RL EXCURSIONS DURING PRE-TRAINING: HOW EARLY IS TOO EARLY FOR ON-POLICY LEARNING?

Anonymous authors
Paper under double-blind review

ABSTRACT

The standard training pipeline for Large Language Models (LLMs) proceeds sequentially through pretraining, supervised fine-tuning (SFT), and reinforcement learning (RL). Motivated by the success of RL in improving reasoning, we investigate the potential for introducing the RL objective at earlier stages of pretraining. Specifically, we study the effect of applying RL directly to intermediate pretraining checkpoints against SFT-only training and the conventional SFT $\rightarrow$ RL pipeline. Our results show that RL alone can substantially improve reasoning performance even when applied after only 25% of pretraining. Moreover, the direct RL approach can achieve performance comparable to the standard SFT $\rightarrow$ RL pipeline. We then analyze how early-RL affects output distribution (as measured by `pass@k`) and show two opposing cases of RL sharpening versus expanding the model's distribution. Finally, we explore the role of roll out budgets in optimizing performance at early stages of training. Overall, our findings offer novel insights into the effects and potential benefits of introducing RL earlier in the pipeline than the current standard practice.

1 INTRODUCTION

In modern Large Language Model (LLM) training, the standard recipe involves distinct pretraining and post-training stages. The current gold standard for post-training consists of Supervised Fine-Tuning (SFT) followed by Reinforcement Learning (RL). These stages contrast in their objectives: pretraining and SFT employ a Next-Token Prediction (NTP) objective via cross-entropy loss, whereas RL employs a policy optimization objective. When training with NTP, the LLM approximates a static, external dataset—effectively an *off-policy* regime. In contrast, when switched to the standard regime of RL training, the model learns from its own *on-policy* generations instead.

The use of two distinct training objectives raises a basic but underexplored question: at what point during training does an LLM become capable of learning from its own generations? Recently, there has been growing interest in expanding the use of RL for pretraining (Hatamizadeh et al., 2025; Li et al., 2025; Xing et al., 2025). As a pre-cursor, in this work we investigate the transition between off-policy and on-policy training objectives, asking a fundamental question:

How and when should an RL objective be used in LLM training?

To answer these questions, we present a rigorous case study of on-policy learning with a focus on LLM reasoning capabilities. Math provides a controlled setting with unambiguous rewards, making it a natural testbed for studying early RL viability. We pretrain a 1B parameter model from scratch on a high-quality, reasoning-heavy corpus, saving checkpoints $\mathcal{M}_t$ throughout the process. At each checkpoint $\mathcal{M}_t$, we train with RL on that checkpoint: **RL Only** ($\mathcal{M}_t^{\text{RL}}$). To quantify how effective the RL objective the reasoning capability of $\mathcal{M}_t$, we establish two baselines: (1) **SFT Only** ($\mathcal{M}_t^{\text{SFT}}$), and (2) **Standard Pipeline** ($\mathcal{M}_t^{\text{SFT}\rightarrow\text{RL}}$). We use $\mathcal{M}_t^{\text{SFT}}$as a NTP objective baseline; it is trained on the same questions used for RL but with human-written ground-truth solutions (i.e.,"off-policy" data). To obtain $\mathcal{M}_t^{\text{SFT}\rightarrow\text{RL}}$, we run RL on top of $\mathcal{M}_t^{\text{SFT}}$. This baseline represents the current gold standard. We perform evaluations on standard benchmarks: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021).

Our experiments offer three main insights into answering the above questions. *First*, as shown in Figure 1, RL is effective surprisingly early in pretraining. Training with RL significantly improves performance across datasets and metrics prior to pretraining on a large number of tokens. *Second*, contrary to recent findings

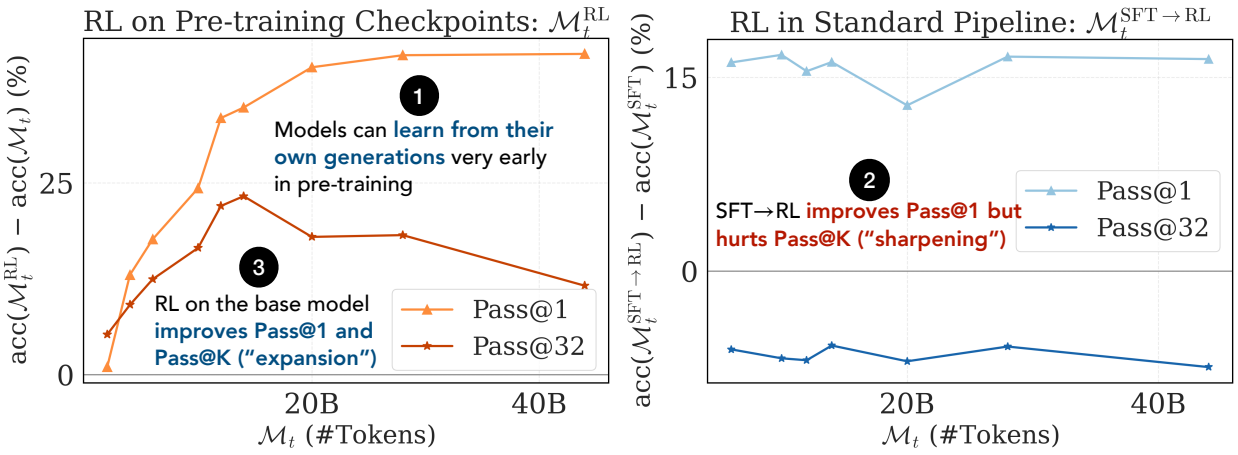

Figure 1: **Overview**. We analyze the effect of RL across intermediate pretraining checkpoints $\mathcal{M}_t$ and across two settings: RL directly on the base model (**RL Only**; $\mathcal{M}_t^{RL}$), and RL after SFT (**Standard Pipeline**; $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$). We observe: ❶ On-policy learning is effective starting very early during standard pretraining. $\mathcal{M}_t^{RL}$ models show significant improvement in both `pass@1` and `pass@k` metrics as soon as $2K$ steps ($\sim$4B tokens) of pretraining. ❷ In line with prior work, $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$ improves pass@1 performance over $\mathcal{M}_t^{\text{SFT}}$, but harms `pass@32` suggesting sharpening. ❸ In contrast, $\mathcal{M}_t^{\text{RL}}$ consistently leads to an increase in `pass@32` performance suggesting that RL can actually expand the model distribution to learn new capabilities.

that RL only sharpens the output distribution, we find that early stage RL considerably improves `pass@k` performance, indicating that RL can lead to distribution "expansion." In contrast, we also reproduce the sharpening effect in our standard pipeline settings. The two opposing results suggest that the effect of RL depends on training pipelines. *Third*, early pre-training checkpoints might yield sparse or noisy reward. Hence, we additionally study the effect of number of rollouts (n) to counter these sparse rewards. We observe that a larger n provides diminishing returns with compute and fewer rollouts could in fact be significantly more FLOP-efficient. Together, these findings suggest that early-stage RL objectives may be effective in improving downstream performance.

Our work serves as a proof-of-concept for *early-stage RL*. We restrict our attention to training and evaluating on math tasks in order to carefully control the environment. The results demonstrate the feasibility of applying RL objectives to what would typically be considered "under-trained" models, opening questions for future investigation regarding reward formulation, scaling behavior, and curriculum design.

Our main contributions are as follows:

- **RL improves performance early in pretraining (§3).** We observe that performing RL on early pre-training checkpoints lead to improved `pass@1` and `pass@k` performance across GSM8K and MATH benchmarks. Notably, these improvements are often comparable to the standard pipeline of pretrain→SFT→RL.
- **RL on the base model improves `pass@k` (§3.2).** Contrary to recent findings that RL only sharpens the LLM output distribution, we show that RL improves `pass@k` performance, hence indicating distribution expansion. Under two contrasting settings, we show the opposing effect of RL: directly training RL on pretraining checkpoints expands the distribution while the standard training pipeline leads to sharpening.
- **Effect of number of rollouts for RL (§4).** We provide empirical insights into how the rollout budget for GRPO affects learning across different stages of pre-training. Our results give practical insights into the appropriate number of rollouts across FLOP and sample budgets.

2 Methodology and Experimental Design

To answer how and when an RL objective should be used in LLM training, and whether we can use RL before the completion of standard pretraining, we establish a controlled experimental environment. Our setup

centers on a custom-trained 1B model, allowing for precise control over data exposure. In the following subsections, we outline our model configurations and define three training pipelines. While we explore reinforcement learning as a general objective, we focus our implementation on Reinforcement Learning from Verifiable Rewards (RLVR) using the Group Relative Policy Optimization (GRPO) algorithm (Shao et al., 2024) and the VeRL codebase (Sheng et al., 2024).

2.1 Pretraining Checkpoints

Choice of base model. While there are open-source models with pretraining checkpoints available (OLMo et al., 2024; Biderman et al., 2023), these models use internet-scale training sets on the order of trillions of tokens. Thus, it is difficult to precisely measure how much math and reasoning data is incorporated in the pretraining corpus (further discussion in §6). Instead of using these publicly available checkpoints, we pretrain a 1B parameter model from scratch using high-quality pretraining data curated by OLMo2 (OLMo et al., 2024). Specifically, we use the high-quality portion of their pretraining mix, DOLMino (OLMo et al., 2024)[1], which is used for stage 2 of OLMo's pretraining pipeline. The DOLMino mix contains 50B tokens, including general domains such as Wikipedia (7%), high-quality web data (60% from DCLM (Li et al., 2024) and FLAN (Wei et al.)), high-quality math data (20%), and other reasoning or code data such as StackExchange (2%) and STEM papers (5%).

Model and training. We use the OLMo2 (OLMo et al., 2024) architecture and training infrastructure to perform our pre-training. Specifically, we pre-train a 1B parameter decoder-only model on 50B tokens ($\sim 2.5\times$ Chinchilla optimal). We use AdamW (Loshchilov & Hutter, 2019) with a cosine learning rate decay and a peak learning rate of 4×10^{-4}. We train the model with a sequence length of 4096 and batch size 512.

2.2 Methods and Baselines

Data and metrics. We use OpenMathInstruct (Toshniwal et al., 2024) as the training data for SFT and RL. OpenMathInstruct consists of math questions and multiple ground truth solutions per question. In SFT, we train on solutions from the dataset, while in RL, the model generates its own solutions and is scored according to the final answer.

OpenMathInstruct contains two main categories of questions: the majority are inspired by the MATH dataset (Hendrycks et al., 2021), which consists of challenging competition-level math problems, while a minority are inspired by the GSM8K dataset (Cobbe et al., 2021), which consists of grade-school level math problems.

We consider two experimental settings to probe different aspects of RL training: **training with the full** OpenMathInstruct (MATH-heavy) and **training with the GSM8K-inspired subset** of OpenMathInstruct. On the GSM8K subset, the base pretraining checkpoints have already achieved non-trivial performance on many problems. In contrast, the full MATH-heavy training set contains problems that remain challenging even for later pre-training checkpoints, allowing us to examine how far different pipelines can push the model's reasoning capabilities. We evaluate the GSM8K training and MATH-heavy training experiments on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) respectively, and report the metric `pass@k` (Chen et al., 2021) for $k = 1, 8, 32$ at temperature $T = 0.6$. `pass@k` estimates the probability of obtaining *at least one* correct response when k responses are generated.

Procedure. Let $\mathcal{M}_t$ denote the pretraining model checkpoint at step t, and $\mathcal{M}_T$ denote the final, fully-pretrained model. We describe the three methods we compare below.

- **RL only ($\mathcal{M}_t^{\mathbf{RL}}$)** We start with $\mathcal{M}_t$ and train with the RL objective until convergence.
- **SFT only ($\mathcal{M}_t^{\mathbf{SFT}}$):** We start with $\mathcal{M}_t$ and perform SFT with ground-truth solutions until convergence.
- **Gold standard pipeline ($\mathcal{M}_t^{\mathbf{SFT}\to\mathbf{RL}}$):** After obtaining $\mathcal{M}_t^{\text{SFT}}$, we train with GRPO on the same set of questions.

[1] `allenai/dolmino-mix-1124`

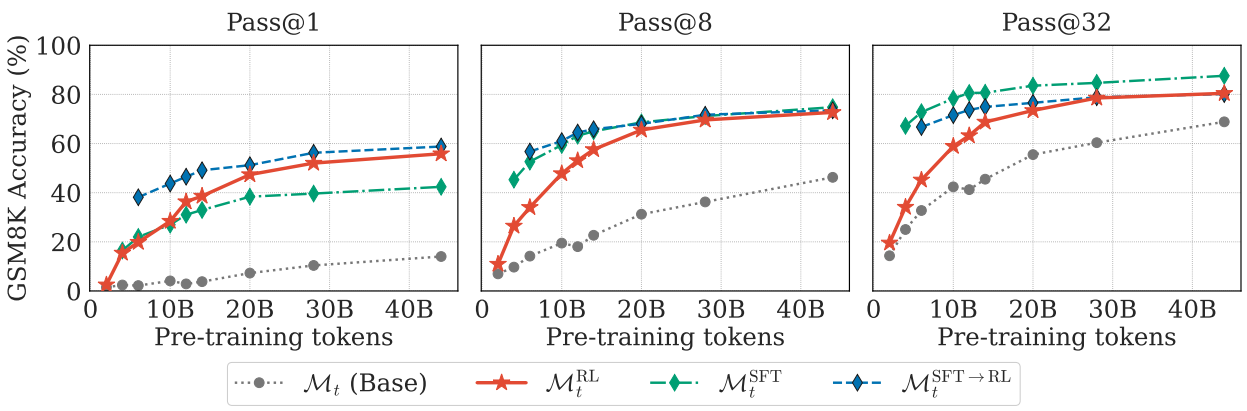

Figure 2: **GSM8K performance.** We report accuracy on `pass@k` performance for $\mathcal{M}_t$, $\mathcal{M}_t^{\text{SFT}}$, $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$ and $\mathcal{M}_t^{\text{RL}}$ evaluated on GSM8K, after training on the GSM8K-like subset of OpenMathInstruct. The base model is evaluated using 8-shot prompts, while all other models are evaluated with 0-shot prompts. For each method on top of $\mathcal{M}_t$, we report accuracy of the final converged checkpoint.

By comparing $\mathcal{M}_t^{\text{RL}}$ against $\mathcal{M}_t^{\text{SFT}}$, we isolate the training objective (RL vs. SFT) to determine if RL provides a superior training signal at early stages of model training pipeline. By comparing $\mathcal{M}_t^{\text{RL}}$ against $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$, we determine if RL alone can provide a superior training signal than the current gold standard.

In this work, we are interested in understanding, given sufficient compute, how well each method performs. Therefore, we train all our RL and SFT runs until convergence, and we confirm the convergence of training in Appendix A.1 and Appendix A.3.

3 Main Results

In Figure 2 and Figure 3 we report the performance of $\mathcal{M}_t$, $\mathcal{M}_t^{\text{RL}}$, $\mathcal{M}_t^{\text{SFT}}$, and $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$ at various pretraining steps t (measured in pretraining tokens) on GSM8K and MATH respectively. Since the pretraining checkpoint $\mathcal{M}_t$ has only seen the pretraining corpus, it cannot consistently follow question-answering instructions. Therefore, we use 8-shot in-context examples so that the pretraining checkpoints can follow the correct answer format. In Appendix A.4, we provide an ablation on the number of in-context examples and show that 8-shots yields the best performance for $\mathcal{M}_t$. For the remaining models, we use a formatting reward during RL and format OpenMathInstruct in the question-answering format for SFT baselines; hence, all other models ($\mathcal{M}_t^{\text{RL}}$, $\mathcal{M}_t^{\text{SFT}}$, and $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$) are able to follow the format without any in-context examples, and thus we default to using 0-shot evaluation.

3.1 RL is Effective Early in Pretraining

We now compare $\mathcal{M}_t^{\text{RL}}$ with the baselines. In Figure 2, we observe that, as early as $t = 4$B pretraining tokens, training with RL significantly improves the model's performance on GSM8K: for example, the `pass@1` accuracy increases from $\sim 2\%$ to $\sim 18\%$. Notably, the fact that this occurs at $t = 4$B tokens indicates *improvement with RL prior to reaching the Chinchilla optimal number of tokens* (Hoffmann et al., 2022). In addition, we observe a significant increase in `pass@k` for $k = 8, 32$, which we discuss in detail in §3.2.

RLVR competes with the gold standard pipeline on GSM8K. In Figure 2, after $t = 10$B tokens, $\mathcal{M}_t^{\text{RL}}$ *outperforms* $\mathcal{M}_t^{\text{SFT}}$ on `pass@1` and performs on-par with $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$. For `pass@8,32`, $\mathcal{M}_t^{\text{RL}}$ performs on-par with both $\mathcal{M}_t^{\text{SFT}}$ and $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$. This result is significant because $\mathcal{M}_t^{\text{RL}}$ never observes ground-truth reasoning traces; unlike the SFT baselines, it develops reasoning capabilities entirely from self-generated on-policy traces and feedback, demonstrating that RL can match supervised learning without training on ground-truth reasoning traces.

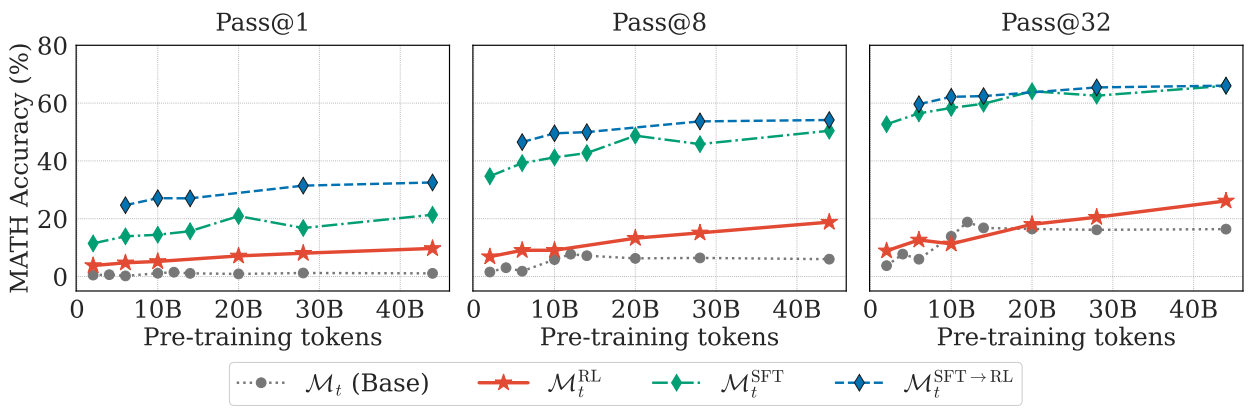

Figure 3: **MATH performance.** We report `pass@k` accuracy for $\mathcal{M}_t$, $\mathcal{M}_t^{\text{SFT}}$, $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$, and $\mathcal{M}_t^{\text{RL}}$ evaluated on MATH after training on full OpenMathInstruct. Unlike GSM8K (Figure 2), on this harder benchmark, $\mathcal{M}_t^{\text{RL}}$ brings substantial gains over $\mathcal{M}_t$ but fails to fully match the two baselines $\mathcal{M}_t^{\text{SFT}}$ and $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$, indicating a limitation of using RL early in pretraining.

Limitations on MATH. On MATH, the use of RL does not appear to have the same benefits. While RL also improves over the base model's performance prior to Chinchilla optimal number of tokens, RL does not eventually perform similarly to the use of SFT $\to$ RL. Specifically, we observe that, while $\mathcal{M}_t^{\text{RL}}$ consistently improves `pass@1,8,32` by 5% to 10% over the base checkpoint $\mathcal{M}_t$, it never catches up to SFT and SFT $\to$ RL. In other words, training on the RLVR objective (or, on-policy data) from pretraining checkpoints has limitations, potentially related to the difficulty of the task at hand.

3.2 Sharpening vs. Expansion with RL

We refer to sharpening (Wu et al., 2025; Yue et al., 2025) as the phenomenon in which training might improve `pass@1` accuracy but have little to no effect on `pass@k` accuracy for larger k. In contrast, we define expansion as the setting in which RL would increase `pass@k` performance across many values of k.

Early stage RL promotes expansion. Many recent works have claimed that RLVR mostly *sharpens* the distribution without bringing the model any "new" reasoning capabilities (Yue et al., 2025; Cheng et al., 2026). These works point to evidence that during RL, `pass@k` does not improve for sufficiently large k. Interestingly, in our experiments we observe two opposing effects depending on the training pipeline. First, when we apply the standard pretrain-SFT-RL pipeline on our pretraining checkpoints (i.e., $\mathcal{M}_t \to \mathcal{M}_t^{\text{SFT}} \to \mathcal{M}_t^{\text{SFT}\to\text{RL}}$), we observe the sharpening effect. In Figure 4 (*left*), we show one such example. We see that `pass@1` continues to improve from $\mathcal{M}_t$ to $\mathcal{M}_t^{\text{SFT}}$, and then during RL training. On the other hand, the SFT stage yields a significant gain in `pass@32`, but the subsequent RL training slightly decreases the performance.

We hypothesize that sharpening occurs because, during SFT, the model has already seen ground-truth solutions on the same set of questions, so RL primarily refines this existing capabilities rather than discovering new reasoning paths. In contrast, by directly training on the RL objective from the same pretraining checkpoint (Figure 4, *middle*), RL training improves both `pass@1` and `pass@32` performance, effectively expanding the base model's distribution. Without prior exposure to ground-truth solutions, the model explores and discovers new reasoning paths through on-policy learning.

Brittleness of RL on early checkpoints. Despite the promising results above, direct RL training exhibits significant instability on early pretraining checkpoints. While generating Figure 2, we observed that between $t = 4$B and $t = 10$B pretraining tokens, the performance of $\mathcal{M}_t^{\text{RL}}$ is highly non-deterministic across training seeds; for some seeds, $\mathcal{M}_t^{\text{RL}}$ yields significant performance improvements on GSM8K, while for others, it fails to improve over $\mathcal{M}_t$ at all. Consequently, for earlier checkpoints, we performed RL training across 4 seeds and report the best-performing seed in Figure 2. In Appendix A.2, we show that, across different RL runs (differing only by random seed), the models achieve comparable training rewards but diverge significantly

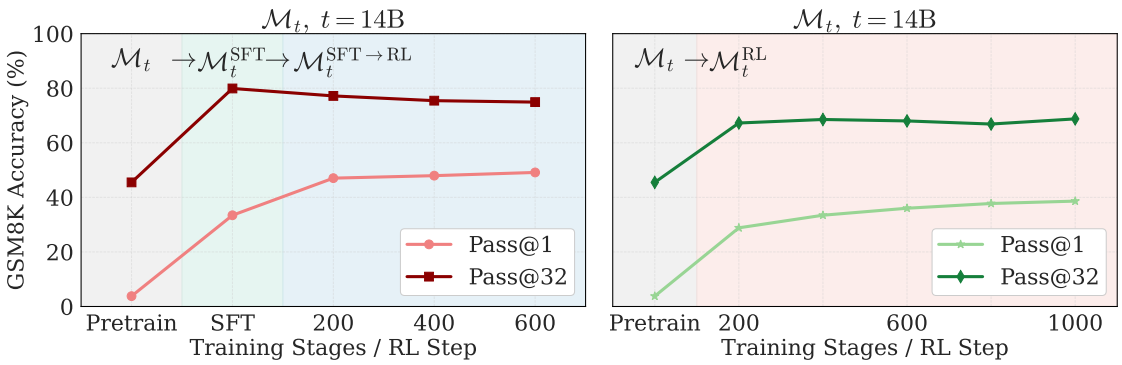

Figure 4: **RL training dynamics with and without SFT.** GSM8K accuracy (`pass@1` and `pass@32`) across training stages. *Left*: Standard SFT→RL pipeline shows sharpening (`pass@1` improves, `pass@32` decreases during RL). *Right*: Direct RL training expands distribution (both metrics improve).

in validation and test performance. This suggests that RL training on early checkpoints can sometimes lead to superficial pattern learning and memorization rather than the development of reasoning capabilities.

4 RL Rollouts

When training with RL on early pretraining checkpoints, the model is likely to have low `pass@k` accuracy on the training questions. Compared to the standard pipeline or a later pretraining checkpoint, applying RL at early pretraining *exacerbates* the reward sparsity problem. On these very early pretraining checkpoints, without sufficient positive samples (i.e., correct rollouts), the learning signal might become sparse or noisy, making it difficult for the model to improve.

A natural strategy to consider in order to obtain higher training signal is to sample a larger number of rollouts at each step in training. In this section, we comprehensively analyze this strategy and study the influence of number of rollouts for RL training. Specifically, we investigate the effect of varying the number of rollouts per prompt (n) in GRPO. We seek to determine if increasing the number of rollouts benefits models that are initially weak on the training distribution. To this end, we partition our training set into two sets: *a hard set* and *an easy set*, simulating early and later stages of pretraining respectively. We perform RL using GRPO on both these splits using settings with few ($n = 5$) and many ($n = 64$) rollouts and report `pass@k` accuracy on the standard GSM8K test set.

4.1 Experimental Setup

Data and Metrics. In order to simulate different stages of pretraining, we partition our training dataset based on proportion of positive samples per example. The OpenMathInstruct dataset (Toshniwal et al., 2024) is composed of questions inspired by either MATH or GSM8K training sets (for details see, §2.2). We focus on only the GSM8K-like subset of OpenMathInstruct (80K examples). To define the training splits based on "difficulty" level, we evaluate our base model on the original dataset in a zero-shot setting. For each question, we generate 64 responses at temperature 1 and record the number of correct solutions. We classify questions with 16 to 64 correct responses as *GSM8K-Easy*, and those with at most 8 correct responses as *GSM8K-Hard*. From these subsets, we randomly sample 10K questions for each split. We train with GRPO (as described in §2.2) and report `pass@k` ($\texttt{k} \in \{1, 8\}$) metrics on the standard GSM8K test set.

Model and Method. We conduct all experiments using the OLMo2 1B model (OLMo et al., 2024). We perform GRPO training for both GSM8K-Easy and GSM8K-Hard using $n = 5$ and $n = 64$ rollouts per prompt, while keeping all other hyperparameters constant. Consequently, $n = 64$ consumes significantly more FLOPs per RL step. To account for this trade-off, we analyze accuracy as a function of both total FLOPs consumed and the number of examples during RL training. For all settings, we train the models until the validation pass@1 metric converges.

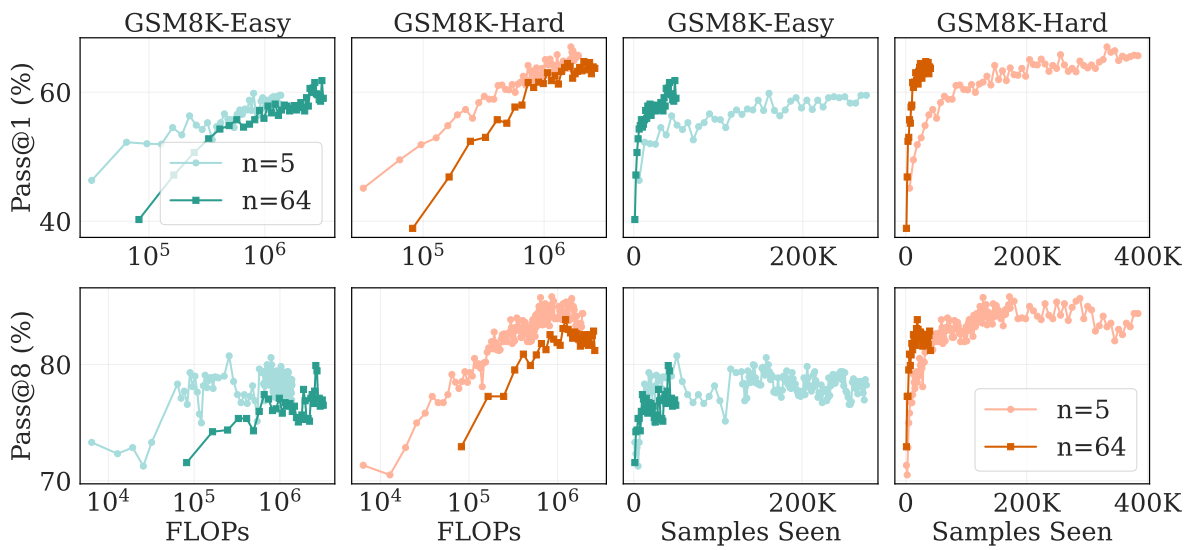

Figure 5: **Effect of number of rollouts on RL performance**. We report `pass@k` performance on the GSM8K test set. We sub-sample GSM8K-Easy and GSM8K-Hard training sets based on the proportions of positive rollouts from the base model on each example of GSM8K. The GSM8K-Hard set simulates an early pretraining setting, where the likelihood of sampling a positive rollout is very low. We observe that both n= 5 and n= 64 typically asymptote to a similar performance across training sets and `pass@k` metrics.

4.2 Main Results

We observe a distinct trade-off between sample efficiency and compute efficiency. As a function of *samples seen*, increasing the number of rollouts to $n = 64$ greatly improves `pass@1` convergence compared to $n = 5$. However, when viewed as a function of *FLOPs*, the lower rollout setting ($n = 5$) is more compute-efficient in the early stages of training. As training progresses toward 10^6 FLOPs, this efficiency gap narrows, with $n = 64$ eventually matching or surpassing the performance of $n = 5$. We observe that the difference between $n = 5$ and $n = 64$ rollouts further diminishes when observing `pass@1`. However, when we match FLOPs, we see that $n = 5$ appears to significantly improve upon $n = 64$, especially when training with GSM8K-Hard.

Our analysis in Figure 5 yields three primary insights regarding the scaling of RL rollouts. **First**, we find that asymptotic performance is largely independent of the number of rollouts; both $n = 5$ and $n = 64$ converge to similar `pass@k` peaks across difficulty levels. **Second**, there is a clear trade-off between sample efficiency and compute efficiency. Increasing the rollout count ($n = 64$) maximizes the utility of each training example, leading to faster convergence in terms of training steps. Conversely, reducing the rollout count ($n = 5$) is significantly more FLOP-efficient, achieving comparable performance with a fraction of the compute budget. **Finally**, this compute advantage is particularly pronounced on the *GSM8K-Hard* split for the `pass@8` metric, suggesting that when rewards are sparse (as with early checkpoints), massive rollout scaling may yield diminishing returns per FLOP compared to processing more batches with fewer rollouts.

5 Prior Work

Reinforcement Learning for Reasoning While the common regime of using next token prediction has been the default for some time, the works of Ouyang et al. (2022); Dai et al. (2023) are among the first to show the benefits of reinforcement learning in language model training. In particular, these works show the benefits of continuing to train with an RL objective *after* a model has been trained with the NTP objective. Since then, various methods such as DPO (Rafailov et al., 2023), PPO (Zheng et al., 2023), and GRPO (Shao et al., 2024) have been shown to be extremely effective in improving the downstream performance of LLMs. Typically, the data used to train with RL involves verifiable rewards such as math or coding tasks. One common view is that RL simply sharpens the distribution of the language model (Cheng et al., 2026; Yue et al., 2025; Wu et al., 2025).

Prerequisites for Post-Training A critical open question in the field is identifying the "readiness" of a pretrained model for post-training. Recent studies investigate the interplay between pretraining and post-training efficacy, arguing that models require a sufficient level of capability before they can benefit from RL (Chen et al., 2025; Foster et al., 2025). Zhang et al. (2025) suggests it may be necessary for the base model to exhibit basic skills to incentivize contextual generalization. Furthermore, they analyze the difficulty of RL data in relation to the success of RL. Our analysis questions this rigidity, investigating whether RL might be effective earlier than previously thought. Other works that study this question include Guo et al. (2025); Zhou et al. (2023).

Integrating RL into pretraining There is a growing need for data that can be used with RL training. Various works attempt to overcome this challenge by using pretraining data, which is more general and abundant, with RL. Li et al. (2025) uses a next-sentence reasoning objective and scores for semantic consistency between the generated sentence and the reference sentence from the training corpora. Hatamizadeh et al. (2025); Dong et al. (2025); Xing et al. (2025) use RL with pretraining data by allowing the model to generate a chain of thought prior to predicting the next token. These methods rely on having a pretrained model which has been trained only via standard next-token prediction. In contrast, our work examines the fundamental questions regarding *when and how we expect an RL training objective* to improve downstream performance. In contrast, our work is generally focused on understanding the relationship between the current stage of training and the training objective.

6 Discussion & Future directions

In this paper, we examine the utility of using an RL training objective in the early stages of LLM training. To that end, we perform experiments starting from pretraining checkpoints where compare the effects of RL, SFT, and SFT followed by RL. Relatively early in training, we see that RL is an effective training objective in improving accuracy on reasoning benchmarks across `pass@k` evaluations. Interestingly, we find that later checkpoints combined with RL are even able to match or exceed the performance of SFT + RL. In regimes in which the base model may be unable to solve difficult problems from the RL dataset, we explore the knob of increasing the number of generations specifically for the sake of GRPO. Our findings suggest that increasing the number of generations per prompts is an effective strategy when training with RL on a difficult dataset. We see many future directions for this work.

Incorporating RL into pretraining Our analysis suggests that RL can be an effective training objective in the early stages of LLM training. This brings into question whether incorporating RL training objectives *during* pretraining may also be effective. While many recent works have been exploring RL-pretraining (Sec. 5), there are many open questions regarding what can be the best pretraining objective. In particular, one potential future direction is to explore the potential of a pretraining stage which leverages the benefits of RL.

Pretraining data Our work shows that, when we pretrain models with a training set containing a lot of math, then RL on math tasks quickly becomes effective. However, this brings up questions regarding *which* data we should use in pretraining. For example, if we combine the NTP objective with the RL objective in pretraining, perhaps a different data mixture is optimal than those commonly used. While the common paradigm is to pretrain with general data, perhaps this might change if we also change the pretraining objective.

REFERENCES

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Fan Chen, Audrey Huang, Noah Golowich, Sadhika Malladi, Adam Block, Jordan T Ash, Akshay Krishnamurthy, and Dylan J Foster. The coverage principle: How pre-training enables post-training. *arXiv preprint arXiv:2510.15020*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mo Bavarian, Clemens Winter, Phil Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, Suchir Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021. URL https://api.semanticscholar.org/CorpusID:235755472.

Zhoujun* Cheng, Yutao* Xie, Yuxiao* Qu, Amrith* Setlur, Shibo Hao, Varad Pimpalkhute, Tongtong Liang, Feng Yao, Hector Liu, Eric Xing, Virginia Smith, Ruslan Salakhutdinov, Zhiting Hu, Taylor Killian, and Aviral Kumar. Isocompute playbook: Optimally scaling sampling compute for rl training of llms. urlhttps://compute-optimal-rl-llm-scaling.github.io/, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe rlhf: Safe reinforcement learning from human feedback. *arXiv preprint arXiv:2310.12773*, 2023.

Qingxiu Dong, Li Dong, Yao Tang, Tianzhu Ye, Yutao Sun, Zhifang Sui, and Furu Wei. Reinforcement pre-training. *arXiv preprint arXiv:2506.08007*, 2025.

Dylan J Foster, Zakaria Mhammedi, and Dhruv Rohatgi. Is a good foundation necessary for efficient reinforcement learning? the computational role of the base model in exploration. *arXiv preprint arXiv:2503.07453*, 2025.

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

Ali Hatamizadeh, Syeda Nahida Akter, Shrimai Prabhumoye, Jan Kautz, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Yejin Choi. Rlp: Reinforcement as a pretraining objective. *arXiv preprint arXiv:2510.01265*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.

Siheng Li, Kejiao Li, Zenan Xu, Guanhua Huang, Evander Yang, Kun Li, Haoyuan Wu, Jiajia Wu, Zihao Zheng, Chenchen Zhang, et al. Reinforcement learning on pre-training data. *arXiv preprint arXiv:2509.19249*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

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.

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

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.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *Advances in Neural Information Processing Systems*, 37:34737–34774, 2024.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why rlvr may or may not escape its origin. *arXiv preprint arXiv:2507.14843*, 2025.

Xingrun Xing, Zhiyuan Fan, Jie Lou, Guoqi Li, Jiajun Zhang, and Debing Zhang. Pretrainzero: Reinforcement active pretraining. *arXiv preprint arXiv:2512.03442*, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Charlie Zhang, Graham Neubig, and Xiang Yue. On the interplay of pre-training, mid-training, and rl on reasoning language models. *arXiv preprint arXiv:2512.07783*, 2025.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, et al. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*, 2023.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36: 55006–55021, 2023.

A Appendix

A.1 RL training dynamics

In Fig. 6, we show that for all $\mathcal{M}_t^{\text{RL}}$ (across all pretraining checkpoints $\mathcal{M}_t$), the RL training reward, validation reward (computed on a manually split subset of OpenMathInstruct), and GSM8K reward have converged. For earlier checkpoints that exhibit seed brittleness (Sec. A.2), we report the favorable seed here. See App. A.2 for examples of favorable and unfavorable seeds.

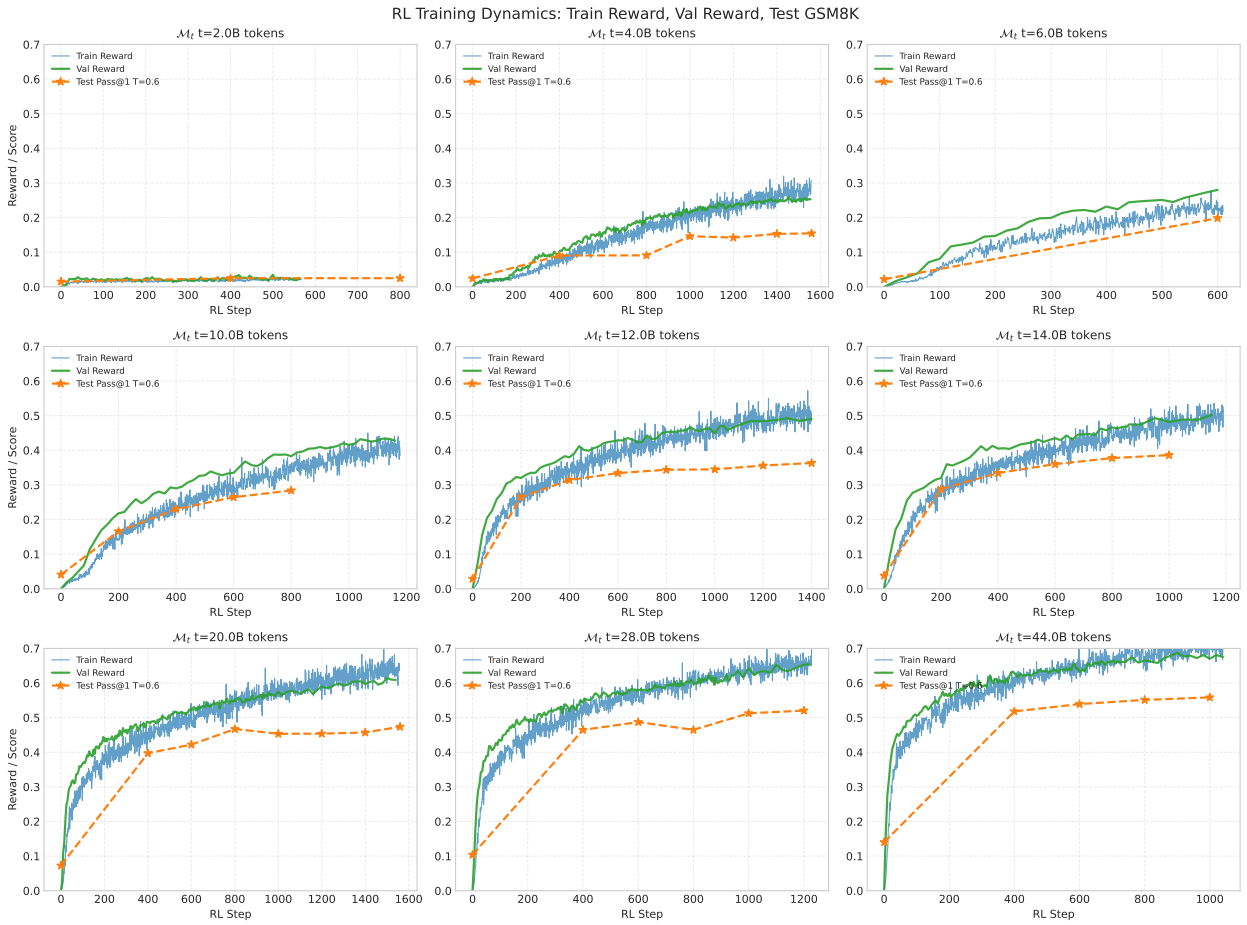

Figure 6: **RL training convergence across pretraining checkpoints.** We show training reward, validation reward, and GSM8K reward curves during RL training for $\mathcal{M}_t^{\text{RL}}$ across different pretraining checkpoints t. All three reward metrics converge by the end of training. For checkpoints that exhibit seed brittleness ($t < 10$B), we report the favorable seed; see App. A.2 for seed comparisons.

A.2 Seed dependency

In Fig. 7 (*right*), we show an example of a favorable seed (*pink*) and a unfavorable seed (*blue*). The favorable seed shows improvement in both `pass@1` and `pass@32`, while the unfavorable seed has no effect on `pass@1`, in fact decreasing `pass@32` performance compared to the pretraining checkpoint. From the training reward curves (Fig. 7), we observe that the training reward for both seeds are *identical*, revealing a disconnect between training and actual reasoning capability. This demonstrates the brittleness of early RL training: on-policy RL learns from its own generations, which may be low-quality reasoning traces that lead the model to learn superficial patterns instead of true reasoning. In contrast, the SFT baselines train on curated, ground-truth reasoning traces and do not exhibit this instability. This brittleness resolves after

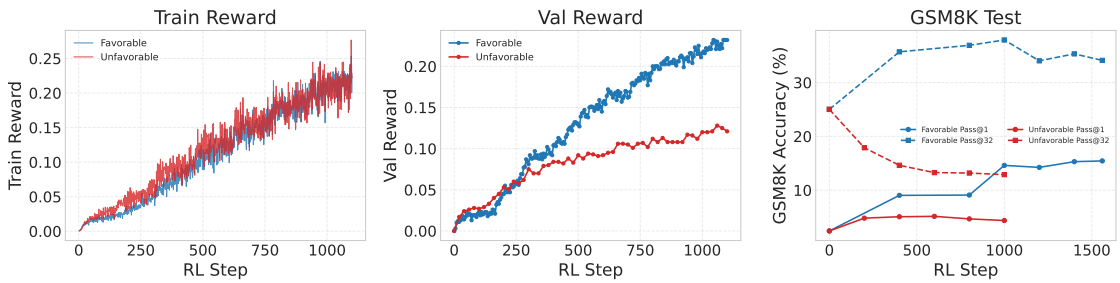

Figure 7: **RL seed brittleness for early pretraining checkpoints.** We compare a favorable seed (*blue*) and an unfavorable seed (*red*) for $\mathcal{M}_t^{\text{RL}}$ at $t = 4$B tokens. *Left*: Training reward curves are nearly identical for both seeds, showing no indication of the divergent outcomes. *Middle*: Validation reward begins to diverge, with the unfavorable seed showing worse performance. *Right*: GSM8K evaluation shows the favorable seed improves both `pass@1` and `pass@32`, while the unfavorable seed shows minimal improvement in `pass@1` and decreased `pass@32` performance.

$t = 10$B tokens, when the model has developed sufficient reasoning foundation to generate higher-quality on-policy data.

A.3 SFT DYNAMICS

In Fig. 8, we experiment with different numbers of SFT epochs to train $\mathcal{M}_t^{\text{SFT}}$ and confirm that 5 epochs leads to convergence in the model's performance.

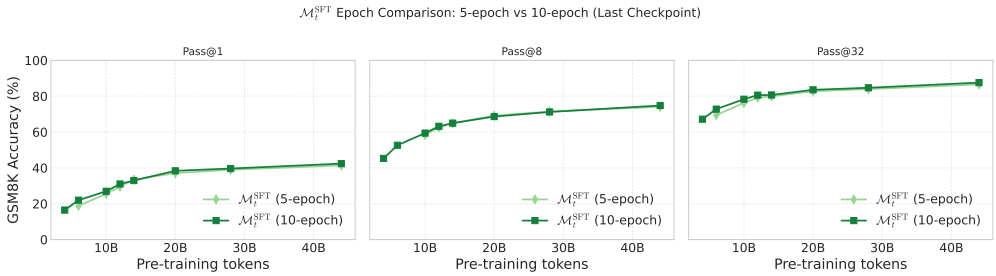

Figure 8: **SFT training convergence.** We evaluate $\mathcal{M}_t^{\text{SFT}}$ on GSM8K after training for different numbers of epochs on OpenMathInstruct. Performance converges after 5 epochs, which we use as the standard training protocol for all $\mathcal{M}_t^{\text{SFT}}$ baselines.

A.4 EVALUATING PRETRAINING CHECKPOINTS

In Fig. 9, we experiment with different numbers of in-context examples (n-shot) to evaluate the reasoning capabilities of pretraining checkpoints $\mathcal{M}_t$. We confirm that by using 8-shot prompting, the base model achieves the best performance on both MATH and GSM8K.

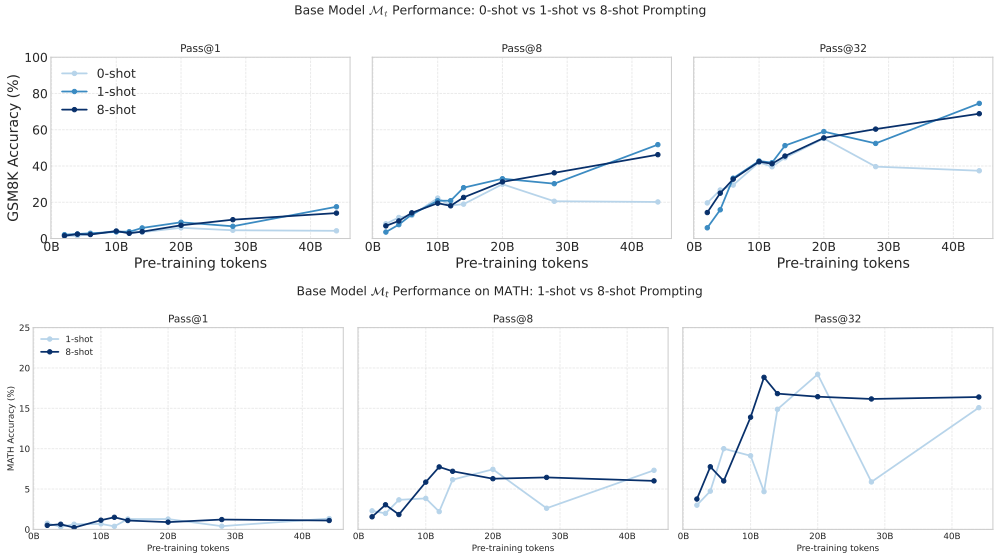

Figure 9: **Impact of n-shot prompting on pretraining checkpoint evaluation.** We evaluate pretraining checkpoints $\mathcal{M}_t$ with varying numbers of in-context examples on GSM8K (*top*) and MATH (*bottom*). Performance peaks at 8-shot prompting for both benchmarks.