Extended Abstract

Motivation and Problem Statement Reinforcement learning with verifiable rewards (RLVR) for language models is highly sensitive to the quality of the supervised fine-tuning (SFT) warmstart. Poor initialization limits the diversity of rollouts the policy can generate, which in turn reduces the signal available for online RL. We investigate how targeted data augmentation during SFT can improve warmstart quality for the Countdown arithmetic reasoning task. Online RL itself also remains expensive because methods such as RLOO require repeatedly generating fresh trajectories during optimization (Ahmadian et al., 2024). We therefore additionally investigate whether replay-based reinforcement learning can improve sample efficiency during RL fine-tuning.

Method We study how supervised fine-tuning (SFT) data quality and reinforcement learning (RL) optimization design affect performance on Countdown arithmetic reasoning under a reinforcement learning with verifiable rewards (RLVR) setup. For SFT, we evaluate augmentation strategies that address approximately $\sim$60% incorrect examples in the base dataset, including correction of erroneous traces, append-based trace refinement, and the addition of synthetic arithmetic and GSM8K-style reasoning problems, all standardized to a consistent `<think>`/`<answer>` format. For RL, we study replay-based optimization with inverse-success weighting inspired by Maximum Likelihood Reinforcement Learning (MaxRL) (Tajwar et al., 2026). Because replay introduces off-policy data, updates apply importance-sampling corrections to mitigate distribution mismatch.

Novelty We study how SFT augmentation and RLVR optimization design interact in Countdown reasoning, comparing multiple data augmentation strategies and finding that preserving exploratory traces via append-based correction is more effective than overwriting or increasing data volume. We further analyze replay-based optimization under fixed gradient budgets in an RLOO-based framework, identifying a consistent tradeoff between pass@1 (accuracy) and pass@k (coverage) across training regimes.

Implementation All experiments use Qwen 2.5-0.5B as the base model. Synthetic problems are generated by brute-force enumeration. Corrections use a verified solver; traces are written to match the original dataset's style and length. GSM8K examples are normalized to the same `<think>`/`<answer>` format as Countdown, with inline calculator annotations stripped and operator spacing standardized. Evaluation uses 50 Countdown problems $\times$ 16 rollouts, scored by the rule-based verifier.

Results Appending format-consistent corrections to original traces achieves pass@16 of 88%, i.e. +10.0 percentage points (pp) over baseline, with competitive pass@1 of 36.8% (+5.4pp), suggesting that preserving exploratory reasoning while adding correctness signal improves high-k performance. Multi-trace augmentation with 3K new problems concentrates its gains at higher k (+4.0pp at pass@16) with more modest gains at low k (+2.6pp at pass@1), while combining it with fixed OG slightly improves low-to-mid-k performance (pass@1 of 36.9%) but slightly reduces pass@16 (86%). Adding GSM8K achieves the strongest pass@1 performance at 39.8% (+8.4pp) while reducing high-k performance, suggesting cross-task transfer primarily improves single-sample accuracy. Replay-based RL similarly exhibits a clear replay–staleness tradeoff. Moderate replay reuse improves sample efficiency and can outperform purely on-policy optimization, while larger replay budgets reduce performance despite importance weighting corrections, suggesting that replay provides useful additional optimization signal but becomes increasingly sensitive to policy drift as trajectories age.

Discussion The key finding is that *how* augmentation is applied matters more than how much data is added. Append-based correction preserves the exploratory search behavior in original traces while adding correctness signal. Multi-trace diversity improves coverage at higher k, while cross-task transfer from GSM8K improves single-sample accuracy at low k, suggesting these two mechanisms target complementary parts of the pass@k curve. Replay-based optimization similarly exhibits a tradeoff: moderate trajectory reuse improves sample efficiency, while aggressive replay introduces instability from stale trajectories. Limitations include the small evaluation set (50 problems), single model scale, and limited repeated trials due to computational constraints. Because most experiments were evaluated using a single training run, some reported differences may partially reflect training stochasticity rather than systematic effects.

Conclusion Effective SFT augmentation for RLVR warmstarting could be done with additive correction, diverse reasoning paths, and format consistency. Append-based correction is the strongest single intervention; combining it with multi-trace synthetic examples gives the best overall warmstart profile for downstream RL fine-tuning. Replay-based optimization further suggests that moderate trajectory reuse can reduce rollout requirements, although aggressive replay introduces instability from stale trajectories.

SFT Augmentation and Replay-Based RL for Countdown Reasoning

Adhi Daiv
Department of Electrical Engineering
Stanford University
adhidaiv@stanford.edu

Aizada Nurdinova
Department of Biomedical Physics
Stanford University
nurdaiza@stanford.edu

Ellie Sampson
Department of Computer Science
Stanford University
esampson@stanford.edu

Abstract

We study how SFT data augmentation affects warmstart quality for reinforcement learning with verifiable rewards on the Countdown arithmetic reasoning task. Using Qwen 2.5-0.5B, we find that appending format-consistent corrections to original traces achieves pass@16 of 88% (+10.0pp over baseline) while preserving exploratory diversity. Multi-trace augmentation with 3K new problems improves coverage at higher k, while GSM8K cross-task mixing improves low-k accuracy; append-based correction dominates at the highest k, suggesting the mechanisms target different aspects of warmstart quality. Preliminary experiments establish that replacing wrong examples hurts, and noisy but exploratory traces in the original dataset carry training signal that cannot be recovered downstream. We further study replay-based RL optimization and find that moderate replay reuse can reduce rollout requirements while remaining competitive with on-policy optimization, although aggressive replay introduces instability from stale trajectories.

1 Introduction

Reinforcement learning with verifiable rewards has emerged as a practical approach to improving reasoning in language models DeepSeek-AI (2025); Pan et al. (2025). A common pipeline applies SFT to warm-start the policy before online RL, using the SFT model as both initialization and KL reference. The quality of this warmstart shapes the diversity of rollouts the policy generates and the density of reward signal available early in training.

A second challenge concerns the cost of online RL itself. Algorithms such as RLOO are typically formulated as on-policy methods and are most commonly applied using freshly sampled trajectories at each update, requiring repeated rollouts throughout training (Ahmadian et al., 2024). While standard formulations assume on-policy data, off-policy reuse of trajectories is possible with appropriate corrections, though it is less commonly explored in RLVR settings due to potential instability from distribution mismatch.

Recent work on Maximum Likelihood Reinforcement Learning (MaxRL) (Tajwar et al., 2026) studies a reweighted perspective on policy optimization that connects reinforcement learning objectives with importance-weighted likelihood maximization, providing a motivation for studying alternative weighting schemes under off-policy data reuse.

We therefore investigate whether explicit trajectory replay, combined with standard importance-style corrections, can improve sample efficiency in RLVR settings for arithmetic reasoning.

The base SFT dataset used in this project (`cog_behav_all_strategies`) contains reasoning traces for Countdown, a task requiring the model to combine a set of numbers using arithmetic operations to reach a target value. An analysis of this dataset reveals that approximately 60% of examples contain arithmetic errors: either wrong final values (42.5%) or violations of the number constraints (17.5%). This raises a natural question: does training on wrong examples hurt, and can targeted augmentation fix it?

We run a series of controlled experiments varying the augmentation strategy while holding the model, task, and evaluation protocol fixed. The goal is to maximize pass@k for $k \in \{1, \ldots, 16\}$, with emphasis on high-k performance as response diversity at inference time provides a richer distribution for downstream RL fine-tuning.

We study (i) how SFT data augmentation affects warmstart quality and (ii) how replay-based RL can reduce sampling costs in RLVR training.

2 Related Work

Data quality for SFT warmstarting has received growing attention in the context of reasoning models. DeepSeek-AI (2025) show that the "aha moment" in RL, where the model learns to self-correct, depends critically on the quality of the SFT initialization. Yang et al. (2024) show that extracting salient entities from a small domain-specific corpus and generating diverse text by drawing connections between them enables more data-efficient knowledge acquisition during continued pretraining, demonstrated on a long-form QA domain. Lee

et al. (2024) show that preference labels generated by an off-the-shelf LLM can match human feedback in downstream RL quality, and that directly using LLM reward scores during RL outperforms training a separate reward model. Pan et al. (2025) have shown that even a small, noisy SFT dataset can serve as an adequate warmstart if it covers the task format, because RLVR will correct the answer distribution.

Replay-based reinforcement learning has long been used to improve sample efficiency by reusing previously collected trajectories, though off-policy training introduces challenges related to distribution shift, including stale trajectories and policy mismatch. Recent work on Maximum Likelihood Reinforcement Learning (MaxRL) (Tajwar et al., 2026) provides a reweighted perspective on policy optimization that connects reinforcement learning with likelihood-based objectives, and highlights how importance-style weighting can be used to reinterpret gradient updates under off-policy data. In contrast, most reinforcement learning with verifiable rewards (RLVR) systems have primarily focused on on-policy optimization due to its simplicity and stability, motivating further investigation into the effectiveness of explicit trajectory replay in reasoning tasks.

3 Methods

3.1 Base Dataset and Audit

The base SFT dataset contains 1,000 training and 200 test examples. Each example is a Countdown problem paired with a reasoning trace in `<think>` tags and a final expression in `<answer>` tags. Our simple scripted analysis finds 40% of examples are fully correct, 42.5% have correct number usage but wrong arithmetic, and 17.5% use numbers not in the input set.

3.2 Preliminary Augmentation Strategies

Preliminary experiments explored three directions to understand what data properties matter for warmstart quality.

Synthetic augmentation. We generate 1,000 new Countdown problems solved by brute-force enumeration. Each example's `<think>` block must end with a `Final expression: solution` line that is character-identical to the `<answer>` tag, directly targeting the connection between the model's reasoning and final answer. Each example shows 1-3 wrong attempts followed by the correct one, with triple verification before saving to ensure answer correctness. Although the style of this reasoning might differ from the original data, synthetic data provides structure and correct examples:

```
<think>
  Try 3 + 5 * 2 = 13, not 11. Skip.
  Try 2 + 9 = 11. Yes!
  Final expression: 2 + 9
</think>
<answer> 2 + 9 </answer>
```

Replacement-based correction. For each wrong example in the SFT dataset, a verified solver finds a valid expression and a correction block is injected. Three variants were explored: appending a self-correction sentence, truncating at a pivot step 4 if no intermediate results were achieved and injecting a scripted bridge, and context-aware bridges matched to the expression structure using an LLM. In all variants, original wrong traces are replaced.

```
<think>
1. First, let's see if we can work backwards from 44:
   - 88/2 = 44, so we need to make 2 from the rest

2. Let's try to get 2 using 83, 10, and 78:
   - 83-78 = 5 (getting closer)
   - 10/5 = 2 (this could work!)        <- truncated here

[LLM or script-generated bridge]:
Wait -- 5 = 83 - 78 is actually the key.
10 / 5 = 2 -- a factor of 44.
Then 88 * 5 = 440, and 440 / 10 = 44. [OK]
</think>
<answer> (88 * (83 - 78)) / 10 </answer>
```

Stacking. Rather than replacing wrong examples, corrected versions are stacked alongside originals. The training set comprises the original 1K noisy traces, 600 programmatically corrected versions, and 200 LLM-written corrections ($\sim$1,800 total).

Multi-task SFT. We also evaluated mixing the countdown dataset with GSM8K arithmetic problems normalized to the same `<think>`/`<answer>` format.

3.3 Final Augmentation Strategies

Informed by preliminary findings, the final runs apply two interventions.

Append-based correction (Fixed OG). Every original trace is preserved in full. For wrong and invalid examples, 2–4 continuation steps are appended that explicitly name the error (wrong arithmetic, illegal number reuse), show corrected intermediate computations, and end with a verification line. Steps are written to match the numbered-step and sub-bullet format of the original dataset. Correct examples receive a short verification step only, e.g. "Let me double-check:", "Verifying the solution:", with a closing "Correct!" / "✓" / "That's the target!". Because corrections are appended in place to each existing trace, the dataset remains $\sim$1,000 training examples.

```
<think>
[full original steps 1-4, including wrong conclusion]

5. Let me write this out cleanly: <- bridge starts
   - 83 - 78 = 5
   - 10 / 5 = 2
   - 88 / 2 = 44
   So (88 * (83 - 78)) / 10 = (88 * 5) / 10 = 440 / 10 = 44.
</think>
<answer> (88 * (83 - 78)) / 10 </answer>
```

Multi-trace augmentation. New Countdown problems not in the original 1K are generated with up to 3 valid solution traces each, covering diverse operator combinations. Traces

follow the original dataset format exactly: motivated exploration, commentary ("too large", "getting closer"), and a separate verification step. We evaluated 3K new traces and found that scaling to 9K hurt performance, suggesting diminishing returns from volume without matched format quality.

Combined dataset. The final combined condition merges 2K multi-trace examples with 1K fixed OG traces ($\sim$3,000 total). A further variant adds 3K GSM8K arithmetic problems normalized to the Countdown format, with inline calculator annotations stripped and operator spacing standardized, giving a 6K mixed dataset.

3.4 RL Objective and Training Setup

We study reinforcement learning with verifiable rewards (RLVR) using a sequence-level policy optimization objective. Let π_θ denote the policy and $R(\tau) \in \{0, 0.1, 1\}$ the terminal correctness reward for a trajectory τ (with 0.1 for correct format but incorrect answer). The objective is:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)].$$

We estimate policy gradients using a group-based leave-one-out (RLOO) estimator. Given a set of K trajectories sampled under the same prompt, the RLOO baseline is computed as the mean reward of the other trajectories in the group:

$$b_{\mathrm{LOO}}(\tau_i) = \frac{1}{K-1}\sum_{j\neq i} R(\tau_j),$$

yielding sequence-level advantages:

$$A_{\mathrm{RLOO}}(\tau_i) = R(\tau_i) - b_{\mathrm{LOO}}(\tau_i).$$

The resulting policy gradient is:

$$\nabla_\theta J \propto \mathbb{E}_{\tau\sim\pi_\theta}\left[A_{\mathrm{RLOO}}(\tau)\nabla_\theta \log \pi_\theta(\tau)\right].$$

MaxRL-style inverse-success weighting. We apply inverse-success weighting inspired by Maximum Likelihood Reinforcement Learning (MaxRL) to reweight trajectory contributions during optimization. This is motivated by the observation that, under both on-policy sampling and replay, high-reward trajectories can become overrepresented in the effective training distribution, reducing gradient diversity and over-amplifying dominant successful modes.

Let $\mathcal{G}(i)$ denote the group of trajectories used to form the baseline for trajectory i (or the set of replay-sampled trajectories in a training step). We define a group-level success estimate:

$$\bar{R}_{\mathcal{G}(i)} = \mathbb{E}_{j\in\mathcal{G}(i)}[R_j],$$

and the inverse-success weight:

$$w_i = \frac{1}{\epsilon + \bar{R}_{\mathcal{G}(i)}}.$$

This weighting increases the contribution of trajectories from low-success groups and reduces the dominance of frequently successful ones, preventing the model from collapsing to a few successful modes already in the replay buffer. This weighting is applied in both training regimes. In both training regimes, regardless of number of gradient updates per rollout, training is performed for exactly 100 gradient updates.

3.5 On-Policy Training (Single-Pass)

Our on-policy baseline performs standard single-pass optimization using freshly sampled trajectories. Each trajectory is used once per gradient update and then discarded. This setting uses RLOO advantages together with MaxRL-style inverse-success weighting, but does not reuse past trajectories.

3.6 Replay-Based Training (Off-Policy Extension)

We extend training with an explicit replay buffer that stores previously sampled trajectories. Each trajectory can be reused for multiple optimization steps, controlled by a gradient budget $G \in \{2, 4, 8\}$.

Concretely, after each newly generated rollout is added to the buffer, we uniformly sample G trajectories from the replay buffer and perform a separate gradient update for each sampled trajectory. This results in G replay-driven optimization steps per newly generated rollout, increasing sample efficiency while keeping rollout generation fixed.

Because replayed trajectories are generated under older policies, we correct for distribution shift using importance sampling. Let π_{old} denote the behavior policy under which a trajectory was collected. We compute:

$$\rho(\tau) = \frac{\pi_\theta(\tau)}{\pi_{\mathrm{old}}(\tau)}.$$

Policy optimization uses a clipped surrogate objective:

$$\mathcal{L}_{\mathrm{policy}}(\theta) = -\mathbb{E}\left[\min\left(\rho(\tau)\,w(\tau)\,A(\tau), \mathrm{clip}(\rho(\tau), 1\pm\epsilon)\,w(\tau)\,A(\tau)\right)\right],$$

where $w(\tau)$ is the inverse-success weight and $\epsilon = 0.2$.

3.7 Advantage Estimation

We evaluate two advantage estimators within both training regimes.

Sequence-level. We use the group-based leave-one-out estimator defined above to compute sequence-level advantages:

$$A(\tau_i) = R(\tau_i) - b_{\mathrm{LOO}}(\tau_i).$$

Value-based. We train a learned value function $V_\phi(s_t)$ over hidden states to predict expected terminal reward from state s_t and compute token-level advantages:

$$A_t = R(\tau) - V_\phi(s_t),$$

where $R(\tau)$ is the terminal reward. The value function is trained using Monte Carlo returns, and we apply a short warmup phase before enabling policy updates.

3.8 Novelty

Our work studies RLVR performance through the joint lens of supervised fine-tuning (SFT) warmstarting and reinforcement learning optimization. Unlike prior work that treats data augmentation or RL training in isolation, we systematically

compare multiple SFT augmentation strategies and show that preserving original erroneous traces with appended corrections is more effective than overwriting them, suggesting that exploratory structure in data plays a key role in downstream RL performance. We further investigate replay-based optimization under fixed gradient budgets in an RLOO-based RLVR setup, isolating the effect of trajectory reuse under controlled off-policy corrections. Finally, we identify a consistent separation between pass@1 (accuracy) and pass@k (coverage), showing that different augmentation strategies preferentially improve one regime over the other.

3.9 Experimental Setup

Task and Dataset. We study Countdown arithmetic reasoning: given a set of 3–4 integers, the model must produce an arithmetic expression (using $+$, $-$, $\times$, $\div$) that equals a target value. The base SFT dataset (`cog_behav_all_strategies`) contains 1,000 training and 200 test examples, each pairing a Countdown problem with a reasoning trace in `<think>` tags and a final expression in `<answer>` tags. RL training uses problem prompts from `asingh15/countdown_tasks_3to4`. All experiments use Qwen 2.5-0.5B as the base model, fine-tuned with full parameter updates in bfloat16 precision.

SFT Training. SFT uses AdamW with learning rate 5×10^{-5}, a cosine schedule with 5% linear warmup, weight decay 0.01, and gradient clipping at 1.0. We train for 6 epochs with per-device batch size 64 and gradient accumulation over 8 steps (effective batch size 512). Maximum prompt and response lengths are 512 and 1,024 tokens, respectively. Gradient checkpointing is enabled throughout.

RL Training. On-policy RLOO uses learning rate 1×10^{-5} with a constant schedule, per-device batch size 128, gradient accumulation 128, rollout group size K=8, entropy coefficient 0.001, and KL penalty coefficient 0.001. Sampling during training uses temperature 1.0 (top-p=1.0, top-k disabled). All conditions train for exactly 100 gradient updates, fixing the gradient budget for controlled comparison. Maximum context length is 2,048 tokens.

Off-Policy Extension. Off-policy runs add a replay buffer (size 10), PPO clipping with ϵ=0.2, and importance-sampling corrections. The gradient budget $G \in \{2, 4, 8\}$ controls the number of replay-driven gradient updates per newly sampled rollout. Value-baseline conditions train a Monte Carlo value function (pure terminal reward targets; n=2048 step returns) with a 50-rollout warmup before policy updates are enabled. Gradient clipping is applied independently to the backbone and value head to prevent the 500M backbone from dominating the value head gradient norm. All off-policy runs otherwise share the same optimizer and budget settings as the on-policy baseline.

Baselines. For SFT, the unaugmented *Baseline SFT* is trained on the original 1,000 examples without modification, establishing a data-quality lower bound against which augmentation gains are measured. For RL, the *On-Policy Max RLOO* baseline applies RLOO with MaxRL inverse-success weighting and no replay. This is the natural comparison point because RLOO is the standard RLVR algorithm in prior work on small-scale reasoning (Ahmadian et al., 2024; Pan et al., 2025), and applying MaxRL weighting uniformly across all conditions isolates the effect of replay from the effect of the weighting scheme.

Evaluation. We evaluate pass@k for $k \in \{1, 2, 4, 8, 16\}$ on 50 held-out Countdown problems with 16 rollouts each (800 total responses per condition), scored by a rule-based verifier that assigns reward 1 for a valid expression using exactly the given numbers that evaluates to the target, and 0 otherwise. Pass@k is estimated as the probability that at least one of k independent samples is correct. We report across the full range of k because pass@1 and pass@16 capture distinct warmstart properties: pass@1 reflects greedy accuracy, while pass@16 reflects coverage—how often any correct solution appears in the rollout distribution. High-k coverage is particularly important for downstream RLVR, where a more diverse rollout distribution provides denser reward signal early in training. Evaluation sampling uses temperature 0.6, top-p=0.95, and top-k=20. The 50-problem evaluation set was chosen to fit within a single GPU evaluation pass while providing stable estimates at k=16; we acknowledge in the limitations that this size constrains statistical power.

Hardware. All training and evaluation runs on a single NVIDIA H100 GPU via Modal, with a 24-hour job timeout.

4 Results

4.1 Preliminary Augmented Dataset Experiments

Figure 1 shows pass@k curves for five preliminary conditions. Three findings shaped the final augmentation strategy.

More Countdown examples drive gains at low k. Adding 1K synthetic examples improves pass@1 from 31.4% to 38.8%, despite the format not matching the original dataset exactly. The gain is concentrated at low k, suggesting the model learns to commit more decisively to a verified answer. We attribute this to the explicit try-evaluate-reject structure in the synthetic traces, which teaches the model that `<answer>` should be the last expression confirmed in `<think>`. Format consistency proved necessary to achieve any pass@k improvement, and scaling synthetic data beyond 1K yielded diminishing returns, likely due to the mechanistic nature of the generated reasoning.

Replacing wrong examples hurts. Programmatically replacing the $\sim$60% of wrong examples degrades performance across all k. Even arithmetically incorrect traces carry useful signal: they demonstrate exploratory search behavior that the model needs to learn *how* to reason, not just what correct answers look like. This signal cannot be recovered by downstream RLVR if it was never present in the warmstart.

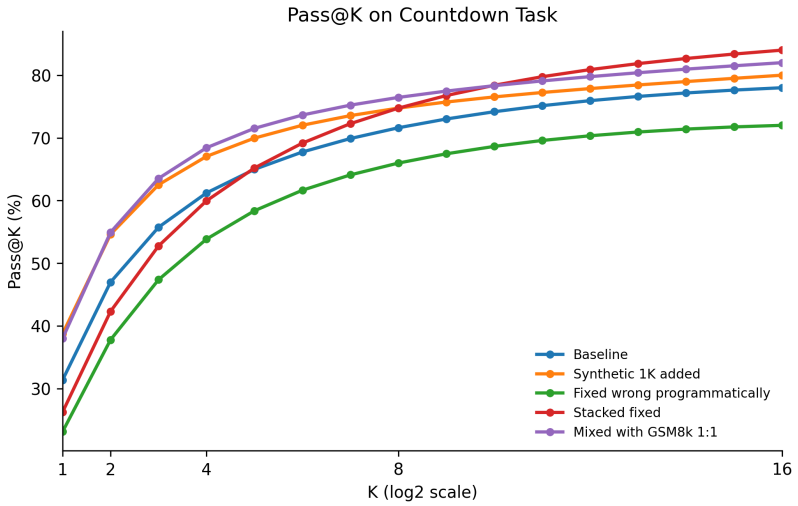

Figure 1: Pass@k curves for preliminary augmentation conditions. Replacing wrong examples programmatically (green) is the only condition that degrades performance across all k. Synthetic augmentation (orange) gives the best pass@1. Stacking corrected traces (red) achieves the best pass@16. GSM8K mixing (brown) closely tracks stacked fixed at low k and converges near it at high k.

Stacking corrections preserves diversity while adding correctness signal. Keeping the original noisy traces and stacking corrected versions alongside them avoids the truncation problem. This achieves the best pass@16 among preliminary conditions (+6.0pp over baseline), at the cost of weaker pass@1, a reasonable tradeoff for a warmstart entering RLVR.

Mixing with GSM8K arithmetic problems gives further gains at high k, suggesting cross-task reasoning transfer, though pass@1 remains below the synthetic augmentation condition.

These experiments establish five design requirements for the final experiment: (1) verified correct expressions; (2) diverse reasoning paths; (3) balanced 3- and 4-number difficulty; (4) trace format and length matched to the original; and (5) additive rather than replacement correction.

4.2 Final Augmentation Results

Results are shown in Table 1 and Figure 2. Due to the large number of experiments and computational constraints, it was not feasible to test multiple seeds.

Appending corrections to original traces is the strongest single intervention. Fixed OG (append) retains every original trace in full and appends 2–4 format-consistent LLM-generated continuation steps that explicitly name the error, show corrected arithmetic, and end with a verification line. This achieves pass@16 of 88% (+10.0pp over baseline) while keeping pass@1 competitive at 36.8%. Critically, this approach preserves the full exploratory structure of the original traces, consistent with the finding from preliminary experiments that truncation is harmful.

Multi-trace augmentation improves high-k coverage but not single-sample accuracy. Adding 3K new problems with multiple valid solution traces per problem concentrates its gains at higher k, improving pass@8 (+5.2pp) and pass@16 (+4.0pp) over baseline, while pass@1 gains are modest (+2.6pp), below both the synthetic and correction conditions. Increasing to 9K multi-trace examples did not further improve performance and in some configurations hurt it, suggesting diminishing returns from volume alone when trace format and length distribution are not tightly controlled.

Combining corrections and multi-trace trades high-k coverage for balanced performance. The combined condition (2K multi-trace + 1K fixed OG) achieves the best pass@8 of 81.8% and edges out fixed OG alone at every $k \leq 9$ (e.g. pass@1 of 36.9% and pass@8 of 81.8% vs 36.8% and 81.4%), but pass@16 drops to 86% compared to 88% for corrections alone. This suggests that while multi-trace diversity broadens mid-range coverage, it partially dilutes the high-k gains from correction, likely by introducing more varied expressions but less varied in reasoning and consistently formatted traces.

Adding GSM8K improves pass@1 at the cost of pass@8. The full combined condition (2K multi-trace + 1K fixed OG + 3K GSM8K) achieves the best pass@1 of 39.8% (+8.4pp over baseline), suggesting cross-task arithmetic transfer from GSM8K improves single-sample accuracy. However, pass@8 drops to 78.9% compared to 81.8% for the combined condition without GSM8K, indicating that the addition of out-of-domain data slightly reduces Countdown-specific coverage at intermediate sample budgets.

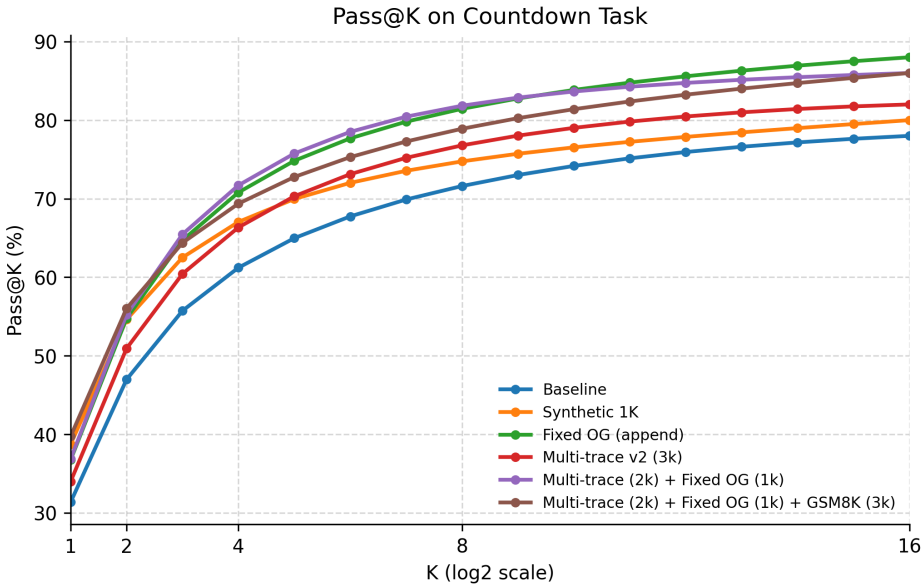

Figure 2: Pass@k curves for final chosen augmentation conditions. The green curve (data with appended corrections) achieves the best pass@16 of 88%. Combining multi-trace new synthetic examples with fixed OG (purple) is slightly better than fixed OG alone at low-to-mid $k \leq 9$ but falls slightly short of it at the highest k. Adding GSM8K (brown) gives the best pass@1 but lower pass@16 than fixed original SFT. Multi-trace alone (red) is inferior to the correction-based conditions across all k; synthetic 1K (orange) is competitive at pass@1 but falls behind at $k \geq 2$.

Table 1: Pass@k on the Countdown test set, final augmentation conditions.

Condition	pass@1	pass@2	pass@4	pass@8	pass@16
Baseline SFT	31.4%	47.0%	61.2%	71.6%	78.0%
Synthetic 1K	38.8%	54.6%	67.0%	74.8%	80.0%
Fixed OG (append)	36.8%	54.8%	70.8%	81.4%	**88.0%**
Multi-trace v2 (3K)	34.0%	50.9%	66.3%	76.8%	82.0%
Multi-trace (2K) + Fixed OG (1K)	36.9%	55.2%	**71.7%**	**81.8%**	86.0%
Multi-trace (2K) + Fixed OG (1K) + GSM8K (3K)	**39.8%**	**56.0%**	69.4%	78.9%	86.0%

4.3 Preliminary Off-Policy Experiments

Four rounds of development shaped the final off-policy configuration, each identifying a concrete failure mode and motivating a targeted fix.

Run 1: Baseline comparison. Three conditions were evaluated: standard on-policy RLOO, on-policy RLOO with the MaxRL inverse-success weighting, and off-policy MaxRL inverse-success weighting with replay enabled (buffer size 10, G=1) Figure 3.

Run 2: PPO clipping and grad-step sweep. With clipping in place, the logged `importance_ratio_clipped_frac` increased monotonically with G, confirming that heavier replay pushes the policy further from the collection policy before the buffer refreshes. Among the sweep conditions, G=4 generally produced the best pass@k, while G=8 began to degrade — consistent with clipping being insufficient to stabilize very stale batches. This established that moderate reuse (G=2–4) is the viable operating range.

Run 3: Learned value function — failure and diagnosis. Run 1 showed that the RLOO leave-one-out baseline has high variance when the replay buffer mixes batches from different policy versions. A learned value function was added as a lower-variance alternative, with n=4 step returns and a 20-step critic warmup. The run failed: `warmup/value_loss` was flat at ∼0.002 across all 20 warmup steps and RL training showed no improvement. The root cause was that with n=4 and responses up to 1024 tokens, only the final 4 tokens of each response received the terminal reward R as their value target; all other positions bootstrapped from a near-zero initialized critic. Approximately 99.6% of tokens therefore trained the critic to predict zero, which it already did. The expected warmup loss if every token saw the real reward is $0.29^2 \approx 0.084$; the observed loss of ∼0.002 is roughly $4/1024 \approx 0.4\%$ of that, confirming the critic received almost no useful signal. As a consequence, token-level advantages were near zero for

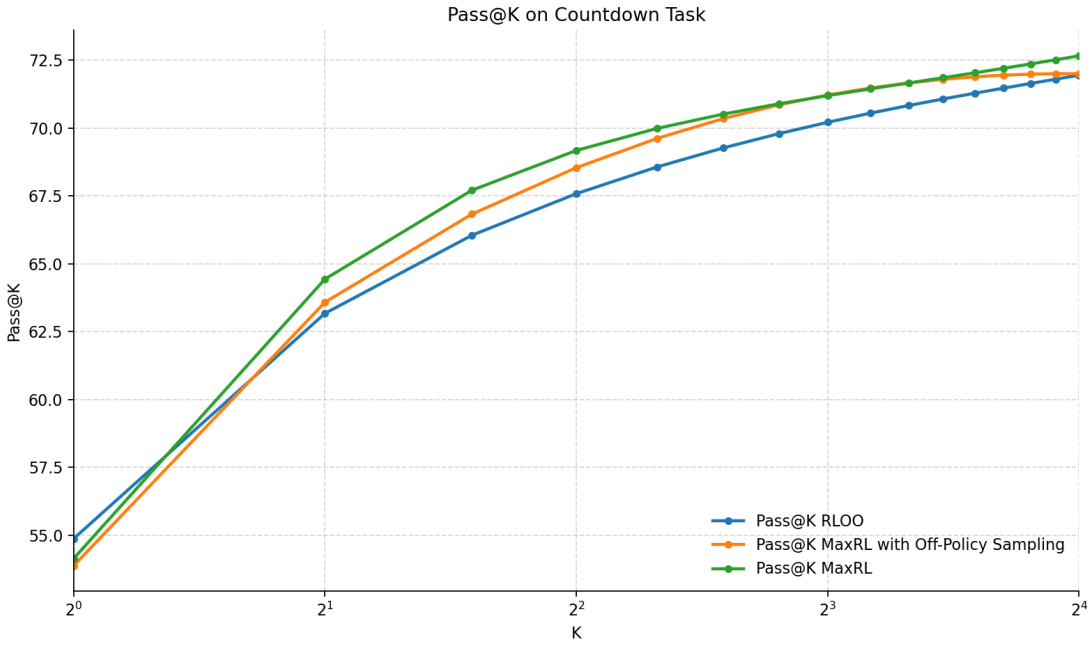

Figure 3: Pass@k curves for baseline comparisons. “MaxRL" indicates inverse-success weighting was used. Inverse-success weighting off-policy uses $G = 1$.

most of the response and the policy received almost no gradient from the vast majority of its outputs.

Run 4: Fixes and final configuration. Three targeted fixes addressed the Run 3 failures. First, n was set to 2048, beyond the maximum response length, so every token's value target is the terminal reward R (pure Monte Carlo). This raised the expected warmup loss to $\sim$0.084 and provided real signal at every token position. Second, gradient clipping was applied independently to the backbone and value head: joint clipping scaled the value head's gradient to near zero on every step because the $\sim$500M backbone parameters dominated the gradient norm. Third, the warmup was extended to 50 rollouts. Additional fixes included correcting an argparse default that silently kept n=4 when not explicitly passed and allowing backbone gradients to flow during warmup (only policy loss is zeroed). After these fixes, `warmup/value_loss` decreased steadily from $\sim$0.084, confirming the critic was learning a meaningful baseline before policy gradients were enabled.

4.4 Final Off-Policy Results

Results are shown in Figure 4 and Table 2. Due to the large number of experiments and computational constraints, it was not feasible to test multiple seeds. All off-policy runs use the RLOO with MaxRL-inspired inverse-success weighting with PPO clipping (ε=0.2), replay buffer size 10, and Monte Carlo value targets (n=2048). The on-policy baseline uses the same, but with no replay buffer (only the most recent rollout is used for gradient updates).

On-policy RLOO is difficult to outperform at low k. The on-policy baseline achieves pass@1 of 54.1%, well above every off-policy variant. Value-baseline conditions cluster between 33–36% at pass@1 while sequence-level conditions span 20–32%, degrading sharply with G. That all off-policy variants underperform at pass@1 despite the MaxRL estimator indicates that importance-sampling correction does not fully compensate for distribution mismatch from stale replay trajectories at the single-sample operating point.

Replay reuse exhibits a clear performance–staleness trade-off. Pass@1 degrades monotonically with G for both baseline types. For sequence-level baselines the collapse is severe: 32.1% at G=2, 25.8% at G=4, and 20.9% at G=8. For value baselines the degradation is milder: 36.0%, 35.4%, and 33.1% respectively, indicating that the learned value function stabilizes training under heavier replay.

Sequence-level G=2 outperforms on-policy at high k. Despite weak pass@1, the sequence-level G=2 condition surpasses the on-policy baseline around k=8–9, reaching pass@16 of 76.0% versus 72.7% for on-policy — a 3.3pp gain. This reflects a genuine sample-efficiency benefit: fewer fresh rollouts are needed to achieve the same high-k coverage, at the cost of substantially worse greedy accuracy.

Value baselines improve robustness under larger reuse budgets. At G=2, the value baseline has slightly lower pass@16 than the sequence-level condition (74.0% vs. 76.0%) but better pass@1 (36.0% vs. 32.1%). As G increases, the value baseline's relative advantage grows: at G=4 it retains pass@16 of 74.0% while the sequence-level condition drops to 68.0%,

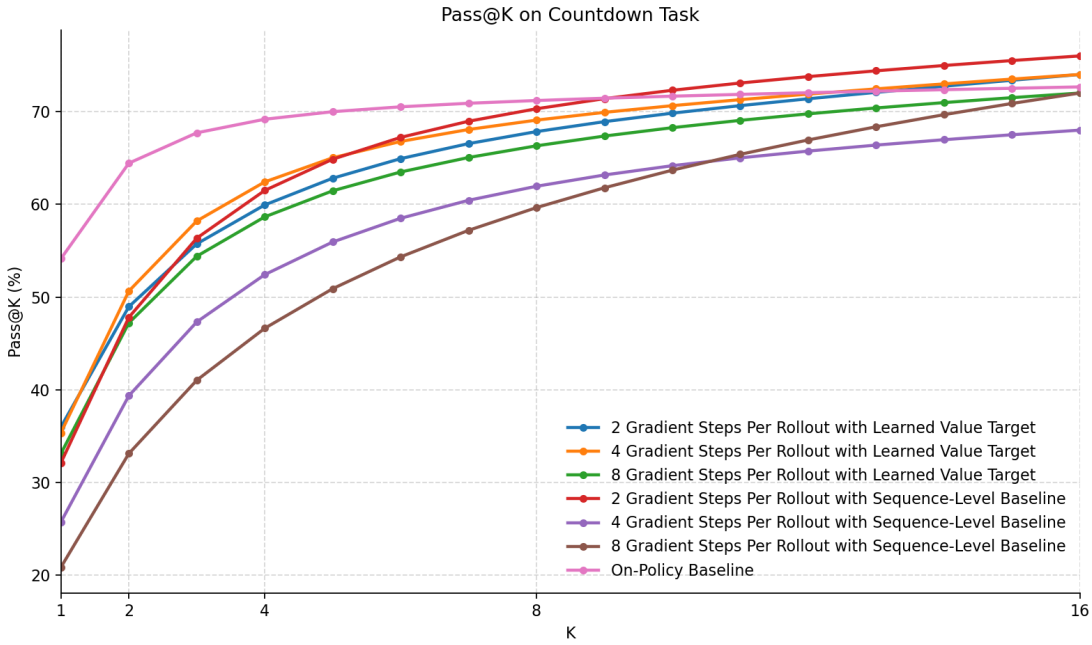

Figure 4: Pass@k curves for off-policy conditions). Replacing wrong examples programmatically (green) is the only condition that degrades performance across all k. Synthetic augmentation (orange) gives the best pass@1. Stacking corrected traces (red) achieves the best pass@16. GSM8K mixing (brown) closely tracks stacked fixed at low k and converges near it at high k.

and at G=8 the value baseline maintains a ∼12pp advantage at pass@1 (33.1% vs. 20.9%) despite both converging to pass@16 of 72.0%. The token-level advantage estimate is less sensitive to the batch mixing introduced by replay, making it the more stable choice when trajectory staleness is high.

Value function warmup is critical. An ablation at matched G=2 with buffer size 10 (Figure 3) reveals a consistent ∼10pp gap in favor of the value baseline across all k. Without it, pass@1 collapses to ∼20%, demonstrating that the sequence-level baseline is insufficient for off-policy batches that mix trajectories from different policy versions.

5 Discussion

A central finding is that for SFT warmstarting before RLVR, *how* augmentation is applied matters more than *how much* data is added. Replacing wrong examples removes the exploratory structure that helps the policy generate diverse rollouts early in RL training. Appending corrections preserves this structure while adding a correctness signal. Volume alone, as seen with 9K multi-trace examples, does not substitute for format fidelity and trace quality/diversity.

A secondary finding is the asymmetry between pass@1 and pass@16 across conditions. Synthetic examples with structured try-evaluate-reject reasoning improve pass@1 most, while data correction and multi-trace augmentation improve higher-k coverage most. This suggests a distinction between *accuracy* (the model reliably commits to the right answer) and *coverage* (the model can find a correct answer given multiple attempts). Both matter for RLVR warmstarting, but through different mechanisms.

A further observation is that multi-trace augmentation underperforms in the final evaluation round relative to what its data volume would predict: despite adding 3K new problems, it improves pass@1 by only +2.6pp, well below the +7.4pp gain from the smaller 1K synthetic condition. We suspect this reflects format inconsistency within the multi-trace dataset rather than a limitation of trace diversity itself, consistent with our broader finding that format fidelity matters more than volume.

A final observation concerns replay-based optimization. Moderate replay reuse improves sample efficiency and can outperform strictly on-policy optimization, suggesting that RLVR does not inherently require fresh trajectories at every update step. However, increasing replay budgets produces a clear performance–staleness tradeoff: repeatedly optimizing over older trajectories eventually degrades performance despite importance-weighting corrections. This suggests that replay is beneficial primarily as a mechanism for moderate data reuse rather than aggressive trajectory recycling (in Table 2, $G = 4$ performs better than $G = 2$ and $G = 8$ when a learned value baseline is used). We further observe that learned value baselines become increasingly important as replay increases (in Table 2, this is evident from the fact that $G = 2$ gives us the best results using the sequence-level baseline). Sequence-level advantages remain competitive at small replay budgets, but value-based estimation becomes more robust under larger replay settings, suggesting that state-conditioned estimates help stabilize optimization when replay introduces distribution shift and stale trajectories.

Table 2: Pass@k on the Countdown test set, final off-policy RL conditions. On-policy baseline is Max RLOO.

Condition	pass@1	pass@2	pass@4	pass@8	pass@16
On-Policy Max RLOO (baseline)	54.1%	64.4%	67.7%	69.2%	72.7%
G=2, Value Baseline	**36.0**%	49.0%	55.7%	59.9%	74.0%
G=4, Value Baseline	35.4%	**50.7**%	**58.2**%	**62.4**%	**74.0**%
G=8, Value Baseline	33.1%	47.2%	54.4%	58.6%	72.0%
G=2, Sequence-Level Baseline	**32.1**%	**47.8**%	**56.4**%	**61.5**%	**76.0**%
G=4, Sequence-Level Baseline	25.8%	39.4%	47.3%	52.4%	68.0%
G=8, Sequence-Level Baseline	20.9%	33.1%	41.0%	46.6%	72.0%

Limitations include the small evaluation set (50 problems), single model scale, and limited repeated trials due to computational constraints. Because most experiments were evaluated using a single training run, some reported differences may partially reflect training stochasticity rather than systematic effects.

6 Conclusion

For SFT warmstarting before RLVR, how augmentation is applied matters more than how much data is added. Append-based correction is the strongest single intervention, reaching pass@16 of 88% by preserving exploratory traces while adding correctness signal. Multi-trace augmentation on its own concentrates its gains at higher k, and GSM8K cross-task mixing improves low-k accuracy; however, combining multi-trace with the append-based corrections improves accuracy at $k \leq 9$ but falls below corrections alone at the highest k. Based on these findings, we use the multi-trace synthetic + append-based correction combination as the SFT warmstart for downstream RL experiments, balancing single-sample accuracy with coverage diversity.

For RL optimization, we find that moderate replay reuse can improve sample efficiency and outperform strictly on-policy training, suggesting that RLVR does not inherently require fresh trajectories for every update. However, larger replay budgets introduce a clear performance–staleness tradeoff, where repeated optimization over older trajectories eventually degrades performance despite importance-weighting corrections. Value-based baselines become increasingly important in this regime, indicating that state-conditioned estimates provide more robust optimization under replay than sequence-level baselines alone.

Taken together, these results suggest that effective RLVR systems require both high-quality warmstarts and carefully controlled replay: better initialization improves the distribution of trajectories available for optimization, while moderate trajectory reuse can reduce online training costs without sacrificing performance.

Future Directions Future work should evaluate whether the observed effects of SFT augmentation and replay-based optimization persist at larger scales and across multiple random seeds, as our experiments were conducted at a single model scale with limited repeated trials. It would also be valuable to study more advanced off-policy RLVR methods beyond RLOO-style updates, including stronger value learning and more principled sequence-level credit assignment. On the data side, further work could develop methods that explicitly balance exploratory trace diversity with correctness, rather than relying on heuristic append-based or synthetic augmentation strategies. Finally, more principled replay mechanisms that adaptively control trajectory reuse could help better characterize the performance–staleness tradeoff observed in our experiments.

7 Team Contributions

- **Adhi Daiv:** SFT Augmentation experiments and synthetic data generation; Off-policy RL algorithm design, implementation, experimental analysis and write-up.
- **Aizada Nurdinova:** SFT augmentation experiments; synthetic data generation and dataset correction pipeline. Write-up on the augmentation.
- **Ellie Sampson:** Off-policy RL algorithm, Evaluation framework; experimental analysis and write-up.

Changes from Proposal Augmentation strategy shifted from pure synthetic generation to a combined correction-plus-augmentation approach after preliminary experiments showed replacement-based correction to be harmful.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. arXiv:2402.14740 [cs.LG] `https://arxiv.org/abs/2402.14740`

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948* (2025).

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash.

2024. RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. *arXiv preprint arXiv:2309.00267* (2024).

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. 2025. TinyZero. `https://github.com/Jiayi-Pan/TinyZero`.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. 2026. Maximum Likelihood Reinforcement Learning. arXiv:2602.02710 [cs.LG] `https://arxiv.org/abs/2602.02710`

Zitong Yang, Neil Band, Shuangping Li, Emmanuel Candès, and Tatsunori Hashimoto. 2024. Synthetic Continued Pretraining. *arXiv preprint arXiv:2409.07431* (2024).