

Rethinking Expert Trajectory Utilization in LLM Post-training for Mathematical Reasoning

Bowen Ding^{1,2}, Yuhan Chen^{2*}, Jiayang Lyu^{2*}, Jiyao Yuan⁴, Qi Zhu⁴, Shuangshuang Tian^{2*}, Dantong Zhu^{2*}, Futing Wang^{1,2}, Heyuan Deng⁴, Fei Mi^{4†}, Lifeng Shang⁴, Tao Lin^{2,3†},

¹ Zhejiang University ² School of Engineering, Westlake University

³ Institute of Advanced Technology, Westlake Institute for Advanced Study

⁴ Huawei Noah’s Ark Lab

²{dingbowen, wangfuting, lintao}@westlake.edu.cn

⁴{yuanjiyao1, zhuqi41, dengheyuan, mifei2, Shang.Lifeng}@huawei.com

Abstract

Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) dominate the post-training landscape for mathematical reasoning, yet differ fundamentally in their reliance on expert trajectories. To understand the optimal way to harness these trajectories for maximizing performance, we propose the **Plasticity-Ceiling Framework**. This framework empirically grounds the post-training landscape by decomposing the final performance ceiling into the foundational SFT performance and the subsequent RL plasticity (i.e., the maximum improvement via RL). Through extensive benchmarking, we establish the **Sequential SFT-then-RL** pipeline as the superior standard, overcoming the stability and premature convergence deficits inherent in synchronized approaches. Furthermore, we derive precise scaling guidelines: (1) Transitioning to RL at the **Stable or Mild Overfitting Regime** of SFT maximizes the final ceiling by securing a robust SFT foundation with substantial RL plasticity; (2) Refuting the “Less is More” hypothesis in SFT-then-RL scaling, we demonstrate that **Data Scale** determines the primary post-training potential, while **Trajectory Difficulty** acts as a performance multiplier; and (3) The **Minimum Validation Loss of SFT** serves as a reliable indicator for selecting the expert trajectories that maximize the ultimate performance ceiling. Our findings provide actionable guidelines for extracting maximum value from expert trajectories. Code: <https://github.com/LINs-lab/RETU>.

1 Introduction

Mathematical reasoning demands strict logic and complex planning, making it a critical benchmark for AI intelligence. Evolving generic Large Language Models (LLMs) into specialized Large Reasoning Models (LRMs) capable of tackling these

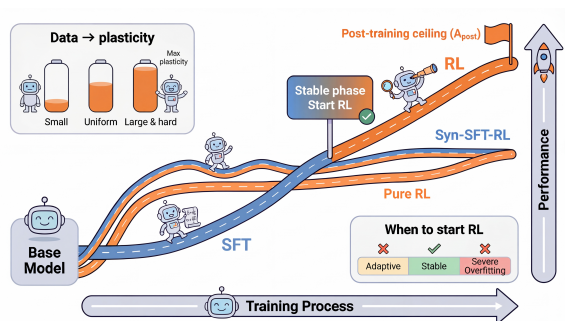


Figure 1: **The post-training overview.** The sequential SFT-then-RL pipeline (blue→orange) achieves the superior performance ceiling A_{post} , outperforming both Pure-RL (orange) and Synchronized SFT-RL (striped) pathways. Insets highlight that: (i) larger and more challenging datasets enhance RL plasticity (top-left); and (ii) RL yields optimal results when initiated from the Stable Regime of SFT (bottom-right).

tasks relies on effective post-training, often combining Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) (Shao et al., 2024; Yang et al., 2024).

These paradigms diverge fundamentally in their reliance on expert trajectories. SFT fully relies on expert trajectories (prompt-solution pairs) to instill reasoning priors via imitation, while RL methods such as GRPO (Shao et al., 2024) allow models to leverage prompt-answer pairs to self-explore reasoning paths through reward incentives, without the expert trajectory utilization. Despite the consensus on the necessity of both, a critical question remains unresolved: *What is the optimal mechanism to utilize expert trajectories (i.e., SFT data) to maximize the post-training performance ceiling in mathematical reasoning?*

The methodology for effective expert trajectory utilization currently faces an unresolved paradigm dilemma. Recent works propose Synchronized SFT-RL (Syn-SFT-RL) algorithms, such as UPT (Lv et al., 2025), SRFT (Fu et al., 2025), and LUFFY (Yan et al., 2025), which integrate imita-

* Work done at Westlake University as an intern.

† Corresponding authors.

tion loss directly into the RL optimization loop. While these methods often claim superior efficiency over sequential approaches, this advantage is critically constrained by their reliance on limited SFT data (only about 46K). This raises a fundamental question: whether Syn-SFT-RL maintains its claimed superiority when measured against the SFT and sequential SFT-then-RL baselines thoroughly optimized on significantly larger expert trajectory datasets.

Conversely, some LLM practitioners (Yang et al., 2025; GLM et al., 2025; DeepSeek-AI, 2025) typically favor the straightforward sequential SFT-then-RL pipeline. However, the principles governing this successful approach remain largely empirical and lack systematic definition in two critical areas. First, concerning the **Optimal Timing** for switching from SFT to RL, the criteria lack a formal guidance. Second, regarding **Data Properties**, although the “Less is More” (Ye et al., 2025; Muennighoff et al., 2025) approach achieves comparable SFT accuracy with minimal data, it is unclear whether this compromises the subsequent RL scaling potential or leads to premature convergence. Similarly, while harder data push SFT boundaries (Tong et al., 2024; Zhang et al., 2025a), its precise influence on the overall post-training ceiling remains unclarified. Consequently, these tensions highlight the urgent need for a unified framework to understand how the characteristics of SFT data dictate the entire post-training performance.

To rigorously address these systemic gaps, we propose a Plasticity-Ceiling analytical framework in § 4. This framework provides a unified view of all paradigms and enables the quantitative decomposition of the theoretical performance ceiling (A_{post}) into two measurable components: the **SFT Performance** (P_{sft}) achieved under SFT compute x_{sft} , and the remaining **RL Plasticity** (PL_{rl}), which represents the maximum potential for subsequent RL improvement.

By conducting extensive experiments with the large-scale (i.e., 889K unique samples in maximum) SFT data on Qwen2.5-7B (Qwen et al., 2025) and validating on Llama3.2-3B (Meta AI, 2024) on the test set including six popular mathematical benchmarks, we demystify expert trajectory utilization and establish a rigorous standard for post-training scaling in mathematical reasoning: **① Sequential Paradigm Dominance** (§ 6.1). We empirically establish the superiority of the Se-

quential SFT-then-RL pipeline over the unstable, sensitive Syn-SFT-RL approaches, as well as pure SFT and RL. A robust SFT phase is necessary to establish the foundational SFT performance (P_{sft}) and unlock the maximum plasticity (PL_{rl}) of subsequent RL. **② Switch RL until SFT Saturation** (§ 6.2.1). We identify the Stable or Mild-Overfitting Regime of validation loss saturation as the optimal SFT-to-RL transition window, both for the in-domain and the out-of-domain tasks. **③ Scale and Difficulty Extend Ceiling** (§ 6.2.2). We refute the “Less is More” hypothesis in the context of SFT-then-RL scaling. While minimal data yields SFT efficiency, the SFT data scale remains the primary determinant of the final ceiling, while the trajectory difficulty acts as a multiplier. Furthermore, the **Minimum Validation Loss of SFT** serves as a robust predictor of the final post-training ceiling.

Our contributions are summarized as follows: **①** We propose the **Plasticity-Ceiling Framework**, a theoretical mechanism that decomposes post-training performance into realized SFT performance and the subsequent RL plasticity to guide paradigm selection. **②** We systematically benchmark diverse training strategies, identifying the **Sequential SFT-then-RL** pipeline as the rigorous standard for stability and performance over other approaches. **③** We formulate **actionable insights** in SFT-then-RL pipeline, linking data properties and training dynamics to the final reasoning ceiling to enable more effective post-training development for mathematical reasoning.

2 Related Works

Post-Training Paradigms. Post-training primarily relies on SFT and RL. While theoretical works attempt to unify them (Swamy et al., 2025; Wang et al., 2025), they exhibit distinct empirical behaviors regarding generalization and distribution shifts (Huan et al., 2025; Shenfeld et al., 2025). The sequential SFT-then-RL strategy is the industrial standard (Yoshihara et al., 2025; Vattikonda et al., 2025), though optimizing the transition is non-trivial; Kang et al. (2025) caution that high SFT scores can be misleading, as over-fitted models may fail to improve during RL. Alternatively, Synchronized SFT-RL methods like LUFFY (Yan et al., 2025), UPT (Lv et al., 2025), SRFT (Fu et al., 2025) and Prefix-RFT (Huang et al., 2025) integrate imitation directly into RL to boost effi-

ciency. Our work systematically compares these paradigms to identify the optimal mechanism for maximizing the performance ceiling.

Expert Trajectories Utilization. The properties of SFT data critically influence the post-training. Regarding **scale**, a “Less is More” philosophy suggests that minimal, high-quality data suffices for SFT (Ye et al., 2025; Muennighoff et al., 2025). However, others argue that scale remains essential for complex reasoning (Sun et al., 2025). Regarding **difficulty**, methods like MetaMath (Yu et al., 2023) and D3 (Zhang et al., 2025a) demonstrate that harder, difficulty-aware data selection improves SFT outcomes. Crucially, prior works often evaluate SFT in isolation. We extend this inquiry to the RL phase, investigating how SFT data scale and difficulty dictate the model’s plasticity (its headroom for subsequent RL scaling) rather than just immediate accuracy after SFT.

3 Preliminary

In this section, we summarize algorithms of various post-training paradigms, each characterized by distinct approaches to utilizing expert trajectories. Details are in Appx. B.

SFT optimizes the policy π_θ via imitation learning on expert trajectory pairs (\mathbf{q}, τ) from \mathcal{D}_{SFT} (Abdulhai et al., 2023). While it reliably instills expert knowledge, its performance is strictly bounded by the quality of the training data and lacks self-exploration capabilities (Ouyang et al., 2022).

RL (GRPO & DAPO) enhances the model’s reasoning capability via reward-guided exploration. In contrast to SFT, standard RL algorithms such as GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025) typically employ outcome reward modeling and operate without the utilization of expert trajectories. We adopt **GRPO**, which eliminates the critic value function by employing group-normalized advantages $A_{i,t}$ derived from a group of G self-generated trajectories. The objective maximizes the expected advantage subject to a KL-divergence constraint against the reference policy π_{ref} . To further enhance stability, we utilize **DAPO** as our primary RL algorithm. DAPO builds upon GRPO by introducing asymmetric clipping ratios $(\epsilon_{\text{low}}, \epsilon_{\text{high}})$ and a dynamic difficulty sampling mechanism to filter extreme easy or hard prompts.

Synchronized SFT-RL (Syn-SFT-RL) paradigm attempts to merge the benefits of both SFT and

RL by explicitly injecting expert trajectories into the RL optimization loop. We consider four representative baselines: (1) **LUFFY** (Yan et al., 2025), which jointly optimizes on-policy rollouts and off-policy expert trajectories using regularized importance shaping; (2) **SRFT** (Fu et al., 2025), which balances SFT, off-policy RL, and likelihood-shaping losses via entropy-guided dynamic weights; and (3) **UPT** (Lv et al., 2025), which employs a hard gating mechanism to switch between SFT (stabilization) and GRPO (exploration) based on prompt-level reward thresholds. (4) **Prefix-RFT** (Huang et al., 2025), which samples prefixes from expert trajectories to guide model exploration, treating the composite sequence of an off-policy prefix and its on-policy continuation as a unified trajectory for reinforcement updates.

4 The Plasticity-Ceiling Framework

To systematically evaluate the trade-offs among post-training methods, we propose the Plasticity-Ceiling analytical framework. Unlike prior works that study SFT or RL scaling in isolation (Chen et al., 2025; Khatri et al., 2025), our framework treats the SFT-then-RL pipeline as a unified continuum. This allows us to quantify the respective contributions of the SFT and RL phases to the overall post-training performance ceiling (A_{post}) in Def. 1.

4.1 Decompose the Post-training Performance

Formally, we decompose the post-training performance P_{post} of the typical SFT-then-RL pipeline into three distinct components based on the training stages:

$$P_{\text{post}}(x_{\text{sft}}, x_{\text{rl}}) = P_0 + \underbrace{(P_{\text{sft}}(x_{\text{sft}}) - P_0)}_{\text{SFT gain, } \Delta P_{\text{sft}}(x_{\text{sft}})} + \underbrace{(P_{\text{rl}}(x_{\text{rl}}) - P_{\text{sft}}(x_{\text{sft}}))}_{\text{RL gain, } \Delta P_{\text{rl}}(x_{\text{rl}})} \quad (1)$$

$$P_{\text{sft}}(x_{\text{sft}} = 0) = P_0, P_{\text{rl}}(x_{\text{rl}} = 0) = P_{\text{sft}}(x_{\text{sft}}),$$

where P_0 denotes the base model’s initial performance, and $x_{\text{sft}}, x_{\text{rl}}$ denote the compute cost in FLOPs (see Appx. E) during the SFT and RL phases, respectively. This decomposition explicitly isolates the performance contributors: ΔP_{sft} represents the gain realized from SFT given cost x_{sft} , while ΔP_{rl} represents the gain from RL given cost x_{rl} . Notably, Eq. 1 reduces to Pure-SFT when $x_{\text{rl}} = 0$, and to Pure-RL (including Syn-SFT-RL variants) when $x_{\text{sft}} = 0$.

4.2 Ceiling and Plasticity

Since RL is the final stage of the post-training pipeline, its scaling behavior determines the ultimate performance ceiling. To quantify this, we model the post-training performance P_{post} as a function of RL compute x_{rl} using a sigmoidal law (Khatri et al., 2025). This law is initialized from $P_{\text{sft}}(x_{\text{sft}})$, which is the realized SFT outcome given the SFT compute x_{sft} :

$$P_{\text{post}}(x_{\text{sft}}, x_{\text{rl}}) = (\epsilon + P_{\text{sft}}(x_{\text{sft}})) + \frac{A_{\text{post}} - (\epsilon + P_{\text{sft}}(x_{\text{sft}}))}{1 + (x_{\text{rl}}/C_{\text{mid}_{\text{rl}}})^{-B_{\text{rl}}}} \quad (2)$$

In the equation, the alignment offset $\epsilon \geq 0$ captures potential discontinuities during the phase transition, providing the flexibility to accommodate empirical misalignments between the SFT endpoint and the global RL scaling trend (see non-zero ϵ fitting cases in Figure 8c). The midpoint parameter $C_{\text{mid}_{\text{rl}}}$ measures the RL compute required to realize half of the gain term, $A_{\text{post}} - (\epsilon + P_{\text{sft}}(x_{\text{sft}}))$, while the scaling steepness B_{rl} controls the rate of convergence. On top of these parameters, this framework allows us to formalize two key metrics as follows:

Definition 1 (Asymptotic Ceiling). For a given SFT compute allocation x_{sft} , the post-training ceiling A_{post} defines the theoretical upper bound of performance achievable in the limit of infinite RL computation. Formally, it is expressed as:

$$\lim_{x_{\text{rl}} \rightarrow \infty} P_{\text{post}}(x_{\text{sft}}, x_{\text{rl}}) = A_{\text{post}}(x_{\text{sft}}). \quad (3)$$

Definition 2 (RL Plasticity). The RL plasticity quantifies the maximum effective headroom available for performance improvement relative to the realized SFT baseline. Formally, it is defined as the difference between the asymptotic post-training ceiling and the corresponding SFT performance:

$$PL_{\text{rl}}(x_{\text{sft}}) = A_{\text{post}}(x_{\text{sft}}) - P_{\text{sft}}(x_{\text{sft}}). \quad (4)$$

These definitions imply that both A_{post} and PL_{rl} are x_{sft} -dependent quantities, whose values vary with the allocated SFT compute. In practice, A_{post} is estimated using a robust estimator detailed in Appx. F, with comprehensive fitting results provided in Table 4.

Theoretical Implication. Such framework reveals a fundamental insight: optimizing the SFT solely improves the P_{sft} , but it may shrink the PL_{rl}

and thereby constrain A_{post} if the SFT data is sub-optimal (e.g., limited in scale). Thus, it is crucial to understand how P_{sft} , PL_{rl} influence A_{post} in the post-training.

5 Experimental Setup

To determine the optimal mechanism for utilizing expert trajectories, we organize experiments progressively to address three core research questions:

RQ1: Paradigm Selection. Among Pure-RL, Pure-SFT, Syn-SFT-RL, and Sequential SFT-then-RL, which paradigm establishes the most effective post-training baseline, and what are their characterizations?

RQ2: Optimal SFT-to-RL Transition. Building upon the optimal paradigm identified in **RQ1**, what is the optimal time to transit to RL from SFT for a maximum final ceiling?

RQ3: Data Properties (Scale & Difficulty). With the paradigm (**RQ1**) and optimal timing strategy (**RQ2**) established, what roles do data scale and difficulty play in maximizing the performance ceiling, and do they support or refute the ‘‘Less is More’’ hypothesis?

5.1 Models and Data

Models. We primarily use Qwen2.5-7B (Qwen et al., 2025) in § 6.1 and 6.2, and Llama3.2-3B (Meta AI, 2024) in § 6.3 for cross-validation. In § 6.1, we apply Syn-SFT-RL algorithms to Qwen2.5-Math-7B (Yang et al., 2024) to further examine the influence of model priors.

Training Data. We construct SFT datasets of varying scales and difficulties by curating mathematical trajectories from distilled DeepSeek-R1 (DeepSeek-AI, 2025) outputs (Zhao et al., 2025; Tian et al., 2025). The resulting datasets include the large-scale SFT889K with around 889K unique samples, three medium-scale variants controlled for difficulty (Uniform/Easy/Hard102K, refer to Table 3 for the difficulty classification), and a held-out validation set Val-199 with 199 prompt and trajectory pairs. To test data efficiency, we also include S1K-1.1 (Muennighoff et al., 2025) (S1K for short), containing 1K high-quality R1-style trajectories.

For RL in the SFT-then-RL pipeline, we use RL62K, a filtered prompt set from Skywork-OR1-RL (He et al., 2025). For Syn-SFT-RL methods, we augment RL62K with expert trajectories in SFT889K to create MIX37K. Refer to Appx. G.1 for details.

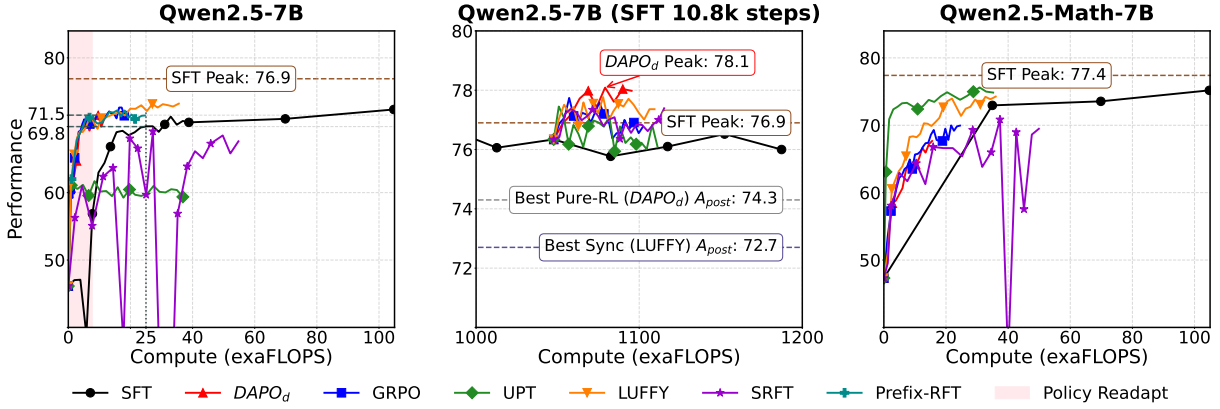


Figure 2: **The training dynamic comparison of post-training paradigms under different initializations.** (Left) Initializing from Qwen2.5-7B. Early RL-like runs converge quickly (except unstable instances), while early SFT shows a mild performance disruption due to policy readaptation (Zhang et al., 2025b). (Middle) Initializing from a saturated SFT checkpoint (10,800 steps on Qwen2.5-7B), where SFT-then-DAPO_d outperforms other paradigms. (Right) Initializing from Qwen2.5-Math-7B, where UPT and LUFFY demonstrate notable efficiency advantages.

Benchmarks. To prevent data leakage, we filter out the benchmark prompts with over 0.8 cosine similarity against our training set using Qwen3-8B-Embedding (Zhang et al., 2025c). We evaluate on the resulting 2,157 unique problems from the following cleaned benchmarks (counts denote original to cleaned): GSM8K (Cobbe et al., 2021) (1319 to 1317), OlympiadBench (He et al., 2024) (675 to 291), Minerva (Lewkowycz et al., 2022) (272 to 262), MATH (Lightman et al., 2023) (500 to 237), and AIME24/25 (LI et al., 2024) (30 to 25). We report the average performance on these unique problems unless otherwise specified. Furthermore, we collect GPQA-Diamond (GPQA-D) (Rein et al., 2023) (198 prompts) for graduate-level science and ARC-C (Clark et al., 2018) (1,172 prompts) for robust open-domain reasoning to evaluate out-of-domain (OOD) generalization.

5.2 Training and Evaluation

Training. Our experiments include two primary paradigms: (1) Syn-SFT-RL: we implement UPT (Lv et al., 2025), SRFT (Fu et al., 2025) and LUFFY (Yan et al., 2025) using the UPT codebase (Lv et al., 2025). Prefix-RFT (Huang et al., 2025) is implemented using its official codebase. (2) Sequential SFT-then-RL: we first fine-tune the base model on SFT data, then apply RL on the fine-tuned checkpoints. For comparison in § 6.1, we adopt GRPO and DAPO_d (GRPO with dynamic difficulty sampling (Yu et al., 2025)) as the Pure-RL baseline. When implementing the SFT-then-RL pipeline in 6.2, we use the enhanced DAPO_{dc} in the RL phase, which further incorporates asym-

metric ratio clipping into DAPO_d. See Appx. C for implementation details.

Evaluation. We report pass@1 accuracy sampled with a temperature of 0.7 and top-p 1.0 to ensure generation diversity. For the smaller AIME24/25 datasets, we use Avg@16 for robust estimation. All responses are generated with a maximum length of 8,192 tokens.

6 Experimental Results

6.1 Paradigms Comparison

To determine the optimal paradigm (RQ1), we systematically benchmark three approaches: Pure-SFT, Pure-RL (GRPO, DAPO_d), Syn-SFT-RL (LUFFY, SRFT, UPT, Prefix-RFT), and the SFT-then-RL pipeline. Following the data filtering setting of Yan et al. (2025), all RL (or Syn-SFT-RL) runs utilize MIX37K, a subset of SFT889K. Our preliminary experiments (Appx. H.1) confirm that training with these methods on the SFT889K leads to the suboptimal convergence. Thus, MIX37K serves as a robust proxy to capture their performance upper bounds. As shown in Figure 2, most of RL or Syn-SFT-RL methods typically saturate or destabilize within a single epoch, demonstrating that MIX37K suffices to capture their performance limits.

Limitations of Syn-SFT-RL. Contrasting prior claims (Yan et al., 2025; Lv et al., 2025; Fu et al., 2025), our experiments reveal severe practical limitations in Syn-SFT-RL methods, which exhibit training instability. For instance, SRFT shows performance fluctuations with a standard deviation

2.6 \times higher than the stable DAPO_d baseline in Figure 2 (Left) and fails to converge stably from a saturated SFT checkpoint (Figure 2 Middle). Furthermore, they are highly sensitive to model priors. For instance, UPT achieves superior convergence on Qwen2.5-Math-7B, outperforming other RL variants (Figure 2 Right). However, this advantage vanishes on general-purpose Qwen2.5-7B, where it even underperforms vanilla GRPO (Figure 2 Left).

RL Variants Trade Ceiling for Efficiency. Pure-RL and stable Syn-SFT-RL methods exhibit a common trade-off: high initial efficiency but a limited ceiling. As shown in the left panel of Figure 2, GRPO, DAPO_d, LUFFY, and Prefix-RFT rapidly reach around 71.5 points, outperforming Pure-SFT (69.8) within 25 exaFLOPs. However, their performance quickly plateaus thereafter. The middle panel of Figure 2 shows that even with infinity RL compute, DAPO_d improves only by an additional 2.8 points to reach a ceiling of 74.3, while LUFFY gains only 1.5 points to reach 72.7. This suggests that without a dedicated SFT phase, the headroom for improvement remains structurally constrained in post-training.

SFT Foundation and Sequential RL Maximization. In contrast, Pure-SFT demonstrates “Slow but High” scaling in the middle panel of Figure 2. SFT achieves the continuous improvement through the extensive training and reach a peak of 76.9 points. It significantly surpasses the highest Pure-RL and Syn-SFT-RL methods’ ceilings, which are 74.3 of DAPO_d and 72.7 of LUFFY, respectively. Crucially, transitioning to RL after SFT saturation successfully unlocks further gains. Specifically, SFT-then-DAPO_d achieves the best performance with 78.1 points, extending the post-training performance frontier by synergizing the SFT performance with further RL improvement.

Answer to RQ1: Sequential SFT-then-RL is the superior paradigm. Large-scale SFT provides the requisite robust foundation, and subsequent RL leverages this foundation to achieve the final performance frontier.

6.2 SFT-then-RL Pipeline

Given SFT-then-RL’s superiority (§ 6.1), we now examine the two determinants of its final ceiling (A_{post}): SFT compute allocation (**RQ2**) and data properties (**RQ3**). We first derive the optimal SFT-to-RL transition timing, then analyze the impact of data scale and difficulty.

6.2.1 The Impact of SFT Compute Allocation

Balancing realized SFT performance (P_{sft}) against preserving RL plasticity (PL_{rl}) is crucial for determining the optimal SFT-to-RL transition.

SFT Regimes. To rigorously identify the optimal transition point, we temporally partition the SFT process based on the SFT validation loss $L(x_{\text{sft}})$. The entire trajectory is segmented based on the following mathematically defined regions:

$$\begin{aligned}\mathcal{T}_{\text{stable}} &= \{x_{\text{sft}} \mid L(x_{\text{sft}}) \leq (1 + \delta)L_{\text{min}}\}, \\ \mathcal{T}_{\text{mild}} &= \{x_{\text{sft}} \mid (1 + \delta)L_{\text{min}} < L(x_{\text{sft}}) \\ &\quad < (1 + \delta_2) \cdot L_{\text{min}}\}, \\ \mathcal{T}_{\text{severe}} &= \{x_{\text{sft}} \mid L(x_{\text{sft}}) \geq (1 + \delta_2) \cdot L_{\text{min}}\}.\end{aligned}\tag{5}$$

where L_{min} is the global minimum validation loss observed during training, (δ, δ_2) are tolerance thresholds being set as (0.02, 0.1) empirically. Therefore, we have four separable regimes characterizing the SFT convergence:

- **Adaptive Regime** ($\mathcal{P}_{\text{adapt}}$), where SFT is underfitting in the region.

$$\mathcal{P}_{\text{adapt}} = \{x_{\text{sft}} \mid x_{\text{sft}} < \min \mathcal{T}_{\text{stable}}\}\tag{6}$$

- **Stable Regime** ($\mathcal{P}_{\text{stable}}$), where the validation loss saturates within a small tolerance threshold of 2% (i.e., $\delta = 0.02$).

$$\mathcal{P}_{\text{stable}} = \mathcal{T}_{\text{stable}}\tag{7}$$

- **Mild Overfitting Regime** ($\mathcal{P}_{\text{mild}}$), where the region where loss rises slightly but remains below the 10% tolerance, representing the “risky sweet spot.”

$$\mathcal{P}_{\text{mild}} = \{x_{\text{sft}} \mid x_{\text{sft}} > \max \mathcal{T}_{\text{stable}} \text{ and } x_{\text{sft}} \in \mathcal{T}_{\text{mild}}\}\tag{8}$$

- **Severe Overfitting Regime** ($\mathcal{P}_{\text{severe}}$), where loss significantly diverges ($\geq 10\%$ rise when $\delta_2 = 0.1$), leading to rapid plasticity collapse (see Easy102K in Figure 3).

$$\mathcal{P}_{\text{severe}} = \{x_{\text{sft}} \mid x_{\text{sft}} > \max \mathcal{T}_{\text{stable}} \text{ and } x_{\text{sft}} \in \mathcal{T}_{\text{severe}}\}\tag{9}$$

The Dynamics of Post-training Ceiling. The blue solid line in Figure 3 illustrates how A_{post} evolves across these regimes. We observe that initiating RL prematurely during the Adaptive Regime is consistently suboptimal because the model lacks foundational competence that subsequent RL cannot fully recover. For instance, on SFT889K, switching early at 69.8 exaFLOPs yields a ceiling of only 80.7 points, whereas extending training to the Stable Regime (698.4 exaFLOPs) boosts the ceiling to its peak of 84.0

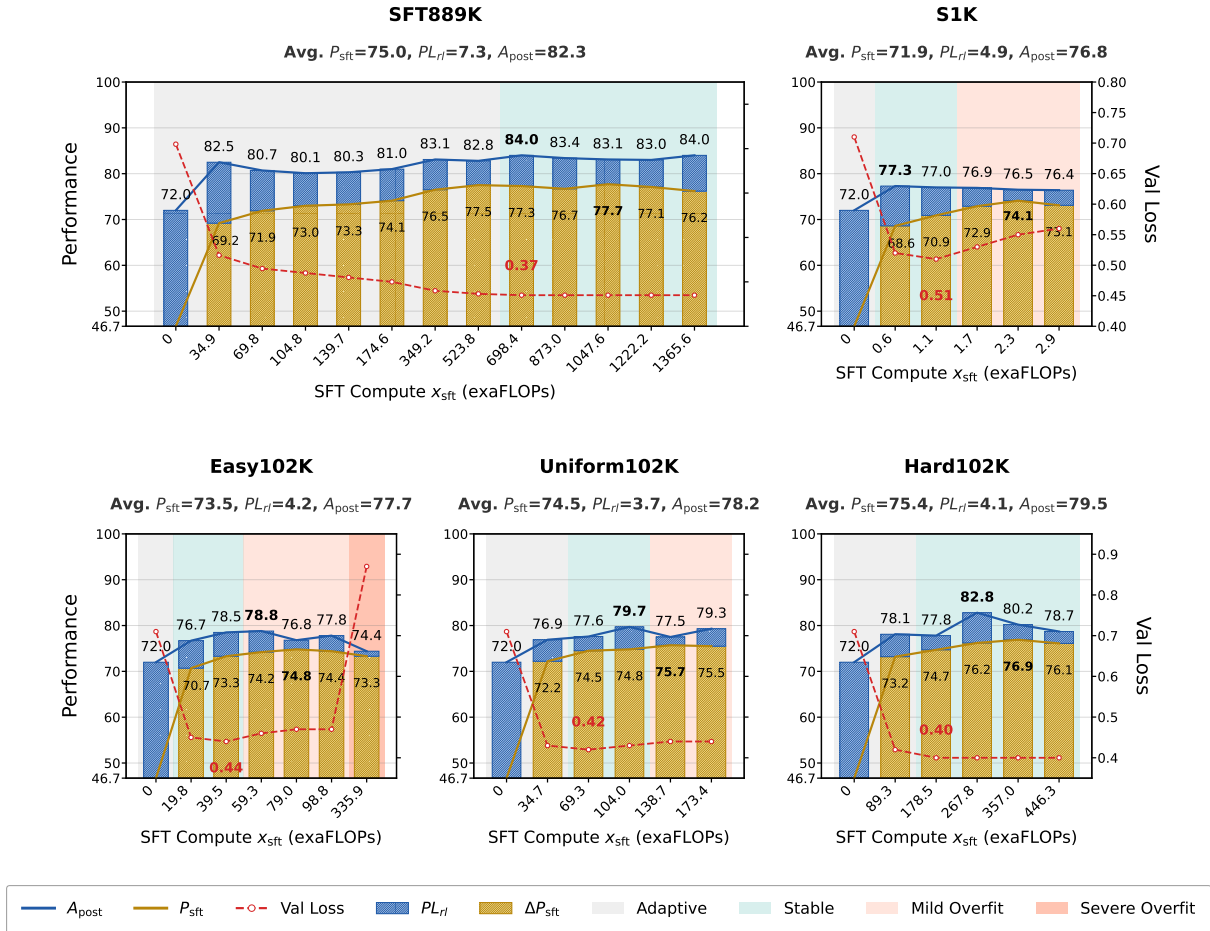


Figure 3: **SFT-then-RL scaling dynamics across data properties.** The post-training ceiling (A_{post}) decomposes into SFT performance (P_{sft}) and RL plasticity (PL_{rl}), plotted against SFT compute x_{sft} (exaFLOPs). Background colors mark the four SFT regimes (§ 6.2.1), segmented by the SFT validation loss curves. Note: Pure-RL ($x_{\text{sft}} = 0$) and the Easy102K severe-overfitting checkpoint ($x_{\text{sft}} = 335.9$) are excluded from the averaging statistics.

points. Ideally, for most of SFT datasets (e.g., SFT889K, S1K, Uniform102K, Hard102K), the Stable Regime aligns perfectly with peak performance. However, on Easy102K, the peak ceiling shifts into the Mild Overfitting Regime, indicating that a slightly delayed transition is acceptable and can even be beneficial due to P_{sft} improvement.

Conversely, aggressively continuing SFT into the Severe Overfitting Regime is detrimental. As demonstrated on Easy102K in Figure 3, training SFT to 335.9 exaFLOPs leads to a rapid decline in the final ceiling due to a collapse in RL plasticity.

Evaluation on OOD Generalization. Figure 4 illustrates the OOD performance of SFT889K checkpoints and their subsequent RL progress. The left panel demonstrates substantial SFT-driven gains: ARC-C improves by 10 points to surpass 90.0 with 104.8 exaFLOPs (1,080 steps), while GPQA-D increases by 30 points to peak at 57.0 after 1047.6 exaFLOPs (10,800 steps). Contrary to claims that intensive SFT degrades generaliza-

tion (Rajani et al., 2025; Shenfeld et al., 2025), our results prove it establishes a robust OOD reasoning foundation. We attribute this discrepancy to initialization, as prior studies evaluated instruction-tuned models rather than pre-trained base models.

The middle and right panels depict end-to-end performance as RL compute evolves. Transitioning to RL prematurely (SFT-360-RL, Adaptive Regime) yields significant RL-phase gains but suffers from the early convergence, restricting optimal scores to 92.0 on ARC-C and 51.0 on GPQA-D. Conversely, transitioning from a saturated SFT foundation (SFT-10800-RL, Stable Regime) provides marginal RL improvements but unlocks higher ultimate ceilings of 94.0 and 57.0, respectively. Thus, training SFT to saturation remains vital for maximizing OOD generalization.

Answer to RQ2: Train SFT to Saturation. Across both In-Domain (ID) and OOD tasks, the optimal strategy targets the **Stable Regime**, avoid-

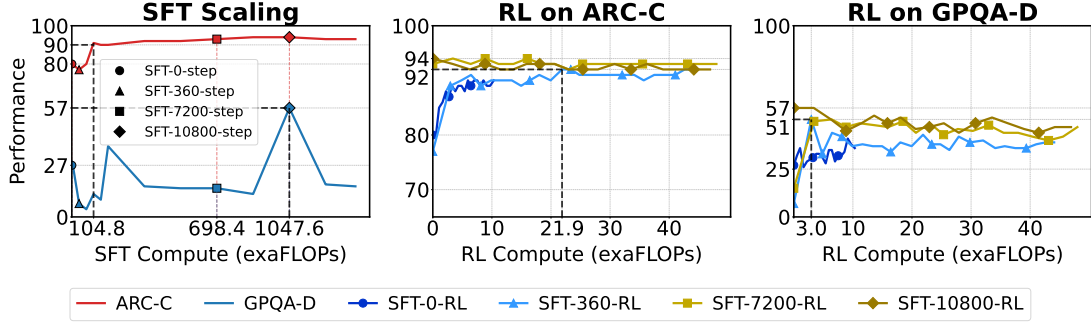


Figure 4: **The OOD generalization analysis for SFT-then-RL on SFT889K.** (Left) The performance dynamics on ARC-C and GPQA-D during SFT. (Middle&Right) The performance dynamics on ARC-C and GPQA-D during RL initiating from different SFT checkpoints.

ing Severe Overfitting to preserve RL plasticity. While Mild Overfitting may be tolerable for select datasets, the Stable Regime provides a robust standard for maximizing the performance ceiling with scalable data.

6.2.2 The Impact of SFT Data Properties

Data scale and difficulty are critical determinants of the quality of the SFT prior. We now focus on investigating how these data properties influence the asymptotic post-training performance ceiling. While we focus on representative scales here, extended results across additional data granularities are provided in Appx. H.2.

Larger Scale Begets Higher Ceiling. Comparing datasets of varying scales (S1K, Uniform102K, and SFT889K in Figure 3) reveals that while minimal data can achieve rapid initial SFT gains, extensive data scale is indispensable for reaching a higher post-training ceiling. Initially, small-scale data exhibits deceptive efficiency: S1K achieves an SFT performance of 74.1 points using only 2.3 exaFLOPs, matching the similar performance level that requires 69.3 exaFLOPs on Uniform102K ($P_{\text{sft}}=74.5$) and 174.6 exaFLOPs on SFT889K ($P_{\text{sft}}=74.1$).

However, this efficiency proves to be unsustainable. The realized SFT performance P_{sft} of S1K saturates prematurely at this level. In contrast, Uniform102K and SFT889K continue to improve with additional compute. Uniform102K reaches SFT performances of 74.8 at 104.0 exaFLOPs while SFT889K reaches 77.3 at 698.4 exaFLOPs, which thereby establish the superior foundation for the subsequent RL phase. Crucially, the large-scale SFT also preserves greater RL plasticity. SFT889K maintains an average PL_{rl} of 7.3, exceeding both S1K and Uniform102K by 2.4 and 3.7 points. Consequently, by leveraging both higher realized SFT performance P_{sft} and en-

hanced RL plasticity PL_{rl} , the large-scale SFT unlocks a significantly higher post-training ceiling.

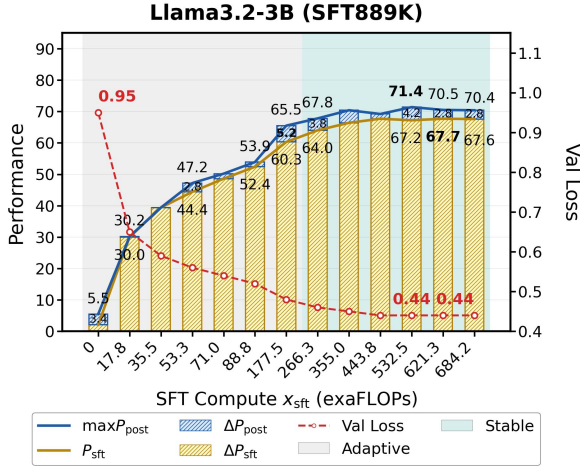
Harder Data Elevates the Ceiling. Controlling the dataset size as 102K, we investigate how trajectory difficulty affects performance by comparing Easy102K, Uniform102K, and Hard102K. As shown in the bottom row of Figure 3, training on harder trajectories leads to higher returns. Specifically, Hard102K achieves a higher SFT performance (P_{sft}) of 75.4 points in average, outperforming Easy102K (73.5) and Uniform102K (74.6) by 1.9 and 0.8 percentage points, respectively. At the same time, Hard102K maintains a similar RL plasticity (PL_{rl}) to its 102K counterparts, with a score of 4.1 points. This combination, which includes higher SFT performance and decent RL plasticity, makes harder data the superior choice for maximizing the post-training ceiling.

Minimum Validation Loss as an Indicator. Across diverse SFT data configurations, the minimum validation loss shows a strong negative correlation with A_{post} (Pearson $r=-0.92$, Figure 8a). Hence, it serves as an efficient a priori indicator: a lower loss reliably predicts a higher post-training ceiling in the SFT-then-RL pipeline, bypassing expensive RL runs.

Answer to RQ3: Scale Dominates, Difficulty Optimizes. Refuting “Less is More”, we establish Data Scale as the primary factor to improve the post-training ceiling, while Difficulty acts as a multiplier. Harder trajectories are helpful when the data scale is limited. Thus, scaling must prioritize volume before difficulty, with the final ceiling reliably indicated by the minimum SFT validation loss.

6.3 The Validation on Llama3.2-3B

We validate our findings on Llama3.2-3B (Meta AI, 2024) to for cross-model generalization. To prioritize practical relevance, we re-



References

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, and 21 others. 2016. [Tensorflow: Large-scale machine learning on heterogeneous distributed systems](#). *Preprint*, arXiv:1603.04467.
- Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. 2023. [Lmrl gym: Benchmarks for multi-turn reinforcement learning with language models](#). *Preprint*, arXiv:2311.18232.
- Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. [Gqa: Training generalized multi-query transformer models from multi-head checkpoints](#). *Preprint*, arXiv:2305.13245.
- Yangyi Chen, Binxuan Huang, Yifan Gao, Zhengyang Wang, Jingfeng Yang, and Heng Ji. 2025. [Scaling laws for predicting downstream performance in llms](#). *Preprint*, arXiv:2410.08527.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. [Think you have solved question answering? try arc, the ai2 reasoning challenge](#). *Preprint*, arXiv:1803.05457.
- 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. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- DeepSeek-AI. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025a. [Deepseek-v3 technical report](#). *Preprint*, arXiv:2412.19437.
- DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025b. [Deepseek-v3.2: Pushing the frontier of open large language models](#). *Preprint*, arXiv:2512.02556.
- Volcano Engine. 2023. [VERL utils: FLOPs counter \(line 149\)](#). https://github.com/volcengine/verl/blob/59049a66/verl/utils/flops_counter.py#L149. Version 59049a6; Accessed: 2024-12-01.
- Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. 2025. [SRFT: A single-stage method with supervised and reinforcement fine-tuning for reasoning](#). *arXiv preprint arXiv:2506.19767*.
- Team GLM, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, and 152 others. 2025. [Glm-4.5: Agentic, reasoning, and coding \(arc\) foundation models](#). *Preprint*, arXiv:2508.06471.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. [Olympiad-bench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems](#). *Preprint*, arXiv:2402.14008.
- Jujie He, Jiakai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. 2025. [Skywork open reasoner 1 technical report](#). *arXiv preprint arXiv:2505.22312*.
- Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. 2025. [Does math reasoning improve general llm capabilities? understanding transferability of llm reasoning](#). *Preprint*, arXiv:2507.00432.
- Zeyu Huang, Tianhao Cheng, Zihan Qiu, Zili Wang, Yinghui Xu, Edoardo M. Ponti, and Ivan Titov. 2025. [Blending supervised and reinforcement fine-tuning with prefix sampling](#). *arXiv preprint arXiv:2507.01679*.
- P.J. Huber and E.M. Ronchetti. 2011. [Robust Statistics](#). Wiley Series in Probability and Statistics. Wiley.
- Hugging Face. 2024. [Math-verify](#). <https://github.com/huggingface/Math-Verify>.
- Boris Iglewicz and David C Hoaglin. 1993. [How to detect and handle outliers](#), volume 16. Asqc Quality Press Milwaukee, WI.
- Feiyang Kang, Michael Kuchnik, Karthik Padthe, Marin Vlastelica, Ruoxi Jia, Carole-Jean Wu, and Newsha Ardalani. 2025. [Quagmires in sft-rl post-training: When high sft scores mislead and what to use instead](#). *Preprint*, arXiv:2510.01624.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. [Scaling laws for neural language models](#). *Preprint*, arXiv:2001.08361.

- Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S. Dhillon, David Brandfonbrener, and Rishabh Agarwal. 2025. [The art of scaling reinforcement learning compute for llms](#). *Preprint*, arXiv:2510.13786.
- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. [Solving quantitative reasoning problems with language models](#). *Preprint*, arXiv:2206.14858.
- Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. 2013. [Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median](#). *Journal of Experimental Social Psychology*, 49(4):764–766.
- Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. NuminaMath. [<https://github.com/project-numina/aimo-progress-prize>](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf).
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. [Let’s verify step by step](#). *Preprint*, arXiv:2305.20050.
- Xingtai Lv, Yuxin Zuo, Youbang Sun, Hongyi Liu, Yuntian Wei, Zhekai Chen, Lixuan He, Xuekai Zhu, Kaiyan Zhang, Bingning Wang, Ning Ding, and Bowen Zhou. 2025. [Towards a unified view of large language model post-training](#). *Preprint*, arXiv:2509.04419.
- Meta AI. 2024. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>. Meta AI blog; accessed 2025-04-13; 15 minute read.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. [s1: Simple test-time scaling](#). *Preprint*, arXiv:2501.19393.
- Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. 2021. [Efficient large-scale language model training on gpu clusters using megatron-lm](#). *Preprint*, arXiv:2104.04473.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.
- Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, and 25 others. 2025. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Neel Rajani, Aryo Pradipta Gema, Seraphina Goldfarb-Tarrant, and Ivan Titov. 2025. [Scalpel vs. hammer: Grpo amplifies existing capabilities, sft replaces them](#). *Preprint*, arXiv:2507.10616.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. [Gpqa: A graduate-level google-proof q&a benchmark](#). *Preprint*, arXiv:2311.12022.
- Peter J Rousseeuw. 1984. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880.
- Peter J. Rousseeuw and Katrien Driessen. 2006. [Computing lts regression for large data sets](#). *Data Min. Knowl. Discov.*, 12(1):29–45.
- Peter J Rousseeuw and Annick M Leroy. 1987. *Robust regression and outlier detection*. John Wiley & Sons.
- Yangjun Ruan, Chris J. Maddison, and Tatsunori Hashimoto. 2024. [Observational scaling laws and the predictability of language model performance](#). *Preprint*, arXiv:2405.10938.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *Preprint*, arXiv:2402.03300.
- Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. 2025. [RL’s razor: Why online reinforcement learning forgets less](#). *Preprint*, arXiv:2509.04259.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. [Hybridflow: A flexible and efficient rlhf framework](#). *arXiv preprint arXiv:2409.19256*.

- Yiyou Sun, Georgia Zhou, Hao Wang, Dacheng Li, Nouha Dziri, and Dawn Song. 2025. [Climbing the ladder of reasoning: What llms can and still can't solve after sft?](#) *Preprint*, arXiv:2504.11741.
- Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J. Andrew Bagnell. 2025. [All roads lead to likelihood: The value of reinforcement learning in fine-tuning.](#) *Preprint*, arXiv:2503.01067.
- Xiaoyu Tian, Sitong Zhao, Haotian Wang, Shuaiting Chen, Yiping Peng, Yunjie Ji, Han Zhao, and Xianggang Li. 2025. [Deepdistill: Enhancing llm reasoning capabilities via large-scale difficulty-graded data training.](#) *Preprint*, arXiv:2504.17565.
- Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. 2024. [Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving.](#) <https://arxiv.org/abs/2407.13690>. ArXiv:2407.13690, cs.CL.
- Dheeraj Vattikonda, Santhoshi Ravichandran, Emiliano Peñaloza, Hadi Nekoei, Megh Thakkar, Thibault Le Sellier de Chezelles, Nicolas Gontier, Miguel Muñoz-Mármol, Sahar Omid Shayegan, Stefania Raimondo, Xue Liu, Alexandre Drouin, Laurent Charlin, Alexandre Piché, Alexandre Lacoste, and Massimo Caccia. 2025. [How to train your LLM web agent: A statistical diagnosis.](#) *arXiv preprint arXiv:2507.04103*.
- Bo Wang, Qinyuan Cheng, Runyu Peng, Rong Bao, Peiji Li, Qipeng Guo, Linyang Li, Zhiyuan Zeng, Yunhua Zhou, and Xipeng Qiu. 2025. [Implicit reward as the bridge: A unified view of sft and dpo connections.](#) *Preprint*, arXiv:2507.00018.
- Hao Wang, Hao Gu, Hongming Piao, Kaixiong Gong, Yuxiao Ye, Xiangyu Yue, Sirui Han, Yike Guo, and Dapeng Wu. 2026. [Learning while staying curious: Entropy-preserving supervised fine-tuning via adaptive self-distillation for large reasoning models.](#) *Preprint*, arXiv:2602.02244.
- Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025. [Learning to reason under off-policy guidance.](#) *Preprint*, arXiv:2504.14945.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report.](#) *Preprint*, arXiv:2505.09388.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. [Qwen2.5-math technical report: Toward mathematical expert model via self-improvement.](#) *arXiv preprint arXiv:2409.12122*.
- Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. [Limo: Less is more for reasoning.](#) *Preprint*, arXiv:2502.03387.
- Hiroshi Yoshihara, Taiki Yamaguchi, and Yuichi Inoue. 2025. [A practical two-stage recipe for mathematical llms: Maximizing accuracy with sft and efficiency with reinforcement learning.](#) *Preprint*, arXiv:2507.08267.
- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhengguo Li, Adrian Weller, and Weiyang Liu. 2023. [Metamath: Bootstrap your own mathematical questions for large language models.](#) *arXiv preprint arXiv:2309.12284*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. [DAPO: An open-source llm reinforcement learning system at scale.](#) *arXiv preprint arXiv:2503.14476*.
- Jia Zhang, Chen-Xi Zhang, Yao Liu, Yi-Xuan Jin, Xiao-Wen Yang, Bo Zheng, Yi Liu, and Lan-Zhe Guo. 2025a. [D3: Diversity, difficulty, and dependability-aware data selection for sample-efficient llm instruction tuning.](#) *Preprint*, arXiv:2503.11441.
- Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2025b. [On-policy rl meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting.](#) *Preprint*, arXiv:2508.11408.
- Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025c. [Qwen3 embedding: Advancing text embedding and reranking through foundation models.](#) *arXiv preprint arXiv:2506.05176*.
- Han Zhao, Haotian Wang, Yiping Peng, Sitong Zhao, Xiaoyu Tian, Shuaiting Chen, Yunjie Ji, and Xianggang Li. 2025. [1.4 million open-source distilled reasoning dataset to empower large language model training.](#) *Preprint*, arXiv:2503.19633.

A Experimental Platforms

All SFT experiments in § 6.1 run on 16 GPUs; RL and Syn-SFT-RL experiments in § 6.1 are implemented on 8 GPUs, and RL experiments in § 6.2 are conducted on 128 Ascend 910B NPUs.

B Post-training Paradigms

We summarize the details of algorithmic foundations used in our study: (1) standard supervised fine-tuning (SFT), (2) reinforcement learning (GRPO and DAPO), and (3) synchronized SFT-RL (Syn-SFT-RL) fusion methods used as single-stage baselines.

B.1 SFT

SFT tunes the policy π_θ via imitation learning using the answer and expert trajectory pair $(\mathbf{q}, \boldsymbol{\tau})$ in the SFT dataset \mathcal{D}_{SFT} :

$$\mathcal{J}_{\text{SFT}}(\theta) = -\mathbb{E}_{(\mathbf{q}, \boldsymbol{\tau}) \sim \mathcal{D}_{\text{SFT}}} \left[\sum_{t=1}^{|\boldsymbol{\tau}|} \log \pi_\theta(\boldsymbol{\tau}_t | \mathbf{q}, \boldsymbol{\tau}_{<t}) \right] \quad (10)$$

This paradigm reliably instills instruction-following and basic reasoning capabilities in the model (Ouyang et al., 2022; DeepSeek-AI, 2025).

B.2 RL

RL extends the model beyond imitation by optimizing reward-guided exploration. GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025) are two widely-used RL algorithms.

GRPO replaces a critic with a group-normalized advantage $(A_{i,t})$. For each query-answer pair (\mathbf{q}, \mathbf{a}) in dataset \mathcal{D}_{RL} , GRPO samples G response trajectories $\{\boldsymbol{\tau}_i\}_{i=1}^G$ based on the old policy $\pi_{\theta_{\text{old}}}$. Each trajectory receives a rule-derived reward score R_i . The group-normalized advantage is computed as:

$$A_{i,t} = \frac{R_i - \text{mean}(\{R_j\}_{j=1}^G)}{\text{std}(\{R_j\}_{j=1}^G)} \quad (11)$$

With the advantage, GRPO aims to maximize the expected advantage while regularizing the policy towards a reference policy π_{ref} via the KL divergence term $\beta \cdot \mathbb{D}_{\text{KL}}[\pi_\theta || \pi_{\text{ref}}]$. The policy loss $\mathcal{J}_{\text{GRPO}}(\theta)$ is:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(\mathbf{q}, \mathbf{a}) \sim \mathcal{D}_{\text{RL}}, \{\boldsymbol{\tau}_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}} \left[\frac{\sum_{i=1}^G \sum_{t=1}^{|\boldsymbol{\tau}_i|} \min(r_i^t A_{i,t}, C_i^t A_{i,t})}{\sum_{i=1}^G |\boldsymbol{\tau}_i|} \right] - \beta \cdot \mathbb{D}_{\text{KL}}[\pi_\theta || \pi_{\text{ref}}] \quad (12)$$

where $r_i^t = \frac{\pi_\theta(\boldsymbol{\tau}_{i,t} | \mathbf{q}, \boldsymbol{\tau}_{i,<t})}{\pi_{\theta_{\text{old}}}(\boldsymbol{\tau}_{i,t} | \mathbf{q}, \boldsymbol{\tau}_{i,<t})}$ represents the importance ratio between the new and old policies for token $\boldsymbol{\tau}_{i,t}$. Its clipped counterpart, $C_i^t = \text{clip}(r_i^t, 1 - \epsilon, 1 + \epsilon)$, confines the policy update within a trust region, preventing excessively large and destabilizing policy updates.

DAPO (Yu et al., 2025) further stabilizes training via asymmetric clipping $(\epsilon_{\text{low}}, \epsilon_{\text{high}})$ and dynamically filter the prompts with all correct or wrong on-policy generations. We adopt DAPO as our primary RL algorithm due to its robustness on mathematical reasoning tasks.

B.3 Syn-SFT-RL

The Syn-SFT-RL paradigm merges SFT and RL by injecting expert trajectories into the optimization loop. We introduce four typical algorithms: LUFFY (Yan et al., 2025), SRFT (Fu et al., 2025), UPT (Lv et al., 2025) and Prefix-RFT (Huang et al., 2025).

LUFFY modifies the $\mathcal{J}_{\text{GRPO}}(\theta)$ in Eq. 12 by jointly optimizing on-policy trajectories and off-policy ones. It removes both the KL regularization and importance-ratio clipping, and aggregates token-level advantages over a mixture of SFT and RL data. The mixture dataset \mathcal{D}_{MIX} contains triplets $(\mathbf{q}, \{\boldsymbol{\tau}_j\}_{j=1}^N, \mathbf{a})$ with the prompt \mathbf{q} , N expert trajectories $\{\boldsymbol{\tau}_j\}_{j=1}^N$ ($N=1$ as the official setup), and answer \mathbf{a} . Hence, LUFFY’s loss is formalized as:

$$\mathcal{J}_{\text{LUFFY}}(\theta) = \mathbb{E}_{(\mathbf{q}, \{\boldsymbol{\tau}_j\}_{j=1}^N, \mathbf{a}) \sim \mathcal{D}_{\text{MIX}}, \{\boldsymbol{\tau}_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{Z} \sum_{j=1}^N \sum_{t=1}^{|\boldsymbol{\tau}_j|} \hat{r}_j^t \hat{A}_{j,t} + \frac{1}{Z} \sum_{i=1}^G \sum_{t=1}^{|\boldsymbol{\tau}_i|} r_i^t \hat{A}_{i,t} \right], \quad (13)$$

where $Z = \sum_{j=1}^N |\boldsymbol{\tau}_j| + \sum_{i=1}^G |\boldsymbol{\tau}_i|$ normalizes over all tokens, and the mixed advantages are computed without normalization:

$$\hat{A}_{i,t} = R_i - \text{mean}(\{R_j\}_{j=1}^N \cup \{R_i\}_{i=1}^G), \quad (14)$$

To avoid entropy collapse on off-policy data, LUFFY further applies regularized importance shaping, which transforms the importance ratio r_j^t to $\hat{r}_j^t = r_j^t / (r_j^t + \gamma)$ with a small constant $\gamma = 0.1$.

SRFT combines four components: (i) the standard SFT loss \mathcal{J}_{SFT} in Eq. 10, (ii) the off-policy loss \mathcal{J}_{off} from LUFFY (the first term in Eq. 13), and (iii) on-policy objectives for positive and negative trajectories in Eq. 15. For M on-policy pos-

itive rollouts $\{\tau_i^+\}_{i=1}^M$ and $G - M$ on-policy negative ones $\{\tau_j^-\}_{j=1}^{G-M}$, SRFT maximizes the likelihood of positive trajectories while suppressing that of negative ones:

$$\begin{aligned}\mathcal{J}_{\text{pos}}(\theta) &= -\mathbb{E}\left[\sum_{t=1}^{|\tau_i^+|} \log \pi_{\theta}(\tau_{i,t}^+ | \mathbf{q}_i, \tau_{i,<t}^+)\right], \\ \mathcal{J}_{\text{neg}}(\theta) &= \mathbb{E}\left[\sum_{t=1}^{|\tau_j^-|} \log \pi_{\theta}(\tau_{j,t}^- | \mathbf{q}_j, \tau_{j,<t}^-)\right].\end{aligned}\quad (15)$$

The final SRFT objective uses entropy-guided dynamic weights:

$$\begin{aligned}\mathcal{J}_{\text{SRFT}} &= w_1 \mathcal{J}_{\text{SFT}} + \mathcal{J}_{\text{off}} + w_2 \mathcal{J}_{\text{pos}} + \mathcal{J}_{\text{neg}}, \\ w_1 &= 0.5 \cdot \text{stop_grad}(e^{-\mathcal{H}(\pi_{\theta})}), \\ w_2 &= 0.1 \cdot \text{stop_grad}(e^{\mathcal{H}(\pi_{\theta})}).\end{aligned}\quad (16)$$

where $\mathcal{H}(\pi_{\theta})$ denotes the policy entropy and `stop_grad` prevents gradients from flowing through the weights.

UPT employs a hard gating mechanism to mix SFT and RL. Let p denote the average reward over the trajectories sampled for the current prompt \mathbf{q} , and γ be a threshold. UPT defines a mixed loss

$$\mathcal{J}_{\text{UPT}} = f_p \mathcal{J}_{\text{SFT}} + g_p \mathcal{J}_{\text{GRPO}},\quad (17)$$

where (f_p, g_p) are determined by p :

$$(f_p, g_p) = \begin{cases} (1, 0), & p \leq \gamma, \\ (0, 1), & p > \gamma. \end{cases}\quad (18)$$

When the model performs poorly on a prompt ($p \leq \gamma$), the gate prioritizes SFT-style imitation. Once the reward exceeds the threshold ($p > \gamma$), the gate switches to pure GRPO optimization to focus on exploration.

Prefix-RFT blends SFT and RL by utilizing sampled expert prefixes to guide online exploration. For a given prompt \mathbf{q} and an expert trajectory τ^* , a prefix sequence of length l is sampled. To mitigate position bias and introduce curriculum learning, the prefix length ratio is governed by a cosine decay scheduler whose lower bound gradually decays toward zero. The current policy $\pi_{\theta_{\text{old}}}$ then generates a continuation τ^{cont} to complete the response, yielding a composite hybrid trajectory τ_{hybrid} . The model optimizes over a mixture

of these hybrid sequences and standard on-policy rollouts using the GRPO objective:

$$\mathcal{J}_{\text{Prefix-RFT}}(\theta) = \mathcal{J}_{\text{GRPO}}(\theta | \{\tau_{\text{hybrid}}\} \cup \{\tau_{\text{on-policy}}\})\quad (19)$$

To prevent off-policy demonstration gradients from dominating the updates, Prefix-RFT incorporates an entropy-based clipping mechanism. This strategy zeroes out the advantages for all but the top- $k\%$ high-entropy demonstration tokens, thereby concentrating optimization on critical uncertain junctures and ensuring stable training.

C Training Configuration

C.1 SFT

We train SFT889K and all 102K variants with batch size of 512 and learning rate 1e-5 for 8 and 9 epochs, respectively. To study severe overfitting, we continue training Easy102K up to 6,120 steps (335.9 exaFLOPs). Checkpoints are saved every 360 steps (0.2 epochs for SFT889K, 1.8 epochs for 102K variants). For S1K, we follow the official setup: batch size 16, learning rate 1e-5, weight decay 1e-4, and train for 5 epochs, with checkpoints saved every 62 steps (1 epoch).

Specifically, for the SFT-then-RL experiments in Table 1, we initialize RL from the checkpoint with minimum SFT validation loss, as these checkpoints typically reside in the Stable Regime and are more likely to achieve the post-training ceiling for the given SFT data. Concretely, we use step 124 (2 epochs) for S1K, step 1800 (9 epochs) for all three 102K variants (Easy102K, Uniform102K, Hard102K), and step 14080 (7.8 epochs) for SFT889K.

C.2 RL Practice

All RL and Syn-SFT-RL runs optimize a binary correctness reward (1 if correct, 0 otherwise). This correctness is verified using a script powered by Math-Verify (Hugging Face, 2024). Furthermore, token-level loss aggregation is uniformly applied across all runs.

C.2.1 RL in Paradigms Comparison

We summarize the RL configuration for the training in § 6.1, including the Pure-RL (GRPO and DAPO_d), and Syn-SFT-RL (LUFFY, SRFT, UPT, Prefix-RFT). The shared training hyperparameters for Pure-RL and Syn-SFT-RL methods are summarized in Table 2. Unless specified, all algorithms use this default configuration.

Pure-RL. GRPO and DAPO_d serve as Pure-RL baselines. DAPO_d adds the dynamic difficulty sampling strategy (Yu et al., 2025) on GRPO. For DAPO_d, the dynamic difficulty sampling strategy employs a batch size of 128 responses per inference round, and the asymmetric clipping ratio strategy is not applied.

Table 2: Shared training hyperparameters for GRPO, DAPO_d, LUFFY, SRFT, UPT and Prefix-RFT in § 6.1.

Hyperparameter	Default value
Batch size	128
Update batch size	64
Rollout number	8
Max prompt length	1,024
Max response length	8,192
Learning rate	1e-6
Entropy coefficient	0.001
KL loss term	removed
Std. in group advantage norm	removed

Syn-SFT-RL. We adopt training configurations from the Unify-Post-Training codebase (Lv et al., 2025), with UPT utilizing a specific learning rate of 5e-6. For rollout generation, UPT adaptively allocates up to 8 trajectories between on-policy and off-policy samples. Conversely, LUFFY (Yan et al., 2025) and SRFT (Fu et al., 2025) maintain a fixed 7:1 ratio of on-policy to expert trajectories per prompt.

Following the similar setting, Prefix-RFT (Huang et al., 2025) generates 8 trajectories per prompt, comprising 7 standard on-policy rollouts (empty prefixes) and 1 hybrid rollout guided by a sampled expert prefix. Specifically, the expert prefix length is dynamically sampled via a Beta distribution, where the lower bound follows a cosine decay schedule from 95% to 5% of the demonstration length throughout training, while the upper bound remains constant at 95%. Additionally, Prefix-RFT incorporates an entropy-based clipping mechanism that zeroes out advantages for the 80% lowest-entropy demonstration tokens, concentrating optimization on high-uncertainty regions.

The maximum trajectory length for all Syn-SFT-RL algorithms is set to 8,192, as consistently adopted in prior works (Yan et al., 2025; Fu et al., 2025; Lv et al., 2025; Huang et al., 2025). Other shared hyperparameters are configured as detailed in Table 2.

C.2.2 RL in SFT-then-RL pipeline

Recognizing the superiority of DAPO_d when starting from an SFT checkpoint in § 6.1, we further improve DAPO_d to DAPO_{dc} as the RL method in the following SFT-then-RL experiments in § 6.2 and § 6.3. DAPO_{dc} adds the asymmetric clipping ratio strategy to DAPO_d, setting $(\epsilon_{\text{high}}, \epsilon_{\text{low}}) = (0.28, 0.2)$. In DAPO_{dc}, each inference round uses 128 responses for dynamic difficulty sampling. The batch size and update batch size are 64, the learning rate is 1e-6, and the rollout number is 8. Maximum lengths for prompt and response are 1,024 and 8,192, respectively. Furthermore, the entropy and KL term coefficients are 0, and the group advantage normalization is enabled.

D Evaluation Configuration

ID Evaluation During evaluation, we find that Math-Verify may fail to parse some semantically correct answers, leading to missed positives. To reduce this bias, we use two complementary evaluators: a *strict* evaluator based on Math-Verify alone, and a *lenient* evaluator that additionally allows a fuzzy string match when symbolic verification fails. For instances on which the two evaluators disagree, we further use an LLM judge with the prompt in Table 7 for adjudication. The final reported results are based on this two-stage verification procedure.

OOD Evaluation Manual inspection of model outputs on GPQA-D and ARC-C reveals that models often generate correct answers but fail to adhere to a standardized format. Taking the Student Answer in Table 7 as an example, the answer is semantically correct but not exactly matches the ground truth. This hinders automatic answer parsing and evaluation.

To address this, we use DeepSeek-V3 (DeepSeek-AI et al., 2025a) as a judge to verify correctness. We validate this approach by manually auditing 50 random samples and observe that the LLM judgments are perfectly consistent with human evaluation. This confirms the judge’s reliability for these tasks. For the DeepSeek-V3 content generation, we use a temperature of 0.6 and a top-p of 0.95, with a maximum response length of 8,192 tokens. An evaluation example is detailed in Table 7.

Table 3: **Statistical summary of the constructed SFT datasets.** It details average prompt/response lengths, logical step counts (derived via semantic analysis in Appx. G.1), and Win Rates (WR) across different DeepSeek model sizes. These metrics confirm the intended difficulty stratification, distinguishing the complexity levels of Easy, Uniform, and Hard subsets. **Note:** Avg. logical steps are computed on 10K randomly sampled instances, while other metrics are computed on the full dataset.

Dataset	Avg. prompt length	Avg. response length	Avg. logical steps	WR (1.5B)	WR (7B)	WR (671B)
Easy102K	64	2253	10	1	0.98	0.99
Uniform102K	74	3673	-	0.79	0.84	0.87
Hard102K	101	8532	13	0.09	0.30	0.41
SFT889K	78	3693	-	-	-	-
S1K	127	9884	-	-	-	-
Uniform220K	74	3674	-	-	-	-
Uniform440K	74	3671	-	-	-	-

E Compute Estimation

We adopt **F**loating-point **O**perations (FLOPs) as our computational metric because it is hardware-agnostic and parallelization-agnostic, depending only on model architecture and sequence lengths during training. We employ the FlopsCounter code (Engine, 2023) of the Verl framework (Sheng et al., 2024) for estimation. For SFT, FLOPs are estimated based on the sequence length of the SFT dataset; For RL and Syn-SFT-RL, we dynamically compute FLOPs using real-time prompt and response lengths recorded in TensorBoard (Abadi et al., 2016) logs. During training, both forward and backward cost the computation.

Forward FLOPs Per-Token Estimation. The theoretical forward FLOPs per token is denoted as $\mathcal{F}_{\text{forward_token}}$, based on the model configuration and average sequence length S . Let L be the number of layers, H the hidden size, H_{ff} the intermediate size of the feed-forward network, and V the vocabulary size. For the attention mechanism, we define D_{KV} as the total dimension of the Key and Value heads, accounting for Grouped Query Attention (GQA) (Ainslie et al., 2023).

First, we define the parameter counts for the constituent dense components. The MLP block, which utilizes a SwiGLU activation function with three linear projections (gate, up, and down), has a parameter count P_{MLP} . The linear projections in the attention layer (comprising W_Q, W_K, W_V, W_O) contribute $P_{\text{attn_linear}}$. The embedding layer and the language model head share the vocabulary-dim parameters, denoted as P_{vocab} . These are formulated as:

$$\begin{aligned}
 P_{\text{MLP}} &= 3HH_{ff} \\
 P_{\text{attn_linear}} &= H(H + 2D_{KV} + H) = 2H(H + D_{KV}) \\
 P_{\text{vocab}} &= 2VH
 \end{aligned}
 \tag{20}$$

The total FLOPs consists of the dense computation part ($\mathcal{F}_{\text{dense}}$) and the attention score computation part ($\mathcal{F}_{\text{attn_core}}$). The dense part aggregates the parameters from all L layers and the vocabulary projections, multiplied by a factor of 2 (for multiply-accumulate operations). The attention core part depends linearly on the sequence length S . The final estimation is given by:

$$\begin{aligned}
 \mathcal{F}_{\text{dense}} &= 2 \cdot [L \cdot (P_{\text{MLP}} + P_{\text{attn_linear}}) + P_{\text{vocab}}] \\
 \mathcal{F}_{\text{attn_core}} &= 4 \cdot S \cdot L \cdot H \\
 \mathcal{F}_{\text{forward_token}} &= \mathcal{F}_{\text{dense}} + \mathcal{F}_{\text{attn_core}}
 \end{aligned}
 \tag{21}$$

Backward FLOPs Per-Token Estimation. According to Narayanan et al. (2021) and Kaplan et al. (2020), the theoretical backward FLOPs per token is approximately two times that of forward. Let $\mathcal{F}_{\text{forward_token}}$ be the theoretical backward FLOPs per token:

$$\mathcal{F}_{\text{backward_token}} = 2 \cdot \mathcal{F}_{\text{forward_token}}
 \tag{22}$$

E.1 SFT Per-step Estimation

Per-step SFT accounts for one forward and one backward pass per step. Let B denote the batch size (number of sequences), S the average sequence length used for fine-tuning.

The total number of tokens processed during SFT per step is given by $T_{\text{total}} = B \cdot S$. Since the backward pass requires approximately twice the FLOPs of the forward pass, the total FLOPs per token during training is $3 \cdot \mathcal{F}_{\text{forward_token}}$. Therefore, the total computational cost for SFT, denoted as \mathcal{F}_{SFT} , is calculated as:

$$\begin{aligned}
\mathcal{F}_{\text{train_token}} &= \mathcal{F}_{\text{forward_token}} + \mathcal{F}_{\text{backward_token}} \\
&= 3 \cdot \mathcal{F}_{\text{forward_token}} \\
\mathcal{F}_{\text{SFT}} &= B \cdot S \cdot \mathcal{F}_{\text{train_token}} \\
&= 3 \cdot B \cdot S \cdot \mathcal{F}_{\text{forward_token}}
\end{aligned} \tag{23}$$

E.2 RL Per-step Estimation

DAPO. For DAPO, the computational cost per step is divided into a Generation Phase (dynamic sampling) and a Training Phase (actor update). Let B_{gen} denote the generation batch size, K the number of dynamic sampling iterations, and G the number of responses per prompt (i.e., group size). In the generation phase, the model explores a large solution space by generating $K \cdot B_{\text{gen}} \cdot G$ sequences. Since this phase involves only inference, the cost is purely forward FLOPs.

In the training phase, a subset of data (removing all correct and wrong trajectories) is selected, denoted by the training batch size B_{train} (where $B_{\text{train}} < K \cdot B_{\text{gen}}$). The update step involves one forward pass to compute new log-probs and one backward pass. Following standard estimation, the combined update cost (forward + backward) is approximately 3 times the forward cost per token (Kaplan et al., 2020).

Given the total sequence length $S = S_{\text{prompt}} + S_{\text{response}}$, the FLOPs for one DAPO step are estimated as:

$$\begin{aligned}
\mathcal{F}_{\text{gen}} &= (K \cdot B_{\text{gen}} \cdot G) \cdot S \cdot \mathcal{F}_{\text{forward_token}} \\
\mathcal{F}_{\text{train}} &= (B_{\text{train}} \cdot G) \cdot S \cdot 3 \cdot \mathcal{F}_{\text{forward_token}} \\
\mathcal{F}_{\text{DAPO}} &= \mathcal{F}_{\text{gen}} + \mathcal{F}_{\text{train}} \\
&= (K \cdot B_{\text{gen}} + 3 \cdot B_{\text{train}}) \cdot G \cdot S \cdot \mathcal{F}_{\text{forward_token}}
\end{aligned} \tag{24}$$

GRPO. The algorithm serves as the baseline where no dynamic difficulty sampling is performed. In this setting, the generation batch size equals the training batch size ($B_{\text{gen}} = B_{\text{train}} = B$) and sampling is performed once ($K = 1$). The model generates responses for all prompts in the batch and updates on all of them. Thus, the FLOPs estimation simplifies to:

$$\begin{aligned}
\mathcal{F}_{\text{gen}} &= (1 \cdot B \cdot G) \cdot S \cdot \mathcal{F}_{\text{forward_token}} \\
\mathcal{F}_{\text{train}} &= (B \cdot G) \cdot S \cdot 3 \cdot \mathcal{F}_{\text{forward_token}} \\
\mathcal{F}_{\text{GRPO}} &= \mathcal{F}_{\text{gen}} + \mathcal{F}_{\text{train}} \\
&= 4 \cdot B \cdot G \cdot S \cdot \mathcal{F}_{\text{forward_token}}
\end{aligned} \tag{25}$$

E.3 Syn-SFT-RL Per-step Estimation

LUFFY and SRFT. Both LUFFY and SRFT integrate expert demonstrations into the RL optimization loop. Let G denote the number of on-policy sampled trajectories (group size) and N denote the number of expert trajectories per prompt.

In the Generation Phase, the model generates G responses for each prompt in the batch B . In the Training Phase, the model updates parameters using both the on-policy generated data and the off-policy expert data. Thus, the effective training batch size per prompt becomes $G + N$ ($G = 7, N = 1$ in § C.2.1).

Given the real-time recorded average on/off-policy sequence length S_{on} and S_{off} , the FLOPs for LUFFY and SRFT are calculated as the sum of inference cost on G samples and update cost on $G + N$ samples:

$$\begin{aligned}
\mathcal{F}_{\text{gen}} &= [(B \cdot G) \cdot S_{\text{on}} + (B \cdot N) \cdot S_{\text{off}}] \cdot \mathcal{F}_{\text{forward_token}} \\
\mathcal{F}_{\text{train}} &= 3[(B \cdot G) \cdot S_{\text{on}} + (B \cdot N) \cdot S_{\text{off}}] \cdot \mathcal{F}_{\text{forward_token}} \\
\mathcal{F}_{\text{Hybrid}} &= \mathcal{F}_{\text{gen}} + \mathcal{F}_{\text{train}} \\
&= 4[(B \cdot G) \cdot S_{\text{on}} + (B \cdot N) \cdot S_{\text{off}}] \cdot \mathcal{F}_{\text{forward_token}}
\end{aligned} \tag{26}$$

Note that for SRFT, although it computes multiple loss terms (Eq. 16), the dominant computational overhead remains the forward and backward passes through the transformer backbone on the combined data tokens ($G + N$), making this estimation applicable to both algorithms.

UPT. The per-step FLOPs of UPT are estimated dynamically based on the actual composition of the training batch, which consists of N_{on} on-policy samples processed via GRPO and N_{off} expert samples processed via SFT.

Let S_{on} be the average on-policy sequence length. The algorithm processes $G \cdot S_{\text{on}}$ tokens during the Generation Phase. In this phase, the computational cost is given by:

$$\mathcal{F}_{\text{gen}} = G \cdot S_{\text{on}} \cdot \mathcal{F}_{\text{forward_token}} \tag{27}$$

Subsequently, the algorithm filters samples based on difficulty, retaining N_{on} on-policy samples and N_{off} off-policy samples per batch. Consequently, the FLOPs consumption during the Training Phase is:

$$\mathcal{F}_{\text{train}} = 3 \cdot (N_{\text{on}} \cdot S_{\text{on}} + N_{\text{off}} \cdot S_{\text{off}}) \cdot \mathcal{F}_{\text{forward_token}} \tag{28}$$

Therefore, the total computational cost for a single UPT step is formulated as:

$$\begin{aligned}
\mathcal{F}_{\text{UPT}} &= \mathcal{F}_{\text{gen}} + \mathcal{F}_{\text{train}} \\
&= [G \cdot S_{\text{on}} + 3 \cdot (N_{\text{on}} \cdot S_{\text{on}} + N_{\text{off}} \cdot S_{\text{off}})] \cdot \mathcal{F}_{\text{forward_token}}
\end{aligned} \tag{29}$$

Prefix-RFT. Since Prefix-RFT only borrows expert trajectory prefixes as a curriculum during RL training, its computational cost is otherwise identical to GRPO. We thus adopt the same FLOPs estimation method as in Eq. 25.

F Robust Curve Fitting

In § 4, we primarily model the RL scaling dynamics using sigmoidal curves (Ruan et al., 2024; Khatri et al., 2025), which effectively characterize the behavior of most RL and Syn-SFT-RL runs. However, for unstable training instances that defy sigmoidal fitting, such as the SRFT case shown in Figure 2 (Left), we adopt a robust fallback strategy. Specifically, we define the post-training ceiling for these instances as the peak performance observed either within the allocated RL compute budget or prior to performance collapse.

For the valid scaling instances, simply applying standard regression is often insufficient due to inherent training noise. To accurately model the relationship between computational investment (FLOPs) and model performance, we employ a robust curve-fitting pipeline. This pipeline integrates an iterative outlier detection mechanism based on Modified Z-scores with a Least Trimmed Squares (LTS) regression optimization.

F.1 Data Formulation

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ denote the dataset, where x_i represents cumulative FLOPs and y_i represents the evaluation metric. The data is partitioned into a training set $\mathcal{D}_{\text{train}}$ (first N_{fit} points) and a held-out validation set \mathcal{D}_{val} . The train-validation split may vary due to differences in training convergence. In most experimental runs, approximately 85% of the data is typically allocated for training, with the remaining 15% reserved for validation.

F.2 Robust Estimation Algorithm

Standard least-squares estimation is highly sensitive to anomalies. To derive a scaling law that reflects the consistent signal rather than transient noise, we employ a hierarchical robust optimization framework that integrates iterative statistical filtering (Modified Z-score) with subset-based optimization (Least Trimmed Squares) to isolate the true performance signal, ensuring that the derived scaling laws are predictive and generalizable across different compute regimes.

Stage-1: Coarse Outlier Rejection (Modified Z-Score). First, we filter gross statistical anomalies. In each iteration, we compute residuals $r_i = y_i - f(x_i; \theta)$ and the median residual $\tilde{r} = \text{median}(\mathbf{r})$. To quantify deviation robustly, we calculate the Median Absolute Devi-

ation (MAD) (Huber and Ronchetti, 2011; Leys et al., 2013):

$$\text{MAD} = \text{median}(|r_i - \tilde{r}|) \quad (30)$$

Subsequently, the Modified Z-score M_i is computed as (Iglewicz and Hoaglin, 1993):

$$M_i = \frac{0.6745 \cdot (r_i - \text{median}(r))}{\text{MAD}} \quad (31)$$

Points with $|M_i| > \tau$ are removed from the active training split, where τ is the **outlier threshold**. The factor 0.6745 normalizes the score such that it is consistent with the standard deviation under a normal distribution, while the use of MAD ensures resilience against extreme values that would skew a standard variance calculation.

Stage-2: Least Trimmed Squares Regression.

To further refine the model against subsets of data that may distort the global trend, we employ Least Trimmed Squares (LTS) (Rousseeuw, 1984; Rousseeuw and Leroy, 1987) regression. Instead of minimizing the sum of all residuals, LTS regression minimizes only the smallest h squared residuals:

$$\hat{\theta}_{\text{LTS}} = \arg \min_{\theta} \sum_{j=1}^h (r^2)_{(j)}(\theta) \quad (32)$$

where $(r^2)_{(1)} \leq \dots \leq (r^2)_{(N_{\text{fit}})}$ are the ordered squared residuals over the training set, and $h = \lfloor N_{\text{fit}} \cdot \alpha \rfloor$ is determined by the parameter α . We define $H^{(k+1)}$ as the set of indices corresponding to the h smallest squared residuals, i.e., $H^{(k+1)} = \{i \mid r_i^2 \leq (r^2)_{(h)}\}$.

We optimize this objective using the Concentration Step (C-step) algorithm (Rousseeuw and Driessen, 2006), which proceeds iteratively as follows:

- **Estimation:** Compute squared residuals $r_i^2 = (y_i - f(x_i; \theta^{(k)}))^2$ for all N_{fit} training points using the current parameters $\theta^{(k)}$.
- **Selection:** Identify the index set $H^{(k+1)}$ corresponding to the h smallest squared residuals.
- **Update:** Update parameters to $\theta^{(k+1)}$ by fitting the model strictly to the data points indexed by $H^{(k+1)}$.
- **Convergence:** Repeat the process until the parameter estimate θ stabilizes.

F.3 Fitting Results and Analysis.

F.3.1 Fitting Results

Across all fitting instances, the pipeline achieves an average Root Mean Square Error (RMSE) of approximately 0.90 on the validation split and an average R^2 of 0.90 on the training split, confirming the overall robustness of the fits.

F.3.2 Fitting Analysis

We visualize the resulting SFT-then-RL scaling curves in Figure 8 and provide the exhaustive parameter estimates in Table 4. This table systematically reports the evolution of key scaling metrics across various SFT data configurations relative to the progress of SFT compute x_{sft} . Specifically, we characterize the realized SFT performance (P_{sft}), the RL plasticity (PL_{rl}), the RL convergence midpoint ($C_{\text{mid}_{\text{rl}}}$), the RL scaling steepness (B_{rl}), and the ultimate post-training ceiling (A_{post}). These results facilitate a detailed analysis of how the SFTed initial point impacts the subsequent RL trend and the final performance limits of the model.

Impact of SFT Compute on RL Plasticity. As illustrated in the PL_{rl} column of Table 4, RL plasticity exhibits an overall downward trend, albeit with minor oscillations, as SFT compute x_{sft} increases. Consequently, allocating minimal compute to SFT generally preserves greater plasticity and yields a larger headroom for subsequent RL-driven improvements. Specifically, across the S1K, Easy102K, Uniform102K, and SFT889K configurations, the lowest SFT compute consistently achieves the maximum PL_{rl} . Although Hard102K presents a minor deviation, its earliest SFT checkpoint still retains the second-highest plasticity in that category.

We attribute this general decline to the inherent imitation nature of SFT. The SFT progress deeply entrenches the policy within the demonstration distribution, prematurely exhausting the exploratory capacity required for effective reward-driven optimization (Wang et al., 2026). This hypothesis is empirically corroborated by the Pure-RL baseline. By entirely bypassing the SFT phase, the Pure-RL instance retains maximum exploratory freedom and achieves the highest absolute plasticity of 25.3, outperforming all SFT-then-RL configurations. However, this plasticity advantage is ultimately offset by its low initial performance ($P_{\text{sft}} = 46.7$), which severely restricts the final

post-training ceiling (A_{post}).

Impact of SFT Compute on RL Training Efficiency. We quantify RL training efficiency using the convergence midpoint $C_{\text{mid}_{\text{rl}}}$, which measures the amount of compute required to achieve half of the total potential gain. Lower values of $C_{\text{mid}_{\text{rl}}}$ indicate faster convergence. However, $C_{\text{mid}_{\text{rl}}}$ does not monotonically decrease with increasing x_{sft} , instead exhibiting substantial fluctuations.

As shown in Table 4, the most efficient RL runs, those with the lowest $C_{\text{mid}_{\text{rl}}}$, typically correspond to relative less allocated SFT compute x_{sft} . Notably, the Pure-RL baseline achieves the lowest $C_{\text{mid}_{\text{rl}}}$ of 1 exaFLOPs, representing the fastest convergence among all evaluated scenarios. This phenomenon is broadly consistent with prior observations regarding the interplay between SFT and subsequent RLHF training (Ouyang et al., 2022). It may be attributed to the fact that relatively lower SFT compute preserves more diverse behaviors in the action space, thereby facilitating exploration and accelerating RL convergence.

G Dataset Curation

We summarize the key characteristics of our SFT data in Table 3.

G.1 Expert Trajectory Collection

We curate high-quality reasoning trajectories from two large-scale datasets: AM-DeepSeek-R1-Distilled-1.4M (Zhao et al., 2025) and AM-DeepSeek-Distilled-40M (Tian et al., 2025). To ensure data distribution consistency and quality, we retain only mathematics-domain data, select trajectories distilled from DeepSeek-R1-671B (DeepSeek-AI, 2025) to unify trajectory style, and perform deduplication based on prompt matching. The resulting filtered datasets are denoted as amthink-1.4m and amthink-40m. From these sources, we construct multiple datasets for our experiments: SFT889K, Uniform102K, Easy102K, and Hard102K for SFT training, and Val-199 for SFT validation. A prompt-trajectory example is shown in Table 8.

Difficulty Classification. To understand the influence of SFT data difficulty on post-training outcomes, we extract data of varying difficulty levels from amthink-40m. We use the Win Rate (WR), defined as the ratio of successful attempts S to total attempts N , as a proxy for problem difficulty. We derive WRs with $N=4$ attempts across

Table 4: **Scaling dynamics across SFT-then-RL configurations.** The first row represents the Pure-RL baseline. x_{sft} denotes cumulative SFT compute (exaFLOPs). **Outlier Threshold** is the Modified Z-score cutoff τ for Stage-1 outlier rejection (points with $|M_i| > \tau$ are removed; Appx. F.2). Use-LTS denotes whether the Least Trimmed Squares (LTS) regression in Appx. F.2 is used during curve fitting. In § 4.2 we define the alignment offset ϵ , PL_{rl} , A_{post} , and the midpoint parameter C_{mid_d} in the RL scaling law (Eq. 2). For PL_{rl} , B_{rl} , P_{sft} , and A_{post} , higher values are better; for C_{mid_d} , lower values are better. The best values within each SFT configuration are bolded.

SFT data	SFT Step	x_{sft}	Outlier Threshold	Use-LTS	ϵ	PL_{rl}	C_{mid_d}	B_{rl}	P_{sft}	A_{post}
-	0	0	2.5	False	0	25.3	1	1.2	46.7	77.3
S1K	62	0.6	2.5	False	4.4	8.7	107	8.5	68.6	77.0
	124	1.1	3	True ($\alpha=0.85$)	0	6.0	44	0.9	70.9	76.9
	186	1.7	2.5	False	0	4.0	99	1.1	72.9	76.5
	248	2.3	2.5	True ($\alpha=0.75$)	0	2.5	93	2.8	74.1	76.4
	310	2.9	3	True ($\alpha=0.75$)	1.5	3.3	134	11.2	73.1	75.9
Easy102K	360	19.8	2.5	False	0	6.0	22	1.0	70.7	76.7
	720	39.5	2.5	False	0	5.3	66	0.8	73.3	78.5
	1080	59.3	2.5	False	0	4.6	135	0.4	74.2	78.8
	1440	79.0	2.5	True ($\alpha=0.75$)	0	2.0	41	2.0	74.8	76.8
	1800	98.8	2.5	False	0	3.4	43	0.3	74.4	77.8
Uniform102K	360	34.7	2.5	False	0	4.8	32	1.8	72.2	76.9
	720	69.3	3	True ($\alpha=0.75$)	0	3.0	28	1.4	74.5	77.6
	1080	104.0	2.5	False	1.5	4.8	118	6.6	74.8	79.7
	1440	138.7	2.5	False	0	1.8	46	2.2	75.7	77.5
	1800	173.4	2.5	True ($\alpha=0.75$)	0	3.8	161	0.7	75.5	79.3
Hard102K	360	89.3	4	False	0	4.9	40	0.9	73.2	78.1
	720	178.5	4	True ($\alpha=0.85$)	0	3.1	32	1.7	74.7	77.8
	1080	267.8	3	True ($\alpha=0.85$)	0	6.7	61	3.5	76.2	82.8
	1440	357.0	2.5	True ($\alpha=0.85$)	0	3.4	48	5.5	76.9	80.3
	1800	446.3	2.5	False	0	2.7	73	1.0	76.1	78.7
SFT889K	360	34.9	2.5	False	0	13.4	10	0.7	69.2	82.5
	720	69.8	3	True ($\alpha=0.75$)	0	8.8	8	1.0	71.9	80.7
	1080	104.8	4	True ($\alpha=0.75$)	0	7.1	10	1.5	73.0	80.1
	1440	139.7	4	True ($\alpha=0.75$)	0	7.0	9	1.7	73.3	80.3
	1800	174.6	3	True ($\alpha=0.75$)	0	6.9	9	1.6	74.1	81.0
	3600	349.2	4	False	0	6.6	13	1.6	76.5	83.1
	5400	523.8	2.5	False	0	5.3	13	2.2	77.5	82.8
	7200	698.4	2.5	True ($\alpha=0.75$)	0	6.7	16	1.6	77.3	84.0
	9000	873.0	2.5	False	0	6.7	13	1.8	76.7	83.4
	10800	1047.6	2.5	True ($\alpha=0.85$)	0	5.4	12	2.5	77.7	83.1
	12600	1222.2	3	True ($\alpha=0.75$)	0	5.9	10	2.2	77.1	83.0
	14080	1365.8	2.5	True ($\alpha=0.75$)	0	7.8	11	1.1	76.2	84.0

three models: DeepSeek-R1-Distill-1.5B, 7B, and DeepSeek-R1-671B (DeepSeek-AI, 2025). Based on WRs from the 1.5B model, problems are classified as Easy (WR=1.0) or Hard (WR \leq 0.25). Consequently, we construct Easy102K and Hard102K, each containing 102K samples from their respective pools, alongside Uniform102K, a randomly sampled neutral-difficulty baseline.

To verify that our WR-based classification captures genuine reasoning complexity rather than mere obscurity, we conduct a supplementary semantic analysis. We randomly sample 10,000 prompts from both Easy102K and Hard102K, employing DeepSeek-V3.2 (DeepSeek-AI et al., 2025b) as an expert evaluator to decompose solution trajectories into distinct **logical steps** while filtering out conversational redundancy. The evaluator follows the fixed instruction prompt in Table 9. Manual verification of 100 sampled outputs per dataset confirms that this automated step-counting aligns closely with human judgment.

As shown in Table 8, trajectories in Hard102K contain an average of 13 logical steps, representing a 30% increase over the 10 steps averaged in Easy102K. This reasoning depth is directly mirrored in trajectory length: Hard102K responses average 8,532 tokens, nearly four times the 2,253 tokens typical of Easy102K. This concurrent increase in both logical steps and trajectory length empirically validates WR as a robust and reliable proxy for problem difficulty in our study.

G.2 RL Data

We curate RL62K (62.3K prompts) from Skywork-OR1-RL (He et al., 2025) by filtering out extreme difficulty levels and prompts containing Chinese characters. For Syn-SFT-RL, we construct MIX37K (36.7K samples) by augmenting these prompts with matched expert trajectories from SFT889K, excluding sequences exceeding 8,192 tokens as suggested by (Yan et al., 2025) to ensure the complete trajectory utilization in each update step. Crucially, this data scale is sufficient, as our experiments show that RL variants typically reach saturation or instability before exhausting a single epoch over MIX37K.

H Complementary Results

H.1 Paradigm Comparison on SFT889K

We compare various expert trajectory utilization paradigms by training on SFT889K. In this setting, we set the batch size to 64 and maximum response

length to 12,000, with RL capped at 400 steps (see Appx. C for details).

As shown in Figure 6 (Left), when starting from the base model, both RL and Syn-SFT-RL methods exhibit premature convergence within 40 exaFLOPs. DAPO_d and LUFFY reach a ceiling (A_{post}) of 71.0 approximately, slightly lower than their MIX37K counterparts (74.3 and 72.7 in Figure 2). However, UPT shows significant scaling gains on this larger dataset (68.6 vs. 60.0), and SRFT demonstrates improved stability compared to MIX37K.

When initialized from a saturated SFT checkpoint (Figure 6, Right), GRPO collapses rapidly, whereas DAPO_d scales steadily to 78.1. This not only outperforms the SFT peak (76.9) by 1.2 points but also surpasses all other baselines. These results highlight the necessity of DAPO_d's dynamic difficulty sampling for stability and emphasize the SFT-then-RL pipeline as the superior paradigm for pushing post-training frontiers.

H.2 SFT Data Scale Impact

In § 6.2.2, we observe a distinct stratification in the SFT-then-RL post-training performance ceilings across three data scales: S1K, Uniform102K, and SFT889K, suggesting that a larger scale begets a higher post-training ceiling.

To further investigate how the SFT volume dictates the ceiling, we introduce Uniform220K and Uniform440K, SFT datasets comprising 220K and 440K prompt-trajectory pairs uniformly drawn from amthink-40m. We maintain hyperparameter consistency with the Uniform102K setup (see § C.1), and we save checkpoints every 215 steps (0.5 epochs) for Uniform220K and 360 steps (0.4 epochs) for Uniform440K. Using validation loss as a guide, we select three checkpoints across distinct SFT regime for each scale to initialize the subsequent RL stage.

Table 5: The relation between the highest A_{post} and the minimum validation loss across different SFT data scales. Increasing the training data scale of SFT leads to a lower minimal validation loss, which indicates a higher post-training ceiling A_{post} .

SFT data	Highest A_{post}	Min. Val Loss
S1K	77.3	0.51
Uniform102K	79.7	0.42
Uniform220K	82.6	0.41
Uniform440K	82.8	0.39
SFT889K	84.0	0.37

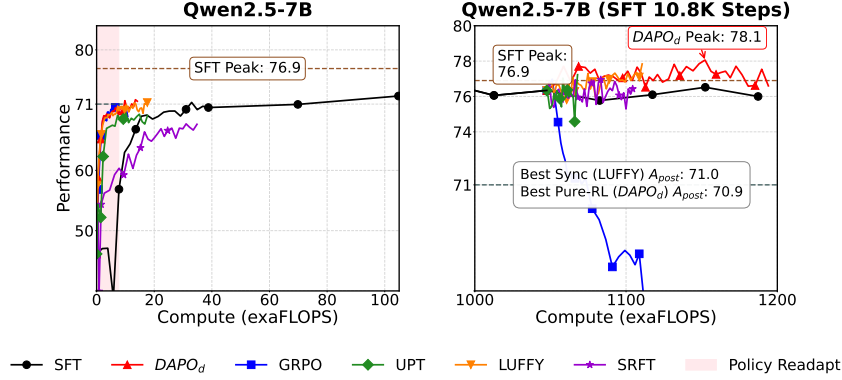


Figure 6: **The training dynamic comparison of post-training paradigms on the SFT889K dataset, covering different initializations. (Left)** Initializing from Qwen2.5-7B, where the all RL and Syn-SFT-RL methods suffer from the prematurely convergence. **(Right)** Initializing from a saturated SFT checkpoint (10,800 steps on Qwen2.5-7B), where DAPO_d outperforms other methods, and GRPO suffers from the performance collapse.

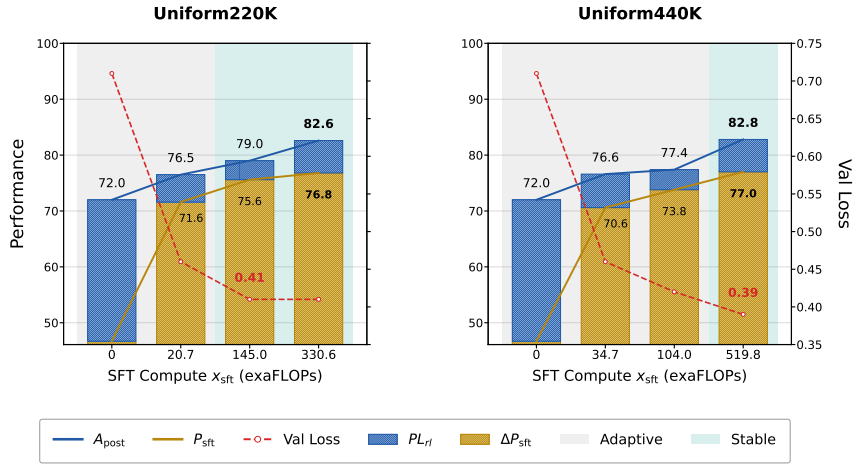


Figure 7: **SFT compute scaling dynamics of the SFT-then-RL pipeline across different scale of the SFT data. (Left)** Performing SFT on Uniform220K, selecting the 215-step (20.7 exaFLOPs), 1505-step (145.0 exaFLOPs) and 3432-step (330.6 exaFLOPs) SFTed checkpoints to initialize the RL training. **(Right)** Performing SFT on Uniform440K, selecting the 360-step (34.7 exaFLOPs), 1080-step (104.0 exaFLOPs) and 5400-step (519.8 exaFLOPs) SFTed checkpoints to initialize the RL training.

As Figure 7 illustrates, initializing RL from the Stable Regime consistently yields the optimal performance ceiling on both datasets. This success stems from a synergy between superior SFT performance and preserved RL plasticity, validating the significance of **training SFT to saturation**. As shown in Table 5, the highest A_{post} continues to trend upward as data scale increases, confirming that the **larger scale, higher ceiling** principle remains robust. Furthermore, we confirm that **the minimum validation loss still serves as a robust indicator for pursuing a higher A_{post}** . Even as the SFT data scale expands, the minimum validation loss remains strongly correlated with the highest A_{post} of each specified SFT dataset, maintaining a Pearson correlation coefficient of -0.94, and

a Spearman correlation coefficient of -1.00.

H.3 The Optimal SFT-to-RL Transition Timing on Larger Model

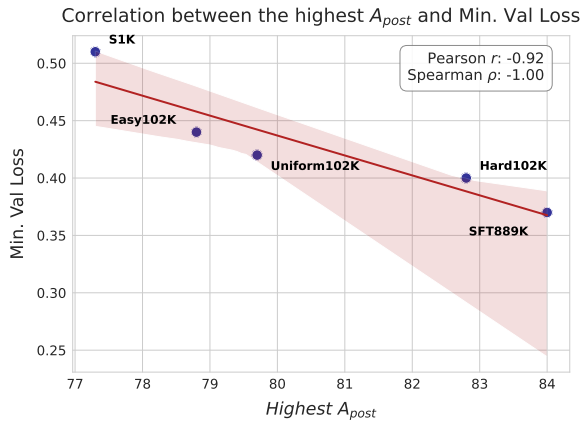
In this section, we validate whether the principle of training the SFT phase to saturation still holds for larger models. We construct a training set of 55K uniformly sampled instances from the amthink-40m dataset, denoted as Uniform55K and perform SFT-then-RL experiments on Qwen2.5-14B (Qwen et al., 2025). We follow the SFT-then-RL configuration used for Qwen2.5-7B + SFT889K (detailed in Appx. C.1 and C.2.1), and fine-tune the Qwen2.5-14B for four epochs. To evaluate the transition timing, we selected SFT checkpoints from various convergence regimes to initialize the subsequent RL phase. The subse-

Table 6: **Qwen2.5-14B on Uniform55K**: SFT-to-RL transition points. Columns report cumulative SFT compute x_{sft} (exaFLOPs), validation loss, SFT regime, realized SFT performance (P_{sft}), maximum RL improvement over the SFT checkpoint, and peak post-training performance ($\max P_{\text{post}}$). Boldface marks the strongest outcome within each column where a single optimum is intended.

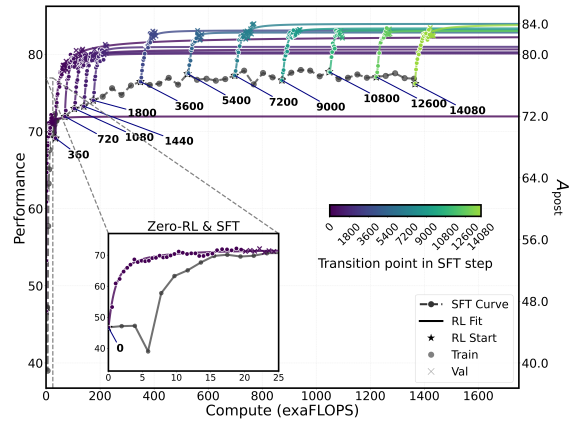
SFT-RL switch	x_{sft}	Val loss	SFT regime	P_{sft}	Max RL improve.	$\max P_{\text{post}}$
0-step (zero-RL)	0	0.67	Adaptive	59.8	14.7	74.5
172-step	32.7	0.44	Adaptive	75.7	3.1	78.8
1720-step	326.6	0.39	Stable	78.6	1.2	79.8

quent RL is trained for 200 steps.

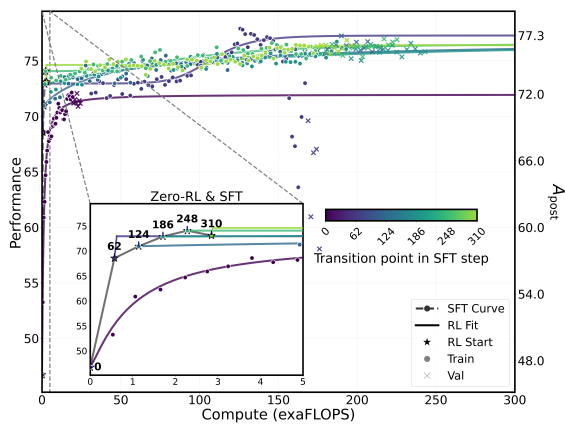
The established SFT foundation (P_{sft}) and the peak post-training performance ($\max P_{\text{post}}$) are summarized in Table 6. Consistent with our observations on Qwen2.5-7B in Figure 3 and Llama3.2-3B in Figure 5, the principle of training the SFT foundation to saturation holds firmly at the 14B scale. We observe that transitioning to RL from the 1720-step checkpoint, which is located in the stable regime, achieve the best overall performance ($\max P_{\text{post}} = 79.8$). This optimal ceiling is primarily attributed to the robust realized SFT performance of 78.6. In contrast, while initiating RL from a zero-shot foundation results in the highest absolute maximum RL improvement of 14.7 points, its weak initial foundation of 59.8 prevented it from reaching a comparable end-to-end performance.



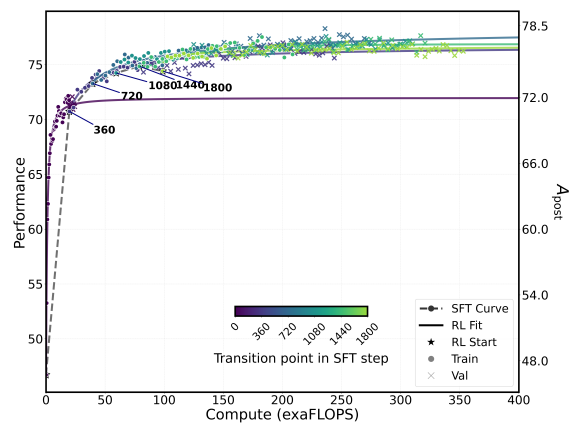
(a) Correlation Analysis



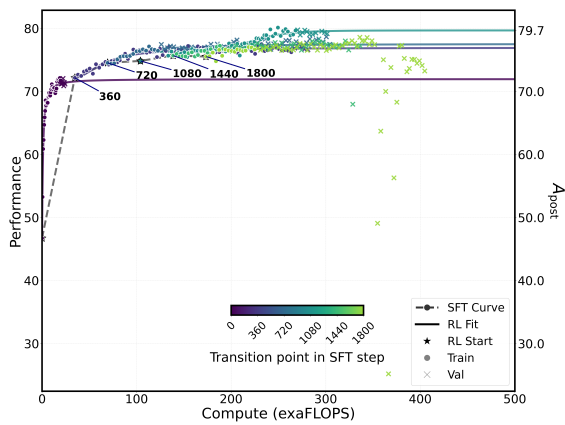
(b) SFT889K



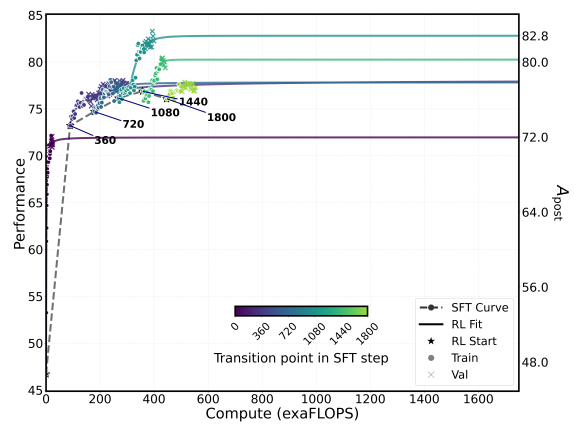
(c) S1K



(d) Easy102K



(e) Uniform102K



(f) Hard102K

Figure 8: **Visualization of SFT-then-RL fitting across different SFT data configurations.** (a) Correlation analysis between the highest A_{post} and the minimum validation loss of each specified SFT dataset. (b)-(f) The SFT-then-RL scaling dynamics under various data configurations. The SFT trajectory is depicted by a black dashed line. RL scaling curves initiated from different SFT steps are distinguished by a color gradient, where lighter shades indicate a higher number of SFT steps. The specific starting SFT step count for each RL curve is annotated in bold black text. Data points from the training split used for fitting the RL scaling curves are marked with solid circles, while those from the validation split used for assessing curve goodness-of-fit are marked with crosses. Magnified views are provided for the low-compute regions of SFT889K and S1K.

System Prompt:

You are a teacher grading student assignments. I will provide you with the question, the student's answer, and the ground truth for the question. Please carefully review each question and grade the student's answer based on the ground truth. If the student's answer matches the ground truth, reply with True; otherwise, reply with False. You need not provide any additional information—only reply with True or False. Your grading response must strictly follow the format: 'Grade: '.

Example:

Question: $1+1 = _$. Please choose the correct option from the following options. A.2, B.3, C.4

Student answer: I think it is A.2

Ground truth: A

Grade: True

User Prompt:

Question:

Which change is the best example of a physical change?

1) a cookie baking 2) paper burning 3) ice cream melting 4) a nail rusting

Student answer:

Okay, so I need to figure out which of these options is a physical change. Let me start by recalling what a physical change is. From what I learned, a physical change is when the form or state of a substance changes, but the chemical composition remains the same. That means no new substances are formed. In contrast, a chemical change involves a chemical reaction where new substances are created.

Now let's look at each option one by one.

1) A cookie baking. Hmm, when you bake a cookie, the dough is transformed. The ingredients like flour, eggs, sugar, etc., are mixed and then heated. Baking usually involves chemical reactions, like the Maillard reaction, where new compounds are formed. So this is probably a chemical change.

2) Paper burning. Burning paper is definitely a chemical change. Fire requires fuel and oxygen, and when paper burns, it combines with oxygen to form ash, carbon dioxide, and other products. The original paper is no longer there; it's changed into different substances. So this is chemical.

3) Ice cream melting. Ice cream is frozen, and when it melts, it goes from solid to liquid. No chemical reactions are happening here. The molecules are just spreading out because the temperature increased, but the actual molecules of the ice cream (like water, fat, sugar) are the same. So this should be a physical change.

4) A nail rusting. Rusting is a chemical process. The iron in the nail reacts with oxygen and water in the air to form iron oxide. This creates a new substance, which is rust. So that's a chemical change.

So the answer should be option 3, ice cream melting. It's a physical change because the state changes without forming new chemicals.

Answer: 3) ice cream melting

Ground truth: 3

Grade:

DeepSeek-V3 Response:

True

Table 7: The example of using DeepSeek-V3 as a judge when evaluating the case of ARC-C. The student answer is given by the 10800-step SFTed checkpoint from Qwen-2.5-7B.

Question:

Let the set $A = \{1, 2, 3\}$, and the set $B = \{0, 1, 2\}$. Then, the number of elements in $A \cup B$ is:

A: 2; B: 3; C: 4; D: 5

Expert Trajectory:

<think> Okay, so I have this problem here about sets A and B. Let me read it again to make sure I understand. It says, "Let the set A be {1, 2, 3}, and the set B be {0, 1, 2}. Then, the number of elements in $A \cup B$ is (). The options are A: 2, B: 3, C: 4, D: 5." Hmm, union of A and B. Right, union means all the elements that are in either set A or set B or in both. So I need to combine them and remove any duplicates. Let me write down the sets first. Set A is {1, 2, 3}. So those are the numbers 1, 2, and 3. Set B is {0, 1, 2}. That has 0, 1, and 2. Now, if I combine these two sets, I should list all the elements from both, but make sure I don't count the same element more than once. Let me list them out. Starting with set A: 1, 2, 3. Then set B has 0, 1, 2. So adding those to the union, we already have 1, 2, 3 from A, and from B we have 0 which isn't in A. The 1 and 2 from B are already in A, so they don't need to be added again. So combining them, the union should have elements 0, 1, 2, 3. That's four elements. Wait, but let me check again. Let me list them all out: $A \cup B = \{0, 1, 2, 3\}$. So the elements are 0, 1, 2, 3. Let's count them: 0 is one, 1 is two, 2 is three, 3 is four. So that's four elements. So the number of elements is 4. The options are C: 4. So the answer should be C? Wait, but let me make sure I didn't miss anything. Let me recall the formula for the number of elements in a union: $|A \cup B| = |A| + |B| - |A \cap B|$. So the size of the union is the size of A plus the size of B minus the size of their intersection. Let's apply that. Set A has 3 elements: 1, 2, 3. Set B has 3 elements: 0, 1, 2. The intersection of A and B would be the elements common to both, which are 1 and 2. So $|A \cap B|$ is 2. Then $|A \cup B| = 3 + 3 - 2 = 4$. Yes, that matches what I got before. So that's confirming it. Therefore, the number of elements in $A \cup B$ is 4. So the answer is C. But wait, just to double-check, maybe I made a mistake in the intersection. Let me verify the intersection. $A \cap B$ is the set of elements that are in both A and B. A is {1, 2, 3}, B is {0, 1, 2}. So elements common to both are 1 and 2. So yes, two elements. So the formula gives $3 + 3 - 2 = 4$. That seems correct. And listing them out gives four elements. So I think that's right. Alternatively, maybe the answer is D: 5? Wait, maybe I miscounted. Let me list the elements again. If we take all elements from A and B: A has 1,2,3; B has 0,1,2. So combining all of them: 0,1,2,3. So that's four distinct elements. There's no 4 or anything else. So four elements. So answer is C:4. So yeah, I think that's the correct answer. The options are A:2, B:3, C:4, D:5. So the correct one is C. </think>

<answer>

Given the sets $A = \{1, 2, 3\}$ and $B = \{0, 1, 2\}$, we need to find the number of elements in $A \cup B$. First, we recall that the union of two sets A and B includes all elements that are in A, in B, or in both. Listing the elements of both sets and combining them while removing duplicates, we get: - Elements of A: 1, 2, 3 - Elements of B: 0, 1, 2 Combining these elements and removing duplicates, the union $A \cup B$ is $\{0, 1, 2, 3\}$. To verify, we use the formula for the number of elements in the union of two sets:

$$|A \cup B| = |A| + |B| - |A \cap B|$$

where $|A| = 3$, $|B| = 3$, and $|A \cap B| = 2$ (since the intersection $A \cap B$ is $\{1, 2\}$). Plugging in these values:

$$|A \cup B| = 3 + 3 - 2 = 4$$

Thus, the number of elements in $A \cup B$ is 4, which corresponds to option C.

C </answer>

Table 8: An example of a R1-style prompt-trajectory pair from SFT889K. The expert trajectory features a reflective thinking process within <think> and </think> tags, followed by a structured solution within <answer> and </answer> tags. The final outcome is boxed.

You are an expert mathematics evaluator. Your task is to analyze the reasoning complexity of a solution trajectory by breaking it down into distinct logical steps.

Problem:

{question}

Thinking Trajectory:

{think}

Task:

1. Analyze the trajectory above. Break down the reasoning process into a numbered list of distinct logical steps.
 - **A valid step** includes: a mathematical deduction, a calculation, a logical transition, setting up an equation, or verifying a condition.
 - **Exclude:** Simple restatements of the question, pure filler phrases (e.g., "Let me see", "Hmm"), or redundant self-corrections that do not advance the logic.
2. Count the total number of valid steps derived in your list.
3. Output the final count inside a box format.

Output Format:

[Your step-by-step analysis here...]

Total Logical Steps:

Table 9: Instruction prompt for logical-step segmentation. Placeholders {question} and {think} are filled at inference time with the problem text and the model's thinking trajectory, respectively.