we first inspect the hidden states of the verifier and drafter. In Table 1, the magnitudes are compared between the verifier's hidden states, verifier's fused hidden state and the drafter's own hidden states at different speculation steps, using root mean square (RMS), $\|x\|_2/\sqrt{d}$. We notice several patterns described next.

Table 1: Hidden state magnitudes (RMS) across model families, averaged over 80 MT-Bench prompts and k=8 speculation rounds. Hidden states from the verifier's low, middle, and high layers (h_{low}, h_{mid}, h_{high}); the fused representation of the verifier's hidden states h_{FC}; and the drafter's output hidden states at speculation steps 1, 2, 3, and 8 (h_1^* through h_8^*)

	Verifier hidden states				Drafter output h_t^*			
Target Model	h_{low}	h_{mid}	h_{high}	h_{FC}	k=1	k=2	k=3	k=8
Llama 3.1 8B	0.56	0.58	0.78	12.46	3.92	4.87	5.86	14.02
Qwen 3 30B	0.33	2.17	2.71	2.56	1.00	1.13	1.25	1.67
Qwen 3.5 35B	0.03	0.12	0.21	0.89	3.47	4.03	4.42	5.92
GPT-oss 20B	3	52	324	96	87	89	95	138
GPT-oss 120B	3	163	**1455**	583	497	511	537	**647**

Observation 1: Hidden-state magnitudes mismatch. The hidden states of the verifier, target, and the fully connected layer have substantially different magnitudes. This is attributed to the positioning of the RMSNorm in the drafter and a norm component being shared between (h_{FC}) and the drafter (h^*) shown in Figure 9. Despite the hidden state magnitudes being vastly different between the target and drafter, we did not see a correlation between the magnitude of hidden states and acceptance rates for models, hinting us that the model learns to compensate for it.

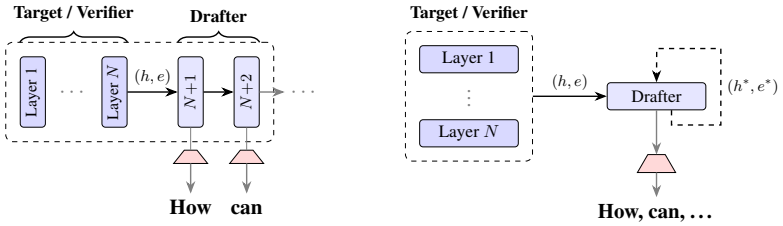
Observation 2: The verifier fused representation, h_{FC}, is imbalanced. All three target families we explored use the pre-norm architecture: RMSNorm is applied inside each transformer block (before attention and MLP), but never to the residual stream itself. As such, each layer's magnitudes accumulate monotonically through depth (prenorm dilution [14]), and $\|h_1\| < \|h_2\| < \|h_3\|$ holds for every row of Table 1. The target's final pre-LM-head RMSNorm normally absorbs this growth when producing logits, but EAGLE-3 uses verifier's internal states captured before this norm (h_{low}, h_{mid}, h_{high}). Because of this h_{high} dominates the h_{FC} signal with its larger magnitude. Thus in future sections **we add RMSNorm layers before each target hidden stream** used as input to generate h_{FC} to prevent this imbalance and generate a more stable verifier representation.

Observation 3: Magnitude growth across speculation depth. The drafter-generated hidden states grow monotonically with speculation depth. Across all model families, h_k^* accumulates magnitude away from the distribution the drafter was trained on at step 0. This means that the drafter does not operate on a depth-invariant hidden-state distribution. Instead, each speculation step changes the scale of the representation consumed by the next step.

How is drift and magnitude related? This scaling magnitude across speculation depth is important in computing attention. Attention depends on computing query-key similarities. As the drafter generates deeper into the speculation chain, its current hidden state increasingly resembles recently generated drafter states rather than verifier-conditioned prompt states. The empirical consequence, quantified in the next section, is a *redistribution* of attention mass away from the original input and onto the drafter's own predictions.

4.2 Norm Position Changes: Layer-stacking vs Autoregression

The monotonic growth of the drafter hidden states suggests a simple interpretation of what a drafter *learns to be*. In a standard pre-norm drafter, the drafter's chain hidden states $h_1^*, h_2^*, \ldots, h_k^*$ accumulate through an unnormalized residual path, causing the hidden states to monotonically grow with speculation depth. This makes the drafter behave less like an independent autoregressive model and more like a sequence of additional transformer layers stacked on top of the target. At speculation step k, the drafter effectively approximates what would happen if the target had $N, N{+}1, \ldots, N{+}k$ layers, rather than acting as a standalone depth-invariant model that autoregressively predicts the next token. The model takes the target's representations and *keeps refining it* with another layer of attention+MLP compute, one per speculation step (Figure 7).



(a) With Pre-norm (b) With Post-norm

Figure 7: Two views of the EAGLE3 drafter at chain depth > 1. With the standard pre-norm structure, the model acts more like additional layers stacked onto the target model, whereas post-norm acts more like an independent auto-regressive drafter that accepts its own output.

To test this interpretation, we compare standard pre-norm and the modified post-norm drafters trained with different train-time-test depths. Train-time-test depth denotes the maximum speculation depth used during training. If a drafter learns a depth-invariant autoregressive rule, then training with a small depth should still generalize reasonably to longer speculation chains at inference. In contrast, if the drafter learns depth-dependent transformations resembling layers $N, N{+}1, \ldots$, then it should fail beyond the depths observed during training. Figure 8 supports the latter interpretation for pre-norm drafters. A pre-norm drafter trained with train-time-test depth 2 performs well at the first few drafting steps but collapses beyond its training horizon. Its hidden-state magnitude grows rapidly, and its conditional acceptance probability drops sharply at deeper speculation steps.

Drafter / TTT	h_1^*	h_8^*	$k{=}1$	$k{=}16$	Accept.
Pre-norm / 8	4.15	13.10	0.91	0.50	**7.12**
Pre-norm / 2	9.76	44.52	**0.93**	0.00	2.64
Post-norm / 2	1.22	1.30	**0.93**	**0.53**	5.05

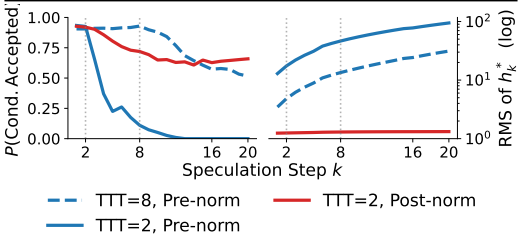


Figure 8: Pre-norm vs. Post-norm at various TTT (max training depth). Magnitude and per-step acceptance on coding (HumanEval) and math (GSM8K), $k{=}20$ steps.

By contrast, the post-norm drafter trained with the same TTT depth remains stable beyond the training horizon. Applying post-normalization to the drafter hidden state between speculation steps prevents

residual-scale accumulation, making the drafter less able to encode depth through magnitude growth. Rather than continuing the verifier as a depth-dependent stack of additional pre-norm transformer layers, the post-norm drafter is regularized toward a stable autoregressive prediction function.

4.3 Are gains attributed to the post-norm or the fc-norm?

To understand whether the acceptance length gains come from the post-norm architecture or the additional fc-norm, we have conducted an ablation on both components by training 4 models with the same configuration. Results in Table 2 show that neither component helps alone, while combining them improves acceptance length on all seven benchmarks. Training at TTT$= 2$ (Table 8) further shows that post-norm is the component responsible for generalizing beyond the trained depth.

Table 2: Component ablation on gpt-oss-20b (TTT$= 8$, temperature 0.7).

		Pre-norm		Post-norm	
Benchmark	n	w/o fc-norm	w/ fc-norm	w/o fc-norm	w/ fc-norm (ours)
MT-Bench	80	2.559	2.576	2.420	**2.688**
GSM8K	200	3.699	3.228	3.001	**4.002**
AlpacaEval	200	2.493	2.430	2.280	**2.558**
HumanEval	164	3.659	3.449	3.228	**3.961**
MATH-500	200	3.768	3.773	3.479	**3.962**
GPQA	200	2.575	**2.870**	2.712	2.760
LiveCodeBench	200	3.140	**3.443**	3.162	3.287
Mean		3.127	3.110	2.897	**3.317**

4.4 Does Fixing Attention Sink Solve the Drift and Magnitude Issue?

The previous section suggests that hidden-state magnitude accumulation is a major contributor to attention drift. However, attention drift could also be interpreted as a consequence of attention-sink collapse: if the sink weakens during drafting, attention mass must move elsewhere. In this section, we test whether eliminating the sink is sufficient to solve drift as well as the magnitude growth.

To test this, we compare several architectural variants on the same target: a standard pre-norm drafter, a gated-attention drafter, a post-norm drafter, and combining gated-attention and post-norm. The gated-attention variant introduces a learned element-wise gate inside the attention layer (Appendix B), which strongly suppresses reliance on a fixed sink token. Table 3 summarizes the results.

Table 3: Gated attention vs. normalization: chain magnitude of hidden states h_k^*, entropy H, sink and recent token attention across EAGLE3 drafters for Llama-3.1 8B on MT-Bench (80 prompts over 8 categories and all drafting rounds).

Drafter	$h^*_{1\to8}$	$H_{1\to8}$	$\text{sink}_{1\to8}$	$\text{recent}_{1\to8}$
Pre-norm	$3.92 \to 14.02$	$2.04 \to 2.73$	$0.46 \to 0.08$	$0.14 \to 0.31$
Gated-Attn	$8.41 \to 39.07$	$2.39 \to 2.13$	$0.03 \to 0.02$	$0.29 \to 0.32$
Post-norm	$1.21 \to 1.20$	$2.24 \to 2.29$	$0.11 \to 0.10$	$0.50 \to 0.53$
Gated + post-norm	$0.87 \to 0.88$	$0.63 \to 0.64$	$0.00 \to 0.00$	$0.50 \to 0.51$

Magnitude and attention are independent failure modes. Gated attention nearly eliminates sink attention and removes the visible drift pattern (≤ 0.03 throughout), but its hidden state magnitude grows $5.0\times$ during drafting and starts from a larger $h_1^* \approx 8.4$. On the other hand, post-norm pins the hidden-state magnitude across speculation steps and stabilizes the attention pattern over the drafting chain. These results indicate that attention drift and magnitude accumulation are independent failure modes and that **solving attention drift does not necessarily eliminate the magnitude issue.**

MTP Heads

We further analyze the MTP head of Qwen3.5 9B to test whether attention drift is specific to EAGLE-3. Unlike EAGLE-3, which trains a separate post-hoc drafter with its own LM head on top of a frozen target, MTP heads are integrated into the target model and trained jointly during pretraining. The Qwen3.5 MTP head consists of a single transformer layer that follows the target model's attention architecture, and the same head weights are reused across consecutive speculation steps.

Despite these architectural and training differences, we observe similar attention behavior. As shown in Figure 21, attention to the sink token decreases sharply during MTP speculation, while attention to the most recently drafted token increases and then stabilizes after a few steps. This transition is sharper than what we observe in pre-norm EAGLE-3 drafters, where drift typically progresses more gradually across the speculation chain. We also see the same observation on our gated attention variants in Figure 20, suggesting a unique interaction of the gated attention and post-norm architecture.

This is consistent with our finding that magnitude accumulation is one contributor to drift but not the only mechanism. We focus our diagnosis and intervention on EAGLE3, which trains a separate drafter with its own LM head post-hoc against the frozen target's output distribution. MTP, in contrast, reuses the target's LM head and is trained jointly with the target during pretraining. The exact training procedure for Qwen3.5's MTP heads has not been publicly disclosed; characterizing the role of training loss vs. architecture in the emergence of attention drift is left as future work.

5 Analysis: Performance Impact

In this section, we evaluate different solutions to the attention and magnitude drift problem. We report acceptance length excluding the bonus token except for SGLang benchmarks, to compare raw performance across speculative decoding models. We trained drafters for four target models spanning different architectures: Llama 3.1 8B (dense), Qwen 3 8B (dense thinking), Qwen 3.5 9B (GDN-hybrid thinking), and GPT-OSS 20B (sparse MoE thinking).

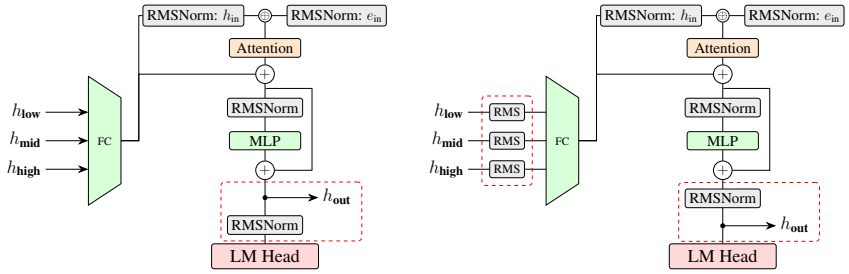


Figure 9: Standard *Pre-norm* (Left) vs proposed *Post-norm* (Right) drafter architectures.

The proposed post-norm architecture places individual RMSNorms after each target hidden state $h_{low}, h_{mid}, h_{high}$ and accumulates the drafter's hidden states *after* the RMSNorm (Figure 9). By using post-norm, we were able to reduce the TTT length from 8 to 4, cutting training time by roughly one third without impacting performance. We also trained baseline models using the same training data and configuration with the regular pre-norm architecture for a fair comparison. Post-norm improved acceptance length consistently across **all four models**, with gains ranging up to 12% (Figure 10, full per-model results in Appendix C).

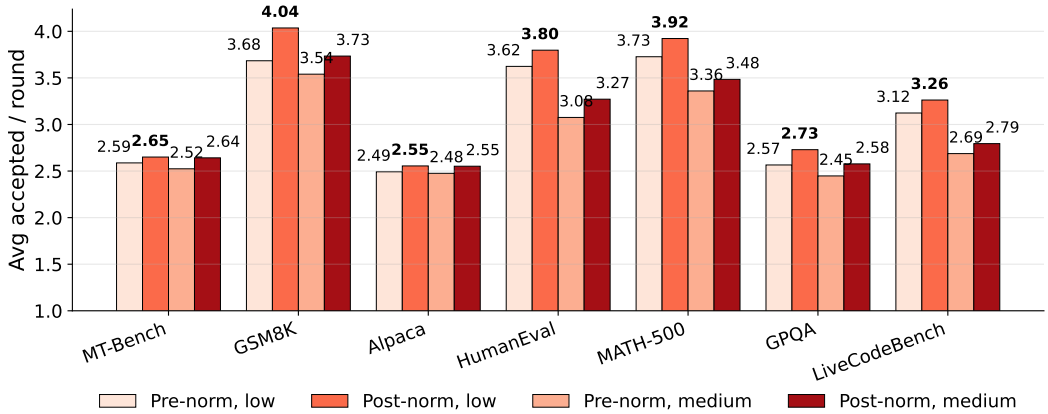


Figure 10: GPT-OSS 20B results on SGLang, acceptance length includes bonus token, temp $= 0.7$. Proposed post-norm shows **consistent improvements** over standard pre-norm.

5.1 Template Sensitivity

Performance of speculative decoding models varies heavily across different deployment settings, system prompts, and chat templates. Since drafters are not trained on raw pre-training data but on processed supervised fine-tuning (SFT) conversations, we observed they heavily overfit to certain elements of the chat template. Templates may be perturbed intentionally (e.g., disabling reasoning tags to save tokens) or unintentionally, as reported in inference engines and benchmarks. We test how well our post-norm architecture generalizes beyond the chat format and improves resilience to deployment conditions.

We designed two experiments. In the first, we removed the chat template and sink token to measure sensitivity (Appendix E). Post-norm was significantly more resilient to every kind of disruption, losing at most 5% accuracy in the worst case, whereas pre-norm dropped 52% (Figure 11).

In the second experiment with temperature 0 and varying system prompt length (Llama's default prompt, trimmed to different lengths), post-norm beat pre-norm at every length tested (Figure 12). More importantly, it was substantially more robust to length variation: post-norm degraded at most 5% from its peak, while pre-norm dropped 13%. Post-norm's worst case still beats pre-norm's best.

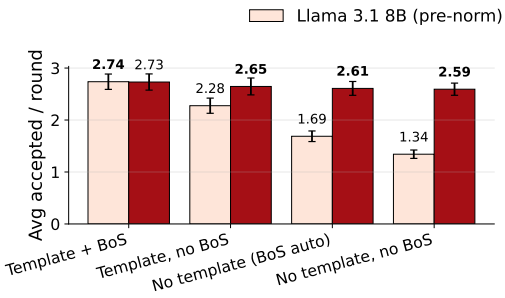


Figure 11: Template sensitivity of Llama 3.1 8B drafter pre-norm and post-norm. Temp = 0.7.

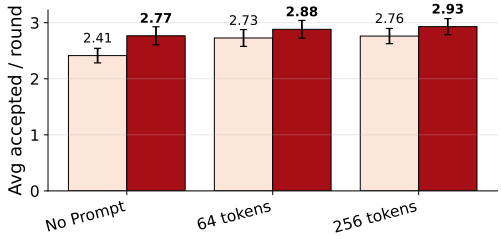


Figure 12: System prompt length's effect on accuracy for Llama 3.1 8B.

Is template sensitivity an artifact of attention sinks? To test this, we compared post-norm against a Gated Attention model, which eliminates the attention sink and, thus, the sensitivity to the beginning-of-sequence (BoS) token. The gated model(s) still showed high template sensitivity and in fact developed *higher* sensitivity to system prompt length, with lower overall accuracy (Figures 13 and 14). This shows that removing the sink alone is not enough to reduce template fragility and that fixing hidden-residual dynamics is required.

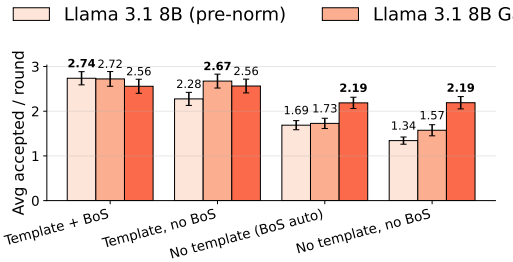


Figure 13: Gating is not sensitive to BoS, but it still overfits to other tokens in the template.

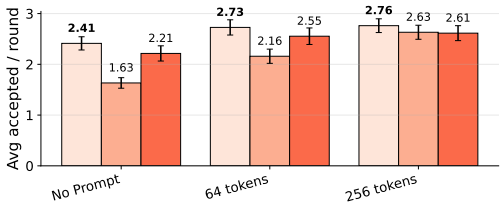


Figure 14: Gated attention overfits to the system prompt, hurting accuracy badly.

5.2 Long Context

Another challenge for speculative decoding models is long context. Drafters are relatively cheap to train and are usually trained with a short context window such as 4096 tokens, which makes them easier and more affordable to train. However, they fail catastrophically outside their trained context length. LLMs have developed various techniques to handle this, one being sliding window attention (SWA), where the model's effective context is a fixed window over the most recent tokens. This

makes intuitive sense for drafters in particular: they are weaker by design, and their predictions on easy tokens shouldn't depend on long-range context. Sliding window attention has been studied for long-context speculative decoding from both serving-systems [15] and drafter-training angles [4]. We use SWA as a long-context evaluation tool rather than as a method we propose; our post-norm fix is orthogonal to it and can be combined with existing long-context speculative decoding techniques.

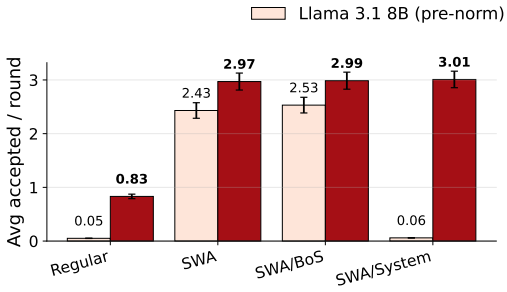


Figure 15: Accuracy of Llama 3.1 8B drafter on various long-context multi-turn chat settings.

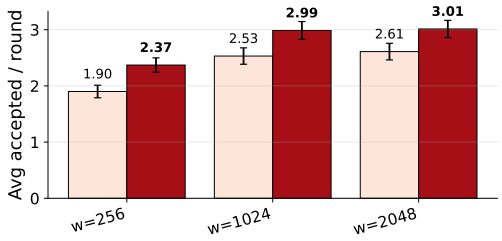


Figure 16: Effect of window size on prediction accuracy.

We created a multi-turn conversation benchmark with context length beyond our models' trained length. A multi-turn setup isolates the model's long-context capability from degradation caused by unrelated filler text (results summarized in Figure 17). Under **full attention** (regular), the pre-norm model fails catastrophically, dropping to 0.05 average acceptance length, while post-norm drops to 0.83, also unusable, but $15\times$ better. With **sliding window attention** (SWA), most of the accuracy is rescued: pre-norm recovers 88% of its single-turn baseline, while post-norm matches and slightly exceeds its single-turn performance. Since pre-norm develops a strong sink, **carrying the BoS token** (SWA/BoS) through the window helps it more (up to 91% recovery), with only a minor bump for post-norm. Following on this, we hypothesized that **carrying the system prompt** (SWA/System) would add additional context. For post-norm this added another 1%, but pre-norm collapsed to near-zero, it could not accommodate the wider range of positional embeddings introduced by the longer prefix. Across all SWA conditions, post-norm outperformed pre-norm by 20%.

True long-context performance. In addition to our multi-turn benchmark, we evaluate on *LongBench* across three task categories: summarization, few-shot learning, and coding. Absolute task scores are lower than in the multi-turn setting due to the inherent difficulty of long-context understanding. Post-norm outperformed pre-norm by 20–25% across all categories and context lengths (Table 4). On the government report summarization task, adding the system prompt yielded an additional 8% gain, suggesting that informative prefixes can further help SWA-based drafters recover context, though we leave a broader study of this effect to future work.

Table 4: LongBench: avg accepted draft tokens per round, per task and SWA mode. Window=1024.

Task	SWA + BoS		SWA + Sys Prompt	
	Pre-norm	Post-norm	Pre-norm	Post-norm
`Gov Report (Summarization)`	1.35	1.57	0.47	**1.70**
`Samsum (Few-Shot)`	1.04	**1.31**	0.39	**1.31**
`Repobench (Coding)`	1.85	**2.26**	0.26	2.25

How much does the drafter rely on long-range context vs. recent tokens? To answer this, we varied the SWA window size (Figure 16). Across all three window sizes tested, post-norm maintained a consistent gap over pre-norm. Even a small 256-token window recovers 80% of the full-context baseline; performance saturates around 1024 tokens, with diminishing returns beyond that. This suggests the model mostly cares about recent tokens and SWA can substantially reduce compute without meaningfully hurting drafter accuracy.

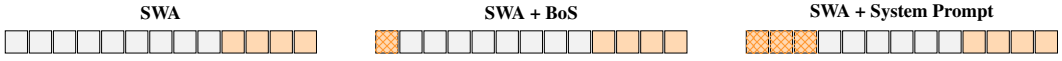


Figure 17: Attended tokens (marked in orange) in different SWA implementations.

5.3 Performance on Real Deployments

To verify that acceptance gains translate into wall-clock speedups despite the added normalization, we deploy GPT-oss 120B drafters in vLLM and SGLang on a B200 GPU across concurrency levels 1, 4, and 16 (Table 5). Even with a naive implementation without fused normalization kernels, post-norm is faster than pre-norm at every setting on GSM8K (up to $2.16\times$ vs. $2.04\times$ in vLLM) and faster or on par on MT-Bench (up to $1.42\times$ vs. $1.28\times$).

Table 5: End-to-end speedup over vanilla AR decoding for gpt-oss-20b (B200, temp 0.7)

		Accept		vLLM		SGLang	
Benchmark	Conc.	Pre	Post	Pre	Post	Pre	Post
GSM8K	1	4.14	**4.35**	1.95×	**2.01**×	1.72×	**1.82**×
	4	4.13	**4.44**	2.04×	**2.16**×	1.66×	**1.76**×
	16	4.15	**4.40**	1.86×	**2.03**×	1.58×	**1.68**×
MT-Bench	1	2.56	**2.61**	1.31×	1.31×	1.24×	**1.28**×
	4	2.48	**2.65**	1.29×	**1.32**×	1.30×	1.30×
	16	2.54	**2.64**	1.28×	**1.42**×	1.33×	1.33×

6 Related Work

Speculative decoding accelerates LLM inference by drafting candidate tokens and verifying them in parallel [1]. Several drafter designs have been proposed: Medusa [16] attaches multiple prediction heads to the target, Hydra [17] introduces sequential dependence between these heads, and D-Flash [18] uses diffusion models to predict sequences in one forward pass. We focus on EAGLE-3 as the dominant auto-regressive drafter design.

Attention sinks and patterns. Xiao et al. [12] was the first to observe attention sinks and show that preserving these sinks is essential for long-context perplexity under sliding-window attention. Qiu et al. [19] proposed gated attention as a mechanism to reduce the model's reliance on a fixed sink token. Both works study sink behavior in the target model itself; we are the first, to our knowledge, to characterize attention behavior in speculative decoding drafters and identify a distinct failure mode (attention drift) specific to the chain-residual structure of these models.

Normalization in transformers. The placement of normalization relative to residual connections has been studied extensively in the standard transformer setting. Xiong et al. [20], Wu et al. [21] showed that pre-norm enables more stable training than post-norm, leading most modern LLMs to adopt pre-norm. Our work revisits this question for speculative decoding drafters, where iterative self-feedback across speculation steps changes the dynamics: we show that post-norm becomes necessary to prevent magnitude accumulation that drives attention drift.

7 Conclusion and Limitations

Our study has two main limitations. First, we focus on EAGLE-3 as the dominant post-training auto-regressive drafter in production. We observe attention drift in MTP heads, suggesting the phenomenon extends beyond EAGLE-3, but we do not investigate the alternative mechanisms at play there or evaluate our fix across other drafter designs. Extending the analysis to MTP, Hydra, and other variants is a natural direction for future work. Second, our experiments are limited to models below 120B parameters due to compute constraints; behavior at larger scale remains unverified.

We identified *attention drift* in auto-regressive speculative decoding drafters: as the drafter generates successive tokens during speculation, attention progressively migrates from the sink onto recently-generated tokens. We characterized the phenomenon across model families and showed it is closely tied to monotonic magnitude growth in the un-normalized residual path between speculation steps, which causes the drafter to behave more like additional transformer layers stacked on the target than as a standalone auto-regressive predictor. Two simple architectural interventions, post-norm on the drafter output and RMSNorm before the target hidden projection, address the underlying dynamics and yield consistent improvements, with the largest gains under deployment shift. We hope that naming attention drift gives the community a useful diagnostic for drafter robustness, and that post-norm proves a useful default for production systems.

Acknowledgments and Disclosure of Funding

References

[1] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[2] 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.

[3] Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, junxian guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context LLM inference with retrieval and streaming heads. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cFu7ze7xUm.

[4] Penghui Yang, Cunxiao Du, Fengzhuo Zhang, Haonan Wang, Tianyu Pang, Chao Du, and Bo An. Longspec: Long-context lossless speculative decoding with efficient drafting and verification. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025. URL https://openreview.net/forum?id=GFN9PWbfHs.

[5] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=4exx1hUffq.

[6] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 15706–15734. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/gloeckle24a.html.

[7] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: speculative sampling requires rethinking feature uncertainty. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 28935–28948. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/li24bt.html.

[8] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 7421–7432, 2024.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[10] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph 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*, pages 611–626, 2023.

[11] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37: 62557–62583, 2024.

[12] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

[13] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

[14] Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, et al. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026.

[15] Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. In *ICLR*, 2025. URL https://openreview.net/forum?id=CS2JWaziYr.

[16] 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 *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=PEpbUobfJv.

[17] Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=FbhjirzvJG.

[18] Jian Chen, Yesheng Liang, and Zhijian Liu. Dflash: Block diffusion for flash speculative decoding. *arXiv preprint arXiv:2602.06036*, 2026.

[19] Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 100092–100118. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/904e89bb4e632e75fb47f093b620b257-Paper-Conference.pdf.

[20] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR, 2020.

[21] Xinyi Wu, Amir Ajorlou, Yifei Wang, Stefanie Jegelka, and Ali Jadbabaie. On the role of attention masks and layernorm in transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=lIH6oCdppg.

[22] Shenggui Li, Chao Wang, Yikai Zhu, Yubo Wang, Fan Yin, Shuai Shi, Yefei Chen, Xiaomin Dong, Qiaoling Chen, Jin Pan, Ji Li, Laixin Xie, Yineng Zhang, Lei Yu, Yonggang Wen, Ivor Tsang, and Tianwei Zhang. Specforge: A flexible and efficient open-source training framework for speculative decoding, 2026. URL https://arxiv.org/abs/2603.18567.

A Attention Drift

In this appendix, we provide additional attention-drift visualizations and analyze whether the phenomenon appears beyond EAGLE3 drafters.

Figures 18 and 19 show attention behavior for Qwen3.5 35B and GPT-oss 120B drafters. Unlike Llama and Qwen models with strong attention sinks, GPT-oss does not develop a clear sink token. Nevertheless, we occasionally observe weaker attention concentration on recurring template tokens. This suggests that some apparent sink-like behavior may arise not only from architectural inductive bias, but also from repeated special tokens or template markers in the supervised fine-tuning (SFT) distribution. Similar mild concentration over template tokens has also been observed in some target LLMs.

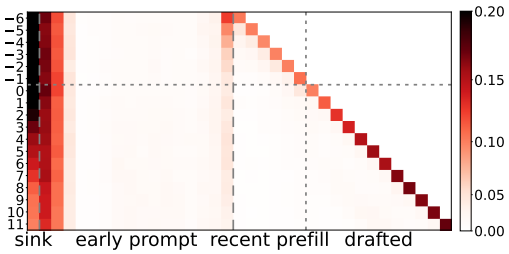


Figure 18: Attention Drift on Qwen3.5 35B

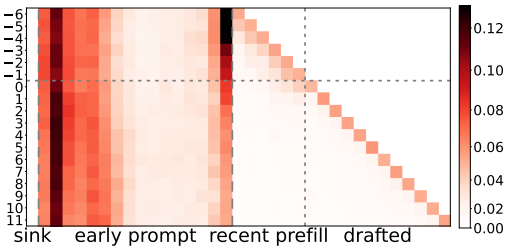


Figure 19: Attention Drift on GPT-oss 120B.

We further inspect the attention patterns of the models trained to mitigate attention drift (post-norm and gated attention). As shown in Figure 20, these variants stabilize both sink attention and self-token attention across speculation steps. Gated-attention models suppress the sink more aggressively and settle into different absolute attention levels, but they do not exhibit progressive drift across the drafting chain. In these models, we observe a difference between prefill attention and autoregressive drafting attention, but not a continuous redistribution of attention within the generation steps themselves.

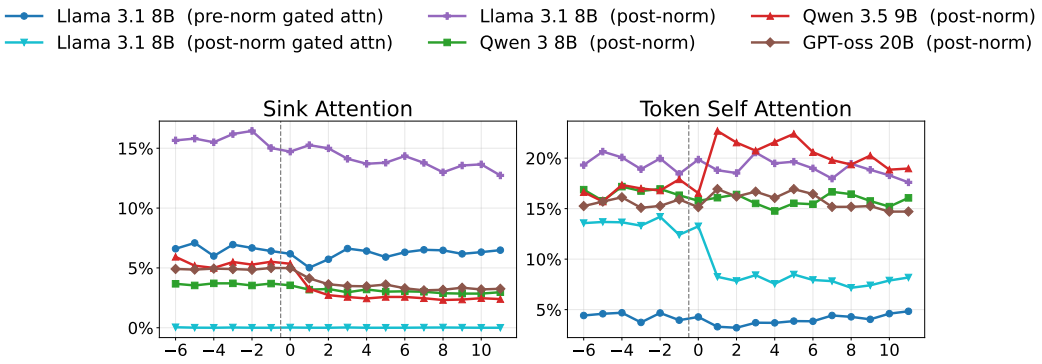


Figure 20: Attention Sink and Self-Token Attention for each model that fixes the attention drift.

A possible additional contributor to attention drift is how the speculative decoding models are trained. Most EAGLE-style trainers use a fixed context window during train-time testing (TTT) and repeatedly drop the oldest tokens as drafting proceeds. This may encourage the drafter to reduce reliance on early prompt positions, including the sink token, as speculation depth increases. While our main results show that hidden-state magnitude accumulation is an important contributor to drift, this training-window effect may further amplify sink weakening. We leave a more detailed study of the interaction between training-window construction and attention drift to future work.

Does shrinking magnitude fix attention drift? We observed fixing the drift does not necessarily fix the magnitude accumulation. However we are not sure whether fixing the magnitude standalone would fix attention drift. We have created an inference-time controlled experiment, the hidden states of drafter (h_{out}) were normalized to match the fc's magnitude during inference. Table 6 show that

normalizing the magnitude without a matched training hurts the accuracy in both cases, pre-norm being effected even more than post-norm. However we observe one interesting thing, the attention drift is significantly lessened on Pre-norm architecture, while still happening weakly. This shows that magnitude accumulation is one of the contributors to the attention drift but not only.

Table 6: Effect of pinning the hidden-state RMS to the FC's RMS on Llama 3.1 8B during test time.

Model	Mode	Accept	ΔAcc.	Sink k=0	Sink k=7	Chain k=7
Pre-norm	Regular	3.06	–	0.40	**0.14**	0.31
Pre-norm	Normed	1.33	−56%	0.41	0.29	0.24
Post-norm	Regular	3.15	–	0.11	0.11	0.35
Post-norm	Normed	2.09	−34%	0.11	0.13	0.33

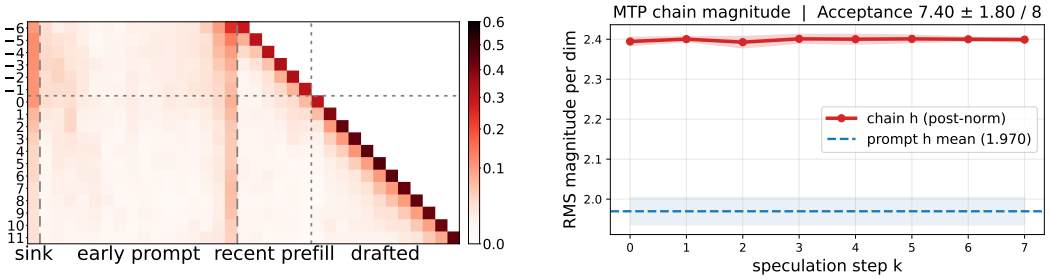


Figure 21: Attention and Magnitude drift on Qwen3.5 9B MTP heads

A.1 Noise Resiliency and Error Accumulation

Table 7: Drafter pathway reliance under noise injection. Each cell reports acceptance length as a percentage of the model's no-perturbation baseline (parenthesized in the model column). Noise is scaled per-tensor RMS: $x \leftarrow x + \alpha \cdot \mathrm{rms}(x) \cdot \varepsilon$.

Model	Target	α=0.1	α=0.25	α=0.5	α=1.0	α=2.0	α=∞
Pre-norm (3.06)	Hidden States	82%	28%	5%	0%	0%	0%
	Embeddings	99%	93%	86%	64%	27%	9%
Post-norm (3.16)	Hidden States	99%	86%	58%	22%	5%	2%
	Embeddings	98%	100%	93%	75%	38%	9%

The drafter consumes its own predictions as inputs to subsequent speculation steps, and these self-generated hidden states are noisier than the clean verifier states seen at t=1. The α-noise sweep therefore test how gracefully each architecture absorbs imprecise hidden-state inputs, determine how quickly errors compound with depth. Post-norm tolerates an order of magnitude more perturbation than pre-norm on the hidden pathway (58% vs. 5% at α=0.5), offering a mechanistic explanation for its better deep-chain behavior: it accumulates less error per speculation step. We further hypothesize that this tolerance may translate to robustness under other small hidden-state perturbations, such as those induced by verifier quantization or mild distribution shift.

Is post-norm responsible for depth generalization?

To further understand whether the depth generalization was caused by adding the fc-norm or switching to the post-norm architecture, we have trained different models with TTT= 2 using 4 different configurations. The results at Table 8 clearly show that pre-norm with fc-norm can't generalize beyond depth 2 any better than pre-norm only. This is a strong evidence on post-norm being the mechanism that helps the model to generalize beyond the trained depth.

Table 8: Depth generalization with a drafter trained at TTT$= 2$.

	fc-norm		no fc-norm	
Draft length	Pre-norm	Post-norm	Pre-norm	Post-norm
2	2.319	**2.331**	2.319	2.312
8	2.812	3.138	2.741	**3.238**

Table 9: Hidden-state magnitudes after the fix on Qwen3. Post-norm keeps magnitude flat across depth.

	FC fusion				Chain step k									
Variant	low	mid	high	out	0	1	2	3	4	5	6	7	k_7/k_0	Accept
Pre-norm	0.264	28.294	30.588	27.748	16.54	18.96	21.58	24.22	26.83	29.50	32.02	34.63	2.09×	1.750
Post-norm	0.436	0.668	0.959	4.010	1.252	1.158	1.131	1.120	1.117	1.115	1.114	1.113	0.89×	**1.834**

Role of Entropy

We also examine whether attention entropy explains drafter failures. Based on our measurements, the role of entropy is not clear in predicting drafter failures. Despite having different base entropy levels for all pre-norm models, the per-token attention entropy *rises* slightly as the k grows. The distribution becomes differently-*shaped*, instead of peakier. However a direct relation between per-token entropy and the acceptance length could not be formed in Pre-norm, Post-norm or gated attention variants, showing the model learns to compensate for the distribution shift in token entropy.

Combining the *Gated Attention* with *Post-norm* creates a new pathology. The Post-norm + Gated variant combines both modifications, and it does what one would hope along each of the two individual axes: chain magnitude is flat, the sink is eliminated, and the recent token attention is stable at 0.50. But entropy collapses to $H \approx 0.62$ already at pre-fill, about a third of the other drafters' values, corresponding to an effective attention support of roughly two positions ($e^{0.62} \approx 1.85$). Applying both changes appears to over-regularize the drafter: the attention distribution collapses onto roughly two tokens.

Focusing on relationship between the target's attention entropy, acceptance rate and the drafter's entropy is left as a future work.

B Gated attention

In the gated attention variant of our models, we add an optional per-element sigmoid gate over the hidden dimension to the EAGLE3 draft self-attention. Let $\mathbf{x} \in \mathbb{R}^{T \times d_{\text{in}}}$ denote the input stream to the attention block. Because EAGLE3 fuses the verifier hidden state with the input-embedding inside the decoder layer, $\mathbf{x}$ is the concatenation of the two and $d_{\text{in}} = 2d$, where d is the draft hidden size. The query, key and value projections operate on $\mathbf{x}$ as in standard Llama, and we write the multi-head attention output *before* the output projection as

$$\mathbf{A} \;=\; \mathrm{Attn}(\mathbf{x}\mathbf{W}_Q, \mathbf{x}\mathbf{W}_K, \mathbf{x}\mathbf{W}_V) \;\in\; \mathbb{R}^{T \times h\, d_h},$$

with h heads of dimension d_h. Our modification introduces a learned gate $\mathbf{W}_g \in \mathbb{R}^{d_{\text{in}} \times h\, d_h}$ that shares its input with the QKV projections,

$$\mathbf{g} \;=\; \sigma\big(\mathbf{x}\mathbf{W}_g\big), \tag{1}$$

where $\sigma(\cdot)$ denotes the element-wise sigmoid ($\sigma(z) = \dfrac{1}{1+e^{-z}}$). The gate is applied to the concatenated head output and *precedes* the output projection $\mathbf{W}_O$:

$$\mathbf{y} \;=\; \big(\mathbf{g} \odot \mathbf{A}\big)\, \mathbf{W}_O. \tag{2}$$

C Benchmark Results

We report SGLang acceptance length (including the bonus token) per benchmark for each target model trained in this work. Tables 6–9 cover GPT-oss 20B, Qwen3.5 9B, Qwen3 8B, and Llama 3.1 8B respectively, sweeping the relevant decoding modes for each (low/medium reasoning effort for GPT-oss; thinking/no-thinking for Qwen3 and Qwen3.5). Across all four models and every decoding mode, post-norm matches or improves on pre-norm; the single regression is Llama 3.1 8B on MATH-500, which falls within evaluation noise. Gains are largest on math and code (e.g., GSM8K +10% on GPT-oss low, HumanEval +5% on Qwen3 8B no-think) and smallest on Alpaca-style open-ended chat, consistent with the harder, longer-horizon completions benefiting more from a stable drafter.

Table 10: SGLang acceptance length for **GPT-oss 20B** + EAGLE3 across pre/post-norm draft and low/medium reasoning effort. Best per row in bold.

Benchmark	Pre, low	Post, low	Pre, medium	Post, medium
MT-Bench	2.59	**2.65**	2.52	2.64
GSM8K	3.68	**4.04**	3.54	3.73
Alpaca	2.49	**2.55**	2.48	2.55
HumanEval	3.62	**3.80**	3.08	3.27
MATH-500	3.73	**3.92**	3.36	3.48
GPQA	2.57	**2.73**	2.45	2.58
LiveCodeBench	3.12	**3.26**	2.69	2.79

Table 11: SGLang acceptance length for **Qwen3.5 9B** + EAGLE3 across pre/post-norm draft and thinking/no-thinking decoding. Best per row in bold.

Benchmark	Pre, no-think	Post, no-think	Pre, think	Post, think
MT-Bench	3.08	3.16	3.60	**3.69**
GSM8K	4.21	4.28	4.32	**4.42**
Alpaca	2.73	2.79	3.49	**3.57**
HumanEval	4.47	**4.49**	4.43	4.43
MATH-500	5.08	**5.12**	4.95	5.01
GPQA	3.54	3.65	3.78	**3.87**
LiveCodeBench	4.31	**4.36**	4.22	4.30

Table 12: SGLang acceptance length for **Qwen3 8B** + EAGLE3 across pre/post-norm draft and thinking/no-thinking decoding. Best per row in bold.

Benchmark	Pre, no-think	Post, no-think	Pre, think	Post, think
MT-Bench	3.55	**3.61**	3.09	3.15
GSM8K	5.58	**5.73**	3.73	3.83
Alpaca	3.14	**3.35**	2.68	2.82
HumanEval	5.00	**5.21**	3.05	3.13
MATH-500	5.80	**5.89**	3.57	3.64
GPQA	3.91	**4.04**	2.98	3.08
LiveCodeBench	4.59	**4.71**	3.06	3.14

Table 13: SGLang acceptance length per benchmark for **Llama 3.1 8B** with EAGLE3 (steps=7, topk=1, draft=8). Best per row in bold. The single regression on MATH-500 is within evaluation noise.

Benchmark	Pre-norm	Post-norm
MT-Bench	3.84	**3.90**
GSM8K	5.28	**5.33**
Alpaca	3.52	**3.59**
HumanEval	5.56	**5.59**
MATH-500	**5.87**	5.81
GPQA	4.74	**4.78**
LiveCodeBench	4.16	**4.22**

C.1 Model Template Sensitivity

Another factor in template sensitivity may lie in how the loss is constructed during training. In the EAGLE series, the loss is computed only on assistant tokens; user tokens don't contribute to the loss directly. So while parameters are shared and trained on the full sequence, the model's outputs at user positions are not directly constrained, only their utility for downstream assistant-position predictions is. This asymmetry could show up especially when the chat template is changed and the boundary between supervised and unsupervised positions change from where the model was trained to expect it.

Table 14: **Llama 3.1 8B**: Avg accepted draft tokens per verification round under different prompt-template manipulations on MT-Bench.

Condition	Pre-norm	Post-norm	Pre-norm Gated	Post-norm Gated
template + BoS	**2.74** $\pm$ 0.15	2.73 $\pm$ 0.16	2.72 $\pm$ 0.16	2.56 $\pm$ 0.16
template, no BoS	2.28 $\pm$ 0.15	2.65 $\pm$ 0.16	**2.67** $\pm$ 0.16	2.56 $\pm$ 0.15
no template (BoS auto)	1.69 $\pm$ 0.10	**2.61** $\pm$ 0.13	1.73 $\pm$ 0.11	2.19 $\pm$ 0.13
no template, no BoS	1.34 $\pm$ 0.08	**2.59** $\pm$ 0.12	1.57 $\pm$ 0.12	2.19 $\pm$ 0.14

Table 15: **Qwen3-8B**: Avg accepted draft tokens per verification round under different prompt-template manipulations on MT-Bench.

Condition	Pre-norm	Post-norm
template + BoS	1.80 $\pm$ 0.07	**1.90** $\pm$ 0.07
template, no BoS	1.76 $\pm$ 0.06	**1.87** $\pm$ 0.06
no template (BoS auto)	1.71 $\pm$ 0.11	**2.15** $\pm$ 0.12
no template, no BoS	1.64 $\pm$ 0.09	**2.21** $\pm$ 0.10

Table 16: **GPT-oss 20B**: Avg accepted draft tokens per verification round under different prompt-template manipulations on MT-Bench.

Condition	Pre-norm	Post-norm
template + BoS	1.66 $\pm$ 0.13	**1.80** $\pm$ 0.14
template, no BoS	1.55 $\pm$ 0.11	**1.73** $\pm$ 0.13
no template (BoS auto)	1.43 $\pm$ 0.07	**1.75** $\pm$ 0.07
no template, no BoS	1.29 $\pm$ 0.05	**1.65** $\pm$ 0.07

C.2 Model Prompt Length Sensitivity

All models are trained with a fixed system prompt. For Llama, the default system prompt is around 500 characters, so the acceptance rate increases with longer system prompts, as the model is used to seeing such inputs during training. This also directly affects which positional embeddings the model sees during training. As discussed in the template-sensitivity section, loss is only computed on

assistant tokens, so the region where the system prompt lives is not directly supervised. However, the *Post-norm* variant is the most resilient to changes in system-prompt length: for example, the *Pre-norm* variant loses around 14% accuracy with a short system prompt, whereas *Post-norm* loses only 5%. Qwen3 and GPT-OSS, in contrast, are trained with fairly short system prompts of around 30–40 characters. We believe this contributes to their resilience to system-prompt length changes, and may explain why these models prefer system prompt lengths around 64 tokens.

Table 17: **Llama 3.1 8B**: Avg accepted draft tokens per verification round as the prepended system context grows from 0 to 256 tokens.

Condition	Pre-norm	Post-norm	Gated Pre-norm	Gated Post-norm
no system	2.41 ± 0.13	**2.77** ± 0.16	1.63 ± 0.10	2.21 ± 0.15
system 64 tok	2.73 ± 0.15	**2.88** ± 0.16	2.16 ± 0.14	2.55 ± 0.16
system 256 tok	2.76 ± 0.14	**2.93** ± 0.14	2.63 ± 0.14	2.61 ± 0.15

Table 18: **Qwen3-8B**: Avg accepted draft tokens per verification round as the system context grows from 0 to 256 tokens.

Condition	Pre-norm	Post-norm
no system	1.82 ± 0.06	**1.94** ± 0.06
system 64 tok	1.92 ± 0.06	**2.00** ± 0.07
system 256 tok	1.89 ± 0.07	**1.96** ± 0.07

Table 19: **GPT-oss 20B**: Avg accepted draft tokens per verification round as the prepended system context grows from 0 to 256 tokens.

Condition	Pre-norm	Post-norm
no system	1.86 ± 0.13	**2.01** ± 0.14
system 64 tok	1.94 ± 0.12	**2.06** ± 0.13
system 256 tok	1.85 ± 0.12	**2.02** ± 0.14

C.3 Model Long-Context SWA Sensitivity

To our surprise, the Gated Post-Norm model did not suffer at all from out-of-distribution inference lengths and showed no accuracy loss in the full-context attention case. This could be attributed to its low attention entropy: it learns to attend to only a couple of tokens, and this sharpened attention pattern may filter the destructive signal via the sigmoid gate. This also suggests the model relies less on absolute positional information than the baseline does.

Table 20: **Llama 3.1 8B**: Avg accepted draft tokens per round at 8k context, comparing full attention to sliding-window variants with optional BoS / system-prompt carry.

Condition	pre-norm	post-norm	Gated pre-norm	Gated post-norm
full attn	0.05 ± 0.00	0.83 ± 0.04	0.84 ± 0.06	**2.74** ± 0.17
SWA	2.43 ± 0.14	**2.97** ± 0.16	1.21 ± 0.09	2.70 ± 0.17
SWA + BoS	2.53 ± 0.15	**2.99** ± 0.16	1.22 ± 0.09	2.69 ± 0.17
SWA + system	0.06 ± 0.00	**3.01** ± 0.15	1.72 ± 0.12	2.70 ± 0.17

Table 21: **GPT-oss 20b**: Avg accepted draft tokens per round at 16k context, comparing full attention to sliding-window variants.

Condition	Pre-norm	Post-norm
full attn	**1.67** ± 0.09	1.67 ± 0.09
SWA	1.09 ± 0.08	**1.94** ± 0.11

Table 22: **Qwen3-8B**: Avg accepted draft tokens per round at 8k context, comparing full attention to sliding-window variants with optional BoS / system-prompt carry.

Condition	Pre-norm	Post-norm
full attn	**1.33** $\pm$ 0.04	1.24 $\pm$ 0.04
SWA	1.87 $\pm$ 0.07	**2.05** $\pm$ 0.07
SWA + BoS	1.97 $\pm$ 0.07	**2.08** $\pm$ 0.07
SWA + system	1.87 $\pm$ 0.07	**2.05** $\pm$ 0.07

C.4 SWA Size / Accuracy

Section 5.2 reports the window-size sweep for Llama 3.1 8B at 8k context. Table 20 confirms the same pattern holds for Qwen3 8B at a 32k target context: post-norm leads pre-norm at every window size, and gains saturate around w=1024 with only marginal improvement at w=2048.

Table 23: **Llama-3.1 8B**: Avg accepted draft tokens per round under SWA + BoS carry as the window size grows from 256 to 2048 (target context 8192).

Condition	Pre-norm	Post-norm
w=256	1.90 $\pm$ 0.11	**2.37** $\pm$ 0.13
w=1024	2.53 $\pm$ 0.15	**2.99** $\pm$ 0.16
w=2048	2.61 $\pm$ 0.15	**3.01** $\pm$ 0.15

Table 24: **Qwen3-8B**: Avg accepted draft tokens per round under SWA + BoS carry as the window size grows from 256 to 2048 (target context 32768).

Condition	Pre-norm	Post-norm
w=256	1.64 $\pm$ 0.08	**1.82** $\pm$ 0.08
w=1024	1.97 $\pm$ 0.09	**2.14** $\pm$ 0.09
w=2048	2.00 $\pm$ 0.09	**2.16** $\pm$ 0.09

C.5 LongBench Results

We further break down LongBench-E results for Llama 3.1 8B by prompt-length bucket, using each architecture's preferred SWA mode (SWA+BoS for pre-norm, SWA+System Prompt for post-norm. Post-norm outperforms pre-norm in every bucket from 0–4k all the way to 32–36k. This indicates that the post-norm advantage seen on the multi-turn benchmark in Section 5.2 transfers to true long-context tasks rather than being specific to the multi-turn setting.

Table 25: **Llama 3.1 8B**, **LongBench-E**: avg accepted draft tokens per round vs prompt length, averaged over all tasks. Each model uses its preferred SWA mode (pre-norm: SWA + BoS, post-norm: SWA + Sys Prompt). n is the number of verification rounds in each bucket.

Ctx (tokens)	Pre-norm (SWA + BoS)		Post-norm (SWA + Sys Prompt)	
	Acceptance Length (τ)	Number of samples	Acceptance Length (τ)	n
0-4k	1.60	7838	**1.99**	6802
4-8k	1.63	11146	**2.07**	9563
8-12k	1.30	9927	**1.73**	8354
12-16k	1.28	6242	**1.60**	5482
16-20k	1.82	3000	**1.98**	2839
20-24k	1.94	3855	**1.96**	3831
24-28k	1.77	2744	**2.20**	2374
28-32k	1.73	1560	**2.25**	1309
32-36k	1.21	542	**1.80**	428

D Training

We have used a modified instance of the *SpecForge* [22] repository to train our models. This repository implements the Train-Time-Test method for EAGLE-3 training and it is the framework used to train current state-of-the-art EAGLE-3 models.

Llama and Qwen3 variants are trained on the Open-PerfectBlend dataset with regenerated answers using target model. The dataset consists over 1.4M samples. We have run it for 2 epochs and 1.5×10^{-4} LR, with effective batch size of 4. On average training took around 48 H200 hours for each model with 8-9B parameters.

Qwen3.5 and Gpt-oss variants were trained on Nemotron post training dataset with regenerated answers using target model. The dataset consist over 1.4M samples. The maximum sequence length was 8K tokens. We have run it for 1 epoch and 1.5×10^{-4} LR. On average training took around 36 to 48 H200 hours based on model size (20B and 120B).

E Template Perturbations

```
--- regular ---
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a helpful assistant.<|eot_id|><|start_header_id|>user<|end_header_id|>

What is the capital of France?<|eot_id|><|start_header_id|>assistant<|end_header_id|>


--- no_bos ---
<|start_header_id|>system<|end_header_id|>

You are a helpful assistant.<|eot_id|>...

--- no_template ---
<|begin_of_text|>Question: What is the capital of France?
Answer:

--- no_bos_no_template ---
Question: What is the capital of France?
Answer:
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract includes the main claims made in the paper supporting them with empirical evidence and actual numbers.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations, in the *Conclusion and Limitations* section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include proofs or theorems.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The architectural changes and datasets used to train the model is fully described and supported by many open source tools. Moreover authors will open-source the trained models and their code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code used to run experiments and the trained models will be shared.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The training and test details for the drafters are explained in detail. Each time a new experiment is introduced, the dataset and experiment settings are clearly demonstrated.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Results are reported as mean ± standard deviation over multiple runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Total training time and required resources for each model is explained in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, research conforms the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work is related to acceleration of large-language-models without altering their behavior. Even though LLMs have huge societal impacts, our work only focuses on optimizing the runtime of the LLMs, which would create neither positive nor negative impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The models are only used to speed-up LLM inference and does not alter or change the behaviour of the models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The authors have acknowledged the licenses of used datasets and models correctly.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper will only share the experimental results and models used to reproduce those results.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are used extensively to review and refactor code, run experiments, create visualizations, literature review and proof-reading for the paper.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.