Performance-Driven Policy Optimization for Speculative Decoding with Adaptive Windowing

Jie Jiang[1*] **Xing Sun**[2*] **Ruotian Chen**[1] **Jianan Su**[1†] **Kaixin Shen**[1†]

[1]Ant International, Ant Group
[2]Fliggy, Alibaba Group

{youzhen.jj, ruotian.crt, jianan.sja, kaxin.skx}@ant-intl.com
ruoxing.sx@alibaba-inc.com

Abstract

Speculative decoding accelerates LLM inference by having a lightweight draft model propose speculative windows of candidate tokens for parallel verification by a larger target model. In practice, speculative efficiency is often bottlenecked by hard-to-draft positions, where an early mismatch truncates the accepted prefix and invalidates the rest of the speculative window. Most learning-based drafters are still optimized with token-level supervised objectives, even though speculative utility is inherently window-level and prefix-sensitive. We propose **PPOW** (**P**erformance-Driven **P**olicy **O**ptimization with Adaptive **W**indowing), a reinforcement learning framework that shifts drafter optimization from token-level imitation to window-level optimization. PPOW combines a Cost-Aware Speedup Reward, a Distribution-Based Proximity Reward, and Adaptive Divergence-Aware Windowing, which prioritizes informative windows with high confidence-weighted draft–target divergence. PPOW achieves average acceptance lengths of 6.29–6.52 and speedups of 3.39–4.36$\times$ across multiple model families and benchmarks under a unified decoding protocol. These results show that performance-driven window-level optimization is a practical approach to improving speculative decoding efficiency.

1 Introduction

Speculative decoding [1, 2] accelerates Large Language Model (LLM) generation while preserving the output distribution of the target model. At each speculative step, a lightweight draft model (drafter) proposes a *speculative window* of candidate tokens, which the target model then verifies in parallel. In practice, the realized speedup is often limited by hard-to-draft positions. Even when most tokens within a speculative window are well modeled by the drafter, a single early mismatch can truncate the accepted prefix and invalidate the rest of the window, limiting overall throughput.

This behavior exposes a mismatch between the inference-time goal of speculative decoding and the training objectives commonly used for drafters. Recent methods, including MEDUSA [3], Hydra [4], EAGLE [5, 6, 7], HASS [8], and GRIFFIN [9], improve drafter quality through architectural design and supervised training, while distillation-based approaches [10, 11] further improve distributional alignment with the target model. However, these methods are still primarily based on token-level supervised objectives. Such objectives improve local next-token prediction, but speculative utility is inherently *window-level* and *prefix-sensitive*: once verification is truncated at an early position, the remaining drafted tokens in the speculative window are invalidated. As a result, improvements

*These authors contributed equally to this work.
†Corresponding authors.

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

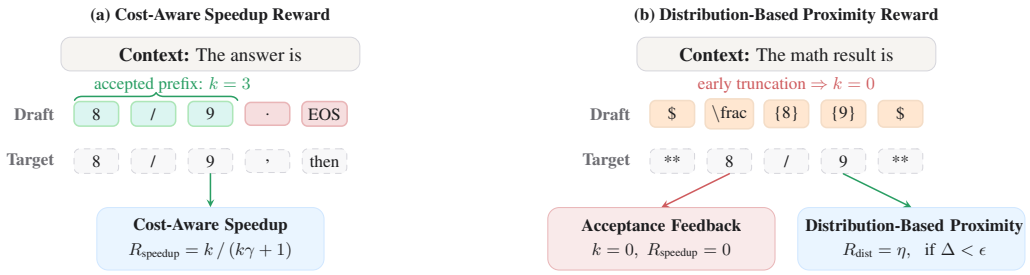


Figure 1: **PPOW uses a Cost-Aware Speedup Reward together with a Distribution-Based Proximity Reward.** (a) The Cost-Aware Speedup Reward increases with accepted prefix length and directly encourages speculative decoding efficiency. (b) When verification is truncated early, resulting in $k = 0$, the Distribution-Based Proximity Reward still provides auxiliary credit if the speculative window remains close to target preferences.

in token-level imitation may not consistently lead to longer accepted prefixes or higher speculative efficiency, especially when end-to-end performance is governed by a few acceptance bottlenecks.

We propose **PPOW** (**P**erformance-Driven **P**olicy **O**ptimization with Adaptive **W**indowing), a reinforcement learning framework that shifts drafter training from token-level supervision to window-level optimization over speculative windows. Figure 2 provides an overview of PPOW. This window-level formulation aligns optimization with the prefix-sensitive structure induced by speculative verification. Concretely, PPOW combines a *Cost-Aware Speedup Reward*, which encourages longer accepted prefixes while accounting for relative drafter cost, with a *Distribution-Based Proximity Reward*, an auxiliary signal that preserves partial credit for speculative windows that remain close to the target model's preferences even when the accepted prefix is truncated early.

Although PPOW optimizes with window-level rewards, not all speculative windows are equally informative during training. Treating all windows uniformly can disperse optimization effort across non-bottleneck windows, limiting focus on the acceptance bottlenecks most critical to speculative efficiency. PPOW therefore introduces *Adaptive Divergence-Aware Windowing* (ADAW), which prioritizes windows with large confidence-weighted draft–target divergence. This criterion is grounded in our analysis of potential acceptance bottlenecks in Appendix B, and our experiments show that prioritizing such windows improves speculative performance.

Our contributions are summarized as follows:

- **A Window-Level RL Framework for Drafter Optimization.** We formulate drafter training for speculative decoding as a reinforcement learning problem over speculative windows, which better matches inference-time acceptance behavior.
- **Performance-Driven Reward Design.** PPOW combines a Cost-Aware Speedup Reward with an auxiliary Distribution-Based Proximity Reward to better align training with speculative decoding; the latter provides additional credit for speculative windows that remain close to the target distribution despite early truncation of the accepted prefix, as illustrated in Figure 1.
- **Adaptive Divergence-Aware Windowing.** We introduce a window prioritization strategy based on confidence-weighted draft–target divergence that focuses training on more informative windows associated with acceptance bottlenecks. Our analysis and experiments show that prioritizing such windows improves speculative performance.

2 Related Work

Recent work has improved speculative decoding through advances in drafter design, draft–target alignment, and inference-time decoding optimization. One approach is to improve the drafter itself. Head-based methods, including MEDUSA [3] and Hydra [4], augment the model with auxiliary draft heads for multi-token prediction. Feature-based methods including EAGLE [5, 6, 7], HASS [8], and GRIFFIN [9], improve drafting by leveraging hidden-state representations from the target model. Draft–target alignment has also been improved through distillation or online adaptation, as in DistillSpec [10], OSD [11], and FastDraft [12]. Lookahead decoding [13] provides a training-free

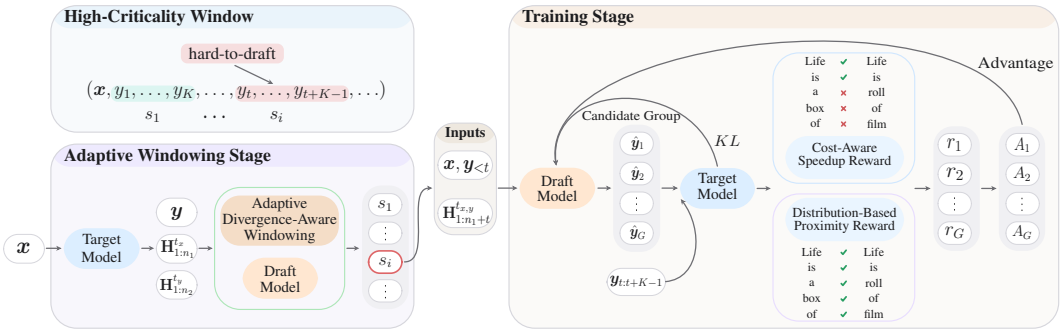


Figure 2: **Overview of PPOW.** PPOW performs policy optimization at the window level for speculative decoding. **Left:** Adaptive windowing uses confidence-weighted draft–target divergence scores to prioritize informative training windows. **Right:** The drafter samples a candidate group of speculative windows for policy optimization with performance-driven Rewards and KL regularization.

alternative by constructing future candidates through Jacobi-style parallel decoding. These approaches are primarily trained with token-level prediction, distillation, or local distribution-matching objectives, rather than objectives defined over speculative windows.

Prior work has also improved speculative decoding at inference time through multi-candidate selection and verification algorithms [14, 15], tree-based speculative inference and verification [16], tree-structured and hardware-aware speculation [17], cascaded or adaptive speculation strategies [18, 19, 20], and parallel draft–verify execution mechanisms [21, 22]. These approaches mainly operate on the inference procedure, such as how draft candidates are constructed, organized, or verified, and are complementary to drafter-training methods.

Reward signals and reinforcement learning have also been explored in speculative decoding. RSD [23] uses an inference-time reward model to guide draft acceptance, introducing a reward-guided verification mechanism that differs from standard rejection-sampling-based speculative decoding. Related efforts include SPEC-RL [24], RLHFSpec [25], and ReSpec [26], which adapt speculative decoding to improve the efficiency of RL training rather than optimizing the drafter for standard speculative decoding inference.

The closest methods explicitly incorporate speculative acceptance into drafter training. LK Losses [27] design per-position losses to improve acceptance probabilities, while VSD [28] formulates drafter training as variational inference over latent paths and updates the drafter with an acceptance-aware posterior in an EM-style procedure. In contrast, PPOW formulates drafter optimization as reinforcement learning over speculative windows, using performance-driven rewards derived from realized verification outcomes together with adaptive prioritization of hard-to-draft windows.

3 Preliminaries

We define the notation for speculative decoding used in this paper. Let π_{target} and π_θ denote the target model and the draft model (drafter), respectively, where θ parameterizes the drafter. Given an input prefix $\boldsymbol{x} = (x_1, \ldots, x_{n_1})$, the drafter proposes a speculative window of K candidate tokens,

$$\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_K).$$

The target model then performs parallel verification with rejection sampling under the standard speculative decoding procedure, resulting in an accepted prefix of length $k \in \{0, \ldots, K\}$.

3.1 Feature-Based Drafting

In this paper, we use feature-based drafters from the EAGLE family [5, 6, 7] as the base drafters in our experiments. Feature-based drafters augment token-based drafting with hidden-state features from the target model. Specifically, at drafting step t, the drafter predicts the next-token distribution as

$$\pi_\theta(\hat{y}_t \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}, \mathbf{H}),$$

where

$$\mathbf{H} = \left[\mathbf{H}^{\text{tgt}}_{1:n_1}, \mathbf{H}^{d}_{<t}\right]$$

denotes the concatenation of the target model's prefix hidden states $\mathbf{H}^{\text{tgt}}_{1:n_1}$ and the drafter's hidden states $\mathbf{H}^{d}_{<t}$ within the current speculative window. Further details of the rejection-sampling verification procedure are provided in Appendix C.1.

3.2 Policy Optimization over Speculative Windows

In contrast to standard policy optimization formulations that operate on full responses, we formulate drafter training over fixed-length speculative windows. For each input prefix $\boldsymbol{x}$, the drafter defines a policy over speculative windows $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_K)$, and a scalar reward $R(\boldsymbol{x}, \hat{\boldsymbol{y}})$ evaluates the window as a whole. The objective is

$$J(\theta) = \mathbb{E}_{\hat{\boldsymbol{y}} \sim \pi_\theta}\left[R(\boldsymbol{x}, \hat{\boldsymbol{y}})\right].$$

This formulation matches the speculative decoding procedure, in which a speculative window is verified as a whole and its utility is determined by the resulting accepted prefix. Appendix C provides further discussion of window-level policy optimization versus token-level training for speculative decoding.

4 PPOW

PPOW is a performance-driven training framework for speculative drafters that optimizes window-level behavior for speculative decoding. Concretely, PPOW uses Adaptive Divergence-Aware Windowing to prioritize informative speculative windows, assigns rewards using the Cost-Aware Speedup Reward, complemented by the Distribution-Based Proximity Reward when accepted-prefix-based signals are sparse, and updates the drafter through window-level reinforcement learning. Figure 2 gives an overview of PPOW, and Algorithm 1 summarizes the full training procedure.

4.1 Window-Level Reinforcement Learning for Speculative Decoding

PPOW formulates drafter training as reinforcement learning over speculative windows, so as to better align optimization with the inference mechanism of speculative decoding. In speculative decoding, the drafter produces speculative windows and the target model verifies them in parallel, yielding accepted-prefix outcomes for different speculative windows under the same prefix. Since these outcomes determine speculative efficiency, PPOW optimizes at the window level and learns from the relative speculative utility of speculative windows.

For a given input prefix $\boldsymbol{x}$, PPOW samples G_{roll} speculative windows $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_K)$ from the drafter policy, forming a rollout group for the same prefix. This grouping can improve accepted-prefix outcomes by considering multiple speculative windows for the same prefix, and the extent of this improvement depends on the quality of the speculative windows in the group. PPOW therefore assigns each speculative window a scalar reward and normalizes rewards within the rollout group to produce group-relative advantages. Each speculative window is thus treated as a single training unit: it receives one window-level scalar reward, which is converted into a group-relative advantage. This grouped optimization provides a relative learning signal and remains aligned with practical speculative decoding settings that consider multiple speculative windows.

In addition, PPOW incorporates a KL regularization term into the objective. We treat the drafter as the policy π_θ and the target model π_{target} as a cross-scale distributional anchor, and use the KL divergence $D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{target}})$ to encourage alignment during RL exploration. This target-anchored regularization is intended to stabilize training while keeping policy updates aligned with the target distribution used for speculative verification.

The resulting objective can be written as

$$\begin{aligned} J(\theta) = & \frac{1}{G_{\text{roll}}} \sum_{i=1}^{G_{\text{roll}}} \frac{1}{K} \sum_{t=1}^{K} \Big[\min\Big(r_{i,t}(\theta)\hat{A}_i,\ \text{clip}\big(r_{i,t}(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}\big)\hat{A}_i \Big) \\ & - \beta\, D_{\text{KL}}\big(\pi_\theta(\cdot \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{i,<t}, \mathbf{H}) \,\|\, \pi_{\text{target}}(\cdot \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{i,<t})\big) \Big], \end{aligned}$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(\hat{y}_{i,t} \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{i,<t}, \mathbf{H})}{\pi_{\text{old}}(\hat{y}_{i,t} \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{i,<t}, \mathbf{H})},$$

and $\hat{A}_i$ denotes the normalized group-relative advantage of the i-th speculative window, capturing its relative speculative utility within the rollout group. Specifically, for a rollout group of speculative windows sampled under the same prefix, we compute

$$\hat{A}_i = \frac{R_i - \mu_R}{\sigma_R + \delta},$$

where R_i is the scalar reward of the i-th speculative window, and μ_R and σ_R are the mean and standard deviation of rewards within the rollout group.

Although the policy is autoregressively factorized over tokens, optimization is performed at the speculative-window level, with all tokens within the same window sharing a common advantage signal.

4.2 Adaptive Divergence-Aware Windowing

Window-level training better matches speculative decoding at inference time, but it also leads to substantial sample redundancy. A full response can be decomposed into many overlapping speculative windows, and many of them are already adequately modeled by a supervised-initialized drafter. At the same time, speculative performance is often constrained by certain hard-to-draft positions, where draft–target mismatch can more strongly affect the accepted prefix. PPOW therefore prioritizes speculative windows that are both more informative for training and more consequential for speculative acceptance length through Adaptive Divergence-Aware Windowing (ADAW).

Let P_t and Q_t denote the target and drafter distributions at position t,

$$P_t = \pi_{\text{target}}(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t}), \qquad Q_t = \pi_\theta(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t}, \mathbf{H}),$$

and define the token-level *criticality score* as

$$v_t = C(P_t) \cdot D_{\text{KL}}(P_t \,\|\, Q_t),$$

where

$$C(P_t) = 1 - \frac{H(P_t)}{\log |\mathcal{V}|},$$

and $\mathcal{V}$ denotes the vocabulary, and $H(P_t)$ is the Shannon entropy of the target distribution P_t.

The criticality score v_t is a confidence-weighted measure of draft–target divergence. Here, $D_{\text{KL}}(P_t \,\|\, Q_t)$ captures mismatch between the drafter and target distributions, while $C(P_t)$ emphasizes contexts in which the target distribution is more concentrated and speculative verification is more sensitive to such mismatch. In this way, ADAW emphasizes contexts that are both difficult for the drafter and more consequential for speculative acceptance.

Because speculative decoding operates on complete windows, PPOW aggregates token-level criticality over each speculative window:

$$s_j = \frac{1}{K} \sum_{t=j}^{j+K-1} v_t,$$

and prioritizes windows with larger s_j during training. This adaptive windowing strategy reduces redundancy while focusing optimization on windows more directly associated with acceptance bottlenecks. Appendix B analyzes the connection between draft–target divergence and speculative acceptance behavior, and further discusses the role of confidence weighting in prioritizing training windows.

4.3 Performance-Driven Rewards

PPOW uses a window-level reward with two components. The primary component is a Cost-Aware Speedup Reward that encourages longer accepted prefixes while accounting for both drafting and verification cost. The second component is a complementary Distribution-Based Proximity Reward that provides auxiliary credit when exact verification terminates early but the speculative window still achieves a similar cumulative target-model log-likelihood.

4.3.1 Cost-Aware Speedup Reward

The primary goal of speculative decoding is to improve inference efficiency. PPOW therefore uses a cost-aware reward based on the accepted prefix length k of a speculative window and the relative cost γ of the drafter:

$$R_{\text{speedup}} = \frac{k}{k\gamma + 1},$$

where $k \in \{0, \ldots, K\}$ is the accepted length and γ denotes the relative computational cost of π_θ with respect to π_{target}, estimated in our implementation by the ratio of non-embedding parameters.

This formulation accounts for the computational structure of speculative decoding by balancing accepted length against relative drafting cost. The term $k\gamma + 1$ combines the drafting cost and the target-model verification cost within a speculative step. Compared with using raw the accepted prefix length k alone, R_{speedup} provides a cost-aware objective intended to better reflect the efficiency trade-off in speculative decoding.

We use this formulation during training instead of rewards derived from directly measured speedup, since measured speedup depends on the execution environment and is therefore less suitable as a general speculative-window-level training reward. Appendix D compares the Cost-Aware Speedup Reward with a Measured-Speedup-Based alternative and shows that it preserves the same acceptance-related trend while remaining effective for optimization.

4.3.2 Distribution-Based Proximity Reward

The accepted-prefix reward alone can become sparse when verification terminates early, even if the drafted window remains broadly compatible with the target model's preferences. To provide auxiliary partial credit in such cases, PPOW introduces the Distribution-Based Proximity Reward, which compares the drafted window with a target-preferred window under the target model's cumulative log-likelihood. As shown in Figure 1(b), early verification truncation yields $k = 0$ when the drafter and target model produce different textual realizations of similar content.

For a drafted speculative window $\hat{\boldsymbol{y}}$, we construct a target-preferred reference window $\boldsymbol{y}$ autoregressively from the target model, and compare their cumulative log-probabilities under the target model:

$$\Delta = \sum_{t=1}^{K} \Big[\log \pi_{\text{target}}(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t}) - \log \pi_{\text{target}}(\hat{y}_t \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}) \Big],$$

where

$$y_t = \arg\max_y \pi_{\text{target}}(y \mid \boldsymbol{x}, \boldsymbol{y}_{<t}).$$

We then define

$$R_{\text{dist}} = \eta \cdot \mathbf{1}[\Delta < \epsilon],$$

where ϵ is a tolerance threshold, and η scales the contribution of this auxiliary reward. Appendix F studies sensitivity to the choice of η.

This auxiliary reward is designed to capture training signal that may be missed by accepted-prefix-based reward alone. Rather than serving as a primary optimization target, R_{dist} provides bounded partial credit to drafted windows whose cumulative target log-probabilities remain close to those of the target-preferred reference window. This term supplies additional signal while preventing the auxiliary term from dominating the primary speedup objective.

As a result, R_{dist} complements the accepted-prefix reward with a softer window-level signal and can provide denser feedback during training. Section 5.5 evaluates its overall contribution in ablations, and Appendix E further examines its effect on easy and hard windows.

5 Experiments

We evaluate PPOW on end-to-end speculative decoding performance across multiple model families and diverse task settings, compare it with continued supervised training, examine its behavior under different inference candidate group sizes, and ablate its reward design and divergence-aware windowing strategy. Additional analyses of the speedup proxy, easy/hard-window behavior, and broader baseline comparisons are provided in the appendix.

5.1 Experimental Setup

Models and Tasks. We evaluate PPOW on two model families: LLaMA-3 [29] (8B, 70B) and Qwen3 [30] (8B, 32B). PPOW is applied to feature-based EAGLE family [5, 6, 7], initialized from supervised checkpoints before RL optimization. We evaluate on MT-Bench [31] for multi-turn dialogue, HumanEval [32] for code generation, and GSM8K [33] for mathematical reasoning.

Metrics. We report two standard metrics for speculative decoding:

- **Average Acceptance Length (τ):** the average number of drafted tokens accepted per speculative verification step.
- **Speedup Ratio:** end-to-end wall-clock speedup relative to standard autoregressive decoding under the same hardware and software setup.

Baselines. We mainly compare with learned drafters, including EAGLE-3 [7] and GRIFFIN [9]. Optimization details and the unified decoding protocol are provided in Appendix A, and additional results for OSD [11] and FastDraft [12] are provided in Appendix G.

5.2 Main Results

Table 1 presents our main end-to-end speculative decoding results under a unified decoding protocol. The table covers two model families (LLaMA-3 and Qwen3), multiple model scales (8B, 32B, and 70B), three representative benchmarks spanning dialogue, code generation, and mathematical reasoning, and two temperature settings, while reporting both average acceptance length and end-to-end wall-clock speedup. Across this broad evaluation setup, PPOW achieves the best mean results for every model family under both temperature settings.

Table 1: **Main speculative decoding results.** Average acceptance length (τ) and speedup across models and benchmarks under a unified decoding protocol. L31, L33, and Q3 refer to LLaMA-3.1-Instruct, LLaMA-3.3-Instruct, and Qwen3, respectively.

Model	Method	MT-Bench		HumanEval		GSM8K		Mean	
		τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup
Temperature = 0.0									
L31-8B	GRIFFIN	5.14	2.58×	6.72	3.96×	5.98	3.38×	5.95	3.31×
	EAGLE-3	**5.53**	**2.91**×	6.63	3.92×	6.12	3.41×	6.09	3.41×
	PPOW	5.47	2.72×	**7.23**	**4.14**×	**6.50**	**3.52**×	**6.40**	**3.46**×
L33-70B	GRIFFIN	5.08	3.60×	6.69	4.78×	5.90	3.99×	5.89	4.12×
	EAGLE-3	5.12	3.63×	6.78	4.80×	5.93	4.02×	5.94	4.15×
	PPOW	**5.45**	**3.73**×	**6.96**	**4.82**×	**6.47**	**4.54**×	**6.29**	**4.36**×
Q3-8B	EAGLE-3	4.95	2.64×	6.68	3.40×	6.86	3.47×	6.16	3.17×
	PPOW	**5.58**	**3.02**×	**7.01**	**3.62**×	**6.97**	**3.54**×	**6.52**	**3.39**×
Q3-32B	EAGLE-3	5.25	3.47×	6.52	4.02×	6.21	3.94×	5.99	3.81×
	PPOW	**5.78**	**3.54**×	**6.91**	**4.16**×	**6.62**	**4.05**×	**6.44**	**3.92**×
Temperature = 1.0									
L31-8B	GRIFFIN	4.03	2.24×	6.01	3.29×	5.13	2.89×	5.06	2.81×
	EAGLE-3	**4.17**	**2.35**×	5.94	3.25×	5.16	2.89×	5.09	2.83×
	PPOW	4.12	2.29×	**6.33**	**3.37**×	**5.63**	**3.13**×	**5.36**	**2.93**×
L33-70B	GRIFFIN	4.70	3.38×	6.31	4.46×	5.54	3.87×	5.52	3.90×
	EAGLE-3	4.82	3.40×	6.28	4.43×	5.98	4.16×	5.69	4.00×
	PPOW	**5.14**	**3.52**×	**6.57**	**4.62**×	**6.29**	**4.49**×	**6.00**	**4.21**×
Q3-8B	EAGLE-3	4.00	2.41×	5.71	2.11×	5.52	2.89×	5.08	2.47×
	PPOW	**4.72**	**2.82**×	**5.96**	**2.33**×	**5.76**	**3.01**×	**5.48**	**2.72**×
Q3-32B	EAGLE-3	4.51	2.60×	5.53	2.59×	5.23	2.81×	5.09	2.67×
	PPOW	**4.95**	**2.81**×	**6.17**	**2.77**×	**5.61**	**3.01**×	**5.58**	**2.86**×

At temperature 0.0, PPOW achieves mean acceptance lengths of 6.29–6.52 and mean speedups of 3.39–4.36× across the evaluated model families, with consistent gains also observed under temperature 1.0. The consistent gains at temperature 1.0 further indicate that PPOW remains beneficial beyond near-greedy decoding, even under less concentrated output distributions. Beyond the mean results, PPOW improves both acceptance length and wall-clock speedup on most benchmark–model combinations, showing that PPOW improves drafting behavior in a way that translates into practical end-to-end efficiency gains rather than acceptance-only improvements.

The gains are especially pronounced on HumanEval and GSM8K. For example, on LLaMA-3.1-8B, PPOW improves HumanEval from $\tau = 6.63$ and $3.92\times$ speedup relative to EAGLE-3 to $\tau = 7.23$ and $4.14\times$, while on GSM8K it improves from $\tau = 6.12$ and $3.41\times$ to $\tau = 6.50$ and $3.52\times$. On MT-Bench, the improvements are more modest, which may be related to the higher diversity and uncertainty of valid continuations in open-ended dialogue. Additional results on broader natural-language tasks and supplementary comparisons with OSD [11] and FastDraft [12] are provided in Appendix G.

5.3 PPOW versus Continued Supervised Training

We compare PPOW with continued supervised training (CST) from the same EAGLE-3 [7] initialization. This allows us to examine both whether PPOW's gains can be matched by additional post-training compute under continued token-level training, and whether any remaining difference is better explained by the training objective itself. For CST, we consider two learning rates: 5e-5 from the EAGLE-3 setup and a smaller 5e-6 as a conservative control to probe whether continued token-level training can still yield further gains. For PPOW, we report the best-performing learning rate. Both methods start from the same checkpoint and are trained for matched numbers of update steps, the comparison helps distinguish gains from additional training alone from gains due to optimization that is more directly aligned with speculative acceptance behavior.

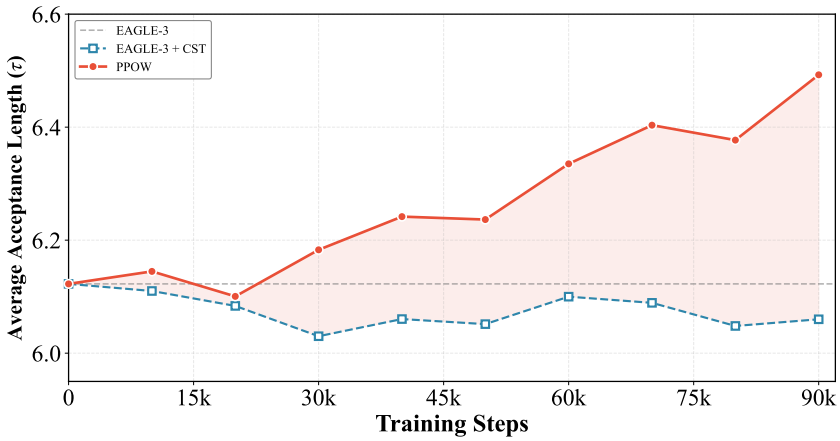


Figure 3: **PPOW versus CST under matched training steps.** On GSM8K with LLaMA-3.1-8B, CST with learning rate 5e-6 fluctuates around the original checkpoint's average acceptance length, while PPOW steadily improves throughout training.

Table 2: **Average acceptance length (τ) under continued training on GSM8K with LLaMA-3.1-8B.** Continued supervised training yields limited gains across both learning rates, while PPOW improves more steadily.

steps	EAGLE-3 (CST)		PPOW
	5×10^{-5}	5×10^{-6}	5×10^{-6}
30k	5.47	6.03	6.12
60k	5.61	6.10	6.32
90k	5.52	6.06	**6.49**

Table 2 reports GSM8K results with LLaMA-3.1-8B for CST at both learning rates and for PPOW under its best-performing learning rate, all starting from the same EAGLE-3 checkpoint. Over 30k / 60k / 90k training steps, CST with 5e-6 fluctuates around the original checkpoint's average acceptance length, reaching 6.03 / 6.10 / 6.06, while PPOW improves more steadily to 6.12 / 6.32 / 6.49. The higher CST learning rate of 5e-5 shows the same overall pattern. Figure 3 further illustrates this difference in training behavior, with CST remaining nearly flat while PPOW continues to improve.

These results indicate that EAGLE-3 is already close to saturation under continued token-level supervised training, with only limited gains from further optimization under the same objective. In contrast, PPOW benefits from optimizing an objective that is more directly aligned with acceptance length in speculative decoding. We discuss the objective-level difference between PPOW and continued supervised training in Appendix C, and observe the same qualitative trend on natural-language tasks, as shown in Appendix G.2.

5.4 Trade-offs with Different Inference Candidate Group Sizes

Table 3 evaluates the same trained PPOW model and the EAGLE-3 [7] across different inference candidate group sizes. In practice, speculative decoding often generates multiple candidate sequences via branching or tree decoding and verifies them in parallel. Larger candidate groups generally increase the chance of obtaining a longer accepted prefix, but they also incur higher verification cost. PPOW reaches strong acceptance with smaller candidate groups. At candidate group size 4, PPOW achieves $\tau = 6.33$, while EAGLE-3 reaches only $\tau = 5.03$ under the same setting and requires a group size of 16 to approach PPOW's performance. This result shows that PPOW can achieve better acceptance under a much smaller verification budget, highlighting its practical advantage in inference settings where verification overhead matters.

Table 3: **Average acceptance length (τ) under different inference candidate group sizes.** PPOW achieves higher acceptance with smaller candidate groups than the supervised baseline on LLaMA-3.1-8B / GSM8K.

Candidate Group Size	PPOW	EAGLE-3
1	4.92	4.10
2	5.23	4.46
4	**6.33**	5.03
8	6.41	5.58
16	6.49	**6.12**

5.5 Ablation Studies

We examine the contribution of PPOW's key components beyond the base speedup reward: the Distribution-Based Proximity Reward, Adaptive Divergence-Aware Windowing, and the confidence factor $C(P_t)$ used in ADAW. Table 4 reports component ablations on LLaMA-3.1-8B.

Table 4: **Ablation of PPOW components on LLaMA-3.1-8B.** "w/o R_{dist}" removes the Distribution-Based Proximity Reward, "w/o $C(P_t)$" removes the confidence factor in ADAW and uses divergence-only window prioritization, "w/o ADAW" replaces ADAW with uniform window sampling, and "w/o both" removes R_{dist} and uses uniform window sampling.

Method	MT-Bench		GSM8K	
	τ	Speedup	τ	Speedup
w/o R_{dist}	5.05	2.52×	6.41	3.48×
w/o $C(P_t)$	4.47	2.39×	5.92	3.12×
w/o ADAW	4.82	2.44×	6.35	3.37×
w/o both	4.38	2.39×	6.05	3.26×
PPOW (Full)	**5.47**	**2.72×**	**6.50**	**3.52×**

Removing either R_{dist} or ADAW reduces both acceptance length and speedup, and removing both further degrades performance on both benchmarks. On MT-Bench, removing both reduces τ from 5.47 to 4.38 and speedup from 2.72× to 2.39×. This suggests that the two components are complementary in practice: R_{dist} provides a denser learning signal when exact verification is sparse, and ADAW improves training efficiency by focusing optimization on more informative windows.

We further evaluate a divergence-only variant of ADAW by removing the confidence factor $C(P_t)$. This variant performs worse than both full ADAW and uniform window sampling. On GSM8K, removing $C(P_t)$ lowers τ from 6.50 to 5.92, compared with 6.35 under uniform sampling. This highlights the role of confidence weighting in encouraging ADAW to focus on bottleneck windows rather than ambiguous high-entropy mismatches.

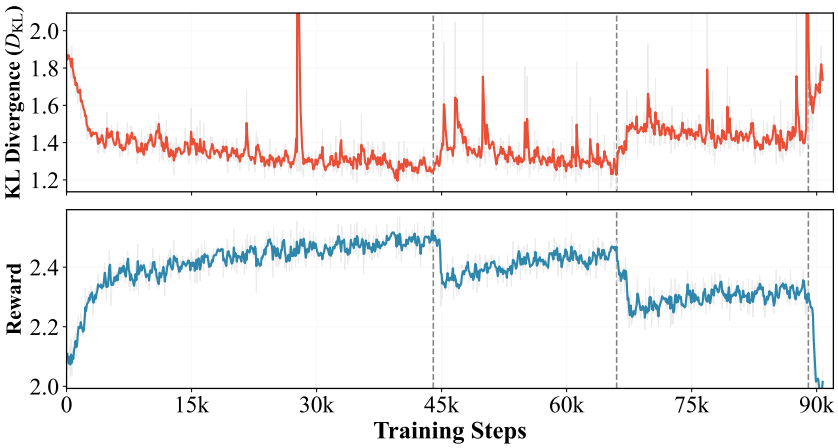


Figure 4: **Training dynamics under ADAW.** The dashed lines mark stage transitions where the hard-window ratio increases from 0.2 to 0.4 and then to 0.6. As more hard-to-draft windows are sampled, reward decreases and KL divergence rises.

Figure 4 further illustrates the training dynamics under ADAW. The dashed lines mark stage transitions in the hard-window curriculum. After 44k steps, training switches from uniform window sampling to ADAW-based sampling, and in later stages the hard-window ratio is further increased from 0.2 to 0.4 and then to 0.6, making the sampled training windows progressively harder. As a result, average reward exhibits stage-wise drops while KL divergence rises, reflecting shifts in the sampled training window distribution toward hard-to-draft windows rather than optimization failure. Within each stage, however, the policy gradually adapts to the harder sampled windows, and performance continues to improve as training proceeds.

Appendix A.4 describes the hard-window curriculum and the role of confidence weighting, and Appendix B provides analytical support for weighting draft–target divergence by target confidence. Appendix E provides additional ablations and easy/hard-window analysis of the Distribution-Based Proximity Reward, and Appendix F studies sensitivity to its weight η. Appendix H discusses the limitations and broader impacts of PPOW.

6 Conclusion

We introduced PPOW, a performance-driven reinforcement learning framework for speculative decoding that optimizes the drafter over speculative windows rather than individual tokens. PPOW combines performance-driven rewards and adaptive hard-to-draft window selection to better align drafter optimization with inference-time acceptance behavior. Across diverse settings, PPOW consistently improves both average acceptance length and speedup. It also remains effective with substantially smaller candidate group sizes at inference time, highlighting its practical value under constrained verification budgets. In addition, we provide analytical support for the confidence-weighted draft–target divergence criterion used in ADAW. Overall, our results show that performance-driven, window-level drafter optimization is an effective way to improve speculative decoding efficiency.

Acknowledgments and Disclosure of Funding

The authors are employed by Ant Group and Alibaba Group, respectively. No third-party funding was received for this work. The authors declare no additional competing interests.

References

[1] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast Inference from Transformers via Speculative Decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR, 23–29 Jul 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] 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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 5209–5235. PMLR, 21–27 Jul 2024.

[4] 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.

[5] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 28935–28948. PMLR, 21–27 Jul 2024.

[6] 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, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[7] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up Inference Acceleration of Large Language Models via Training-Time Test. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 136737–136756. Curran Associates, Inc., 2025.

[8] Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning Harmonized Representations for Speculative Sampling. In *International Conference on Learning Representations*, volume 2025, pages 35367–35388, 2025.

[9] Shijing Hu, Jingyang Li, Xingyu Xie, Zhihui Lu, Kim-chuan Toh, and Pan Zhou. GRIFFIN: Effective Token Alignment for Faster Speculative Decoding. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 125648–125673. Curran Associates, Inc., 2025.

[10] Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. DistillSpec: Improving Speculative Decoding via Knowledge Distillation. In *International Conference on Learning Representations*, volume 2024, pages 32011–32050, 2024.

[11] Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Online Speculative Decoding. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 31131–31146. PMLR, 21–27 Jul 2024.

[12] Ofir Zafrir, Igor Margulis, Dorin Shteyman, Shira Guskin, and Guy Boudoukh. FastDraft: How to Train Your Draft. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22488–22505, Vienna, Austria, July 2025. Association for Computational Linguistics.

[13] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the Sequential Dependency of LLM Inference Using Lookahead Decoding. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 14060–14079. PMLR, 21–27 Jul 2024.

[14] Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. SpecTr: Fast Speculative Decoding via Optimal Transport. In *Advances in Neural Information Processing Systems*, volume 36, pages 30222–30242. Curran Associates, Inc., 2023.

[15] Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae H Ro, Ahmad Beirami, and Ananda Theertha Suresh. Block Verification Accelerates Speculative Decoding. In *International Conference on Learning Representations*, volume 2025, pages 24965–24994, 2025.

[16] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. SpecInfer: Accelerating Large Language Model Serving with Tree-based Speculative Inference and Verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 932–949, New York, NY, USA, 2024. Association for Computing Machinery.

[17] 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*, volume 37, pages 129531–129563. Curran Associates, Inc., 2024.

[18] Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chen-Chuan Chang, and Jie Huang. Cascade Speculative Drafting for Even Faster LLM Inference. In *Advances in Neural Information Processing Systems*, volume 37, pages 86226–86242. Curran Associates, Inc., 2024.

[19] Jonathan Mamou, Oren Pereg, Daniel Korat, Moshe Berchansky, Nadav Timor, Moshe Wasserblat, and Roy Schwartz. Dynamic Speculation Lookahead Accelerates Speculative Decoding of Large Language Models. In *Proceedings of The 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, volume 262 of *Proceedings of Machine Learning Research*, pages 456–467. PMLR, 14 Dec 2024.

[20] Xukun Liu, Bowen Lei, Ruqi Zhang, and Dongkuan DK Xu. Adaptive Draft-Verification for Efficient Large Language Model Decoding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24668–24676, 2025.

[21] Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. SpecExec: Massively Parallel Speculative Decoding For Interactive LLM Inference on Consumer Devices. In *Advances in Neural Information Processing Systems*, volume 37, pages 16342–16368. Curran Associates, Inc., 2024.

[22] Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, Winston Hu, and Xiao Sun. PEARL: Parallel Speculative Decoding with Adaptive Draft Length. In *International Conference on Learning Representations*, volume 2025, pages 1085–1104, 2025.

[23] Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-Guided Speculative Decoding for Efficient LLM Reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 37555–37572. PMLR, 13–19 Jul 2025.

[24] Bingshuai Liu, Ante Wang, Zijun Min, Liang Yao, Haibo Zhang, Yang Liu, Xu Han, Peng Li, Anxiang Zeng, and Jinsong Su. SPEC-RL: Accelerating On-Policy Reinforcement Learning with Speculative Rollouts. *arXiv preprint arXiv:2509.23232*, 2026.

[25] Siqi Wang, Hailong Yang, Junjie Zhu, Xuezhu Wang, Yufan Xu, and Depei Qian. RLHFSpec: Breaking the Efficiency Bottleneck in RLHF Training via Adaptive Drafting. *arXiv preprint arXiv:2512.04752*, 2025.

[26] Qiaoling Chen, Zijun Liu, Peng Sun, Shenggui Li, Guoteng Wang, Ziming Liu, Yonggang Wen, Siyuan Feng, and Tianwei Zhang. ReSpec: Towards Optimizing Speculative Decoding in Reinforcement Learning Systems. In *Proceedings of Machine Learning and Systems*, volume 8, pages 367–379. MLSys, 2026.

[27] Alexander Samarin, Sergei Krutikov, Anton Shevtsov, Sergei Skvortsov, Filipp Fisin, and Alexander Golubev. LK Losses: Direct Acceptance Rate Optimization for Speculative Decoding. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 107441–107454. PMLR, 06–11 Jul 2026.

[28] Xiandong Zou, Jianshu Li, Jing Huang, and Pan Zhou. Variational Speculative Decoding: Rethinking Draft Training from Token Likelihood to Sequence Acceptance. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 168768–168789. PMLR, 06–11 Jul 2026.

[29] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

[30] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

[31] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc., 2023.

[32] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*, 2021.

[33] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021.

[34] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality, March 2023.

[35] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing Chat Language Models by Scaling High-quality Instructional Conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051, Singapore, December 2023. Association for Computational Linguistics.

[36] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel. *Proc. VLDB Endow.*, 16(12):3848–3860, August 2023.

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[38] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024.

[39] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[40] Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.

A Optimization and Implementation Details

This appendix summarizes the training setup, optimization procedure, hyperparameter settings, and full PPOW training algorithm used in our experiments.

A.1 Training Setup

Our training uses a two-stage pipeline: supervised initialization followed by PPOW-based reinforcement learning. For LLaMA-3, we initialize from the official EAGLE-3 checkpoints [7]. For Qwen, we first train the feature-based drafter with supervised fine-tuning on ShareGPT [34] and UltraChat-200k [35], then further optimize it with PPOW on the same data mixture. All experiments are run on NVIDIA H100 (80GB) GPUs using PyTorch with FSDP [36] to support a frozen target model and a trainable drafter. PPOW training costs about 50, 100, and 200 GPU-hours for 8B, 32B, and 70B targets, respectively.

A.2 Policy Optimization and Target-Anchored KL Regularization

PPOW is implemented with a group-relative clipped policy objective [37, 38]. For each selected speculative window, we sample a rollout group of G_{roll} speculative windows, compute a scalar reward for each draft, and normalize rewards within the rollout group to obtain group-relative advantages. A single normalized advantage is then applied across the token-level updates within that speculative window. Unless otherwise specified, we do not use a separate value function.

A key difference from standard RL training setups lies in the KL regularization. Rather than constraining the current policy toward an initialization policy or an auxiliary reference model, PPOW computes KL regularization between the drafter policy and the frozen target model. Specifically, we compute the token-wise KL divergence between the draft and target distributions at each drafted position and average it over the speculative window. This keeps exploration anchored to the distribution that ultimately governs speculative verification.

A.3 Training and Decoding Configurations

Table 5 summarizes the default training hyperparameters, hard-window curriculum schedule, and decoding settings used throughout our experiments. Under hard-window curriculum scheduling, PPOW begins with a smaller proportion of ADAW-selected hard windows and gradually increases this proportion over the course of training.

Table 5: Training hyperparameters, curriculum schedule, and decoding configurations used in our experiments.

Hyperparameters	**Value**
Training Configuration	
Clip ratio ϵ_{clip}	0.2
KL coefficient β	0.03
Rollout group size G_{roll}	8
Rollout / Sampling temperature	1.0
Window size K	10
Hard-window curriculum	0.2 / 0.4 / 0.6
Learning rate	5×10^{-6}
Warmup ratio	0.05
Precision	bf16
Decoding Configurations	
Decoding strategy	tree decoding
KV cache	true
Speculative depth / num_steps	10
Branching factor / top-k	3
Verification rule	rejection sampling
Temperatures	0.0 / 1.0

A.4 Hard-Window Curriculum in ADAW

ADAW's curriculum designed to avoid overwhelming the policy too early. For the hard-window curriculum, the $0.2 \to 0.4 \to 0.6$ schedule is used to gradually increase the fraction of hard windows during training. If the proportion of hard windows is increased too aggressively early in training, accepted prefixes tend to be very short and many rollout groups provide little useful relative signal, which slows training progress. Gradually increasing the hard-window ratio allows PPOW to first adapt on easier windows and then increasingly focus on the true bottlenecks. This is also consistent with the training dynamics already shown in Figure 4, where enabling ADAW initially lowers reward and raises KL before the policy adapts to the harder window distribution.

The confidence factor $C(P_t)$ helps distinguish **true bottlenecks** from **ambiguous high-entropy mismatches**. When the target distribution is concentrated, large draft–target divergence is more likely to indicate a consequential bottleneck, since an early mismatch is more likely to truncate the acceptance chain. Prioritizing such positions encourages the drafter to correct errors with larger impact on speculative efficiency.

When the target distribution is diffuse, however, divergence alone can over-prioritize positions where the target itself has no strong preference. Confidence weighting suppresses these noisy cases and keeps ADAW focused on windows that matter more for speculative efficiency.

A.5 Full PPOW Training Algorithm

Algorithm 1 PPOW Training Procedure

Inputs: drafter π_θ, target model π_{target}, window size K, group size G_{roll}, clip ratio ϵ_{clip}, KL coefficient β, relative cost γ, proximity threshold ϵ, proximity weight η
for each training prefix $\boldsymbol{x}$ **do**
  ▷ *Generate target-side outputs and draft distributions*
    $(\boldsymbol{y}, \mathbf{H}, \{P_t\}_{t=1}^{T}) \leftarrow \pi_{\text{target}}(\boldsymbol{x})$
    $\{Q_t\}_{t=1}^{T} \leftarrow \pi_\theta(\boldsymbol{x}, \boldsymbol{y}, \mathbf{H})$
  ▷ *Compute criticality score*
  for $t = 1, \ldots, T$ **do**
    $v_t \leftarrow \left(1 - \frac{H(P_t)}{\log |\mathcal{V}|}\right) \cdot D_{\text{KL}}(P_t \,\|\, Q_t)$
  end for
  ▷ *ADAW-based window selection*
  for each window start index $j \in \{1, \ldots, T - K + 1\}$ **do**
    $s_j \leftarrow \frac{1}{K} \sum_{t=j}^{j+K-1} v_t$
  end for
  Sample a training window start $j^\star$ from the normalized weights proportional to $\{s_j\}$
  ▷ *Grouped rollout and reward computation*
  for $g = 1, \ldots, G_{\text{roll}}$ **do**
    Rollout a speculative window $\hat{\boldsymbol{y}}_g \sim \pi_\theta(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<j^\star}, \mathbf{H}_{<j^\star})$
    $k_g \leftarrow \textsc{Verify}(\hat{\boldsymbol{y}}_g, \pi_{\text{target}})$
    Compute

$$\Delta_g \leftarrow \sum_{t=1}^{K} \left[\log \pi_{\text{target}}(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t}) - \log \pi_{\text{target}}(\hat{y}_{g,t} \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{g,<t}) \right]$$

    $r_g \leftarrow \dfrac{k_g}{k_g \gamma + 1} + \eta \cdot \mathbf{1}[\Delta_g < \epsilon]$
  end for
  Compute normalized group-relative advantages $\{\hat{A}_g\}_{g=1}^{G_{\text{roll}}}$ from rewards $\{r_g\}_{g=1}^{G_{\text{roll}}}$
  ▷ *Policy update*
  Update θ using the clipped group-relative policy objective with target-anchored KL regularization
end for

B Analysis of Confidence-Weighted Draft–Target Divergence

In Section 4.2, we define the token-level *criticality score*

$$v_t = C(P_t)\, D_{\mathrm{KL}}(P_t \,\|\, Q_t),$$

where P_t and Q_t denote the target and drafter distributions at position t, and $C(P_t)$ is an entropy-normalized confidence score of the target distribution. In the main text, this score serves as a prioritization signal for training windows. This appendix provides analytical support for this design by relating draft–target divergence to speculative acceptance behavior and clarifying the role of confidence weighting.

B.1 Acceptance Probability and Draft–Target Divergence

Let α_t denote the expected acceptance probability at position t under speculative decoding. Following [1], when a draft token is sampled from Q_t and verified against P_t, the expected acceptance probability is

$$\begin{aligned}\alpha_t &= \mathbb{E}_{y\sim Q_t}\left[\min\left(1, \frac{P_t(y)}{Q_t(y)}\right)\right] \\ &= \sum_{y\in\mathcal{V}} \min(P_t(y), Q_t(y)).\end{aligned}$$

Using the identity $\min(a,b) = \frac{1}{2}(a+b-|a-b|)$, we obtain

$$\begin{aligned}\alpha_t &= \frac{1}{2}\sum_{y\in\mathcal{V}} (P_t(y) + Q_t(y) - |P_t(y) - Q_t(y)|) \\ &= 1 - \frac{1}{2}\sum_{y\in\mathcal{V}} |P_t(y) - Q_t(y)| \\ &= 1 - \delta(P_t, Q_t),\end{aligned}$$

where $\delta(P_t, Q_t)$ denotes the total variation distance.

Applying Pinsker's inequality,

$$\delta(P_t, Q_t) \le \sqrt{\frac{1}{2} D_{\mathrm{KL}}(P_t \,\|\, Q_t)},$$

yields

$$\alpha_t \ge 1 - \sqrt{\frac{1}{2} D_{\mathrm{KL}}(P_t \,\|\, Q_t)}.$$

This bound shows that smaller draft–target KL divergence implies a higher lower bound on the acceptance probability. It therefore supports the use of draft–target divergence as a proxy of speculative difficulty.

B.2 Confidence Modulation of Draft–Target Divergence

The analysis above shows that draft–target divergence is informative about acceptance behavior. However, the practical significance of a given divergence also depends on the uncertainty of the target distribution. For the same level of draft–target divergence, discrepancies are typically more consequential when the target distribution is concentrated than when the target itself is uncertain.

To account for this effect, we introduce the confidence factor

$$C(P_t) = 1 - \frac{H(P_t)}{\log|\mathcal{V}|} \in [0,1],$$

which assigns larger values to lower-entropy target distributions and smaller values to higher-entropy ones.

The resulting criticality score,

$$v_t = C(P_t)\, D_{\mathrm{KL}}(P_t \,\|\, Q_t),$$

can be viewed as a confidence-modulated measure of draft–target divergence. It increases the priority of positions where the drafter distribution diverges from a confident target distribution, while reducing the influence of divergence in higher-entropy contexts. Aggregating v_t over a speculative window therefore yields a window-level signal that emphasizes regions more likely to constrain speculative acceptance in practice.

C Optimization Differences Between PPOW and Continued Supervised Training

PPOW and continued supervised training (CST) target different optimization objectives in speculative decoding. CST improves token-level imitation of the target model, whereas PPOW directly optimizes the window-level utility aligned with speculative decoding efficiency at inference time.

C.1 Speculative Verification and Window-Level Utility

Speculative decoding commonly uses rejection-sampling verification to preserve the target-model distribution during inference. Under this verification rule, the practical gain of a speculative window depends on how many of its tokens are accepted by the target model, making the accepted prefix length, rather than token-level likelihood alone, the relevant measure of speculative utility.

For a speculative window $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_K)$, the acceptance probability of the token at position t under rejection-sampling verification is

$$\alpha_t = \min\left(1,\, \frac{\pi_{\text{target}}(\hat{y}_t \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<t})}{\pi_\theta(\hat{y}_t \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<t})}\right).$$

The acceptance length τ is then determined by the first position at which verification fails:

$$\tau = \sum_{n=1}^{K} \prod_{t=1}^{n} \mathbf{1}[u_t \le \alpha_t], \qquad u_t \sim U(0,1).$$

Because τ is defined by consecutive acceptance from the beginning of the speculative window, early truncation under verification immediately eliminates the contribution of all subsequent tokens in that window. In this sense, speculative utility is inherently window-level, and positions that cause early verification truncation can become bottlenecks with disproportionate impact on inference-time efficiency. This creates a mismatch with token-level training objectives and motivates the window-level optimization used in PPOW.

C.2 Limitations of Continued Supervised Training (CST) for Speculative Decoding

CST retains the same token-level cross-entropy objective as standard supervised training, and this token-level training objective is not fully aligned with the window-level speculative utility relevant at inference time:

$$\begin{aligned}\mathcal{L}_{\text{CST}} &= \mathbb{E}_{\boldsymbol{y}\sim\pi_{\text{target}}}\left[-\log \pi_\theta(\boldsymbol{y} \mid \boldsymbol{x})\right] \\ &= \mathbb{E}_{\boldsymbol{y}\sim\pi_{\text{target}}}\left[\sum_{t=1}^{K} -\log \pi_\theta(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t})\right].\end{aligned}$$

This objective improves draft–target imitation. However, its effect on speculative decoding is indirect, because the optimized quantity is token-level likelihood rather than the realized utility of a speculative window. In other words, matching the target model token by token does not necessarily maximize the number of tokens that survive verification within an accepted speculative window.

As a result, CST does not directly optimize accepted prefix length, explicitly focus on bottleneck positions that cause early verification truncation, or optimize across multiple sampled speculative windows for the same context. These mismatches help explain why CST may yield limited additional gains in inference-time speculative efficiency, especially once token-level imitation has largely saturated, and motivate the window-level optimization used in PPOW.

C.3 PPOW as Window-Level Policy Optimization

PPOW formulates speculative decoding as a policy optimization problem, where the drafter is optimized over speculative windows using window-level rewards aligned with inference-time speculative performance (Section 4.3):

$$J(\theta) = \mathbb{E}_{\hat{\boldsymbol{y}} \sim \pi_\theta} \left[R(\boldsymbol{x}, \hat{\boldsymbol{y}}) \right].$$

Here, optimization is driven by the realized reward of sampled speculative windows rather than by token-level supervision on a fixed target token sequence. Since a full response can be decomposed into many overlapping speculative windows, not all windows contribute equally informative training signals. Window-level optimization therefore makes it possible to focus learning on more informative speculative windows, especially those containing positions that are more critical to early truncation under verification.

PPOW also samples multiple speculative windows for the same context and computes group-relative advantages among them. This grouped optimization reflects practical settings that evaluate multiple candidate token sequences during decoding and provides a stronger relative learning signal toward speculative windows with better acceptance behavior.

D Cost-Aware Speedup Reward versus Measured Speedup Reward

A natural alternative to the Cost-Aware Speedup Reward in Section 4.3 is to define training rewards directly from measured inference speedup. In practice, however, measured speedup depends strongly on the serving environment, including hardware, backend implementation, cache behavior, batching, and verification configuration. It can also provide noisy or low-resolution feedback at the speculative-window level. This appendix compares the Cost-Aware Speedup Reward and a Measured-Speedup-Based Reward alternative from two perspectives: whether they exhibit similar reward trends as accepted length increases, and whether training with measured speedup yields meaningful gains in a fixed setup.

Table 6: **Reward values induced by accepted prefix length.** Comparison between a Measured-Speedup-Based Reward and PPOW's Cost-Aware Speedup Reward as accepted length increases. The two rewards are not numerically identical, but they preserve the same monotonic trend.

Accepted Prefix Length	Measured Speedup Reward	Cost-Aware Speedup Reward
1	0.81	0.89
2	1.49	1.60
3	1.81	2.18
4	2.06	2.67
5	2.49	3.08
6	2.62	3.43
7	2.88	3.74

Table 7: **Training with PPOW's Cost-Aware Speedup Reward vs. Measured-Speedup-Based Reward.** Final average acceptance length (τ) on GSM8K with LLaMA-3.1-8B under the same training budget.

Reward formulation	Average acceptance length (τ)
Cost-Aware Speedup	6.50
Measured Speedup	6.61

Table 6 compares the reward values induced by accepted prefix length under the two formulations. Although the Cost-Aware Speedup Reward does not numerically match the Measured-Speedup-Based Reward, the two exhibit the same monotonic trend: longer accepted prefixes receive larger reward under both formulations. The two rewards differ in scale because the measured reward is system-dependent, but they preserve the same ordering over accepted-prefix outcomes. This shared trend is important for training, since it preserves the relative preference for speculative windows with better efficiency characteristics.

Table 7 compares training with the Cost-Aware Speedup Reward against training with a Measured-Speedup-Based Reward under the same budget. In this fixed setup, the measured-speedup-based reward yields only a small improvement in final acceptance length. However, such a reward remains tightly coupled to the specific serving stack used during training and may provide noisy or low-resolution feedback at the window level. The cost-aware formulation therefore offers a more portable and practical default objective.

Overall, these results support the use of the Cost-Aware Speedup Reward during training: it tracks the same acceptance-related trend as measured speedup and remains effective for optimization, while avoiding the system dependence of direct runtime-based rewards.

E Effect of the Distribution-Based Proximity Reward Across Easy and Hard Windows

To characterize the effect of the Distribution-Based Proximity Reward R_{dist}, we partition speculative windows according to the acceptance behavior of the supervised baseline. We refer to windows with full acceptance ($k = K$) as *easy*, and to those for which verification terminates before the end of the speculative window ($k < K$) as *hard*.

Under this partition, we compare the EAGLE-3 supervised baseline, PPOW without R_{dist}, and full PPOW. Table 8 reports both average acceptance length τ and an alignment divergence metric ∇, defined token-wise as

$$\nabla = \exp(\delta) - \delta - 1, \qquad \delta = \log \pi_{\text{target}}(\hat{y}_t \mid \cdot) - \log \pi_\theta(\hat{y}_t \mid \cdot).$$

This metric measures draft–target alignment on drafted tokens through token-level log-probability differences, with lower values indicating better alignment.

Table 8: **Effect of the Distribution-Based Proximity Reward on easy and hard windows.** Easy windows are fully accepted by the supervised baseline ($k = K$), whereas hard windows terminate early within the speculative window ($k < K$). Adding R_{dist} preserves performance on easy windows and further improves acceptance length while reducing the alignment divergence metric ∇ on hard windows.

Window type	Model	τ	∇
Easy	EAGLE-3	6.0	**0.009**
	PPOW (w/o R_{dist})	6.0	0.012
	PPOW (Full)	6.0	0.017
Hard	EAGLE-3	2.32	7.091
	PPOW (w/o R_{dist})	4.58	3.644
	PPOW (Full)	4.86	**3.275**

On easy windows, the supervised baseline, PPOW without R_{dist}, and full PPOW have identical acceptance length, and their ∇ values differ only slightly. On hard windows, however, full PPOW improves acceptance length (4.58 $\rightarrow$ 4.86) while further reducing ∇ (3.644 $\rightarrow$ 3.275) relative to PPOW without R_{dist}. These results support the inclusion of R_{dist}, which preserves performance on easy windows while yielding additional gains on hard windows.

F Sensitivity to the Weight (η) of the Distribution-Based Proximity Reward

The Distribution-Based Proximity Reward R_{dist} is only applied when the accepted length is $k = 0$, where the accepted-prefix reward alone provides no signal about whether the drafted window remains broadly compatible with the target model's preferences. R_{dist} therefore serves as a bounded auxiliary reward that provides partial credit in this sparse-reward case.

The coefficient η controls the strength of this auxiliary reward relative to the main accepted-prefix / cost-aware speedup objective. In Table 4, the settings "w/o R_{dist}" and "PPOW (Full)" correspond to $\eta = 0$ and $\eta = 1.0$, respectively. To further study sensitivity to η, we additionally train models

with $\eta = 0.5$ and $\eta = 2.0$ under the same LLaMA-3.1-8B setup and evaluate them on MT-Bench and GSM8K.

Table 9 shows that PPOW is reasonably stable when η remains in a moderate range: $\eta = 0.5$ and $\eta = 1.0$ achieve similar results on both benchmarks. However, performance degrades at $\eta = 2.0$, indicating that assigning too much weight to R_{dist} can overemphasize distributional proximity and weaken optimization toward the main accepted-prefix / cost-aware speedup objective. Overall, these results suggest that η should remain moderate rather than too large.

Table 9: **Sensitivity to the weight η of R_{dist}.** Results are reported under the same LLaMA-3.1-8B setup as Table 4.

η	MT-Bench		GSM8K	
	τ	Speedup	τ	Speedup
0.0	5.05	2.52×	6.41	3.48×
0.5	5.33	2.63×	6.47	3.50×
1.0	**5.47**	**2.72**×	**6.50**	**3.52**×
2.0	5.18	2.57×	6.32	3.36×

G Additional Baseline Comparisons

G.1 Comparisons with OSD and FastDraft

Table 10 reports additional comparisons against OSD [11] and FastDraft [12]. In the reported comparison, PPOW substantially improves over OSD on Vicuna-7B [31] across both GSM8K [33] and HumanEval [32]. Compared with FastDraft on LLaMA-3.1-8B-Instruct, PPOW shows much larger gains on GSM8K and remains competitive on HumanEval, achieving a slightly higher acceptance length with comparable speedup. These results suggest that PPOW remains effective beyond the primary baselines reported in the main text.

Table 10: **Additional baseline comparisons against OSD and FastDraft.** PPOW substantially outperforms OSD on Vicuna-7B and consistently improves over FastDraft on GSM8K, while remaining competitive on HumanEval with LLaMA-3.1-8B-Instruct.

Target Model	Method	Benchmark	τ	Speedup
Vicuna-7B	OSD	GSM8K	3.53	1.35×
	PPOW	GSM8K	**6.12**	**4.40**×
	OSD	HumanEval	2.57	1.58×
	PPOW	HumanEval	**6.75**	**4.87**×
LLaMA-3.1-8B-Instruct	FastDraft	GSM8K	2.67	1.63×
	PPOW	GSM8K	**6.50**	**3.52**×
	FastDraft	HumanEval	6.99	**4.29**×
	PPOW	HumanEval	**7.23**	4.14×

G.2 Additional Results on Natural-Language Tasks

Table 11 reports additional results on X-SUM [39] and WMT14 [40]. Compared with the stronger gains observed on reasoning tasks such as GSM8K, PPOW shows only modest improvements on X-SUM and limited gains on WMT14. We attribute this gap to the task properties of natural-language generation: these tasks admit a much more diverse set of valid token sequences, which weakens the acceptance-aware reward signal used by PPOW. The effect is less severe on X-SUM, where summaries remain partially anchored to source content, but becomes more pronounced on WMT14, where lexical and syntactic diversity greatly enlarges the space of valid outputs. These observations are consistent with the broader pattern that PPOW is most beneficial when speculative performance is controlled by relatively structured bottleneck decisions.

Table 11: **Additional results on natural-language tasks.** PPOW slightly improves acceptance length and speedup on X-SUM, while gains on WMT14 are limited.

	X-SUM		WMT14		Mean	
Method	τ	Speedup	τ	Speedup	τ	Speedup
Temperature = 0.0						
EAGLE3	4.94	2.81×	2.72	**1.56**×	3.83	**2.19**×
EAGLE3 + CST	4.58	2.60×	**2.73**	**1.56**×	3.66	2.08×
PPOW	**5.13**	**2.87**×	2.61	1.49×	**3.87**	2.18×
Temperature = 1.0						
EAGLE3	4.15	2.30×	**2.56**	**1.44**×	3.36	1.87×
EAGLE3 + CST	3.87	2.17×	2.50	1.37×	3.19	1.77×
PPOW	**4.44**	**2.53**×	2.37	1.30×	**3.41**	**1.92**×

H Limitations and Broader Impacts

H.1 Limitations

PPOW improves speculative decoding by aligning training with inference-time acceptance behavior, but this stronger alignment comes with additional training overhead. Compared with supervised drafter training, PPOW requires grouped rollouts, speculative verification, and target-anchored KL regularization, and our current implementation uses both a frozen target model and a trainable drafter. This makes PPOW more resource-intensive and more complex to implement than continued supervised training.

This alignment objective also introduces several hyperparameters beyond standard reinforcement learning, including speculative-decoding-specific settings such as the window size, the relative cost γ, the proximity reward weight η, the proximity threshold ϵ, and the hard-window curriculum in ADAW. While we use a unified configuration across experiments and observe stable gains, reducing such dependence through more adaptive or self-tuning variants is a natural direction for future work.

H.2 Broader Impacts

PPOW suggests an additional algorithmic perspective on LLM inference: some components of the inference pipeline may be optimized directly with performance-driven objectives rather than only with token-level supervision. In this setting, reinforcement learning provides a practical way to align training with system-level inference behavior when the utility of a model component is determined by structured interactions during decoding rather than by local prediction quality alone.

Framing speculative decoding as a learning problem may also relate to other inference-time decisions, such as candidate allocation, request routing, scheduling, and load balancing. These components are often optimized separately, but in practice they jointly affect end-to-end serving efficiency. The formulation in PPOW may offer a useful perspective for future work that treats such inference-time modules as part of a unified decision-making environment.

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: The abstract and introduction state the main claims of the paper: PPOW shifts drafter optimization from token-level imitation to window-level performance-driven policy optimization, and combines performance-driven rewards with adaptive window prioritization to better align training with the prefix-sensitive acceptance behavior of speculative decoding. These claims are supported by the discussion in Appendix C and by the empirical results in Section 5.2 and Section 5.3.

   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: We discuss the limitations of our approach in Appendix H.1, including the additional training overhead introduced by RL-based post-training and the extra hyperparameters required to align optimization with speculative decoding.

   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: [Yes]

   Justification: The paper's analytical results are provided in Appendix B, which states the assumptions used in the acceptance-probability analysis and gives the corresponding derivation relating draft–target divergence to speculative acceptance behavior.

   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: We provide a detailed description of the experimental setup, datasets, baselines, and decoding protocols in Section 5.1. Comprehensive hyperparameter settings and the complete training algorithm (Algorithm 1) are provided in Appendix A.

   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: We provide the source code in the supplementary material, along with detailed guidelines for reproducing our experimental results. The training datasets used in our experiments, including ShareGPT [34] and the UltraChat-derived UltraChat-200k subset [35], are publicly available and properly cited in the paper.

   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: Training and decoding configurations, including detailed hyperparameter settings, scheduling, and sampling temperatures, are comprehensively listed in Table 5 (Appendix A).

   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: [No]

Justification: Following common practice in speculative decoding [1, 2, 7], we report average acceptance length and end-to-end speedup over fixed evaluation workloads under a unified decoding protocol. We do not include error bars or statistical significance tests, as these metrics are generally stable under fixed evaluation workloads and decoding configurations in our setting.

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: We detail the compute resources, distributed training framework, and approximate optimization costs (in GPU-hours) in Appendix A.1.

   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: Our research complies with the NeurIPS Code of Ethics. The study focuses on optimization algorithms for language model inference.

   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: [Yes]

    Justification: We discuss broader impacts in Appendix H.2, focusing on the algorithmic implications of optimizing LLM inference components with performance-driven objectives and on the possibility of viewing inference-time decisions in a more unified, system-level optimization framework.

    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 work proposes an algorithm for inference acceleration and does not release new pre-trained language models, image generators, or scraped datasets.

    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: We credit the original sources for the pretrained models, baselines, datasets, and benchmarks used in our experiments, and we use these assets subject to their respective licenses and terms of use.

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 evaluates optimization algorithms on existing benchmarks and does not release new datasets or base model assets.

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: This paper focuses on LLM inference acceleration and does not involve crowdsourcing nor research with human subjects.

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: This paper focuses on LLM inference acceleration and does not involve crowdsourcing nor research with human subjects.

    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: [N/A]

    Justification: This work studies target and draft models directly as the object of research. It does not use additional LLMs as non-standard tools for data generation, annotation, or evaluation.

    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.