MTP-RL: Acceleration of Reinforcement Learning Rollouts with Policy-Aligned Multi-Token Prediction

Ke Wang[1,2]*, Aohan Zeng[4,5], Zhengxiao Du[5], Bohan Zhang[1,2], Yuxuan Hu[1,2], Xinyi Wang[6], Jie Tang[4] Jing Zhang[1,3]†,
[1]School of Information, Renmin University of China, Beijing, China,
[2]Key Laboratory of Data Engineering and Knowledge Engineering, Beijing, China,
[3]Engineering Research Center of Database and Business Intelligence, Beijing, China
[4]Tsinghua University, [5]Z.ai, [6]Nankai University
{wakeeys, zhang-jing}@ruc.edu.cn

Abstract

Reinforcement learning (RL) is widely applied to boost the performance of pretrained models, yet its training efficiency is severely constrained by rollout generation. While speculative decoding based on multi-token prediction (MTP) offers a potential acceleration pathway, its widespread adoption is hindered by the absence of MTP in vanilla pretrained models and the rapid degradation of the MTP acceptance length in RL training. To address these issues, this paper proposes MTP-RL, a two-stage framework that pioneers effective training of MTPs in RL and accelerates the rollout phase for diverse models. It involves a pipeline to equip the multi-layer parameter-sharing MTP for all models and an innovative advantage-aware MTP optimization strategy to facilitate policy-aligned training of MTPs. Experiments demonstrate that our method not only achieves stable growth of acceptance length during RL training, but also accelerates RL rollouts, achieving an average 23.1%–55.3% reduction in rollout time compared to baselines.

1 Introduction

Reinforcement learning (RL) has already been a functional and powerful strategy to optimize large language models (LLMs) by maximizing expected rewards (Guo et al., 2025; Yang et al., 2025; Zeng et al., 2025). In mainstream RL tasks for intensive reasoning generation, we are increasingly requiring models to generate lengthy chains of thought with innumerable tokens to solve problems effectively (Muennighoff et al., 2025; Shi et al., 2024). However, during the rollout phase of RL, the strictly autoregressive generation of LLMs results in latency that scales approximately quadratically with sequence length, which consumes a substantial majority of total training time and severely limits the efficiency of RL training.

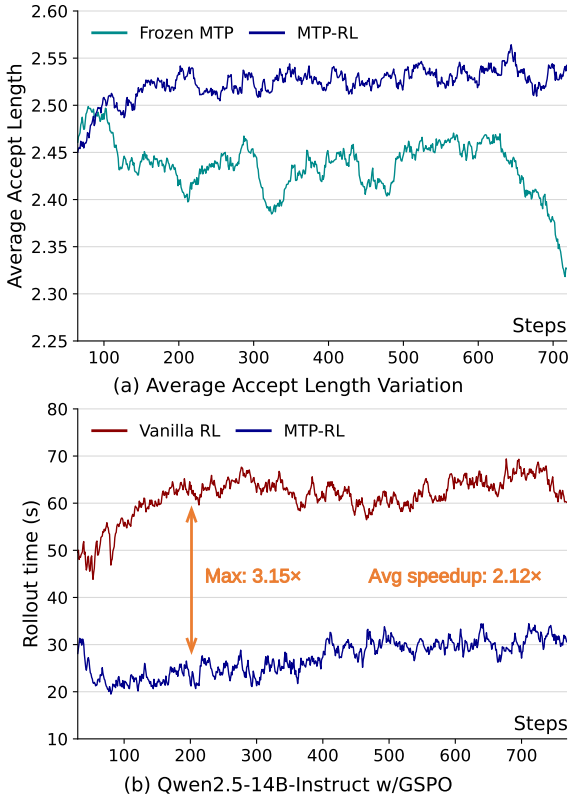

Figure 1: (a) Variation of the average acceptance length between a frozen MTP module and MTP-RL during RL training. In contrast to the collapse of acceptance length in frozen MTP, MTP-RL achieves a continuous improvement. (b) Comparison of rollout time on Qwen2.5-14B-Instruct with GSPO. MTP-RL substantially reduces rollout time and achieves a maximum speedup of 3.15× compared to vanilla RL.

To alleviate the overhead of rollout generation in RL training, efficiency-oriented approaches can be broadly categorized into three orthogonal paradigms: model-level, pipeline-level, and algorithm-level. Model-level optimization methods such as quantization (Huang et al., 2025; Liu et al., 2025) map high-precision weights and activations to lower precisions, compressing the model parameter sizes to reduce memory bandwidth requirements and computational cost during the roll-

*Work was done when interned at Z.ai.
†Corresponding Author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 37530–37542

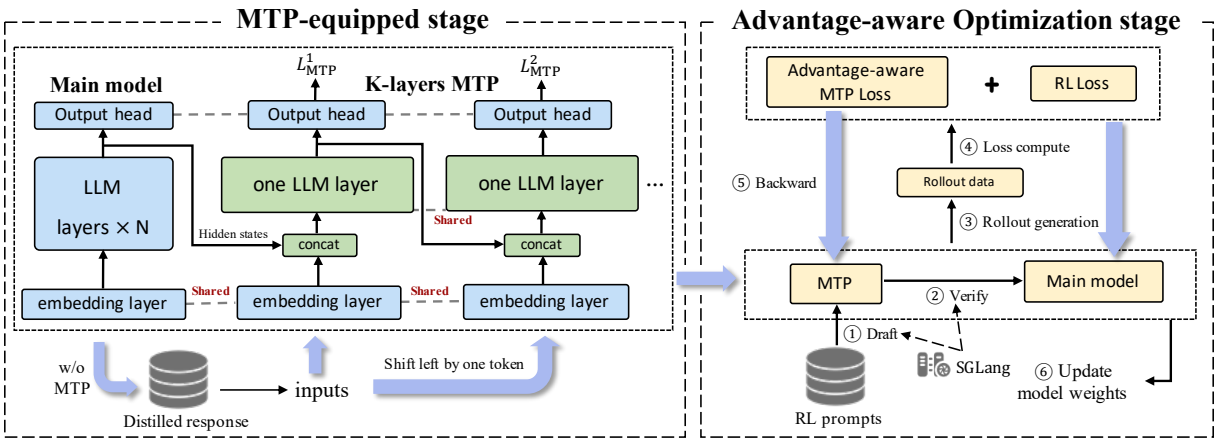

Figure 2: Overview of our two-stage framework. It encompasses the design of multi-layer, parameter-sharing MTP modules at the MTP-equipped stage, the incorporation of MTP during RL rollouts, and the backpropagation of MTP gradients in the advantage-aware RL optimization stage.

out phase. However, low-precision models often produce rollouts that deviate from those generated by full-precision models, making it challenging to improve efficiency without degrading performance. Pipeline-level optimization focuses on increasing execution parallelism between training and rollout phases to reduce pipeline bubbles triggered by waiting for long-tail rollouts (Zhang et al., 2025). This approach necessitates extensive modifications to the RL framework, resulting in considerable engineering complexity.

Speculative decoding (SD) has emerged as the widely adopted algorithm-level paradigm for accelerating the rollout phase, encompassing prebuilt, natively trained, and train-free approaches. Prebuilt methods (Chen et al., 2025) typically reuse draft models directly adapted from EAGLE, which limits their generalization across diverse LLM architectures. In contrast, train-free SD approaches (He et al., 2025) leverage n-gram matching to extract patterns from historical rollouts. Nevertheless, their acceleration remains belated, yielding noticeable speedups only in later training stages. Natively trained SD is enabled by multi-token prediction (MTP) (Cai et al., 2025; Liu et al., 2024), but its application to RL remains non-trivial and faces two significant obstacles. First, most open-source pretrained models lack native MTP architectures, rendering them incompatible with existing acceleration techniques. More critically, as illustrated in Figure 1 (a), existing RL training paradigms (Xiaomi et al., 2025) typically freeze MTP modules, under which an "acceptance collapse" phenomenon frequently emerges during training. Consequently, the potential speedup achievable with MTP modules is severely diminished.

To address both the difficulty of training models equipped with MTPs in RL and the inefficiency of the RL rollout phase, we propose MTP-RL, a two-stage framework that pioneers the viable training of MTP in RL and accelerates RL rollouts for any MTP-equipped LLM, as illustrated in Figure 2. In the MTP-equipped stage, we train the MTP module from scratch in a lightweight manner to endow vanilla models with SD capability, adopting a multi-layer parameter-sharing design to enhance draft quality with minimal training parameters. In the advantage-aware optimization stage, we introduce an MTP training strategy that reformulates the MTP optimization objective using an advantage-weighted KL divergence combined with a cross-entropy loss. By leveraging advantage signals to filter low-reward trajectories and emphasize high-reward samples, this strategy effectively suppresses the occurrence of "acceptance collapse". Moreover, unlike train-free approaches, our method yields models that retain native SD capability throughout RL training, enabling faster decoding on downstream tasks without requiring additional adaptation.

Our contributions are summarized as follows:

- We design MTP-RL, a two-stage framework that pioneers policy-aligned MTP training throughout RL and simultaneously leverages MTPs to accelerate RL rollouts.
- Our framework incorporates two key techniques: a lightweight training of multi-layer parameter-sharing MTP that enhances initial draft prediction quality and an advantage-aware optimization strategy that prioritizes

high-reward trajectories to prevent "acceptance collapse" effectively.

- We conduct experiments on math and code tasks employing Qwen models of various sizes under multiple RL algorithms. Empirical results demonstrate that our approach acquires increasing acceptance lengths throughout the training lifecycle. This enhancement reduces average rollout time by $23.1\% - 55.3\%$ and delivers a maximum speedup of $3.15\times$ during the total RL rollout.

2 Pilot Study on the Use of MTP in RL

In this section, we introduce the MTP module and conduct an in-depth exploration of "acceptance collapse" in the joint RL training of main models and MTP modules. Our investigation centers on three progressive key questions: **(1)** *Must MTP be trained during RL to remain effective?* **(2)** *Is cross-entropy loss sufficient for MTP training in RL?*(**3**) *Can blocking the propagation of MTP gradients into the main model overcome the limitation of cross-entropy supervision?*

2.1 Preliminaries of Multi-Token Prediction

Multi-token prediction (MTP) equips the main LLM with an auxiliary prediction module that mirrors a transformer layer in the main model. During training, we instantiate K MTP modules. For the k-th MTP module, the prediction of the $(i+1)$-th token $\hat{x}_{i+1}^k$ is decoded by a transformer layer, which takes as input the concatenation of the final-layer hidden states h^{k-1} produced by the $(k-1)$-th MTP and the corresponding input tokens $x_{<i+1}$. The input sequence is then left-shifted by one token under the teacher-forcing paradigm, and combined with the final-layer hidden states h^k from the k-th MTP to produce the $(i+2)$-th token prediction $\hat{x}_{i+2}^{k+1}$ through the $(k+1)$-th MTP module. This process propagates sequentially through all K MTP modules, with K cross-entropy losses to optimize predictions against the ground-truth sequence of length S, as illustrated below.

$$\mathcal{L}_{\text{MTP-CE}} = -\sum_{k=1}^{K} \frac{\lambda_k}{S-k} \sum_{i=1+k}^{S} \Big[\log \pi_{\text{MTP}}^k \Big(x_i \mid h_{<i-1}^{k-1}, x_{k:i-1} \Big) \Big] \tag{1}$$

During inference, a single MTP module is selected to predict several draft tokens, with a substantially lower computational cost compared to the main model. These draft tokens are jointly verified by the main model in a single verification step. Upon encountering the first mismatched token, that token and all remaining draft tokens are discarded.

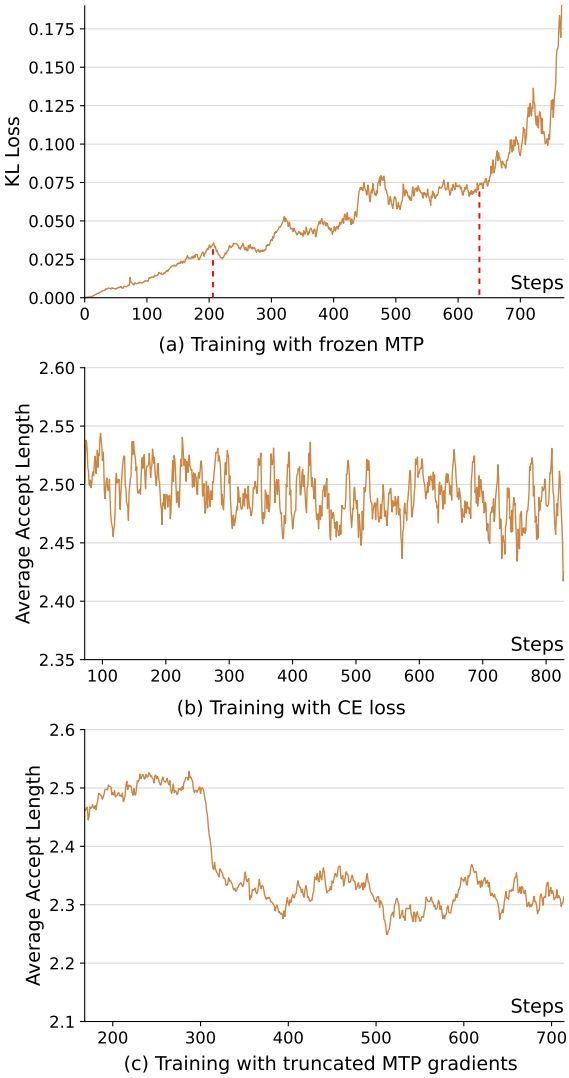

Figure 3: The evolution of the KL loss when training with a frozen MTP, together with the trend in the average acceptance length under training with CE loss and with truncated MTP gradients in RL, respectively.

2.2 Analysis of MTP Training in RL

(1) ***Must MTPs be trained during RL to remain effective?*** A straightforward approach to accelerating RL rollouts with MTP is to freeze its parameters during training. However, as shown in Figure 1 (a), this naive strategy leads to a rapid decline in acceptance length over training. We examine the training dynamics of the main model under RL optimization. As the main model is continuously updated while the MTP module remains frozen, their output distributions gradually diverge. This divergence is reflected by the increasing KL loss, which coin-

cides with the regions on both sides of the red lines in Figure 3 (a), indicating a growing mismatch between the frozen MTP and the evolving main model and ultimately leading to "acceptance collapse".

(2) ***Is cross-entropy loss sufficient for MTP training in RL?*** As demonstrated in Section 2.1, the optimization objective of MTP-equipped models in supervised training is formulated from cross-entropy (CE) loss, which is successfully applied to pretrain models and further enhances their capacity (Liu et al., 2024). However, we realize that this naive approach is insufficient in RL training. Empirical evidence displays that merely utilizing CE loss as the objective for MTPs fails to inhibit the decay of the acceptance length, as illustrated in Figure 3 (b). Moreover, this drawback is amplified as training progresses, leading to persistent degradation of acceptance length in the mid-to-late stage of training. We attribute this limitation to the fundamental difference between reward-driven optimization in RL and supervised training.

(3) ***Can blocking the propagation of MTP gradients into the main model overcome the limitations of cross-entropy supervision?*** Another intuitive alternative strategy is to decouple the optimization objectives by truncating the gradient from the MTP module, preventing its gradient from backpropagating to the main model. However, as illustrated in Figure 3 (c), the acceptance length continues to decline under this decoupled training configuration. We suggest that the main model remains unaware of updates to the MTP module, leading to repeated abrupt shifts in its output distribution, which in turn hinder the MTP module from effectively tracking the behavior of the main model.

Conclusion for pilot studies. We conduct pilot studies on MTP-equipped models to answer the three questions above. Specifically, for the first question, we evaluate whether freezing MTP modules in RL training preserves acceptance length; for the second, we train MTPs only with CE loss; for the last, we block the gradient flow from MTPs to the main model during RL. In all three settings, we observe persistent "acceptance collapse" arising from different underlying causes, which motivates the design of a new joint optimization strategy in RL for the main model and the MTP module.

3 MTP-RL Framework

To achieve reliable MTP training and efficient acceleration in the RL rollout phase, we propose MTP-RL, a two-stage framework that equips arbitrary LLMs with SD capability. As illustrated in Figure 2, the framework first trains MTP modules from scratch for models lacking them. It subsequently enters the advantage-aware optimization stage for RL, where main models and MTP modules are jointly optimized using our proposed advantage-aware MTP optimization strategy. Through this two-stage design, MTP-RL enables reliable SD across diverse LLMs while eliminating "acceptance collapse" and delivering sustained acceleration for RL rollout generation.

3.1 MTP-equipped Stage

To adapt SD for LLMs without native MTP support (e.g., vanilla models), we introduce an MTP-equipped stage. This stage extends the architecture with MTP modules and trains them for SD while keeping the main model frozen. Specifically, for vanilla models without MTP modules, we attach a new MTP module and perform a preliminary training process from scratch using self-distilled data generated by vanilla models themselves. Conversely, models already equipped with MTP modules will be reused directly to ensure efficiency. This stage pledges that the LLM is architecturally ready to employ MTP acceleration in RL.

Prior works (Guo et al., 2025) pretrain the main model with multiple different MTP modules, substantially increasing the number of trainable parameters. In contrast, as displayed in the left part of Figure 2, we initialize the MTP module from scratch, where the module consists of K layers that share parameters across all layers, while the embedding layer and the output head are also shared with the main model. Our approach achieves a competitive final average acceptance length while significantly reducing the parameter footprint. When performing a K-layers forward, we employ the sum of K CE losses to train the MTP module, with the hyperparameters λ_1 to λ_k of each layer loss defined in Equation 1, in contrast to conventional pretraining where all layer losses are treated equally.

3.2 Advantage-aware Optimization Stage

To remedy the "acceptance collapse" dilemma mentioned above, we propose an advantage-aware MTP optimization strategy (Adv-Opt) for RL training. This approach introduces the advantage-aware KL divergence loss as a strict alignment objective, forcing the distribution of MTP modules to maintain tight consistency with the target distribution and

continuously tracking the rapid evolution of the main model. Additionally, unlike naive methods that treat all samples equally, our strategy leverages the advantage estimate A obtained from the RL process to filter trajectories for MTP modules dynamically. The core intuition is that RL seeks to optimize the main model by maximizing the expectation of advantages. Consequently, only utilizing samples with positive advantages while disregarding those with negative advantages can effectively prevent MTP modules from excessively deviating in undesirable directions.

Assuming that j indexes the sample, we define an advantage-based indicator $w(A_j)$ as follows:

$$w(A_j) = \begin{cases} 0, & \text{if } A_j < 0 \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

This weighting function is applied to a KL divergence loss, ensuring alignment only on positive trajectories. We let π_ϕ represent the distribution of the MTP module, π_θ represent the distribution of the main model, S is the sample length and V is the vocabulary size. The advantage-aware KL loss in the j-th sample x_j is defined as:

$$\mathcal{L}_{\text{MTP-KL}}(x_j) = \frac{w(A_j)}{S \cdot V} \sum_{i=1}^{S} \sum_{v=1}^{V} \Big[\pi_\theta(x_{j,i}^v) \log \frac{\pi_\theta(x_{j,i}^v)}{\pi_\phi(x_{j,i}^v)} \Big] \tag{3}$$

where $x_{j,i}^v$ denotes the probability of v-th vocabulary entry at the i-th token in the j-th sample. Simultaneously, we observe that CE loss can further enhance the average acceptance length in RL training. Therefore, we maintain a standard CE loss in Equation 1, whose contribution is modulated by a hyperparameter β. The overall loss is computed over trajectories x sampled from the main model.

$$\mathcal{L}_{\text{Adv-Opt}} = \mathbb{E}_{x \sim \pi_\theta} \left[\mathcal{L}_{\text{MTP-KL}}(x) + \beta \, \mathcal{L}_{\text{MTP-CE}}(x) \right] \tag{4}$$

We employ a joint optimization strategy to train the main model and the MTP module concurrently. The MTP module adopts an advantage-aware optimization strategy to minimize the deviation of its prediction distribution from the main model, while the total loss for the main model combines the RL objective with the MTP auxiliary loss. Losses for both the main model and the MTP module can be expressed in the following form:

$$\begin{cases} \mathcal{L}_{\text{Main-loss}} = \mathcal{L}_{\text{RL-loss}} + \mathcal{L}_{\text{Adv-Opt}} \\ \mathcal{L}_{\text{MTP-loss}} = \mathcal{L}_{\text{Adv-Opt}} \end{cases} \tag{5}$$

Ultimately, the model proceeds to the RL training under our proposed advantage-aware strategy. The complete training procedure is formally detailed in Appendix E, and this co-training strategy guarantees that the MTP modules maintain synchronization with the evolving main model, preventing "acceptance collapse".

4 Experiments

Models, Algorithms and Baselines. We conduct experiments using the popular Qwen2.5 series as our main models, explicitly focusing on Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct. To evaluate robustness and universality of our proposed method, we employ two distinct RL algorithms (token-level and sequence-level): group relative policy optimization (GRPO) and generalized speculative policy optimization (GSPO). For comparison, we include five baselines:

- **Vanilla RL**, which performs RL training without any rollout acceleration.
- **MTP w/o Training**, where MTP modules are frozen during RL.
- **MTP w/ Naive CE**, where MTP modules are trained with CE loss during RL.
- **MTP w/o Backprop**, where MTP modules are trained while preventing the gradients from back-propagating to the main model.
- **Lookahead (Zhao et al., 2024a)**, a train-free SD method for rollout acceleration.

Evaluation Metrics. We evaluate the effects of our method with respect to efficiency and performance across different models and algorithms. When verifying draft tokens, we employ strict acceptance criteria rather than the more lenient conditions used in Medusa (Cai et al., 2024) to provide a more rigorous assessment of model performance.

- **Average Acceptance Length (τ):** The average number of tokens accepted by the main model at each RL step, where each draft-verification cycle comprises three draft forwards.
- **Acceptance Rate (α):** The proportion of speculative tokens that are ultimately accepted, averaged across 20 RL steps to mitigate variance.
- **Speedup:** The acceleration of total rollout time in RL compared to the vanilla RL baseline.
- **Accuracy:** Performance on benchmarks to assess whether the MTP training strategy compromises model capabilities.

Model	RL Algorithm	Training: DAPO-Math, Testing: MATH-500				Training: KodCode-Light-RL, Testing: HumanEval			
		Speedup ↑	τ ↑	α ↑	Accuracy ↑	Speedup ↑	τ ↑	α ↑	Accuracy ↑
7B	GRPO	1×	1.00	0.00%	76.8	1×	1.00	0.00%	82.9
	+ MTP w/o Training	1.38×	2.32	39.99%	–	1.29×	2.38	46.89%	–
	+ MTP w/ Naive CE	1.44×	2.42	47.34%	–	1.26×	2.39	48.21%	–
	+ MTP w/o Backprop	1.16×	2.31	42.76%	–	1.21×	2.36	46.29%	–
	w/ Lookahead	1.42×	1.87	29.24%	76.0	1.14×	1.40	13.87%	82.3
	↪ + MTP-RL	**1.60×**	**2.55 (+0.31)**	**51.02%**	76.2	**1.42×**	**2.54 (+0.17)**	**50.29%**	83.5
	GSPO	1×	1.00	0.00%	77.0	1×	1.00	0.00%	81.1
	w/ Lookahead	1.40×	1.85	28.84%	76.4	1.20×	1.79	26.33%	80.5
	↪ + MTP-RL	**1.51×**	**2.56 (+0.34)**	**52.64%**	77.2	**1.30×**	**2.58 (+0.20)**	**52.40%**	81.1
14B	GRPO	1×	1.00	0.00%	79.8	1×	1.00	0.00%	85.4
	w/ Lookahead	1.97×	1.89	29.81%	77.4	1.14×	1.56	18.46%	85.3
	↪ + MTP-RL	**2.24×**	**2.46 (+0.57)**	**48.77%**	79.2	**1.50×**	**2.66 (+0.22)**	**54.40%**	84.2
	GSPO	1×	1.00	0.00%	79.6	1×	1.00	0.00%	85.3
	w/ Lookahead	2.03×	1.76	24.61%	79.2	1.17×	1.43	15.34%	84.1
	↪ + MTP-RL	**2.12×**	**2.47 (+0.57)**	**48.87%**	79.2	**1.59×**	**2.64 (+0.23)**	**53.20%**	84.2

Table 1: Rollout efficiency and effectiveness of optimization strategies are compared across different models and RL algorithms. MTP-RL achieves an average 1.30× - 2.24× acceleration and an additional 0.17 – 0.57 increase in τ during RL training, as reported in parentheses.

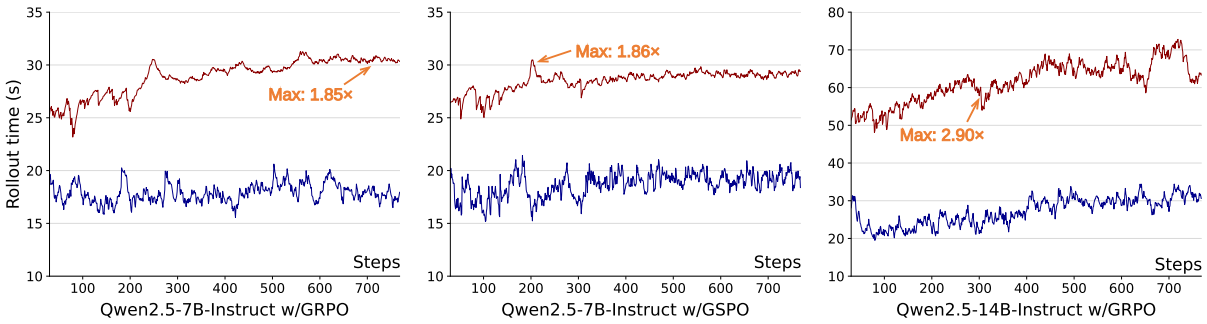

Figure 4: Comparison of rollout time across different models and algorithms in the math task. The upper curve shows the per-step rollout time without MTPs, while the lower curve shows the per-step rollout time under MTP-RL.

Implementation. Our experimental framework integrates three high-performance libraries: Slime (Zhu et al., 2025), SGLang (Zheng et al., 2024) and Swift (Zhao et al., 2024b). We utilize SGLang for both self-distilled data generation and efficient RL rollouts. Specifically, the main model is employed to synthesize trajectories, after which the MTP module is trained from scratch using Swift. For the RL stage, we adopt a prevalently used framework that combines Slime and SGLang: Slime handles RL training, and SGLang is responsible for rollout generation. To evaluate efficiency, we set the number of parameter-shared MTP layers to 3 in the MTP-equipped stage and perform three successive MTP forwards, followed by a single verification of the main model during RL. Other training hyperparameters are provided in Appendix A.

Datasets. We construct training responses generated by the main model and a composite prompt corpus to cover general instruction following capabilities, comprising 44K samples from ShareGPT, 25K samples from UltraChat200K (Ding et al., 2023), 22K samples from OpenThoughts-114k-math, and 22K samples from KodCode-V1-SFT-R1 (Xu et al., 2025). We perform RL training on the DAPO-Math-17K and KodCode-Light-RL-10K datasets for math and code reasoning, respectively, with evaluation on the MATH-500 (Lightman et al., 2023) and HumanEval (Chen, 2021) benchmarks for model capabilities.

4.1 Effectiveness

Efficiency. As presented in Table 1, we effectively train MTP modules in RL, and MTP-RL demonstrates consistent efficiency across different model scales and RL algorithms. For the math task, compared with native GRPO and GSPO, MTP-RL achieves rollout speedups of 2.24× and 2.12×, respectively, while the corresponding speedups on the code task are 1.50× and 1.59×. Meanwhile, as

displayed in the parentheses of τ, our method consistently improves the average acceptance length during RL, effectively preventing "acceptance collapse" and achieving increases ranging from 0.17 to 0.57 compared to MTP-equipped models without RL training on both math and code tasks. We measure the speedup against baselines to further validate the efficiency of our optimization strategy. The experimental results indicate that our method substantially outperforms the three strategies corresponding to the issues identified in Section 2, achieving a 10% improvement over the strongest alternative (from $1.38\times$ to $1.60\times$ on math and from $1.29\times$ to $1.42\times$ on code). Additionally, compared to lookahead, our approach delivers 4% – 44% improvements on both math and code tasks across different model scales and RL algorithms.

Figure 1 (b) and Figure 4 report rollout times per step, showing that our method can reach a maximum acceleration of 3.15× during RL with Qwen2.5-14B and GSPO. Notably, the acceleration effect scales positively with model sizes, and the Qwen2.5-14B series models experience even higher acceleration, as the computational overhead of the lightweight MTP module becomes increasingly negligible relative to the main model.

Capacity Acquisition. A critical concern in SD for RL is whether the auxiliary training objectives interfere with the reasoning capabilities of the main model. We evaluate model performance on Math-500 and HumanEval benchmarks after RL training. As shown in the accuracy columns of Table 1, our approach achieves lossless performance and in some configurations even exceeds the original RL method. These results confirm that our advantage-aware strategy preserves training efficiency while ensuring that the propagated gradient does not degrade model reasoning abilities.

4.2 Ablation Study

Ablation of Multi-layer MTP. We train MTP modules in Qwen2.5-7B-Instruct with different numbers of MTP layers and evaluate the initial τ achieved by it during the first step of RL. As shown in Table 2, we assume K denotes the number of layers and empirically observe that consistently increasing K enhances the initial τ of MTP modules, achieving a 27% improvement (from 1.76 to 2.25) over training a single-layer MTP module. Meanwhile, our approach attains comparable acceptance lengths to those of the unshared-parameter configuration, revealing the potential of multi-layer parameter sharing.

K	**1**	**2**	**3**	**4**	**3** + unshared parameters
τ	1.76	1.93	2.24	2.25	2.20

Table 2: Ablation studies in different layer numbers for MTP-equipped training

MTP Optimization Strategy	τ
MTP w/ Advantage-aware KL	1.20
MTP w/ CE + Naive KL	2.38
MTP w/ CE + Advantage-aware KL	2.49

Table 3: Ablation studies in different MTP optimization strategies

Ablation of MTP Optimization Strategy. We evaluate the effect of different MTP training strategies in the average acceptance length after 200 RL steps. Compared with "CE + Naive KL", "CE + Advantage-aware KL" incorporates the selection of positive-advantage samples into the KL divergence loss. As shown in Table 3, introducing the "Advantage-aware KL" enables our method to reach a higher average acceptance length more rapidly than the variants. We also observe that using advantage-aware KL loss alone leads to a sharp drop in the acceptance length, underscoring the necessity of including the CE loss.

4.3 Acceptance Length Dynamics

As shown in Figure 5, we compared evolutions of average acceptance lengths for MTP modules with different initial τ under varying numbers of K layers and draft structures during RL training. Experiments are conducted in Qwen2.5-7B-Instruct to keep the same model size. Despite the significant disparity in their initial τ values, all curves exhibit a similar trend and the acceptance length increases by approximately 0.3 to 0.4 after RL. This phenomenon demonstrates that our training method does not cause models with a high initial τ to increase less or even stagnate during RL.

4.4 Analysis of Total Training Cost

We further analyze the overall training cost of incorporating the MTP-equipped stage versus the savings in the RL rollout phase to provide a more complete picture of the net benefit. Additionally, we only conduct 0.4 RL epochs to demonstrate the effectiveness of our method under short RL runtime

Method	MTP-equipped stage time (min)	Rollout time (min)	Training time (min)	Total time (min)	Time reduction ratio	Accuracy
7B w/GRPO	-	361.75	319.92	681.67	-	76.8
7B w/MTP-RL	50.74	223.93	295.47	570.14	16.36%	76.2
14B w/GRPO	-	775.94	584.20	1360.14	-	79.8
14B w/MTP-RL	64.40	340.89	470.94	876.23	35.58%	79.2

Table 4: End-to-end training cost comparison of GRPO and MTP-RL. The evaluation accounts for the additional overhead introduced by the MTP-equipped stage.

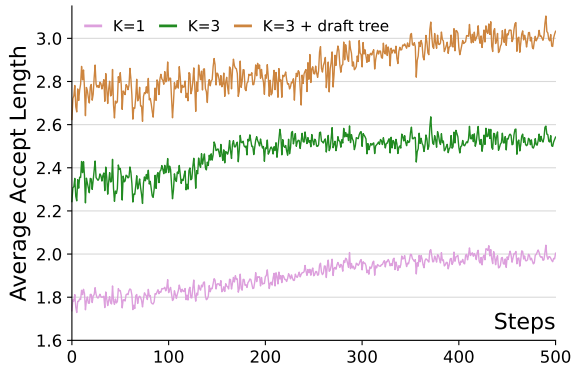

Figure 5: The variation of average acceptance lengths with different initial τ values during RL

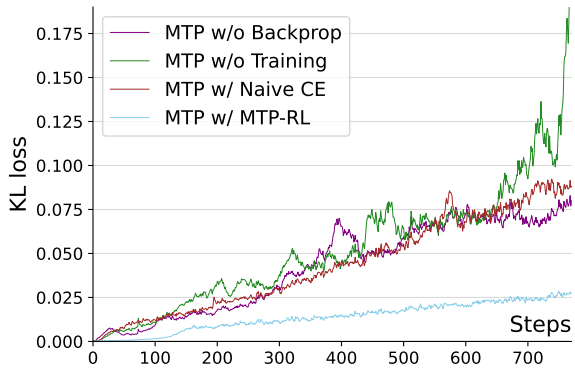

Figure 6: The variation tendency of KL loss among different MTP training strategy in RL

settings. We measure the actual training time required of Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct with GRPO and our proposed MTP-RL on the math task. As detailed in Table 4, despite the relatively short RL training duration, our method achieves a 16% - 35% reduction in total training time, demonstrating a clear advantage in training efficiency.

4.5 Analysis of KL Loss Trends

As shown in Figure 6, we compare trajectories of KL divergence under different MTP training strategies. Our method exhibits the most stable KL loss curve between the main model and the MTP during training, in contrast to baseline approaches that carry fluctuating KL loss. This stability indicates that the MTP module is able to closely track the evolving distribution of the main model and maintain consistent alignment during RL, further providing indirect evidence that our method avoids the "acceptance collapse" phenomenon.

5 Related Works

5.1 Efficient Reinforcement Learning

RL has become a central paradigm for aligning LLMs with expert preferences. Vanilla RL pipelines for LLMs, such as PPO (Schulman et al., 2017), typically consist of three stages: *generation*, *evaluation*, and *optimization*. Improving end-to-end training efficiency requires addressing bottlenecks that arise across these stages.

The evaluation and optimization stages are primarily constrained by synchronization delay and memory overhead. In the optimization stage, maintaining multiple models with distinct roles substantially increases memory consumption. To alleviate this issue, algorithmic innovations such as DPO (Rafailov et al., 2023) and SPO (Lou et al., 2024) eliminate explicit reward models, while GRPO (Shao et al., 2024) and GSPO (Zheng et al., 2025) further remove the critic model, performing token-level and sequence-level optimization, respectively. Meanwhile, evaluation efficiency is severely degraded by long-tail rollout trajectories, which is commonly referred to as the bucket effect. System-level solutions, including AgentRL (Zhang et al., 2025) and Seer (Qin et al., 2025), address this issue by decoupling generation from training via asynchronous frameworks to execute in parallel.

5.2 Speculative Decoding

Speculative Decoding (SD) (Leviathan et al., 2023; Xia et al., 2023) is a widely adopted inference acceleration paradigm that leverages a draft-then-verify mechanism to mitigate the memory bound of serial decoding by increasing computations. Existing SD methods can be broadly categorized into distillation-based and train-free approaches. Distillation-based SD relies on a separate and small

language model for drafting. The EAGLE (Li et al., 2024b, 2025) series focus on enhancing the input feature representation through the concatenation of several hidden states extracted from different layers of the target LLM. Eagle2 (Li et al., 2024a) and Medusa (Cai et al., 2024) propose tree-shaped attention mechanisms and dynamically adjust the draft tree based on draft probabilities, further accelerating attention computation in SD and improving the acceptance length. Train-free SD (Zhao et al., 2024a; Luo et al., 2025; He et al., 2024) relies on context retrieval and is particularly effective for tasks that follow consistent generation patterns. SAM-Decoding (Hu et al., 2025) leverages a suffix automaton to retrieve relevant draft tokens, thereby accelerating the construction of draft and improving overall decoding speed.

6 Conclusion

In this work, we successfully address the "acceptance collapse" issue of training MTP-equipped models in RL and accelerate the rollout phase. We propose a two-stage framework comprising lightweight training of a multi-layer parameter-sharing MTP, followed by an advantage-aware MTP optimization objective to achieve policy-aligned MTP training in RL. Extensive experiments demonstrate the effectiveness of our framework, achieving a maximum speedup of $3.15\times$ while maintaining consistent performance across various RL algorithms and model sizes. Furthermore, our method natively supports SD, accelerating downstream tasks without requiring any fine-tuning. Building on these efficiency gains, our future work will explore integrating this approach with various sparse attention mechanisms to push the limits of rollout acceleration in RL.

Limitations

While we adopt a lightweight approach to train the MTP module from scratch, it nonetheless introduces non-negligible engineering complexity and computational overhead. Additionally, although our experiments are conducted with a batch size of 256, the achievable speedup still differs from that observed under smaller batch sizes (e.g., batch size = 1), where SD yields substantially higher acceleration. Consequently, it remains an important problem to realize comparable speedups in large-batch settings.

Acknowledgments

This work is supported by the National Key Research & Development Plan (2023YFF0725100) and the National Natural Science Foundation of China (92570121, 62322214, U23A20299, U24B20144).

References

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

Yuxuan Cai, Xiaozhuan Liang, Xinghua Wang, Jin Ma, Haijin Liang, Jinwen Luo, Xinyu Zuo, Lisheng Duan, Yuyang Yin, and Xi Chen. 2025. Fastmtp: Accelerating llm inference with enhanced multi-token prediction. *arXiv preprint arXiv:2509.18362*.

Mark Chen. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Qiaoling Chen, Zijun Liu, Peng Sun, Shenggui Li, Guoteng Wang, Ziming Liu, Yonggang Wen, Siyuan Feng, and Tianwei Zhang. 2025. Respec: Towards optimizing speculative decoding in reinforcement learning systems. *arXiv preprint arXiv:2510.26475*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *Preprint*, arXiv:2305.14233.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jingkai He, Tianjian Li, Erhu Feng, Dong Du, Qian Liu, Tao Liu, Yubin Xia, and Haibo Chen. 2025. History rhymes: Accelerating llm reinforcement learning with rhymerl. *arXiv preprint arXiv:2508.18588*.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. 2024. Rest: Retrieval-based speculative decoding. In *Proceedings of the 2024 conference of the North American chapter of the association for computational linguistics: Human language technologies (volume 1: long papers)*, pages 1582–1595.

Yuxuan Hu, Ke Wang, Xiaokang Zhang, Fanjin Zhang, Cuiping Li, Hong Chen, and Jing Zhang. 2025. Sam decoding: Speculative decoding via suffix automaton. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12187–12204.

Wei Huang, Yi Ge, Shuai Yang, Yicheng Xiao, Huizi Mao, Yujun Lin, Hanrong Ye, Sifei Liu, Ka Chun Cheung, Hongxu Yin, and 1 others. 2025. Qerl: Beyond efficiency–quantization-enhanced reinforcement learning for llms. *arXiv preprint arXiv:2510.11696*.

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

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024a. 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.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Liyuan Liu, Feng Yao, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. 2025. Flashrl: 8bit rollouts, full power rl.

Xingzhou Lou, Junge Zhang, Jian Xie, Lifeng Liu, Dong Yan, and Kaiqi Huang. 2024. Spo: Multi-dimensional preference sequential alignment with implicit reward modeling. *arXiv preprint arXiv:2405.12739*.

Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, and Dongliang Xu. 2025. Turning trash into treasure: Accelerating inference of large language models with token recycling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6816–6831.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332.

Ruoyu Qin, Weiran He, Weixiao Huang, Yangkun Zhang, Yikai Zhao, Bo Pang, Xinran Xu, Yingdi Shan, Yongwei Wu, and Mingxing Zhang. 2025. Seer: Online context learning for fast synchronous llm reinforcement learning. *arXiv preprint arXiv:2511.14617*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. 2024. Direct multi-turn preference optimization for language agents. *arXiv preprint arXiv:2406.14868*.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2023. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925.

LLM Xiaomi, Bingquan Xia, Bowen Shen, Dawei Zhu, Di Zhang, Gang Wang, Hailin Zhang, Huaqiu Liu, Jiebao Xiao, Jinhao Dong, and 1 others. 2025. Mimo: Unlocking the reasoning potential of language model–from pretraining to posttraining. *arXiv preprint arXiv:2505.07608*.

Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. 2025. Kodcode: A diverse, challenging, and verifiable synthetic dataset for coding. *arXiv preprint arXiv:2503.02951*.

Shu Yang, Junchao Wu, Xin Chen, Yunze Xiao, Xinyi Yang, Derek F Wong, and Di Wang. 2025. Understanding aha moments: from external observations to internal mechanisms. *arXiv preprint arXiv:2504.02956*.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, and 1 others. 2025. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*.

Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, Bohao Jing, Iat Long Iong, Zhenyu Hou, Zehan Qi, Hanyu Lai, Yifan Xu, and 1 others. 2025. Agentrl: Scaling agentic reinforcement learning with a multi-turn, multi-task framework. *arXiv preprint arXiv:2510.04206*.

Yao Zhao, Zhitian Xie, Chen Liang, Chenyi Zhuang, and Jinjie Gu. 2024a. Lookahead: An inference acceleration framework for large language model with lossless generation accuracy. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6344–6355.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. 2024b. Swift:a scalable lightweight infrastructure for fine-tuning. *Preprint*, arXiv:2408.05517.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, and 1 others. 2024. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. 2025. slime: An llm post-training framework for rl scaling. https://github.com/THUDM/slime. GitHub repository. Corresponding author: Xin Lv.

A Configurations for Training

Train-from-scratch configurations. The MTP module is initialized and trained from scratch. Key hyperparameters, including the optimizer, learning rate, batch size, training epochs and the loss-weighting hyperparameters for each MTP layer are summarized in Table 5.

Hyperparameter	Value
Epoch	5
batch size	16
Temperature	0.8
Learning rate	5e-5
Optimizer(AdamW)	$(\beta_1, \beta_2) = (0.9, 0.95)$
λ_1, λ_2, λ_3	0.6, 0.3, 0.1

Table 5: Train-from-Scratch hyperparameters for the MTP module

RL configurations. During RL, the MTP module is trained with settings summarized in Table 6. The parameter β controls the scaling of the $L_{\text{MTP-CE}}$ loss, while a top-k value of 1 indicates that we employ the draft sequence instead of the draft tree.

Hyperparameter	Value
β	0.2
Top-k	1
Clip range	0.2 – 0.28
Learning rate	1e-6
Prompts per step	8
Sampling temperature	0.8
Candidate responses per prompt	32

Table 6: RL hyperparameters for the MTP modules

B Lightweight Train-from-scratch

During the train-from-scratch of MTP, only a selected proportion of model parameters are updated, reflecting a lightweight training strategy. This approach reduces the computational cost and memory usage, as summarized in Table 7.

Models	Total Size (M)	Update Ratio
3B-Instruct	3171	2.69%
7B-Instruct	7874	3.29%
14B-Instruct	15098	2.17%

Table 7: Parameter update ratios for training the MTP module from scratch across different model sizes

Benchmark	Model	Accuracy	Speedup
Math-500	GRPO	76.8	1.00×
	+MTP-RL	76.2	1.59×
AIME-2024	GRPO	20.0	1.00×
	+MTP-RL	20.0	1.53×
HumanEval	GRPO	82.9	1.00×
	+MTP-RL	83.5	1.45×
MBPP	GRPO	63.7	1.00×
	+MTP-RL	63.4	1.44×

Table 8: Accuracy and speedup of Qwen2.5-7B-Instruct with GRPO in downstream math and code benchmarks.

C Comparison of Speedups and Capacity in Downstream Tasks

As illustrated in Table 8, we compare the speedup and performance of Qwen2.5-7B-Instruct, trained with GRPO, and our proposed method across several downstream math and code tasks. The results demonstrate that MTP-RL consistently improves inference efficiency, achieving speedups ranging from 1.44× to 1.59×, while maintaining comparable performance across all tasks.

D Supplementary Evaluation on the GPQA Dataset

To further demonstrate the acceleration capability of MTP-RL on other reinforcement learning tasks and its effectiveness in addressing the acceptance collapse issue, we conduct experiments on the GPQA dataset, which is authored and verified by domain experts in biology, physics, and chemistry. We apply the GRPO algorithm to perform RL training on Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct. As shown in Table 9, MTP-RL achieves a speedup of 1.33×–1.65× on this task, while effectively mitigating acceptance collapse, yielding an increase of 0.24–0.39 in the average acceptance length. Moreover, our method matches or even surpasses the performance of the baselines, indicating that the proposed acceleration strategy does not compromise model quality.

E Algorithm of MTP-RL in Advantage-aware Optimization Stage

We describe the execution process of MTP-RL during the advantage-aware optimization stage.

Method	Speedup	τ	Accuracy
7B w/GRPO	1.00×	1.00	43.9
7B w/MTP-RL	1.33×	2.09 (+0.39)	43.9
14B w/GRPO	1.00×	1.00	55.5
14B w/MTP-RL	1.65×	1.91 (+0.24)	56.0

Table 9: Comparison of training efficiency between GRPO and MTP-RL on the GPQA dataset using Qwen2.5 models.

Algorithm 1 RL with MTP Optimization

Input: main model π_θ, MTP π_ϕ, Hyperparameters β, N_{epochs}
for epoch = 1 to N_{epochs} **do**
  Generate trajectory data $\mathcal{D}$ using π_θ (with π_ϕ for acceleration).
  Compute advantage values A_j for each sample in D.
  for minibatch in $\mathcal{D}$ **do**
    Compute $w(A_j) = \mathbb{I}[A_j \geq 0]$.
    Compute $\mathcal{L}_{\text{MTP-KL}}$ by Equation 3.
    Compute $\mathcal{L}_{\text{MTP-CE}}$ by Equation 1.
    Compute RL loss $\mathcal{L}_{\text{RL-loss}}$.
    Compute $\mathcal{L}_{\text{MTP-loss}}$ and $\mathcal{L}_{\text{Main-loss}}$ by Equation 5.
    Update θ via $\nabla_\theta \mathcal{L}_{\text{Main-loss}}$.
    Update ϕ via $\nabla_\phi \mathcal{L}_{\text{MTP-loss}}$.
  end for
end for
Output: Optimized policy π_θ and MTP π_ϕ.

F Trends in Average Acceptance Length across Models and RL Algorithms

In this section, we present the transformations in average acceptance lengths across all test models and algorithms during RL training on the math task, exhibited in Figure 1 (a) and Figure 7 – Figure 10. Across different model sizes ranging from 3B to 14B, and under two distinct training paradigms of GRPO and GSPO (token-level and sequence-level), our approach increases the average acceptance length by approximately 0.3 – 0.6, with the improvement being particularly pronounced in the Qwen-14B-Instruct model. These results demonstrate that our approach effectively resolves the "acceptance collapse" phenomenon and acquires a measurable improvement of average acceptance lengths with RL training.

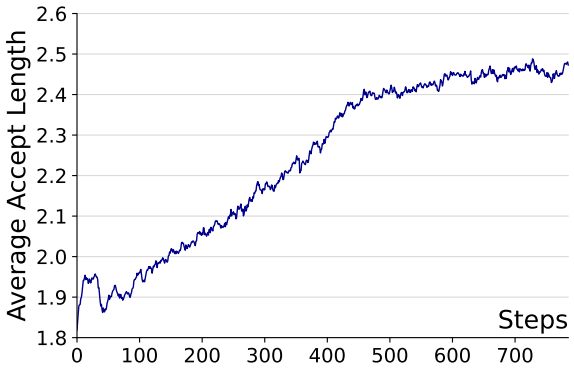

Figure 7: The variation of the average acceptance length in Qwen2.5-14B-Instruct and GRPO during RL training

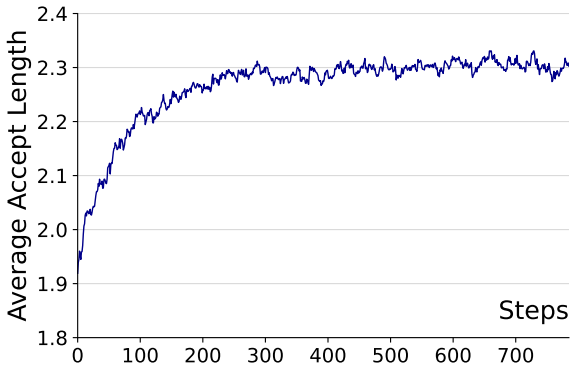

Figure 10: The variation of the average acceptance length in Qwen2.5-3B-Instruct and GSPO during RL training

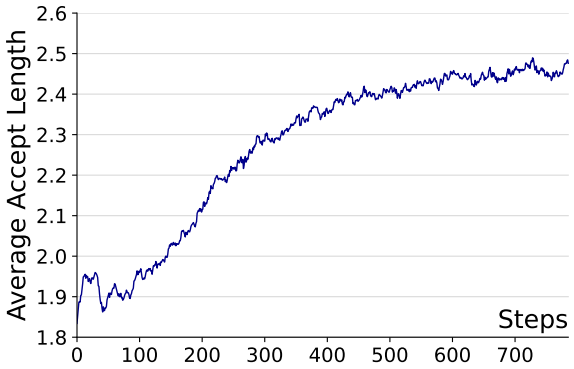

Figure 8: The variation of the average acceptance length in Qwen2.5-14B-Instruct and GSPO during RL training

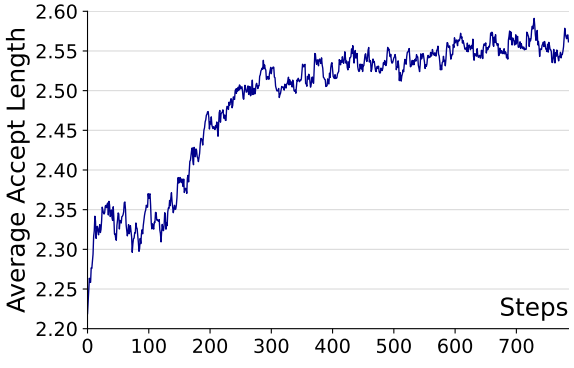

Figure 9: The variation of the average acceptance length in Qwen2.5-7B-Instruct and GSPO during RL training