Mitigating Forgetting Between Supervised and Reinforcement Learning Yields Stronger Reasoners

Anonymous Authors[1]

Abstract

State-of-the-art (SOTA) reasoning post-training pipelines increasingly combine supervised fine-tuning (SFT) with reinforcement learning (RL) to balance external knowledge injection and self-exploration. In this work, we identify a critical but underexplored failure mode of these SFT+RL pipelines: *catastrophic forgetting*. To analyze the cause, we reveal a fundamental asymmetry between SFT and RL. SFT typically induces much larger, redundant parameter updates than RL, while RL's updates are smaller and more parsimonious. Therefore, SFT updates can easily overwrite RL's small non-robust updates, causing the forgetting of the knowledge learned by RL. To address this issue, we propose **MIFO** (**MI**tigating **FO**rgetting between SFT and RL), a plug-and-play framework that integrates SFT into RL while explicitly constraining SFT-induced updates. MIFO mitigates forgetting through two complementary designs: *(i) Data selection*, which interleaves RL and SFT by selecting challenging samples based on rollout accuracy and optimizing only high-entropy tokens for SFT. *(ii) Parameter freezing*, which identifies RL-critical parameters and temporarily freezes them during SFT to prevent overwrite. Beyond stability, MIFO is data- and response-efficient and remains agnostic to the underlying SFT or RL algorithm. Empirically, MIFO achieves SOTA reasoning performance using only 1.5% of the SFT data and 20.4% of the RL data required by prior SOTA methods, while producing reasoning traces up to 10× shorter than the standard SFT→RL pipeline.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Recent Large Language Models (LLMs) have shown reasoning capability (Jaech et al., 2024; Guo et al., 2025; Anthropic, 2025). The reasoning capability are highly dependent on the use of the Chain-of-Thought (CoT) thinking pattern trained by supervise fine-tuning (SFT) or reinforcement learning (RL). Although popular RL algorithms such as PPO (Schulman et al., 2017), GRPO (Guo et al., 2025), and DAPO (Yu et al., 2025) are promising in multiple reasoning tasks, recent studies argue that RL training does not truly extend a model's reasoning boundaries. Because RL trains the LLMs based on self-generated rollout samples, RL primarily reshapes the model's internal probability distribution rather than enabling the acquisition of new knowledge (Yue et al., 2025; Wang et al., 2025b). Compared with the weaknesses of RL, SFT can import external out-of-distribution samples into the model, although it carries risks of memorization or overfitting (Chu et al., 2025) and reduced generalization (Yuan et al., 2025).

Consequently, except for the common recipe (Yang et al., 2024; Shao et al., 2024) that uses SFT then RL to post-train the model, many recent studies with better reasoning performances (Chen et al., 2025; Yan et al., 2025) explore how to leverage the strengths of SFT and RL while avoiding their drawbacks by strategically combining them into an interconnected RL–SFT optimization process to balance knowledge injection with self-exploration. For example, DeepSeek-Math (Shao et al., 2024) applies multiple SFT→RL stages, ReLIFT (Ma et al., 2025) alternates between RL and SFT, and LUFFY (Yan et al., 2025) and SRFT (Fu et al., 2025) merge SFT supervision into a single RL optimization objective. Despite their empirical success, we identify a critical but previously overlooked issue shared by these approaches: combining SFT and RL in reasoning post-training can induce *catastrophic forgetting.* Through systematic analysis, we find that this forgetting arises from a fundamental asymmetry between SFT and RL updates. SFT induces parameter updates with substantially larger magnitudes and redundancy, which overwrite the smaller, more parsimonious updates learned during RL. As a result, knowledge acquired through self-exploration in RL can be rapidly erased once SFT is introduced, leading to a degradation in reasoning

performance (forgetting happens). Except for catastrophic forgetting, these methods also suffer from one or more of the following limitations: *(i)* reliance on large quantities of high-quality SFT data, *(ii)* tight coupling to specific RL algorithms, and *(iii)* not efficient with long Chain-of-Thought responses, as summarized in Table 1.

In light of this, we propose **MIFO** (**MI**tigating **FO**rgetting between SFT and RL), a plug-and-play reasoning post-training framework explicitly designed to prevent catastrophic forgetting. The core principle of MIFO is simple yet effective: *combine SFT and RL while constraining SFT-induced parameter updates, thereby preserving the knowledge acquired through RL.* MIFO achieves this through two complementary mechanisms. First, we dynamically interleave RL and SFT, selecting only a small subset of difficult questions for SFT based on RL rollout accuracy, and further restricting the SFT objective to high-entropy tokens. This design substantially reduces unnecessary SFT updates while still enabling effective knowledge injection. Second, we track parameters that are critical to RL learning and temporarily freeze them during subsequent SFT phases, preventing SFT from overwriting RL-acquired knowledge. Together, these mechanisms directly target the root cause of forgetting in SFT+RL pipelines.

Beyond mitigating forgetting, MIFO exhibits several desirable properties simultaneously. First, it is data-efficient, achieving the best reasoning performance using only 1.5% of the SFT data and 20.4% of the RL data required by prior SOTA methods. Second, it is response-efficient, producing concise reasoning traces with 10× fewer tokens compared to the representative SFT→RL pipeline. Finally, because MIFO does not rely on a unified SFT-RL objective (Yan et al., 2025; Fu et al., 2025), it is algorithm-agnostic and can be seamlessly integrated with different SFT or RL algorithms as a plug-and-play module.

In summary, this work makes the following contributions:

- We first identify and empirically demonstrate catastrophic forgetting as a fundamental issue in existing combining SFT+RL reasoning post-training pipelines. Then systematically provide an interpretation via parameter updating perspective.
- We propose **MIFO**, a framework that contains data processing and RL-aware parameter freezing, to mitigate forgetting by constraining SFT-induced larger, redundant parameter updates, protecting RL's smaller, parsimonious parameter updates.
- We show that **MIFO** achieves SOTA reasoning performance with substantially improved data and response efficiency, while remaining algorithm-agnostic and easy to integrate into existing SFT and RL algorithms.

Table 1. Feature comparison of different SFT+RL frameworks. MIFO is the only framework that simultaneously achieves data efficiency, algorithm agnosticism, response efficiency, while most importantly, effectively mitigating catastrophic forgetting.

Method	Data Efficient	Algorithm-Agnostic	Response-Efficient	Mitigate Forgetting
SFT → RL	✗	✓	✗	✗
DeepSeekMath (Shao et al., 2024)	✗	✓	-	✗
LUFFY (Yan et al., 2025)	✗	✗	✓	✗
SRFT (Fu et al., 2025)	✗	✗	✓✓	✗
ReLIFT (Ma et al., 2025)	✓	✓	✓✓	✗
MIFO (Ours)	✓	✓	✓✓	✓

2. Forgetting between SFT and RL

In this section, we first illustrate that existing combining SFT and RL post-training pipelines introduce the risk of catastrophic forgetting, especially when SFT follows RL. Then, we hypothesize that this phenomenon is caused by SFT updates overwriting RL learned information because: *(i)* SFT applies much larger updates than RL; *(ii)* RL updates parameters with much less redundancy compared with SFT. If some RL updates are randomly removed or interfered with, it will experience notable performance degradation without reliable robustness. Finally, we propose our motivation according to these properties to avoid forgetting.

2.1. Forgetting Measurement

Recent SOTA reasoning post-training pipelines, such as DeepSeekMath (Shao et al., 2024), LUFFY (Yan et al., 2025), ReLIFT (Ma et al., 2025), and SRFT (Fu et al., 2025), integrate SFT and RL together throughout the training process, either explicitly or implicitly. However, we identify that within these combinations, the SFT component induces the forgetting of information previously learned via RL component. We quantitatively measure this by first training a model for 50 steps using RL, and subsequently shifting to these pipelines' training objectives containing SFT for 10 steps. If no forgetting happens, the reasoning performance will increase after SFT. However, as shown in Table 2, all other methods exhibit a degradation in performance except for our MIFO, indicating the occurrence of forgetting.

2.2. Why Forgets? Contrasting the Roles of SFT and RL

We hypothesize that this degradation occurs because SFT induces much larger and redundant magnitudes of parameter updates, thereby overwriting RL-induced smaller, parsimonious (non-redundant) updates and triggering catastrophic forgetting of knowledge learned by RL.

SFT updates more, RL Less. Using the same configuration as Section 4.1 with epochs set to 1, we measure the *magnitude of parameter updates* for SFT or RL from the start step 0 to the end of training in step T with the same learning rate, defined as $\Delta\theta = \|\theta_T - \theta_0\|_2$. Figure 1 shows that across all layers, SFT produces much larger parameter changes than

Table 2. The changes (Δ) are relative to the 50-step RL Baseline. RL and MIFO perform best while forgetting happens to others.

Method	*Baseline*	RL	RL→SFT	LUFFY	SRFT	ReLIFT	MIFO
Score	*24.3*	**25.4**	8.7	11.7	15.7	13.2	25.2
Δ	–	**+1.1**	-15.6	-12.6	-8.6	-11.1	**+0.9**

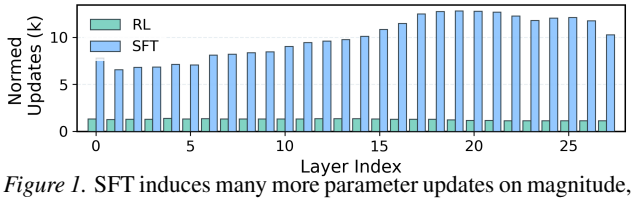

Figure 1. SFT induces many more parameter updates on magnitude, compared with RL.

RL, which can overwrite RL updates and cause catastrophic forgetting of RL-acquired information.

SFT redundant, RL parsimonious. We conduct two experiments to demonstrate that SFT exhibits greater redundancy in parameter updates compared to RL. We set the same learning rate and data amount, epochs $= 1$, and other configurations are the same as in Appendix C. We study two sparsification ways on model updating by observing model performance (average pass@1 on 6 reasoning test sets). *(i) Online gradient sparsification.* Let $p_{\text{on}} \in [0, 1]$ denote the drop rate per step. At each gradient step t, we have model parameters $\theta_t \in \mathbb{R}^d$ with loss gradient $g_t = \nabla_\theta \mathcal{L}(\theta_t)$. We draw an i.i.d. coordinate mask $m_t \sim \text{Bernoulli}(1 - p_{\text{on}})^d$ (1=*keep*, 0=*drop*) and update with the masked gradient

$$\tilde{g}_t \;=\; m_t \odot g_t, \qquad \theta_{t+1} \;=\; \text{AdamW}(\theta_t, \tilde{g}_t),$$

where AdamW represents one AdamW optimization step. In our runs we set $p_{\text{on}} = 0.5$, randomly drop 50 percent of gradient coordinates each step, independently across steps and coordinates. *(ii) Post-hoc update sparsification.* Let $p_{\text{post}} \in [0, 1]$ denote the pruning rate applied once after training. First train unperturbed for T steps to obtain $\theta_0 \to \theta_T$, define the net update $\Delta\theta = \theta_T - \theta_0$, then sample a single mask $m \sim \text{Bernoulli}(1 - p_{\text{post}})^d$ and form

$$\theta_T^{\text{post}} \;=\; \theta_0 \;+\; m \odot \Delta\theta,$$

to randomly prune a p_{post}-fraction of parameter changes.

Both experiments indicate that SFT produces redundant parameter updates, whereas RL induces more non-redundant, parsimonious ones. *(i) Online gradient sparsification.* As shown in Figure 2, the SFT w. Gradient Drop model matches the SFT baseline after roughly the 40th step. By contrast, the RL w. Gradient Drop model suffers a larger performance drop relative to the RL baseline, and the gap only begins to the 110th step. It the end, the average score gap is 1.07 for SFT versus 4.23 for RL. *(ii) Post-hoc update sparsification.* Figure 3 demonstrates that RL performance degrades

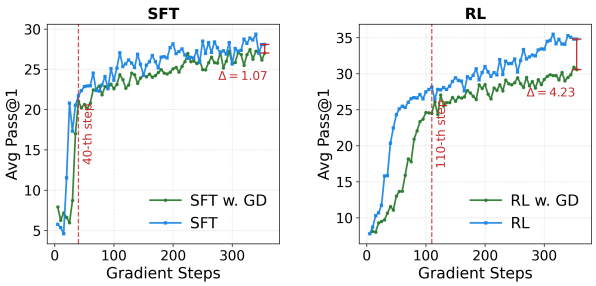

Figure 2. Gradients Dropping (GD) during training causes more performance drop on RL (right) compared with SFT (left).

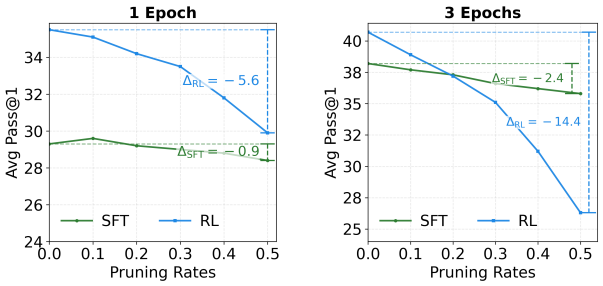

Figure 3. SFT shows more redundancy compared with RL, as it experiences less performance drop if they are randomly pruned.

sharply as the pruning rate increases (Δ=-5.6, -14.4). Conversely, SFT exhibits signs of redundancy and overfitting, where slight pruning initially boosts performance, resulting in a negligible decline (Δ=-0.9, -2.4).

Summary. The contrast in update magnitudes and redundancy offers an explanation for the catastrophic forgetting. RL's updates are smaller and parsimonious: the reasoning capabilities learned by RL rely on precise, non-redundant parameter updates, and therefore are not robust when combined with SFT's high redundancy and larger updates during post-training. Consequently, when SFT is combined with RL, its broad updates easily interfere with or forget the RL's non-redundant, small parameter updates. This suggests that the full potential of SFT+RL training pipelines remains untapped because they did not explicitly mitigate forgetting risk. Besides, experiments in Section 4.3 further show that, by decreasing the SFT's larger, redundant updates while preserving RL's smaller, non-redundant updates in combined training, the reasoning performance is improved, indicating the existence of original catastrophic forgetting. For our empirical results, we also provide a theoretical insights for the redundancy difference in Appendix B.

2.3. Motivation: Mitigates Forgetting

When SFT overwrites RL due to larger, redundant updates, can we reduce its update magnitude to mitigate this forgetting? Section 2.2 shows that SFT exhibits substantial redun-

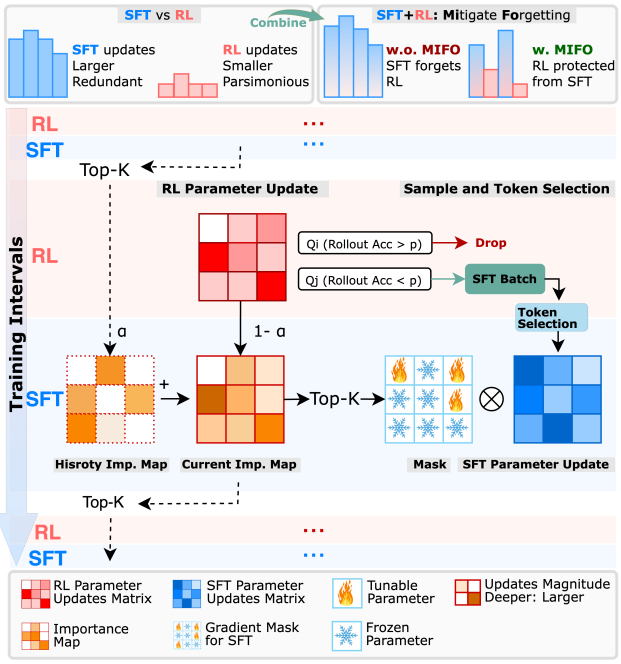

Figure 4. **Overview.** MIFO is a training pipeline with multiple connected RL→ SFT intervals. In **RL** training: it selects data for SFT and decides the important RL parameters at the end; In **SFT** training: RL-important parameters are frozen, and only selected data samples and high-entropy tokens are used for loss calculation.

dancy in parameter updates relative to RL and carries a risk of overfitting. Our motivation follows directly: *We combine SFT with RL for reasoning post-training while constraining SFT-induced parameter updates to reduce catastrophic forgetting on the information learned by RL.*

3. Methodology

Based on the insights in Section 2, we propose **MIFO** as illustrated in Figure 4, which incorporates multiple connected RL→ SFT intervals and mitigates forgetting. To achieve it, our principle is to limit SFT parameter updates, thereby reserving parameter space for the more parsimonious, non-robust updates of RL. Guided by this, MIFO introduces two key designs: *(i) Data processing*. At the sample level, we use RL rollouts to identify questions the model currently struggles with, and we curate an online SFT set containing only these cases with verified solutions. At the token level, we focus the SFT objective on the most informative, high-uncertainty tokens rather than spreading gradients over easy tokens. Therefore, less SFT data samples and tokens after filtering decrease the magnitudes of SFT updates. *(ii) Parameter freezing*. To directly reduce forgetting by limiting SFT updates and protect RL learned information, we track which parameters were most updated during RL and temporarily freeze these RL-critical components during subsequent SFT, then unfreeze them before the next RL step.

3.1. Data Processing

Selecting SFT Samples. We adopt GRPO (can be changed to another RL algorithm) for RL training, and select challenging data samples according to the GRPO rollout accuracy for the following SFT training. This design limits SFT updates with less samples while still keeps challenging data to push the reasoning boundary of the model. Let the policy model before and after the update be $\pi_{\theta_{\text{old}}}$ and π_θ. Given a question q, a set of solutions $\{o_i\}_{i=1}^N$ generated by $\pi_{\theta_{\text{old}}}$, and a reward function $R(\cdot)$, the objective is

$$\mathcal{L}_{\text{RL}}(\theta) = \frac{1}{\sum_{i=1}^{N}|o_i|}\sum_{i=1}^{N}\sum_{t=1}^{|o_i|} \min\Big[r_{i,t}(\theta)\,A_i,\ \text{clip}\big(r_{i,t}(\theta),\,1-\epsilon,\,1+\epsilon\big)\,A_i\Big].$$

where $r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q,o_{i<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i<t})}$, $A_i = \frac{R(o_i)-\text{mean}\left(\{R(o_j)\}_{j=1}^N\right)}{\text{std}\left(\{R(o_j)\}_{j=1}^N\right)}$. For each question q, GRPO produces N solutions and yields an accuracy $\text{acc}(q)$. We then collect questions with $\text{acc}(q) \le p$, where the policy exhibits a low success rate under self-exploration, we assume that additional out-of-distribution knowledge is required. We obtain high-quality SFT solutions which can be from stronger reasoning models or from human experts, and accumulate them into an SFT batch $\mathcal{B}_{\text{FT}}$:

$$\mathcal{B}_{\text{FT}} = \Big\{(q,s)\ \Big|\ \text{acc}(q) \le p, s = D(q), \text{extract}(s) = a\Big\},$$

where $D(q)$ denotes a CoT solution obtained either offline or online from an external model or a human annotator, $\text{extract}(s)$ is the final answer extracted from s, and a is the ground-truth answer for q. We retain only pairs satisfying $\text{extract}(s) = a$. Once the batch size exceeds a preset threshold S ($|\mathcal{B}_{\text{FT}}| \ge S$), training shifts from RL to SFT. Subsequently, the process reverts to RL once the current SFT batch is fully consumed, and the batch is emptied ($\mathcal{B}_{\text{FT}}$=$\emptyset$).

SFT with High-entropy Tokens. After RL, MIFO SFT the model only on high-entropy tokens from SFT batch $\mathcal{B}_{\text{FT}}$ to limit the magnitude of parameter updates. Entropy (Shannon, 1948) describes the uncertainty, and high-entropy tokens mark positions of model uncertainty. Concentrating learning on these tokens encourages acquisition of missing knowledge while avoiding unnecessary fitting of low-entropy (confident) tokens. For each token position t in a response, define the entropy $H_t = -\sum_{j=1}^{V} p_{t,j}\,\log p_{t,j}$, where $\mathbf{p}_t = \pi_\theta(\cdot \mid q, s_{<t})$, and V is the vocabulary size. We then compute an example-specific threshold $\tau_\rho(q,s)$ such that the top $\rho \in (0,1]$ fraction of tokens by entropy in (q,s) satisfy $H_t \ge \tau_\rho(q,s)$. The SFT objective minimizes

loss only over these high-entropy tokens:

$$\mathcal{L}_{\text{SFT}}(\theta) \;=\; -\frac{1}{|s|}\sum_{t=1}^{|s|}\mathbb{I}[H_t \geq \tau_\rho(q,s)]\,\log \pi_\theta(s_t \mid q, s_{<t}),$$

where (q, s) is sampled from SFT batch $\mathcal{B}_{\text{FT}}$.

3.2. Parameter Freezing

The second component of MIFO tracks RL-induced parameter updates and freezes the most important ones based on historical magnitude before the subsequent SFT phase. This protects important RL-acquired knowledge, thereby preventing catastrophic forgetting. Simultaneously, because SFT updates are inherently redundant, the remaining unfrozen parameters provide sufficient capacity for SFT to learn effectively, ensuring reasoning performance is maintained even with a constrained parameter space. Let $\boldsymbol{\theta}_i = \{\theta_i^1, \ldots, \theta_i^d\}$ denotes the model parameters in the i-th RL→SFT interval, where d is the number of parameters. To reduce forgetting, we maintain a history importance map $\mathbf{C}_i \in \mathbb{R}^d$ that tracks important RL updates, and we freeze parameters with the largest importance scores for the following SFT. Let $\boldsymbol{\theta}_{\text{RL,start,i}}$ and $\boldsymbol{\theta}_{\text{RL,end,i}}$ be the parameters at the start and end of the RL session in interval i. Considering the connection between parameter importance and magnitudes, we first define the magnitudes of RL update $\Delta\boldsymbol{\theta}_{\text{RL},i}$ by

$$\Delta\boldsymbol{\theta}^j_{\text{RL},i} = \left\|\theta^j_{\text{RL,end},i} - \theta^j_{\text{RL,start},i}\right\|_2, \quad j = 1, \ldots, d.$$

Then with $\mathbf{C}_0 = \mathbf{0}$ initialization, let $\alpha \in [0, 1)$ be the history coefficient and define the current importance map as

$$\tilde{\mathbf{C}}_i \;=\; \alpha\,\mathbf{C}_{i-1} \;+\; (1-\alpha)\,\Delta\boldsymbol{\theta}_{\text{RL},i}.$$

Here $\mathbf{C}$ serves as history importance map for RL parameter updates, with global information along the training. We then select the top-k entries by magnitude to find the most-learned important parameters during RL training, $\mathcal{I}_i \;=\; \text{TopK}\big(\tilde{\mathbf{C}}_i,\, k\big)$, and define a binary mask $\mathbf{M}_i \in \{0, 1\}^d$ with

$$\mathbf{M}_i^j = \mathbf{1}[\,j \in \mathcal{I}_i\,], \qquad \mathbf{C}_i \;=\; \mathbf{M}_i \odot \tilde{\mathbf{C}}_i,$$

New obtained masked $\mathbf{C}_i$ is then passed to the next interval as history to calculate next importance map $\mathcal{I}_{i+1}$. To avoid forgetting in subsequent SFT, we freeze RL-important parameters indexed by $\mathcal{I}_i$ by zeroing their SFT gradients:

$$\nabla_{\boldsymbol{\theta}_i}\mathcal{L}_{\text{SFT}} \;\leftarrow\; (\mathbf{1} - \mathbf{M}_i) \odot \nabla_{\boldsymbol{\theta}_i}\mathcal{L}_{\text{SFT}}.$$

After SFT completes, all parameters are unfrozen for the next RL session. RL updates produce smaller parsimonious updates, which almost do not induce forgetting of SFT-acquired knowledge. This RL→SFT cycle repeats throughout post-training, promoting long-term training stability. When $\alpha = 0$, we denote the variant as $\mathbf{MIFO}^\dagger$, meaning no historical RL-importance is carried across intervals. Pseudocode of MIFO is provided in Algorithm 1.

4. Experiment

4.1. Setups

Evaluation. We evaluate MIFO on five widely used mathematical reasoning benchmarks: AIME 2024 (A-24), AIME 2025 (A-25), AMC (Li et al., 2024), OlympiadBench (He et al., 2024) (Olym), and MATH500 (Hendrycks et al., 2021) (MATH). For the smaller test sets (AIME 2024, AIME 2025, and AMC), we report *avg@32* (*pass@1* for Section 2). For the larger test sets (OlympiadBench and MATH500), we report *pass@1*. We also include MMLU-Pro (Wang et al., 2024) (M-Pro) to assess the model's generalized reasoning ability. Then we summarize the average score (Avg.) and average response length (Len.).

Datasets. For the training set, we utilize the version of the dataset OpenR1-Math-46k from (Yan et al., 2025), which contains a subset of OpenR1-Math-220k (Face, 2025), with prompts from NuminaMath (Li et al., 2024) in 45.8k prompts and their CoT demonstrations.

Baselines. We compare against four categories of baselines. (i) **Base models**: **Qwen-Math-2.5** and **Qwen-Math-2.5-Instruct**. (ii) **SFT-only**: **SFT**, a supervised fine-tuned variant of Qwen-Math-2.5. (iii) **RL-only**: **RL**, which is trained by vanilla GRPO; **Oat-Zero** (Liu et al., 2025c), which uses GRPO with variance removal in advantage computation and token-level normalization; **PRIME-Zero** (Cui et al., 2025), which employs implicit process rewards; **Open-ReasonerZero** (Hu et al., 2025), which uses rule-based rewards. (iv) **Joint SFT+RL**: **SFT→RL**, which first applies SFT then runs RL; **LUFFY** (Yan et al., 2025), which integrates on-policy rollouts with off-policy reasoning traces within RL training. **SRFT**(Fu et al., 2025), a single-stage method that unifies both fine-tuning paradigms through weighting mechanisms (only the 7B model accessible).

We provide other implementation details in Appendix C.

4.2. Main Results

Table 3 presents the main results regarding reasoning performance, response efficiency, and training data efficiency for baselines, our method **MIFO**, and $\mathbf{MIFO}^\dagger$. $\mathbf{MIFO}^\dagger$ is a special case when the history coefficient α is set to 0.

Reasoning Performance. Table 3 shows that across five competition-level benchmarks and one out-of-distribution benchmark, our method consistently attains the best or second-best results. On the 1.5B model, it yields large gains on AMC (+4.0 over LUFFY) and OlympiadBench (+7.9 over ReLIFT). On the 7B model, it substantially outperforms SRFT on AIME 2025 (+5.8) and OlympiadBench (+1.5). Overall, our approach achieves the best average reasoning performance for both 1.5B and 7B models, high-

Table 3. Overall reasoning performances on reasoning benchmarks (Avg. denotes average) ↑ and average response length (Len., #token) ↓. **Bold** and underline indicate the best and second-best results, respectively.

	Qwen2.5-Math-1.5B								Qwen2.5-Math-7B							
Model	**A-24**	**A-25**	**AMC**	**MATH**	**Olym**	**M-Pro**	**Avg.**	**Len.**	**A-24**	**A-25**	**AMC**	**MATH**	**Olym**	**M-Pro**	**Avg.**	**Len.**
Base																
Qwen-Math	3.0	1.4	19.4	44.4	16.7	5.0	15.0	4043	5.1	1.4	22.6	37.2	19.0	30.5	19.3	2408
Qwen-Math-Instruct	8.1	6.6	36.9	66.0	30.7	24.2	28.8	5806	9.7	8.2	39.9	77.2	34.1	33.0	33.7	14179
SFT																
SFT	12.7	13.0	40.9	71.8	33.8	23.5	32.6	13403	26.9	23.1	58.9	85.0	50.7	48.6	48.9	10166
RL																
RL	10.2	8.7	44.7	71.4	34.5	28.7	33.0	**913**	22.1	13.9	59.5	84.2	44.7	48.3	45.5	**653**
Oat-Zero	17.8	10.8	47.2	73.0	35.8	20.9	34.3	2605	33.4	11.9	61.2	78.0	43.4	41.7	44.9	1767
OpenReasoner	3.2	1.3	27.6	55.2	24.0	27.9	23.2	2445	16.5	15.0	52.1	82.4	47.1	58.7	45.3	1652
SFT+RL																
SFT→RL	12.5	9.2	43.4	72.0	34.9	29.0	33.5	12601	29.6	19.6	61.1	84.2	51.6	49.6	49.3	10832
ReLIFT	13.1	8.8	42.5	73.6	36.0	30.1	34.0	4421	27.2	20.1	64.9	85.2	53.6	52.5	50.6	1299
LUFFY	15.9	**13.1**	46.3	**80.0**	41.0	34.2	38.4	3406	27.2	18.1	61.1	85.6	53.6	52.6	49.7	1762
SRFT	-	-	-	-	-	-	-	-	**33.4**	23.1	65.6	88.6	57.9	**55.8**	54.1	1112
MIFO†	19.0	12.6	49.1	78.0	**43.9**	**36.2**	39.8	985	28.2	**28.9**	65.6	**89.0**	56.4	54.0	53.7	1371
MIFO	**19.2**	12.0	**50.3**	78.8	43.3	36.1	**40.0**	2518	31.7	26.0	**66.2**	87.8	**59.4**	54.8	**54.3**	1067

lighting the effectiveness of MIFO.

Data Efficiency. Because of the introduced data processing and parameter freezing design, MIFO maximizes the utility of existing data and thus reduces data requirements. Table 5 compares data usage among joint SFT+RL methods. For 7B model, MIFO uses only 1.5% of the SFT data and 20.4% of the RL data required by the prior SOTA SRFT, and 5.8% of the SFT data and 41.6% of the RL data used by LUFFY. For 1.5B model, MIFO uses only 12% of the SFT data and 41.6% of the RL data used by LUFFY. MIFO employs a similar data budget compared to ReLIFT, and achieves notably better performance.

Response Length Efficiency. Tables 3 show that MIFO yields more concise reasoning traces beyond accuracy. On the 1.5B model, MIFO and MIFO† produce outputs with average lengths of 2,518 and 985 tokens, respectively, shorter than all baselines except the RL-only model. On the 7B model, the gap is even more pronounced: MIFO and MIFO† average 1,067 and 1,371 tokens, respectively, with only the RL-only model shorter at 653 tokens. These results indicate that MIFO improves reasoning while also producing more efficient responses with less tokens.

4.3. Ablation Study

Table 4 reports the ablation study. Starting from the base SFT model, we progressively evaluate three components: interleaved training (Int), entropy-based token selection (ES), and parameter freezing (PF). **+ Interleave.** Simply interleaving SFT with RL-based data provides a substantial baseline boost, increasing the average accuracy from 15.0 to 34.0 for the 1.5B model and from 33.7 to 50.5 for the 7B model. This confirms the fundamental benefit of combining SFT and RL.

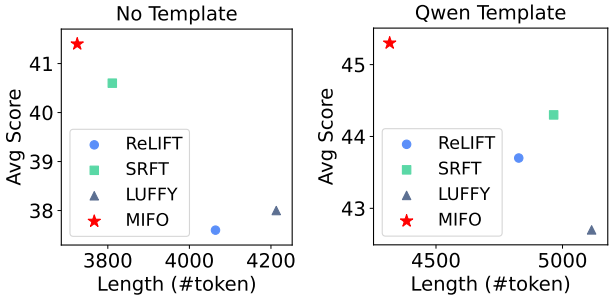

Figure 5. Gradients Dropping (GD) during training causes more performance drop on RL (right) compared with SFT (left).

+ Interleave + ES. Integrating entropy-based selection further improves average performance, such as reaching 39.2 on 1.5B. By focusing training on high-uncertainty tokens, ES effectively reduces redundant SFT updates and enhances reasoning performance. **+ Interleave + PF.** Applying parameter freezing to mitigate SFT–RL interference yields stable gains, particularly on challenging sets like AIME-25, where the 1.5B model achieves 14.5. This suggests that preserving RL learned reasoning capabilities is crucial. **MIFO.** By combining all components, **MIFO** achieves the superior performance across both scales, reaching an average of 40.0 (1.5B) and 54.3 (7B). MIFO consistently delivers the best or second-best results on nearly all benchmarks, demonstrating that ES and PF are highly complementary.

4.4. Analyzing

Templates Ablation. Besides using the LUFFY template (Yan et al., 2025) for results in Section 4.2, we evaluate MIFO under different reasoning templates to assess robustness in real-world deployments. Figure 5 reports performance for MIFO and the strongest baselines with no

Table 4. Ablation study. **Bold** and underline indicate the best and second-best results, respectively.

	Qwen2.5-Math-1.5B							Qwen2.5-Math-7B						
Model	**A-24**	**A-25**	**AMC**	**M-500**	**Olym**	**M-Pro**	**Avg.**	**A-24**	**A-25**	**AMC**	**MATH**	**Olym**	**M-Pro**	**Avg.**
Qwen-Math	3.0	1.4	19.4	44.4	16.7	5.0	15.0	9.7	8.2	39.9	77.2	34.1	33.0	33.7
+ Interleave	13.1	8.8	42.5	73.6	36.0	30.1	34.0	27.5	20.4	65.1	84.0	53.2	52.6	50.5
+ Int. + ES	16.6	12.5	49.1	**79.4**	42.3	35.2	39.2	27.9	24.2	63.3	84.2	55.2	53.5	51.4
+ Int. + PF	15.7	**14.5**	45.8	77.0	41.7	35.9	38.4	30.0	25.0	64.8	87.6	57.2	54.3	53.2
MIFO	**19.2**	12.0	**50.3**	78.8	**43.3**	**36.1**	**40.0**	**31.7**	**26.0**	**66.2**	**87.8**	**59.4**	**54.8**	**54.3**

Table 5. Training data usage (#Samples, K).

#Samples (K)	**SRFT**	**LUFFY**	**SFT+RL**	**ReLIFT**	**MIFO**
Qwen2.5-Math-1.5B					
SFT	-	110.0	45.8	12.3	13.2
RL	-	110.0	45.8	45.8	45.8
Qwen2.5-Math-7B					
SFT	436.5	110.0	45.8	6.6	6.4
RL	225.1	110.0	45.8	45.8	45.8

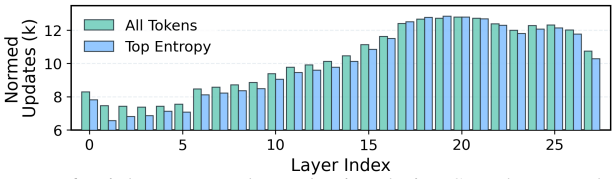

Figure 6. High-entropy token selection during SFT decrease the magnitude of SFT parameter updates.

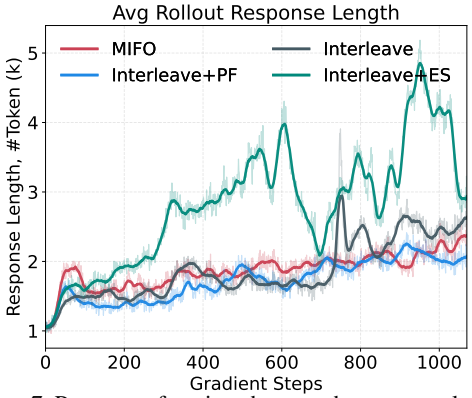

Figure 7. Parameter freezing shortens the response length.

template (left) and with the Qwen system template (right). MIFO's performance drops about 10 points when using simple templates, yet it still surpasses the baselines with best response efficiency. Detailed results by dataset and results with the Prime template are provided in Appendix D.2.

MIFO Mitigates the Forgetting. MIFO is motivated by the observation that SFT produces redundant parameter updates, whereas RL induces more parsimonious updates. We therefore reduce the SFT data size and limit updates by freezing parameters. To further quantify the decreased updates via calculating SFT loss only on high-entropy tokens, we analyze parameter changes under pure SFT with two objectives: (i) loss on full-token and (ii) loss on high-entropy–only, and we plot layer-wise changes. Figure 6 shows that for most layers the high-entropy objective yields noticeably smaller weight updates than the full-token objective. Only layers 17–19 exhibit slightly larger changes under the high-entropy objective. Overall, high-entropy token selection mitigates SFT updates on magnitude.

Parameter Freezing Shortens Response. Mirroring Section 4.3, we conduct an ablation to identify which components improve response efficiency and present results in Figure 7. Comparing **Interleave** with **Interleave+ES**, we observe that adding ES increases rollout length. We hypothesize that emphasizing high-entropy tokens during SFT encourages exploration of alternative reasoning branches (Wang et al., 2025a), which expands the trace. Comparing **Interleave+PF** with **Interleave**, PF yields shorter responses in most training stages, especially in the final epoch (three in total). This suggests that reducing SFT updates also improves efficiency, since SFT data are typically longer due to being sourced from stronger models with longer CoT. Finally, comparing **MIFO** with **Interleave+ES**, we find that PF substantially reduces response length, demonstrating its effectiveness for efficiency.

4.5. Additional Models, Domains, and Algorithms

We conduct extensive experiments to validate the versatility of MIFO across diverse model architectures, reasoning domains, and algorithm combinations. These results validate MIFO as a robust, plug-and-play solution for the SFT-RL forgetting problem, delivering consistent gains.

Scientific and Knowledge Reasoning. Following (Fu et al., 2025; Yan et al., 2025), we utilize scientific and knowledge-based tasks (ARC-c, GPQA, MMLU-Pro) as an out-of-distribution (OOD) evaluation for general reasoning capabilities. As shown in Table 8, MIFO achieves an average score of 61.8% on Qwen2.5-Math-7B. This represents a substantial improvement of 17.0 points over the SFT→RL baseline (44.8%). Notably, MIFO performs comparably to the data-intensive SRFT method (61.9%) while requiring significantly fewer data resources.

Code Generation. We assess coding performance using the Qwen2.5-7B-Instruct-1M model within the Code-

Table 6. Performance comparison on Qwen2.5-7B-Instruct-1M for coding tasks. **Bold** and underline indicate the best and second-best results, respectively.

Datasets	Base	SFT	RL	SFT→RL	MIFO
LiveCodeBench	24.0	24.5	28.6	28.8	**29.1**
HumanEval+	80.5	81.7	84.8	83.5	**86.6**
MBPP+	66.7	66.9	70.1	71.4	**73.0**
Avg	57.1	57.7	61.2	61.2	**62.9**

Table 7. Performance comparison on Llama3.2-8B on math reasoning tasks. **Bold** and underline indicate the best and second-best results, respectively.

Models	A-24	A-25	AMC	MATH	Olym	M-Pro	Avg
SFT	0.8	1.5	11.5	28.6	8.6	28.2	13.2
RL	1.8	0.0	10.8	28.2	7.3	39.4	14.6
SFT→RL	1.3	0.3	11.4	36.4	10.1	43.9	17.2
ReLIFT	1.3	0.2	11.9	35.2	11.0	44.2	17.3
MIFO	**2.1**	**2.5**	**13.9**	**40.6**	**12.3**	**46.6**	**19.7**

R1 pipeline (Liu & Zhang, 2025). We evaluate on LiveCodeBench (LCB) (Jain et al., 2024), HumanEval+, and MBPP+ (Liu et al., 2023). As detailed in Table 6, MIFO demonstrates superior generalization, achieving the highest Pass@1 performance across all benchmarks with an average score of 62.9%, outperforming the SFT→RL by 1.7%.

Model Architecture. To test effectiveness on different architectures, we apply MIFO to Llama-3.2-8B (Table 7). MIFO consistently enhances mathematical reasoning, outperforming the competitive interleaved baseline, ReLIFT, by an average of 2.4 points. We also perform a template robustness analysis in Appendix D.2 for Llama model.

Algorithm Agnosticism. Finally, we demonstrate that MIFO is algorithm-agnostic in Table 9. MIFO yields consistent gains whether combined with SFT variants or other RL methods like DFT (Wu et al., 2025) and DAPO (Yu et al., 2025), consistently surpassing SFT→RL baseline.

In addition to the above experiments, we further provide the experimental results on hyperparameter analysis, training records for dynamic analysis, SFT batch size analysis, additional template analysis, and LoRA baseline in Appendix D, and additional discussions on experimental results, method designs, and baselines in Appendix E.

5. Related Works

RL for Reasoning. RL emerges as an important paradigm to incentivize LLMs' reasoning capability (Jaech et al., 2024; Shao et al., 2024; Team et al., 2025) since SFT is identified having the drawback of memorization effect (Chu et al., 2025; Yuan et al., 2025). Popular approaches like PPO (Schulman et al., 2017), GRPO (Guo et al., 2025), Dr.GRPO (Liu et al., 2025c) and DAPO (Yu et al., 2025)

Table 8. Performance comparison on Qwen2.5-Math-7B for scientific and knowledge reasoning tasks. **Bold** and underline indicate the best and second-best results, respectively.

Datasets	Base	SFT	RL	SFT→RL	LUFFY	SRFT	MIFO
ARC-c	70.3	75.2	83.2	72.4	80.5	**85.3**	83.8
GPQA	24.7	24.7	40.4	24.2	39.9	46.4	**47.4**
MMLU-Pro	34.1	42.7	49.3	37.7	52.6	54.1	**54.3**
Avg	43.0	47.5	57.6	44.8	57.7	**61.9**	61.8

Table 9. Performance comparison on Qwen2.5-Math-1.5B with different SFT and RL algorithm combinations. **Bold** and underline indicate the best and second-best results, respectively.

Algorithms	SFT+GRPO	DFT+GRPO	SFT+DAPO	DFT+DAPO
SFT	23.6	23.1	23.6	23.1
RL	27.5	27.5	27.8	27.8
SFT→RL	27.9	27.3	28.2	27.9
MIFO	**30.7**	**29.2**	**30.5**	**29.7**

have shown the great improvement for reasoning tasks. Recent research also tends to question how RL works. Yue et al. (2025) claims that RL doesn't expand the boundary of reasoning capability but lets the right answer emerge early for multiple tryouts. Wang et al. (2025b) discovers that the main role of RL is to incentivize the knowledge pre-trained to the model. In light of this, our work aims to balance RL and SFT to learn and explore in balance.

Interplaying SFT and RL. Combining SFT with RL has emerged as an effective way for boosting LLM reasoning. Some methods interleave SFT with RL using demonstrations mined during training (Ma et al., 2025), or imitate off-policy traces from stronger teachers (Yan et al., 2025; Fu et al., 2025); others balance the two signals via schedules (SASR) (Chen et al., 2025), prompt-level supervision during rollouts (Liu et al., 2025a), or joint loss formulations (SuperRL) (Liu et al., 2025b). While successful, these approaches have shorts summarized in Table 1.

6. Conclusion

In this work, we identify a critical degradation mode in combining SFT with RL for reasoning post-training: SFT typically applies large, redundant parameter updates that overwrite the smaller, parsimonious knowledge acquired by RL, leading to catastrophic forgetting. To address this, we propose **MIFO**, a plug-and-play framework comprising data selection to minimize redundant SFT gradients, and RL-aware parameter freezing to protect critical parameters from being overwritten. MIFO is algorithm-agnostic and achieves state-of-the-art reasoning performance. Notably, it demonstrates superior efficiency, requiring significantly less training data than prior methods while producing far more concise reasoning traces in responses.

Impact Statement

This paper presents work aiming to improve the reasoning capability of LLMs on reasoning problems. Our study utilizes publicly available training datasets and standard reasoning benchmarks (including AIME, AMC, OlympiadBench, MATH-500, MMLU-Pro, other coding and scientific-based reasoning benchmarks), and does not involve the collection of sensitive or personally identifiable data. We believe that enhancing the performance and efficiency of reasoning in LLMs poses no immediate safety risks and may yield positive societal impacts in the research community. There are no other potential societal consequences of our work that we feel must be specifically highlighted here.

References

Anthropic. Introducing claude 4, 2025. URL https://www.anthropic.com/news/claude-4. Accessed: 2025-07-07.

Baldi, P. and Sadowski, P. J. Understanding dropout. *Advances in neural information processing systems*, 2013.

Ben Zaken, E., Goldberg, Y., and Ravfogel, S. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *SIAM review*, 2018.

Chen, J., Liu, F., Liu, N., Luo, Y., Qin, E., Zheng, H., Dong, T., Zhu, H., Meng, Y., and Wang, X. Step-wise adaptive integration of supervised fine-tuning and reinforcement learning for task-specific llms. *arXiv preprint arXiv:2505.13026*, 2025.

Chu, T., Zhai, Y., Yang, J., Tong, S., Xie, S., Schuurmans, D., Le, Q. V., Levine, S., and Ma, Y. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Cui, G., Yuan, L., Wang, Z., Wang, H., Li, W., He, B., Fan, Y., Yu, T., Xu, Q., Chen, W., et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Face, H. Open r1: A fully open reproduction of deepseek-r1. https://huggingface.co/datasets/open-r1/OpenR1-Math-220k, 2025.

Fu, Y., Chen, T., Chai, J., Wang, X., Tu, S., Yin, G., Lin, W., Zhang, Q., Zhu, Y., and Zhao, D. Srft: A single-stage method with supervised and reinforcement fine-tuning for reasoning. *arXiv preprint arXiv:2506.19767*, 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

He, C., Luo, R., Bai, Y., Hu, S., Thai, Z. L., Shen, J., Hu, J., Han, X., Huang, Y., Zhang, Y., et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, 2019.

Howard, J. and Ruder, S. Universal language model fine-tuning for text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2018.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *The International Conference on Learning Representations*, 2022.

Hu, J., Zhang, Y., Han, Q., Jiang, D., Zhang, X., and Shum, H.-Y. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Lee, J., Tang, R., and Lin, J. What would elsa do? freezing layers during transformer fine-tuning. *arXiv preprint arXiv:1911.03090*, 2019.

Li, J., Beeching, E., Tunstall, L., Lipkin, B., Soletskyi, R., Huang, S., Rasul, K., Yu, L., Jiang, A. Q., Shen, Z., et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 2024.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th*

Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, 2021.

Liu, J. and Zhang, L. Code-r1: Reproducing r1 for code with reliable rewards. 2025.

Liu, J., Xia, C. S., Wang, Y., and Zhang, L. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Liu, M., Farina, G., and Ozdaglar, A. Uft: Unifying supervised and reinforcement fine-tuning. *arXiv preprint arXiv:2505.16984*, 2025a.

Liu, Y., Li, S., Cao, L., Xie, Y., Zhou, M., Dong, H., Ma, X., Han, S., and Zhang, D. Superrl: Reinforcement learning with supervision to boost language model reasoning. *arXiv preprint arXiv:2506.01096*, 2025b.

Liu, Z., Chen, C., Li, W., Qi, P., Pang, T., Du, C., Lee, W. S., and Lin, M. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025c.

Ma, L., Liang, H., Qiang, M., Tang, L., Ma, X., Wong, Z. H., Niu, J., Shen, C., He, R., Cui, B., et al. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. *arXiv preprint arXiv:2506.07527*, 2025.

Rücklé, A., Geigle, G., Glockner, M., Beck, T., Pfeiffer, J., Reimers, N., and Gurevych, I. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shannon, C. E. A mathematical theory of communication. *The Bell system technical journal*, 1948.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 2014.

Team, K., Du, A., Gao, B., Xing, B., Jiang, C., Chen, C., Li, C., Xiao, C., Du, C., Liao, C., et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Wang, S., Yu, L., Gao, C., Zheng, C., Liu, S., Lu, R., Dang, K., Chen, X., Yang, J., Zhang, Z., et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025a.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 2024.

Wang, Y., Yang, Q., Zeng, Z., Ren, L., Liu, L., Peng, B., Cheng, H., He, X., Wang, K., Gao, J., et al. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*, 2025b.

Wu, Y., Zhou, Y., Ziheng, Z., Peng, Y., Ye, X., Hu, X., Zhu, W., Qi, L., Yang, M.-H., and Yang, X. On the generalization of sft: A reinforcement learning perspective with reward rectification. *arXiv preprint arXiv:2508.05629*, 2025.

Yan, J., Li, Y., Hu, Z., Wang, Z., Cui, G., Qu, X., Cheng, Y., and Zhang, Y. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*, 2025.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yuan, X., Zhang, C., Liu, Z., Shi, D., Vosoughi, S., and Lee, W. Superficial self-improved reasoners benefit from model merging. *arXiv preprint arXiv:2503.02103*, 2025.

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Song, S., and Huang, G. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Zhang, C., Bengio, S., and Singer, Y. Are all layers created equal? *Journal of Machine Learning Research*, 2022.

A. Limitation and Future Works

Our study focuses on small (1.5B) and medium (7B) models due to limited computational resources. Scaling the method to larger models is a promising direction for future work. Our approach primarily targets full-parameter RL and SFT because LoRA baselines are weaker than fully finetuning D.4; when applying parameter-efficient fine-tuning (PEFT) if resource is restricted, the freezing strategy may need to be adjusted to prevent catastrophic forgetting between SFT and RL. As stronger SFT and RL algorithms continue to emerge, we release this plug-and-play framework to encourage the community to explore improved post-training reasoning with advanced algorithm combinations.

B. Theoretical Insights

Here we provide theoretical insights trying to explain *why SFT exhibits greater redundancy in parameter updates than RL.* The following discussion and proofs rest on simplifying assumptions intended to interpret our empirical findings, instead of providing a rigorous theoretical guarantee. Because large language models involve extremely high-dimensional parameters and massive training data, fully rigorous modeling is challenging.

B.1. Redundancy Ratio Definition for Parameter Updates

In this section, we introduce Lemma 1 to describe the *parameter–redundancy ratio* (PR) as how much a parameter update exceeds the minimal necessary change to adjust the decision margin of logits for the right token prediction. For a given input and target token, we look at the strongest non-target competitor under the current parameters and measure the logit margin between the target and that competitor. Assuming the margin changes approximately linearly with small parameter moves and the same competitor remains active, there exists a unique, shortest move in parameter space that achieves any desired margin increase; this move points along the margin's gradient. PR is then defined as the size of the actual parameter change divided by the size of the shortest move. A larger PR means the training trajectory moved farther than necessary to achieve the same decision margin, indicating more redundant parameter updating.

Lemma 1. *Fix a context x and target token $y \in \{1, \dots, V\}$ with vocabulary size V. Let $\theta \in \mathbb{R}^d$ be model parameters and θ_0 a reference point (e.g., pre-update). For logits $z_\theta(x) \in \mathbb{R}^V$ and probabilities $p_\theta = \mathrm{softmax}(z_\theta)$, define the active competitor at θ_0 as $k^\star = \arg\max_{k \neq y} z_{\theta_0,k}(x)$. The (logit) decision margin is $m(\theta) = z_{\theta,y}(x) - z_{\theta,k^\star}(x)$, with $m_0 = m(\theta_0)$, and its parameter gradient is $g = \nabla_\theta m(\theta_0) \in \mathbb{R}^d$. Assume (A1) local linearity $m(\theta_0 + \Delta) \approx m_0 + g^\top \Delta$ for small $\Delta \in \mathbb{R}^d$. Then for any desired margin $\varepsilon > 0$, the minimal parameter displacement $\Delta^\star$ that achieves margin ε is*

$$\Delta^\star = \arg\min_{\Delta} \{\|\Delta\|_2 : \; g^\top \Delta \geq \varepsilon - m_0\} = \frac{\varepsilon - m_0}{\|g\|_2^2}\, g, \qquad \|\Delta^\star\|_2 = \frac{\varepsilon - m_0}{\|g\|_2},$$

and for any training trajectory with net update $\Delta\theta = \theta_T - \theta_0$, we have parameter–redundancy ratio

$$\mathrm{PR}_x \equiv \frac{\|\Delta\theta\|_2}{\|\Delta^\star\|_2} = \|\Delta\theta\|_2 \cdot \frac{\|g\|_2}{\varepsilon - m_0}. \tag{1}$$

Proof. Let $\beta \coloneqq \varepsilon - m_0$. Under (A1) local linearity (a first-order Taylor approximation, which is standard for analyzing the local geometry of optimization landscapes (Bottou et al., 2018).) and (A2) competitor stability, achieving the (linearized) target margin ε is equivalent to requiring

$$g^\top \Delta \;\geq\; \beta,$$

where $g = \nabla_\theta m(\theta_0)$. Hence the minimal-displacement problem is

$$\Delta^\star = \min_{\Delta \in \mathbb{R}^d} \|\Delta\|_2 \quad \text{s.t.} \quad g^\top \Delta \geq \beta.$$

If $\beta \leq 0$, then $\Delta^\star = 0$ is feasible and optimal (the target margin is already met), and the result is trivial. In the remainder, assume $\beta > 0$ and $g \neq 0$.

For any feasible Δ, by Cauchy–Schwarz,

$$\beta \;\leq\; g^\top \Delta \;\leq\; \|g\|_2 \, \|\Delta\|_2 \quad \Longrightarrow \quad \|\Delta\|_2 \;\geq\; \frac{\beta}{\|g\|_2}.$$

Equality holds if and only if Δ is colinear and aligned with g, i.e., $\Delta = \alpha g$ with $\alpha \geq 0$. Choosing

$$\alpha \;=\; \frac{\beta}{\|g\|_2^2} \quad \Rightarrow \quad \Delta^\star \;=\; \frac{\beta}{\|g\|_2^2}\, g, \qquad \|\Delta^\star\|_2 \;=\; \frac{\beta}{\|g\|_2}.$$

Thus $\Delta^\star$ achieves the lower bound and is the unique minimizer (any deviation from the g direction increases the norm for fixed inner product $g^\top \Delta = \beta$).

Now consider any training trajectory with net update $\Delta\theta = \theta_T - \theta_0$ that attains the target margin in the linearized model, so $g^\top \Delta\theta \geq \beta$. By optimality of $\Delta^\star$, $\|\Delta\theta\|_2 \geq \|\Delta^\star\|_2$. Therefore, for $\beta > 0$,

$$\mathrm{PR}_x \;\equiv\; \frac{\|\Delta\theta\|_2}{\|\Delta^\star\|_2} \;=\; \|\Delta\theta\|_2 \cdot \frac{\|g\|_2}{\varepsilon - m_0}.$$

(When $\beta \leq 0$, we have $\Delta^\star = 0$ and the margin is already satisfied.) □

B.2. Redundancy Analysis of SFT and RL

Based on the parameter–redundancy ratio (PR) defined in the previous section, we now analyze the properties of SFT and RL based on their gradient. We first borrow the policy gradient form of SFT from (Wu et al., 2025), and then compare their parameter–redundancy ratios.

SFT Gradient. Let $\mathcal{D} = \{(x, y^*)\}$ denote a corpus of annotated demonstrations, where y^* is the reference response for query x. SFT minimizes the cross-entropy loss:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = {}_{(x,y^*)\sim\mathcal{D}}\left[-\log \pi_\theta(y^* \mid x)\right],$$

and its gradient is:

$$\nabla_\theta \mathcal{L}_{\mathrm{SFT}}(\theta) = {}_{(x,y^*)\sim\mathcal{D}}\left[-\nabla_\theta \log \pi_\theta(y^* \mid x)\right]. \tag{2}$$

RL Gradient. Let y denote a rollout response sampled from the policy model $\pi_\theta(\cdot \mid x)$ for query x. Given a reward function $r(x, y) \in \mathbb{R}$, the policy objective is

$$J(\theta) = {}_{x\sim\mathcal{D}_x,\; y\sim\pi_\theta(\cdot|x)}\left[r(x, y)\right],$$

and its policy gradient is

$$\nabla_\theta J(\theta) = {}_{x\sim\mathcal{D}_x,\; y\sim\pi_\theta(\cdot|x)}\left[\nabla_\theta \log \pi_\theta(y \mid x)\; r(x, y)\right]. \tag{3}$$

Rewriting SFT Gradient as Policy Gradient. Here we directly quote the conclusion from (Wu et al., 2025) to rewrite SFT gradient to policy gradient. The SFT gradient in Equation 2 taken under the fixed demonstration distribution is convert it to an on-policy expectation by inserting an importance weight that compares the expert (Dirac Delta) distribution with the model distribution:

$${}_{(x,y^*)\sim\mathcal{D}}[-\nabla_\theta \log \pi_\theta(y^* \mid x)] = {}_{x\sim\mathcal{D}_x}\; \underbrace{{}_{y\sim\pi_\theta(\cdot|x)}\left[\frac{\mathbb{I}[y = y^*]}{\pi_\theta(y \mid x)}\left(-\nabla_\theta \log \pi_\theta(y \mid x)\right)\right]}_{\text{resample + reweight}}. \tag{4}$$

Define the auxiliary variables

$$w(y \mid x) = \frac{1}{\pi_\theta(y \mid x)}, \qquad r(x, y) = \mathbb{I}[y = y^*].$$

Reorganize Equation 4 and rewrite it using the above auxiliary variables; we obtain

$$\nabla_\theta \mathcal{L}_{\mathrm{SFT}}(\theta) = -\,{}_{x\sim\mathcal{D}_x,\; y\sim\pi_\theta(\cdot|x)}\Big[w(y \mid x)\; \nabla_\theta \log \pi_\theta(y \mid x)\; r(x, y)\Big]. \tag{5}$$

This form of the SFT gradient now closely aligns with the policy gradient in Equation 3: Conventional SFT can be viewed as an on-policy gradient where the reward is an indicator of matching the expert response, modulated by an importance weight $1/\pi_\theta$.

Then, we will utilize obtained conclusions to analyze PR of SFT and RL.

Theorem 1. *Define the reward function as r for the policy model π_θ. For a input context x, policy model output token y_t according to x and target token y_t^*, if $\mathbb{I}[y_t = y_t^*] \geq r\pi_\theta(y_t \mid x)$, SFT (policy model expression) and RL predict y^* that both attain the same target margin ε satisfy*

$$\mathrm{PR}_x^{\mathrm{SFT}} \geq \mathrm{PR}_x^{\mathrm{RL}}.$$

Proof. According to Equation 1 in Lemma 1, we have the following DR for SFT and RL:

$$\mathrm{PR}_x^{SFT} \equiv \frac{\|\Delta\theta^{SFT}\|_2}{\|\Delta^\star\|_2} = \|\Delta\theta^{SFT}\|_2 \cdot \frac{\|g\|_2}{\varepsilon - m_0} = \|\eta\nabla_\theta \mathcal{L}_{\mathrm{SFT}}(\theta)\|_2,$$

$$\mathrm{PR}_x^{RL} \equiv \frac{\|\Delta\theta^{RL}\|_2}{\|\Delta^\star\|_2} = \|\Delta\theta^{RL}\|_2 \cdot \frac{\|g\|_2}{\varepsilon - m_0} = \|\eta\nabla_\theta J(\theta)\|_2.$$

Then we utilize the sentence level policy gradient expressions in Equation 4 and Equation 3. Here we make slight change from the sentence level expression to token level expression here to simplify the illustration and have:

$$\frac{\mathrm{PR}_x^{SFT}}{\mathrm{PR}_x^{RL}} = \frac{\|\nabla_\theta \mathcal{L}_{\mathrm{SFT}}\|_2}{\|\nabla_\theta J(\theta)\|_2} = \frac{\mathbb{I}[y_t = y_t^*]}{r\pi_\theta(y_t \mid x)} \geq 1 \tag{6}$$

So we have

$$\mathrm{PR}_x^{\mathrm{SFT}} \geq \mathrm{PR}_x^{\mathrm{RL}}.$$

□

Theorem 1 provides the condition when PR_x^{SFT} will larger then PR_x^{RL}. Since when SFT updates, the gradient in Equation 4 can't be zero, therefore $\mathbb{I}[y_t = y_t^*] = 1$. And for hard reasoning case, the policy probability $\pi_\theta(y_t \mid x) \ll 1$ while reward function r is usually bounded (PPO/GRPO), therefore $\mathbb{I}[y_t = y_t^*] \geq r\pi_\theta(y_t \mid x)$, and SFT has more parameter updating redundancy compared to RL.

Our theoretical analysis, formalized in Lemma 1 and Theorem 1, provides a causal mechanism for the empirical phenomena observed in Section 2.2. The core theoretical finding is that the Parameter-Redundancy (PR) on logits for predicting the target token of the SFT is intrinsically higher than that of the RL after parameter updates (i.e., $\mathrm{PR}^{\mathrm{SFT}} \geq \mathrm{PR}^{\mathrm{RL}}$). This conclusion directly explains why SFT exhibits *sparsification robustness*: Theorem 1 predicts SFT has a high PR, implying its parameter updates are redundant. This redundancy allows SFT to maintain performance even when 50% of updates are pruned (as shown in Figures 2 and Figure 3), as it has sufficient redundant updates to secure the correct token prediction. Conversely, RL's low PR suggests its updates are *parsimonious* or not robust, which explains its fragility and immediate performance degradation under similar pruning conditions. Since calculating the exact PR is computationally intractable for large language models, our sparsification experiments serve as the empirical way to validate this theory. Theorem 1 predicts this imbalance, and the experiments (Figures 2 and Figure 3) confirm it by demonstrating SFT's better tolerance to pruning.

C. Implementation Details

Method Implementation We implement RL and SFT following widely used settings. For RL, we remove the KL penalty, length normalization, and standard-error normalization, as in Dr.GRPO (Liu et al., 2025c) and LUFFY (Yan et al., 2025). GRPO hyperparameters are: rollout batch size 128, number of rollouts per query 8, update batch size 64, learning rate 1×10^{-6}, rollout temperature 1.0, and entropy coefficient 0.001. For SFT, the training batch size is 128 and the learning rate is 1×10^{-5}. For MIFO, the rollout-accuracy threshold for collecting SFT batch $\mathcal{B}_{\mathrm{FT}}$ is $p = \frac{1}{8}$, the high-entropy token ratio is $\rho = 0.2$, the TopK freezing fraction is $k = 0.5$, and the decay coefficient is $\alpha = 0.5$. The SFT batch $\mathcal{B}_{\mathrm{FT}}$ size is set to be 64, with a study on SFT batch $\mathcal{B}_{\mathrm{FT}}$ size showing in Table 11. Pseudocode of MIFO is provided in Algorithm 1.

Training Following prior work (Yan et al., 2025; Ma et al., 2025; Fu et al., 2025; Cui et al., 2025; Liu et al., 2025c; Hu et al., 2025), we train on Qwen2.5-Math (1.5B and 7B) (Yang et al., 2024). As in (Fu et al., 2025; Yan et al., 2025; Ma et al., 2025), we increase the RoPE base from 10,000 to 40,000 and extend the context window to 16,384 tokens to encourage

Algorithm 1 MIFO: MItigating FOrgetting between SFT and RL

Require: Policy π_θ, old policy $\pi_{\theta_{\text{old}}}$, reward R, entropy ratio ρ, decay α, SFT batch $\mathcal{B}_{\text{FT}}$ size S, SFT data D.
Initialize SFT batch $\mathcal{B}_{\text{FT}} \leftarrow \emptyset$, $C_0^j = 0$ for all j, Define SFT $\rightarrow$ RL iteration i as $\text{Interval}_i^{RL \rightarrow SFT}$
for each $\text{Interval}_i^{RL \rightarrow SFT}$ interval iteration i **do**
  Record $\theta^{\text{j}}_{RL,start,i}$
  for each question q **do**
    $\pi_{\theta_{\text{old}}}(q)$ rollout, compute $R(o_i)$ and advantages A_i. Update π_θ via GRPO
    if $\text{acc}(q) < a$ **and** $\text{extract}(D(q)) = a$ **then**
      Add $(q, D(q))$ to SFT batch $\mathcal{B}_{\text{FT}}$
    end if
  end for
  Record $\boldsymbol{\theta}^{\text{j}}_{RL,end,i}$
  $\Delta\boldsymbol{\theta}^j_{\text{RL},i} \leftarrow \left\| \theta^j_{\text{RL,end},i} - \theta^j_{\text{RL,start},i} \right\|_2$, $\tilde{\boldsymbol{C}}_i \;=\; \alpha\, \boldsymbol{C}_{i-1} \;+\; (1-\alpha)\, \Delta\boldsymbol{\theta}_{\text{RL},i}$
  Obtain the mask $\mathbf{M}_i^j$: $\mathbf{M}_i^j = \mathbf{1}[\, j \in \mathcal{I}_i\,]$; $\mathbf{M}_i^j = \mathbf{1}[\, j \in \mathcal{I}_i\,], \mathbf{C}_i \;=\; \mathbf{M}_i \odot \tilde{\mathbf{C}}_i$,
  Freeze Selected parameters $\nabla_{\boldsymbol{\theta}_i} \mathcal{L}_{\text{SFT}} \;\leftarrow\; (\mathbf{1} - \mathbf{M}_i) \odot \nabla_{\boldsymbol{\theta}_i} \mathcal{L}_{\text{SFT}}$..
  SFT only calculate loss for tokens with $H_t \geq \tau_\rho^q$ in SFT batch $\mathcal{B}_{\text{FT}}$, and unfreeze.
end for

Table 10. GPU hours for different methods.

Method	SFT	RL	SFT-RL	LUFFY	LUFFY+	ReLIFT	MIFO
GPU Hour	8×8	40×8	67×8	77×8	130×8	52×8	74×8

exploration. Models are trained for three epochs on 4×8 A100 GPUs using the VERL framework (Sheng et al., 2025). The reward is binary:

$$R = \begin{cases} 1, & \text{if the answer is correct,} \\ 0, & \text{otherwise.} \end{cases}$$

The training GPU hours comparison information is provided in Table 10. Note that the hours information is based on 8 GPUs to fairly compare with baselines.

Evaluation Except for Oat-Zero and OpenReasoner-7B, we independently evaluate publicly released baseline checkpoints to reproduce and verify prior results. The remaining baseline scores are taken from (Ma et al., 2025), which adopts the same evaluation protocol as MIFO. All evaluations are conducted with vLLM (Sheng et al., 2025), using a temperature of 0.6 and a maximum sequence length of 8,192 tokens. Additional dataset statistics are provided in Table 12.

Chat Template For our main experiments, we follow (Ma et al., 2025; Yan et al., 2025; Fu et al., 2025) and adopt the LUFFY chat template across all training paradigms. This unified system prompt encourages systematic reasoning—analyze, summarize, explore, reassess, and refine—as illustrated in Figure 8. We also evaluate with other popular templates shown in Figure 8 to assess robustness under template shifts.

Table 11. The average pass@1 accuracy for training Qwen2.5-1.5B-Math with 1 epoch.

$\mathcal{B}_{\text{FT}}$ Size	8	16	64	128	256
Avg	32.4	34.4	**36.3**	36.1	35.8

Table 12. The statistics of evaluation datasets

Dataset	#Test	Task Type	Domain	#k
AIME24	30	Math competition	Mathematics	32
AIME25	30	Math competition	Mathematics	32
AMC	83	Math competition	Mathematics	32
MATH-500	500	Mathematical reasoning	Mathematics	1
Olympiad	674	Math competition	Mathematics	1
MMLU-Pro	12,102	Multi-task understanding	Multidisciplinary	1

Table 13. Performance comparison on Qwen-Math-2.5 with different templates. **Bold** and underline indicate the best and second-best results, respectively.

Templates	Model	AIME-24	AIME-25	AMC	MATH-500	Olympiad	MMLU-Pro	Average
No	ReLIFT	23.3	14.1	54.2	75.0	41.8	16.4	37.6
	LUFFY	25.4	14.1	55.9	75.4	41.0	16.3	38.0
	SRFT	**27.7**	14.3	**59.2**	77.8	42.9	21.7	40.6
	MIFO	26.1	**17.7**	58.7	**79.6**	**43.4**	**22.9**	**41.4**
Qwen	ReLIFT	25.3	14.7	59.5	76.6	40.7	45.1	43.7
	LUFFY	25.3	13.0	56.6	76.4	40.2	44.8	42.7
	SRFT	27.3	15.2	58.9	77.0	42.1	45.3	44.3
	MIFO	**28.2**	16.3	**61.9**	**77.4**	**42.4**	**45.7**	**45.3**
Prime	ReLIFT	27.2	**21.9**	63.4	86.0	**52.1**	49.2	50.0
	LUFFY	27.4	17.0	61.2	84.4	46.1	47.8	47.3
	SRFT	26.7	20.1	68.1	86.2	49.3	49.9	50.1
	MIFO	**28.5**	**21.9**	**68.3**	**87.4**	50.2	**50.0**	**51.1**

D. Additional Experiments

D.1. Hyperparameter Analysis

Our method uses four hyperparameters: the rollout-accuracy threshold p for collecting SFT questions, the high-entropy token ratio ρ, the TopK freezing fraction k, and the history coefficient α. As shown in Figures 10, these hyperparameters are insensitive and easy to tune. In Figure 10 (left), the model performs best at $p = \frac{2}{8}$. Since performance is comparable across nearby settings, we choose $p = \frac{1}{8}$ to reduce SFT data usage. Since the limit of computational resources, this analysis is conducted for only 1 epoch of training.

D.2. Template Ablation Study

In real-world deployment, users may not provide carefully designed chat prompts for LLM reasoning. It is therefore important to assess robustness in the wild. We evaluate MIFO under multiple reasoning templates to test whether it maintains strong performance without curated prompts. In Section 4.4, we reported average reasoning accuracy and response length under the *No template* and *Qwen* (Yang et al., 2024) settings. Here we add results with another popular template, *Prime* (Cui et al., 2025), which is commonly used for RL-trained reasoning models, and we provide per-dataset scores in Table 13. Table 13 and Figure 9 further indicates that our method is template-robust, outperforming competitive baselines across all three templates.

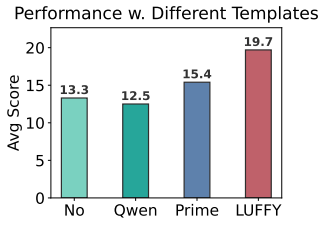

Figure 9. Average score for Llama 3.2-8B w. different templates.

D.3. Training Dynamics

Batch Solve Figure 11 presents training dynamics for ablated models on *batch solved rate* (left) and *batch all solved rate* (right). For both **batch solved rate** and **batch all solved rate**, Interleave + ES has comparable performance with PF, and they all outperform Interleave. **MIFO** outperforms other ablated models at most of

Chat Template (Qwen)

Please reason step by step, and put your final answer within \\box{}.
Question: {question}
Answer: {answer}

Chat Template (Prime)

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>
Question: {question}
Answer: {answer}

Chat Template (LUFFY)

Your task is to follow a systematic, thorough reasoning process before providing the final solution. This involves analyzing, summarizing, exploring, reassessing, and refining your thought process through multiple iterations. Structure your response into two sections: Thought and Solution. In the Thought section, present your reasoning using the format:"\n {thoughts} \n". Each thought should include detailed analysis, brainstorming, verification, and refinement of ideas. After "\n" in the Solution section, provide the final, logical, and accurate answer, clearly derived from the exploration in the Thought section. If applicable, include the Answer in \boxed{} for closed-form results like multiple choices or mathematical solutions.
Question: {question}
Answer: {answer}

Figure 8. Chat template details.

the steps, indicating the effectiveness of combining each module: increasing the rollout accuracy during the RL training.

Reward Value Figure 12 (left) reports the average rollout reward during training. All variants improve rapidly at the start, then separate modestly mid-training. **Interleave+ES** and **Interleave+PF** have comparable reward value, outperforming **Interleave** and falling behind MIFO. The combined (**MIFO**) stays at or near the top throughout and converges to the highest point. These trends indicate that each module contributes positively to MIFO.

Average Validation Score Similarly, Figure 12 (right) supplements the ablation in Section 4.3. The ablated models **Interleave+ES** and **Interleave+PF** generally surpass the **Interleave** baseline, confirming the effectiveness of ES and PF. The combined **MIFO** outperforms the ablations for most of the training, supporting the choice to combine Interleave, ES, and PF for stronger reasoning performance.

D.4. Comparison with LoRA Baseline

Beyond our parameter freezing strategy, one might consider straightforward baselines such as employing LoRA or reducing the learning rate (LR) for SFT, while utilizing a larger LR or full-parameter updates for RL. Regarding the SFT phase, we have already reduced the learning rate from the standard 10^{-5} to 10^{-6}. This aligns with established practices in prior

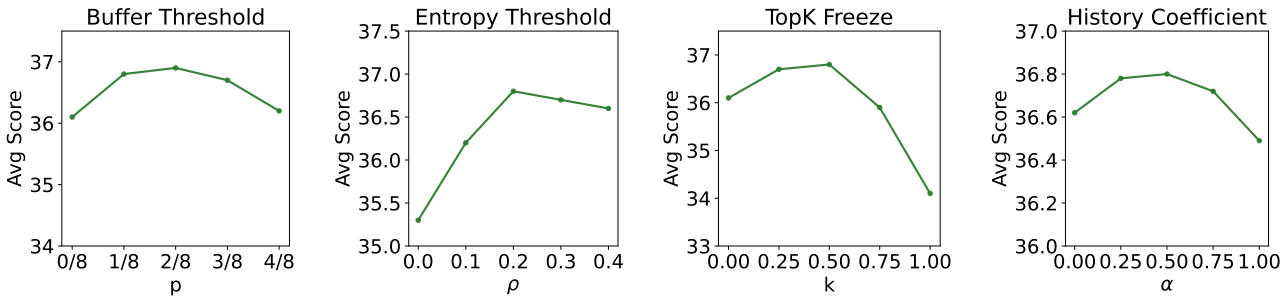

Figure 10. Hyperparameter analysis.

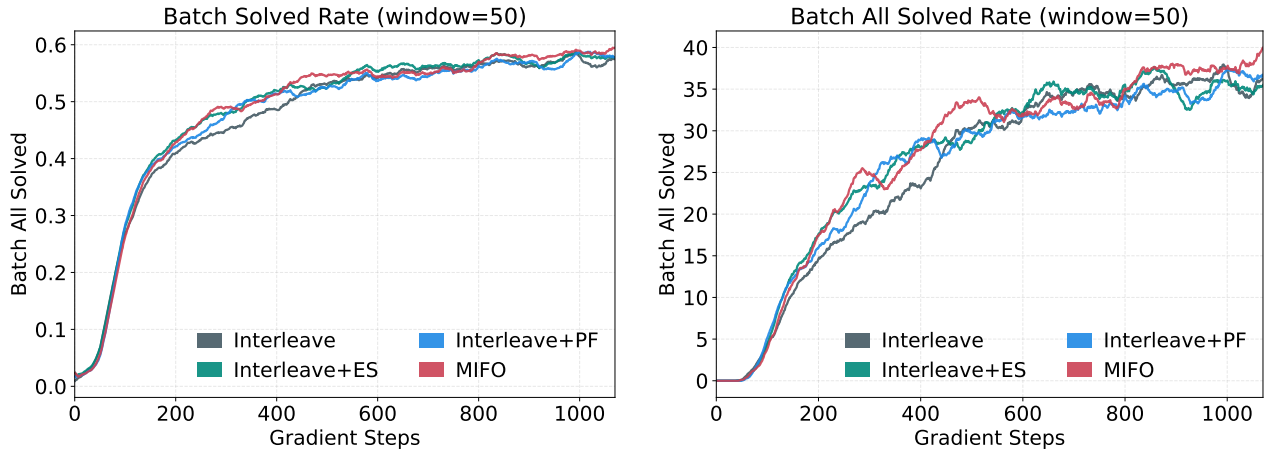

Figure 11. Training dynamics for ablated models on batch solving with running average.

SFT+RL pipelines (LUFFY, ReLIFT, and SRFT), where larger SFT learning rates were observed to degrade reasoning performance, while rates below 10^{-6} proved too weak to be effective. The necessity of shrinking the SFT learning rate to preserve RL-trained capabilities corroborates our hypothesis that naive SFT induces catastrophic forgetting; specifically, larger SFT updates tend to aggressively overwrite RL-acquired information. For the RL phase, we utilize a learning rate of 10^{-6}, consistent with standard PPO and GRPO implementations. Simply increasing the RL learning rate is not a viable alternative, as it typically leads to mode collapse, reward hacking, and training instability driven by excessive KL divergence.

As for using LoRA for SFT while keeping a larger learning rate or full-parameter RL, we do not find SFT-LoRA to be a prioritized solution because its performance gains are limited by the small number of tunable parameters. We run experiments where we replaced full-parameter SFT with LoRA SFT (r=64,α=128) for multiple SFT+RL methods. As shown in Table 14, the LoRA-adapted variants consistently underperform their fully fine-tuned counterparts, even though LoRA naturally reduces forgetting by separating parameter subsets. Our interpretation is that, compared with our freezing strategy, LoRA exposes far fewer effective degrees of freedom to SFT, so the model cannot learn the SFT data as comprehensively. Thus, while smaller SFT LR and LoRA are reasonable baselines, their limitations further motivate our approach of explicitly identifying and protecting RL-critical parameters via MIFO, rather than relying solely on generic tricks like lowering LR or swapping in LoRA.

Table 14. The comparison of methods' pass@1 test accuracies w. and w.o. lora, 1 epoch training for 15k data with Qwen2.5-1.5B-Math-Instruct.

Methods	**SFT-RL**	**ReLIFT**	**MIFO**
w. lora	26.9	28.1	28.6
w.o. lora	27.9	29.5	30.7
Δ	+1.0	+1.4	+2.1

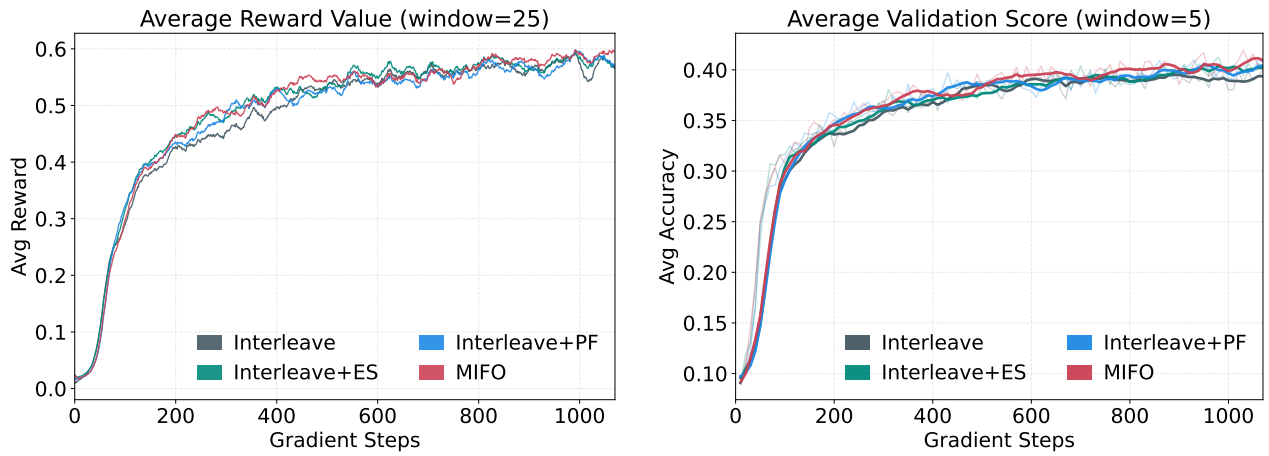

Figure 12. Training dynamics for ablated models on average reward value (left) and average validation scores (right) with running average.

E. Discussion

E.1. Experiment Design for Observations

Here we further justify the experimental design in Section 2.2. Using both *online* and *post-hoc* sparsification probes complementary forms of redundancy and yields a stronger diagnosis of SFT and RL learning. *Online sparsification* evaluates training-time resilience. If performance remains stable when randomly dropping half of the gradient coordinates, the learning signal and optimizer dynamics are overparameterized (gradient-level redundancy) rather than dependent on precise dense updates. *Post-hoc sparsification* evaluates endpoint redundancy. If the final predictor maintains accuracy after pruning the net parameter change, a substantial portion of the learned displacement Δ is superfluous (parameter-change redundancy). This separates trajectory effects from the equivalence class of solutions at convergence. Together, these two regimes provide orthogonal evidence for redundancy within each training paradigm.

E.2. Why Freezing instead of SFT Variants

Prior work on fine-tuning LMs offers a key insight: *fully fine-tuning is not always optimal; selectively updating a subset of parameters can match or even surpass full-model tuning*. Evidence for layer heterogeneity suggests that focusing updates on important components is preferable to uniform tuning (Zhang et al., 2022; Lee et al., 2019). ULMFiT (Howard & Ruder, 2018) progressively unfreezes layers to retain prior knowledge and reduce catastrophic forgetting. Parameter-efficient methods (Houlsby et al., 2019; Li & Liang, 2021; Hu et al., 2022) train only a small fraction of parameters yet achieve comparable performance. BitFit (Ben Zaken et al., 2022) reaches strong results by updating only bias terms. AdapterDrop (Rücklé et al., 2021) shows that adapting only important layers can be more effective and efficient.

These lessons highlight inherent drawbacks of SFT, including overfitting and unequal utility across parameters. Our dropout-like selective freezing for SFT (Srivastava et al., 2014; Baldi & Sadowski, 2013) both protects RL-acquired knowledge and reduces overfitting risk. While reducing data or lowering the learning rate can also lessen SFT-induced forgetting, such approaches risk underutilizing SFT knowledge.

E.3. The Relationship and Difference with Previous Works

We want to clarify again that MIFO's central idea of mitigating forgetting is completely different with previous methods. The only similar design with DeepSeekMath (Shao et al., 2024) and ReLIFT (Ma et al., 2025) is interleaving SFT+RL. Actually, our main contribution is the first to identify the forgetting between SFT and RL training, and providing the corresponding explanation and two main contribution to mitigate the forgetting. 1) data processing for SFT+RL interleaved training. Our approach sets a threshold $\text{acc}(q) \leq p$ and combines high-entropy token selection with parameter freezing. These modules mitigate SFT–RL forgetting, enlarge the usable data pool, and exploit more of the available reasoning signal. This extensions for interleaved SFT+RL is only one part of our first contribution. 2) Parameter tracking and freezing. According to our motivation, we design a parameter freezing strategy for RL updates to explicitly control the forgetting, which is another

novel contribution.

E.4. Response Length Discrepancy of MIFO and MIFO†

Table 3 show that**MIFO** produces shorter response length than **MIFO**† for the 7B model, while **MIFO**† yields shorter response length than **MIFO** for the 1.5B model. To interpret these results, we first note the baseline behaviors: SFT tends to be extremely verbose (>10k tokens) while RL encourages conciseness (<1k tokens). Therefore, the output length serves as a proxy for our claim on catastrophic forgetting:a longer output indicates that the SFT phase (verbose) has partially overwritten the RL phase (concise). The role of the history parameter α is to stabilize the identification of "RL-critical" parameters over time. For the 7B model, which has a stable training process because of its larger parameter capacity, the history mechanism ($\alpha > 0$) effectively accumulates long-term signals to identify truly critical RL parameters. This allows MIFO to freeze them robustly, preventing SFT overwriting and keeping the output concise (1067 tokens, closer to the RL baseline). MIFO† ($\alpha = 0$) lacks this long-term stability, allowing more SFT verbosity to leak through (1371 tokens). Conversely, the 1.5B model is more volatile and unstable during training because of the limited parameter capacity. In this unstable regime, history can become noise: the accumulated importance map may lag behind the model's rapid shifts. Consequently, MIFO's historical mask is less precise for the small model, allowing SFT verbosity to influence the generation (2518 tokens). MIFO†, by relying solely on the most recent, immediate update, captures the volatile 1.5B model's current state more accurately, blocking SFT influence and retaining the short RL length (985 tokens). Note that despite the length difference, both methods achieve similar high accuracy, validating the general efficacy of the freezing approach.