Sample Learning State Guided Dynamic SFT–RL Integration Post-training

Zheng Yang[♠] **Yilin Liu**[♠] **Zhuoran Li**[♠] **Wenhao Gao**[♠] **Bingwei Li**[♠]
Haozhe Sun[♠] **Caixia Yuan**[♠] **Fan Wan**[♢] **Chen Wu**[♢] **Xiaojie Wang**[♠*]
[♠]Beijing University of Posts and Telecommunications
[♢]Central Research Institute, Tongfang Knowledge Network Digital Technology Co., Ltd.

Abstract

Integrating supervised fine-tuning (SFT) and reinforcement learning (RL) has become a common post-training paradigm for improving the reasoning capabilities of large language models. Existing methods typically rely on sequential SFT–RL pipelines or static loss weights, overlooking sample-level heterogeneity and evolving learning states. We propose Sample Learning State (SLS), a rollout-based characterization of each training sample using two complementary signals: sample mastery and exploration-trajectory dispersion. These signals estimate whether reward-driven updates are reliable for the core task and whether current rollouts provide informative contrast for RL optimization, respectively. Based on SLS, we design a state-aware coefficient that dynamically balances SFT and RL losses, increasing the RL weight only when the model demonstrates sufficient core-task competence and the sampled trajectories provide useful reward contrast. Experiments on Qwen2.5-Math models show that SLS achieves state-of-the-art results on four in-distribution and two out-of-distribution mathematical reasoning benchmarks. In multi-reward settings, SLS preserves core reasoning performance while mitigating reward hacking induced by auxiliary rewards.

1 Introduction

Large language models (LLMs) acquire broad knowledge during pre-training and are further adapted through post-training (Vaswani et al., 2017; Wei et al., 2022; Touvron et al., 2023a; Stiennon et al., 2020; Ouyang et al., 2022). During post-training, supervised fine-tuning (SFT) (Wang et al., 2023; Chung et al., 2024; Zhang et al., 2025b) provides stable learning signals from demonstrations, while reinforcement learning (RL) optimizes model behavior through reward feedback. For reasoning tasks with verifiable answers, reinforcement learning with verifiable rewards (RLVR) (Zhang et al., 2025a; Shao et al., 2024) has emerged as an effective paradigm for improving reasoning capabilities.

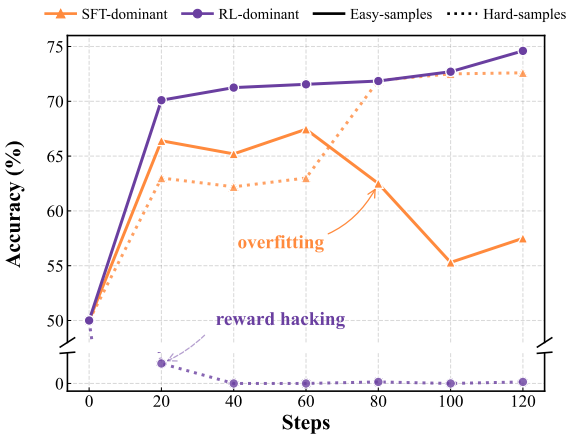


Figure 1: Static SFT–RL weighting lacks adaptivity to samples in different learning states.

SFT provides reliable guidance from demonstrations, but excessive imitation can encourage surface-level pattern replication and reduce generalization (Wei et al., 2021; Zhou et al., 2023; Wu et al., 2025). In contrast, RL enables reward-driven policy improvement through exploration, but unreliable reward signals can lead to unstable updates or reward hacking (Skalse et al., 2022; Gao et al., 2023). Consequently, recent post-training pipelines often combine SFT and RL, either sequentially (Ouyang et al., 2022) or through interleaved stages, as in reasoning models such as DeepSeek-R1 (DeepSeek-AI et al., 2025).

However, stage-wise pipelines still treat SFT and RL as largely independent processes, and RL model performance can be constrained by the initial policy (Yue et al., 2025; Nguyen et al., 2025). To address this limitation, recent studies explore tighter integration, including trajectory-guided learning (Yan et al., 2025), regularized RL

*Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 42798–42818
October 24-29, 2026 ©2026 Association for Computational Linguistics

objectives (Xu et al., 2025), and static SFT–RL loss weighting (Zhang et al., 2025c). Although these methods improve upon stage-wise pipelines, they still rely on predefined fusion rules that are not designed to adapt to individual samples or evolving training states.

Predefined weighting limits SFT–RL synergy because different samples require different balances between supervised guidance and reward-driven policy improvement. To illustrate this, we group samples by the rollout accuracy of Qwen2.5-Math-7B, treating those with at least three correct rollouts out of four as easy and those with at most one as hard. Figure 1 compares SFT-dominant and RL-dominant static fusion. The former helps hard samples but can overfit easy ones, whereas the latter benefits easy samples but provides insufficient supervision for hard ones and can lead to reward hacking. These results suggest that SFT–RL integration should be conditioned on sample learning states rather than governed by fixed global weights.

To address this limitation, we formulate SFT–RL integration as dynamic fusion based on sample learning states. At each training step, the objective should emphasize supervised learning when reward-driven updates are unreliable and emphasize RL when rollouts provide trustworthy, informative reward contrast. We introduce **Sample Learning State** (SLS), which captures these conditions with two online-rollout-based signals: sample mastery and exploration-trajectory dispersion. Sample mastery estimates core-task competence, while trajectory dispersion measures reward contrast among current rollouts. Based on SLS, we compute a safety-gated coefficient for each sample to dynamically interpolate between SFT and RL losses. This coefficient increases the SFT weight for low-mastery samples where reward-driven updates are unreliable, and increases the RL weight only when the model has sufficiently mastered the sample and multiple rollouts exhibit clear reward differences. This design suppresses unreliable RL updates in low-mastery regimes and mitigates reward hacking.

Experiments on Qwen2.5-Math-1.5B and 7B show that our method, SLS, achieves state-of-the-art performance across four in-distribution (ID) and two out-of-distribution (OOD) mathematical reasoning benchmarks, outperforming static-weighting and hybrid baselines, and in multi-reward settings, SLS optimizes auxiliary objectives while preserving core-task performance. In summary, our contributions are as follows:

- We propose Sample Learning State (SLS), which characterizes sample-level learning states via online rollout mastery and dispersion.
- We introduce a safety-gated coefficient $\alpha_i(t)$ that dynamically balances SFT and RL losses for each sample.
- SLS achieves SOTA results on ID/OOD mathematical reasoning benchmarks and is robust under auxiliary preference rewards.

2 Related Work

Existing post-training approaches (Achiam et al., 2023; Touvron et al., 2023b; DeepSeek-AI et al., 2025) commonly adopt a sequential "SFT warm-up followed by RL optimization" pipeline. Such pipelines benefit from the stability of supervised learning but may leave the subsequent RL phase constrained by the initial policy and vulnerable to objective drift during exploration.

Subsequent research has explored several forms of SFT–RL integration. *Alternating-update* methods, such as ReLIFT (Ma et al., 2025) and SASR (Chen et al., 2025), interleave RL policy updates with supervised fine-tuning. *Trajectory-guided* methods, such as LUFFY (Yan et al., 2025), SRFT (Fu et al., 2025), and TRAPO (Su et al., 2025), use offline expert demonstrations or trusted trajectories to constrain RL exploration. *Loss-weighting* methods, such as KDRL (Xu et al., 2025), RPO (Liu et al., 2024), and CHORD (Zhang et al., 2025c), integrate SFT-related regularization or weighting terms into RL training. Our method also belongs to the broad family of loss-level SFT–RL integration, but differs by replacing static global weights with a sample-state-aware coefficient determined by rollout-based learning states.

Recent work further suggests that SFT and RL can be viewed under a unified gradient-optimization perspective, differing mainly in data sources and gradient weighting (Lv et al., 2025; Wu et al., 2025). This view motivates more fine-grained integration strategies. SLS follows this direction by treating SFT–RL fusion as a sample-wise adaptive weighting problem: supervised guidance dominates when RL feedback is unreliable, while reward-driven updates receive greater weight when the current sample provides informative and trustworthy reward contrast.

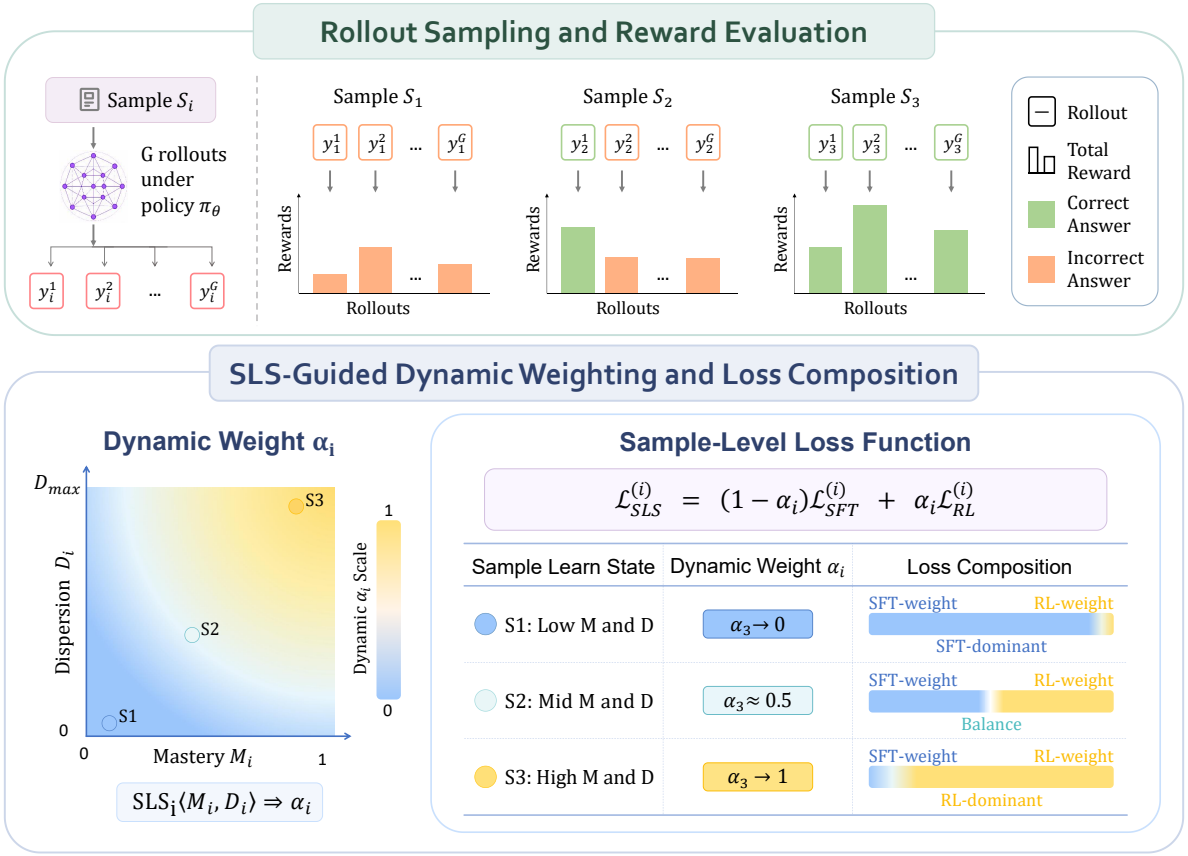


Figure 2: Overview of SLS-guided dynamic SFT–RL fusion. Online rollouts estimate sample mastery and trajectory dispersion, which determine the sample-wise coefficient for balancing SFT and RL losses.

3 Method

In this section, we introduce the proposed Sample Learning State (SLS)-guided dynamic SFT–RL fusion method, which consists of two components: sample learning state estimation and SLS-guided dynamic loss weighting. Figure 2 presents an overview of the framework.

Let θ_0 denote the initial model parameters and θ_t the parameters after t updates, corresponding to model m_{θ_t} and policy π_{θ_t}. Given the post-training set $\mathcal{S} = \{(x_i, \mathbf{y}_i)\}_{i=1}^{N}$, we feed each prompt x_i into the model m_{θ_t} and independently sample G rollout responses $\{\hat{y}_{i,j}\}_{j=1}^{G}$. We define each resulting input–output pair as a trajectory: the j-th trajectory is $\tau_{i,j} = (x_i, \hat{y}_{i,j})$, and the trajectory set for sample $s_i = (x_i, \mathbf{y}_i)$ is $\mathcal{T}_i = \{\tau_{i,j}\}_{j=1}^{G}$. Each trajectory $\tau_{i,j}$ is assigned a core-task reward $r_{i,j}^{\text{core}} \in \{0, 1\}$ and K auxiliary preference rewards $\{r_{i,j}^{k}\}_{k=1}^{K}$, such as format compliance.

Our goal is to determine the SFT–RL mixture ratio for each sample under the current model parameters. Online rollouts provide two useful observations for this decision: whether the model can reliably solve the sample, and whether the sampled trajectories provide meaningful contrastive signal for RL.

3.1 Sample Learning State

We define the **Sample Learning State** (SLS) of sample s_i at training step t as

$$\mathrm{SLS}_i(t) = \big(M_i(t), D_i(t)\big), \tag{1}$$

where $M_i(t)$ estimates the model's average competence on sample s_i, and $D_i(t)$ measures reward differences among its exploration trajectories. Together, they characterize the sample learning state of s_i. We omit t when unambiguous.

Degree of Sample Mastery M_i. The mastery score M_i quantifies how reliably the current model m_{θ_t} solves sample s_i, defined as the average core-task reward over G sampled trajectories:

$$M_i = \frac{1}{G}\sum_{j=1}^{G} r_{i,j}^{\text{core}}. \tag{2}$$

This score provides a sample-level estimate of competence: low-mastery samples benefit from

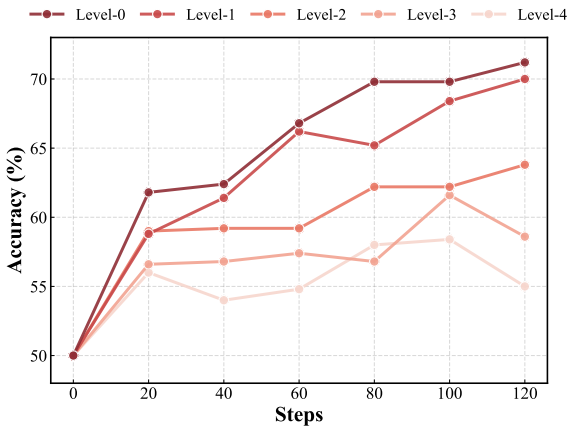


Figure 3: SFT performance across sample mastery (M) levels. **Darker** colors denote **lower** mastery.

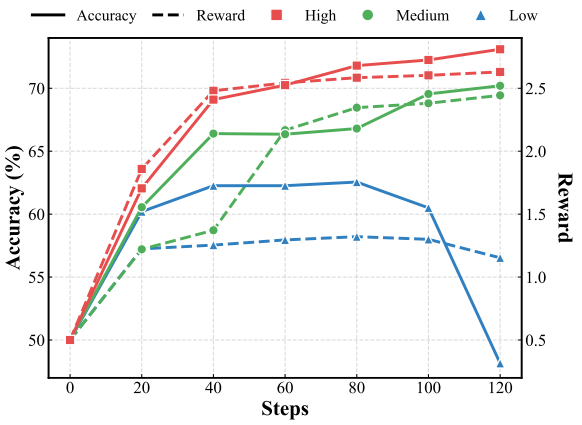


Figure 4: RL performance across varying dispersion levels (D). **Accuracy** and **Reward** denote core-task accuracy and total reward, respectively.

supervised guidance, whereas high-mastery samples may receive limited or even negative returns from excessive imitation.

To validate this relation, we divide the OpenR1-Math and GSM8K datasets into five mastery levels (Level 0–4) according to $M \in [0, 1]$. We reserve 100 random instances from each level for validation and use the remaining samples for SFT training. As shown in Figure 3, SFT gains decrease as mastery increases: low-mastery samples yield consistent improvements, while high-mastery samples show diminishing gains and sometimes performance degradation. These results suggest that SFT is most useful for samples the model has not yet mastered, whereas continued supervision on mastered samples may reduce generalization through surface-level overfitting.

Dispersion of Exploration Trajectories D_i. The dispersion score D_i quantifies reward variability among rollouts of the same sample, defined as the standard deviation of total rewards over G sampled trajectories:

$$D_i = \mathrm{Std}\left(\{R_{i,j}\}_{j=1}^{G}\right), \tag{3}$$

where $R_{i,j} = r_{i,j}^{\mathrm{core}} + \sum_{k=1}^{K} r_{i,j}^{k}$ is the composite reward of the j-th trajectory. Auxiliary rewards are optional; in the core-only setting, $K = 0$ and $R_{i,j} = r_{i,j}^{\mathrm{core}}$.

D_i estimates the informativeness of reward feedback. Larger dispersion indicates greater differences among sampled trajectories, allowing RL to extract more effective learning signals. Conversely, when dispersion is small, trajectories are less distinguishable and provide limited RL signal.

To validate the role of dispersion, we use a composite reward combining binary core correctness $r_{i,j}^{\mathrm{core}}$ with an auxiliary format reward $r_{i,j}^{\mathrm{format}}$ (Appendix C.2). We split samples from both datasets into Low, Medium, and High groups by equal-width intervals of the normalized dispersion $D_i/D_{\max}$, and train RL separately on each group. As shown in Figure 4, high-dispersion samples achieve faster reward convergence and more pronounced improvements in core-task ability, whereas low-dispersion samples show only marginal reward growth and exhibit declines in core-task ability. These findings indicate that low-contrast trajectories provide weak RL signals, which may bias optimization toward secondary objectives and impair core-task ability.

Overall, these analyses show that the effectiveness of SFT and RL varies with the learning state of each sample. This motivates a sample-adaptive fusion strategy that dynamically adjusts the loss weights based on SLS.

3.2 SLS-Guided Dynamic SFT–RL Fusion

Building on Eq. (1), we formulate post-training as a single-stage joint optimization of SFT and RL with sample-adaptive loss fusion. At training step t, each sample s_i is assigned an SLS-guided coefficient $\alpha_i(t) \in [0, 1]$ that governs its SFT–RL trade-off: smaller values emphasize SFT, whereas larger values emphasize RL. We omit t when unambiguous. The resulting objective is

$$\mathcal{L}_{\mathrm{SLS}}^{(i)}(\theta_t) = (1 - \alpha_i)\mathcal{L}_{\mathrm{SFT}}^{(i)}(\theta_t) + \alpha_i \mathcal{L}_{\mathrm{RL}}^{(i)}(\theta_t), \tag{4}$$

For each prompt x_i, the SFT term uses its demonstration response $\mathbf{y}_i$, whereas the RL term uses the online trajectory group $\mathcal{T}_i$ generated by the cur-

rent policy. SLS therefore performs sample-wise SFT–RL interpolation instead of alternating between separate SFT and RL stages. Detailed objective definitions are provided in Appendix A.

The coefficient α_i should assign more weight to RL only when RL updates are both reliable and informative. Reliability is captured by mastery M_i, which reflects whether the model can solve the core task, while informativeness is captured by dispersion D_i, which reflects trajectory-level reward contrast.

Accordingly, α_i should be bounded in $[0, 1]$ and increase with both mastery and dispersion. Importantly, these two factors should interact multiplicatively rather than additively. In multi-reward settings, high dispersion may arise from auxiliary rewards instead of meaningful reasoning differences; thus, dispersion alone should not induce strong RL updates when mastery is low. We therefore use the form

$$\alpha_i \propto g(M_i) \cdot \frac{D_i}{D_{\max}}.$$

We instantiate the mastery gate as $g(M_i) = e^{M_i} - 1$, which is zero at $M_i = 0$ and increases smoothly and convexly on $M_i \in [0, 1]$. Thus, when mastery is low, even large dispersion contributes little to the RL weight; as mastery increases, dispersion is allowed to play a stronger role. The resulting coefficient is

$$\alpha_i = \min\left(1,\ \lambda\left(e^{M_i} - 1\right)\frac{D_i}{D_{\max}}\right), \quad (5)$$

where $D_{\max}$ is the maximum attainable standard deviation for each reward configuration, and $\lambda > 0$ controls the strength of RL updates. The outer clipping keeps $\alpha_i \in [0, 1]$. The multiplicative gate promotes RL only when rollout reward contrast is informative and core-task competence is sufficient, while suppressing dispersion-driven RL updates on low-mastery samples.

We further filter out fully saturated samples whose rollouts all achieve the maximum composite reward, i.e., $\sum_{j=1}^{G} R_{i,j} = GR_{\max}$, where $R_{\max}$ denotes the maximum attainable composite reward under the current reward configuration. These samples provide no contrastive signal for group-relative RL, and continued supervised imitation on them may increase the risk of overfitting. We therefore exclude them before loss computation and accumulate unfiltered samples until the effective batch size is reached, reducing redundant computation and improving training efficiency. Together, SLS uses α_i to adaptively balance SFT and RL for informative samples, while the filtering rule excludes samples that no longer provide useful training signal.

4 Experimental Setup

Datasets and Evaluation. The training set contains 32k samples: 30k filtered instances from OpenR1-Math-220k-Default (Hugging Face, 2025) (abbreviated as OpenR1-Math) and 2k samples from GSM8K (Cobbe et al., 2021). OpenR1-Math uses prompts from NuminaMath1.5 (LI et al., 2024) and responses from DeepSeek-R1 (DeepSeek-AI et al., 2025). Unless otherwise stated, all compared methods are trained on this same 32k training set.

We evaluate all methods on four in-distribution (ID) math benchmarks: AIME24, AIME25, AMC23 (Li et al., 2024), and MATH-500 (Hendrycks et al., 2021). We further evaluate out-of-distribution (OOD) generalization on MMLU-Pro (Wang et al., 2024) and GPQA-Diamond (Rein et al., 2024), abbreviated as GPQA. Given the relatively small test sets of AIME24, AIME25, and AMC23, we report avg@32 on these three benchmarks. We additionally report the standard deviation over the 32 sampled responses in Appendix B.2. This standard deviation measures the variability of evaluation outcomes under a fixed model. For MATH-500, MMLU-Pro, and GPQA, we report pass@1. All evaluations use a temperature of 0.6, with option shuffling applied to multiple-choice tasks to mitigate contamination.

Training Configuration. Experiments use the Qwen2.5-Math series models (Yang et al., 2024). Following (Yan et al., 2025), we apply RoPE scaling to extend the context window to 16,384 tokens, and use an 8,192-token limit in both training and evaluation unless otherwise stated.

The training batch size is 32. For each sample, we generate $G = 4$ trajectories with vLLM (Kwon et al., 2023), using a sampling batch size of 32×4 and a temperature of 1.0. The learning rate is 1×10^{-5} for SFT; for RL and hybrid methods, we follow their respective configurations and select from $\{1 \times 10^{-6}, 5 \times 10^{-6}\}$. Hybrid and RL methods are trained for one epoch, while SFT is trained for three epochs. For Eq. (5), we set

Model	In-Distribution (Core Task)										OOD			
	MATH-500		AIME24		AIME25		AMC23		Avg.		MMLU-Pro		GPQA	
	1.5B	7B	1.5B	7B	1.5B	7B	1.5B	7B	1.5B	7B	1.5B	7B	1.5B	7B
Qwen2.5-Math														
Base	38.6	67.0	4.2	16.7	2.6	7.0	26.4	49.7	18.0	35.1	18.5	34.1	21.7	28.3
Instruct	71.6	82.0	9.7	9.3	7.6	7.9	44.7	49.3	33.4	37.1	30.0	45.7	30.3	32.2
Single Method														
GRPO	68.7	82.6	10.0	17.5	5.2	12.8	46.5	59.5	32.6	43.1	27.1	50.0	24.2	36.3
DAPO	67.6	81.2	9.4	16.8	5.2	10.5	42.2	59.3	31.1	42.0	24.3	51.4	25.7	34.3
SFT	65.6	80.8	10.4	24.1	10.9	**22.1**	39.7	61.1	31.7	47.0	27.1	47.1	12.6	31.8
Hybrid Method														
LUFFY	75.4	84.2	13.1	17.9	8.8	14.4	50.5	63.4	37.0	45.0	34.3	52.9	29.3	37.4
SFT-then-RL	73.5	83.1	10.7	20.5	9.0	16.8	44.6	63.1	34.5	45.9	30.3	44.4	23.2	36.5
ReLIFT	74.0	84.6	11.4	23.7	9.3	17.3	48.7	66.8	35.9	48.1	27.1	51.4	24.2	35.9
CHORD	76.2	87.2	13.1	23.2	10.5	19.3	51.3	65.8	37.8	48.9	40.0	51.4	25.8	40.4
SLS	**80.2**	**90.0**	**14.8**	**26.9**	**13.3**	21.3	**54.2**	**68.8**	**40.6**	**51.8**	**42.9**	**54.3**	**30.8**	**42.4**

Table 1: Overall performance on four in-distribution math benchmarks and two OOD benchmarks using Qwen2.5-Math-1.5B and 7B models. **Bold** and underlined denote the best and second-best results, respectively.

$\lambda = 2$, with $D_{\max}$ computed separately for each reward configuration (Eq. (21)). All experiments are conducted on four identical NVIDIA A800-80GB GPUs. Additional implementation details are provided in Appendix B.

Baseline Methods. Using Qwen2.5-Math-1.5B and 7B as base models, we compare SLS with single-paradigm references and hybrid baselines. Single-paradigm references include **SFT**, **GRPO** (Shao et al., 2024) (group-wise relative policy optimization), and **DAPO** (Yu et al., 2025), which filters out all-correct and all-incorrect samples from the rollout buffer. The primary hybrid comparisons include **SFT-then-RL**, a two-stage pipeline in which RL (GRPO) follows an initial SFT phase; **LUFFY** (Yan et al., 2025), which uses online and offline policies to guide learning; **ReLIFT** (Ma et al., 2025), which performs interleaved online fine-tuning on the hardest questions; and **CHORD** (Zhang et al., 2025c), which fuses SFT and RL with preset global weights. Like SLS, these methods combine reward-driven training with offline demonstrations or SFT-style supervision. All methods are trained on the same 32k problem set. In efficiency comparisons, we report the amount of *additional* expert demonstrations required for SFT-style supervision, together with training steps and GPU hours.

Model	GPU Hours		Extra Demos		Training Steps	
	1.5B	7B	1.5B	7B	1.5B	7B
SFT	6×4	20×4	32k	32k	3000	3000
GRPO	15×4	36×4	-	-	1000	1000
DAPO	10×4	25×4	-	-	364	396
LUFFY	20×4	47×4	32k	32k	1000	1000
ReLIFT	16×4	39×4	15k	9k	1000	1000
CHORD	24×4	56×4	32k	32k	1000	1000
SLS	17×4	44×4	17k	10k	771	574

Table 2: Training efficiency comparison. All methods are trained on the same 32k problem set. "Extra Demos" denotes additional supervised demonstrations beyond the shared training set. GPU hours are reported as $a \times 4$, i.e., a hours on each of 4 GPUs.

5 Experimental Results

5.1 Main Experimental Results

Performance on Qwen2.5-Math Series. Table 1 shows that SLS consistently outperforms both single-method and hybrid baselines on the Qwen2.5-Math models. It achieves the highest average in-distribution accuracy, reaching 40.6% with the 1.5B model and 51.8% with the 7B model. Across both in-distribution and OOD benchmarks, SLS consistently attains the best or second-best performance, indicating that its gains are robust across model sizes and are not merely due to overfitting to the in-distribution reward.

The advantage of SLS comes from a tighter sample-level integration of SFT and RL. Compared with SFT, SLS enables RL-based explo-

Setting	Model	In-Distribution (Core Task)					OOD		Aux.
		MATH-500	AIME24	AIME25	AMC23	Avg. (Δ)	MMLU-Pro	GPQA	Score
Core	GRPO	68.7	10.0	5.2	46.5	32.6	27.1	24.2	–
	CHORD	76.2	13.1	10.5	51.3	37.8	40.0	25.8	–
	SLS	**80.2**	**14.8**	**13.3**	**54.2**	**40.6**	**42.9**	**30.8**	–
Format	GRPO	0.0	0.0	0.0	0.0	0.0 (-32.6)	0.0	0.0	90.9
	CHORD	58.2	5.5	7.2	38.4	27.3 (-10.5)	32.9	27.3	91.6
	SLS	**79.6**	**17.1**	**12.6**	**50.7**	**40.0** (-0.6)	**34.3**	**35.4**	**96.5**
Type	GRPO	67.4	8.8	5.6	44.0	31.5 (-1.1)	31.4	25.3	75.8
	CHORD	71.4	8.3	7.0	45.7	33.1 (-4.7)	28.6	25.8	32.3
	SLS	**80.4**	**14.7**	**13.3**	**54.2**	**40.7** (+0.1)	**42.9**	**28.8**	**79.8**

Table 3: Robustness under multi-reward settings. Parenthesized values indicate gaps relative to the *Core* setting. Zero scores indicate reward-hacking collapse on the core task.

ration beyond fixed demonstrations. Compared with pure RL methods such as GRPO and their filtering variants such as DAPO, it provides supervised guidance for low-mastery samples, including "all-incorrect" samples that would otherwise be discarded. Additional results on models of different sizes and families, including **Qwen2.5** and **Qwen3**, are provided in Appendix Table 8.

Table 2 further shows that these gains do not simply come from more supervision or more training. Compared with one-epoch SFT–RL fusion baselines, SLS requires fewer additional supervised demonstrations and fewer training steps while achieving better final performance, suggesting that SLS-guided allocation of SFT and RL updates is more efficient than static fusion schedules.

Training Dynamics and Exploration. Figure 5 illustrates the Shannon entropy dynamics (Eq. (22)) of RL and SLS during training on Qwen2.5-Math-1.5B. Under pure RL, entropy steadily declines, suggesting reduced exploration and a progressively narrower output distribution. In contrast, SLS first increases entropy and then stabilizes at a higher level, indicating that it preserves a broader generation space while still optimizing for task reward.

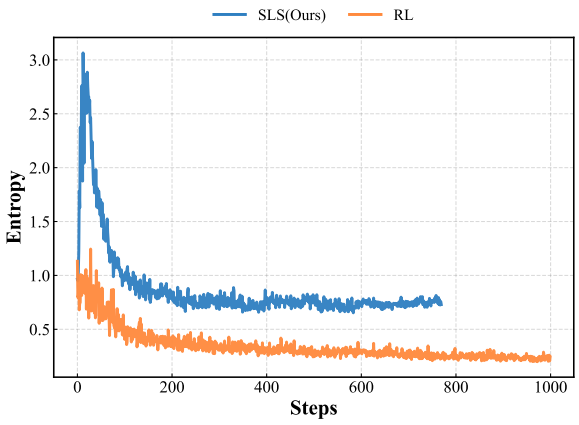


Figure 5: Entropy dynamics during RL and SLS training on Qwen2.5-Math-1.5B.

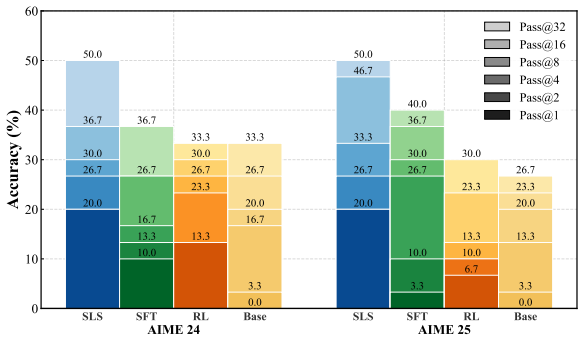


Figure 6: pass@k performance on AIME24 and AIME25. SLS improves single-sampling accuracy and maintains strong multi-sampling coverage.

Figure 6 further compares pass@k on AIME24 and AIME25. SLS improves pass@1 over RL, indicating higher single-sampling accuracy, and surpasses SFT at pass@32, suggesting broader problem-solving coverage under repeated sampling. Together with the entropy trends, these results show that SLS balances reliable generation with sustained exploration.

Robustness under Multi-Reward Settings. To evaluate robustness under multi-objective optimization, we compare SLS with GRPO and CHORD (static hybrid) using Qwen2.5-Math-1.5B. We consider three reward configurations: (1) **Core**, using only the core task reward (r^{core}); (2) **Format**, adding format compliance ($r^{\text{core}} + r^{\text{format}}$); and (3) **Type**, adding a problem classification objective ($r^{\text{core}} + r^{\text{type}}$). The corresponding prompts and reward definitions are provided in

Model	In-Distribution					OOD		Aux.
	MATH-500	AIME24	AIME25	AMC23	Avg.	MMLU-Pro	GPQA	Format
Random	73.5	9.1	9.4	47.5	34.9	18.9	23.2	90.7
Static	75.8	11.2	9.3	45.0	35.3	27.5	21.2	87.3
M-only	78.4	14.9	12.4	**52.1**	39.5	30.7	23.4	94.5
D-only	77.5	13.2	12.1	48.0	37.7	31.2	24.1	95.4
M-linear	76.2	15.6	12.2	49.7	38.4	28.7	27.3	92.7
SLS	**79.6**	**17.1**	**12.6**	50.7	**40.0**	**34.3**	**35.4**	**96.5**

Table 4: Ablation results for different α configurations under the format-augmented reward setting. Static: $\alpha_i = 0.5$; Random: $\alpha_i \sim \mathcal{U}(0,1)$; M-only: $\alpha_i = \min(1, \lambda(e^{M_i} - 1))$; D-only: $\alpha_i = \min(1, \lambda D_i / D_{\max})$; M-linear: $\alpha_i = \min(1, \lambda M_i D_i / D_{\max})$. See Appendix D.5 for details.

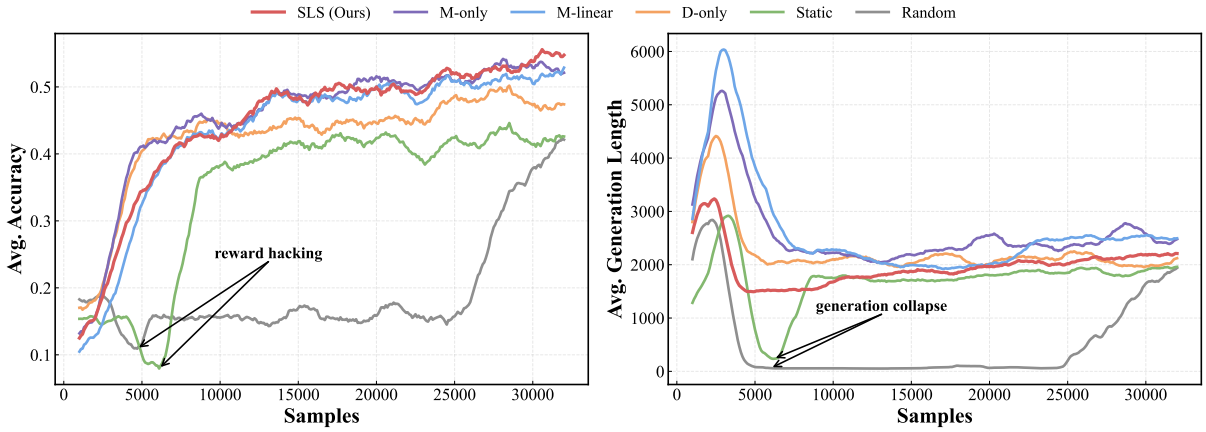


Figure 7: Training dynamics of different α configurations. **Left:** average accuracy on the core task. **Right:** average generation length. Static and Random baselines exhibit length collapse, indicating reward hacking, whereas SLS maintains stable performance.

Appendix C.1 and Appendix C.2.

Table 3 shows that the baselines are vulnerable to over-optimizing auxiliary objectives at the expense of core-task performance. Under the Format setting, GRPO achieves a high format score but loses its core-task performance entirely, leading to zero accuracy on all in-distribution and OOD benchmarks. Under the Type setting, GRPO avoids complete collapse but still degrades core-task performance, with average accuracy dropping from 32.6% to 31.5%.

CHORD is more stable than GRPO, but still suffers core-task degradation after adding auxiliary rewards. In the Format setting, its average core-task performance drops from 37.8% to 27.3%. In the Type setting, CHORD assigns the same category to all problems, a typical reward-hacking pattern that yields a low auxiliary score despite reasonable task accuracy.

In contrast, SLS maintains nearly unchanged core performance across all three settings (Core: 40.6%, Format: 40.0%, Type: 40.7%) while still achieving high auxiliary scores, reaching 96.5% for Format and 79.8% for Type. Thus, SLS can fit auxiliary preference objectives while preserving core-task performance.

These results are consistent with the safety-gated design of SLS. Auxiliary rewards can induce large trajectory dispersion on low-mastery samples, which pure RL or static fusion methods may over-amplify. SLS uses mastery to gate this signal, so the RL weight increases only when core-task competence is sufficiently established, thereby preventing auxiliary-reward contrast from dominating early optimization.

5.2 Ablation Study

Impact of Different α Configurations. We conduct ablation experiments on Qwen2.5-Math-1.5B under the $r^{\text{core}} + r^{\text{format}}$ reward setting to isolate the contribution of each component in the adaptive coefficient.

As shown in Table 4, the full SLS model achieves the highest average performance (40.0%)

Model	In-Distribution (Core Task)					OOD		Cost (GPU / Extra Demos / Steps)
	MATH-500	AIME24	AIME25	AMC23	Avg.	MMLU-Pro	GPQA	
SLS w/o Filtering	72.6	7.2	10.7	40.3	32.7	38.6	27.7	22×4 / 24k / 1000
SLS w/ Filtering	**80.2**	**14.8**	**13.3**	**54.2**	**40.6**	**42.9**	**30.8**	**17×4 / 17k / 771**

Table 5: Ablation on sample filtering. All variants use the same 32k problem set. "Extra Demos" denotes additional expert demonstrations used for supervised learning.

Model	In-Distribution (Core Task)					OOD		Aux.	Training
	MATH-500	AIME24	AIME25	AMC23	Avg.	MMLU-Pro	GPQA	Format	Steps
$\lambda = 1$	<u>78.6</u>	<u>14.5</u>	**12.9**	<u>50.1</u>	<u>39.0</u>	<u>33.5</u>	<u>26.3</u>	94.4	<u>855</u>
$\lambda = 3$	69.8	11.1	9.9	40.6	32.9	32.9	14.1	<u>95.7</u>	913
SLS ($\lambda = 2$)	**79.6**	**17.1**	<u>12.6</u>	**50.7**	**40.0**	**34.3**	**35.4**	**96.5**	**779**

Table 6: Effect of λ on Qwen2.5-Math-1.5B under the $r^{\text{core}} + r^{\text{format}}$ reward setting. **Bold** and <u>underlined</u> indicate the best and second-best results, respectively.

and the strongest OOD results. By contrast, the Static and Random baselines suffer from degraded core-task performance and output-length collapse (Figure 7), suggesting that policies that do not condition on sample learning states are more susceptible to noisy signals from trajectory differences, leading to degraded performance.

The ablation results further confirm that both components of SLS are necessary. The M-only variant (39.5%) lacks reward-feedback informativeness, limiting gains from high-dispersion samples, whereas the D-only variant (37.7%) lacks the mastery gate and is more vulnerable to auxiliary-reward-induced dispersion. The **M-linear** variant preserves multiplicative coupling but delays RL-weight amplification after competence improves. By combining nonlinear competence gating with reward-feedback informativeness, the full SLS coefficient enables a more effective and robust allocation of RL weight.

Effect of Sample Filtering. To isolate the effect of sample filtering, we compare the default SLS configuration with a full-data variant on Qwen2.5-Math-1.5B. The full-data variant retains fully saturated samples, for which all rollouts obtain the maximum composite reward. Such samples provide no contrastive signal for the group-relative RL update, and continuing to apply SFT supervision may perturb the current policy.

In the full-data variant, fully saturated samples are retained and assigned a fixed fusion coefficient of $\alpha_i = 0.5$. Compared with the default SLS configuration, retaining these samples degrades both in-distribution performance and OOD generalization while requiring more GPU hours, additional demonstrations, and training steps. This shows that filtering saturated samples removes uninformative computation and improves both effectiveness and efficiency.

Impact of Scaling Factor λ. We evaluate $\lambda \in \{1, 2, 3\}$ on Qwen2.5-Math-1.5B under the $r^{\text{core}} + r^{\text{format}}$ reward setting. As shown in Table 6, $\lambda = 2$ achieves the best ID average of 40.0% and the strongest OOD performance among the tested values. In contrast, $\lambda = 1$ limits reward-driven updates, while $\lambda = 3$ over-amplifies noisy or auxiliary reward signals and degrades core-task and OOD performance. These results indicate that $\lambda = 2$ provides a balanced scale for allocating RL weight after mastery gating.

6 Conclusion

We propose Sample Learning State (SLS), a rollout-based characterization of sample mastery ($M_i(t)$) and trajectory dispersion ($D_i(t)$). Based on this state, SLS derives a sample-wise coefficient $\alpha_i(t)$ to balance supervised guidance and reward-driven exploration. By increasing the RL contribution only when core-task competence and rollout contrast are sufficient, SLS avoids unreliable updates in low-mastery regimes. Within the evaluated mathematical RLVR settings, experiments show that SLS improves reasoning performance, mitigates reward hacking, and remains effective across model scales and model families. These results support sample-state-aware dynamic control as an effective alternative to fixed global SFT–RL trade-offs.

7 Limitations

Our method is primarily developed for RL with verifiable rewards (RLVR), where the sample mastery score $M_i(t)$ relies on deterministic binary core rewards. Such outcome-level rewards provide clear correctness signals but do not assess intermediate reasoning quality. When a model reaches the correct answer through unreliable reasoning, the corresponding sample may be treated as mastered, which can affect the learning of core reasoning skills. The stability and effectiveness of SLS should therefore be further validated with intermediate feedback or process-level supervision. Our empirical evidence is also concentrated on mathematical reasoning tasks, and extending SLS to instruction-alignment scenarios without verifiable objectives remains future work. Finally, although our experiments cover multiple model scales and families, larger-scale evaluations are needed to better understand the scaling behavior of SLS.

Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2024YFF0907003). We are grateful to the anonymous reviewers for their constructive comments.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jack Chen, Fazhong Liu, Naruto Liu, Yuhan Luo, Erqu Qin, Harry Zheng, Tian Dong, Haojin Zhu, Yan Meng, and Xiao Wang. 2025. Step-wise adaptive integration of supervised fine-tuning and reinforcement learning for task-specific llms. *Preprint*, arXiv:2505.13026.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and 1 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

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. *Preprint*, arXiv:2110.14168.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Hugging Face. 2024. Math-verify. https://github.com/huggingface/Math-Verify. Accessed: 2024-06-10.

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. *Preprint*, arXiv:2506.19767.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

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://huggingface.co/AI-MO/NuminaMath-1.5.

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.

Zhihan Liu, Miao Lu, Shenao Zhang, Boyi Liu, Hongyi Guo, Yingxiang Yang, Jose Blanchet, and Zhaoran Wang. 2024. Provably mitigating overoptimization in RLHF: Your SFT loss is implicitly an adversarial regularizer. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xingtai Lv, Yuxin Zuo, Youbang Sun, Hongyi Liu, Yuntian Wei, Zhekai Chen, Lixuan He, Xuekai Zhu, Kaiyan Zhang, Bingning Wang, and 1 others. 2025. Towards a unified view of large language model post-training. *arXiv preprint arXiv:2509.04419*.

Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Hao Wong, Junbo Niu, Chengyu Shen, Runming He, Yanhao Li, Bin Cui, and Wentao Zhang. 2025. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. *Preprint*, arXiv:2506.07527.

Phuc Minh Nguyen, Chinh D La, Duy MH Nguyen, Nitesh V Chawla, Binh T Nguyen, and Khoa D Doan. 2025. The reasoning boundary paradox: How reinforcement learning constrains language models. *arXiv preprint arXiv:2510.02230*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

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

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward hacking. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021.

Mingyu Su, Jian Guan, Yuxian Gu, Minlie Huang, and Hongning Wang. 2025. Trust-region adaptive policy optimization. *arXiv preprint arXiv:2512.17636*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. 2025. On the generalization of sft: A reinforcement learning perspective with reward rectification. *arXiv preprint arXiv:2508.05629*.

Hongling Xu, Qi Zhu, Heyuan Deng, Jinpeng Li, Lu Hou, Yasheng Wang, Lifeng Shang, Ruifeng

Xu, and Fei Mi. 2025. Kdrl: Post-training reasoning llms via unified knowledge distillation and reinforcement learning. *Preprint*, arXiv:2506.02208.

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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv 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. *Preprint*, arXiv:2409.12122.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, and 17 others. 2025. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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

Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, Yu Fu, Xingtai Lv, Yuchen Zhang, Sihang Zeng, Shang Qu, Haozhan Li, Shijie Wang, Yuru Wang, Xinwei Long, and 20 others. 2025a. A survey of reinforcement learning for large reasoning models. *Preprint*, arXiv:2509.08827.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Guoyin Wang, and Fei Wu. 2025b. Instruction tuning for large language models: A survey. *ACM Comput. Surv.* Just Accepted.

Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2025c. On-policy rl meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting. *CoRR*, abs/2508.11408.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*.

A Method Details

This appendix provides additional details on the method, experimental protocols, supplementary results, and qualitative analyses.

A.1 Training Objectives

Supervised Fine-Tuning (SFT) Loss. For sample $s_i = (x_i, \mathbf{y}_i)$, SFT minimizes the negative log-likelihood of the reference reasoning sequence $\mathbf{y}_i$. Let $y_{i,l}$ denote the l-th token of $\mathbf{y}_i$, and let $y_{i,<l}$ denote the preceding tokens:

$$\mathcal{L}_{\mathrm{SFT}}^{(i)}(\theta_t) = -\frac{1}{|\mathbf{y}_i|}\sum_{l=1}^{|\mathbf{y}_i|}\log \pi_{\theta_t}(y_{i,l} \mid x_i, y_{i,<l}). \tag{6}$$

Reinforcement Learning (RL) Loss. We use Group Relative Policy Optimization (GRPO) (Shao et al., 2024) as the RL loss. Following (Cui et al., 2025; Yu et al., 2025; Hu et al., 2025), we remove the KL-divergence regularizer to avoid additional constraints on policy-update magnitudes during exploration.

For input x_i, we generate G trajectories $\mathcal{T}_i = \{\tau_{i,1}, \ldots, \tau_{i,G}\}$ from the rollout policy $\pi_{\theta_{\text{old}}}$, where $\tau_{i,j} = (x_i, \hat{y}_{i,j})$. Here, $\pi_{\theta_{\text{old}}}$ denotes the policy distribution used for rollout sampling, while π_{θ_t} denotes the current policy distribution being optimized during the parameter update. For compactness, we write the PPO-clipped GRPO objective in a trajectory-level form:

$$\mathcal{L}_{\mathrm{RL}}^{(i)}(\theta_t) = -\frac{1}{G}\sum_{j=1}^{G}\min\Big(\rho_{i,j}\hat{A}_{i,j}, \\ \mathrm{clip}(\rho_{i,j}, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})\hat{A}_{i,j}\Big), \tag{7}$$

where $\epsilon_{\text{clip}} = 0.2$. The response-sequence importance ratio for the j-th trajectory $\rho_{i,j}$ is:

$$\rho_{i,j} = \frac{\pi_{\theta_t}(\hat{y}_{i,j} \mid x_i)}{\pi_{\theta_{\text{old}}}(\hat{y}_{i,j} \mid x_i)}. \tag{8}$$

The advantage $\hat{A}_{i,j}$ is computed from rewards within the same trajectory group $\mathcal{T}_i$:

$$\begin{aligned} \hat{A}_{i,j} &= \frac{R_{i,j} - \bar{R}_i}{\sigma_i + \epsilon_A}, \\ \bar{R}_i &= \frac{1}{G}\sum_{\ell=1}^{G} R_{i,\ell}, \\ \sigma_i &= \sqrt{\frac{1}{G}\sum_{\ell=1}^{G}\left(R_{i,\ell} - \bar{R}_i\right)^2}. \end{aligned} \tag{9}$$

We set $\epsilon_A = 10^{-8}$ for numerical stability, preventing division by zero when the group-level reward standard deviation σ_i is zero. In this case, all trajectories receive identical rewards and the normalized advantages are zero, so the RL term provides no policy-gradient signal. Consistently, SLS assigns such samples $D_i = 0$ and hence $\alpha_i = 0$, leaving them to be handled by the SFT component unless they are saturated and removed by the filtering rule. When computed on the same rollout group, σ_i is identical to the dispersion score D_i.

A.2 Gradient-Level View of SFT–RL Fusion

A unified gradient-level view. SFT and RL updates can be interpreted under a common log-probability-gradient view.

For SFT, the loss for a single sample is:

$$\mathcal{L}_{\mathrm{SFT}}^{(i)}(\theta) = -\frac{1}{|\mathbf{y}_i|}\sum_{l=1}^{|\mathbf{y}_i|}\log \pi_\theta(y_{i,l} \mid x_i, y_{i,<l}). \tag{10}$$

Its gradient is

$$\begin{aligned}&\nabla_\theta \mathcal{L}_{\mathrm{SFT}}^{(i)}(\theta)\\ &= -\frac{1}{|\mathbf{y}_i|}\sum_{l}\nabla_\theta \log \pi_\theta(y_{i,l} \mid x_i, y_{i,<l}).\end{aligned} \tag{11}$$

This corresponds to a log-probability gradient with a constant weight.

For RL, the GRPO objective in Eq. (7) uses an importance ratio and clipping. Ignoring clipping for interpretation and taking the ratio locally around 1, the update has the following first-order form:

$$\nabla_\theta \mathcal{L}_{\mathrm{RL}}^{(i)}(\theta) \approx -\frac{1}{G}\sum_{j=1}^{G}\hat{A}_{i,j}\nabla_\theta \log \pi_\theta(\hat{y}_{i,j} \mid x_i). \tag{12}$$

Here, the response log-probability for the trajectory $\tau_{i,j} = (x_i, \hat{y}_{i,j})$ decomposes as

$$\log \pi_\theta(\hat{y}_{i,j} \mid x_i) = \sum_{l}\log \pi_\theta(\hat{y}_{i,j,l} \mid x_i, \hat{y}_{i,j,<l}). \tag{13}$$

Thus, RL has the same basic response log-probability-gradient structure as SFT, but uses trajectory-level advantage weights.

The SLS objective is

$$\mathcal{L}_{\mathrm{SLS}}^{(i)}(\theta) = (1-\alpha_i)\mathcal{L}_{\mathrm{SFT}}^{(i)}(\theta) + \alpha_i \mathcal{L}_{\mathrm{RL}}^{(i)}(\theta), \tag{14}$$

where α_i is computed from the sample learning state (M_i, D_i) and treated as a stop-gradient scalar during the corresponding parameter update. The resulting gradient is:

$$\nabla_\theta \mathcal{L}_{\mathrm{SLS}}^{(i)}(\theta) = (1-\alpha_i)\nabla_\theta \mathcal{L}_{\mathrm{SFT}}^{(i)}(\theta) + \alpha_i \nabla_\theta \mathcal{L}_{\mathrm{RL}}^{(i)}(\theta). \tag{15}$$

This shows that SLS can be interpreted as a state-conditioned interpolation between a constant-weight supervised update and an advantage-weighted policy update.

This view also clarifies the role of $\alpha_i = f(M_i, D_i)$. When mastery and dispersion are low, reward differences across trajectories are weak or unreliable, so a smaller α_i favors the supervised signal. When mastery is high and dispersion is informative, a larger α_i increases the RL contribution and exploits stronger contrastive signals from exploration.

A.3 Sample Learning State Details

Degree of Sample Mastery ($M_i(t)$). In the main RLVR setting, we compute the mastery of sample s_i at step t as the average binary core reward over G rollouts. We also present a normalized form for possible extensions to non-binary reward settings:

$$M_i(t) = \frac{1}{G}\sum_{j=1}^{G}\tilde{r}_{i,j}^{\mathrm{core}}, \tag{16}$$

where $\tilde{r}_{i,j}^{\mathrm{core}} \in [0,1]$ is the normalized core reward. In our experiments, $r_{i,j}^{\mathrm{core}} \in \{0,1\}$, so $\tilde{r}_{i,j}^{\mathrm{core}} = r_{i,j}^{\mathrm{core}}$. For continuous or otherwise non-comparable core rewards, $r_{i,j}^{\mathrm{core}}$ can be mapped to $[0,1]$ using thresholds $\tau_{\mathrm{low}} < \tau_{\mathrm{high}}$:

$$\tilde{r}_{i,j}^{\mathrm{core}} = \begin{cases} 0, & r_{i,j}^{\mathrm{core}} < \tau_{\mathrm{low}}, \\ \dfrac{r_{i,j}^{\mathrm{core}} - \tau_{\mathrm{low}}}{\tau_{\mathrm{high}} - \tau_{\mathrm{low}}}, & \tau_{\mathrm{low}} \le r_{i,j}^{\mathrm{core}} < \tau_{\mathrm{high}}, \\ 1, & r_{i,j}^{\mathrm{core}} \ge \tau_{\mathrm{high}}. \end{cases} \tag{17}$$

This makes $M_i(t)$ comparable across reward specifications beyond binary RLVR. Here, t indexes the current model state m_{θ_t}. When it is clear from context, we omit t and write α_i, M_i, and D_i.

Dispersion of Exploration Trajectories ($D_i(t)$). We measure trajectory dispersion for sample s_i at step t by the standard deviation of composite rewards across G independent rollouts:

$$D_i(t) = \mathrm{Std}\big(\{R_{i,j}\}_{j=1}^{G}\big), \tag{18}$$

where the composite reward of the j-th trajectory is:

$$R_{i,j} = r_{i,j}^{\text{core}} + \sum_{k=1}^{K} r_{i,j}^{k}. \tag{19}$$

The standard deviation is computed as

$$\begin{aligned} \text{Std}\big(\{R_{i,j}\}_{j=1}^{G}\big) &= \sqrt{\frac{1}{G}\sum_{j=1}^{G}\big(R_{i,j} - \bar{R}_i\big)^2}, \\ \bar{R}_i &= \frac{1}{G}\sum_{j=1}^{G} R_{i,j}. \end{aligned} \tag{20}$$

A.4 Fusion Coefficient Details

Granularity of the fusion coefficient. SLS defines the fusion coefficient at the sample level because both of its state variables are sample-level quantities. The mastery score M_i is computed from final-task rewards over rollouts, while the dispersion score D_i measures reward variation among multiple trajectories generated for the same prompt. This design also matches the reward granularity in RLVR, where supervision from rewards is typically available only as sequence-level outcomes. Applying the SFT–RL trade-off at the token level would instead couple the coefficient to local token losses, introducing a mismatch with the sequence-level reward signal and potentially making credit assignment less stable. Therefore, SLS adjusts the SFT–RL balance at the sample level.

$D_{\max}$ for normalization. In Eq. (5), D_i is normalized by $D_{\max}$, which is computed separately for each reward configuration. Suppose each reward component is bounded, with $r^{\text{core}} \in [r_{\min}^{\text{core}}, r_{\max}^{\text{core}}]$ and $r^k \in [r_{\min}^k, r_{\max}^k]$. Then the composite reward $R_{i,j} = r_{i,j}^{\text{core}} + \sum_{k=1}^{K} r_{i,j}^k$ is bounded by $R_{i,j} \in [R_{\min}, R_{\max}]$, where

$$\begin{aligned} R_{\min} &= r_{\min}^{\text{core}} + \sum_{k=1}^{K} r_{\min}^{k}, \\ R_{\max} &= r_{\max}^{\text{core}} + \sum_{k=1}^{K} r_{\max}^{k}. \end{aligned}$$

Let $\Delta = R_{\max} - R_{\min}$. For G rollouts, the largest attainable standard deviation of a bounded variable in this range is obtained when the samples are split as evenly as possible between the two endpoints. Thus,

$$D_{\max} = \Delta\sqrt{\frac{\lceil G/2 \rceil \lfloor G/2 \rfloor}{G^2}}. \tag{21}$$

Because $D_{\max}$ is determined by the reward range rather than minibatch statistics, the normalized quantity $D_i/D_{\max}$ remains comparable across different reward settings. In our experiments with $G = 4$, this yields $D_{\max} = 0.5$ for the Core setting, $D_{\max} = 1.55$ for the Core+Format setting with $r_{\max}^{\text{format}} = 2.1$, and $D_{\max} = 1.0$ for the Core+Type setting.

Functional form of the fusion coefficient. The fusion coefficient α_i is designed to satisfy four properties: it should (1) lie in $[0, 1]$, (2) increase with sample mastery M_i, (3) increase with informative trajectory dispersion D_i, and (4) suppress RL updates when mastery is low, even if auxiliary rewards create superficial trajectory variation. The last property is particularly important in multi-reward settings, where a large D_i may reflect auxiliary-reward differences rather than meaningful progress on the core task.

These requirements motivate a multiplicative coupling between mastery and dispersion. An additive scoring form,

$$f_{\text{add}}(M_i, D_i) = a g(M_i) + b\frac{D_i}{D_{\max}}, \qquad b > 0,$$

does not provide such a safety gate. Even if $g(0) = 0$, a sample with $M_i = 0$ and $D_i > 0$ still receives $f_{\text{add}}(0, D_i) = bD_i/D_{\max} > 0$, allowing auxiliary-reward-induced dispersion to increase the RL weight before the model has demonstrated any core-task competence.

SLS instead uses a multiplicative form,

$$\alpha_i \propto g(M_i) \cdot \frac{D_i}{D_{\max}},$$

where mastery gates the contribution of dispersion. With $g(0) = 0$, samples with no successful core-task rollout receive zero RL weight regardless of auxiliary-reward dispersion. Thus, dispersion can promote RL only after the model has acquired sufficient competence on the sample.

We instantiate the mastery gate as

$$g(M_i) = e^{M_i} - 1.$$

The gate satisfies $g(0) = 0$, with $g'(M_i) = g''(M_i) = e^{M_i} > 0$ on $M_i \in [0, 1]$. It is therefore zero when mastery is absent, monotonic, and convex. The zero value suppresses RL updates when no rollout solves the core task, while the convex growth increasingly amplifies the effect of trajectory dispersion as mastery improves.

Combining these components gives Eq. (5): $g(M_i)$ serves as the mastery gate, $D_i/D_{\max}$ measures normalized trajectory contrast, λ controls the overall RL scale, and the outer clipping operation keeps α_i bounded in $[0, 1]$.

A.5 Training Procedure

Algorithm 1 summarizes the SLS-guided training procedure. A key implementation detail is the use of an effective sample buffer. Fully saturated samples, whose rollouts all obtain the maximum composite reward, are filtered out because they provide no contrastive signal for the group-relative RL update and continuing to apply SFT supervision may perturb the current policy. Since the number of valid samples obtained from each rollout collection may vary, samples that exceed the current update batch size are retained in the buffer together with their rollout statistics and fusion coefficients computed at collection time.

When the buffer does not contain enough valid samples, the procedure collects online rollouts, filters fully saturated samples, estimates sample learning states, and assigns sample-wise fusion coefficients before updating the model. The effective buffer improves sample utilization by retaining valid samples that exceed the current update batch size.

Algorithm 1 SLS-Guided Dynamic SFT–RL Training

Require: Training set $\mathcal{S} = \{(x_i, \mathbf{y}_i)\}_{i=1}^{N}$, initial policy π_{θ_0}, number of rollouts G, scaling factor λ, composite reward upper bound $R_{\max}$, normalization constant $D_{\max}$, batch size B

Initialize effective buffer $\mathcal{Q}_{\text{eff}} \leftarrow \emptyset$
for each training step t **do**
  if $|\mathcal{Q}_{\text{eff}}| < B$ **then**
    repeat
      Sample candidate batch $\mathcal{B}_{\text{cand}}$ from $\mathcal{S}$
      for each sample $s_i \in \mathcal{B}_{\text{cand}}$ **do**
        Generate rollout responses $\{\hat{y}_{i,j}\}_{j=1}^{G}$ using π_{θ_t} and form trajectories $\mathcal{T}_i = \{\tau_{i,j}\}_{j=1}^{G}$ with $\tau_{i,j} = (x_i, \hat{y}_{i,j})$
        Compute composite rewards $R_{i,j} = r_{i,j}^{\text{core}} + \sum_{k=1}^{K} r_{i,j}^{k}$
        Compute sample learning state $(M_i(t), D_i(t))$
        if $\sum_{j=1}^{G} R_{i,j} = GR_{\max}$ **then**
          Discard s_i
        else
          Compute $\alpha_i(t) = \min\left(1, \lambda(e^{M_i(t)} - 1)\frac{D_i(t)}{D_{\max}}\right)$
          Add $(s_i, \mathcal{T}_i, \{R_{i,j}\}_{j=1}^{G}, \alpha_i(t))$ to $\mathcal{Q}_{\text{eff}}$
        end if
      end for
    until $|\mathcal{Q}_{\text{eff}}| \geq B$
  end if
  Pop B samples from $\mathcal{Q}_{\text{eff}}$ to form $\mathcal{B}_{\text{eff}}$
  Update θ_t on $\mathcal{B}_{\text{eff}}$ using $\mathcal{L}_{\text{SLS}}$ with the stored coefficients α_i
end for

B Experimental Setup Details

B.1 Data, Models, and Hyperparameters

Dataset. The training set contains 32k samples, consisting of 30k filtered samples from **OpenR1-Math-220k-Default** (Hugging Face, 2025) and 2k randomly sampled instances from **GSM8K** (Cobbe et al., 2021). Unless otherwise stated, the two sources are mixed uniformly during training. We abbreviate OpenR1-Math-220k-Default as **OpenR1-Math**. For OpenR1-Math, we filter samples according to four criteria: (i) answer correctness verified by `math-verify` (Face, 2024), (ii) format compliance, (iii) response length shorter than 8,192 tokens, and (iv) absence of hyperlinks.

Models. For Qwen2.5-Math models, we follow (Yan et al., 2025) and apply RoPE scaling by increasing the RoPE base from 10,000 to 40,000, which extends the supported context window to 16,384 tokens. Unless otherwise stated, we use an 8,192-token limit for both training and evaluation. In addition to Qwen2.5-Math models, we evaluate the generality of our method on **Qwen2.5-7B** and **Qwen3-4B-Base** (Yang et al., 2025).

Hyperparameters. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, no weight decay, a constant learning rate, and no warmup steps. For SFT, the learning rate is set to 1×10^{-5}. For RL-based and hybrid baselines, we follow their respective original configurations where applicable, using 1×10^{-6} for GRPO, DAPO, LUFFY, and ReLIFT, and 5×10^{-6} for CHORD and SLS. For RL-based methods, trajectories are sampled with temperature 1.0. We set the per-update prompt batch size to 32 and generate $G = 4$ rollouts per prompt, resulting in 32×4 **trajectories** per update.

Model	AIME24				AIME25				AMC23			
	1.5B		7B		1.5B		7B		1.5B		7B	
	avg@32	std	**avg@32**	std	**avg@32**	std	**avg@32**	std	**avg@32**	std	**avg@32**	std
Qwen2.5-Math												
Base	4.2	3.0	16.7	4.0	2.6	2.7	7.0	3.8	26.4	5.0	49.7	4.0
Instruct	9.7	3.6	9.3	2.7	7.6	3.3	7.9	3.2	44.7	5.6	49.3	5.1
Single Method												
GRPO	10.0	2.9	17.5	5.7	5.2	3.3	12.8	3.2	46.5	5.4	59.5	4.5
DAPO	9.4	2.4	16.8	5.2	5.2	3.3	10.5	3.6	42.2	5.1	59.3	4.5
SFT	10.4	3.0	<u>24.1</u>	4.4	<u>10.9</u>	4.0	**22.1**	4.4	39.7	6.0	61.1	4.9
Hybrid Method												
LUFFY	<u>13.1</u>	4.0	17.9	4.5	8.8	4.1	14.4	4.1	50.5	5.8	63.4	4.1
SFT-then-RL	10.7	3.3	20.5	4.9	9.0	3.7	16.8	4.7	44.6	5.6	63.1	4.6
ReLIFT	11.4	3.7	23.7	4.7	9.3	3.6	17.3	4.3	48.7	6.0	<u>66.8</u>	4.8
CHORD	<u>13.1</u>	4.0	23.2	4.6	10.5	3.6	19.3	3.2	<u>51.3</u>	4.5	65.8	3.7
SLS	**14.8**	3.3	**26.9**	4.1	**13.3**	3.1	<u>21.3</u>	3.6	**54.2**	4.8	**68.8**	5.8

Table 7: avg@32 and corresponding standard deviations over 32 sampled responses on AIME24, AIME25, and AMC23. **Bold** and <u>underlined</u> denote the best and second-best avg@32 results, respectively.

Evaluation. We use a unified evaluation protocol with sampling temperature 0.6 and a maximum generation length of 8,192 tokens. Answer correctness is verified by `math-verify`. For AIME24, AIME25, and AMC23, we report avg@32 accuracy by generating 32 responses per problem and averaging the correctness over all generated responses. For MATH-500, MMLU-Pro, and GPQA, we report pass@1 accuracy using a single sampled response per problem. All baselines are evaluated under the same settings. In Figure 6, pass@k denotes the probability that at least one of k sampled responses solves the problem.

B.2 Sampling Variability on Small Benchmarks

Table 7 reports avg@32 and the corresponding standard deviation over the 32 sampled responses on AIME24, AIME25, and AMC23. Because these benchmarks have relatively small test sets, their evaluation results can vary noticeably across sampled responses. We therefore additionally report the standard deviations to provide context for the avg@32 results in Table 1.

B.3 Analysis Metrics

Shannon Entropy (H). To analyze output diversity during training, we compute the mean token-level Shannon entropy of each rollout response. For the response sequence $\hat{y}_{i,j} = (\hat{y}_{i,j,1}, \ldots, \hat{y}_{i,j,|\hat{y}_{i,j}|})$ in the j-th trajectory $\tau_{i,j} = (x_i, \hat{y}_{i,j})$ generated for sample s_i under policy π_{θ_t}, the entropy is defined as

$$H(\hat{y}_{i,j}) = -\frac{1}{|\hat{y}_{i,j}|} \sum_{l=1}^{|\hat{y}_{i,j}|} \sum_{v \in \mathcal{V}} \pi_{\theta_t}(v \mid x_i, \hat{y}_{i,j,<l}) \cdot \log \pi_{\theta_t}(v \mid x_i, \hat{y}_{i,j,<l}) \,. \tag{22}$$

Here, $\mathcal{V}$ denotes the vocabulary, and $\hat{y}_{i,j,<l}$ denotes the prefix tokens before position l in the generated response. We report the average entropy over sampled rollout responses as an auxiliary metric for tracking changes in exploration and output diversity.

C Prompts and Reward Functions

C.1 System Prompts

We use task-specific system prompts and keep them fixed across training and evaluation. The prompts for the three reward configurations are listed below.

Core Reward Only.

```
You are a helpful AI assistant that provides detailed thought processes and correct answers. Your final answer must be placed within \boxed{}.
```

Core + Format Reward.

```
You are a helpful AI assistant that provides
detailed thought processes and correct
answers. Please respond in the following
format:
<think>
[Your step-by-step thinking process here]
</think> <answer>
[Your clear solution and answer here, with
the answer in \boxed{}]
</answer>
```

Core + Problem-Type Reward.

```
You are a helpful AI assistant that provides
detailed thought processes and correct
answers. Your final answer must be placed
within \boxed{}.
After answering, select the most appropriate
category from: Algebra, Geometry, Number
Theory, Combinatorics, Primary Math,
Calculus, Logic and Puzzles, Inequalities,
or Other. End your response strictly with:
This is a [Category] question.
```

C.2 Reward Functions

Unless otherwise specified, we use the following reward configurations in our training experiments.

Core Task Reward. We use a binary outcome reward to measure mathematical correctness:

$$r^{\text{core}} = \begin{cases} 1, & \text{if the answer is correct,} \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

Format Compliance Reward. To encourage a "Think-then-Answer" response format, the output is expected to follow `<think>...</think><answer>...</answer>`. Let $\mathcal{Z} = (z_1, z_2, z_3, z_4)$ denote the four required tags `<think>`, `</think>`, `<answer>`, and `</answer>`. To reduce reward sparsity, especially for smaller models, we use a dense format reward:

$$\begin{aligned} r^{\text{format}} = w_1 \sum_{q=1}^{4} \mathbb{I}(C_q^{\text{exist}}) \\ + w_2 \sum_{q=1}^{4} \mathbb{I}(C_q^{\text{unique}}) \\ + w_3 \sum_{q=1}^{4} \mathbb{I}(C_q^{\text{struct}}) \\ + w_4 \mathbb{I}(C^{\text{ans}}), \end{aligned} \tag{24}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. Here, C_q^{exist} and C_q^{unique} check whether tag z_q appears and appears exactly once, respectively. The structural constraints C_q^{struct} check whether z_1 precedes z_2, z_2 precedes z_3, z_3 precedes z_4, and no extra content appears after z_4. Finally, C^{ans} checks whether the final answer is placed inside the `<answer>...</answer>` span. The reward consists of the following components:

$$\begin{aligned} w_1 &= 0.075, \quad & C_1 &: \text{tag existence}, \\ w_2 &= 0.10, & C_2 &: \text{tag uniqueness}, \\ w_3 &= 0.10, & C_3 &: \text{structure}, \\ w_4 &= 1.00, & C_4 &: \text{answer placement}. \end{aligned}$$

Since there are four required tags, four uniqueness constraints, four structural constraints, and one answer-placement constraint, the maximum format reward is

$$\begin{aligned} r_{\max}^{\text{format}} &= 4 \times 0.075 + 4 \times 0.10 \\ &\quad + 4 \times 0.10 + 1.00 \\ &= 2.1. \end{aligned}$$

Format tags are already present in the OpenR1 dataset and are manually annotated for GSM8K.

Problem-Type Reward. To simulate auxiliary preference alignment, we introduce a problem-type classification objective. We use the OpenR1 problem types as labels and assign `Primary Math` as the default label for GSM8K:

$$r^{\text{type}} = \begin{cases} 1 & \text{if prediction matches label,} \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

Role of Auxiliary Rewards. Auxiliary rewards represent secondary preferences rather than the primary task objective. The format reward encourages well-structured responses, while the problem-type reward encourages consistency with problem-type annotations. These rewards are intended to complement, not replace, core correctness; therefore, improvements on auxiliary objectives are meaningful only when core-task performance is preserved.

D Analysis Protocols and Ablation Variants

This section describes the experimental protocols used for the motivation and analysis results in Sections 1 and 3.

D.1 Dataset Construction and Mastery Grouping

We use the full OpenR1-Math and GSM8K training sets as the data source. Using Qwen2.5-Math-7B, we generate $G = 4$ rollouts for each sample and compute its rollout correctness ratio. Samples are then stratified into five mastery levels: Level 0 for $[0, 0.2)$, Level 1 for $[0.2, 0.4)$, Level 2 for $[0.4, 0.6)$, Level 3 for $[0.6, 0.8)$, and Level 4 for $[0.8, 1.0]$.

We randomly sample 100 instances from each level to construct a 500-sample validation set, which is excluded from training. The remaining samples are used for the following analysis experiments.

D.2 Static Weight Analysis

Figure 1 evaluates how static SFT–RL weights behave on samples in different learning states. We first group samples according to their rollout correctness ratio under Qwen2.5-Math-7B. Following Section 1, easy samples are those with at least 3 correct rollouts out of 4, while hard samples are those with at most 1 correct rollout. For each static-weight experiment, we filter the training data and select the corresponding easy or hard subset, allowing us to isolate how fixed SFT–RL weights behave under different sample learning states.

We train Qwen2.5-Math-7B on the selected subset with both core-task and format rewards, using an online replay buffer of filtered trajectories. The loss follows Eq. (4) under two fixed-weight configurations:

- **SFT-dominant ($\alpha = 0.1$):** high SFT weight and low RL weight.
- **RL-dominant ($\alpha = 0.9$):** low SFT weight and high RL weight.

We train each configuration for 120 update steps and evaluate checkpoints every 20 steps on the 500-sample validation set. We separately report the performance trajectories of models trained on the easy and hard subsets to illustrate the failure modes of fixed SFT–RL weighting.

D.3 Analysis of Sample Mastery $M_i(t)$

Figure 3 analyzes the relationship between sample mastery $M_i(t)$ and the effectiveness of supervised fine-tuning. We conduct five independent SFT experiments, each using samples from one mastery level (Level 0–4), while keeping the same hyperparameters as the main SFT baseline. Performance is evaluated on the 500-sample validation set.

D.4 Analysis of Trajectory Dispersion $D_i(t)$

Figure 4 analyzes how trajectory dispersion $D_i(t)$ affects RL optimization. We train with both core-task and format rewards and monitor total reward and core-task accuracy on the 500-sample validation set. Since D_i is normalized by the maximum attainable dispersion $D_{\max}$, we divide the normalized dispersion $D_i/D_{\max}$ into three equal-width intervals: *Low* for $[0, 1/3)$, *Medium* for $[1/3, 2/3)$, and *High* for $[2/3, 1]$.

D.5 Ablation Variants of the Fusion Coefficient

The ablation study in Section 5.2 uses Qwen2.5-Math-1.5B with the reward configuration $r^{\text{core}} + r^{\text{format}}$. All variants follow the same SLS training protocol and differ only in how the fusion coefficient α_i is computed.

Random. The Random variant assigns a uniformly sampled fusion coefficient to each sample:

$$\alpha_i \sim \mathcal{U}(0, 1).$$

This baseline tests whether arbitrary sample-wise variation in the SFT–RL mixture is sufficient.

Static. The Static variant uses a fixed global coefficient for all samples:

$$\alpha_i = 0.5.$$

This corresponds to a non-adaptive SFT–RL mixture with equal weighting.

M-only. The M-only variant uses only sample mastery and removes the dispersion term:

$$\alpha_i = \min\left(1,\ \lambda(e^{M_i} - 1)\right).$$

This variant tests whether core-task competence alone is sufficient for deciding the RL weight.

D-only. The D-only variant uses only trajectory dispersion and removes the mastery gate:

$$\alpha_i = \min\left(1,\ \lambda \frac{D_i}{D_{\max}}\right).$$

This variant tests whether reward contrast alone can safely guide the SFT–RL balance.

Model	In-Distribution (Core Task)					OOD	
	MATH-500	AIME24	AIME25	AMC23	Avg.	MMLU-Pro	GPQA
Qwen2.5-14B							
Qwen2.5-14B	63.4	4.3	3.8	35.8	26.8	55.7	28.3
SFT	81.0	23.6	21.9	60.0	46.6	72.9	32.3
GRPO	82.2	16.8	14.7	62.5	44.1	58.6	36.4
SLS (Ours)	**85.6**	**24.6**	**23.3**	**65.2**	**49.7**	**77.1**	**38.4**
Qwen2.5-7B							
Qwen2.5-7B	57.4	7.1	4.8	36.7	26.5	44.2	25.8
Qwen2.5-7B-Instruct	77.0	10.5	7.1	50.8	36.4	62.9	34.3
SFT	69.8	12.6	**14.1**	40.0	34.1	50.0	18.1
GRPO	75.4	12.1	6.6	52.0	36.5	57.6	33.8
SLS (Ours)	**82.0**	**15.0**	12.8	**56.9**	**41.7**	**64.3**	**34.8**
Qwen3-4B-Base							
Qwen3-4B-Base	69.4	9.7	7.6	45.0	32.9	54.2	32.8
SFT	75.0	16.8	**19.1**	51.4	40.6	62.9	24.7
GRPO	82.6	19.0	15.1	60.1	44.2	68.6	39.9
ReLIFT	83.5	18.6	15.4	59.7	44.3	66.4	40.9
CHORD	83.1	18.9	15.2	61.4	44.7	69.8	41.3
SLS (Ours)	**84.4**	**19.1**	16.3	**62.3**	**45.5**	**72.9**	**42.4**

Table 8: Results on larger-scale and additional base models. We report results on **Qwen2.5-14B** to evaluate scaling behavior, and on **Qwen2.5-7B** and **Qwen3-4B-Base** to evaluate transfer beyond the Qwen2.5-Math family. **Bold** and underlined denote the best and second-best results, respectively.

D.6 Results on Additional Base Models

M-linear. The M-linear variant keeps the multiplicative structure but replaces the nonlinear mastery gate $e^{M_i} - 1$ with a linear gate M_i:

$$\alpha_i = \min\left(1,\ \lambda M_i \frac{D_i}{D_{\max}}\right).$$

This variant isolates the effect of nonlinear mastery gating.

SLS. The full SLS variant uses the proposed nonlinear mastery gate and normalized trajectory dispersion:

$$\alpha_i = \min\left(1,\ \lambda(e^{M_i} - 1)\frac{D_i}{D_{\max}}\right).$$

This variant combines competence gating with reward-contrast estimation.

E Additional Results and Ablations

This section reports additional results across model scales and families and examines the effect of data-mixing choices.

To assess the generality of SLS beyond Qwen2.5-Math, we evaluate it across different model scales and model families, including **Qwen2.5-14B**, **Qwen2.5-7B**, and **Qwen3-4B-Base**. As shown in Table 8, SLS shows trends consistent with the main experiments. On Qwen2.5-7B, SLS achieves the strongest OOD performance while remaining competitive on in-distribution benchmarks. On Qwen3-4B-Base, it obtains the best in-distribution average and the strongest OOD results, outperforming SFT, RL, and hybrid baselines such as **ReLIFT** and **CHORD**. Results on Qwen2.5-14B further indicate that the benefits of SLS persist as model capacity increases.

E.1 Robustness to Data Mixing Configurations

The two training sources exhibit a clear difficulty gap: on the base model, OpenR1-Math achieves only 12.28% accuracy, whereas GSM8K reaches 57.43%. To examine whether SLS depends on the order and proportion of training samples, we compare three data mixing strategies: (1) random sampling, (2) curriculum learning from easier to harder samples, and (3) uniform mixing. As shown in Table 9, the three strategies obtain the same in-distribution average accuracy of 40.6%, with only modest variation in OOD performance and training steps. Unless otherwise stated, we use uniform mixing as the default configuration.

These results indicate that the effectiveness of

Mixing Strategy	In-Distribution (Core Task)					OOD		Training
	MATH-500	AIME24	AIME25	AMC23	Avg.	MMLU-Pro	GPQA	Steps
Random Sampling	79.6	16.4	12.2	54.0	40.6	40.0	30.3	744
Curriculum Learning	79.8	14.7	13.1	54.6	40.6	38.5	27.8	754
Uniform (SLS)	80.2	14.8	13.3	54.2	40.6	42.9	30.8	771

Table 9: Impact of data mixing configurations on Qwen2.5-Math-1.5B. All variants use the same training sources and differ only in the sampling schedule.

SLS mainly comes from sample-level adaptive weighting rather than from a carefully tuned data curriculum. By estimating the learning state of each sample under the current policy, SLS dynamically adjusts the SFT–RL balance during training, reducing the need for manually designed curriculum-learning or data-mixing schedules.

F Qualitative Case Studies

This section provides qualitative examples that illustrate the failure modes observed in the multi-reward robustness experiments and the corresponding behavior of SLS under the same auxiliary objectives.

F.1 Failure Modes under Multi-Reward Settings

Appendix C.1 and Appendix C.2 specify the prompts and reward definitions used in the multi-reward settings. We present two representative failure cases corresponding to the patterns in Table 3. Response lengths are measured by the tokenizer used for model generation.

Shared Query. If A and B are vertices of a polyhedron, define the distance $d(A, B)$ to be the minimum number of edges of the polyhedron one must traverse in order to connect A and B. For example, if $\overline{AB}$ is an edge of the polyhedron, then $d(A, B) = 1$, but if $\overline{AC}$ and $\overline{CB}$ are edges and $\overline{AB}$ is not an edge, then $d(A, B) = 2$. Let Q, R, and S be randomly chosen distinct vertices of a regular icosahedron (regular polyhedron made up of 20 equilateral triangles). Find the probability that $d(Q, R) > d(R, S)$. The final answer can be written in the form $\frac{m}{n}$, where m and n are relatively prime positive integers. What is $m + n$?
Problem type. Combinatorics
Gold answer. $\frac{7}{22}$, hence $m + n = 29$.

GRPO reward-hacking case under the Format setting. Under the Format setting, pure RL can obtain the auxiliary format reward by reproducing the required response template rather than solving the task. A representative GRPO reward-hacking output is shown below.

Method: GRPO
Response length: ~40 tokens
Output:

```
You are a helpful AI assistant that provides detailed thought processes and correct answers. Please respond in the following format:
<think> [Your step-by-step thinking process here] </think>
<answer> [Your clear solution and answer here, with the answer in \boxed{}] </answer>
```

The output matches the required surface structure but contains no task-specific reasoning or valid final answer. It can therefore receive a high format reward while failing the core task, illustrating the reward-hacking collapse observed for GRPO in Table 3.

CHORD label-collapse case under the Type setting. Under the Type setting, the model receives an auxiliary reward for predicting the problem category. This objective can lead to label collapse, where the model repeatedly emits an easy or dominant category instead of performing problem-specific classification. In our observation, CHORD collapses toward predicting *Primary Math* across queries. For the shared query above, this prediction is incorrect; the problem is a combinatorics problem. CHORD also produces inconsistent reasoning, as shown below.

Method: CHORD
Response length: ~203 tokens
Output excerpt:

```
This is a Primary Math question.
. . . the possible distances are 1, 2, 3, 4, and 5 edges . . .
. . . later: the possible distances are 1, 2, 3, and 4 . . .
. . . exact calculation shows that the probability is \boxed{1} . . .
```

In this example, CHORD fails both the auxiliary type prediction and the core mathematical reasoning, consistent with the degradation observed under auxiliary rewards.

F.2 SLS Cases under Multi-Reward Settings

The previous cases show that auxiliary rewards can dominate optimization when reward-driven updates are applied without sufficient sample-level safeguards. In contrast, SLS adjusts the SFT–RL balance according to the current sample learning state, allowing auxiliary rewards to improve response preferences without overriding core-task correctness.

SLS under the Format setting. Under the Format setting, SLS satisfies the required response structure while preserving task-grounded reasoning. The model follows the <think> and <answer> format, but the generated content remains tied to the mathematical problem rather than collapsing into template repetition. A representative output excerpt is shown below.

Method: SLS
Response length: ~260 tokens
Output excerpt:
<think>
. . . for each fixed R, there are 35 valid ordered pairs (Q, S) out of 110 total . . .
. . . the probability is $35/110 = 7/22$ . . .
</think>
<answer>
. . . $m+n = 7+22 = 29$, so the final answer is \boxed{29}.
</answer>

This example satisfies the auxiliary format preference without collapsing into template repetition. Unlike the GRPO failure case, the output contains task-specific reasoning and reaches the correct final answer.

SLS under the Type setting. Under the Type setting, SLS predicts the correct problem category and preserves the core mathematical solution.

Method: SLS
Response length: ~171 tokens
Output excerpt:
This is a Combinatorics question.
. . . for each fixed R, there are 35 valid ordered pairs (Q, S) . . .
. . . the probability is $7/22$, and therefore \boxed{$m + n = 29$} . . .

In this example, SLS preserves both the auxiliary type prediction and the final answer. Together with the Format-setting case, these examples illustrate that SLS can satisfy auxiliary preferences while maintaining core-task performance, whereas GRPO and CHORD are more susceptible to auxiliary-objective interference.