SPARSE ATTENTION FOR EFFICIENT LLM REINFORCEMENT LEARNING

Yang Zhou[1*] Ranajoy Sadhukhan[1*] Zhaofeng Sun[2] Zhuoming Chen[1] Souvik Kundu[3]
Saket Dingliwal[4] Sai Muralidhar Jayanthi[4] Aram Galstyan[4] Haizhong Zheng[1] Beidi Chen[1]
[1]Department of Electrical and Computer Engineering, Carnegie Mellon University
[2]Independent Researcher [3]Intel [4]Amazon

ABSTRACT

Reinforcement learning (RL) is a key driver of recent progress in large language model reasoning, but its scalability is increasingly limited by the cost of online rollouts, especially for long chain-of-thought generation and large-batch sampling. Sparse attention is a promising way to reduce per-token attention cost and improve rollout throughput, yet we find that practical sparse rollouts often *destabilize training*: approximation errors bias likelihood estimates, causing large actor–policy distribution mismatch that compounds over long trajectories and can collapse training. We propose DISTILLSPARSE, a robust sparse-rollout framework that restores distribution alignment while preserving speed. DISTILLSPARSE co-trains a sparse rollout policy via lightweight, LoRA-based on-policy distillation from the dense policy to prevent mismatch from accumulating across RL iterations. For long generations and high sparsity, DISTILLSPARSE further oversamples rollout candidates and applies reward-aware filtering to focus updates on trajectories that are both high-quality and closer to the dense distribution. We evaluate on POLARIS across 4B–8B models and mathematical reasoning benchmarks including AIME24/25, AMC23, and Math500. Across settings where training-free sparse rollouts degrade or collapse, DISTILLSPARSE matches dense-rollout training performance while providing substantial practical acceleration, achieving a **1.72**× rollout speedup on NVIDIA H200 at 16K generation length with minimal overhead.

1 INTRODUCTION

Reinforcement learning (RL) has recently driven major gains in LLM reasoning and agentic behavior (OpenAI et al., 2024; DeepSeek-AI et al., 2025; Shao et al., 2024a). However, scaling RL remains fundamentally bottlenecked by the cost of online rollout generation (Zhang et al., 2025). This is getting worse as the field increasingly relies on long chain-of-thought (CoT) reasoning and parallel sampling at training time, which multiply the inference workload (Sadhukhan et al., 2025b). Recent evidence suggests that sparse attention can improve *test-time scaling* (Beeching et al.) by lowering per-token attention cost, enabling longer generations and more parallel samples under a fixed compute budget (Sadhukhan et al., 2025b). This makes sparse attention a promising rollout policy for producing high-quality RL trajectories at substantially lower latency.

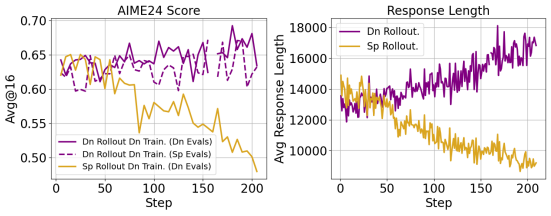

Figure 1: Qwen3-1.7B Thinking training with Polaris dataset. Compared to Dense rollout, naive sparse rollout lead to context length shrinking, even if the sparse rollout correctness is comparable to the dense rollout.

However, under realistic long output generation settings, we found that practical sparse attention for rollout faces great training stability challenges, illustrated in Figure 1. We observe that it is difficult to jointly achieve ***efficiency*** and ***training robustness*** through sparse attention. As shown in Figure 2 (c),

*Equal contribution

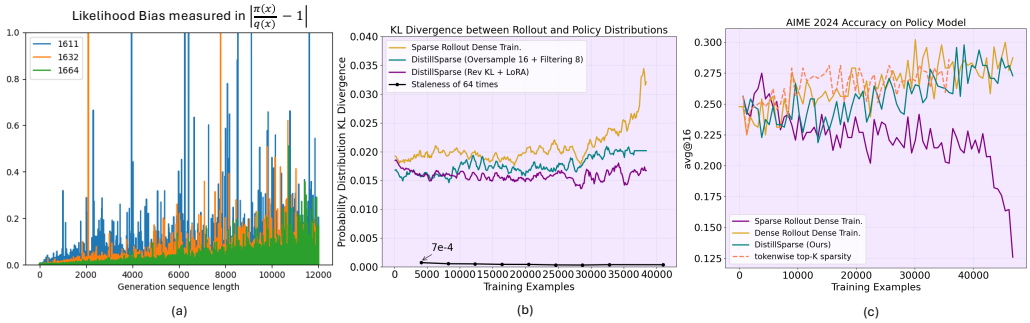

Figure 2: The main challenge to Sparse Attention for speeding up RL rollout lies in the distribution mismatch. (a) shows the bias in sparse attention estimating the importance ratio, a crucial term in RL training for weighing the advantage. The importance ratio should be exactly one, but we observe that in sparse rollout, the estimation of p_{ref} is usually biased, leading the importance ratio to deviates from 1. Also, as the generation length increases and as we lowers the sparsity ratio, the deviation exacerbates. (b) shows the striking contrast in KL divergence between the sparse rollout to dense training problem versus the commonly studied staleness problem in previous works on distribution alignment techniques: the difference in KL divergence is almost two orders of magnitude. We can also see that two core components in DISTILLSPARSE can individually reduce the KL divergence of the sparse rollout to dense. (c) shows that when combining the two techniques, DISTILLSPARSE can recover from crashing and achieve on-par results as dense rollout dense training baseline, while still preserving the sparse rollout efficiency.

a precise sparse attention algorithm like *exact* top-k can maintain training stability and performance, but it is impractical. Conversely, the practical alternatives with actual speedup benefits *shift the token distribution* substantially, resulting in training instability. Furthermore, this mismatch compounds with rollout length as the approximation errors accumulate over autoregressive steps. Consequently, the policy divergence grows across iterations, and rollouts tend to collapse to shorter, lower-diversity responses, preventing long-CoT behavior from emerging (Fig.1). A standard approach to mitigate this issue is to use importance sampling-based realignment techniques (Ionides, 2008; Team et al., 2025a; Liu et al., 2025b). However, they are unable to stabilize training. As these techniques are only *effective when the importance weights stay reasonably close to 1* (Zheng et al., 2025a), which is not satisfied by sparse attention policies as demonstrated in Fig. 2a. Alternatively, the distribution mismatch can be nullified by speculative decoding using sparse attention-based self-speculation or a small draft model. But the large verification overhead makes it intractable for large batch sizes commonly observed in RL rollout workload (Liu et al., 2025d; Sadhukhan et al., 2025a). Thus we observe that there is a tradeoff between efficiency and effectiveness.

These observations suggest that an ideal rollout strategy should remain **(1)** keep to actor-policy distribution between sparse and dense small for *training stability*, **(2)** *substantially faster* than dense rollouts without adding excessive *overhead* to the overall RL pipeline, and **(3)** maintain *training accuracy* and *sample efficiency*.

One fundamental reason behind sparse rollout and dense training instability is the dynamics and continuous nature of RL training. Unlike inference, approximation errors in sparse rollouts tend to **accumulate** across training iterations: every step, the approximation bias causing the policy model to deviate from its counterpart that trained on dense rollout, which exacerbates the the training instability and actor-policy misalignment. As shown in Fig. 2b, the KL divergence between the sparse rollouts and dense policies increase across iterations and eventually explodes, causing training to crash. Moreover, RL training for reasoning makes the situation worse for sparse, as the response lengths will **continously grow over time**. Unfortunately, sparse attention likelihood approximation bias increases with longer response length (Fig. 2a), causing the training signal for tokens at long sequence length chaotic to learn. The noisy training signals trigger the sparse rollout training to see response length continously dropping till training collapse, as shown in Figure 1.

The training dynamics suggest that the actor must actively bridge the distribution mismatch, which motivates training it via on-policy distillation from the dense policy. As shown in Fig. 2b, distillation

effectively prevents the KL divergence from exploding over training. That said, additional actor training can substantially increase memory overhead and, if implemented naively, may erode the cost savings of sparse rollouts. This makes careful system design essential. Moreover, even with KL regularization toward the dense policy, rollouts under extreme sparsity remain fragile: as response lengths grow, the training process remains prone to collapse. Fortunately, sparse attention exhibits a useful property: increasing the number of rollout samples can improve trajectory quality in a cost-effective manner Sadhukhan et al. (2025b). Oversampling increases the chance of encountering trajectories that better match the dense-policy distribution, thereby reducing mismatch. However, simply training on all oversampled rollouts does not resolve instability. *How can we efficiently identify the favorable trajectories that truly stabilize training?* Empirically, we observe that trajectories closer to the dense policy tend to achieve higher reward. Motivated by this, we filter low-quality rollouts from the additional samples, providing a stronger training signal while avoiding unnecessary overhead.

Putting these pieces together, we propose DISTILLSPARSE . It uses sparse-attention rollouts for efficiency and co-trains a sparse sampler via lightweight LoRA-based on-policy distillation to stabilize training with minimal overhead. For very long CoT rollouts and high sparsity, DISTILLSPARSE oversamples candidates per prompt and applies reward-aware filtering before updates, improving both reward (Fig. 4b) and distribution alignment (Fig. 4c). Overall, DISTILLSPARSE matches the stability and performance of dense RL training while retaining the rollout speedups of sparse attention.

We validate these insights on POLARIS (An et al., 2025) across 4B–8B models and evaluate on mathematical reasoning benchmarks (AIME24/25, AMC23, Math500). *Training-free* sparse rollouts show increasing mismatch and eventually collapse, degrading accuracy and shortening generations. In contrast, DISTILLSPARSE matches dense training without collapse. Under extreme sparsity, oversampling with reward-aware filtering further improves performance and robustness. As a by-product, the co-trained sparse sampler becomes a strong sparse policy, outperforming baselines trained from (i) a stronger dense policy or (ii) a minimally mismatched sparse policy.

Empirically, DISTILLSPARSE matches dense-rollout performance across downstream tasks and delivers a 1.72× speedup over the dense baseline on NVIDIA H200 at 16K generation length, with minimal overhead.

2 BACKGROUND

We will only go through the most relevant techniques and background context to our problem. We provide extended related works in Appendix A. Also, we provide detailed rollout cost analysis in Appendix B.

2.1 DISTRIBUTION MISMATCH CORRECTION IN RL.

Actor (i.e., the rollout model) and policy (i.e., the trained model) mismatch is a common problem that has long been studied (Espeholt et al., 2018). To alleviate the actor–policy distribution gap, prior methods go beyond clipping the importance sampling ratio and apply token-level scaling or masking functions to improve training stability (like TIS and IcePop):

$$\mathcal{L}^{\text{IS}}(\theta) = \mathbb{E}_{y\sim\mu_{\theta_{\text{old}}}}\Big[\boldsymbol{F}\Big(\frac{\pi_{\theta_{\text{old}}}(y|x)}{\mu_{\theta_{\text{old}}}(y|x)}\Big)\cdot \ell_{\text{clip}}(r, A; \epsilon)\Big], \tag{1}$$

where $\mu_{\theta_{\text{old}}}$ is the distribution generated by the inference engine using old parameters θ_{old}. In the subsequent text, we express this distribution by μ for better readability. $\boldsymbol{F}$ is an adjustment function applying token-level scaling or masking. The new importance weights are not included directly into the clipped PPO surrogate as it can not mitigate the bias in the gradients and gives suboptimal performance. The choice of adjustment function $\boldsymbol{F}$ is a key design choice in prior work, ranging from simple identity scaling to truncated/masked importance-weight corrections (Fu et al., 2025; Zheng et al., 2025c; Yu et al., 2025; Zheng et al., 2025b; Team et al., 2025b).

On the other hand, rejection sampling-based methods have also been studied Liu et al. (2025b); Verine et al. (2024) to solve actor–policy misalignment problems. Specifically, Verine et al. (2024) proposed a rejection criterion and showed that under some conditions, rejection sampling-based

methods outperform TIS. Here is the objective:

$$\mathcal{L}^{\text{PPO-OBRS}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\mu(\cdot|x)}\Big[w(x,y)\,\mathcal{S}_{\text{PPO}}(x,y;\theta)\Big], \quad w(x,y) \triangleq \boxed{\text{Mask}(x,y)}\ \text{SG}\left(\boldsymbol{F}\left(\frac{\pi_\theta(y|x)}{\boxed{\mu'(y|x)}}\right)\frac{\pi_{\theta_{\text{old}}}(y|x)}{\pi_\theta(y|x)}\right). \tag{2}$$

where rejected samples are removed by $\text{Mask}(x, y)$, and accepted samples are reweighted according to $\mu'_{\theta_{\text{old}}}(y|x)$. The function $\boldsymbol{F}$ is the truncated IS correction, and empirically π_θ can be replaced by $\pi_{\theta_{\text{old}}}$ for distribution alignment. **Jackpot will be used in most of the experiments of the paper as the standard baseline for the training-free distribution alignment technique**.

3 Insights and Methodology

3.1 Sparse Attention Distribution Gap to Dense

The importance sampling approach in Eq. 1 should ideally be implemented at sequence level. However, because of numerical instability issues, it is approximated by token-level approaches i.e.,

$$\mathcal{L}^{\text{IS}}_{\text{token}}(\theta) = \mathbb{E}_{\substack{x\sim\mathcal{D}\\ y\sim\mu(\cdot|x)}}\left[\sum_{t=1}^{|y|}\text{SG}\left(\frac{\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})}{\mu(y_t \mid x, y_{<t})}\right)\cdot \ell_{\text{clip}}\big(r_t(\theta), A_t\big)\right]. \tag{3}$$

As argued by (Zheng et al., 2025a), this approximation works well only when the *probability ratio is close to 1*. However, as Fig. 2a demonstrates, the ratio diverges significantly from 1 as the generation length increases. While truncated importance sampling-based approaches find it difficult to identify a suitable truncation factor with the best bias-variance tradeoff across all generation lengths, the rejection sampling-based approaches lose sample efficiency.

We identify this as a fundamental flaw of practical sparse attention algorithms. Unlike exact top-k attention, they cannot closely match the dense attention distribution, even if they can produce rollouts with high rewards (Fig. 2c).

This difficulty prompts us to distill the dense attention policy into the sparse one in an online manner.

3.2 LoRA Distillation

To minimize the training overhead and additional memory footprint (for extra parameters, and optimizer states), we resort to LoRA-based distillation. We do not generate fresh trajectories using the dense policy as that will defeat the purpose of efficient training. Instead, we repurpose the existing sparse attention rollouts for this distillation process. The complete policy update process is described in Algorithm 1.

training overhead. The on-policy distillation process saves us the extra computation required for new trajectory generation. Furthermore, the dense attention log-probs used to compute dense policy update (step 4 in Algorithm 1) can be repurposed to compute the distillation loss in step 7. The only extra overhead comes from the additional forward pass through the sparse attention policy and a relatively inexpensive backward pass which only computes gradients for the LoRA modules.

3.3 Parallel Sampling and Filtering

To further improve the output quality for extreme sparse attention rollouts, we perform oversampling for the given prompts followed by a reward-score aware filtering. Suppose, for a given prompt x, the base algorithm generates n samples Ω, which are used to estimate the policy objective.

This decision is motivated by a simple observation. In Figure 4 (Middle), we rank oversampled trajectories by dense-model acceptance rate and evaluate the oracle Top-8 rewards. Although computing dense log-probabilities is prohibitively expensive, we find that increasing the oversampling factor N consistently raises the average reward of the dense-selected Top-8, while the average reward over all N samples stays roughly constant. This indicates a strong correlation between trajectories that are closer to the dense policy and higher rewards.

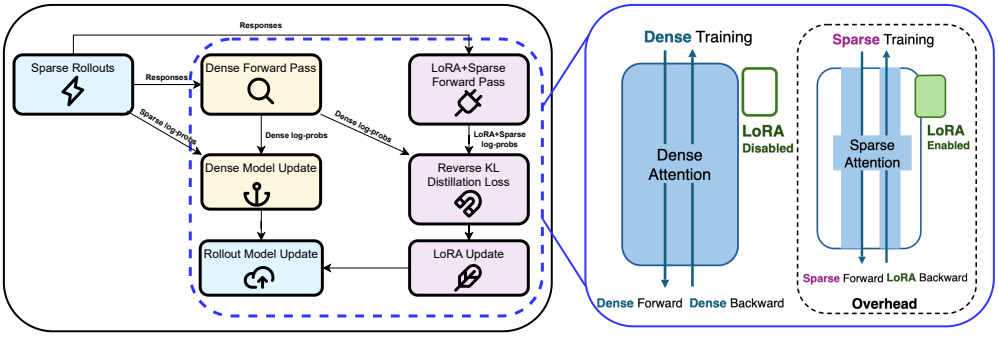

Figure 3: Illustrations of proposed LoRA sparse training.

Algorithm 1 Online LoRA distillation for sparse RL training

Require: Base model parameters θ; LoRA parameters ϕ; prompt batch $x \sim \mathcal{D}$

1: Construct **dense policy** $\pi_\theta^{\text{dense}}$ (full attention) and **sparse policy** $\pi_{\theta,\phi}^{\text{sparse}}$ (sparse attention + LoRA)
2: **for** each RL iteration **do**
3: Sample rollouts $y \sim \pi_{\theta,\phi}^{\text{sparse}}(\cdot \mid x)$ and compute rewards / advantages
4: Recompute token log-probs with the dense model (FSDP): $\log \pi_\theta^{\text{dense}}(y_t \mid x, y_{<t})$
5: Compute token log-probs with the sparse model: $\log \pi_{\theta,\phi}^{\text{sparse}}(y_t \mid x, y_{<t})$
6: Update the base model $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{PPO}}\big(\theta; x, y, \log \pi_\theta^{\text{dense}}\big)$
7: Freeze θ and update LoRA $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \mathcal{L}_{\text{LoRA}}\big(\phi; x, y, \theta\big)$ $\{\mathcal{L}_{\text{PPO\_Sparse}} + \lambda \mathcal{L}_{\text{KL}}\}$
8: Refresh inference engine weights with updated base model θ and LoRA ϕ
9: **end for**

$$\mathcal{L}(\theta, \Omega) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \ell_{\text{clip}}\big(r_{i,t}(\theta), \hat{A}_{i,t}\big)$$

where $\hat{A}_{i,t}$ are the group normalized advantage scores in GRPO-style training (Shao et al., 2024a). Instead, we generate M samples where $M > n$, and select the subset ω of size n with highest rewards to estimate the objective and the group normalized rewards.

$$\omega(x) \triangleq \Big\{ y_i \in \Omega(x) \;:\; \hat{A}_i \in \text{TopK}\left(\{\hat{A}_j\}_{y_j \in \Omega(x)}, n\right) \Big\},$$
$$\mathcal{L}(\theta, \omega) = \frac{1}{n} \sum_{y_i \in \omega} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \ell_{\text{clip}}\big(r_{i,t}(\theta), \hat{A}_{i,t}\big) \tag{4}$$

4 Empirical Analysis

We put DistillSparse through comprehensive evaluations. The second is subdividied into three aspects. First, we evaluate DistillSparse under three different scenarios of sparse attention for rollout to train dense policy models. Second, we show that oversampling in DistillSparse can be applied to scenarios outside sparse to dense settings. Third, we show that as a by-product of our DistillSparse , we also get a stronger sparse model.

Models and Datasets: Due to computation infeasibility, we cannot afford to iterate on thinking models where the output generation length is 32K on average. However, to the best of our effort, we approximate the ideal settings by training on top of warmup base models. We first train Qwen3-4B-Base and Qwen3-8B-Base (Yang et al., 2025) models on DeepScaleR Luo et al. (2025) dataset. The warmup is usually 200 steps, where we see the response length roses to roughly 4K under 8K

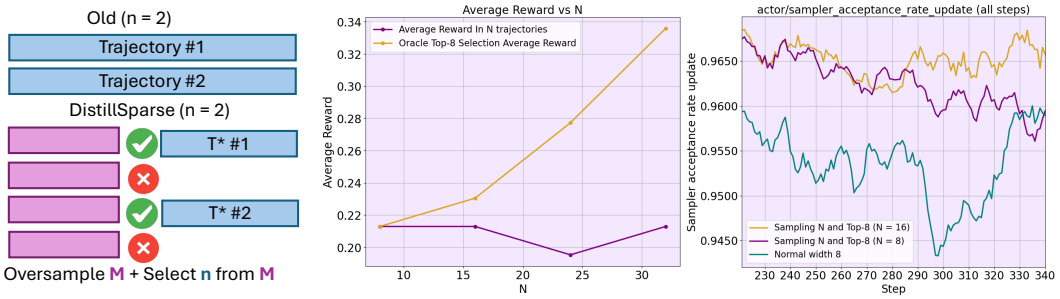

Figure 4: Demonstration of Increase rollout width and then Filtering on top. (a) shows the contrast between normal n width (n = 8 in our settings) rollout, almost free lunch from the system analysis in sparse inference; (b) show the oracle filtering strategy which is prioritizing traces closer in distribution with the policy naturally exhibit increase average accuracies which inspires our design; (c) shows that by only filtering based on the trajectory-level reward, we are able to decrease the distribution gap to the policy, and further increase rollout width N brings additional benefit

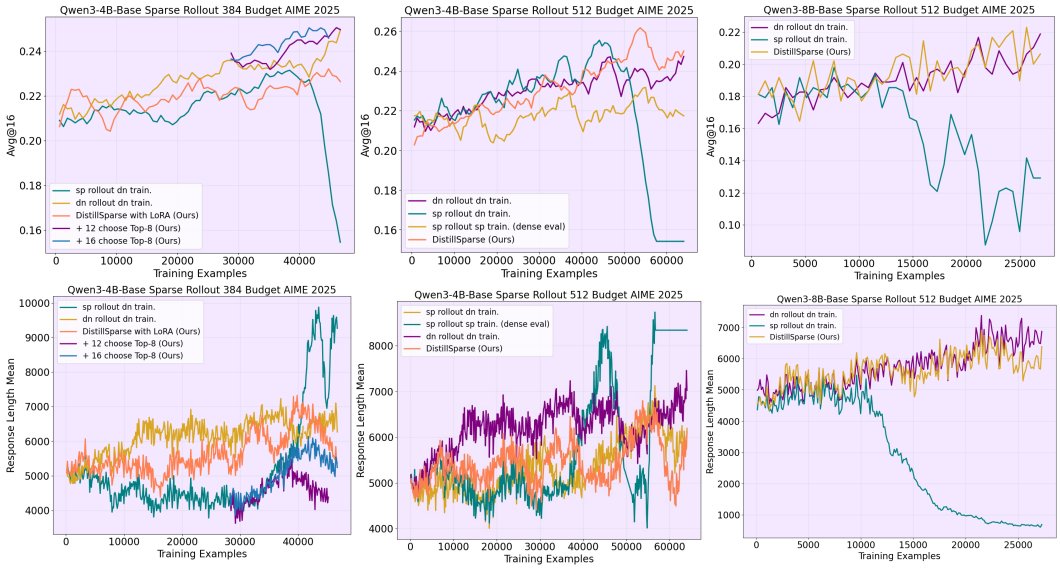

Figure 5: The training landscape of the performance of three different settings, for each of which, we present both the AIME25 performance and the response length dynamics; Left columns are from Qwen3-4B-Base with 16x24 sparsity (7%), where we demonstrate the effectiveness of oversampling; Middle column shows the 16x32 sparsity (10%) on 4B; Right column show the 16x32 sparsity (10%) on 8B (Yang et al., 2025).

warmup length limit, where we take the checkpoints to the harder polaris An et al. (2025) dataset, where we give 12K and 16K length limit to 4B and 8B models respectively. The training usually would see length scaling from 5K to beyond 8K. Single epoch of training takes 2xDGX (H100) nodes for 4B and 4xDGX (H100) nodes for 8B 5-6 days to complete, roughly for both models.

4.1 Sparse Rollout Dense Training through DistillSparse

Shown in Figure 5, we see that even with Jackpot (Liu et al., 2025b), a powerful distribution alignment method that uses both IS and RS, training-free sparse rollout either crashes or gets to suboptimal results. We also generally see that the sparse rollout dense training see shrinking rollout generation length, and blows up when policy crashes. Moreover, we see DistillSparse generally can help recover the distribution gap of the sparse rollout to dense, resulting in convergence and performance on par with the dense baseline. Specifically, on Figure 5, left-top, we see that continuously scaling

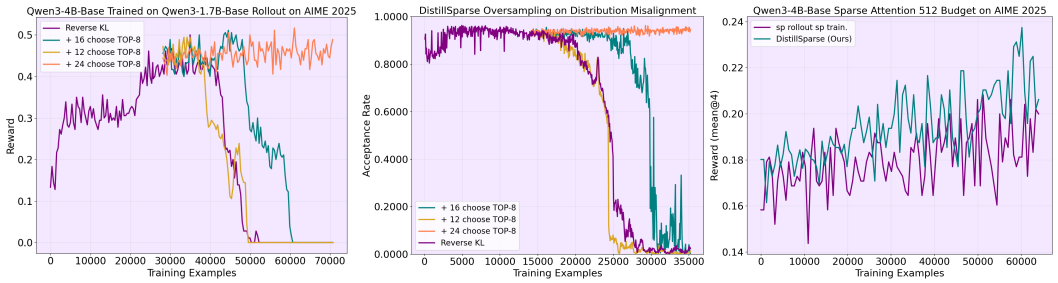

Figure 6: Oversampling can be applied outside sparse to dense training as well. Left and middle show that we take the same reward-based filtering oversampling to two completely different model training scenarios, where we steadily see the improvement of training stability once the computational cost of oversampling increases. Right shows that DISTILLSPARSE not only brings us a strong dense model, but it can also bring us a strong sparse model as well, benefiting from the reverse KL distillation.

up oversampling compute (from 8 to 12, then to 16) continuously leads to better performance. The findings validate the observation presented in Figure 4, right figure. Since the distribution mismatch is the problem behind RL training instability, one baseline is sparse rollout + sparse training, evaluated with dense attention at inference, as it keeps the actor-policy performance close. However, in evaluation, we see that the method consistently leads to lower performance for a lower sparsity budget than DISTILLSPARSE . To summarize DISTILLSPARSE overall performance, we conduct a comprehensive evaluation in Table 1, where we survey the best performance under each downstream task through the entire training run. In general, we see DISTILLSPARSE getting performance on par with dense rollout dense training across different downstream tasks and training settings.

4.2 SPARSE MODEL IMPROVEMENT

Benefited from the reverse KL training from the dense policy, the sparse models see improvement in performance. The two baselines we compare are the training-free sparse models from the dense rollout setting, while another one is the sparse rollout sparse training, a dedicated training run for the sparse model. The training progress situation is demonstrated in Figure 6 Right. We see that from the KL, the sparse model, after combining the LoRA weights into the model, achieves stronger downstream task performance on difficult tasks such as AIME25, and the performance improvement is consistent throughout the training. More evaluations are reported in Table 2.

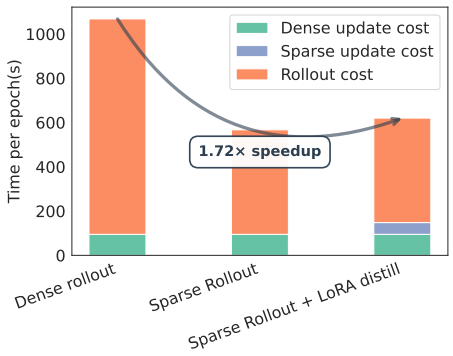

Figure 7: Cost breakdown of three RL training pipelines: dense-rollout training, sparse-rollout training, and DISTILLSPARSE . All measurements use a 16k-token response length, 8 parallel samples per prompt, a sparse KV budget of 512, a per-GPU micro-batch size of 2, and NVIDIA H200 GPUs.

4.3 CAN OVERSAMPLING BE APPLIED TO SETTINGS OUTSIDE SPARSE ROLLOUT?

We further show that oversampling plus reward-based filtering works well even under more general settings of weak model rollout and strong model training. Here, we select Qwen3-1.7B-Base model as the weaker rollout model, and the Qwen3-4B-Base model as a stronger training model. We train them using the DeepScaleR dataset. The training performance is shown in Figure 6 (a). The 1.7B model will roll out and onpolicy train itself together with a reverse KL from 4B, while the 4B model will train on the rollout sampled by the 1.7B model. The difference to sparse rollout is that the two models no longer tie weights.

With the Jackpot distribution alignment technique, the 4B model can train stably for roughly 300 steps, or 19k examples. Interestingly, if we ask the 1.7B model to apply oversampling + reward-based

Table 1: Comprehensive Evaluation of DISTILLSPARSE under three different settings, where we report the best-achieved scores under each downstream task and task-related metrics. DISTILLSPARSE consistently shows performance on par with the dense-to-dense training baseline.

Models	GSM8K Mean@4	MATH-500 Mean@4	AMC22/23 Mean@4	AMC12 Mean@4	AIME24 Mean@4	AIME24 Pass@4	AIME25 Mean@4	AIME25 Pass@4
Qwen3-4B-Base (After warmup) Under Block-size 16 and Number of block 24 (12k generation length cutoff, 53K Training Examples in Polaris)								
Qwen3-4B-Base (Dense Rollout)	0.9329	0.8480	0.6656	0.5944	0.3020	0.4134	0.2583	0.3433
Sparse Rollout Dense Training	0.92753	0.83	0.6265	0.5333	0.2750	0.3771	0.2364	0.3227
Sparse Rollout Sparse Training (dense eval)	0.92753	0.8345	0.5333	0.5444	0.2812	0.3945	0.2333	0.3072
DISTILLSPARSE (Ours LoRA Only)	0.9363	0.851	0.6747	0.5444	0.2895	0.4128	0.2500	0.3396
DISTILLSPARSE (Ours with Oversampling)	0.9304	0.838	0.6656	0.5666	0.2879	0.4200	0.2604	0.335
Qwen3-4B-Base (After warmup) Under Block-size 16 and Number of block 32 (12k generation length cutoff, 53K Training Examples in Polaris)								
Sparse Rollout Dense Training	0.9327	0.8350	0.6355	0.5333	0.2666	0.3954	0.2625	0.3625
Sparse Rollout Sparse Training (dense eval)	0.9289	0.8390	0.6475	0.5777	0.2916	0.4137	0.2375	0.3300
DISTILLSPARSE (Ours)	0.9332	0.8485	0.6656	0.5777	0.3104	0.4240	0.2666	0.3471
Qwen3-8B-Base (After warmup) Under Block-size 16 and Number of block 32 (16k generation length cutoff, 32k Training Examples in Polaris)								
Qwen3-8B-Base (Dense Rollout)	0.9304	0.837	0.6656	0.5944	0.3208	0.4536	0.2308	0.3067
Sparse Rollout Dense Training	0.9404	0.8225	0.6295	0.5555	0.2729	0.3875	0.1979	0.2733
DISTILLSPARSE (Ours)	0.9306	0.8355	0.6656	0.6166	0.3229	0.4646	0.2291	0.3131

Table 2: Evaluation of the Sparse Attention. The models are asked to generate responses using Sparse attention. We show that as a by-product of DISTILLSPARSE , we achieve a strong sparse model compared to training-free sparsity from dense rollout and a sparse rollout, sparse training scenarios.

Models	AIME25 Mean@4	AIME25 Pass@4	AIME24 Mean@4	AIME24 Pass@4
Qwen3-4B-Base (After warmup) Under Block-size 16 and Number of block 24 (length 12k)				
Dense Rollout Dense Training (Sparse)	0.1812	0.2314	0.2354	0.3176
Sparse Rollout Sparse Training	0.1666	0.2314	0.2354	0.3301
DISTILLSPARSE (Ours)	0.2166	0.2921	0.2395	0.3457
Qwen3-4B-Base (After warmup) Under Block-size 16 and Number of block 32 (length 12k)				
Sparse Rollout Sparse Training	0.2083	0.2882	0.2583	0.3588
DISTILLSPARSE (Ours)	0.2187	0.3151	0.2687	0.3613

filtering as in DISTILLSPARSE , we see better training stability continuously as the number of samples increases from 12 to 16, later to 24, where the training stays stable even after 550 steps, 35k training examples. Together, we also see that the acceptance rate improves as the oversampling scales up in Figure 6 (b), where if N = 24, the distribution between the two distributions will get closer as the acceptance rate continues to increase. We highlight the generality of the oversampling technique and its usefulness outside the sparse rollout and dense training domain.

System implementation & Efficiency: To conduct our experiments, we use Verl (Sheng et al., 2024) with FSDP as the training engine and SGLang (Zheng et al., 2024) as the inference engine. For efficient sparse-attention rollouts, we use Vortex_torch (Chen, 2025). We adopt block top-k attention with a page size of 16, and set the number of top-k pages according to the sparse KV budget. In addition, we use Flash Sparse Attention (Yan et al., 2025) for efficient sparse-attention training and PEFT (Mangrulkar et al., 2022) for LoRA adaptation.

Below we report the empirical efficiency of our implementation. As shown in Fig. 7, dense rollouts account for roughly 90% of the per-epoch time. Sparse attention directly alleviates this bottleneck and accelerates rollouts by more than $2\times$. Although the dense policy update contributes only about 10% of the total cost, our sparse distillation increases this component by approximately 55%. Overall, we obtain a $1.72\times$ end-to-end speedup.

5 CONCLUSION

We show that naive sparse-attention rollouts can destabilize dense-policy RL by introducing sequence-length–dependent likelihood bias. We propose DISTILLSPARSE , which stabilizes PPO by re-

computing dense log-probabilities and maintains alignment by distilling the dense policy into a sparse-attention LoRA, combined with reward-aware oversampling and filtering. Experiments on Qwen3-4B/8B achieve near-dense performance with major rollout-time savings, making sparse attention a practical lever in scaling long-context RL.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL https://hkunlp.github.io/blog/2025/Polaris.

Edward Beeching, Lewis Tunstall, and Sasha Rush. Scaling test-time compute with open models. URL https://huggingface.co/spaces/HuggingFaceH4/blogpost-scaling-test-time-compute.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023. URL https://arxiv.org/abs/2302.01318.

Zhuoming Chen. Vortex documentation, 2025. URL https://infini-ai-lab.github.io/vortex_torch/.

DeepSeek-AI. Deepseek-v3.2-exp: Boosting long-context efficiency with deepseek sparse attention, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018. URL https://arxiv.org/abs/1802.01561.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025.

Horace He and Thinking Machines Lab. Defeating nondeterminism in llm inference. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250910. https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/.

Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

Wei Huang, Yi Ge, Shuai Yang, Yicheng Xiao, Huizi Mao, Yujun Lin, Hanrong Ye, Sifei Liu, Ka Chun Cheung, Hongxu Yin, Yao Lu, Xiaojuan Qi, Song Han, and Yukang Chen. Qerl: Beyond efficiency – quantization-enhanced reinforcement learning for llms, 2025. URL https://arxiv.org/abs/2510.11696.

Edward L Ionides. Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 17(2):295–311, 2008.

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

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*, 2023.

Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. Clusterkv: Manipulating llm kv cache in semantic space for recallable compression, 2025a. URL https://arxiv.org/abs/2412.03213.

Hongyi Liu, Zhuoming Chen, Yang Zhou, Haizhong Zheng, and Beidi Chen. Jackpot: Optimal budgeted rejection sampling for extreme actor-policy mismatch rl. https://github.com/Infini-AI-Lab/jackpot.git, 2025b. Official GitHub repository.

Liyuan Liu, Feng Yao, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. Flashrl: 8bit rollouts, full power rl, August 2025c. URL https://fengyao.notion.site/flash-rl.

Xiaoxuan Liu, Jongseok Park, Langxiang Hu, Woosuk Kwon, Zhuohan Li, Chen Zhang, Kuntai Du, Xiangxi Mo, Kaichao You, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Turbospec: Closed-loop speculation control system for optimizing llm serving goodput, 2025d. URL https://arxiv.org/abs/2406.14066.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005b 2025. Notion Blog.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, Benjamin Bossan, and Marian Tietz. PEFT: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Alexandre Piché, Ehsan Kamalloo, Rafael Pardinas, Xiaoyin Chen, and Dzmitry Bahdanau. Pipelinerl: Faster on-policy reinforcement learning for long sequence generation, 2025. URL https://arxiv.org/abs/2509.19128.

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

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding, 2025a. URL https://arxiv.org/abs/2408.11049.

Ranajoy Sadhukhan, Zhuoming Chen, Haizhong Zheng, Yang Zhou, Emma Strubell, and Beidi Chen. Kinetics: Rethinking test-time scaling laws, 2025b. URL https://arxiv.org/abs/2506.05333.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024a. URL https://arxiv.org/abs/2402.03300.

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*, 2024b.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Qidong Su, Christina Giannoula, and Gennady Pekhimenko. The synergy of speculative decoding and batching in serving large language models, 2023. URL https://arxiv.org/abs/2310.18813.

Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding, 2024. URL https://arxiv.org/abs/2404.11912.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference, 2024. URL https://arxiv.org/abs/2406.10774.

Ling Team, Anqi Shen, Baihui Li, Bin Hu, Bin Jing, Cai Chen, Chao Huang, Chao Zhang, Chaokun Yang, Cheng Lin, Chengyao Wen, Congqi Li, Deng Zhao, Dingbo Yuan, Donghai You, Fagui Mao, Fanzhuang Meng, Feng Xu, Guojie Li, Guowei Wang, Hao Dai, Haonan Zheng, Hong Liu, Jia Guo, Jiaming Liu, Jian Liu, Jianhao Fu, Jiannan Shi, Jianwen Wang, Jianxin Lai, Jin Yang, Jun Mei, Jun Zhou, Junbo Zhao, Junping Zhao, Kuan Xu, Le Su, Lei Chen, Li Tang, Liang Jiang, Liangcheng Fu, Lianhao Xu, Linfeng Shi, Lisha Liao, Longfei Zheng, Meng Li, Mingchun Chen, Qi Zuo, Qiang Cheng, Qianggang Cao, Qitao Shi, Quanrui Guo, Senlin Zhu, Shaofei Wang, Shaomian Zheng, Shuaicheng Li, Shuwei Gu, Siba Chen, Tao Wu, Tao Zhang, Tianyu Zhang, Tianyu Zhou, Tiwei Bie, Tongkai Yang, Wang Hong, Wang Ren, Weihua Chen, Wenbo Yu, Wengang Zheng, Xiangchun Wang, Xiaodong Yan, Xiaopei Wan, Xin Zhao, Xinyu Kong, Xinyu Tang, Xudong Han, Xudong Wang, Xuemin Yang, Xueyu Hu, Yalin Zhang, Yan Sun, Yicheng Shan, Yilong Wang, Yingying Xu, Yongkang Liu, Yongzhen Guo, Yuanyuan Wang, Yuchen Yan, Yuefan Wang, Yuhong Guo, Zehuan Li, Zhankai Xu, Zhe Li, Zhenduo Zhang, Zhengke Gui, Zhenxuan Pan, Zhenyu Huang, Zhenzhong Lan, Zhiqiang Ding, Zhiqiang Zhang, Zhixun Li, Zhizhen Liu, Zihao Wang, and Zujie Wen. Every step evolves: Scaling reinforcement learning for trillion-scale thinking model, 2025a. URL https://arxiv.org/abs/2510.18855.

Ling Team, Anqi Shen, Baihui Li, Bin Hu, Bin Jing, Cai Chen, Chao Huang, Chao Zhang, Chaokun Yang, Cheng Lin, et al. Every step evolves: Scaling reinforcement learning for trillion-scale thinking model. *arXiv preprint arXiv:2510.18855*, 2025b.

Alexandre Verine, Muni Sreenivas Pydi, Benjamin Negrevergne, and Yann Chevaleyre. Optimal budgeted rejection sampling for generative models. In *International Conference on Artificial Intelligence and Statistics*, pp. 3367–3375. PMLR, 2024.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Bo Wu, Sid Wang, Yunhao Tang, Jia Ding, Eryk Helenowski, Liang Tan, Tengyu Xu, Tushar Gowda, Zhengxing Chen, Chen Zhu, Xiaocheng Tang, Yundi Qian, Beibei Zhu, and Rui Hou. Llamarl: A distributed asynchronous reinforcement learning framework for efficient large-scale llm training, 2025. URL https://arxiv.org/abs/2505.24034.

Ran Yan, Youhe Jiang, and Binhang Yuan. Flash sparse attention: More efficient natively trainable sparse attention. *arXiv preprint arXiv:2508.18224*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention, 2025. URL https://arxiv.org/abs/2502.11089.

Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, et al. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2o: Heavy-hitter oracle for efficient generative inference of large language models, 2023. URL https://arxiv.org/abs/2306.14048.

Chujie Zheng, Kai Dang, Bowen Yu, Mingze Li, Huiqiang Jiang, Junrong Lin, Yuqiong Liu, Hao Lin, Chencan Wu, Feng Hu, An Yang, Jingren Zhou, and Junyang Lin. Stabilizing reinforcement learning with llms: Formulation and practices, 2025a. URL https://arxiv.org/abs/2512.01374.

Haizhong Zheng, Jiawei Zhao, and Bedi Chen. Prosperity before collapse: How far can off-policy rl reach with stale data on llms? *arXiv preprint arXiv:2510.01161*, 2025b.

Haizhong Zheng, Yang Zhou, Brian R Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *NeurIPS*, 2025c.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs, 2024. URL https://arxiv.org/abs/2312.07104.

Yuzhen Zhou, Jiajun Li, Yusheng Su, Gowtham Ramesh, Zilin Zhu, Xiang Long, Chenyang Zhao, Jin Pan, Xiaodong Yu, Ze Wang, Kangrui Du, Jialian Wu, Ximeng Sun, Jiang Liu, Qiaolin Yu, Hao Chen, Zicheng Liu, and Emad Barsoum. April: Active partial rollouts in reinforcement learning to tame long-tail generation, 2025. URL https://arxiv.org/abs/2509.18521.

A Extended Related Works

We would like to divide the discussion of the related works into four aspects: RL for LLMs, prior works on distribution alignment in RL, general rollout speedup methods, and sparse attention application in LLM.

RL for LLM. Reinforcement learning has been widely applied to LLMs to improve human alignment, reasoning, coding, and other complex tasks. Beyond PPO, memory efficient methods have been proposed, including ReMax (Li et al., 2023), RLOO (Ahmadian et al., 2024), and GRPO (Shao et al., 2024b). In addition, methods such as SimPO (Meng et al., 2024) and DPO (Rafailov et al., 2023), which are based on offline RL, have also been employed for human alignment. RL training systems for LLMs, such as Verl (Sheng et al., 2025), AReal (Fu et al., 2025), TRL (von Werra et al., 2020), and OpenRLHF (Hu et al., 2024), have been developed to improve training throughput and scalability.

Distribution Mismatch Correction in RL. Actor-policy mismatch is a common problem that has long been studied, e.g. Impala Espeholt et al. (2018). To alleviate the actor-policy distribution gap, the method introduces a truncated importance sampling (TIS) to approximate the true PPO objective.

$$\mathcal{L}^{\text{PPO}}(\theta) = \mathbb{E}_{x\sim P_{\text{inf}}}\left[\min\left(\frac{p_{\text{ref}}(x)}{p_{\text{inf}}(x)}, C\right)\min\left(r_\theta(x)\,\hat{A}(x), \text{clip}(r_\theta(x), 1-\epsilon, 1+\epsilon)\,\hat{A}(x)\right)\right] \quad (5)$$

The truncation threshold C is for maintaining the stability of the range of the importance ratio. Recently, several methods apply the truncated importance sampling method to RL of LLMs. Methods such as FlashRL Liu et al. (2025c), AReal (Fu et al., 2025), and LlamaRL (Wu et al., 2025) address distribution mismatch by introducing (truncated) importance sampling ratios, typically of the form $p_{\text{ref}}/p_{\text{inf}}$, to correct the impact of mismatch on advantage estimation. From system perspective, FP32 LM heads Liu et al. (2025c) and deterministic LLM Inference (He & Lab, 2025) are implemented to mitigate the numerical issue of serving systems when rollout.

Prior Rollout Speedup Methods. Many recent works have been proposed to address this rollout efficiency challenge, but have several key limitations. Several recent works (Zheng et al., 2025b; Piché et al., 2025; Zhou et al., 2025) have designed asynchronous RL training systems that accelerate training by decoupling the rollout and training phases to better utilize computational resources. However, although this line of work prioritizes training throughput, the high resource utilization in the rollout phase remains unaddressed. Moreover, the asynchrony can result in staleness of samples which can potentially lead to inferior training. Another line of work aims to accelerate rollouts with model quantization (Liu et al., 2025c; Huang et al., 2025) and speculative decoding (Leviathan et al., 2023; Chen et al., 2023). Despite that model quantization can significantly reduce the cost of loading model weights, it cannot effectively mitigate the rollout overhead for long-sequence generation, where KV-cache loading remains the primary bottleneck (Sadhukhan et al., 2025b). Conversely, speculative decoding can accelerate rollouts without altering the sampling distribution. However, it is largely unsuitable for large-batch rollout setting (Liu et al., 2025d; Su et al., 2023) in RL training because the verification process becomes compute-intensive. Furthermore, speculative decoding introduces an additional draft model which requires extra training resources and thus complicates the whole training pipeline.

Sparse Attention. Attention operations cost dominates the latency of generating long-context output, consensus shared by many prior studies Sadhukhan et al. (2025b); Yuan et al. (2025). An extended amount of works have worked on reducing and alleviating the attention cost by pruning out redundant token computation and only spending time loading and computing on essential tokens. Training-free approaches revolve around fine-grained token-level decisions Zhang et al. (2023) or more accurate dynamic block-sparse attention Tang et al. (2024); Sun et al. (2024); Liu et al. (2025a). Despite robust performance in general tasks, under aggressive sparsity settings, these methods incur an unacceptable accuracy drop. Pretrained sparse attention methods Yuan et al. (2025); DeepSeek-AI (2025), on the other hand, are achieving scalable results.

B Rollout Cost Analysis and Sparse Attention Motivation

To understand why rollout generation dominates RL training, prior work models decoding cost as the combination of computation and KV-cache memory traffic (Sadhukhan et al., 2025b). We adopt the same additive view and analyze rollout latency independently.

For dense autoregressive decoding, the per-layer cost consists of parametric computation plus self-attention:

$$C_{\text{comp}} = 2PBNL_{\text{out}} \ + \ rBN(2L_{\text{in}} + L_{\text{out}})L_{\text{out}}D,$$

while memory traffic is driven by parameter and KV-cache access:

$$C_{\text{mem}} = 2PL_{\text{out}} \ + \ BN(2L_{\text{in}}L_{\text{out}} + L_{\text{out}}^2)D.$$

Although parameter access can be amortized with large batch size and model parallel inference, the attention term grows quadratically with response length. As a result, rollout decoding becomes increasingly memory-bound for long generations, and self-attention quickly emerges as the dominant bottleneck.

Sparse Attention. With a KV budget K, sparse decoding replaces the quadratic dependence on L_{out} with a budgeted cost:

$$C_{\text{comp}}^{(\text{sparse } K)} = 2PBNL_{\text{out}} + rBND \cdot H(L_{\text{in}}, L_{\text{out}}, K),$$

$$C_{\text{mem}}^{(\text{sparse } K)} = 2PL_{\text{out}} + BND \cdot H(L_{\text{in}}, L_{\text{out}}, K),$$

where

$$H = (2L_{\text{in}} + L_{\text{dense}})L_{\text{dense}} + 2K\big(L_{\text{out}} - L_{\text{dense}}\big)_+,$$

$$L_{\text{dense}} = \min(L_{\text{out}}, K - L_{\text{in}}).$$

Thus sparse attention avoids $\mathcal{O}(L_{\text{out}}^2)$ scaling and instead yields $\mathcal{O}(KL_{\text{out}})$ memory growth.

End-to-End Implications: An RL iteration decomposes into rollout generation, logits recomputation, and policy update:

$$C_{\text{total}} = C^{\text{rollout}} + C^{\text{recomp}} + C^{\text{update}}.$$

Recomputation and update operate on full sequences via prefill-style passes, achieving high arithmetic intensity and remaining largely compute-bound. In contrast, rollout decoding performs per-token KV access with low arithmetic intensity, making it fundamentally memory-bound.

Superior inference scaling with Sparse Attention. Overall, the rollout phase dominates end-to-end training time (often $> 90\%$, Figure 7), primarily because the KV loading cost increases quadratically with generation length (Sadhukhan et al., 2025b). Increasing number of rollouts can not amortize the decoding cost as the dominant KV memory also grows linearly with it. Sparse attention directly targets this bottleneck by reducing rollout memory cost from $\mathcal{O}(L_{\text{out}}^2)$ to $\mathcal{O}(KL_{\text{out}})$, where K is the sparse KV cache size. This allows us to increase the *generation length* and *number of samples* to achieve high-quality rollouts within reasonable cost. In other words, **sparse attention unlocks superior inference scaling in the usual long CoT regime.**