Evaluating the Limitations of Reinforcement Learning with Verifiable Rewards in Small Large Language Models

Brennen Hill [1] **Albert Wu** [1]

1. Introduction

We investigate whether reinforcement learning with verifiable reward (RLVR) can be used to effectively post-train small language models (SLMs), which we define as models with fewer than 7B parameters. The prevailing consensus is that RLVR fails to improve SLM performance and often leads to training collapse, due to the models' limited capacity to generate improved outputs and the high-entropy exploration inherent to RLVR. If this limitation could be overcome, RLVR would offer a compelling alternative to the current practice of over-training large language models with excessive compute to handle a broad range of tasks. Through empirical analysis on SLMs such as the Qwen3 models, we examine whether RLVR training can be stabilized to yield meaningful performance gains.

On top of being able to gain performance on one task through RLVR, for SLMs to function as versatile agents, they must also retain sufficient plasticity to acquire new reasoning capabilities without catastrophic forgetting of previously learned skills—a challenge that is particularly acute under RLVR, where aggressive policy updates and high-entropy exploration can destructively overwrite existing representations. To rigorously study this trade-off, we subject SLMs to sequential reasoning tasks, transitioning from mathematical problem solving (GSM8K) to logical deduction (bAbI), and quantify the resulting negative transfer. We evaluate a range of mitigation strategies, spanning parameter-centric methods such as Orthogonal LoRA and Task Arithmetic, as well as data-centric approaches including Experience Replay and data mixing. Our results indicate that data mixing remains the most effective strategy for preserving reasoning depth and stability in small architectures under RLVR.

All of our code can be accessed at our GitHub repository.

[1]Department of Computer Science, University of Wisconsin Madison, United States. Correspondence to: Brennen Hill <bahill4@wisc.edu>, Albert Wu <twu376@wisc.edu>.

Preprint. December 13, 2025.

2. Related Work

RLVR is a post-training method introduced by DeepSeek (DeepSeek-AI Team, 2025) that relies on deterministic verifiers for reward signals, such as proof checkers or unit tests, instead of a pre-trained neural reward model. It has been widely believed that RLVR does not work well with SLMs, which are more vulnerable to issues such as gradient collapse, reward hacking, and catastrophic forgetting due to their limited capacity for complex reasoning (Dang and Ngo, 2025; Sarangi and Salam, 2025). Furthermore, recent investigations into the reasoning ability of small models suggest that model compression techniques inherently degrade reasoning capabilities, making the stabilization of RLVR on SLMs particularly precarious (Yang et al., 2025).

However, foundational work on compute-optimal scaling laws (Raschka et al., 2022) suggests that under a fixed compute budget, performance can favor smaller models trained on significantly more data. Recent studies confirm this potential, showing that reasoning tasks can be optimized on smaller architectures given sufficient data density (Roberts et al., 2025). Despite this insight, developing a stable RLVR framework that can surface these latent capabilities in SLMs remains a challenge, which this project seeks to address.

Substantial prior work has focused on improving the stability of RLVR training. Early approaches include warming up models with supervised fine-tuning (SFT) (Hong et al., 2025) and adopting conservative hyperparameter choices such as anchoring (OpenLMLab et al., 2023). More advanced methods introduce learnable curricula to ease optimization, such as length-aware updates that prioritize shorter rollouts (Xiong et al., 2025), bootstrapping short-horizon tasks to enable longer-horizon reasoning (Guo et al., 2025), and step-by-step verifiers that provide denser reward signals (Guan et al., 2025). Beyond curricula, recent algorithmic innovations have focused on update dynamics; for instance, Wang et al. Wang et al. (2025) propose a Slow-Fast policy optimization that stabilizes training by repositioning the policy before applying updates, while Zheng et al. Zheng et al. (2025) demonstrate that balanced actor initialization, merging SFT models with pre-trained bases, can significantly stabilize distillation-based reasoning.

Our work builds on these approaches and introduces further innovations by developing a curriculum that is learnable even for SLMs. Specifically, we leverage the StepCoder dataset (Dou et al., 2024), which is sorted according to hierarchical problem structure, allowing the model to learn simpler sub-problems before progressing to more complex composite tasks.

In addressing reward hacking and diversity collapse, prior work has shown that penalizing disagreement across an ensemble of reward models can be effective (Zhai et al., 2024). Extending this line of inquiry, Li et al. Li et al. (2025) argue that the specific choice of divergence measure (e.g., mass-covering f-divergences) is a neglected key to mitigating diversity collapse in verifiable reward settings. Similarly, Wolf et al. Wolf et al. (2025) highlight the risk of over-optimization in static setups, advocating for iterated RLHF frameworks. Inspired by these approaches, we mitigate hacking by introducing two distinct groups of reward functions, functional and structural, and applying penalties based on disagreement between them.

Finally, catastrophic forgetting remains a major concern when adapting large language models. DeepSeekMath (Shao et al., 2024) emphasizes that robust SFT is a strict prerequisite for RL to prevent immediate performance degradation. Recent studies emphasize minimizing unnecessary model drift during training. Shenfeld et al. Shenfeld et al. (2025) show that online reinforcement learning leads to less forgetting and argue that methods should be evaluated on how conservatively they move in KL divergence relative to the base model. Song et al. (2025) propose regularization strategies that constrain updates at both the layer and parameter levels. To achieve effective KL regularization, we experiment with several techniques, including experience replay (Lin, 1993), dataset mixing, and training multiple adapters.

3. Implementation of Stability Optimization in Curriculum Learning Experiments

3.1. Task and Training Pipeline

Given a dataset $D = \{(x_i, y_i^*)\}_{i=1}^N$ of prompts x and reference code y^*, we train a code policy $\pi_\theta(y \mid x)$ in two stages.

First, we run supervised fine-tuning (SFT) on MBPP using reference solution. Second, we perform reinforcement learning (RL) with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) on APPS/APPS-CCCS using multiple reward signals and conservative updates, including a small learning rate, gradient clipping, a trust-region style update, and KL regularization.

3.2. GRPO Objective

For each prompt x, we sample K completions $y^{(1:K)} \sim \pi_\theta(\cdot \mid x)$. GRPO updates π_θ to increase reward while limiting drift from a reference policy π_{ref}.

A high-level objective is: $\max_\theta \ \mathbb{E}\big[R(x,y)\big] - \beta\,\mathrm{KL}(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x))$, with a trust-region style clip parameter ϵ.

In our configuration, $\epsilon = 0.18$ and $\beta = 0.01$, with $\texttt{learning\_rate} = 0.98 \times 10^{-6}$ and $\texttt{max\_grad\_norm} = 0.8$.

3.3. Reward Design

We combine five reward functions with fixed weights: $R(x,y) = \sum_{j=1}^{5} w_j\, r_j(x,y)$, $w = [0.25,\ 0.30,\ 0.20,\ 0.15,\ 0.10]$.

Compile reward. $r_{\text{comp}}(y) = \begin{cases} +1, & \text{if the code compiles or parses successfully,} \\ -1, & \text{otherwise.} \end{cases}$

Heuristic unit test (static, no execution). We parse the abstract syntax tree (AST) and score five aspects: syntax correctness, input/output pattern matching, complexity, algorithmic patterns, and solution completeness. These aspects are weighted as $(0.35,\ 0.15,\ 0.20,\ 0.15,\ 0.15)$.

The weighted score is mapped into the interval $[-1, 1]$ using a piecewise linear transformation and then clipped.

Structural similarity (AST-based). We compute an AST-based similarity score $s \in [0,1]$: $s = 0.15\, s_{\text{func}} + 0.20\, s_{\text{loop}} + 0.15\, s_{\text{cond}} + 0.25\, J(\text{calls}) + 0.15\, s_{\text{io}} + 0.10\, s_{\text{cx}}$, where $J(\cdot)$ denotes the Jaccard overlap between sets of called functions.

The corresponding reward is: $r_{\text{struct}} = \text{clip}(1.5\, s - 0.5,\ -1,\ 1)$.

BLEU similarity. We compute sentence-level BLEU between the generated code and the reference solution using smoothing, and return zero for empty outputs.

Disagreement penalty. We define a functional score F and a similarity score S: $F = 0.4\, r_{\text{comp}} + 0.6\, r_{\text{heur}}$, $S = 0.4\, r_{\text{bleu}} + 0.6\, r_{\text{struct}}$, $d = |F - S|$.

If d exceeds a fixed threshold (default 0.5), we apply a capped linear penalty to discourage disagreement between functional and similarity-based signals.

3.4. Strengths and Limitations

Strengths. Using GRPO without reliance on a reward model, which saves significant amount of memory.

Hierarchy-aware curriculum and conservative GRPO hyperparameters that significantly improve training stability. Disagreement penalty between the structural and the functional reward functions, making the training highly robust.

Limitations. Due to training time, no unit tests are executed, which introduces bias into the model's behavior and creates uncertainty over the actual quality of the model's code. Due to time and compute limit, the model could only be trained on a small dataset (about 5000 problems), which is tiny compared to the sizes of the typical datasets that are used in RLVR (in the magnitude of millions). This reduces the generalizability of our study to other settings.

4. Implementation of Catastrophic Forgetting Experiments

To investigate the phenomenon of catastrophic forgetting (McCloskey and Cohen, 1989) in Small Language Models (SLMs) trained via Reinforcement Learning (RL) and to evaluate potential mitigation strategies, we designed a three-part experimental framework. Our approach utilizes Group Relative Policy Optimization (GRPO) with verifiable reward functions to enforce reasoning capabilities in sequential tasks.

4.1. Tasks and Dataset Preparation

We selected two distinct reasoning domains to simulate sequential learning: Mathematical Reasoning (Task A) utilizing the **GSM8K** dataset (Cobbe et al., 2021) for multi-step word problems, and Logical Reasoning (Task B) utilizing **bAbI Task 1** (Weston et al., 2015) for identifying supporting facts in narrative contexts. To enable verifiable rewards, all data was reformatted into a strict XML-based chat template. The system prompt instructed the model to output reasoning within `<reasoning>` tags and the final conclusion within `<answer>` tags, decoupling the reasoning process from the final answer for granular reward scoring.

4.2. Model Architecture and Fine-Tuning

We employed the **Qwen2.5-Instruct** (Bai et al., 2023) family due to their strong baseline reasoning capabilities, utilizing `Qwen/Qwen2.5-0.5B-Instruct` (0.5×10^9 parameters) for Experiments I & II, and `Qwen/Qwen2.5-1.5B-Instruct` (1.5×10^9 parameters) for Experiment III. To minimize computational overhead and isolate the effects of forgetting to adapter weights, we utilized Low-Rank Adaptation (LoRA) (Hu et al., 2021) with the following configuration: Rank (r) 16, Alpha (α) 32, Dropout 0.05, and full projection coverage targeting `q_proj`, `k_proj`, `v_proj`, `o_proj`, `up_proj`, `down_proj`, and `gate_proj`. Due to the long training time of these models, we selected hyperparameters by identifying common values used in the literature and by how much time we would have to run the programs.

4.3. Reinforcement Learning Framework (GRPO)

We trained the models using **Group Relative Policy Optimization (GRPO)**, which optimizes the policy by normalizing rewards across a group of generations for the same prompt. We implemented a composite reward function defined as $R_{\text{total}} = R_{\text{format}} + R_{\text{correctness}}$

1. Format Reward (R_{format}): A discrete reward enforcing adherence to the XML schema:

$$R_{\text{format}} = \begin{cases} 1.0 & \text{if both } \texttt{<reasoning>} \text{ and } \texttt{<answer>} \text{ present} \\ 0.5 & \text{if only one tag set is present} \\ 0.0 & \text{otherwise} \end{cases} \tag{1}$$

2. Correctness Reward ($R_{\text{correctness}}$): We utilized three criteria: **Strict Match (+2.0)** for exact string matching (normalized to lowercase); **Numerical Tolerance (+2.0)** for mathematical answers if $|\text{pred} - \text{truth}| < 10^{-6}$; and **Partial Credit (+1.0)** if the ground truth string is contained within the prediction but is not an exact match.

4.4. Experimental Design

The study was divided into three distinct phases.

4.4.1. Experiment I: Establishing Catastrophic Forgetting

The objective was to quantify the degradation of the first task after training on the second task. We evaluated four scenarios: (1) **Baseline:** Zero-shot performance of the base model. (2) **Sequential A $\rightarrow$ B:** The model is trained on Math (A), establishing a baseline, and then the *same* weights are further trained on Logic (B). (3) **Sequential B $\rightarrow$ A:** The reverse order to check for task asymmetry. (4) **Joint Training:** Multi-task training where datasets A and B are concatenated and shuffled ($D_{\text{joint}} = D_A \cup D_B$), serving as the upper bound for performance.

4.4.2. Ablation Experiment: Evaluation of Mitigation Strategies

Using the 0.5B model, we perform an ablation study by evaluating five distinct strategies to mitigate forgetting during the A $\rightarrow$ B sequence: (1) **Experience Replay:** We constructed a mixed dataset for the second phase containing 100% of Task B and a random 25% subsample of Task A ($D_{\text{train}} = D_B \cup 0.25 D_A$). (2) **Task Arithmetic (Model Merging) (Ilharco et al., 2023):** We trained independent adapters θ_A and θ_B starting from the base model, creating the final model via linear combination

$\theta_{\text{final}} = \theta_{\text{base}} + \lambda(\theta_A + \theta_B)$, where $\lambda = 1.0$. (3) **EWC-Lite (Anchoring) (Kirkpatrick et al., 2017):** We utilized the KL-divergence penalty inherent in GRPO as a regularizer. By increasing the KL coefficient β from 0.04 to 0.20 during the second task, we penalized deviations from the Task A reference policy. (4) **Orthogonal LoRA:** We implemented parameter isolation by assigning disjoint subsets of model layers to specific tasks. Task A was trained on $\{W_q, W_v, W_{\text{up}}, W_{\text{down}}\}$, while Task B was trained on $\{W_k, W_o, W_{\text{gate}}\}$. (5) **Ensemble (Oracle):** We maintained two separate adapters. At inference time, a heuristic routes the query to the appropriate adapter.

4.4.3. Sensitivity Experiment: Scaling Analysis

To determine if model capacity influences susceptibility to forgetting, we experimented with changing the hyperparameter of model size. To do so, we replicated the primary findings on the larger **Qwen2.5-1.5B** model. Due to compute constraints, this experiment only compared the Sequential Baseline against the most effective mitigation strategy, Experience Replay.

4.5. Implementation Details

All experiments were implemented using PyTorch (Paszke et al., 2019), Transformers (Wolf et al., 2020), and TRL (von Werra et al., 2020). The optimizer used was AdamW (Loshchilov and Hutter, 2019) with a learning rate of 2×10^{-5}. Key parameters included a per-device batch size of 4 (for 0.5B model) or 2 (for 1.5B model); gradient accumulation adjusted to maintain consistent effective batch sizes; and 4 generations per prompt for GRPO group normalization. To ensure reproducibility, random seeds were fixed at 42 for dataset shuffling and sampling. Evaluation was performed on a held-out subset of 50 samples per task using a greedy decoding strategy to minimize variance.

4.6. Evaluation of Good and Bad Aspects of Implementation for Catastrophic Forgetting Experiments

A primary strength is the enforcement of a strict XML schema that decouples the generation of rationale from the final conclusion. By requiring the model to output distinct `<reasoning>` and `<answer>` tags, the reward function could assign partial credit for sound logic even if the final answer was incorrect, providing a denser learning signal than binary correctness allows. Furthermore, the inclusion of a Joint Training baseline established a crucial theoretical upper bound for performance, while the bidirectional sequencing (A $\rightarrow$ B and B $\rightarrow$ A) effectively controlled for task asymmetry, ensuring that observed forgetting was driven by parameter plasticity rather than inherent task difficulty. The study also benefited from a comprehensive ablation strategy that tested a diverse spectrum of mitigation techniques—ranging from data-centric experience replay to parameter-centric orthogonal adaptation—allowing for specific conclusions regarding the efficacy of gradient flow versus data distribution in stabilizing small models.

However, several limitations constrain the generalizability of these findings. First, the reliance on Low-Rank Adaptation (LoRA) restricts the scope of the forgetting analysis to adapter weights, potentially failing to capture the deeper, systemic forgetting dynamics that occur during full-parameter fine-tuning. Second, the selected task domains of mathematics and logic, while distinct, share significant cognitive overlaps in reasoning structure; the study does not address catastrophic forgetting in more divergent semantic shifts, such as moving from code generation to creative writing, where interference patterns may be more severe. In addition, due to the long training time of these experiments, hyperparameter sweeps were not feasible.

5. Empirical Analysis of Our Training Stabilization and Reward Hacking Prevention Approaches

Intuitively, curriculum learning should help improve training stability, while the reward disagreement penalty should reduce the level of reward hacking. Meanwhile, a lower learning rate and lower temperature, together with gradient clipping, trust region clipping, and non-zero KL regularization, should discourage the model from quickly moving away from the reference model and hence further stabilize training.

We verified this hypothesis by obtaining and comparing two baselines. The first baseline ran GRPO on the Qwen base model naively using the default hyperparameters without any modifications, using the BLEU score against the reference solution as the reward signal. The second baseline ran GRPO with all of the proposed stabilization techniques, including a more conservative set of hyperparameters. Due to compute and time constraints (one run takes 15 hours with an optimized inference pipeline), we only ran the ablation and sensitivity experiments on the Catastrophic Forgetting side. However, it remains clear that the ensemble of approaches was able to achieve the intended effect of improving training performance. Specifically, the post-training improvement margin of the optimized baseline was as large as 53$\times$ that of the naïve version.

We then ran evaluations on the test split of the APPS dataset that our models had not been trained on, using the next-token negative log-likelihood (NLL) against the ground truth solution as the metric. Unfortunately, although our optimized GRPO pipeline (0.87) showed performance improvement compared to the naïve version (0.95), the actual

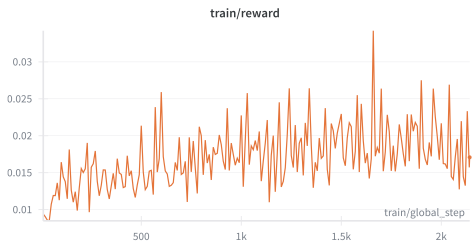

Figure 1. Reward over time for the naïve baseline, which increases from around 0.01 to 0.015 (150% relative improvement and +0.05 absolute improvement), with a high level of fluctuation and instability.

Figure 2. Reward over time for the optimized baseline, which increases from around -0.4 to 0.4 (200% relative improvement and +0.8 absolute improvement, much larger than that of the naïve baseline), with much lower variation and higher stability.

performance of the model degraded compared to the pre-trained base model (0.83). This may indicate that the model was still prone to reward hacking, explaining why it was able to achieve high rewards but underperformed on the actual benchmark. It is also possible that our reward functions did not adequately reflect the true objectives. Nevertheless, we believe that training stability did improve following our optimizations, and that similar procedures should be applied with better-designed reward functions and stronger penalties for reward hacking in order to truly realize the goal of improving the performance of small language models (SLMs) through RLVR.

6. Empirical Analysis of Catastrophic Forgetting

In this section, we present a comprehensive analysis of catastrophic forgetting in Small Language Models (SLMs) trained via Group Relative Policy Optimization (GRPO). We dissect the training dynamics, evaluate the efficacy of mitigation strategies, and investigate scaling behaviors.

6.1. Experiment Establishing Catastrophic Forgetting

The primary objective of Experiment I was to quantify the degradation of reasoning capabilities in a 0.5B parameter model under sequential fine-tuning, and to evaluate our

Hyperparameter	Baseline 1	Baseline 2
learning_rate	1.00×10^{-6}	9.80×10^{-7}
max_grad_norm	None	0.8
temperature	1.0	0.8
epsilon (trust region)	0.2	0.18
beta (KL regularization term)	0	0.01
batch_size	28	28
num_generations	4	4

Table 1. Hyperparameters used for the two baselines.

Model	NLL-1	NLL-2	NLL-3	Mean	Std
Qwen/Qwen3-1.7B-Base	0.824	0.837	0.836	0.832333	0.007234
GRPO	0.956	0.947	0.944	0.949	0.006245
GRPO + Curriculum + Disagreement Penalty + Conservative Params	0.865	0.860	0.882	0.869	0.011533

Table 2. Evaluation results on the APPS test split using next-token negative log-likelihood (NLL).

hypothesis that these techniques would reduce catastrophic forgetting. We compared sequential protocols (Task A $\rightarrow$ B and B $\rightarrow$ A) against a Joint Training baseline.

6.1.1. Training Stability and KL Divergence

Figure 3 illustrates the internal state of the model across the five experimental phases. A critical indicator of stability is the Kullback-Leibler (KL) divergence between the active policy and the reference model.

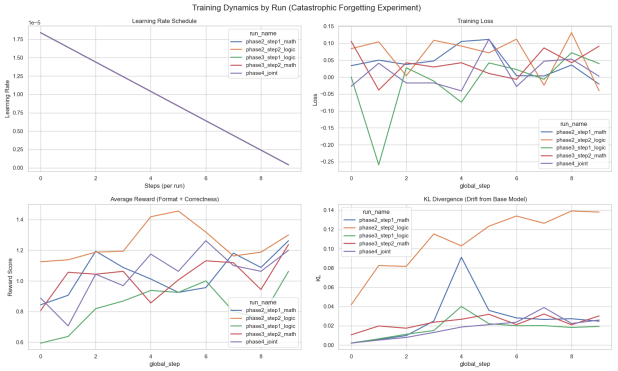

Figure 3. **Training Dynamics Analysis (0.5B Model).** The bottom-right panel highlights the significant KL divergence spike (≈ 0.14) during the Sequential Logic phase (orange), indicating aggressive parameter drift compared to the stable Joint Training run (purple).

As observed in Figure 3 (Bottom Right), the `phase2_step2_logic` run (orange) exhibits the highest KL divergence, peaking near 0.14. This suggests that to accommodate the Logic task after learning Math, the model was forced to aggressively update its weights, effectively drifting away from its pre-trained manifold. In contrast, Joint Training (purple) acted as a potent regularizer, maintaining a low and stable KL divergence

($\sim 0.02 - 0.04$), thereby preserving generalizability.

Furthermore, the Average Reward trajectory (Bottom Left) for the sequential logic phase peaks near 1.5, yet this does not correlate with high validation accuracy (as shown in Figure 4). This discrepancy is indicative of **Reward Hacking,** where the SLM prioritizes the syntactic structure required by the reward function (e.g., XML formatting).

6.1.2. Quantifying Catastrophic Forgetting

Figure 4 presents the definitive performance metrics. The results confirm the presence of severe catastrophic forgetting and negative transfer.

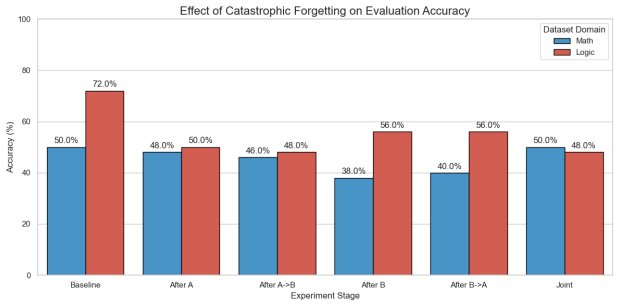

Figure 4. **Catastrophic Forgetting Impact.** Sequential training results in severe degradation of the non-target task. Notably, the sequence A $\rightarrow$ B results in lower Logic performance (48%) than the baseline (72%), indicating negative forward transfer.

- **Forgetting (Sequence B $\rightarrow$ A):** Upon retraining on Math (Task A), Logic performance drops from the 72.0% baseline to 56.0%. While Math accuracy recovers partially (38.0% to 40.0%), it fails to return to the 50.0% baseline, indicating permanent information loss.
- **Negative Forward Transfer (Sequence A $\rightarrow$ B):** This sequence revealed a critical failure mode.
  1. **Step 1 (Train Math):** Logic accuracy plummeted from 72.0% (Baseline) to 50.0%.
  2. **Step 2 (Train Logic):** Crucially, even after explicit training on Logic in the second phase, accuracy dropped *further* to 48.0%.

  This counter-intuitive result indicates that the weights learned during the Math phase created an interference pattern that rendered the model incapable of effectively relearning Logic, a form of plasticity collapse.

6.1.3. Plasticity-Stability and Reward Decomposition

To understand the mechanism behind these drops, we analyzed the Plasticity-Stability trajectory (Figure 5) and decomposed the reward signals (Figure 6).

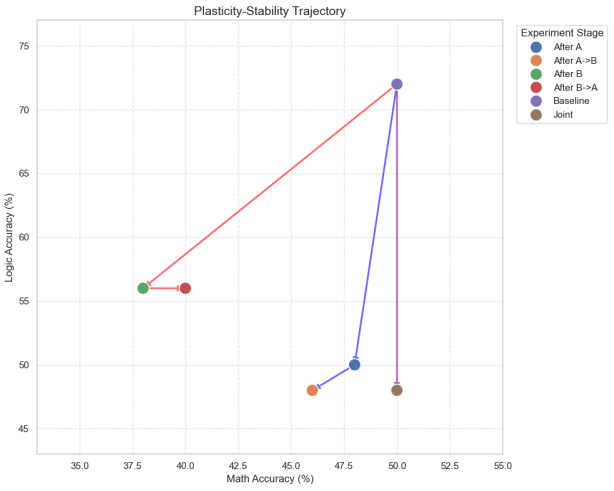

Figure 5. **Plasticity-Stability Trajectory.** Unlike standard Pareto frontiers, the 0.5B model exhibits a generalized collapse, moving downward and to the left of the baseline, indicating insufficient capacity to maintain multi-task representations.

The trajectory analysis reveals a generalized collapse. All sequential fine-tuning paths move away from the optimal top-right quadrant. This confirms that for 0.5B models, sequential instruction tuning is destructive to pre-trained representations.

Figure 6. **Reward Decomposition.** The divergence between Format Reward (Blue) and Correctness Reward (Red) illustrates the alignment tax, where the model optimizes syntax at the expense of reasoning.

Figure 6 explains this collapse via the alignment tax. Across runs, the Format Reward (blue) steadily increases, indicating successful syntactic alignment. However, the Correctness Reward (red) remains volatile. In the case of `P2 S2_logic`, the training reward spikes (> 0.9) while validation accuracy remains low, a signature of overfitting to specific training examples rather than acquiring generalizable logic.

6.2. Ablation Experiment: Evaluation of Mitigation Strategies

In the Ablation Experiment, we evaluated five strategies to mitigate forgetting during the Math (A) → Logic (B) sequence and compared them to the baseline. Figure 7 provides an overview of the training dynamics.

Figure 7. **Training Dynamics of Mitigation Strategies.** Experience Replay (Orange) demonstrates the most stable reward recovery in Phase 2.

6.2.1. Optimization Dynamics

The Learning Rate schedule (Figure 7, Top Left) uses a sawtooth pattern with peaks at Step 0 and Step 13. This confirms that all mitigation strategies were subjected to an identical reset stimulus, ensuring that performance differences are attributable to the strategies themselves rather than optimization artifacts.

6.2.2. Comparative Performance Analysis

Table 3 and Figure 8 summarize the final performance. We utilize the Harmonic Mean (H-Score) of task accuracies as the primary metric for stability-plasticity balance.

Table 3. Summary of Mitigation Strategy Performance

Method	Logic Task B	Math Task A	Peak KL	Avg Rwd	Grad Norm	H-Score
Baseline	68.0%	20.0%	0.14	1.09	1.23	30.91
Anchoring (EWC)	66.0%	18.0%	0.04	1.01	0.90	28.29
Ensemble	70.0%	18.0%	0.06	0.97	1.04	28.64
Orthogonal LoRA	66.0%	22.0%	0.04	1.11	0.63	33.00
Task Arith.	**78.0%**	20.0%	0.06	0.97	1.04	31.84
Exp. Replay	76.0%	**26.0%**	0.09	1.17	0.90	**38.75**

Experience Replay emerged as the superior strategy, achieving the highest H-Score (38.75). It improved Math retention by 6% over the baseline and achieved 76% accuracy on Logic. **Task Arithmetic** (Model Merging) demonstrated the highest plasticity (78% Logic) but failed to recover Math capabilities, suggesting that linear combinations of weights are insufficient for restoring reasoning circuits in small models. **Anchoring** proved detrimental; the low KL divergence (0.04) and low Logic score (66%) indicate that the regularization penalty was too severe, causing rigidity that hampered new learning.

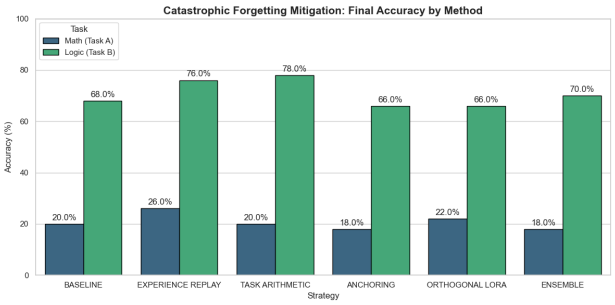

Figure 8. **Final Accuracy by Method.** Experience Replay achieves the highest retention of the old task (26%) while maintaining competitive performance on the new task.

6.2.3. The Stability-Plasticity Frontier

To visualize the trade-offs, we plotted the Pareto frontier in Figure 9.

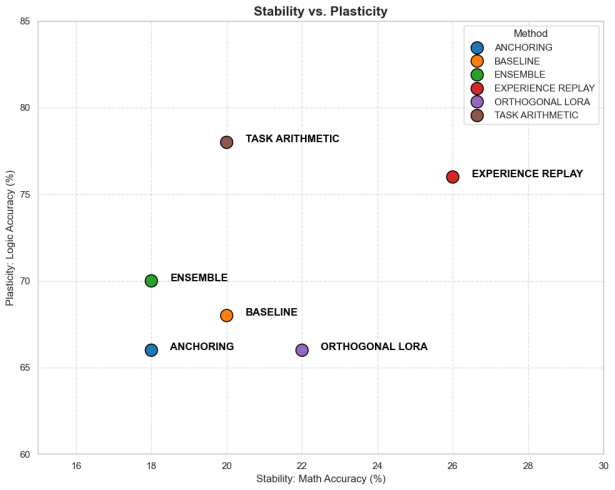

Figure 9. **The Stability-Plasticity Trade-off.** Experience Replay (Red) is the only method that dominates the Baseline on both axes, pushing the frontier outwards.

The analysis is supported by the gradient dynamics shown in Figure 10. The Baseline model exhibits a massive spike in gradient norm (> 2.0) at the task transition point, driving the catastrophic weight updates. Orthogonal LoRA maintains the flattest gradient profile (~ 0.7), effectively isolating parameters, but this lack of gradient flow likely limited its peak performance.

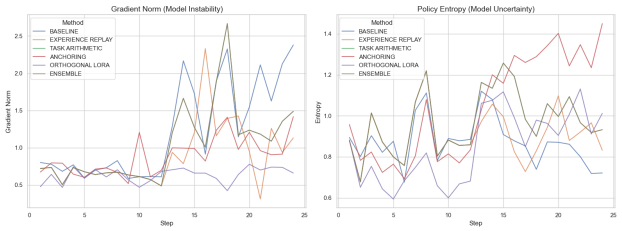

Figure 10. **Gradient Norm and Policy Entropy.** The Baseline's gradient spike (Left, Blue) coincides with forgetting. Anchoring (Right, Red) results in high entropy, indicating model uncertainty.

6.3. Sensitivity Experiment: Scaling Analysis

Finally, we created a Sensitivity Experiment by changing the hyperparameter of model size. We hypothesized that increasing model capacity to 1.5B parameters would reduce catastrophic forgetting and compared the Baseline against Experience Replay.

6.3.1. Stabilization of Training Dynamics

Figure 11 compares the training stability. While the 0.5B model showed moderate instability, the 1.5B Baseline (Blue) exhibited extreme volatility.

- **KL Divergence:** The Baseline drifted to extreme values (> 250.0 nats) around Step 7, compared to typical stable values of < 0.1.
- **Loss Explosion:** Coinciding with the drift, training loss spiked to > 9.

In contrast, Experience Replay (Orange) maintained a near-zero KL drift, proving that data mixing acts as a powerful regularizer even when the model is undergoing massive updates.

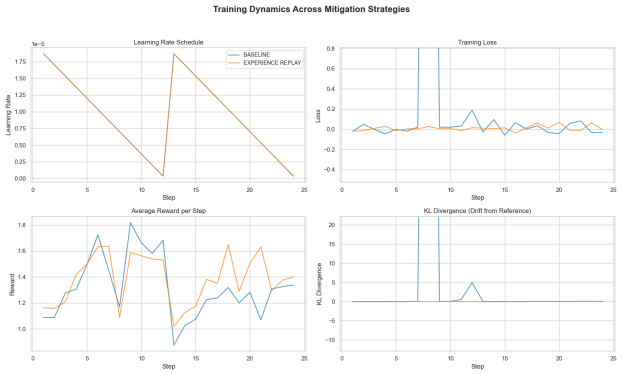

Figure 11. **Scaling Dynamics (1.5B).** Experience Replay (Orange) prevents the explosive KL divergence (> 20.0) and loss spikes (> 0.8) observed in the Baseline (Blue).

6.3.2. Preservation of Cognitive Style

A deeper investigation into the Phase 2 training (Figure 12) reveals an interesting finding regarding Reasoning Depth.

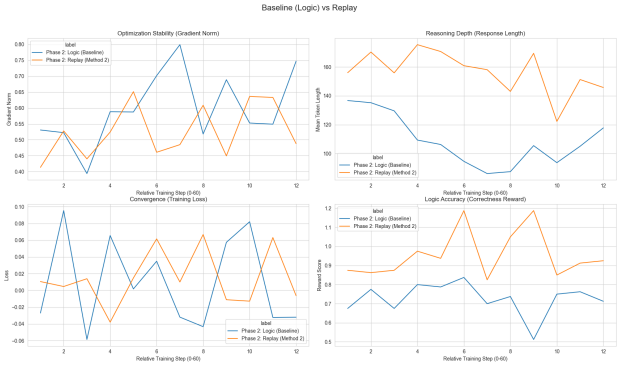

Figure 12. **Phase 2 Deep Dive.** Top-Right: The Baseline model suffers a Reasoning Collapse, with response lengths dropping from 138 to 85 tokens. Replay preserves Chain-of-Thought behavior (120-175 tokens).

As shown in the Reasoning Depth chart (Figure 12, Top-Right), the Baseline model undergoes a cognitive collapse, where the average response length drops from **138 tokens to 85 tokens** before partially recovering. This indicates the model abandoned Chain-of-Thought reasoning to minimize loss via shortcuts. Experience Replay, by enforcing the generation of complex math solutions, preserved the model's tendency to output longer, reasoned responses ($\sim$120-175 tokens).

This structural preservation translated to performance: Experience Replay improved retention of Math (+2.0%) and counter-intuitively improved acquisition of the new Logic task by **4.0%** (94.0% vs 90.0%, see Figure 13). This suggests that preserving the reasoning style of the model is critical in addition to preserving the knowledge itself.

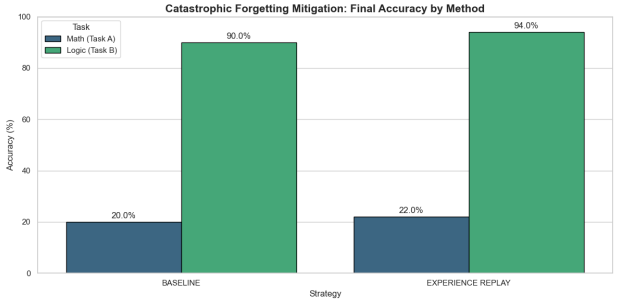

Figure 13. **Final Performance (1.5B).** Experience Replay outperforms Baseline on both retention (Math) and acquisition (Logic).

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023. URL https://arxiv.org/abs/2309.16609.

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

Quy-Anh Dang and Chris Ngo. Reinforcement learning for reasoning in small llms: What works and what doesn't, 2025. URL https://arxiv.org/abs/2503.16219.

DeepSeek-AI Team. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948. arXiv:2501.12948.

Shihan Dou, Yan Liu, Haoxiang Jia, Limao Xiong, Enyu Zhou, Wei Shen, Junjie Shan, Caishuang Huang, Xiao Wang, Xiaoran Fan, Zhiheng Xi, Yuhao Zhou, Tao Ji, Rui Zheng, Qi Zhang, Xuanjing Huang, and Tao Gui. Stepcoder: Improve code generation with reinforcement learning from compiler feedback, 2024. URL https://arxiv.org/abs/2402.01391.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking, 2025. URL https://arxiv.org/abs/2501.04519.

D. Guo et al. Bootstrapping llms to reason over longer horizons via reinforcement learning, 2025. URL https://arxiv.org/pdf/2510.07312. arXiv:2510.07312.

Ilgee Hong, Changlong Yu, Liang Qiu, Weixiang Yan, Zhenghao Xu, Haoming Jiang, Qingru Zhang, Qin Lu, Xin Liu, Chao Zhang, and Tuo Zhao. Think-rm: Enabling long-horizon reasoning in generative reward models, 2025. URL https://arxiv.org/abs/2505.16265.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic, 2023. URL https://arxiv.org/abs/2212.04089.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

Long Li, Jiaran Hao, Jason Klein Liu, Zhijian Zhou, Yanting Miao, Wei Pang, Xiaoyu Tan, Wei Chu, Zhe Wang, Shirui Pan, Chao Qu, and Yuan Qi. The choice of divergence: A neglected key to mitigating diversity collapse in reinforcement learning with verifiable reward, 2025. URL https://arxiv.org/abs/2509.07430.

Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical Report CMU-CS-93-103, Carnegie Mellon University School of Computer Science, Pittsburgh, PA, 1993.

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

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

OpenLMLab, R. Zheng, et al. Secrets of rlhf in large language models part i: Ppo, 2023. URL https://arxiv.org/abs/2307.04964. arXiv:2307.04964.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit

Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

S. Raschka et al. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556. arXiv:2203.15556.

Nicholas Roberts, Niladri Chatterji, Sharan Narang, Mike Lewis, and Dieuwke Hupkes. Compute optimal scaling of skills: Knowledge vs reasoning, 2025. URL https://arxiv.org/abs/2503.10061.

Sneheel Sarangi and Hanan Salam. Small llms do not learn a generalizable theory of mind via reinforcement learning, 2025. URL https://arxiv.org/abs/2507.15788.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

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

Shezheng Song, Hao Xu, Jun Ma, Shasha Li, Long Peng, Qian Wan, Xiaodong Liu, and Jie Yu. How to alleviate catastrophic forgetting in llms finetuning? hierarchical layer-wise and element-wise regularization, 2025. URL https://arxiv.org/abs/2501.13669.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Ziyan Wang, Zheng Wang, Jie Fu, Xingwei Qu, Qi Cheng, Shengpu Tang, Minjia Zhang, and Xiaoming Huo. Slow-fast policy optimization: Reposition-before-update for llm reasoning, 2025. URL https://arxiv.org/abs/2510.04072.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks, 2015. URL https://arxiv.org/abs/1502.05698.

Lorenz Wolf, Robert Kirk, and Mirco Musolesi. Reward model overoptimisation in iterated rlhf, 2025. URL https://arxiv.org/abs/2505.18126.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020. URL https://arxiv.org/abs/1910.03771.

Y. Xiong et al. Sortedrl: Accelerating rl training for llms through online length-aware scheduling. In *OpenReview*, 2025. URL https://openreview.net/pdf?id=YoV9lIZ827.

Z. Yang et al. Towards reasoning ability of small language models, 2025. URL https://arxiv.org/abs/2502.11569. arXiv:2502.11569.

Y. Zhai et al. Uncertainty-penalized reinforcement learning from human feedback with diverse reward lora ensembles, 2024. URL https://arxiv.org/abs/2401.00243. arXiv:2401.00243.

Chen Zheng, Yiyuan Ma, Yuan Yang, Deyi Liu, Jing Liu, Zuquan Song, Yuxin Song, Cheng Ren, Hang Zhu, Xin Liu, Yiyuan Ma, Siyuan Qiao, Xun Zhou, Liang Xiang, and Yonghui Wu. Balanced actor initialization: Stable rlhf training of distillation-based reasoning models, 2025. URL https://arxiv.org/abs/2509.00309.

7. AI usage

Used AI to proofread report, such as by prompting "How can this report be improved" and "Does this report meet the requirements". Used AI to assist in producing code, such as by providing an algorithm and prompting "Provide the python script to implement this algorithm".