RL Excursions During Pre-Training: Re-Examining Policy Optimization for LLM Training

Rachit Bansal[1,2,*] **Clara Mohri**[1,2,*] **Tian Qin**[1,*] **David Alvarez-Melis**[1,2,†] **Sham Kakade**[1,2,†]
[1]Harvard University [2]Kempner Institute
{rachitbansal,cmohri,tqin}@g.harvard.edu

Abstract

The standard LLM training pipeline applies reinforcement learning (RL) only after pre-training and supervised fine-tuning (SFT). We question this status quo by training a LLM from scratch and applying RL, SFT, and SFT followed by RL directly to intermediate pre-training checkpoints. We find that RL is effective very early, and often matches the full SFT→RL pipeline early as well. Through experiments on harder problems, we find that targeted pre-training data composition is a strong lever for RL effectiveness, even more so than model scale. Beyond reasoning accuracy, applying RL directly to base checkpoints expands the model's distribution; the sharpening effect reported in recent work arises only when RL follows SFT. The general capabilities of the model remain essentially unchanged by RL, while they degrade following SFT. Finally, we merge RL and SFT objectives by *parallel averaging*, which outperforms all other training methods discussed, across metrics, while preserving general capabilities. Together, these results suggest that LLM training might benefit from an expanded use of RL.

1 Introduction

Until recently, the training recipe for Large Language Models (LLM) exclusively used the next-token prediction (NTP) objective via cross-entropy loss. However, with the advent of RL for language models (Ouyang et al., 2022; Shao et al., 2024), a newer advancement is the now-standard post-training phase which sequentially employs supervised finetuning (SFT) followed by RL. The NTP objective for pre-training and SFT is typically used over a static, external dataset, i.e., an *off-policy* regime. Instead, for the RL objective, the model learns from its own *on-policy* generations.

Under this standard training regime, RL training only occurs after a substantial amount of NTP training. It is unclear whether this is fundamentally necessary for RL training or simply a design choice (Foster et al., 2025). There has also been growing interest in changing this standard and expanding the use of RL for pretraining (Hatamizadeh et al., 2026; Li et al., 2025; Xing et al., 2025). In this work we attempt to answer a more fundamental question:

When and how should an RL objective be used in LLM training?

While it has been widely observed that post-training dramatically improves the reasoning of the model, RL's influence on model capabilities has been the subject of recent debate. For example, a growing body of work argues that RL primarily sharpens the model's existing output distribution (Yue et al., 2025; Wu et al., 2025; Karan & Du, 2025; Qin et al., 2025). It is unclear whether these findings are inherent to the RL objective or an artifact of the standard training regime. By studying

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

*Equal contribution. Order decided by a dice roll.
†Equal advising.

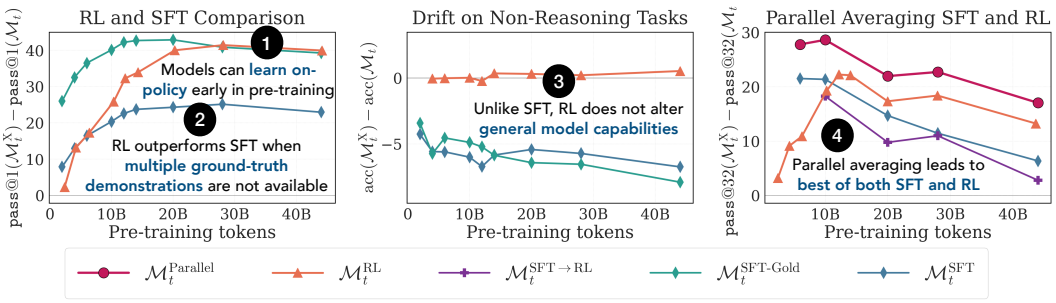


Figure 1: **Overview.** We compare several post-training recipes applied to intermediate pre-training checkpoints $\mathcal{M}_t$: direct RL ($\mathcal{M}_t^{\text{RL}}$), SFT with one solution per question ($\mathcal{M}_t^{\text{SFT}}$), SFT with multiple solutions ($\mathcal{M}_t^{\text{SFT-Gold}}$), the standard pipeline of RL after SFT ($\mathcal{M}_t^{\text{SFT}\to\text{RL}}$), and parallel averaging of RL and SFT gradients ($\mathcal{M}_t^{\text{Parallel}}$). ❶ RL improves both `pass@1` and `pass@32` on checkpoints trained for as low as 4B pretraining tokens (§3.1). ❷ RL is the more effective post-training objective when ground-truth demonstrations are scarce: $\mathcal{M}_t^{\text{RL}}$ substantially outperforms $\mathcal{M}_t^{\text{SFT}}$ on `pass@1` but matches $\mathcal{M}_t^{\text{SFT-Gold}}$ (§3.2). ❸ SFT degrades general, non-reasoning benchmarks, whereas RL leaves these capabilities largely unchanged (§4.2). ❹ Parallel averaging of RL and SFT gradients combines their strengths: $\mathcal{M}_t^{\text{Parallel}}$ attains the strongest `pass@32` for every pre-training checkpoint (§5), consistently better than the standard SFT→RL pipeline ($\mathcal{M}_t^{\text{SFT}\to\text{RL}}$).

various training objectives comprehensively across stages of pre-training, we shed light on a second fundamental question:

What is the influence of RL on model capabilities?

To answer these questions, we perform a large-scale rigorous study of on-policy learning for LLM training. We pretrain an LLM from scratch on a high-quality, reasoning-heavy corpus, saving checkpoints throughout the process. For each base model checkpoint, we perform various different training runs: (**Direct RL**) RL on the base model checkpoint; (**SFT**) SFT using a single ground-truth demonstration per example; (**SFT-Gold**) SFT using multiple ground-truth demonstrations per example; and (**SFT→RL**) RL on top of the SFT models, for both SFT and SFT-Gold, representing the standard LLM training pipeline.

We present comprehensive findings that answer fundamental questions about RL for LLM training:

When does RL work? (§3) We find that RL is effective surprisingly early in pretraining. Training with direct RL on checkpoints that have seen as few as 4B tokens significantly improves performance on both GSM8K and MATH, with gains often comparable to the standard SFT→RL pipeline (§3.1). Moreover, we find that RL is significantly more effective than SFT when we have limited target demonstrations (SFT) (§3.2). The effectiveness of RL varies with task difficulty: RL gains are weaker on harder MATH-style problems. In such cases, we find that adding targeted data to the pretraining corpus is effective and a better strategy than scaling model size (§3.3).

What does RL do? (§4) Contrary to recent claims that RL primarily sharpens the output distribution (Yue et al., 2025; Wu et al., 2025), we find that RL applied directly to base checkpoints *expands* the distribution: `pass@1` *and* `pass@k` both improve substantially (§4.1). The sharpening effect we do reproduce arises only when RL is applied following SFT. This suggests that SFT, rather than RL itself, is what constrains exploration. Further, we find that SFT consistently degrades general (non-reasoning) capabilities, while RL leaves these capabilities unchanged (§4.2).

How should RL be used? (§5) Finally, we investigate whether interleaving SFT and RL gradients within a single training step can capture the complementary strengths of both objectives. We propose a parallel-averaging update that combines updates from SFT and direct RL. We find that this simple objective yields better `pass@32` than all other recipes that use a single demonstration per problem, including SFT→RL, indicating that using RL and NTP objectives simultaneously can be beneficial.

Overall, our findings make headway in understanding RL in contrast to other training objectives. Through our controlled experiments across different stages of pre-training, we find that a lot of assumptions about the RL objective are artifacts of the current training regime. Our results indicate that isolat-

ing the objectives from the setting reveals surprising aspects about RL as an objective for LLM training. We focus our experiments on math reasoning capabilities to maintain a controlled training and evaluation environment. We view our results as evidence that introducing RL earlier and more centrally in the LLM training pipeline is both feasible and, in several respects, preferable to the current standard.

2 Methodology and Experimental Design

To answer foundational questions around the RL objective for LLM training, as stated above, beyond the standard training pipeline, we first establish a controlled experimental environment. Our setup centers on a custom-trained 1B model, allowing for precise control over data exposure. In this section, we detail pre-training checkpoints, define three post-training pipelines, and describe data and evaluation. While we explore RL as a general objective, we focus our implementation on Reinforcement Learning via Verifiable Rewards (RLVR) using the GRPO algorithm (Shao et al., 2024).

2.1 Pre-training checkpoints

Base model and data. We pre-train a 1B parameter model based on OLMo2's (OLMo Team et al., 2025b) architecture and training infrastructure. We perform our pre-training from scratch using high-quality data based on a high-quality subset of OLMo2's pre-training mix, DOLMino (OLMo Team et al., 2025b)[1]. 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., 2022)), high-quality math data (20%), and other reasoning or code data such as StackExchange (2%) and STEM papers (5%).

Pre-training details. We pre-train our 1B parameter model on 50B tokens ($\sim 2.5\times$ Chinchilla optimal tokens). 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. For experiments in Section 3.3, we perform two additional pre-training experiments. In the first, we keep the model architecture and size fixed, but add an additional 10B tokens from the DOLMino-3 mixture (OLMo Team et al., 2025a)[2] throughout training. In the second, we pre-train using the same 50B tokens but scale the model size to 4B parameters.

2.2 Training Pipelines

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

- **Direct RL** ($\mathcal{M}_t^{\mathbf{RL}}$) We start with $\mathcal{M}_t$ and train with the RL objective.
- **SFT only** ($\mathcal{M}_t^{\mathbf{SFT}}$)**:** We start with $\mathcal{M}_t$ and perform SFT with ground-truth solutions.
- **Standard pipeline** ($\mathcal{M}_t^{\mathbf{SFT}\to\mathbf{RL}}$)**:** We train $\mathcal{M}_t^{\text{SFT}}$ with RL on the same set of questions.

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. By comparing $\mathcal{M}_t^{\text{RL}}$ with $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$, we isolate if RL alone can provide a superior training signal than the standard pipeline.

In this work, we are interested in understanding, if 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 B.3 and Appendix B.5.

2.3 Data and Evaluation

Training data. We use OpenMathInstruct (Toshniwal et al., 2024), which consists of math problems paired with multiple ground-truth demonstrations per problem. For SFT, by default, we randomly pick a single solution per prompt for our training (**SFT**) since that is a more realistic SFT setting as

[1] allenai/dolmino-mix-1124
[2] allenai/dolma3_dolmino_mix-100B-1125

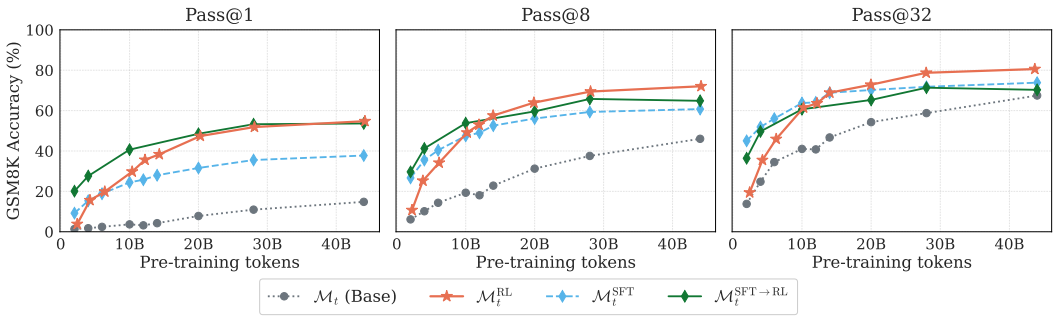


Figure 2: **RL is effective early in pre-training.** GSM8K `pass@k` for $\mathcal{M}_t$, $\mathcal{M}_t^{\text{SFT}}$, $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$, and $\mathcal{M}_t^{\text{RL}}$ across pre-training tokens t, with all SFT baselines trained on the SFT set (one ground-truth solution per problem). $\mathcal{M}_t^{\text{RL}}$ improves over $\mathcal{M}_t$ from as few as 4B tokens. By 10B tokens, $\mathcal{M}_t^{\text{RL}}$ matches the standard $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$ pipeline, and outperforms $\mathcal{M}_t^{\text{SFT}}$ alone.

obtaining multiple ground-truth reasoning traces for each problem is typically infeasible. However, we also consider training with all solutions (**SFT-Gold**) for completeness. For RL, we only consider the final answer for each problem and define a binary reward based on whether the model generation reaches the same final answer.

Difficulty splits. OpenMathInstruct contains two categories of questions: a majority inspired by the MATH dataset (Hendrycks et al., 2021) (competition-level) and a minority inspired by GSM8K (Cobbe et al., 2021) (grade-school level). We consider two experimental settings to probe different aspects of RL training: training with the full OpenMathInstruct and training with the GSM8K-inspired subset of OpenMathInstruct. On the GSM8K subset, the base pre-training checkpoints already achieve non-trivial performance. 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.

Evaluation. We evaluate on GSM8K and MATH respectively, reporting `pass@k` (Chen et al., 2021), which estimates the probability of obtaining at least one correct response when k responses are generated, for $k = 1, 8, 32$ and at temperature $T = 0.6$.

3 RL is Effective Early in Pre-Training

In this section, we study the effect of RL at different stages of pre-training and contrast with training objectives. On GSM8K, for our 1B parameter model, we find that RL is effective from as early as 4B pre-training tokens, and often matches the full SFT→RL pipeline (§3.1). RL is also the more effective objective when ground-truth demonstrations are scarce (§3.2). On harder problems, pre-training data composition is a stronger lever for RL effectiveness than model scale (§3.3). Finally, we identify the base model `pass@k` on the test set as a lightweight diagnostic for whether RL will succeed (§3.4).

3.1 RLVR competes with the standard pipeline on GSM8K

In Figure 2, 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 pre-training steps t on GSM8K, using the GSM8K subset of OpenMathInstruct for post-training. We evaluate base checkpoints $\mathcal{M}_t$ with 8-shot prompting, as they cannot reliably follow question-answering instructions[3]. All post-trained models use 0-shot evaluation, as RL includes a formatting reward and SFT data is formatted accordingly.

We observe that, as early as $t = 4$B pre-training 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*

[3]In Appendix B.6, we ablate the number of in-context examples and confirm that 8-shot yields the best performance for $\mathcal{M}_t$.

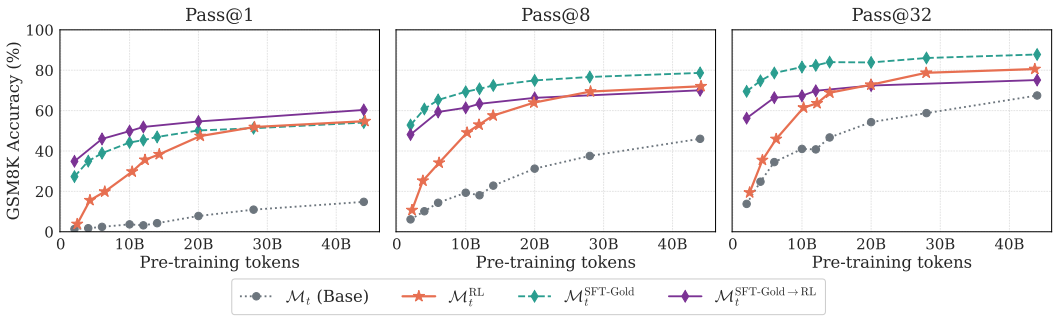


Figure 3: **Diverse SFT data shifts the balance toward SFT-Gold.** In contrast with Figure 2, SFT baselines are trained on SFT-Gold (all ∼23 ground-truth solutions per problem). With access to many ground-truth solutions, $\mathcal{M}_t^{\text{SFT-Gold}}$ alone surpasses $\mathcal{M}_t^{\text{RL}}$ on `pass@8` and `pass@32`, while $\mathcal{M}_t^{\text{SFT-Gold}\to\text{RL}}$ remains best on `pass@1`. $\mathcal{M}_t^{\text{SFT-Gold}}$'s advantage requires multiple high-quality solutions per problem, which is rarely realistic in practice.

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 §4.1.

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, demonstrating that RL can match supervised learning without training on ground-truth reasoning traces.

For some early $\mathcal{M}_t$ model checkpoints between $t = 4$B and $t = 10$B pre-training tokens, we observe brittleness across seeds: RL performance on some seeds fails to improve. It is likely that the model sometimes falls into a distinct failure mode for early pre-training checkpoints (Appendix B.4). Above, we report the RL runs with non-trivial performance.

3.2 RL outperforms when SFT data is scarce

OpenMathInstruct contains an average of 23 ground-truth completions per problem. Our main results use the one randomly chosen completion per problem, which we consider the more realistic SFT setting. We also consider a setting that uses the full OpenMathInstruct dataset for SFT training (i.e., multiple ground-truth completions per problem). We refer to this setting as SFT-Gold since obtaining multiple high-quality solutions per problem typically requires expensive human supervision or generation from frontier models, making it an ideal setting which is impractical for many domains. Figure 2 shows that $\mathcal{M}_t^{\text{RL}}$ outperforms $\mathcal{M}_t^{\text{SFT}}$ on `pass@1` and is competitive on `pass@8,32`. With SFT-Gold, the story changes (Figure 3): SFT-Gold→RL performs best on `pass@1`, but SFT-Gold alone surpasses both RL and SFT→RL on `pass@8,32`. This suggests that access to diverse ground-truth reasoning traces can provide coverage benefits that on-policy exploration does not.

3.3 Targeted pre-training data is more essential than model size for RL

In Figure 9, we train on the MATH-like subset of OpenMathInstruct and evaluate `pass@k` accuracy on MATH. Unlike the GSM8K setting in Figure 2, directly applying RL to pre-training checkpoints is less effective with respect to SFT with SFT and SFT-Gold on this harder benchmark. We hypothesize that the *efficacy* of RL training from pre-training checkpoints has limitations, potentially related to the difficulty of the task at hand. Therefore, we study two natural interventions: scaling N, the model size, and scaling D, the amount of pre-training data, especially task-relevant math data.

To test the effect of scaling N, we pre-train a 4B model from scratch using the same 50B-token mix and training recipe as the original 1B model. As expected, the 4B checkpoints achieve higher base MATH accuracy, and direct RL on these checkpoints also yields higher absolute performance than RL on the 1B checkpoints at matched pre-training steps. However, when measuring the gain from RL relative to each checkpoint's own base performance, the 4B model does *not* obtain larger

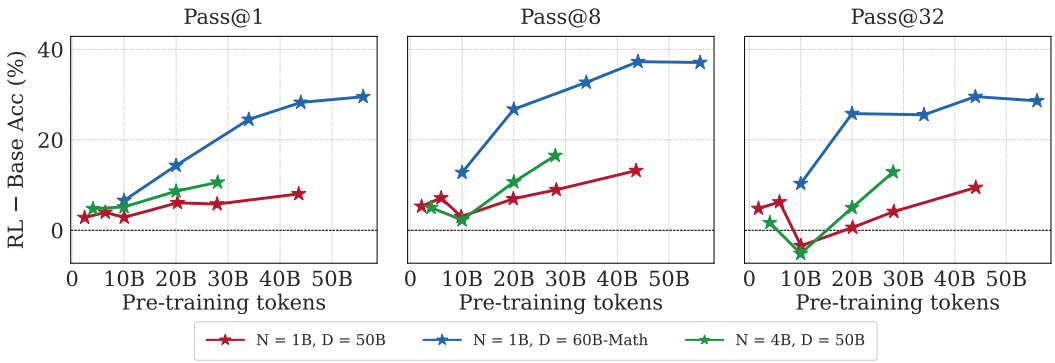


Figure 4: **Targeted pre-training data beats model scale for RL.** Improvement on MATH from RL over the base model, across pre-training configurations: (i) *1B-50B*, original pre-trained model; (ii) *Scaling D (1B-60B)*, 1B model pre-trained from scratch with an additional 10B math-heavy tokens mixed in; (iii) *Scaling N (4B-50B)*, 4B model trained on same 50B-token mix as original 1B model. Adding task-relevant pre-training data (Scaling D) yields substantially larger RL gains on MATH.

improvements. As shown in Figure 4, scaling model size improves the base model, but does not substantially improve the effectiveness of RL itself.

We then test the effect of scaling D while keeping the model size fixed at 1B. We pre-train from scratch with an additional 10B math- and reasoning-heavy tokens from the Dolma 3 Dolmino Mix (OLMo Team et al., 2025a), described in Appendix B.2. In this setting, direct RL on the resulting checkpoints matches the SFT baseline on MATH (Figure 10) and recovers the qualitative behavior observed on GSM8K (Figure 2). Moreover, Figure 4 shows that the RL gain over the base model is substantially larger than in either the original 1B setting or the 4B scaling-N setting.

Overall, targeted pre-training data is the more effective intervention: adding math-specific data during pre-training substantially improves the gains achievable by direct RL, whereas increasing model size primarily improves the base checkpoint. In Appendix D, we further study scaling G, the number of RL rollouts, and find that increasing G does *not* change the final outcome of direct RL.

3.4 Base model performance is predictive of RL effectiveness

Given that we train with RL on early pre-training checkpoints, a natural question is, *how can we predict if RL training will be effective?* In Figure 5, we compare the base model's `pass@k` accuracy on the test set with that of the model after RL. For MATH, we also report the comparison for the two additional pre-training regimes discussed in §3.3. We observe a generally monotonically increasing relationship in which increasing `pass@k` for the base model corresponds to increased `pass@k` for the model after RL. In practice, this suggests that a model's `pass@k` accuracy on the test set might serve as a lightweight metric for whether RL training will yield downstream gains.

This relationship is consistent with successful trajectories already being accessible, albeit rarely, from the base model and providing a signal for RL. `pass@k` alone cannot distinguish this explanation from the acquisition of new reasoning strategies.

4 The Effects of RL Beyond Downstream Accuracy

In this section, we examine the effects of RL on the trained model beyond the downstream accuracy. First, we evaluate whether RL unlocks new reasoning capabilities or merely *sharpens* the base model's existing output distribution (§4.1). Then, we study whether RL alters general capabilities inherited from pretraining (§4.2). We find that the answer to both depends on the training pipeline.

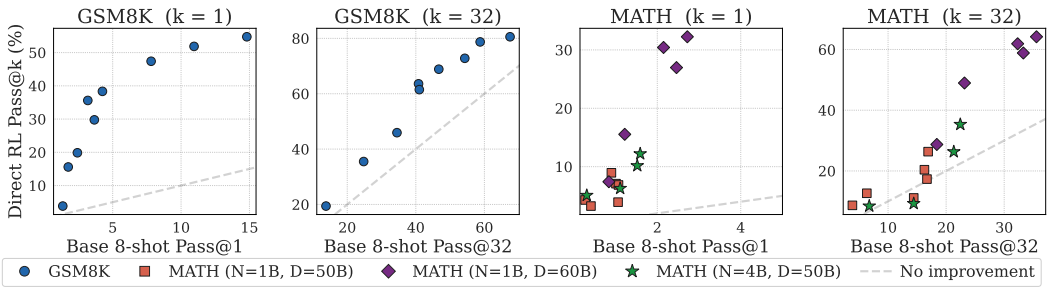


Figure 5: **Base `pass@k` on test set predicts RL effectiveness.** Base model 8-shot `pass@k` on the test set (x-axis) vs. after RL (y-axis), for GSM8K (*left*) and MATH (*right*). `pass@k` accuracy on the test set might serve as a lightweight metric for whether RL training will yield downstream gains.

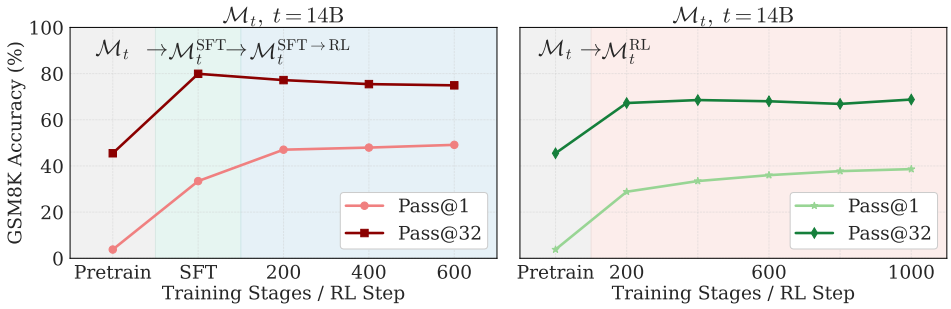


Figure 6: **Direct RL expands while SFT→RL sharpens.** GSM8K `pass@1` and `pass@32` tracked across training stages on the same pretraining checkpoint $\mathcal{M}_t$. *Left*: under the standard $\mathcal{M}_t \to \mathcal{M}_t^{\text{SFT}} \to \mathcal{M}_t^{\text{SFT}\to\text{RL}}$ pipeline, `pass@1` continues to improve during RL but `pass@32` *decreases*, reproducing the sharpening effect reported in prior work. *Right*: applying RL directly to $\mathcal{M}_t$ improves both `pass@1` and `pass@32`, expanding the model's distribution rather than merely sharpening it.

4.1 Early stage RL can expand the model's distribution

We refer to sharpening (Wu et al., 2025; Yue et al., 2025) as the phenomenon in which training improves `pass@1` accuracy but has little or even negative effect on `pass@k` accuracy for larger k. In contrast, we define expansion as the setting in which training increases `pass@k` performance for large k.

Many recent works have claimed that RLVR largely *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 outcomes depending on the training pipeline. First, when we apply the standard pipeline on 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 6 (*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 to $\mathcal{M}_t^{\text{SFT}} \to$ RL. On the other hand, the SFT stage yields a significant gain in `pass@32`, but the subsequent RL stage slightly degrades the performance.

We hypothesize that sharpening occurs because, during SFT, the model has already seen ground-truth solutions on the same set of questions, thus RL primarily refines these existing capabilities rather than discovering new reasoning paths. In contrast, by directly training on the RL objective from the same pretraining checkpoint Figure 6 (*right*), RL training improves both `pass@1` and `pass@32` performance, expanding the base model's distribution. These results do not determine whether RL creates new reasoning strategies or makes successful trajectories already latent in the base model easier to sample.

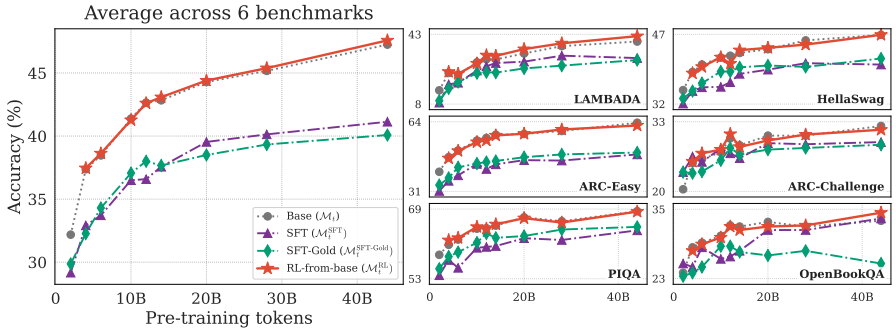


Figure 7: **RL preserves general capabilities while SFT degrades them.** Performance on six general-purpose (non-math) benchmarks for the base model $\mathcal{M}_t$ and three post-trained variants: $\mathcal{M}_t^{\text{RL}}$, $\mathcal{M}_t^{\text{SFT}}$, and $\mathcal{M}_t^{\text{SFT-Gold}}$. Both SFT and SFT-Gold consistently degrade performance by 4–8 pp on average across the benchmarks, while RL leaves these capabilities essentially unchanged.

Algorithm 1 Parallel averaging update

Input: parameters θ; optimizer states $s_{\text{RL}}, s_{\text{SFT}}$; batches $\mathcal{B}_{\text{RL}}, \mathcal{B}_{\text{SFT}}$; learning rates $\eta_{\text{RL}}, \eta_{\text{SFT}}$
// Snapshot current parameters
$\bar{\theta} \leftarrow \theta$
// Compute both objectives at the same snapshot
$g_{\text{RL}} \leftarrow \nabla_\theta \mathcal{L}_{\text{RL}}(\theta; \mathcal{B}_{\text{RL}})\big|_{\theta=\bar{\theta}}$
$g_{\text{SFT}} \leftarrow \nabla_\theta \mathcal{L}_{\text{SFT}}(\theta; \mathcal{B}_{\text{SFT}})\big|_{\theta=\bar{\theta}}$
// Compute optimizer updates from the snapshot
$(\Delta_{\text{RL}}, s_{\text{RL}}) \leftarrow \text{OptUpdate}(g_{\text{RL}}, s_{\text{RL}}, \eta_{\text{RL}})$
$(\Delta_{\text{SFT}}, s_{\text{SFT}}) \leftarrow \text{OptUpdate}(g_{\text{SFT}}, s_{\text{SFT}}, \eta_{\text{SFT}})$
// Average the two gradient updates
$\theta \leftarrow \bar{\theta} + \frac{1}{2}(\Delta_{\text{RL}} + \Delta_{\text{SFT}})$
return θ

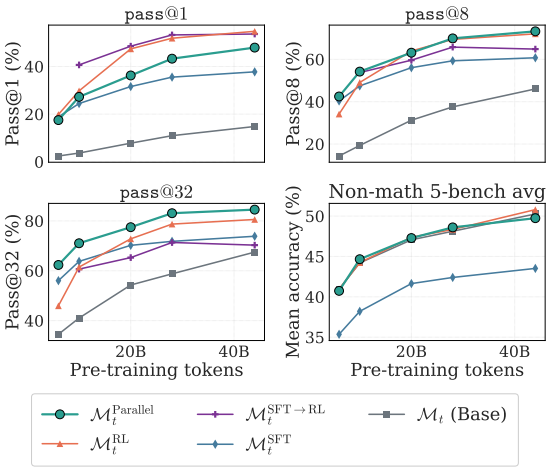


Figure 8: **Parallel averaging combines the strengths of RL and SFT across pre-training.** *(Left)* The parallel-averaging update (Algorithm 1): at each step we take a single optimizer update from each of an RL gradient and an SFT gradient (each with its own optimizer state) and use their average to update the model weights. *(Right)* Parallel-averaging ($\mathcal{M}_t^{\text{Parallel}}$) achieves the strongest `pass@32` across pre-training checkpoints, surpassing the standard pipeline ($\mathcal{M}_t^{\text{SFT}\to\text{RL}}$). Unlike SFT-based regimes, $\mathcal{M}_t^{\text{Parallel}}$ does not regress on non-math benchmarks and retains base model performance.

4.2 RL does not affect general model capabilities

A natural concern with applying RL to intermediate pretraining checkpoints is whether it degrades capabilities outside the training domain. To assess this, we evaluate on several general-purpose benchmarks and report results in Figure 7. We report benchmark results for the base model, the base model after training directly with RL, and training with SFT. For SFT, we report accuracy both for training with SFT and SFT-Gold. Interestingly, we find that RL from the base model has the least effect on general model capability, while SFT training typically degrades the general model capability regardless of using one or many completions per prompt.

5 Parallel RL and SFT

The previous sections expose complementary strengths of the RL and SFT objectives applied directly to pretraining checkpoints. Direct RL ($\mathcal{M}_t^{\text{RL}}$) can develop new reasoning capabilities, expand the model's `pass@k` distribution, and leave general (non-math) capabilities intact (§4). This expansion, however, is only reliable when the underlying pretraining mix and model size yield enough latent

capability to bootstrap from the base model (§3.3). SFT instead provides reliable supervision from ground-truth reasoning traces. However, the efficacy of SFT relies on the diversity of the SFT data, and might have a negative impact on general capabilities. Having studied RL and SFT in isolation, we next consider whether a combined objective might enjoy the benefits of both.

Method. To test whether the two update signals can be combined, we use a simple parallel-averaging experiment (Algorithm 1). At each training step, starting from the same parameter snapshot θ, we run one batch through an RL optimizer, and in parallel, run a separate batch of SFT data through an SFT optimizer. We obtain gradients from the two optimizers and average the gradients to update θ. Critically, the two optimizers maintain independent first- and second-moment estimates, so the adaptive step sizes and preconditioning do not interfere. We refer to the resulting model as $\mathcal{M}_t^{\text{Parallel}}$.

Findings. We report results in Figure 8 (see Figure 16 for the per-checkpoint training trajectories). Across every pre-training checkpoint we evaluated, parallel averaging attains the strongest `pass@32` among recipes that use a single demonstration per problem, surpassing direct RL, SFT, and the standard pipeline. It also preserves the base model's general (non-math) capabilities on par with direct RL, whereas every SFT-based recipe regresses on this axis by 5–8 percentage points. However, we also observe that these strong `pass@k` improvements come with a trade-off of a lower `pass@1` relative to the direct RL and SFT baselines.

The result shows that combining the two updates can retain the pass@32 gains and measured non-math performance observed here. Overall, we view these results as evidence that the RL and SFT signals are complementary rather than merely additive. Our implementation uses equal-weight averaging with no scheduling, leaving room for more deliberate combinations of RL and next-token-prediction objectives which we view as a productive direction for future work.

6 Prior Work

Reinforcement Learning for LLM Reasoning RL has become a standard post-training stage for LLMs (Ouyang et al., 2022; Dai et al., 2024; Jaech et al., 2024), using modern policy-gradient methods (Rafailov et al., 2023; Shao et al., 2024; Yu et al., 2025; Khatri et al., 2025; Mohri et al., 2026) with verifiable rewards (Guo et al., 2025; Zheng et al., 2023). Whether gains via these methods reflect new capabilities or merely a sharpening remains contested (Yue et al., 2025; Wu et al., 2025; Karan & Du, 2025; Cheng et al., 2026; Chu et al., 2025), as does whether RL erodes abilities inherited from pretraining (Shenfeld et al., 2025). Our findings suggest a nuanced view on these questions.

Integrating RL into Pretraining A recent line of work brings RL into pretraining itself, either by scoring next-sentence reasoning against the training corpus (Li et al., 2025) or by inserting chain-of-thought rollouts before each next-token prediction (Hatamizadeh et al., 2026; Dong et al., 2025; Xing et al., 2025). These methods modify the pretraining objective. Our work instead keeps the standard NTP and RL objectives unchanged. We view our results as a precursor: before adding RL *into* pretraining, it is worth knowing how early in pretraining RL on top of NTP already pays off.

Interleaving SFT and RL A growing body of work performs mixed-policy training. Approaches include importance-weighted off-policy expert traces (Yan et al., 2025), alternating SFT and RL passes targeted at unsolved problems (Dong et al., 2026), joint losses with adaptive weighting (Fu et al., 2025; Lv et al., 2025; Zhang et al., 2026), and hybrid trajectories that blend expert prefixes with on-policy continuations (Huang et al., 2026). Limozin et al. (2026) caution that several of these methods were compared against deflated SFT baselines, and that a correctly implemented SFT→RL pipeline can match or exceed them. In our work, we evaluate a simple alternate approach that maintains independent Adam moments for SFT and RL and averages their proposed updates after each step.

Prerequisites for Post-Training A small but growing literature studies what level of pretraining is required before post-training becomes effective. Chen et al. (2025); Foster et al. (2025) argue that the base model must reach a minimum capability for RL to yield gains. Zhang et al. (2025) relates this threshold to the base model's basic skills and to the difficulty of the RL data; see also Guo et al. (2025); Zhou et al. (2023). Our work tests this premise empirically by tracing RL effectiveness across

pretraining tokens, and finds that the threshold is far lower than commonly assumed: RL is effective on GSM8K from as few as 4B tokens, well below the Chinchilla-optimal point.

7 Discussion & Future Directions

In this work, we provide a comprehensive and nuanced picture of the RL objective for LLM training beyond how it is used in the current standard pipeline. We find that RL can be effective starting early in pre-training, well before the Chinchilla-optimal regime, and often matches the full SFT→RL pipeline on GSM8K despite never seeing a ground-truth reasoning trace. Further, the dominant lever for whether early RL succeeds is pre-training data composition, not model scale. Perhaps most strikingly, the two effects most commonly attributed to RL, distribution sharpening and regression on general capabilities, are largely artifacts of a preceding SFT stage rather than of the RL objective itself: applied directly to base checkpoints, RL instead expands the `pass@k` distribution and leaves non-math capabilities essentially intact, unlike SFT that consistently degrades them significantly.

Our results open several exciting research directions. The most consequential is rethinking how data and objectives are coordinated end-to-end: if RL is effective well inside pre-training, and the pre-training mix controls its ceiling, the practical question becomes what pre-training recipes could look like once RL is treated as a first-class training objective rather than a final post-training step. Our parallel-averaging experiment (§5) is an early data point towards that end. Work can be done towards more careful designs, with adaptive weighting, scheduling, or importance sampling on top of independent optimizer states. The finding that base `pass@k` already predicts RL effectiveness (§3.4) further motivates adaptive rollout strategies that concentrate compute on prompts where the base model has non-trivial coverage but has not yet converged. Our experiments are at 1B and 4B parameters and 50–60B tokens; whether the same picture holds at frontier scale is essential future work.

Limitations We discuss a few limitations in our work. First, while our pre-training mix is designed to be reflective of general pre-training settings, it is more math-heavy than typical web-scale corpora. Second, we focus on standard GRPO as a representative RLVR objective and do not study the growing family of variants (e.g., those explicitly targeting entropy preservation or `pass@k` expansion), which may interact differently with the pre-training stage. Given that are main conclusions are demonstrated for GRPO on math tasks, 1B and 4B models, and math-enriched pretraining mixtures, they do not establish the same behavior for other domains, RL algorithms, or frontier-scale models.

8 Acknowledgements

DAM acknowledges support from the Kempner Institute and NSF award no. 2229881.

References

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. URL https://arxiv.org/abs/2510.15020.

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 preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

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. https://compute-optimal-rl-llm-scaling.github.io/, 2026. URL https://compute-optimal-rl-llm-scaling.github.io/.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025. URL https://arxiv.org/abs/2501.17161.

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. URL https://arxiv.org/abs/2110.14168.

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. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.12773.

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

Yihong Dong, Xue Jiang, Yongding Tao, Huanyu Liu, Kechi Zhang, Lili Mou, Rongyu Cao, Yingwei Ma, Jue Chen, Binhua Li, et al. RL-PLUS: Countering capability boundary collapse of LLMs in reinforcement learning with hybrid-policy optimization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. URL https://arxiv.org/abs/2508.00222.

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. URL https://arxiv.org/abs/2503.07453.

Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. SRFT: A single-stage method with supervised and reinforcement fine-tuning for reasoning. *arXiv preprint arXiv:2506.19767*, 2025. URL https://arxiv.org/abs/2506.19767.

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. *Nature*, 645:633–638, 2025. URL https://arxiv.org/abs/2501.12948.

Ali Hatamizadeh, Syeda Nahida Akter, Shrimai Prabhumoye, Jan Kautz, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Yejin Choi. RLP: Reinforcement as a pretraining objective. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2510.01265.

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. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021. URL https://arxiv.org/abs/2103.03874.

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. URL https://arxiv.org/abs/2203.15556.

Zeyu Huang, Tianhao Cheng, Zihan Qiu, Zili Wang, Yinghui Xu, Edoardo M Ponti, and Ivan Titov. Blending supervised and reinforcement fine-tuning with prefix sampling. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2507.01679.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. URL https://arxiv.org/abs/2412.16720.

Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think. *arXiv preprint arXiv:2510.14901*, 2025. URL https://arxiv.org/abs/2510.14901.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S. Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for LLMs, 2025. URL https://arxiv.org/abs/2510.13786.

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. URL https://arxiv.org/abs/2406.11794.

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. URL https://arxiv.org/abs/2509.19249.

Alexis Limozin, Eduard Durech, Torsten Hoefler, Imanol Schlag, and Valentina Pyatkin. Sft-then-rl outperforms mixed-policy methods for llm reasoning. *arXiv preprint arXiv:2604.23747*, 2026.

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

Xingtai Lv, Yuxin Zuo, Youbang Sun, Hongyi Liu, Yuntian Wei, Zhekai Chen, Xuekai Zhu, Kaiyan Zhang, Bingning Wang, Ning Ding, et al. Towards a unified view of large language model post-training. *arXiv preprint arXiv:2509.04419*, 2025. URL https://arxiv.org/abs/2509.04419.

Clara Mohri, Amir Globerson, Haim Kaplan, Tomer Koren, and Yishay Mansour. Cost-aware learning. *arXiv preprint arXiv:2604.28020*, 2026.

OLMo Team, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. OLMo 3. *arXiv preprint arXiv:2512.13961*, 2025a. URL https://arxiv.org/abs/2512.13961.

OLMo Team, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 OLMo 2 furious. In *Conference on Language Modeling (COLM)*, 2025b. URL https://arxiv.org/abs/2501.00656.

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. URL https://arxiv.org/abs/2203.02155.

Tian Qin, Naomi Saphra, and David Alvarez-Melis. Sometimes I am a tree: Data drives unstable hierarchical generalization. *arXiv [cs.LG]*, 2024. URL http://arxiv.org/abs/2412.04619.

Tian Qin, Core Francisco Park, Mujin Kwun, Aaron Walsman, Eran Malach, Nikhil Anand, Hidenori Tanaka, and David Alvarez-Melis. Decomposing elements of problem solving: What "math" does rl teach?, 2025. URL https://arxiv.org/abs/2505.22756.

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. URL https://arxiv.org/abs/2305.18290.

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. URL https://arxiv.org/abs/2402.03300.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. *arXiv preprint arXiv:2509.04259*, 2025. URL https://arxiv.org/abs/2509.04259.

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. URL https://arxiv.org/abs/2402.10176.

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 (ICLR)*, 2022. URL https://arxiv.org/abs/2109.01652.

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. URL https://arxiv.org/abs/2507.14843.

Xingrun Xing, Zhiyuan Fan, Jie Lou, Guoqi Li, Jiajun Zhang, and Debing Zhang. PretrainZero: Reinforcement active pretraining. *arXiv preprint arXiv:2512.03442*, 2025. URL https://arxiv.org/abs/2512.03442.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*, 2025. URL https://arxiv.org/abs/2504.14945.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

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? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2504.13837. Oral.

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. URL https://arxiv.org/abs/2512.07783.

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. On-policy RL meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting, 2026. URL https://arxiv.org/abs/2508.11408.

Rosie Zhao, Tian Qin, David Alvarez-Melis, Sham Kakade, and Naomi Saphra. Random scaling of emergent capabilities, 2026. URL https://arxiv.org/abs/2502.17356.

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. URL https://arxiv.org/abs/2307.04964.

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. URL https://arxiv.org/abs/2305.11206.

Contents

1 Introduction **1**

2 Methodology and Experimental Design **3**
2.1 Pre-training checkpoints . . . 3
2.2 Training Pipelines . . . 3
2.3 Data and Evaluation . . . 3

3 RL is Effective Early in Pre-Training **4**
3.1 RLVR competes with the standard pipeline on GSM8K . . . 4
3.2 RL outperforms when SFT data is scarce . . . 5
3.3 Targeted pre-training data is more essential than model size for RL . . . 5
3.4 Base model performance is predictive of RL effectiveness . . . 6

4 The Effects of RL Beyond Downstream Accuracy **6**
4.1 Early stage RL can expand the model's distribution . . . 7
4.2 RL does not affect general model capabilities . . . 8

5 Parallel RL and SFT **8**

6 Prior Work **9**

7 Discussion & Future Directions **10**

8 Acknowledgements **10**

A Experiment Details **16**
A.1 Resources . . . 16
A.2 Hyperparameters . . . 16

B Additional Results For Section 3 **17**
B.1 MATH performance . . . 17
B.2 Added data for scaling D . . . 17
B.3 RL training dynamics . . . 17
B.4 Seed dependency . . . 18
B.5 SFT dynamics . . . 18
B.6 Evaluating Pretraining Checkpoints . . . 18

C Full Parallel Average Results **19**

D RL Rollouts **19**
D.1 Experimental Setup . . . 19
D.2 Main Results . . . 21

A Experiment Details

In this section, we provide the details necessary to replicate our experiments. For pretraining, we use the Olmo pretraining library, and for RL/ SFT we use the VeRL library.

A.1 Resources

For our experiments, we use a combination of NVIDIA A100 GPUs and NVIDIA H100 GPUs. Pretraining takes several days, GRPO training takes several days, and SFT takes a few hours.

A.2 Hyperparameters

In the following tables, we report hyperparameter choices for GRPO, SFT, and pretraining.

Table 1: GRPO training hyperparameters (OLMo2-1B on GSM8K subset).

Category	**Hyperparameter**	**Value**
Data	Train batch size	512
	Max prompt length	1024
	Max response length	2048
	Rollouts per prompt (n)	32
Optimization	Learning rate	1×10^{-6}
	Optimizer	AdamW
	(β_1, β_2)	$(0.9,\ 0.999)$
	Weight decay	0.01
	Gradient clip	1.0
	Mini-batch size	128
	KL loss coefficient	1×10^{-3}
	KL loss type	low-variance KL
Reward	Advantage estimator	GRPO
	Format score (partial)	0.1
Infrastructure	Total epochs	10
	GPU memory utilization	0.6

Table 2: GRPO training hyperparameters (OLMo2-1B on OpenMathInstruct-2).

Category	**Hyperparameter**	**Value**
Data	Train batch size	512
	Max prompt length	1024
	Max response length	2048
	Rollouts per prompt (n)	32
Optimization	Learning rate	1×10^{-6}
	Optimizer	AdamW
	(β_1, β_2)	$(0.9,\ 0.999)$
	Weight decay	0.01
	Gradient clip	1.0
	KL loss coefficient	1×10^{-3}
	KL loss type	low-variance KL
Reward	Advantage estimator	GRPO
	Format score (partial)	0.1
Infrastructure	Total epochs	10
	GPU memory utilization	0.8

Table 3: SFT training hyperparameters (OLMo2-1B on OpenMathInstruct-2).

Category	Hyperparameter	Value
Data	Train batch size	512
	Max prompt length	2560
	Max response length	1024
	Rollouts per prompt (n)	32
Optimization	Learning rate	4×10^{-5}
	Optimizer	AdamW
	(β_1, β_2)	$(0.9,\ 0.999)$
	Weight decay	0.01
	Gradient clip	1.0
SFT Schedule	Mode	interleaved
	SFT steps per cycle	50000
	RL steps per cycle	0
Infrastructure	Total epochs	5
	GPU memory utilization	0.6

Table 4: Pretraining hyperparameters (OLMo2-1B, 50B tokens).

Category	Hyperparameter	Value
Data	Total training tokens	50B
	Global batch size (sequences)	512
	Gradient accumulation steps	64
Optimization	Learning rate	4×10^{-4}
	Optimizer	AdamW
	(β_1, β_2)	$(0.9,\ 0.95)$
	Weight decay	0.1
	Gradient clip	1.0
LR Schedule	Schedule	cosine with warmup
	Warmup tokens	1B
	Min LR ratio (α_f)	0.1
	Units	tokens
Regularization	Precision	BF16 (AMP)
	Softmax auxiliary loss	✓
	Auxiliary loss multiplier	1×10^{-5}

B Additional Results For Section 3

B.1 MATH performance

See Fig. 9 for MATH performance on original 1B model, Fig. 10 for MATH performance on 1B model trained on 60B tokens and finally, Fig. 11 for MATH performance on 4B model.

B.2 Added data for scaling D

We detail the source of the 10B tokens we add into training for the MATH benchmark.

B.3 RL training dynamics

In Fig. 12, 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

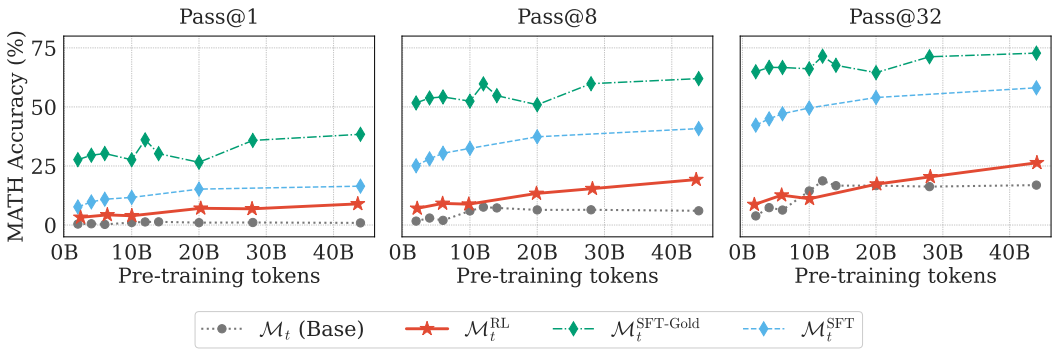


Figure 9: **RL underperforms SFT→RL on harder MATH problems.** MATH `pass@k` for $\mathcal{M}_t$, $\mathcal{M}_t^{\text{SFT}}$, $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$, and $\mathcal{M}_t^{\text{RL}}$ trained on the full OpenMathInstruct, with the base model at $N = 1$B parameters and $D = 50$B pretraining tokens. $\mathcal{M}_t^{\text{RL}}$ still improves over $\mathcal{M}_t$ before Chinchilla-optimal token counts, but a persistent gap to $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$ remains throughout pretraining, indicating that direct RL is insufficient on harder reasoning tasks.

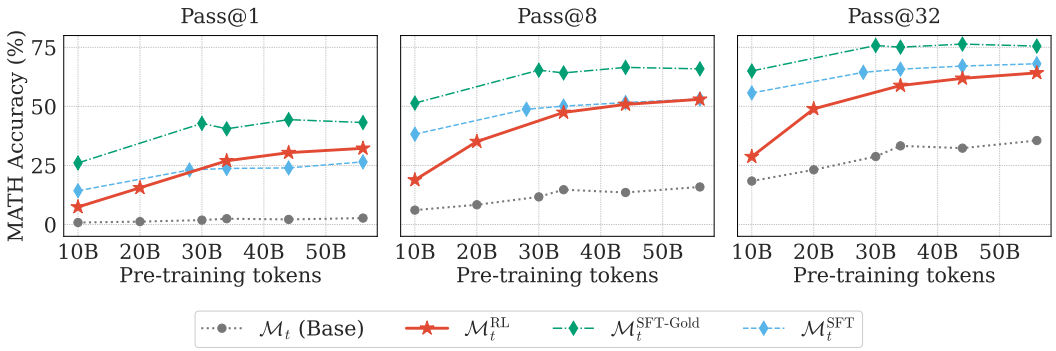


Figure 10: **Adding math pretraining data narrows the MATH gap.** Same setup as Figure 9, but with 10B additional math-heavy tokens mixed into pretraining ($N = 1$B, $D = 60$B). Including task-relevant pretraining data substantially boosts $\mathcal{M}_t^{\text{RL}}$ on MATH and narrows the gap to $\mathcal{M}_t^{\text{SFT}\to\text{RL}}$, supporting pretraining data composition as the binding constraint on early-RL effectiveness.

reward have converged. For earlier checkpoints that exhibit seed brittleness (Sec. B.4), we report the favorable seed here. See App. B.4 for examples of favorable and unfavorable seeds.

B.4 Seed dependency

We visualize the outcomes in Figure 13. Random seed dependency in LLM training has also been observed in Zhao et al. (2026); Qin et al. (2024) as a potential explanation of the emergence phenomenon.

B.5 SFT dynamics

In Fig. 14, 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.

B.6 Evaluating Pretraining Checkpoints

In Fig. 15, 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.

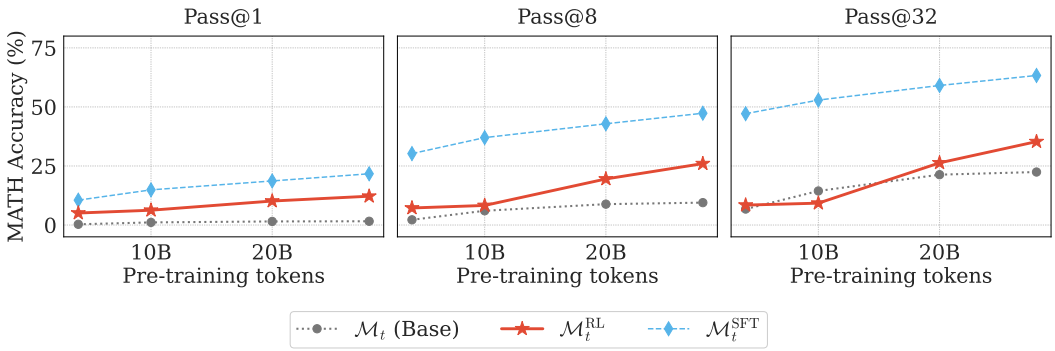


Figure 11: **Scaling parameters does not close the MATH gap.** Same setup as Figure 9, but at $N = 4$B parameters with the same $D = 50$B-token pretraining mix. Increasing model scale improves base-model performance, but does *not* unlock additional RL gains on MATH. In contrast to the data-scaling intervention in Figure 10, the gap to $\mathcal{M}_t^{\mathrm{SFT}\to\mathrm{RL}}$ persists.

Table 5: Composition of the additional math tokens mixed into pretraining for the 1B-60B model (Section 3.3). All sources are drawn from the math subset of the Dolma 3 Dolmino Mix (OLMo Team et al., 2025a).

Source	Description	Tokens
TinyMATH Mind	Conversational solutions to MATH-style problems	898M
TinyMATH PoT	Program-of-thought solutions to MATH-style problems	241M
CraneMath	Swallow Math reproduction	5.62B
MegaMatt	MegaMath-Web-Pro-Max reproduction	1.73B

C Full Parallel Average Results

In Figure 16, we show the full parallel-average training trajectories at each pre-training checkpoint. Figure 8 summarizes these as a single (final-RL-step) point per checkpoint.

D 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.

D.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, §2.3). We focus on only the GSM8K-like subset of OpenMathInstruct (80K examples).

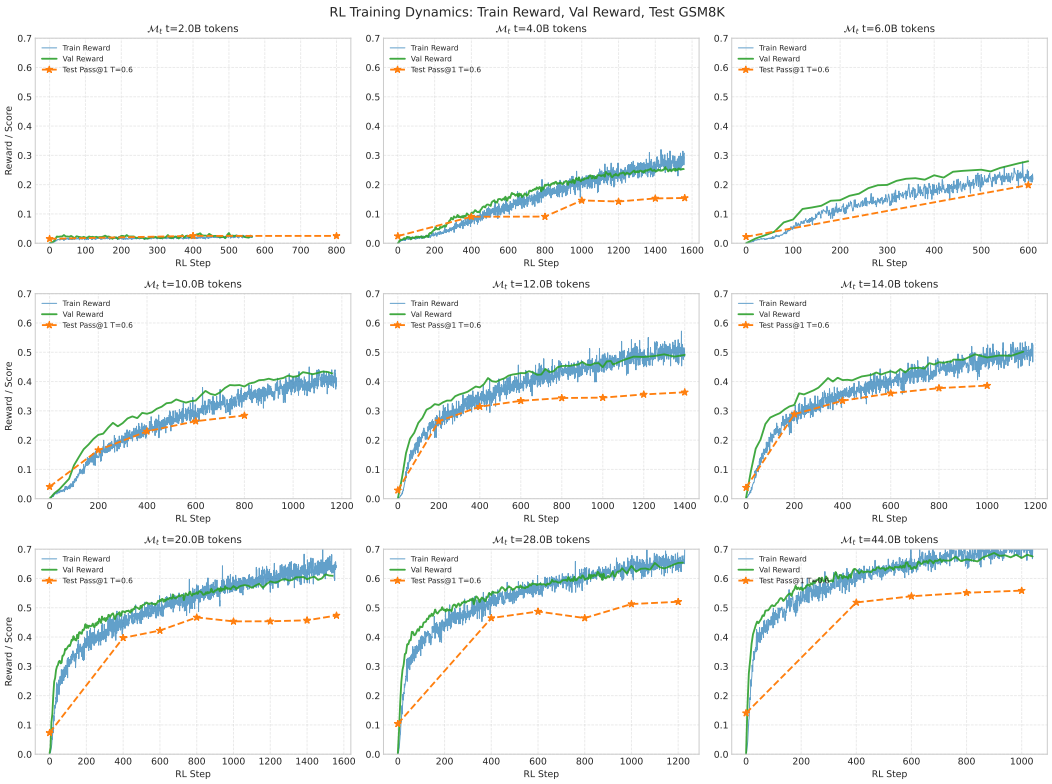


Figure 12: **RL training reaches convergence at all checkpoints.** Training reward, validation reward, and GSM8K test reward during RL training for $\mathcal{M}_t^{\text{RL}}$ across pretraining checkpoints t. All three reward metrics converge by end-of-training, confirming that performance differences between checkpoints are not artifacts of insufficient RL optimization. For checkpoints with seed brittleness ($t < 10$B), we plot the favorable seed; see App. B.4 for seed comparisons.

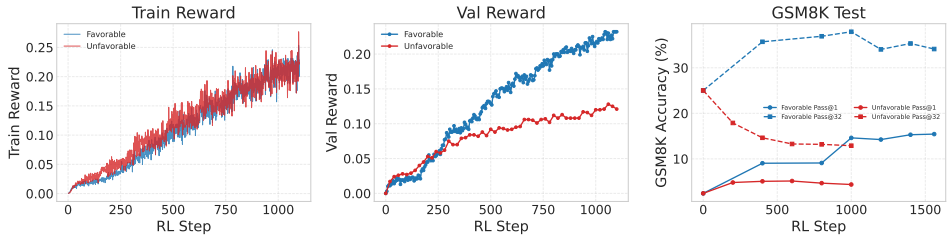


Figure 13: **Training reward hides RL seed brittleness on early checkpoints.** A favorable seed (*blue*) and an unfavorable seed (*red*) for $\mathcal{M}_t^{\text{RL}}$ at $t = 4$B tokens. *Left*: training reward curves are nearly identical between seeds, offering no warning of divergent test outcomes. *Middle*: validation reward begins to diverge mid-training and unfavorable seed only reaches 10% which comes from format reward. *Right*: on GSM8K, the favorable seed gains substantially on both `pass@1` and `pass@32`, while the unfavorable seed shows minimal `pass@1` gain and worsens `pass@32`. This brittleness resolves by $t = 10$B tokens.

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.

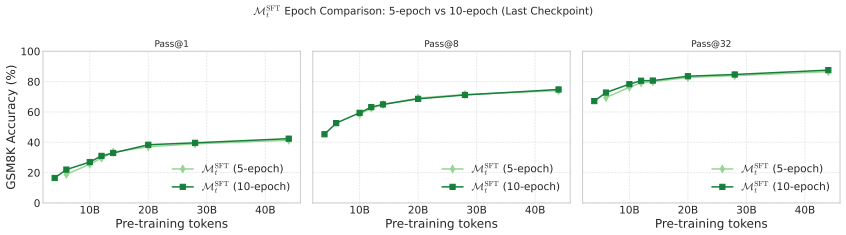


Figure 14: **SFT converges by 5 epochs.** GSM8K accuracy of $\mathcal{M}_t^{\text{SFT}}$ after training for different numbers of epochs on OpenMathInstruct. Performance plateaus by 5 epochs, which we use as the standard SFT training length for all $\mathcal{M}_t^{\text{SFT}}$ baselines.

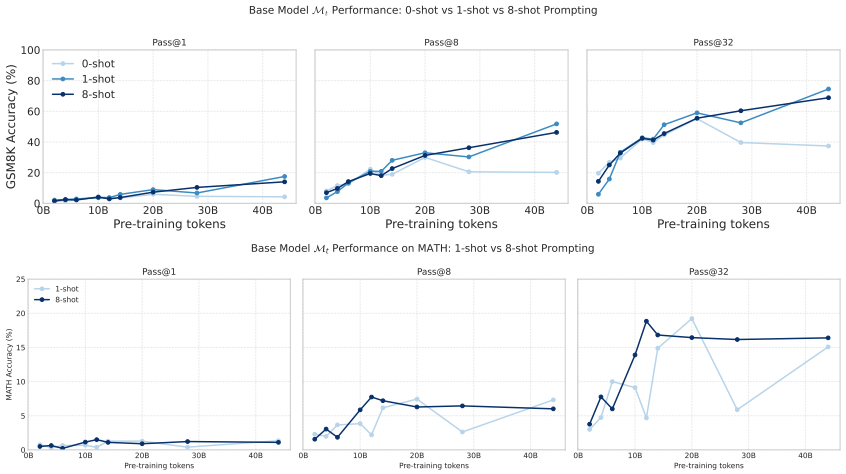


Figure 15: **Base checkpoints peak at 8-shot prompting.** Performance of pretraining checkpoints $\mathcal{M}_t$ on GSM8K (*top*) and MATH (*bottom*) under varying numbers of in-context examples. Across both benchmarks, accuracy is maximized at 8-shot, which we use throughout for $\mathcal{M}_t$ evaluation.

Model and Method. We conduct all experiments using the OLMo2 1B model (OLMo Team et al., 2025b).

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.

D.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 17 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 ($n = 64$)

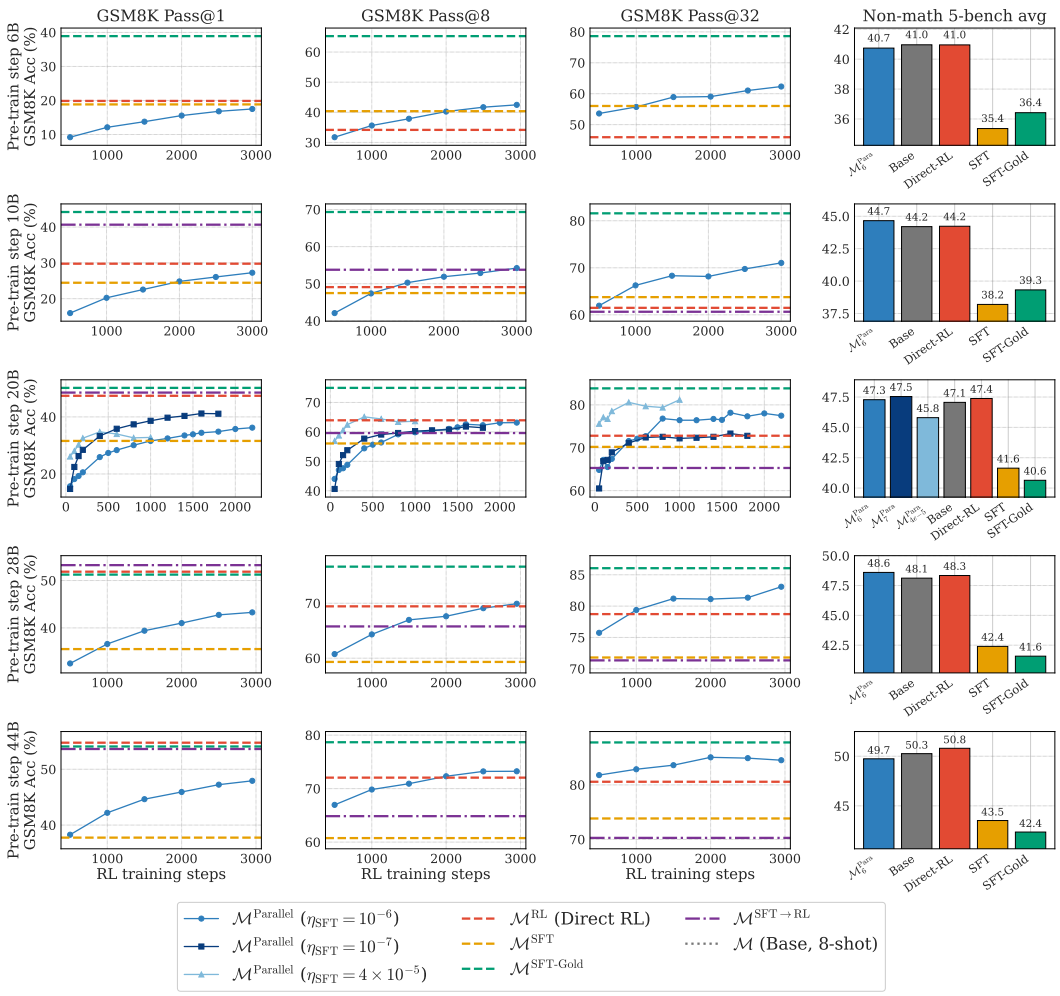


Figure 16: **Full results for parallel average algorithm.**

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.

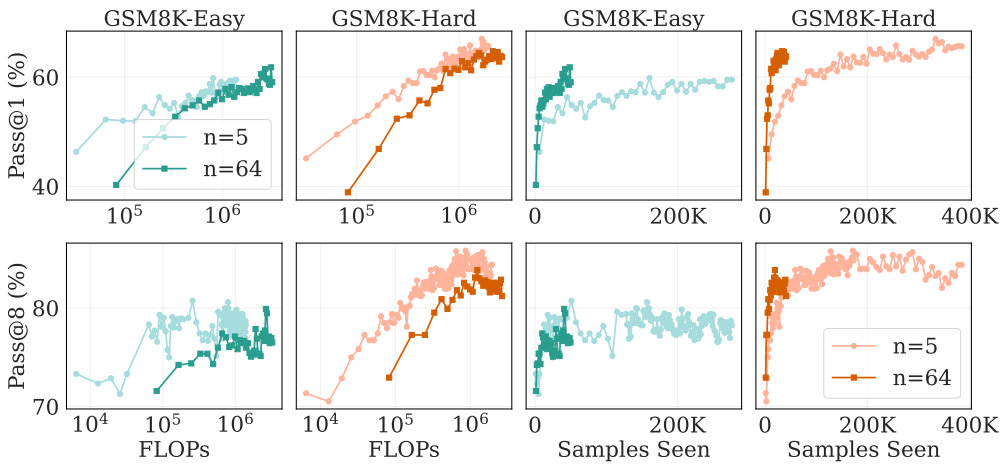


Figure 17: **Fewer rollouts are more FLOP-efficient at convergence.** GSM8K `pass@k` during RL training with $n = 5$ versus $n = 64$ rollouts per prompt, on training sets sub-sampled to be relatively easy or hard for the base model (a proxy for late vs. early pretraining). Asymptotic performance is similar across rollout counts. However, $n = 5$ achieves comparable accuracy at substantially lower FLOPs, especially on the harder split. Larger n is more sample-efficient per training example, but not per FLOP.