Extended Abstract

Motivation Large-language models (LLMs) struggle with multi-step arithmetic reasoning despite strong performance on language tasks. Supervised fine-tuning (SFT) can teach correct reasoning traces but offers no self-correction at inference time, while existing reward-driven methods either suffer high variance (vanilla REINFORCE) or over-dampen the learning signal (large baselines). We investigate Tapered Off-Policy REINFORCE (TOPR), which dynamically clips importance weights to retain strong updates from rare successful rollouts and gradually anneals toward unbiased on-policy gradients, aiming to combine stability with efficient exploration in the "Countdown" arithmetic task.

Method TOPR uses importance sampling to downweight negative trajectories that are not likely under the policy, while allowing positive trajectories to be upweighted irrespective of the policy. Unlike other LLM algorithms in the REINFORCE family, TOPR does not require an explicit KL penalty to guarantee stable behavior, making it both simpler to implement and computationally more efficient. TOPR itself corresponds to a range of truncation limits that combine the desirable properties of each of these methods into one learning rule.

Implementation We fine-tuned Qwen2.5-0.5B on randomly sampled prompts from the TinyZero Countdown dataset in experimenting the TOPR method. We experimented four TOPR configurations based on RLOO by varying the sample size (500 to 1500), batch size and k (1 to 4) and epochs (1–2). All runs used greedy decoding, a learning rate of $1e-6$ and maximum sequence length 256. We tracked batch-average policy-gradient (PG) loss and the "valid equation" rate (fraction of generated equations that exactly solved the puzzle).

Results In our TOPR experiments, we find that the interplay between the off-policy baseline size k, batch size, and number of data passes critically governs both the signal-to-noise ratio of the policy-gradient updates and their effective magnitude. A modest baseline (k=2) combined with small batch size (2 samples) and two training epochs produces the most rapid and stable reduction in PG loss. By contrast, increasing k and the batch size to 4 under a single epoch reduces per-step variance further but also shrinks the expected gradient, yielding only a gradual loss climb.

Discussion TOPR extends RLOO by tapering the importance-weight clipping threshold: high early thresholds preserve strong gradient "kicks" from rare correct rollouts, while gradual decay ensures unbiased on-policy learning. Empirically, a moderate baseline (k = 2), small mini-batches (2), and two epochs yield the fastest, lowest-variance PG-loss reduction, whereas larger baselines or batch sizes overly damp updates and k = 1 reintroduces noise, slowing convergence. This identifies a practical sweet spot—balancing baseline size, batch dimensions, and data passes is key for stable yet responsive learning under sparse, binary rewards.

Conclusion Supervised Fine-Tuning (SFT) reliably imitates gold reasoning but lacks self-correction. We experimented Tapered Off-Policy REINFORCE (TOPR), which dramatically attenuates the variance of reused samples without discarding their learning signal by clipping importance weights and asymmetrically weighting positive vs. negative reward trajectories. With a moderate baseline (k = 2), small batches, and multiple epochs, TOPR achieves fast, low-variance convergence on the Countdown task. Our results highlight the necessity of variance-reduction strategies for sparse, binary rewards and show that combining supervised imitation with calibrated off-policy corrections yields robust, self-correcting mathematical reasoning in LLMs. Future work will explore adaptive clipping, richer rewards, and scaling to more complex reasoning benchmarks.

Fine-tuning Large Language Models via Tapered Off-Policy REINFORCE (TOPR)

Mengge Pu
Department of Computer Science
Stanford University
mpu217@stanford.edu

Abstract

As modern Reinforcement Learning (RL) fine-tuning of large language models (LLMs) becomes ever more compute- and data-intensive, improving sample efficiency and training stability is critical. Although simply reapplying vanilla policy gradients to stale data can blow up variance and introduce bias, recent work on Tapered Off-Policy REINFORCE (TOPR) (Roux et al. 2025 (1)) offers a principled remedy. By clipping importance weights and asymmetrically weighting positive vs. negative reward trajectories, TOPR dramatically attenuates the variance of reused samples without discarding their learning signal. In addition, its baseline term doubles as a 'difficulty knob', allowing us to emphasize harder examples through a simple constant change. Integrating these techniques into our RLOO loop promises to reduce environment calls by an order of magnitude, stabilize convergence, and accelerate fine-tuning of Qwen 2.5 on both instruction-following and multi-step math tasks, all while preserving the core on-policy guarantees that make policy gradient methods so effective.

1 Introduction

The ability of large language models (LLMs) to perform multi-step mathematical reasoning remains a critical challenge in natural language processing. In this work, we focus on the "Countdown" task — a constrained arithmetic puzzle that probes an LLM's capacity to combine digits and operators to reach a target number—using the Cognitive Behaviors Dataset as our benchmark. While supervised fine-tuning (SFT) has shown promising improvements by teaching an LLM to mimic correct step-by-step reasoning, reinforcement-based approaches such as Reinforcement Learning from Learned Off-Policy Optimizer (RLOO) and Direct Preference Optimization (DPO) offer pathways to directly incorporate reward signals reflecting solution correctness. Building on these foundations, we introduce Tapered Off-Policy REINFORCE (TOPR), an off-policy policy gradient variant tailored for stable and efficient fine-tuning of LLMs. By comparing SFT, RLOO, DPO, and TOPR in accommodating complex arithmetic prompts, our goal is to quantify how each method affects both the accuracy of final expressions and the fluency of intermediate reasoning steps. In doing so, we aim to identify which combination of supervised and reward-driven techniques yields the most reliable performance on countdown-style math problems, thereby contributing practical insights into fine-tuning protocols for enhanced numerical reasoning in LLMs.

2 Related Work

Reinforcement learning from human feedback (RLHF) has rapidly evolved from classic on-policy PPO methods toward more sample-efficient and stable alternatives. Direct Preference Optimization (DPO) reframes the RLHF pipeline as a single classification objective, delivering stable and lightweight fine-tuning without explicit reward modeling or policy sampling loops (Rafailov et al. 2024 (2)). However, DPO treats each full response as a bandit arm and therefore cannot exploit tokenlevel credit assignment, limiting its granularity and preventing the reuse of negative examples in a principled way (Rafailov et al. 2024 (3)).

Recently, Any-Generation Reward Optimization (AGRO) unifies on- and off-policy data under a single policy-gradient objective, leveraging "generation consistency" to blend freshly sampled and replayed trajectories with theoretical convergence guarantees (Tang et al. 2025 (4)). Although AGRO demonstrates that off-policy data can in principle improve downstream performance, it still relies on full importance ratios, leaving gradient variance high and failing to distinguish between positive vs. negative reward trajectories.

Similarly, Trajectory Balance with Asynchrony (TBA) decouples exploration and learning for LLM post-training by running distributed actors into a shared replay buffer, then asynchronously updating the policy via a trajectory-balance loss (Bartoldson et al. 2025 (5)). TBA uses a larger fraction of compute on search, constantly generating off-policy data for a central replay buffer. A training node simultaneously samples data from this buffer based on reward or recency to update the policy using Trajectory Balance (TB), a diversity-seeking RL objective introduced for GFlowNets (10). On mathematical reasoning, preference-tuning, and automated red-teaming (diverse and representative post-training tasks), TBA produces speed and performance improvements over strong baselines.

Other recent off-policy methods are, for example, Weighted Preference Optimization (WPO) reweights off-policy preference pairs by current policy likelihood (Zhou et al. 2024 (6)). This method adapts off-policy data to resemble on-policy data more closely by reweighting preference pairs according to their probability under the current policy. This method not only addresses the distributional gap problem but also enhances the optimization process without incurring additional costs. Another example would be approaching RHLF from a game-theoretic perspective (Zhang et al. 2025 (7)) by formulating the problem as a two-player game via a novel online algorithm, iterative Nash policy optimization (INPO). The key idea is to let the policy play against itself via no-regret learning, thereby approximating the Nash policy.

By contrast, the TOPR method we are exploring in this project as an extension introduces truncated importance sampling and asymmetric weighting, fully weighting positive rewarded trajectories while damping negative ones, and treats the baseline constant as a 'difficulty knob' to maximize data reuse.

3 Method

Supervised Fine-Tuning (SFT):
Supervised Fine-Tuning is the same next-token prediction objective that is used in pre-training. However, no loss is applied to the query tokens. This supervised learning objective is optimized over queries x and completions y that are drawn from an expert distribution. In the context of preference datasets, the completion y is typically the preferred completion y_w and is alternatively referred to as Pref-FT. The objective can be written as follows:

$$\max_{\theta} \mathbb{E}_{(x,y)\in D} \sum_{t=1}^{|y|} \log \pi_\theta\big(y_t \mid x,\, y_{<t}\big) \tag{1}$$

REINFORCE Leave One-Out (RLOO):
REINFORCE Leave One-Out (RLOO)(8) is a policy gradient estimator based on REINFORCE with a baseline to reduce variance of samples that is the weighted average of the reward of other samples from that policy. More formally the objective can be written as follows:

$$\frac{1}{k}\sum_{i=1}^{k}\Big[R\big(y_{(i)},x\big) \;-\; \frac{1}{k-1}\sum_{j\neq i} R\big(y_{(j)},x\big)\Big]\nabla_\theta \log \pi_\theta\big(y_{(i)} \mid x\big), \quad y_{(1)},\dots,y_{(k)} \overset{\text{i.i.d.}}{\sim} \pi_\theta(\cdot \mid x). \tag{2}$$

For preference datasets, we will need to learn a parametric reward function through the Bradley Terry objective (10). More formally, the objective can be written as follows:

$$\mathcal{L}_{\mathrm{BT}}(\phi) = \max_{\phi} \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_{\mathrm{pref}}} \big[\log \sigma\big(r_\phi(x,y_w) - r_\phi(x,y_l)\big)\big] \tag{3}$$

Preference Optimization (DPO):
In Rafailov et al. (9), DPO trains the model to directly maximize the likelihood of preferred outputs, treating the LLM itself as an implicit reward model. More formally, the DPO loss is:

$$\mathcal{L}_{\mathrm{DPO}}\big(\pi_\theta;\pi_{\mathrm{ref}}\big) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\Big(\beta\log\frac{\pi_\theta(y_w\mid x)}{\pi_{\mathrm{ref}}(y_w\mid x)} \;-\; \beta\log\frac{\pi_\theta(y_l\mid x)}{\pi_{\mathrm{ref}}(y_l\mid x)}\Big)\right] \tag{4}$$

where x is the prompt, y_w is the preferred responses, y_l is the dispreferred responses, π_θ is the policy that is being optimized and $\pi_r ef$ is the reference policy.

Tapered Off-Policy REINFORCE (TOPR):
As proposed in the Roux et al. 2025 (1), TOPR uses importance sampling to down-weight negative trajectories that are not likely under π, while allowing positive trajectories to be up-weighted irrespective of π as follows:

$$\nabla J_{\mathrm{TOPR}}(\pi) = \sum_{\tau:\,R(\tau)\geq 0} \mu(\tau)\,R(\tau)\,\nabla\log\pi(\tau) + \sum_{\tau:\,R(\tau)<0} \mu(\tau)\,\left[\tfrac{\pi(\tau)}{\mu(\tau)}\right]_0^1 R(\tau)\,\nabla\log\pi(\tau) \tag{5}$$

$$\big[x\big]_0^1 \equiv \min\big(\max(x,0),1\big) \tag{6}$$

where τ is a response (or trajectory) sampled from some data-generating policy μ, $R(\tau)$is the reward associated with this trajectory. Unlike other LLM algorithms in the REINFORCE family, TOPR does not require an explicit KL penalty to guarantee stable behaviour, making it both simpler to implement and computationally more efficient. TOPR itself corresponds to a range of truncation limits that combine the desirable properties of each of these methods into one learning rule.

Unlike naive off-policy REINFORCE (which uses full importance weights and suffers unbounded gradient variance) or uniform clipping schemes, TOPR's asymmetric tapering retains full learning signal from positive-reward trajectories while safely damping negative ones, and leverages the baseline as a difficulty knob—a combination not previously applied in LLM fine-tuning.

4 Experimental Setup

We begin by training the language model ($Qwen2.5-0.5B$) via standard next-token supervision on the Countdown dataset as a warm start.

SFT:
Data: We used the Cognitive Behaviors Dataset to increase the base model performance by biasing the on-policy rollouts of the model to desirable strategies such as backtracking and verification. The dataset contains 1,000 rows in train dataset and 200 rows in test dataset.

Experiment Hyperparameters: Training was conducted on all 1,000 rows in the train dataset with a batch size of 4, learning rate of 1e-6, a maximum sequence length of 512 tokens, and 3 epochs. The 512-token limit was informed by exploratory data analysis, revealing that the 95th percentile of total input-output pair lengths was 512 tokens, with 149 and 368 tokens for prompt and response lengths, respectively.

RLOO:
Data: We used the TinyZero to obtain a set of prompts for on-policy sampling during optimization. The dataset contains 490,000 rows in train dataset. We randomly sampled 1,000 rows for online RL using the same prompt format as the SFT dataset.

Experiment Hyperparameters: Training was conducted on 1,000 rows randomly sampled

Table 1: TOPR Experiments Setup

Experiment	Sample Size	Batch Size	K	Epoch
Experiment 1	500	2	2	2
Experiment 2	1000	4	4	1
Experiment 3	1300	2	2	1
Experiment 4	1500	1	1	1

from the TinyZero train dataset with a batch size of 4, learning rate of 1e-6, a maximum sequence length of 256 tokens, and 1 epochs. We reduced the token length to 256 due to computing cost. The epoch was reduced to one as we see less loss descreased in SFT from using more epochs.

DPO: Data: We also used the TinyZero to obtain a set of prompts for on-policy sampling during optimization. The dataset contains 490,000 rows in train dataset. We randomly sampled 2,000 rows to construct an on-policy preference dataset by sampling 2 examples from the SFT model with a non-zero temperature and labeling a preferred and dispreferred response using the rule-based reward, filtering out for ties.

Experiment Hyperparameters: Training was conducted on 2,000 rows randomly sampled from the TinyZero train dataset with a batch size of 4, learning rate of 1e-6, 0.8 temperature, 0.1 beta a maximum sequence length of 256 tokens, and 1 epoch. We reduced the token length to 256 due to computing cost. The epoch was reduced to one as we see less loss descreased in SFT from using more epochs.

TOPR:
Data: We also used the TinyZero to obtain a set of prompts for on-policy sampling during optimization. The dataset contains 490,000 rows in train dataset. We randomly sampled 1,000 rows for online RL using the same prompt format as the SFT dataset.

Experiment Hyperparameters: We performed multiple trainings on 500 to 1500 rows randomly sampled from the TinyZero train dataset with different batch sizes ranging from 1 to 4, learning rate of 1e-6, a maximum sequence length of 256 tokens, and 1 to 2 epochs. We reduced the token length to 256 due to computing cost. The epoch was reduced to less than three as we see less loss decreased in SFT from training 3 epochs as opposed to 2 epochs. We also used greedy decoding sampling for stability therefore there is less variation in changing the batch size. Specifically we experimented following hyperparameters on the same TOPR algorithm:

5 Results

5.1 Quantitative Evaluation

The initial results from SFT and RLOO methods are analyzed in two key areas: Training losses and log probability / Perplexity. For TOPR, we compared the training losses, valid equation rate and evaluation scores from different experiments outlined in Table 1 and Table 2.

SFT:
Training Losses: We observed a consistent decrease in average training loss (computed as total loss divided by the number of valid batches) across epochs: from 0.4966 in Epoch 1, to 0.3621 in Epoch 2, and 0.3222 in Epoch 3. This trend suggests that the fine-tuning process is effective. Notably, the steepest decline occurs between the first two epochs, indicating that future training runs might require only two epochs to achieve comparable results.

Log Probability / Perplexity: We evaluated token-level log probabilities and perplexities across both training and validation sets, based on the predefined train/test split in the Cognitive Behaviors Dataset (see Figure 1). The distributions of log probabilities are centered around –2.0 to

–2.1, corresponding to perplexities of approximately 7.8 to 8.5, both values typical for medium-sized LLMs performing next-token prediction on structured prompts. The training and validation curves largely overlap, with only minor deviations, indicating minimal risk of overfitting. This is further supported by similar average log probabilities in both datasets (see Table 1). Interestingly, the validation set shows a slightly higher average log probability (+0.0003) and a subtle shift toward lower perplexities, possibly due to shorter or more formulaic examples in the validation data.

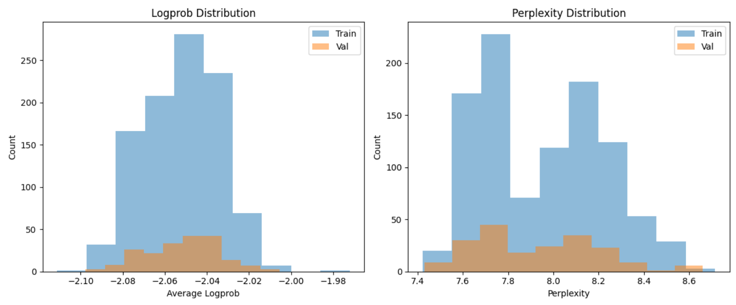


Figure 1: Distribution of Logprob and Perplexity on Train/Val Data using SFT Model

RLOO:
Training Losses: We observed BT loss quite fluctuating at 0.2 to 0.6 as the early guesses are quite random but later plateaus around 0.2 to 0.4 once the reward model has roughly learned easy distinctions but still struggles on hard pairs. The policy-gradient loss started large at 5 to 10 then dropped below 1.

Log Probability / Perplexity: Similarly to the SFT method, we did not observe massive overfitting concerns in the RLOO method.

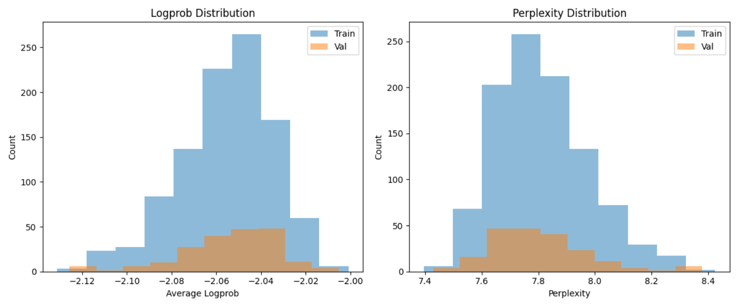


Figure 2: Distribution of Logprob and Perplexity on Train/Val Data using RLOO Model

TOPR:
Training Loss: In our Tapered Off-Policy REINFORCE (TOPR) setup, we turn the usual REINFORCE gradient into a mini batch Policy Gradient (PG) loss by simply taking the negative of the weighted reward-times-log-prob term. Specifically, Given a prompt x and a sampled continuation $y = (y_1, \ldots, y_T)$, we define:

$$ {}_{\theta}(x, y) \;=\; \sum_{t=1}^{T} \log \pi_\theta\big(y_t \mid x,\, y_{<t}\big), \tag{7}$$

$$ r(x, y) \;=\; r_f(x, y) \;+\; r_v(x, y), \quad r_f, r_v \in \{-1, 1\}, \tag{8}$$

$$ w_\tau(x, y) \;=\; \min\Big\{\tau,\; \frac{\pi_\theta(y \mid x)}{\beta(y \mid x)}\Big\}, \tag{9}$$

Table 2: TOPR Method Experiments Performance Comparison

Experiment	PG Loss	Valid Equation Rate
Experiment 1	-34.5635	0.0740
Experiment 2	-100.9669	0.0830
Experiment 3	-49.8187	0.0915
Experiment 4	-65.6271	0.0893

where β is the behavior policy and τ decays toward 1 over training.

We then define our PG loss as the negative expected weighted reward-log-prob:

$$\mathcal{L}_{\text{PG}}(\theta) = -\,\mathbb{E}_{x,y\sim\beta}\big[w_\tau(x,y)\,r(x,y)\,s_\theta(x,y)\big] \approx -\frac{1}{N}\sum_{i=1}^{N} w_\tau(x_i,y_i)\,r(x_i,y_i)\,\sum_{t=1}^{T}\log\pi_\theta\big(y_{i,t}\mid x_i,y_{i,<t}\big) \tag{10}$$

Training Losses: Overall, we observed batch-average PG losses less fluctuated than the RLOO method in all experiments: while the batch-average PG losses fluctuated in early stage when there are just few samples used in the training process, they started to steadily increase with more training samples (See details in Appendix). Such upward trend indicates that the model is learning as it begins to put more probability mass on sequences that actually solve the puzzle, reward goes from -1 to 1 more often, and typically tapers toward 0.

When comparing across those four experiments (See Table 2), we noted that the experiment 1 which was trained with less samples but more epochs actually outperforms other three experiments which were trained with more samples but less epochs. This is because even though the second curve plots 1,000 gradient steps, it has only seen each example once, whereas the first curve saw them twice. More passes gives the model more opportunity to push reward-producing trajectories up. To verify this, we ran experiment 1 twice and saw similar PG loss trend (See details in Appendix).

Additionally, a larger batch size and larger k indicates averaging over more trajectories/blanks, which dampens each individual reward signal and shrinks the policy gradient magnitudes. That's why we see experiment 2 with the largest batch size and k has the lowest PG loss after 1,000 samples. Interestingly, as we increased the sample size to 1,500 and reduce the batch size and k to 1 in experiment 4, we did not obtain the best result in terms of PG loss. This is because with k=1 there is no "other" roll-out, so this experiment either defaults the baseline to zero or crashes (division by zero). Either way we lost the single biggest variance-reduction benefit we had in the k=2 or k=4 runs (experiments 1 to 3). Also because that we used greedy sampling so the model always takes its highest-probability next token. That means it almost never "stumbles" upon new, better solutions, so out reward stays stuck (and negative) for far longer, further slowing the upward drift of that loss curve (Experiment 4).

5.2 Qualitative Analysis

In our empirical comparison of four TOPR experiments, we find that the interplay between the off-policy baseline size k, batch size, and number of data passes critically governs both the signal-to-noise ratio of the policy-gradient updates and their effective magnitude. A modest baseline (k=2) combined with small batch size (2 samples) and two training epochs produces the most rapid and stable reduction in PG loss: the off-policy correction dampens variance enough to avoid extreme updates, but still allows each positive-reward rollout to contribute a substantial gradient "kick," driving the loss from around –85 to around –35 within a few hundred steps.

By contrast, increasing k and the batch size to 4 under a single epoch reduces per-step variance further but also shrinks the expected gradient, yielding only a gradual loss climb. At the other extreme, eliminating the baseline (k=1) and using singleton batches generates highly stochastic updates that are dominated by noise: the loss curve exhibits steep, erratic drops for incorrect rollouts and slow recoveries for occasional correct ones, resulting in very slow net progress. These results underscore that a balanced choice of baseline and batch dimensions—together with multiple data passes—is essential for efficient, low-variance policy gradient learning in arithmetic reasoning tasks.

6 Discussion

In this work, we have systematically compared four fine-tuning strategies—Supervised Fine-Tuning (SFT), Reinforcement Learning from Learned Off-Policy Optimizer (RLOO), Direct Preference Optimization (DPO), and our proposed Tapered Off-Policy REINFORCE (TOPR)—on the Countdown arithmetic reasoning task. As expected, SFT provides a stable foundation: by directly minimizing token cross-entropy on gold reasoning traces, it reliably learns to imitate step-by-step solutions, but exhibits limited capacity to correct its own mistakes during generation. DPO, which treats pairwise preference comparisons as an implicit reward, succeeds in biasing the model toward preferred reasoning styles, yet its reliance on a fixed reference policy can lead to over-conservatism when the reference itself underperforms. RLOO further incorporates an explicit reward model into an off-policy policy-gradient framework: by subtracting the average reward of other samples as a baseline, it reduces variance and modestly improves final-expression accuracy, but its effectiveness is highly sensitive to the choice of baseline size k and mini-batch configuration.

Our TOPR method builds on RLOO by introducing a “tapered” importance-weight clipping schedule. Early in training, a high clipping threshold preserves strong gradient updates from rare correct rollouts; as training proceeds, the threshold decays toward unity, yielding unbiased on-policy gradients. Empirically, we observe that a moderate baseline (k=2) with small mini-batches (2 examples) and at least two full epochs produces the fastest reduction in policy-gradient loss within a few hundred updates—while maintaining low variance. In contrast, excessively large baselines or batch sizes (e.g.k=4, batch size=4) over-dampen the gradient, slowing convergence, and eliminating the baseline (k=1, batch size=1) re-introduces high stochasticity that requires many more steps to overcome.

7 Conclusion

In summary, our comparative study demonstrates that while Supervised Fine-Tuning (SFT) provides a reliable baseline for learning to imitate gold reasoning traces, the Tapered Off-Policy REINFORCE (TOPR) method effectively helps balance exploration and stability on the Countdown task. By dynamically clipping importance weights, TOPR allows strong gradient updates from initial rare successful rollouts and then gradually annealing toward unbiased on-policy gradients. It achieves rapid and low-variance convergence when paired with a modest baseline size (k=2), small mini-batches, and multiple epochs. These results underscore the necessity of tailored variance-reduction strategies in reinforcement-based fine-tuning for sparse, binary rewards, and suggest that integrating supervised imitation with carefully calibrated off-policy corrections may be endowing large language models with more robust, self-correcting mathematical reasoning capabilities. Future work should explore adaptive clipping schedules, richer multi-step reward definitions (e.g. partial credit for correct sub-expressions), and extensions to more complex reasoning benchmarks. Ultimately, combining supervised imitation with well-calibrated off-policy reinforcement offers a promising avenue to endow large language models with robust, self-correcting mathematical reasoning skills.

8 Team Contributions

- **Group Member 1:** Mengge Pu is the only contributor to this paper

Changes from Proposal Focus on Countdown task only

References

[1] Nicolas Le Roux, Marc G. Bellemare, Jonathan Lebensold, Arnaud Bergeron, Joshua Greaves, Alex Fréchette, Carolyne Pelletier, Eric Thibodeau-Laufer, Sándor Toth, and Sam Work

(2025). Tapered off-policy reinforce: Stable and efficient reinforcement learning for LLMs. *arXiv:2503.14286*

[2] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, Chelsea Finn (2024). Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *arXiv:2305.18290*

[3] Rafael Rafailov, Joey Hejna, Ryan Park, Chelsea Finn (2024). From r to Q*: Your Language Model is Secretly a Q-Function. *arXiv:2404.12358*

[4] Yunhao Tang, Taco Cohen, David W. Zhang, Michal Valko, Rémi Munos (2025). RL-finetuning LLMs from on- and off-policy data with a single algorithm. *arXiv:2503.19612*

[5] Brian R. Bartoldson, Siddarth Venkatraman, James Diffenderfer, Moksh Jain, Tal Ben-Nun, Seanie Lee, Minsu Kim, Johan Obando-Ceron, Yoshua Bengio, Bhavya Kailkhura (2025). Trajectory Balance with Asynchrony: Decoupling Exploration and Learning for Fast, Scalable LLM Post-Training. *arXiv:2503.18929*

[6] Wenxuan Zhou, Ravi Agrawal, Shujian Zhang, Sathish Reddy Indurthi, Sanqiang Zhao, Kaiqiang Song, Silei Xu, Chenguang Zhu (2024). WPO: Enhancing RLHF with Weighted Preference Optimization. *arXiv:2406.11827*

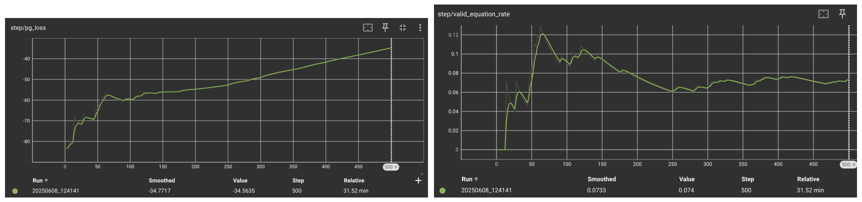
[7] Yuheng Zhang, Dian Yu, Baolin Peng, Linfeng Song, Ye Tian, Mingyue Huo, Nan Jiang, Haitao Mi, Dong Yu (2025). Iterative Nash Policy Optimization: Aligning LLMs with General Preferences via No-Regret Learning. *arXiv:2407.00617*

[8] 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, 2024. *arXiv:2402.14740*

[9] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. arXiv preprint, 2023. *arXiv:2305.18290*

[10] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika, 39(3/4):324–345, 1952. ISSN 00063444*

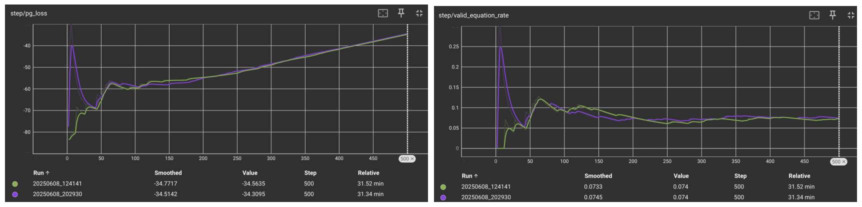
A Experiments Charts



Run ↑	Smoothed	Value	Step	Relative
20250608_124141	-34.7717	-34.5635	500	31.52 min

Run ↑	Smoothed	Value	Step	Relative
20250608_124141	0.0733	0.074	500	31.52 min

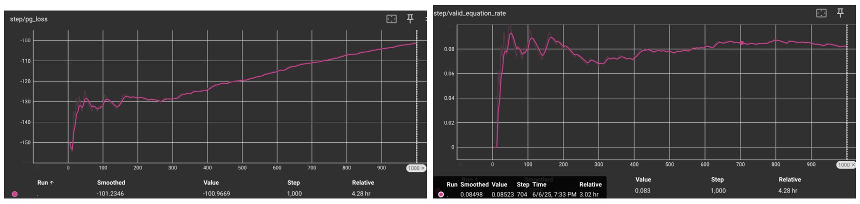
Figure 3: TOPR Experiment 1 (1st run): PG loss and valid equation rate by steps



Run ↑	Smoothed	Value	Step	Relative
20250608_124141	-34.7717	-34.5635	500	31.52 min
20250608_202930	-34.5142	-34.3095	500	31.34 min

Run ↑	Smoothed	Value	Step	Relative
20250608_124141	0.0733	0.074	500	31.52 min
20250608_202930	0.0745	0.074	500	31.34 min

Figure 4: TOPR Experiment 1 (2nd run): PG loss and valid equation rate by steps

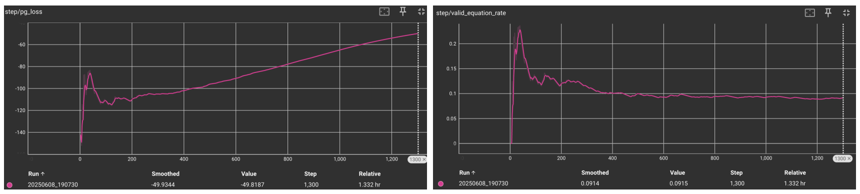


Run ↑	Smoothed	Value	Step	Relative
.	-101.2346	-100.9669	1,000	4.28 hr

Run	Smoothed	Value	Step	Time	Relative
.	0.08498	0.08523	704	6/6/25, 7:33 PM	3.02 hr

Value	Step	Relative
0.083	1,000	4.28 hr

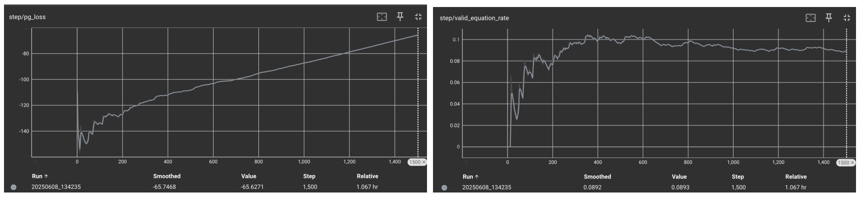
Figure 5: TOPR Experiment 2: PG loss and valid equation rate by steps



Run ↑	Smoothed	Value	Step	Relative
20250608_190730	-49.9344	-49.8187	1,300	1.332 hr

Run ↑	Smoothed	Value	Step	Relative
20250608_190730	0.0914	0.0915	1,300	1.332 hr

Figure 6: TOPR Experiment 3: PG loss and valid equation rate by steps



Run ↑	Smoothed	Value	Step	Relative
20250608_134235	-65.7468	-65.6271	1,500	1.067 hr

Run ↑	Smoothed	Value	Step	Relative
20250608_134235	0.0892	0.0893	1,500	1.067 hr

Figure 7: TOPR Experiment 4: PG loss and valid equation rate by steps