ENTROPY-PRESERVING REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy gradient algorithms have been a driver of much recent advancement in language model reasoning. One of their most appealing properties is the ability to learn from exploration on their own trajectories, a process crucial for discovering diverse approaches and fostering creative solutions. As we show in this paper, most policy gradient algorithms naturally reduce the entropy—and thus the diversity of explored trajectories—as part of training, yielding a policy increasingly limited in its ability to explore. However, not all algorithms exhibit this collapse in entropy equally. In this paper, we formally analyze the contributions of leading policy gradient objectives on entropy, show which mechanisms they employ to implicitly limit entropy collapse, and propose a new regularization method, REPO, that stabilizes entropy over training through the use of an adaptive controller. Models trained with REPO preserve entropy throughout training, yielding final policies that are, on average, more performant. By preserving entropy in the final policy, REPO-trained models can even be re-trained on evolving data distributions in new environments, unlike their non-entropy-preserving counterparts.

1 INTRODUCTION

Online policy gradient reinforcement learning (RL) has become the standard for boosting the reasoning abilities of language models (Jaech et al., 2024; Comanici et al., 2025; Guo et al., 2025). This approach involves sampling trajectories from the current policy within a given environment and reward function, then using these to estimate a gradient that maximizes expected reward. Effective RL optimization requires balancing exploration and exploitation (Thrun, 1992; Sutton et al., 1998), where a robust learner should generate diverse trajectories to cover the spectrum of potential solutions. Maximum entropy reinforcement learning offers a framework for achieving this balance (Ziebart et al., 2008; Haarnoja et al., 2017; 2018; Eysenbach & Levine, 2022). While trivially the optimal solution to a finite Markov decision process (MDP) is a deterministic stationary policy, optimization over the intermediate landscape requires a balance of exploration and exploitation.

A common issue observed in online algorithms like GRPO (Shao et al., 2024) is entropy collapse. This phenomenon occurs when training excessively narrows the distribution around already high-probability solutions from the base model, neglecting other correct but less probable options. This often yields premature convergence to a local optimum, enhancing `pass@1` relative to base model at the expense of `pass@k` (Shao et al., 2024; Dang et al., 2025; Yue et al., 2025). This challenge has spurred innovations in policy gradient algorithm design, e.g. directly optimizing for `pass@k` performance (Chen et al., 2025b). Concurrently, research has highlighted GRPO's training instability and the complex interplay between off-policy drift, importance weight clipping, and entropy, inspiring modifications such as DAPO (Yu et al., 2025) and GSPO (Zheng et al., 2025).

In this work, we analyze entropy preservation as a unifying lens for understanding the successes of recent algorithms and to propose a novel family of policy gradient objectives. An important observation from our work is that, while a correlation exists between final entropy and performance, a more informative measure is the entropy trajectory throughout the optimization process. As the saying goes, "it's not the destination, it's the journey." Figure 1 tracks this effect. A trajectory characterized by lower entropy throughout yields lower performance. Conversely, if entropy trajectories are similar for most of the optimization but differ only in the final steps, performance is largely unaffected.

Given this observation, we turn to study the entropy behavior of various leading RL algorithms. We begin by theoretically analyzing how the REINFORCE policy gradient objective modulates entropy

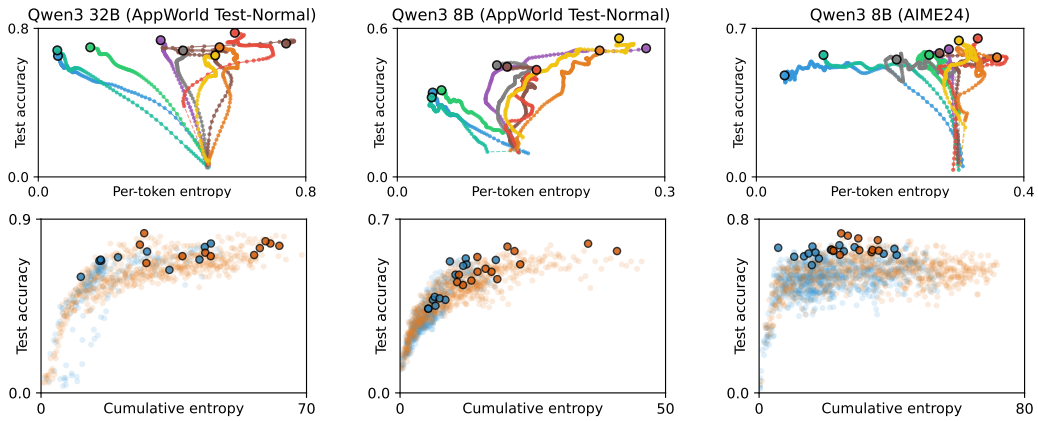

Figure 1: Top: Evolution of the average per-token entropy and test accuracy during training for several baseline (GRPO, LOOP, DAPO, GSPO) and their entropy regularized versions (REPO). Each curve shows the average trajectory over several training runs with different seeds. Bottom: Cumulative entropy experienced during training up to a given checkpoint is positively correlated with the test accuracy. Each point is a checkpoint of a single training run (best-performing checkpoint per run highlighted). Algorithms that collapse the entropy early (see `Qwen-3-8B` on AppWorld; middle column) perform significantly worse than algorithms that maintain a steady entropy during training. See App. E for a detailed study and breakdown of this phenomena across algorithms.

dynamics, explaining how entropy can decrease during training. This effect is amplified when talking multiple policy update steps, typical of GRPO and other PPO-like algorithms. We also show how importance weight clipping and its modifications as seen in DAPO and GSPO, can mitigate this pressure. Finally, we show that regularizing the entropy during training allows a broad family of policy gradient algorithms to train more performant policies. Specifically, we introduce Regulated Entropy Policy Optimization (REPO), an approach to policy gradient optimization that adaptively reweighs advantages and log-probabilities online to preserve entropy. REPO uses an adaptive controller, tracking entropy dynamics live, and adjusting regularization strength accordingly. Training with REPO achieves state-of-the-art results on AppWorld and strong performance on AIME 2024 and AIME 2025. Furthermore, we demonstrate that policies trained with REPO retain their trainability, allowing for iterative learning on new tasks in novel environments, a capability often lost in policies trained without explicit entropy preservation.

2 Preliminaries

Language modeling. Let $x \in \mathcal{X}$ denote the tokens in a vocabulary and $\boldsymbol{x} \in \mathcal{X}^*$ the strings expressible via concatenation of those tokens. A **language model** (LM) π_θ parameterized by θ defines a probability distribution over strings that factors autoregressively such that $\pi_\theta(\boldsymbol{x}) = \pi_\theta(\square \mid \boldsymbol{x}) \prod_{i=1}^{|\boldsymbol{x}|} \pi_\theta(x_i \mid \boldsymbol{x}_{<i})$, where $\square$ denotes an end of sequence (EOS) marker. Note that for notational convenience we will use π_θ to express probabilities on both tokens and strings.

Language modeling as a Markov decision process. Let the policy π_θ sample **actions** $a \in \mathcal{A} = \mathcal{X} \cup \{\square\}$ (any token or EOS) given a **state** $\boldsymbol{s} \in \mathcal{X}^*$ (a string context). Let state transitions append generated actions to the state.[1] Let $\boldsymbol{\tau}$ denote a **trajectory**, a sequence of states and actions generated by the policy and environment. Let $\boldsymbol{\tau} \sim \pi_\theta$ denote the trajectory distribution. We consider tasks with **terminal rewards** $R(\boldsymbol{c}, \boldsymbol{\tau})$. Given some task context $\boldsymbol{c}$ sampled from some dataset $\mathcal{D}$, the MDP objective is to maximize $\mathcal{J}_{\text{MDP}} \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{c} \sim \mathcal{D}, \boldsymbol{\tau} \sim \pi_\theta(\cdot|\boldsymbol{c})}[R(\boldsymbol{c}, \boldsymbol{\tau})]$.

[1] State transitions deterministically append the generated action to the context, terminating generation at EOS or upon some other environment condition. In some domains, e.g., those involving tool calls, state transitions may also append additional tokens to the state that were generated by some unobservable process such as executing a code interpreter.

Policy gradient reinforcement learning directly computes a gradient through the REINFORCE algorithm (Williams, 1992), which is amenable to Monte Carlo estimation:

$$\nabla_\theta \mathcal{J}_{\text{MDP}} = \mathbb{E}_{\boldsymbol{c}\sim\mathcal{D},\boldsymbol{\tau}\sim\pi_\theta(\cdot|\boldsymbol{c})}\left[A(\boldsymbol{c},\boldsymbol{\tau})\cdot\nabla_\theta \log \pi_\theta(\boldsymbol{\tau}\mid\boldsymbol{c})\right],$$

where $A(\boldsymbol{c},\boldsymbol{\tau}) = R(\boldsymbol{c},\boldsymbol{\tau}) - b$ is an advantage function shifting the return $R(\boldsymbol{c},\boldsymbol{\tau})$ by a baseline b.

REINFORCE leave-one-out (RLOO) (Kool et al., 2019; Ahmadian et al., 2024; Kazemnejad et al., 2024; Chen et al., 2025a) is one of the most popular estimates of advantage for language modeling. It generates K independent samples on-policy $\boldsymbol{\tau}_1, ..., \boldsymbol{\tau}_k \sim \pi_\theta(\cdot \mid \boldsymbol{c})$ for each task $\boldsymbol{c}$. The reward for each trajectory may then be baselined against the remaining $K-1$ independent samples, yielding an unbiased, low variance advantage estimator:

$$\widehat{A}_{\text{RLOO}}(\boldsymbol{c},\boldsymbol{\tau}_i) \stackrel{\text{def}}{=} R(\boldsymbol{c},\boldsymbol{\tau}_i) - \frac{1}{K-1}\sum_{j=1}^{K} R(\boldsymbol{c},\boldsymbol{\tau}_j)\mathbb{1}_{[i\neq j]} = \frac{K}{K-1}\left(R(\boldsymbol{c},\boldsymbol{\tau}_i) - \frac{1}{K}\sum_{j=1}^{K} R(\boldsymbol{c},\boldsymbol{\tau}_j)\right).$$

Policy gradient algorithms are on-policy by nature: They rely on a new set of trajectories in each context $\boldsymbol{\tau}\sim\pi_\theta(\cdot\mid\boldsymbol{c})$ after each gradient update of the policy π_θ.

Proximal policy optimization (PPO) allows the updated policy to deviate slightly from a sampling policy (Schulman et al., 2017). It uses an importance weight to correct the magnitudes of parameter updates such that the expected policy gradient remains unbiased. These importance weights are typically clipped to avoid divergence from a local trust region (Schulman et al., 2015).

$$\mathcal{J}_{\text{PPO}} \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{c}\sim\mathcal{D},\boldsymbol{\tau}\sim\pi_\theta(\cdot|\boldsymbol{c})}\left[\frac{1}{|\boldsymbol{\tau}|}\sum_{a_t\in\boldsymbol{\tau}} \min\left(A(\boldsymbol{c},\boldsymbol{\tau})\cdot w_t, A(\boldsymbol{c},\boldsymbol{\tau})\cdot w_t|_{1-\epsilon}^{1+\epsilon}\right)\right] \quad w_t \stackrel{\text{def}}{=} \frac{{\pi_\theta}^{\text{new}}(a_t\mid\boldsymbol{c},\boldsymbol{a}_{<t})}{{\pi_\theta}^{\text{old}}(a_t\mid\boldsymbol{c},\boldsymbol{a}_{<t})},$$

where $w_t|_{1-\epsilon}^{1+\epsilon}$ clips the importance ratio from below $1-\epsilon$ and above $1+\epsilon$. In our theoretical analysis, we will examine PPO with and without clipping. The version studied will be clear from the context.

LOOP (Chen et al., 2025a) and **GRPO** (Shao et al., 2024) combine the above PPO objective with RLOO leave-one-out advantage estimates. GRPO rescales advantages by the standard deviation of the sample returns, which introduces a small bias (Liu et al., 2025b),

$$\widehat{A}_{\text{GRPO}}(\boldsymbol{c},\boldsymbol{\tau}_i) \stackrel{\text{def}}{=} \frac{R(\boldsymbol{c},\boldsymbol{\tau}_i) - \text{mean}(R(\boldsymbol{c},\boldsymbol{\tau}_1)),\dots,R(\boldsymbol{c},\boldsymbol{\tau}_k))}{\text{std}(R(\boldsymbol{c},\boldsymbol{\tau}_1)),\dots,R(\boldsymbol{c},\boldsymbol{\tau}_k))}$$

while LOOP uses $\widehat{A}_{\text{RLOO}}$ directly.

Group Sequence Policy Optimization (GSPO) Zheng et al. (2025) uses a trajectory-level trust region defined by the geometric average of a sequence's probability ratios

$$\mathcal{J}_{\text{GSPO}} \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{c}\sim\mathcal{D},\boldsymbol{\tau}\sim\pi_\theta(\cdot|\boldsymbol{c})}\left[\min\left(A(\boldsymbol{c},\boldsymbol{\tau})\cdot w^{\text{GSPO}}, A(\boldsymbol{c},\boldsymbol{\tau})\cdot w^{\text{GSPO}}|_{1-\epsilon}^{1+\epsilon}\right)\right] \quad w^{\text{GSPO}} \stackrel{\text{def}}{=} \left(\frac{{\pi_\theta}^{\text{new}}(\boldsymbol{\tau}\mid\boldsymbol{c})}{{\pi_\theta}^{\text{old}}(\boldsymbol{\tau}\mid\boldsymbol{c})}\right)^{\frac{1}{|\boldsymbol{\tau}|}}.$$

GSPO yields an equivalent gradient estimator to GRPO, LOOP, and RLOO on-policy, but clips tokens and trajectories differently as the updated policy ${\pi_\theta}^{\text{new}}$ drifts from the sampling policy ${\pi_\theta}^{\text{old}}$.

Policy entropy. The inherent uncertainty that a policy places over its generations may be expressed from an information theoretic standpoint as **entropy** – expected surprise: $\mathcal{H}_{\pi_\theta}(\mathcal{D}) = -\mathbb{E}_{\boldsymbol{c}\sim\mathcal{D}}\left[\mathbb{E}_{\boldsymbol{\tau}\sim\pi_\theta(\cdot|\boldsymbol{c})}\left[\log\pi_\theta(\boldsymbol{\tau}\mid\boldsymbol{c})\right]\right]$. In addition to global entropy, we may consider the entropy over actions at any given state $\boldsymbol{s} = (\boldsymbol{c},\boldsymbol{a}_{<t})$ as $\mathcal{H}_{\pi_\theta}(\cdot\mid\boldsymbol{s}) = -\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[\log\pi_\theta(a\mid\boldsymbol{s})\right]$.

In this paper, we show how state-wise entropy changes as variants of policy gradient optimize their objectives. We show which variants are naturally entropy preserving, and which variants lead to a rapid collapse. Finally, we show that a simple class of transformations applied to the advantages lead to a very simple and effective regularization of entropy.

3 The Entropy Dynamics of Policy Gradient

The entropy dynamics of policy gradient RL boils down to the relationship between two values: (1) action log-probabilities, and (2) the advantages yielded by those actions. Intuitively, assigning

a positive advantage to some action increases its probability. For high probability actions, this effect sharpens the distribution, and entropy decreases. For low probability actions, this flattens the distribution, increasing entropy. The opposite pattern holds for negative advantages. This effect is quite natural. After all, sharpening an uncertain policy around correct actions directly maximizes the expected return. However, as we will see, not all RL algorithms sharpen the distribution equally.

Formally, consider the policy gradient update with on-policy actions in state $\boldsymbol{s}$. Under a first-order Taylor approximation to the training dynamics, the expected change in entropy is as follows.

Theorem 1. *Given a policy gradient update $\widehat{\theta} := \theta + \alpha \cdot \nabla_\theta \mathcal{J}_{\mathrm{MDP}}(\boldsymbol{s})$, the expected change in entropy is approximately:*

$$\Delta \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) \approx -\alpha \cdot \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s}),a'\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[A(\boldsymbol{s},a)\cdot L(\boldsymbol{s},a')\cdot u(\boldsymbol{s},a)^\top u(\boldsymbol{s},a')\right],$$

where $L(\boldsymbol{s},a) \stackrel{\text{def}}{=} \log \pi_\theta(a \mid \boldsymbol{s}) - \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}[\log \pi_\theta(a \mid \boldsymbol{s})]$ denotes mean-centered log-probabilities and $u(\boldsymbol{s},a) \stackrel{\text{def}}{=} \nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s})$ denotes the score function for some policy π_θ evaluated at state $\boldsymbol{s}$ and action a.

[Proof in App. C.2]. The entropy change is driven by a multiplicative relationship between action log-probabilities and the advantages yielded by those actions. In an exact derivation, these are weighted by the score vector outer product. With additional independence assumptions or a tabular softmax policy parameterization, this expression can be further simplified, resulting in a weighting by the action probabilities. This yields the following corollary:

Corollary 1. *Assuming $u(\boldsymbol{s},a)^\top u(\boldsymbol{s},a') = 0$ for all $a \neq a'$, the change in entropy is approximately:*

$$\Delta \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) \propto -\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[A(\boldsymbol{s},a)\cdot L(\boldsymbol{s},a)\cdot \pi_\theta(a \mid \boldsymbol{s})\right]$$

[Proof in App. C.3]. This latter form encodes the dominant behavior of entropy dynamics in a manner that is inherent to policy gradient. Using this form, we explain the observed behaviors of various RL algorithms. A similar derivation can be shown for tabular softmax policies (Cui et al., 2025, see Corollary 2 in App. C.4). Thm. 1 and Corollary 1 tell us that the change in entropy is governed by a correlation between advantages and log-probabilities, weighted by action probability.

Entropy dynamics of PPO. The biggest feature of PPO is its ability to train on slightly off-policy trajectories, given that the updated policy does not deviate from a trust region around the current policy. This allows PPO to take multiple policy-improvement steps for a single set of trajectories. The effect of these repeated updates are much larger policy updates between consecutive PPO steps, which empirically amplifies entropy collapse. This being said, the clipping on PPO, when appropriately orchestrated, can protect against entropy collapse as well. Clipping ensures that no policy gradient update is performed if the policy drifts outside a trust region $(1-\epsilon_{\text{low}})\cdot\pi_\theta^{\text{old}}(a \mid \boldsymbol{s}) \leq \pi_\theta^{new}(a \mid \boldsymbol{s}) \leq (1+\epsilon_{\text{high}})\cdot\pi_\theta^{\text{old}}(a \mid \boldsymbol{s})$. This bounds the change in entropy:

Theorem 2. *Proximal Policy Optimization (PPO) bounds the entropy $\mathcal{H}_{\pi_{\theta^{new}}}(\cdot \mid \boldsymbol{s})$ of the updated policy by the original policy entropy $\mathcal{H}_{\pi_{\theta^{old}}}(\cdot \mid \boldsymbol{s})$ such that:*

$$(1-\epsilon_{low})\cdot\mathcal{H}_{\pi_{\theta^{old}}}(\cdot \mid \boldsymbol{s}) \leq \mathcal{H}_{\pi_{\theta^{new}}}(\cdot \mid \boldsymbol{s}) \leq (1+\epsilon_{high})\cdot\mathcal{H}_{\pi_{\theta^{old}}}(\cdot \mid \boldsymbol{s})$$

[Proof in App. C.5]. The clipping thresholds directly limit the maximum induced change in entropy per token. Intuitively, the change in entropy per token is stochastic: some actions have a large correlation between advantage and log probability; others do not, or even have an anti-correlation. For a symmetric clipping regime, this results in an entropy change that largely follows the statistical trends outlined above, but at a lower magnitude.

Entropy dynamics of DAPO. Now consider DAPO (Yu et al., 2025), with an asymmetric clipping regime $\epsilon_{\text{low}} < \epsilon_{\text{high}}$. This allows for larger entropy increases, while limiting the entropy decrease. Due to the stochastic nature of the entropy changes, this directly contributes to an overall increase in per-token entropy over sufficient samples. Threshold values $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$ proposed in Yu et al. (2025) stabilize the entropy throughout training, as we show experimentally.

Entropy dynamics of GSPO. GSPO defines a trust region $1 - \epsilon_{\text{low}}^{\text{GSPO}} \leq w^{\text{GSPO}} \leq 1 + \epsilon_{\text{high}}^{\text{GSPO}}$, or equivalently $\left(1 - \epsilon_{\text{low}}^{\text{GSPO}}\right)^{|\boldsymbol{\tau}|} \leq \frac{\pi_\theta{}^{\text{new}}(\boldsymbol{\tau}|\boldsymbol{c})}{\pi_\theta{}^{\text{old}}(\boldsymbol{\tau}|\boldsymbol{c})} \leq \left(1 + \epsilon_{\text{high}}^{\text{GSPO}}\right)^{|\boldsymbol{\tau}|}$. This induces an equivalent bound to Thm. 2; however, the bound now depends on the trajectory length $|\boldsymbol{\tau}|$. Longer trajectories may induce a larger change in entropy, shorter trajectories induce a smaller change in entropy. With parameter values suggested in Zheng et al. (2025), $\epsilon_{\text{low}}^{\text{GSPO}} = 3 \times 10^{-4}$ and $\epsilon_{\text{high}}^{\text{GSPO}} = 4 \times 10^{-4}$, the entropy bound is tighter for trajectories $|\boldsymbol{\tau}| < \frac{\ln(1\pm\epsilon)}{\ln(1\pm\epsilon^{\text{GSPO}})} \approx 600$ tokens compared to DAPO. Like DAPO, the clipping range is asymmetric $\epsilon_{\text{low}}^{\text{GSPO}} < \epsilon_{\text{high}}^{\text{GSPO}}$ leading to a stochastic increase in entropy.

Regulated entropy policy optimization (REPO) changes the advantage function to include a scaled policy log-likelihood term $A_{\text{REPO}}(\boldsymbol{s}, a) = A(\boldsymbol{c}, a) - \beta_{\boldsymbol{s}} \cdot L(\boldsymbol{s}, a)$ for each $\boldsymbol{s} = (\boldsymbol{c}, \boldsymbol{a}_{<t})$. This updated advantage is no longer constant throughout the trajectory like in RLOO and variants, but differs for individual tokens $a_t \in \boldsymbol{\tau}$. Following Thm. 1 by Prop. 3 the induced change in entropy with A_{REPO} is:

$$\Delta\mathcal{H}_{\pi_\theta}^{\text{REPO}}(\cdot \mid \boldsymbol{s}) \approx \Delta\mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) + \beta_{\boldsymbol{c}} \cdot \underbrace{\alpha \cdot \left\| \mathbb{E}_{a \sim \pi_\theta(\cdot|\boldsymbol{s})} \left[L(\boldsymbol{s}, a) \cdot u(\boldsymbol{s}, a) \right] \right\|^2}_{\geq 0}.$$

This provides us with a direct mechanism to control the entropy. A positive $\beta_{\boldsymbol{c}} > 0$ increases the entropy over actions in a state relative to the default dynamic. A value $\beta_{\boldsymbol{c}} = 0$ allows the natural entropy decrease to proceed. A negative $\beta_{\boldsymbol{c}} < 0$ collapses the entropy. Note, this holds for any parametrization of the policy and does not rely on approximations.

How should we choose $\beta_{\boldsymbol{c}}$ to preserve entropy? One natural choice is to counter-act the entropy collapse on a per-token level and set $\beta_{\boldsymbol{c}}^{\text{REPO-D}} \propto -\Delta\mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s})$ as approximated in Corollary 1, thus neutralizing $\Delta\mathcal{H}_{\pi_\theta}$ and allowing $\Delta\mathcal{H}_{\pi_\theta}^{\text{REPO}}$ to approach 0 with the right scale of the regularizer. We call this variant REPO-D.

While the above heuristic provides us with an overall mechanism to control entropy, the exact scale of the regularizer depends on many aspects of the policy gradient optimization: the learning rate, the structure of the gradient, second order effects, etc. We learn the magnitude ζ of the regularizer using a simple control heuristic similar to the adaptive D_{KL} penalty presented in Schulman et al. (2017). Let $\beta_{\boldsymbol{c}}^{\text{REPO-D}} = -\zeta \cdot \Delta\mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s})$. The heuristic proceeds as follows: (1) Estimate $\mathcal{H}_{\pi_\theta}^{\text{init}}$, the policy entropy over the experience collected in this first iteration. (2) Initialize $\zeta = 10^{-3}$. (3) On each iteration, estimate $\mathcal{H}_{\pi_\theta}$, the current policy entropy, and compare it to $\mathcal{H}_{\pi_\theta}^{\text{init}}$. If $\mathcal{H}_{\pi_\theta} < \mathcal{H}_{\pi_\theta}^{\text{init}}$, update $\zeta \leftarrow \zeta \times 2$. If $\mathcal{H}_{\pi_\theta} > \mathcal{H}_{\pi_\theta}^{\text{init}}$, update $\zeta \leftarrow \zeta \div 2$. (4) Clip ζ to the window $\zeta_{\text{min}} \leq \zeta \leq \zeta_{\text{max}}$.

Supporting rare correct actions (REPO-R). Looking back through our learnings thus far, it appears that the most important bang-for-buck in preserving entropy is through raising low probability correct actions. This intuitively corresponds to reinforcing rare but correct solutions under our policy optimization, which is a behavior that we hope to encourage. We can thus apply an entropy regularizer on positive advantage actions only: $\beta_{a,\boldsymbol{c}}^{\text{REPO-R}} = \zeta \cdot \max(A(\boldsymbol{c}, a), 0)$. The effect of this is simple: Wrong (negative-advantage) actions are unaffected and penalized by the negative advantage. For correct actions (positive-advantage) the entropy regularizer reduces advantages for high-probability outcomes, but amplifies low-probability samples (towards a higher entropy state). This does however introduce a small bias to the gradient estimate, as it treats positive and negative advantage samples differently. Different scales of regularizers $\beta_{\boldsymbol{c}}^{\text{REPO-D}}$ and $\beta_{a,\boldsymbol{c}}^{\text{REPO-R}}$ demand different clipping ranges $[\zeta_{\text{min}}, \zeta_{\text{max}}]$: $[10^{-3}, 10^{1}]$ for REPO-D and $[10^{-5}, 10^{-1}]$ for REPO-R.

4 Experiments

With the theory established, we evaluate whether training with REPO yields improvements to strong models on challenging environments when compared to state-of-the-art learning algorithms. We choose `Qwen-3-8B` and `Qwen-3-32B` as our starting policies (Yang et al., 2025).

Environments. *Interactive tool-use agent.* Training scenarios are drawn from the train split (90 problems) of the AppWorld benchmark (Trivedi et al., 2024). The AppWorld Test Normal (*TN*,

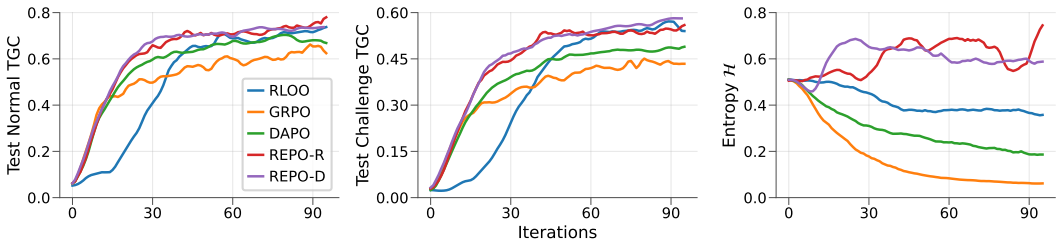

Figure 2: `Qwen-3-32B` AppWorld test performance and token entropy across iterations of training. Each curve shows a mean across multiple random seeds.

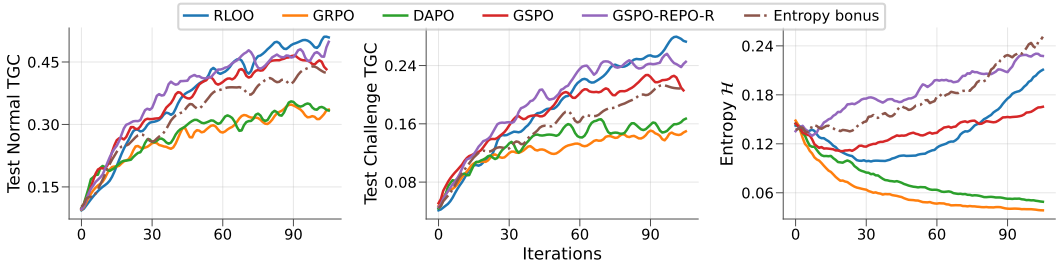

Figure 3: `Qwen-3-8B` AppWorld test performance and token entropy across iterations of training. Each curve shows a mean across multiple random seeds.

168 tasks) and Test Challenge (*TC*, 417 tasks) splits are used for evaluation. Terminal reward is calculated via task-provided unit-tests that check the final state of the environment against ground truth (additional details in App. D.1). *Competition-level mathematics*. Training scenarios are drawn from a non-overlapping quality-filtered subset of the AMC/AIME section of NuminaMath-1.5 (563 problems; Li et al., 2024). AIME 2024 (30 problems) and AIME 2025 (30 problems) are used as evaluation datasets. Terminal reward indicates whether the generated answer matches the reference.

Training details. We run training for 100 iterations on AppWorld, and 200 on AIME, where each iteration consists of an experience collection stage followed by policy optimization on the collected data. For each iteration, 64 (AppWorld) or 128 (AIME) scenarios are sampled without replacement and $K = 6$ rollouts are generated per scenario. Model outputs are generated with the thinking template enabled using `temp` $= 1.0$ and a maximum generation length of 16384 (AppWorld) or 4096 (AIME) tokens.[2] All scenarios yielding identical in-group returns ($\widehat{A}_{\text{RLOO}} = 0$) are filtered out to increase throughput. Additional training details in App. D.2.

Algorithms. For each algorithm, we highlight its distinguishing features with otherwise minimal deviations from the base policy gradient to aid reproducibility (thus, some details and hyperparameter choices may differ slightly from original sources).

RLOO: REINFORCE with the $\widehat{A}_{\text{RLOO}}$ advantage estimator. The collected experience is trained strictly on-policy for 1 epoch using a large minibatch that comprises all collected experience. *GRPO*: Off-policy extension of RLOO that uses normalized Leave None Out (LNO) estimator $\widehat{A}_{\text{GRPO}}$ and symmetric PPO clipping with $\epsilon = 0.2$. With GRPO and all algorithms below we train on the collected experience for 2 epochs with the minibatch size of 128 (AppWorld) or 256 (AIME) trajectories. *LOOP*: A variant of GRPO with non-normalized estimator $\widehat{A}_{\text{RLOO}}$ (RL SOTA on AppWorld at the time of writing). *DAPO*: a variant of LOOP with asymmetric clipping ($\epsilon_{\text{low}} = 0.2$, $\epsilon_{\text{high}} = 0.28$). *GSPO*: An adjustment of LOOP using w^{GSPO} importance weighting and trajectory-based clipping with $\epsilon_{\text{low}}^{\text{GSPO}} = 3 \times 10^{-4}$ and $\epsilon_{\text{high}}^{\text{GSPO}} = 4 \times 10^{-4}$. *REPO* algorithms incorporate entropy control over either DAPO (e.g. in REPO-R) or GSPO (in GSPO-REPO-R) as their base algorithm.

[2]This restricted context window was chosen for experiment iteration speed. Note that `Qwen-3` models can be trained to higher accuracy on AIME with increased token budget.

Algorithm	Test Normal	Best TN	Test Challenge	Best TC	$\mathcal{H}$	$\Delta\mathcal{H}$
RLOO	0.71 ± 0.02	0.73	0.52 ± 0.02	0.54	0.32	-0.19
GRPO	0.61 ± 0.01	0.62	0.42 ± 0.03	0.45	0.06	-0.45
LOOP	0.64 ± 0.04	0.67	0.40 ± 0.01	0.41	0.05	-0.46
DAPO	0.70 ± 0.03	0.73	0.48 ± 0.01	0.49	0.18	-0.33
GSPO	0.62 ± 0.06	0.68	0.51 ± 0.05	0.56	0.42	-0.09
REPO-R	**0.73 ± 0.03**	**0.75**	**0.55 ± 0.05**	**0.62**	0.48	-0.03
REPO-D	0.71 ± 0.03	0.74	0.54 ± 0.02	0.55	0.63	+0.12
GSPO-REPO-R	0.71 ± 0.00	0.71	0.54 ± 0.03	0.57	0.57	+0.06
GSPO-REPO-D	0.64 ± 0.02	0.66	0.41 ± 0.02	0.43	0.52	+0.01

Table 1: Task goal completion scores for AppWorld `Qwen-3-32B` by training algorithm.

Algorithm	Test Normal	Best TN	Test Challenge	Best TC	$\mathcal{H}$	$\Delta\mathcal{H}$
RLOO	**0.48 ± 0.07**	**0.54**	0.25 ± 0.02	**0.28**	0.14	-0.00
GRPO	0.33 ± 0.01	0.34	0.14 ± 0.01	0.15	0.04	-0.10
LOOP	0.27 ± 0.01	0.28	0.14 ± 0.00	0.14	0.04	-0.10
DAPO	0.34 ± 0.02	0.36	0.17 ± 0.01	0.17	0.05	-0.09
GSPO	0.45 ± 0.01	0.46	0.21 ± 0.01	0.21	0.15	+0.01
$\mathcal{H}$ Bonus	0.42 ± 0.04	0.45	0.20 ± 0.01	0.22	0.22	+0.08
REPO-R	0.43 ± 0.01	0.45	0.21 ± 0.00	0.21	0.15	+0.01
REPO-D	0.43 ± 0.00	0.43	0.19 ± 0.01	0.20	0.14	-0.00
GSPO-REPO-R	**0.48 ± 0.04**	0.51	**0.27 ± 0.01**	**0.28**	0.21	+0.07
GSPO-REPO-D	0.44 ± 0.03	0.48	0.22 ± 0.02	0.24	0.17	+0.03

Table 2: Task goal completion scores for AppWorld `Qwen-3-8B` by training algorithm.

5 Results and Discussion

5.1 Distinct algorithms show variable entropy dynamics

Results for AppWorld are in Tabs. 1 and 2 and Figs. 2 and 3. Results for AIME are in Tab. 3. We narrate these results bellow. The conclusions presented hold across all model and dataset combinations.

PPO-like algorithms deplete entropy faster than strictly on-policy. GRPO and LOOP reduce entropy by nearly 90% over training. While RLOO loses considerably less.

Clipping modifications can protect entropy. Following the intuition provided in §3, both DAPO and particularly GSPO retain more entropy than GRPO or LOOP. DAPO and GSPO are methods that have been empirically found to perform better. Here we show that one possible explanation for the improved performance of these methods is that they preserve entropy.

REPO is most effective at entropy preservation. The REPO-D and REPO-R variants, built on top of DAPO and GSPO, consistently yield no loss or even gains in entropy over training. This confirms the effectiveness of our proposed regularization and the understanding it builds upon.

REPO outperforms an entropy bonus. An entropy bonus in reinforcement learning is an additional term in the optimization objective that increase entropy directly (Williams, 1992; Mnih et al., 2016). We compare DAPO with an entropy bonus[3] with REPO. While an entropy bonus aids DAPO, it is worse than REPO and uses more memory[4] (Fig. 3).

[3] We use the same adaptive algorithm as REPO-D to set the coefficient β.

[4] Computing an entropy bonus requires storing the logits in GPU memory whereas computing the log-probability for the select token does not with CCE (Wijmans et al., 2025). Modifying CCE is non-trivial.

Model	Algorithm	AIME24@1	AIME24@64	AIME25@1	AIME25@64	Combined	$\mathcal{H}$	$\Delta\mathcal{H}$
8B	RLOO	0.62	0.79	0.44	0.67	0.63	0.280	-0.020
8B	GRPO	0.59	0.80	0.41	0.53	0.58	0.107	-0.193
8B	LOOP	0.58	0.81	0.41	0.60	0.60	0.074	-0.226
8B	DAPO	0.63	0.80	0.42	0.63	0.62	0.239	-0.061
8B	GSPO	0.63	0.82	0.43	0.67	0.64	0.193	-0.107
8B	REPO-R	0.63	0.78	0.47	0.63	0.63	0.397	+0.097
8B	REPO-D	0.63	0.83	0.42	0.63	0.63	0.353	+0.053
8B	GSPO-REPO-R	0.64	0.81	0.40	0.62	0.62	0.336	+0.036
8B	GSPO-REPO-D	0.64	0.80	0.46	0.66	0.64	0.332	+0.032
32B	RLOO	0.68	0.88	0.50	0.66	0.68	0.145	-0.235
32B	GRPO	0.64	0.82	0.48	0.64	0.65	0.047	-0.333
32B	LOOP	0.67	0.80	0.50	0.68	0.66	0.033	-0.347
32B	DAPO	0.65	0.87	0.55	0.68	0.69	0.319	-0.061
32B	GSPO	0.63	0.83	0.47	0.70	0.66	0.297	-0.083
32B	REPO-R	0.68	0.88	0.47	0.68	0.68	0.469	+0.089
32B	REPO-D	0.64	0.84	0.50	0.72	0.68	0.442	+0.062
32B	GSPO-REPO-R	0.68	0.87	0.50	0.72	0.69	0.422	+0.042
32B	GSPO-REPO-D	0.70	0.86	0.47	0.66	0.67	0.343	-0.037

Table 3: AIME results by parameter count and training algorithm.

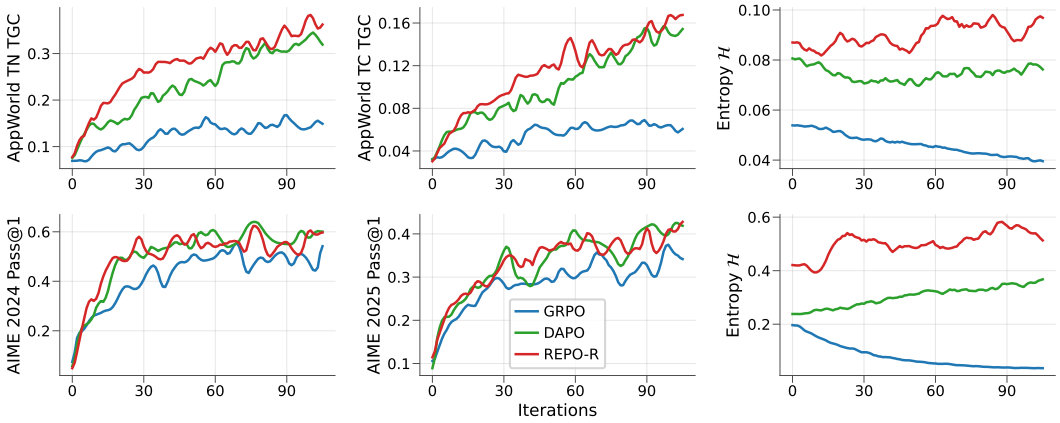

Figure 4: Sequential learning experiment. Top row: We use an AIME-trained model for GRPO, DAPO, REPO-R, and continue training the model on AppWorld. The left and middle plots show Task Goal Completion (TGC) on the normal (TN) and challenging (TC) test sets. A collapsed model (GRPO) does significantly worse than one in which entropy is preserved. Bottom row: We use an AppWorld-trained model and continue training on AIME. The same trends hold. All curves reflect the mean across multiple seeds.

5.2 Entropy preservation and downstream performance

We evaluate the effect of entropy preservation on downstream performance. See Fig. 1 for a preview of these results and App. E for a full breakdown and analysis. We find that methods that preserve per-token entropy over the course of training tend to yield higher final test accuracy than those that don't. This is also captured in the cumulative entropy over training. Methods (and checkpoints) that maintain a higher cumulative entropy over training yield a higher final test accuracy. These trends are stronger on AppWorld than AIME. We hypothesize that this is due to the following: The `Qwen-3` family of models is already heavily optimized for AIME, and so this optimization may have primarily involved sharpening around existing solutions. AppWorld, on the other hand, requires discovering new capabilities (and thus requires more entropy to explore).

5.3 Entropy preservation assists sequential training

Entropy is critical for online reinforcement learning. In this regime, a policy must generate trajectories that yield different returns to collect non-zero advantage samples. If entropy is exhausted, then

learning stops. Here we ask whether entropy collapse hinders the ability of a trained policy to be re-trained in a novel environment.

We train `Qwen-3-8B` first on either the AIME or AppWorld using the same settings as §4. We then take the best checkpoint as the starting point for training on the opposing environment. Fig. 4 shows that policies trained with GRPO in one environment have low entropy even once transferred to the other. This results in a lower peak performance during re-training. On the other hand, DAPO, and especially REPO, start re-training with ample entropy and retain more entropy over the course of training. This results in performance comparable to starting training from `Qwen-3-8B`, thereby demonstrating that collapsing entropy harms re-training.

6 Related work

Reinforcement learning has emerged as the dominant paradigm for aligning pre-trained language models (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022). This approach has been successfully scaled in environments yielding verifiable rewards such as programming and mathematics (Jaech et al., 2024; Lambert et al., 2024; Comanici et al., 2025; Guo et al., 2025; Team et al., 2025).

Empirically, training in this setting has typically been viewed as sharpening the base policy around existing solutions rather than yielding new ones (Gandhi et al., 2025; Liu et al., 2025b; Yue et al., 2025; Zhao et al., 2025). A good pre-trained base policy starts off already calibrated to many reasonable reward functions, and post-training can be viewed as tempering this distribution (Kadavath et al., 2022; Cui et al., 2025). In fact, several works directly exploit this calibration to drive accuracy improvements via unsupervised post-training. Agarwal et al. (2024) simply minimize entropy, Prasad et al. (2024); Zhang et al. (2025); Zuo et al. (2025) align to the model's majority vote distribution, Wang et al. (2025) get by with a single labeled sample, and Shao et al. (2025) even use random rewards. All of these works can be explained by simply allowing policy gradient to sharpen an already calibrated base policy. While this type of approach can help `pass@1`, it harms `pass@k` (Shao et al., 2024; Dang et al., 2025; Yue et al., 2025).

Some works protect against this pathological entropy collapse using modified policy gradient objectives. He et al. (2025) add auxiliary rewards to solutions as a function of their probability rank within a batch. Yu et al. (2025) introduce wider PPO clipping to encourage stronger reinforcement of low probability correct actions. Zheng et al. (2025) propose sequence-level clipping more independent of individual action probabilities. Chen et al. (2025b) reformulate online policy gradient to optimize `pass@k` as opposed to `pass@1`. Most similarly to our work, Cui et al. (2025) derive theoretical results regarding the covariance between advantages and probabilities mediating entropy collapse and then identify the individual tokens most responsible for sharpening and detach their gradients.

Other works impose a D_{KL} penalty during training as an approach for preserving the base policy (e.g., Ziegler et al., 2019; Guo et al., 2025, etc.). However, it has been shown that such an approach limits how much the policy can learn (Korbak et al., 2022; Yang et al., 2024; Wu & Choi, 2025). For this reason, Chen et al. (2025a); Yu et al. (2025) remove the D_{KL} penalty, (Vassoyan et al., 2025) ignore it for a subset of tokens, and (Liu et al., 2025a) iteratively reset the reference policy.

7 Conclusion

In this work, we begin with a theoretical explanation for entropy collapse under policy gradient. We show that this process accelerates under PPO relative to strict on-policy, and how recent policy gradient variants like GSPO or DAPO, implicitly prevent this collapse. We then propose REPO, a novel approach to policy gradient optimization that uses an adaptive controller to stabilize entropy dynamics online. We provide empirical evidence for REPO's effectiveness, training in challenging environments and evaluating on AppWorld, AIME 2024, and AIME 2025. In addition to strong benchmark performance, REPO-trained models yield final policies that have retained their entropy, which we demonstrate enables sequential learning of trained checkpoints in new environments. Overall, we highlight the importance of entropy—and the corresponding online exploration capabilities—for effective policy optimization.

Ethics statement

This paper investigates the properties of policy gradient algorithms for language model reasoning, specifically focusing on the tendency for entropy collapse during training. Our research is primarily theoretical and analytical, involving mathematical analysis and algorithm development. Our work aims to improve entropy during reinforcement learning, which can lead to better exploration and wider diversity in generated outputs. We acknowledge the potential for misuse of advanced language models, including the generation of biased, harmful, or misleading content. We believe that responsible research practices, including transparency in model limitations and potential societal impacts, are crucial for mitigating these risks, and we hope that our research contributes to the development of more robust, creative, and beneficial language models.

Reproducibility statement

Complete proofs for all theoretical claims, along with experimental details and hyperparameters, are included in the appendix. All data points presented in this work are the result of multiple repetitions of each experiment using independent random seeds.

Use of large language models for writing

We acknowledge the use of large language models to assist with typographical corrections, phrasing, and self-review aimed at improving the clarity and structure of this manuscript.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW. (Cited on p. 9)

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024. URL https://aclanthology.org/2024.acl-long.662. (Cited on p. 3)

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive LLM agents. *arXiv preprint arXiv:2502.01600*, 2025a. URL https://arxiv.org/abs/2502.01600. (Cited on p. 3, 9, 18, 25, 30)

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025b. URL https://arxiv.org/pdf/2508.10751. (Cited on p. 1, 9)

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. URL https://arxiv.org/abs/2507.06261. (Cited on p. 1, 9)

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025. URL https://arxiv.org/abs/2505.22617. (Cited on p. 4, 9)

Xingyu Dang, Christina Baek, J Zico Kolter, and Aditi Raghunathan. Assessing diversity collapse in reasoning. In *Scaling Self-Improving Foundation Models without Human Supervision*, 2025. URL https://openreview.net/forum?id=AMiKsHLjQh. (Cited on p. 1, 9)

Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PtSAD3caaA2. (Cited on p. 1)

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025. URL https://arxiv.org/abs/2503.01307. (Cited on p. 9)

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602. (Cited on p. 25)

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948. (Cited on p. 1, 9)

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017. URL https://proceedings.mlr.press/v70/haarnoja17a.html. (Cited on p. 1)

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b. (Cited on p. 1)

Andre He, Daniel Fried, and Sean Welleck. Rewarding the unlikely: Lifting GRPO beyond distribution sharpening. *arXiv preprint arXiv:2506.02355*, 2025. URL https://arxiv.org/abs/2506.02355. (Cited on p. 9)

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9. (Cited on p. 25)

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. URL https://arxiv.org/abs/2412.16720. (Cited on p. 1, 9)

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022. URL https://arxiv.org/abs/2207.05221. (Cited on p. 9)

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Unlocking RL potential for LLM reasoning through refined credit assignment. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5mJrGtXVwz. (Cited on p. 3)

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! *Deep RL Meets Structured Prediction Workshop at ICLR*, 2019. URL https://openreview.net/forum?id=r1lgTGL5DE. (Cited on p. 3)

Tomasz Korbak, Ethan Perez, and Christopher Buckley. RL with KL penalties is better viewed as Bayesian inference. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 1083–1091, 2022. URL https://aclanthology.org/2022.findings-emnlp.77. (Cited on p. 9)

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023. URL https://dl.acm.org/doi/abs/10.1145/3600006.3613165. (Cited on p. 25)

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024. URL https://arxiv.org/abs/2411.15124. (Cited on p. 9)

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath-1.5, 2024. URL https://huggingface.co/datasets/AI-MO/NuminaMath-1.5. (Cited on p. 6)

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025a. URL https://arxiv.org/abs/2505.24864. (Cited on p. 9)

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b. URL https://arxiv.org/abs/2503.20783. (Cited on p. 3, 9)

Sami Marreed, Alon Oved, Avi Yaeli, Segev Shlomov, Ido Levy, Aviad Sela, Asaf Adi, and Nir Mashkif. Towards enterprise-ready computer using generalist agent. *CoRR*, 2025. (Cited on p. 18)

Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, 2017. URL https://peerj.com/articles/cs-103/. (Cited on p. 25)

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016. URL https://proceedings.mlr.press/v48/mniha16.html. (Cited on p. 7)

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html. (Cited on p. 9)

Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason E Weston, and Jane Yu. Self-consistency preference optimization. In *Forty-second International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=94G4eL3RWi. (Cited on p. 9)

Penghui Qi, Zichen Liu, Xiangxin Zhou, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Defeating the training-inference mismatch via fp16. *arXiv preprint arXiv:2510.26788*, 2025. (Cited on p. 15)

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015. URL https://proceedings.mlr.press/v37/schulman15.html. (Cited on p. 3)

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347. (Cited on p. 3, 5)

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, et al. Spurious rewards: Rethinking training signals in RLVR. *arXiv preprint arXiv:2506.10947*, 2025. URL https://arxiv.org/abs/2506.10947. (Cited on p. 9)

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300. (Cited on p. 1, 3, 9)

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1f89885d556929e98d3ef9b86448f951-Abstract.html. (Cited on p. 9)

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998. (Cited on p. 1)

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025. URL https://arxiv.org/abs/2501.12599. (Cited on p. 9)

Sebastian B Thrun. *Efficient exploration in reinforcement learning*. Carnegie Mellon University, 1992. (Cited on p. 1)

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024. URL https://aclanthology.org/2024.acl-long.850/. (Cited on p. 5)

Jean Vassoyan, Nathanaël Beau, and Roman Plaud. Ignore the KL penalty! boosting exploration on critical tokens to enhance RL fine-tuning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6108–6118, 2025. URL https://aclanthology.org/2025.findings-naacl.340. (Cited on p. 9)

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, et al. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*, 2025. URL https://arxiv.org/abs/2504.20571. (Cited on p. 9)

Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gX8C4YPr. (Cited on p. 25)

Erik Wijmans, Brody Huval, Alexander Hertzberg, Vladlen Koltun, and Philipp Kraehenbuehl. Cut your losses in large-vocabulary language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=E4Fk3YuG56. (Cited on p. 7, 25)

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992. URL https://link.springer.com/article/10.1007/bf00992696. (Cited on p. 3, 7, 20)

Fang Wu and Yejin Choi. On the limits of RLVR: Support, entropy, and the illusion of reasoning. In *2nd AI for Math Workshop@ ICML 2025*, 2025. URL https://openreview.net/forum?id=KXtLWJAzgh. (Cited on p. 9)

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388. (Cited on p. 5)

Joy Qiping Yang, Salman Salamatian, Ziteng Sun, Ananda Theertha Suresh, and Ahmad Beirami. Asymptotics of language model alignment. In *2024 IEEE International Symposium on Information Theory (ISIT)*, pp. 2027–2032. IEEE, 2024. URL https://ieeexplore.ieee.org/abstract/document/10619456. (Cited on p. 9)

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476. (Cited on p. 1, 4, 9)

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025. URL https://arxiv.org/abs/2504.13837. (Cited on p. 1, 9)

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised LLM reasoning incentivization. *arXiv preprint arXiv:2504.05812*, 2025. URL https://arxiv.org/abs/2504.05812. (Cited on p. 9)

Ruisi Zhang, Tianyu Liu, Will Feng, Andrew Gu, Sanket Purandare, Wanchao Liang, and Francisco Massa. SimpleFSDP: Simpler fully sharded data parallel with torch. compile. *arXiv preprint arXiv:2411.00284*, 2024. URL https://arxiv.org/abs/2411.00284. (Cited on p. 15, 25)

Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: RL post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*, 2025. URL https://arxiv.org/abs/2504.07912. (Cited on p. 9)

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025. URL https://arxiv.org/pdf/2507.18071. (Cited on p. 1, 3, 5, 9)

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008. URL https://dl.acm.org/doi/10.5555/1620270.1620297. (Cited on p. 1)

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019. URL https://arxiv.org/abs/1909.08593. (Cited on p. 9)

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. TTRL: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025. URL https://arxiv.org/abs/2504.16084. (Cited on p. 9)

A Numerical Considerations

After our initial round of experiments §5, we identified implementation details and numerical effects that substantially influence the experimental results and the entropy dynamics of RL algorithms. We believe similar numerical peculiarities can affect many practitioners and thus the overall story would be incomplete without discussing them.

A.1 Loss of Model Output Precision from FSDP2 Output Casting

As described in App. D.2, we use the FSDP2 framework for distributed training on multiple GPUs (Zhang et al., 2024). In the HuggingFace *Accelerate* library, FSDP2 is configured to cast all module outputs to the chosen floating-point type (e.g., BF16), including the final model outputs, even when the computations involving logits (such as softmax) are performed in full 32-bit precision.

This is the default behavior of the library, and there is no single configuration parameter to switch it off. To preserve full-precision log probabilities, the user must explicitly override the `output_dtype` of the `MixedPrecisionPolicy` (MPP) object (see `fsdp/_fully_shard/_fsdp_api.py` for details).

Naively, this cast should not affect the RL gradients, as the backward pass of such a casting operation is the identity function. Indeed, there appears to be no measurable difference for fully on-policy algorithms like RLOO. The half-precision downcast, however, does measurably impact the numerical stability of the importance weight and thus can affect off-policy algorithms that use clipping, such as LOOP, GRPO, and DAPO.

Fig. 5 empirically demonstrates the clipping bias introduced by the 16-bit rounding when training with DAPO. We observe that when the rounding is present (before the `MixedPrecisionPolicy` fix), more tokens get clipped due to exceeding the higher end of the range ϵ_{high} preventing probability increase for low probability tokens and thus reducing overall entropy. At the same time, fewer tokens are clipped due to ϵ_{low}. The overall effect is the tightening of the clipping on the higher end of the range while relaxing it on the lower end, resulting in the reduced effectiveness of entropy preservation from the asymmetric clipping. It can be further noted that the 16-bit rounding changes the clipping outcome only for a tiny fraction of tokens, fewer than 0.1% of the total number of output tokens. This suggests that a very small number of pivotal tokens play an essential role in learning and warrants further study of this effect.

App. A.3 empirically confirms the significant impact of half-precision rounding on the overall performance and entropy dynamics (see Fig. 7).

A.2 Float16 Training

In our original experiments, the models were trained exclusively in *bfloat16* (BF16), which has become common practice in LLM training because of its higher dynamic range. Recent publications (Qi et al., 2025) reported improved training with *float16* (FP16) floating-point format as its additional 3 mantissa bits enable more accurate gradient representation.

In addition, the choice of floating-point format affects the discrepancy between inference (vLLM) and training policies. These discrepancies are inherent to RL systems with a separate inference server and arise from small differences in model-layer implementations as well as from the lack of batch-size invariance in GPU kernels. In our experiments, we find that FP16 training significantly reduces the inference-training discrepancy (see Fig. 6).

A.3 Ablation Study

Fig. 7 summarizes the ablation study of the numerical tweaks described in Apps. A.1 and A.2 performed for DAPO training on Qwen3 8B. We observe that when the MPP fix and FP16 training are used together, the entropy dynamics of DAPO change completely, from collapse and sub-par exploration to a rapid increase in entropy over the course of training. More generally, we observed improved training across models and algorithm variants when both of the above changes were applied (Tabs. 4 and 5).

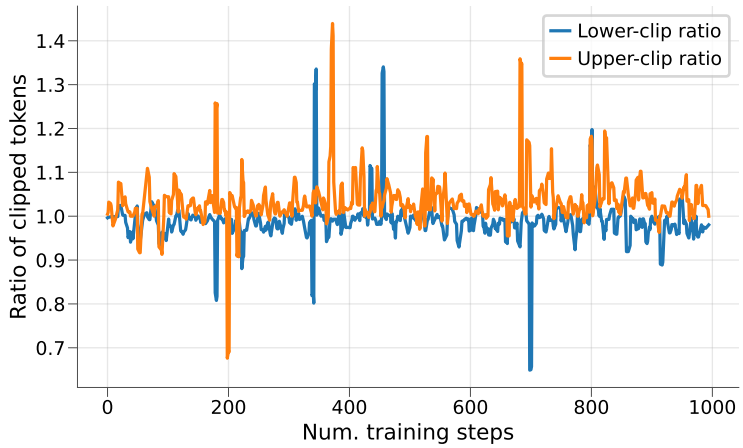

Figure 5: Ratio of clipped tokens before and after the `MixedPrecisionPolicy` fix. 16-bit rounding introduces a subtle bias that causes more tokens to be clipped on the upper end and fewer tokens to be clipped on the lower end of the clipping range $[\epsilon_{\text{low}}, \epsilon_{\text{high}}]$. If not addressed, this asymmetrical bias promotes entropy collapse in algorithms with asymmetric clipping like DAPO. Here the measurements are shown for Qwen3 8B trained with DAPO on AppWorld.

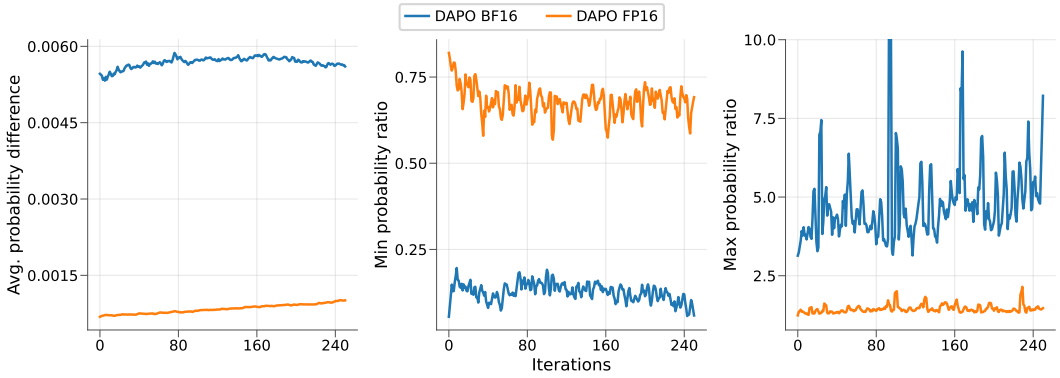

Figure 6: Differences between inference (vLLM) and training policies under BF16 and FP16 training. Additional mantissa bits in the FP16 setup enable much smaller deviations from the behavior policy. Shown from left to right: average differences between token probabilities (lower is better), minimal probability ratio between vLLM and training policies across the experience batch (closer to 1.0 is better), and max. probability ratio (closer to 1.0 is better).

B Bidirectional Entropy Control

Results in App. A.3 show that entropy dynamics can vary significantly in response to relatively minor modifications, and suggest that bidirectional entropy control, rather than simply collapse prevention, is a better framing. We propose two algorithm variants designed to control entropy in both directions in response to the observed behavior.

Bidirectional REPO-R. The first is the bidirectional variant of REPO-R. It is identical to the REPO-R described in §3, except that the sign of the adaptive coefficient ζ flips when the entropy exceeds the target value (e.g., the initial entropy), and the adaptive control is then applied in the range $[-\zeta_{\text{max}}, -\zeta_{\text{min}}]$ instead.

Note that REPO-R is base method agnostic and can be used even with a fully on-policy method like RLOO.

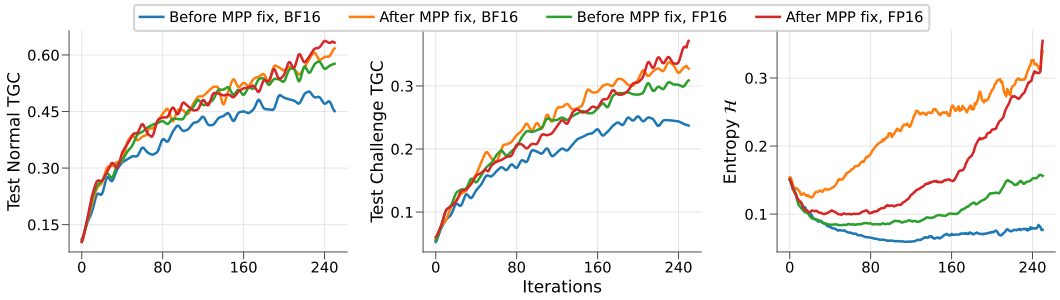

Figure 7: Cumulative effect of the MixedPrecisionPolicy (MPP) fix and FP16 training when applied to DAPO algorithm with Qwen3 8B. Each curve represents the mean of three independent runs (seeds).

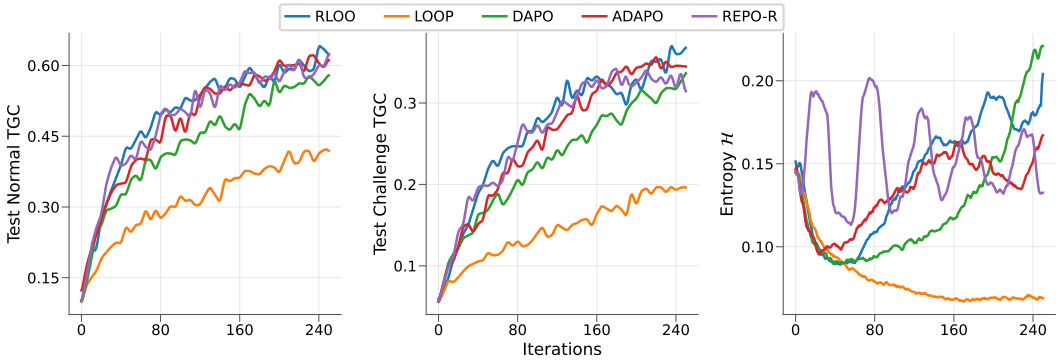

Figure 8: AppWorld test scores and token-level entropy for Qwen3 8B after numerical fixes. ADAPO and REPO-R are used for bidirectional entropy control. GRPO performed very similarly to LOOP and omitted for clarity. Curves show mean values across three seeds.

ADAPO. The second proposed algorithm for bidirectional entropy control is called ADAPO ("Adaptive DAPO"). It utilizes the built-in ability of DAPO's asymmetric clipping to affect entropy and adds an adaptive controller similar to REPO-R. To stabilize entropy with ADAPO we set $\epsilon_{\text{low}} = 0.2$ and allow ϵ_{high} to vary in $[0.2, 0.3]$ range in response to the observed entropy. Specifically:

1. Estimate $\mathcal{H}_{\pi_\theta}^{\text{init}}$, the policy entropy over the experience collected in this first iteration (same as REPO-R).
2. Initialize $\epsilon_{\text{high}} = 0.28$ (initial value used by DAPO).
3. On each iteration, estimate $\mathcal{H}_{\pi_\theta}$, the current policy entropy, and compare it to $\mathcal{H}_{\pi_\theta}^{\text{init}}$. If $\mathcal{H}_{\pi_\theta} < \mathcal{H}_{\pi_\theta}^{\text{init}}$, update $\epsilon_{\text{high}} \leftarrow \epsilon_{\text{high}} \times 1.01$. If $\mathcal{H}_{\pi_\theta} > \mathcal{H}_{\pi_\theta}^{\text{init}}$, update $\epsilon_{\text{high}} \leftarrow \epsilon_{\text{high}} \div 1.01$.
4. Clip ϵ_{high} to the window $[0.2, 0.3]$.

Note that this idea can be applied to any algorithm with asymmetric clipping (e.g. GSPO) therefore an alternative disambiguation is "**ADA**ptive Asymmetric Clipping **P**olicy **O**ptimization"

B.1 Bidirectional Entropy Control: Experiments

We rerun a select subset of experiments with Qwen3 8B and 32B incorporating changes from App. A and bidirectional entropy control mechanisms (see Figs. 8 and 9).

Key observations:

- Both the bidirectional version of REPO-R and ADAPO succeed at keeping entropy close to $\mathcal{H}_{\pi_\theta}^{\text{init}}$. This suggests that the adaptive nature of both methods is more important than the specific entropy control lever.

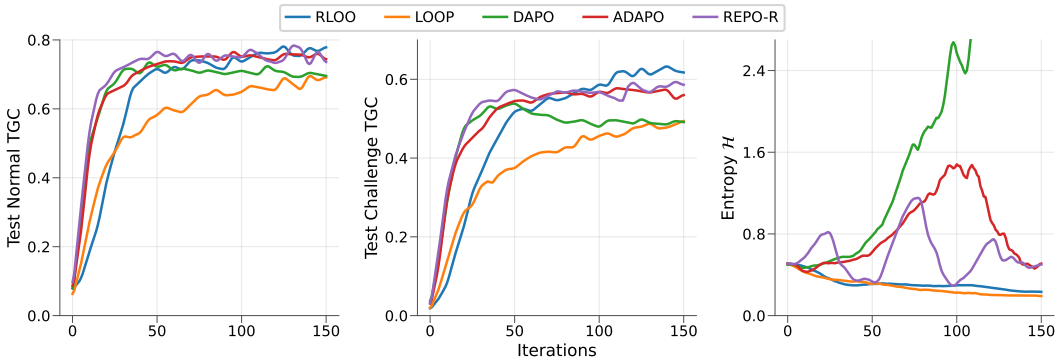

Figure 9: AppWorld test scores and token-level entropy for Qwen3 32B after numerical fixes. ADAPO and REPO-R are used for bidirectional entropy control. GRPO performed very similarly to LOOP and omitted for clarity. DAPO's entropy explodes, leading to instability. Curves show mean values across three seeds.

Algorithm	Test Normal	Best TN	Test Challenge	Best TC
RLOO	**0.59 ± 0.05**	0.64	**0.35 ± 0.06**	**0.41**
LOOP	0.40 ± 0.02	0.42	0.19 ± 0.02	0.22
GSPO	0.56 ± 0.04	0.60	0.32 ± 0.03	0.36
DAPO	0.57 ± 0.03	0.62	0.33 ± 0.03	0.37
ADAPO	**0.59 ± 0.01**	0.60	0.34 ± 0.03	0.36
REPO-R	0.58 ± 0.07	**0.67**	0.32 ± 0.03	0.37

Table 4: Task goal completion scores for AppWorld Qwen3 8B by training algorithm after numerical fixes. REPO-R is the bidirectional version. Test Normal and Test Challenge columns show mean and standard deviation across three independent runs. Best TN/TC columns report the highest evaluation score of any checkpoint across three runs.

- REPO-R and ADAPO are the best-performing off-policy methods in this domain.
- Entropy values oscillate for REPO-R suggesting it could benefit from further improvement of the adaptive heuristic for precise control (e.g. exponential coefficient smaller than 2).
- For non-adaptive DAPO, the entropy explodes in the 32B setup which leads to early deterioration of performance, highlighting the importance of bidirectional control.
- LOOP underperforms despite showing entropy dynamics similar to RLOO. LOOP uses restrictive [0.9,1.1] clipping range for stability which may hinder promotion of high advantage low probability tokens, slowing down learning.
- Remarkably, a fully on-policy method RLOO is firmly among the best methods after the numerical fixes, albeit showing slower initial training in the 32B setup. The entropy dynamics for RLOO changes even between two models in the same model family, highlighting the complexity of exploration behavior.

We recorded the highest score among all our experiments using a simple on-policy algorithm RLOO (Tab. 5) after introducing numerical tweaks described in App. A. We reach 78% success rate on Test Normal and 71% on Test Challenge, significantly exceeding the highest previously reported scores (`https://appworld.dev/leaderboard`) achieved with an agentic GPT-4.1-based system (Marreed et al., 2025). We improve by 7% for TN and 26% for TC compared to previous RL SOTA based on an open-weight model (Chen et al., 2025a).

Algorithm	Test Normal	Best TN	Test Challenge	Best TC
RLOO	**0.78** ± 0.00	**0.79**	**0.62** ± 0.07	**0.71**
LOOP	0.66 ± 0.02	0.68	0.45 ± 0.03	0.47
GSPO	0.69 ± 0.01	0.70	0.50 ± 0.01	0.51
DAPO	0.73 ± 0.04	0.77	0.52 ± 0.02	0.55
ADAPO	0.77 ± 0.01	0.78	0.59 ± 0.04	0.65
REPO-R	0.75 ± 0.02	0.78	0.56 ± 0.06	0.63

Table 5: Task goal completion scores for AppWorld Qwen3 32B by training algorithm after numerical fixes. REPO-R is the bidirectional version.

C Proofs & Derivations

C.1 Broadly used Lemmas

Lemma 1. *The expected score function of policy π_θ at some state $\boldsymbol{s}$ is:*

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[\nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s})\right] = 0$$

Proof.

$$\begin{aligned}
\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[\nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s})\right] &= \sum_a \pi_\theta(a \mid \boldsymbol{s}) \cdot \nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s}) \\
&= \sum_a \nabla_\theta \pi_\theta(a \mid \boldsymbol{s}) \\
&= \nabla_\theta \sum_a \pi_\theta(a \mid \boldsymbol{s}) \\
&= \nabla_\theta(1) \\
&= 0
\end{aligned}$$

■

Lemma 2. *The gradient of a sample estimate $\mathbb{E}_{x\sim P_\theta}\left[f_\theta(x)\right]$ of function f_θ over distribution P_θ is:*

$$\nabla_\theta \mathbb{E}_{x\sim P_\theta}\left[f_\theta(x)\right] = \mathbb{E}_{x\sim P_\theta}\left[\nabla_\theta f_\theta(x) + f_\theta(x) \cdot \nabla_\theta \log P_\theta(x)\right]$$

Proof.

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{x\sim P_\theta}\left[f_\theta(x)\right] &= \sum_x \nabla_\theta \left(P_\theta(x) \cdot f_\theta(x)\right) \\
&= \sum_x \left(P_\theta(x) \cdot \nabla_\theta f_\theta(x) + f_\theta(x) \cdot \underbrace{\nabla_\theta P_\theta(x)}_{P_\theta(x)\nabla_\theta \log P_\theta(x)} \right) \\
&= \sum_x P_\theta(x) \left(\nabla_\theta f_\theta(x) + f_\theta(x) \cdot \nabla_\theta \log P_\theta(x)\right) \\
&= \mathbb{E}_{x\sim P_\theta}\left[\nabla_\theta f_\theta(x) + f_\theta(x) \cdot \nabla_\theta \log P_\theta(x)\right]
\end{aligned}$$

■

Lemma 3. *The gradient of a sample estimate $\mathbb{E}_{x\sim P_\theta}\left[f_\theta(x)\right]$ of function f_θ over distribution P_θ can be baselined for any arbitrary arbitrary b independent of x:*

$$\nabla_\theta \mathbb{E}_{x\sim P_\theta}\left[f_\theta(x) - b\right] = \nabla_\theta \mathbb{E}_{x\sim P_\theta}\left[f_\theta(x)\right]$$

Proof.

$$\begin{aligned}\nabla_\theta \mathbb{E}_{x\sim P_\theta}\left[f_\theta(x) - b\right] &= \mathbb{E}_{x\sim P_\theta}\left[(f_\theta(x) - b)\cdot \nabla_\theta \log P_\theta(x)\right] \\ &= \mathbb{E}_{x\sim P_\theta}\left[f_\theta(x)\cdot \nabla_\theta \log P_\theta(x)\right] - \mathbb{E}_{x\sim P_\theta}\left[b\cdot \nabla_\theta \log P_\theta(x)\right] \\ &= \mathbb{E}_{x\sim P_\theta}\left[f_\theta(x)\cdot \nabla_\theta \log P_\theta(x)\right] - b\cdot \underbrace{\mathbb{E}_{x\sim P_\theta}\left[\nabla_\theta \log P_\theta(x)\right]}_{0} \\ &= \mathbb{E}_{x\sim P_\theta}\left[f_\theta(x)\cdot \nabla_\theta \log P_\theta(x)\right] \\ &= \nabla_\theta \mathbb{E}_{x\sim P_\theta}\left[f_\theta(x)\right]\end{aligned}$$

■

Lemma 4. *The gradient of MDP objective $\mathcal{J}_{\mathrm{MDP}}$ at some state $\boldsymbol{s}$ is:*

$$\nabla_\theta \mathcal{J}_{\mathrm{MDP}}(\boldsymbol{s}) = \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[(R(\boldsymbol{s},a) - b)\cdot \nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s})\right]$$

for any arbitrary baseline b independent of a.

Proof. Largely following (Williams, 1992), Lemma 2, and Lemma 3

$$\begin{aligned}\nabla_\theta \mathcal{J}_{\mathrm{MDP}}(\boldsymbol{s}) &= \nabla_\theta \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[R(\boldsymbol{s},a)\right] \\ &= \nabla_\theta \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[(R(\boldsymbol{s},a) - b)\right] \\ &= \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[(R(\boldsymbol{s},a) - b)\cdot \nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s})\right] + \underbrace{\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[\nabla_\theta\left(R(\boldsymbol{s},a) - b\right)\right]}_{0} \\ &= \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[(R(\boldsymbol{s},a) - b)\cdot \nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s})\right]\end{aligned}$$

■

Lemma 5. *The gradient of the policy entropy at some state $\boldsymbol{s}$ is:*

$$\nabla_\theta \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) = -\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[(\log \pi_\theta(a \mid \boldsymbol{s}) - b)\cdot \nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s})\right]$$

for any arbitrary baseline b independent of a.

Proof. Follows directly from Lemma 4 with $R(\boldsymbol{s},a) = -\log \pi_\theta(a \mid \boldsymbol{s})$.

$$\begin{aligned}\nabla_\theta \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) &= -\nabla_\theta \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[\log \pi_\theta(a \mid \boldsymbol{s})\right] \\ &= -\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[(\log \pi_\theta(a \mid \boldsymbol{s}) - b)\cdot \nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s})\right]\end{aligned}$$

■

Lemma 6. *The expected advantage function $A(\boldsymbol{s},a) \overset{\text{def}}{=} R(\boldsymbol{s},a) - b$, with baseline $V(\boldsymbol{s}) \overset{\text{def}}{=} \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}[R(\boldsymbol{s},a)]$, at some state $\boldsymbol{s}$ is:*

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}[A(\boldsymbol{s},a)] = 0$$

Proof.

$$\begin{aligned}\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}[A(\boldsymbol{s},a)] &= \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}[R(\boldsymbol{s},a) - V(\boldsymbol{s})] \\ &= \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}[R(\boldsymbol{s},a)] - V(\boldsymbol{s}) \\ &= V(\boldsymbol{s}) - V(\boldsymbol{s}) \\ &= 0\end{aligned}$$

■

C.2 Entropy dynamics under policy gradient

Theorem 1. *Given a policy gradient update* $\widehat{\theta} := \theta + \alpha \cdot \nabla_\theta \mathcal{J}_{\text{MDP}}(\boldsymbol{s})$*, the expected change in entropy is approximately:*

$$\Delta \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) \approx -\alpha \cdot \mathbb{E}_{a \sim \pi_\theta(\cdot|\boldsymbol{s}), a' \sim \pi_\theta(\cdot|\boldsymbol{s})} \left[A(\boldsymbol{s}, a) \cdot L(\boldsymbol{s}, a') \cdot u(\boldsymbol{s}, a)^\top u(\boldsymbol{s}, a') \right],$$

where $L(\boldsymbol{s}, a) \stackrel{\text{def}}{=} \log \pi_\theta(a \mid \boldsymbol{s}) - \mathbb{E}_{a \sim \pi_\theta(\cdot|\boldsymbol{s})}[\log \pi_\theta(a \mid \boldsymbol{s})]$ *denotes mean-centered log-probabilities and* $u(\boldsymbol{s}, a) \stackrel{\text{def}}{=} \nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s})$ *denotes the score function for some policy* π_θ *evaluated at state* $\boldsymbol{s}$ *and action* a*.*

Proof. Let $L(\boldsymbol{s}, a) \stackrel{\text{def}}{=} \log \pi_\theta(a \mid \boldsymbol{s}) - \mathbb{E}_{a \sim \pi_\theta(\cdot|\boldsymbol{s})}[\log \pi_\theta(a \mid \boldsymbol{s})]$ denote mean-centered log-probabilities and let $u(\boldsymbol{s}, a) \stackrel{\text{def}}{=} \nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s})$ denote the score function of policy π_θ evaluated at action a and state $\boldsymbol{s}$. Let $g(\boldsymbol{s})$ and $h(\boldsymbol{s})$ denote the respective mean-baselined policy gradient and entropy gradient evaluated on-policy in some state $\boldsymbol{s}$:

$$g(\boldsymbol{s}) = \nabla_\theta \mathcal{J}_{\text{MDP}}(\boldsymbol{s}) = \mathbb{E}_{a \sim \pi_\theta(\cdot|\boldsymbol{s})} \left[A(\boldsymbol{s}, a) \cdot u(\boldsymbol{s}, a) \right]$$
$$h(\boldsymbol{s}) = \nabla_\theta \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) = -\mathbb{E}_{a \sim \pi_\theta(\cdot|\boldsymbol{s})} \left[L(\boldsymbol{s}, a) \cdot u(\boldsymbol{s}, a) \right]$$

Here, each estimator allows for an arbitrary baseline that cancels through the parameter gradient ∇_θ. While the baseline does not influence the exact mathematical construction, it does influence approximations to the change in entropy. Here we chose mean baselines to center the policy, minimize variance in each gradient estimator, and to agree with a tabular softmax approximation of the change in entropy (see Corollary 2).

Using the first-order Taylor approximation: $\mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s} \,;\, \theta + \alpha \cdot g) \approx \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s} \,;\, \theta) + \alpha \cdot g^\top h$, for small learning rate α, the expected change in entropy from a policy gradient update in state $\boldsymbol{s}$ is:

$$\begin{aligned}
\Delta \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) &\approx \alpha \cdot g(\boldsymbol{s})^\top h(\boldsymbol{s}) \\
&= -\alpha \cdot \left(\mathbb{E}_{a \sim \pi_\theta(\cdot|\boldsymbol{s})} \left[A(\boldsymbol{s}, a) \cdot u(\boldsymbol{s}, a) \right] \right)^\top \left(\mathbb{E}_{a' \sim \pi_\theta(\cdot|\boldsymbol{s})} \left[L(\boldsymbol{s}, a') \cdot u(\boldsymbol{s}, a') \right] \right) \\
&= -\alpha \cdot \mathbb{E}_{a \sim \pi_\theta(\cdot|\boldsymbol{s}), a' \sim \pi_\theta(\cdot|\boldsymbol{s})} \left[A(\boldsymbol{s}, a) \cdot L(\boldsymbol{s}, a') \cdot u(\boldsymbol{s}, a)^\top u(\boldsymbol{s}, a') \right]
\end{aligned}$$

■

C.3 Approximate entropy dynamics under policy gradient

Corollary 1. *Assuming* $u(\boldsymbol{s}, a)^\top u(\boldsymbol{s}, a') = 0$ *for all* $a \neq a'$*, the change in entropy is approximately:*

$$\Delta \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) \propto -\mathbb{E}_{a \sim \pi_\theta(\cdot|\boldsymbol{s})} \left[A(\boldsymbol{s}, a) \cdot L(\boldsymbol{s}, a) \cdot \pi_\theta(a \mid \boldsymbol{s}) \right]$$

Proof. Assuming the score vectors satisfy orthogonality of the off-diagonal terms such that $u(\boldsymbol{s}, a)^\top u(\boldsymbol{s}, a') = 0$ for $a \neq a'$, the double expectation can be collapsed, yielding:

$$\Delta \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) \approx -\alpha \cdot \mathbb{E}_{a \sim \pi_\theta(\cdot|\boldsymbol{s})} \left[\pi_\theta(a \mid \boldsymbol{s}) \cdot A(\boldsymbol{s}, a) \cdot L(\boldsymbol{s}, a) \cdot \|u(\boldsymbol{s}, a)\|^2 \right]$$

Assuming independence of the squared gradient norm magnitude, such that it can be treated as a constant with respect to the expectation,

$$\Delta \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) \propto -\mathbb{E}_{a \sim \pi_\theta(\cdot|\boldsymbol{s})} [A(\boldsymbol{s}, a) \cdot L(\boldsymbol{s}, a) \cdot \pi_\theta(a \mid \boldsymbol{s})]$$

■

C.4 Entropy dynamics under policy gradient for tabular softmax policies

Proposition 1. *For two functions* $f(x)$ *and* $g(x)$ *over samples* $x \sim \pi_S$ *of a softmax distribution* $\pi_S(x) = \exp(S_x) / \sum_k \exp(S_k)$*, the dot product of expected gradients is:*

$$\left\langle \mathbb{E}_{x \sim \pi_S}[f(x) \cdot \nabla_S \log \pi_S(x)] \quad , \quad \mathbb{E}_{y \sim \pi_S}[g(y) \cdot \nabla_S \log \pi_S(y)] \right\rangle = \mathbb{E}_{x \sim \pi_S}[\pi_S(x) \cdot (f(x) - \bar{f}) \cdot (g(x) - \bar{g})],$$

where $\bar{f} = \mathbb{E}_{x \sim \pi_S}[f(x)]$ *and* $\bar{g} = \mathbb{E}_{x \sim \pi_S}[g(x)]$*.*

Proof. First, let's compute $\nabla_S \log \pi_S(x)$ for the softmax distribution:

$$\log \pi_S(x) = \log \frac{\exp(S_x)}{\sum_k \exp(S_k)} = S_x - \log \sum_k \exp(S_k)$$

$$\nabla_{S_z} \log \pi_S(x) = \mathbb{1}_{x=z} - \frac{\exp(S_z)}{\sum_k \exp(S_k)} = \mathbb{1}_{x=z} - \pi_S(z)$$

where $\mathbb{1}_{x=y}$ is the indicator function (1 if $x = y$, 0 otherwise).

Now let's compute the dot product $\nabla_S \log \pi_S(x)^\top \nabla_S \log \pi_S(y)$:

$$\begin{aligned}
\nabla_S \log \pi_S(x)^\top \nabla_S \log \pi_S(y) &= \sum_j (\mathbb{1}_{x=j} - \pi_S(j))(\mathbb{1}_{y=j} - \pi_S(j)) \\
&= \sum_j (\mathbb{1}_{x=j} \cdot \mathbb{1}_{y=j} - \mathbb{1}_{x=j}\pi_S(j) - \pi_S(j) \cdot \mathbb{1}_{y=j} + \pi_S(j)^2) \\
&= \mathbb{1}_{x=y} - \pi_S(x) - \pi_S(y) + \mathbb{E}_{z\sim\pi_S}[\pi_S(z)]
\end{aligned}$$

Now we can compute the dot product of expected gradients:

$$\begin{aligned}
&\left\langle \mathbb{E}_{x\sim\pi_S}[f(x) \cdot \nabla_S \log \pi_S(x)], \mathbb{E}_{y\sim\pi_S}[g(y) \cdot \nabla_S \log \pi_S(y)] \right\rangle \\
&= \mathbb{E}_{x\sim\pi_S, y\sim\pi_S}[f(x) \cdot g(y) \cdot \nabla_S \log \pi_S(x)^\top \nabla_S \log \pi_S(y)] \\
&= \mathbb{E}_{x\sim\pi_S, y\sim\pi_S}[f(x) \cdot g(y) \cdot (\mathbb{1}_{x=y} - \pi_S(x) - \pi_S(y) + \mathbb{E}_{z\sim\pi_S}[\pi_S(z)])]
\end{aligned}$$

Let's compute each term separately:

$$\begin{aligned}
\mathbb{E}_{x\sim\pi_S, y\sim\pi_S}[f(x) \cdot g(y) \cdot \mathbb{1}_{x=y}] &= \mathbb{E}_{x\sim\pi_S}[\pi_S(x) \cdot f(x) \cdot g(x)] \\
\mathbb{E}_{x\sim\pi_S, y\sim\pi_S}[f(x) \cdot g(y) \cdot \pi_S(x)] &= \mathbb{E}_{x\sim\pi_S}[f(x) \cdot \pi_S(x)] \cdot \mathbb{E}_{y\sim\pi_S}[g(y)] = \mathbb{E}_{x\sim\pi_S}[\pi_S(x) \cdot f(x)] \cdot \bar{g} \\
\mathbb{E}_{x\sim\pi_S, y\sim\pi_S}[f(x) \cdot g(y) \cdot \pi_S(y)] &= \mathbb{E}_{x\sim\pi_S}[f(x)] \cdot \mathbb{E}_{y\sim\pi_S}[g(y) \cdot \pi_S(y)] = \bar{f}\mathbb{E}_{y\sim\pi_S}[\pi_S(y) \cdot g(y)] \\
\mathbb{E}_{x\sim\pi_S, y\sim\pi_S}[f(x) \cdot g(y) \cdot \mathbb{E}_{z\sim\pi_S}[\pi_S(z)]] &= \mathbb{E}_{z\sim\pi_S}[\pi_S(z)] \cdot \mathbb{E}_{x\sim\pi_S}[f(x)] \cdot \mathbb{E}_{y\sim\pi_S}[g(y)] = \mathbb{E}_{x\sim\pi_S}[\pi_S(x)] \cdot \bar{f} \cdot \bar{g}
\end{aligned}$$

Therefore:

$$\begin{aligned}
&\left\langle \mathbb{E}_{x\sim\pi_S}[f(x) \cdot \nabla_S \log \pi_S(x)], \mathbb{E}_{y\sim\pi_S}[g(y) \cdot \nabla_S \log \pi_S(y)] \right\rangle \\
&= \mathbb{E}_{x\sim\pi_S}[\pi_S(x) \cdot f(x) \cdot g(x)] - \mathbb{E}_{x\sim\pi_S}[\pi_S(x) \cdot f(x)] \cdot \bar{g} - \bar{f} \cdot \mathbb{E}_{x\sim\pi_S}[\pi_S(x) \cdot g(x)] + \mathbb{E}_{x\sim\pi_S}[\pi_S(x)] \cdot \bar{f} \cdot \bar{g} \\
&= \mathbb{E}_{x\sim\pi_S}[\pi_S(x) \cdot (f(x) \cdot g(x) - f(x) \cdot \bar{g} - \bar{f} \cdot g(x) + \bar{f} \cdot \bar{g})] \\
&= \mathbb{E}_{x\sim\pi_S}[\pi_S(x) \cdot (f(x) - \bar{f}) \cdot (g(x) - \bar{g})]
\end{aligned}$$

where $\bar{f} = \mathbb{E}_{x\sim\pi_S}[f(x)]$ and $\bar{g} = \mathbb{E}_{x\sim\pi_S}[g(x)]$. ■

The above proposition holds for simple softmax policies, but involves a much more complex gradient term and inner product for generic transformer-based policies.

Corollary 2. *Under a tabular softmax policy, a policy gradient update* $\widehat{\theta} := \theta + \alpha \cdot \nabla_\theta \mathcal{J}_{\text{MDP}}$ *changes the entropy approximately:*

$$\Delta \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) \approx -\alpha \cdot \mathbb{E}_{a\sim\pi_S(\cdot|\boldsymbol{s})}\left[\pi_S(a \mid \boldsymbol{s}) \cdot \left(\log \pi_S(a \mid \boldsymbol{s}) - \overline{\log \pi_S(\cdot \mid \boldsymbol{s})}\right) \cdot \left(R(\boldsymbol{s}, a) - \overline{R(\boldsymbol{s})}\right)\right]$$

where $\overline{\log \pi_S(\cdot \mid \boldsymbol{s})} = \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[\log \pi_S(a \mid \boldsymbol{s})\right]$ *and* $\overline{R(\boldsymbol{s})} = \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[R(\boldsymbol{s}, a)\right]$.

Proof. Let $g(\boldsymbol{s})$ and $h(\boldsymbol{s})$ denote the respective policy gradient and entropy gradient evaluated on-policy in some state $\boldsymbol{s}$:

$$\begin{aligned}
g(\boldsymbol{s}) &= \nabla_\theta \mathcal{J}_{\text{MDP}}(\boldsymbol{s}) = \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[R(\boldsymbol{s}, a) \cdot \nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s})\right] \\
h(\boldsymbol{s}) &= \nabla_\theta \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) = -\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})}\left[\log \pi_\theta(a \mid \boldsymbol{s}) \cdot \nabla_\theta \log \pi_\theta(a \mid \boldsymbol{s})\right]
\end{aligned}$$

Using the first-order Taylor approximation: $\mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}\,;\,\theta + \alpha \cdot g) \approx \mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}\,;\,\theta) + \alpha \cdot g^\top h$, for small learning rate α, the expected change in entropy from a policy gradient update in state $\boldsymbol{s}$ is:

$$\begin{aligned}\Delta\mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) &\approx \alpha \cdot g(\boldsymbol{s})^\top h(\boldsymbol{s}) \\ &= -\alpha \cdot \mathbb{E}_{a\sim\pi_S(\cdot|\boldsymbol{s})}\left[\pi_S(a \mid \boldsymbol{s}) \cdot \left(\log \pi_S(a \mid \boldsymbol{s}) - \overline{\log \pi_S(\cdot \mid \boldsymbol{s})}\right) \cdot \left(R(\boldsymbol{s}, a) - \overline{R(\boldsymbol{s})}\right)\right]\end{aligned}$$

The second line follows Prop. 1. Note that gradient interactions through the softmax automatically center the reward function, i.e., $A(\boldsymbol{s}, a) = R(\boldsymbol{s}, a) - V\boldsymbol{s} = R(\boldsymbol{s}, a) - \overline{R(\boldsymbol{s})}$. The log-probabilities, too, are centered as here they reflect $R(\boldsymbol{s}, a) = -\log \pi_S(a \mid \boldsymbol{s})$. This yields a form equivalent to Corollary 1. ■

C.5 Entropy dynamics under clipped PPO

Proposition 2. *Given two distributions $\pi(x)$ and $\phi(x)$ with constraint $\frac{\pi(x)}{\phi(x)} \le 1 + \epsilon$ for all x, their relative entropy is bound by*

$$\mathcal{H}(\pi) \le (1 + \epsilon) \cdot \mathcal{H}(\phi)$$

Proof. Let's parametrize $\pi(x) = \beta_x \phi(x)$ with $\beta_x \ge 0$ and compute its probability

$$\begin{aligned}\mathcal{H}(\pi) &= -\mathbb{E}_{x\sim\pi}\left[\log \pi(a)\right] \\ &= -\mathbb{E}_{x\sim\pi}\left[\log \phi(a)\right] - \mathbb{E}_{x\sim\pi}\left[\log \beta_x\right] \\ &= -\mathbb{E}_{x\sim\pi}\left[\log \phi(a)\right] - \underbrace{\mathbb{E}_{x\sim\pi}\left[\log \frac{\pi(x)}{\phi(x)}\right]}_{D_{\mathrm{KL}}(\pi\|\phi)\ge 0} \\ &\le -\mathbb{E}_{x\sim\pi}\left[\log \phi(a)\right] \\ &= -\mathbb{E}_{x\sim\phi}\left[\frac{\pi(a)}{\phi(a)} \log \phi(a)\right] \\ &= \mathbb{E}_{x\sim\phi}\left[\beta_x \cdot -\log \phi(a)\right] \\ &\le \mathbb{E}_{x\sim\phi}\left[(1 + \epsilon) \cdot -\log \phi(a)\right] \\ &= (1 + \epsilon) \cdot \mathcal{H}(\phi)\end{aligned}$$

The second-last line uses $-\log \phi(a) \ge 0$ and $\beta_x \le (1 + \epsilon)$ by definition, hence $\beta_x \cdot -\log \phi(a) \le (1 + \epsilon) \cdot -\log \phi(a)$. ■

Theorem 2. *Proximal Policy Optimization (PPO) bounds the entropy $\mathcal{H}_{\pi_\theta^{new}}(\cdot \mid \boldsymbol{s})$ of the updated policy by the original policy entropy $\mathcal{H}_{\pi_\theta^{old}}(\cdot \mid \boldsymbol{s})$ such that:*

$$(1 - \epsilon_{low}) \cdot \mathcal{H}_{\pi_\theta^{old}}(\cdot \mid \boldsymbol{s}) \le \mathcal{H}_{\pi_\theta^{new}}(\cdot \mid \boldsymbol{s}) \le (1 + \epsilon_{high}) \cdot \mathcal{H}_{\pi_\theta^{old}}(\cdot \mid \boldsymbol{s})$$

Proof. Applying Prop. 2 to $\frac{\pi_\theta^{\text{new}}}{\pi_\theta^{\text{old}}} \le 1 + \epsilon_{\text{high}}$ yields the upper bound

$$\mathcal{H}_{\pi_\theta^{\text{new}}}(\cdot \mid \boldsymbol{s}) \le (1 + \epsilon_{\text{high}}) \cdot \mathcal{H}_{\pi_\theta^{\text{old}}}(\cdot \mid \boldsymbol{s}).$$

Applying Prop. 2 to $1 - \epsilon_{\text{low}} \le \frac{\pi_\theta^{\text{new}}}{\pi_\theta^{\text{old}}}$ (eqivalently $\frac{\pi_\theta^{\text{old}}}{\pi_\theta^{\text{new}}} \le \frac{1}{1-\epsilon_{\text{low}}}$ yields the lower bound

$$\mathcal{H}_{\pi_\theta^{\text{old}}}(\cdot \mid \boldsymbol{s}) \le \frac{1}{1 - \epsilon_{\text{low}}} \cdot \mathcal{H}_{\pi_\theta^{\text{new}}}(\cdot \mid \boldsymbol{s})$$

or equivalently

$$(1 - \epsilon_{\text{low}}) \cdot \mathcal{H}_{\pi_\theta^{\text{old}}}(\cdot \mid \boldsymbol{s}) \le \mathcal{H}_{\pi_\theta^{\text{new}}}(\cdot \mid \boldsymbol{s}).$$

■

C.6 Entropy change under A_{REPO} advantage function

Proposition 3. *For advantage* $A_{\mathrm{REPO}}(\boldsymbol{s}, a) \stackrel{\text{def}}{=} A(\boldsymbol{s}, a) - \beta_{\boldsymbol{s}} \cdot L(\boldsymbol{s}, a)$*, the first–order change in entropy induced by a policy–gradient step is:*

$$\Delta\mathcal{H}^{\mathrm{REPO}}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) \approx \Delta\mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) + \beta_{\boldsymbol{s}} \cdot \alpha \cdot \left\| \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})} \left[L(\boldsymbol{s}, a) \cdot u(\boldsymbol{s}, a) \right] \right\|^2 .$$

Proof. Let $g(\boldsymbol{s}) = \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})} \left[A(\boldsymbol{s}, a) \cdot u(\boldsymbol{s}, a) \right]$ and $h(\boldsymbol{s}) = -\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})} \left[L(\boldsymbol{s}, a) \cdot u(\boldsymbol{s}, a) \right]$ denote the respective policy gradient and entropy gradient evaluated on-policy in some state $\boldsymbol{s}$.

Using A_{REPO}, the policy gradient becomes:

$$g_{\mathrm{REPO}}(\boldsymbol{s}) = \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})} \left[(A(\boldsymbol{s}, a) - \beta_{\boldsymbol{s}} L(\boldsymbol{s}, a)) \cdot u(\boldsymbol{s}, a) \right]$$

The first–order entropy change is:

$$\begin{aligned}
\Delta\mathcal{H}^{\mathrm{REPO}}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) &\approx \alpha \cdot g_{\mathrm{REPO}}(\boldsymbol{s})^\top h(\boldsymbol{s}) \\
&= \alpha \cdot \left(\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})} \left[(A(\boldsymbol{s}, a) - \beta_{\boldsymbol{s}} L(\boldsymbol{s}, a)) \cdot u(\boldsymbol{s}, a) \right] \right)^\top h(\boldsymbol{s}) \\
&= \alpha \cdot \left(\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})} \left[A(\boldsymbol{s}, a) \cdot u(\boldsymbol{s}, a) \right] \right)^\top h(\boldsymbol{s}) - \beta_{\boldsymbol{s}} \cdot \alpha \cdot \left(\mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})} \left[L(\boldsymbol{s}, a)) \cdot u(\boldsymbol{s}, a) \right] \right)^\top h(\boldsymbol{s}) \\
&= \alpha \cdot g(\boldsymbol{s})^\top h(\boldsymbol{s}) + \beta_{\boldsymbol{s}} \cdot \alpha \cdot h(\boldsymbol{s})^\top h(\boldsymbol{s}) \\
&= \Delta\mathcal{H}_{\pi_\theta}(\cdot \mid \boldsymbol{s}) + \beta_{\boldsymbol{s}} \cdot \alpha \cdot \left\| \mathbb{E}_{a\sim\pi_\theta(\cdot|\boldsymbol{s})} \left[L(\boldsymbol{s}, a) \cdot u(\boldsymbol{s}, a) \right] \right\|^2 .
\end{aligned}$$

■

D ADDITIONAL EXPERIMENT DETAILS

D.1 ENVIRONMENTS

Interactive tool-use agent. For the AppWorld benchmark, rollouts proceed in turns (up to 30 turns during training and 50 during evaluation), in a manner akin to an interactive notebook.

During each turn, the model generations are parsed to extract any Python code blocks, potentially containing calls to AppWorld API. These are executed to retrieve information or alter the environment state. The outputs of successful API calls or the error trace of incorrect calls appear in the agent's context after each turn. Once done, the agent may mark a task as completed at which point it is assessed whether the task state was updated successfully. Failure to mark the task as complete within the turn or context limit (32K) results in a failure. Sparse outcome-based rewards in $[0, 1]$ are assigned during training as the fraction of passing unit-tests. Binary rewards in $\{0, 1\}$ are used during evaluation requiring complete correctness.

Mathematical reasoning. For the AIME benchmarks, model responses are processed and scored using the Eleuther AI lm-eval-harness Minerva math parsing utilities (Gao et al., 2024). The final unnormalized answer is first identified and parsed, then the answer is normalized to remove units, formatting, etc., and finally equivalence between the model answer and reference answer is determined using Sympy (Meurer et al., 2017).

D.2 TRAINING

Experiments are executed on 3 NVIDIA H100 8-GPU nodes. One node is used for rollout generation one for learning, and one for evaluation. Rollouts are generated using two instances of vLLM (Kwon et al., 2023) servers using 4 GPUs each with tensor parallelism. Custom RL implementation based on FSDP2 (Zhang et al., 2024) is used for training. To account for any discrepancies between sampling and training subsystems, the log-probabilities of rollout tokens are recalculated on the training node to ensure accurate importance weights for backpropagation. Cut-Cross-Entropy (CCE) is used to reduce the memory footprint during training by preventing the materialization of all logits except the target (Wijmans et al., 2025). Models are fine-tuned with LoRA ($\texttt{rank} = 16$, $\alpha = 32$) on the self-attention (key, value, query, output) and MLP modules (Hu et al., 2022). We use an `AdamW` optimizer with a constant learning rate of 5×10^{-5}, $\texttt{weight-decay} = 0.01$, and gradient clipping with $\texttt{max-norm} = 0.1$. To speed up rollout collection, we introduce an early stopping criteria. Once at least 4/6 rollouts per task and 90% of total rollouts are collected, we immediately proceed to training to prevent bottlenecks caused by very few extra long generations (Wijmans et al., 2020; Chen et al., 2025a).

E ADDITIONAL RESULTS

E.1 GEOMETRIC INTERPRETATION OF REPO

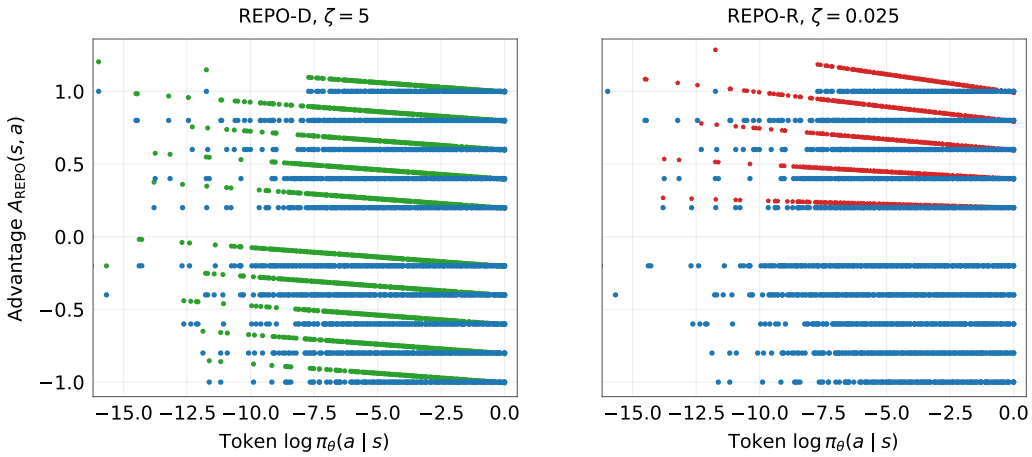

Figure 10: The REPO transformation rotates $(A, \log \pi)$ pairs, promoting low probability actions. Original unmodified advantages shown in blue, advantages A_{REPO} are shown in green for REPO-D and in red for REPO-R.

The transformation induced by each REPO algorithm can be viewed in Fig. 10. REPO-D reflects a consistent rotation across the space, boosting the advantages of actions proportional to their surprisals ($-\log \pi_\theta$). REPO-R instead rotates only positive advantage actions, and does so proportionally, not only to the surprisal, but to the magnitude of the advantage. This strongly reinforces low-probability correct actions, especially when they yield outcomes significantly better than average for a given batch of experience.

Fig. 10 uses data from the `Qwen-3-8B` AIME experiment. The parameters of the algorithm are revealed in the structure of the data: there are 5 distinct positive and negative advantage values, corresponding to 5 unique outcomes of group-based advantage estimation (1 success / 5 failures, 2 successes / 4 failures, etc.). Groups with zero advantages are filtered out.

With the appropriate value of ζ, REPO-D transformation counteracts the covariance-like term in $\Delta\mathcal{H}$ approximation, therefore REPO-D is short for REPO-Decorrelate. REPO-R is a shorthand for REPO-Rescale, as it rescales the advantages by $1 - \zeta \cdot L(\boldsymbol{s}, a)$.

E.2 DYNAMICS OF ENTROPY AND TEST-ACCURACY DURING TRAINING

We computed the average per-token entropy at each iteration of training for all of our training runs (averaging over all tokens generated during a rollout, and averaging over all rollouts at a given iteration). We studied how this quantity evolved over the course of training for several baseline algorithms (RLOO, GRPO, LOOP, GSPO) and several variants of our REPO algorithm (REPO-R, REPO-D, GSPO-REPO-R, and GSPO-REPO-D). Figure 11 shows how the per-token entropy co-evolves with the test accuracy over the course of training for each algorithm. We first observe that the REPO algorithms typically preserve much higher entropy than baselines. For challenging model-task pairs where baselines reduce policy entropy prior to achieving high test accuracy (e.g. `Qwen-3-8B` on all tasks, and `Qwen-3-32B` on AppWorld Test-Normal), REPO algorithms preserve entropy for longer and achieve higher peak test accuracy. For model-task pairs where the baselines reduce entropy late in training, after test accuracy is largely saturated, REPO achieves comparable peak test accuracy to baselines.

E.3 DEPENDENCE OF TEST ACCURACY ON CUMULATIVE ENTROPY DURING TRAINING

We hypothesized that the test accuracy at a given checkpoint is highly dependent on the *cumulative* entropy (intuitively, the time integral of the average per-token entropy) experienced over the course of previous training iterations. Figure 12 plots the test accuracy and cumulative entropy for each checkpoint of each training run of each learning algorithm studied. We observe test accuracy on a more difficult learning task (AppWorld Test-Normal) show sustained increases in test accuracy with additional cumulative entropy even late into training, whereas AIME24 and AIME25 require less cumulative entropy to achieve peak test accuracy. We quantified the dependence of test accuracy on cumulative entropy via mutual information (Table 6). We estimated (i) the mutual information between the test accuracy and the cumulative entropy and (ii) the mutual information between the test accuracy and the iteration using histograms. We found that cumulative entropy is more predictive than the iteration number. We also confirm the relatively stronger dependence of test accuracy on cumulative entropy in AppWorld Test-Normal (MI=0.858 for `Qwen-3-8B` and MI=0.612 for `Qwen-3-32B`) compared to the AIME24 and AIME25 environments (MI $\approx$ 0.2 for both models).

	MI (Iteration)	MI (Cumulative Entropy)
Qwen3 8B - AIME25	0.182	**0.205**
Qwen3 32B - AIME25	0.183	**0.191**
Qwen3 8B - AIME24	0.131	**0.170**
Qwen3 32B - AIME24	**0.146**	0.133
Qwen3 8B - AppWorld Test-Normal	0.566	**0.858**
Qwen3 32B - AppWorld Test-Normal	0.507	**0.612**

Table 6: Quantifying the dependence of test accuracy on cumulative entropy during training.

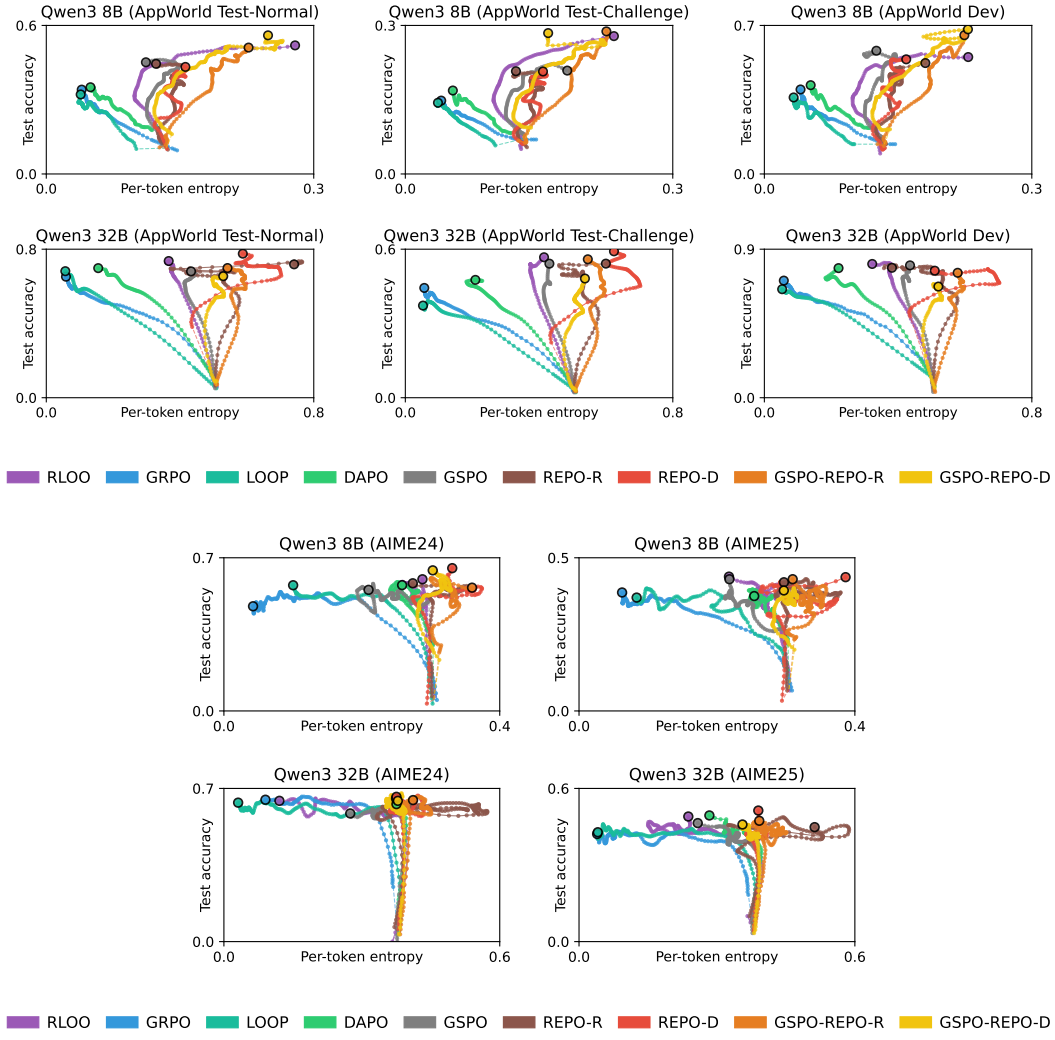

Figure 11: Trajectory of per-token entropy and test accuracy during training for several baseline algorithms several REPO algorithms in the AppWorld environment (top grid) and AIME (bottom grid). Each curve shows the average trajectory over multiple training runs with different seeds.

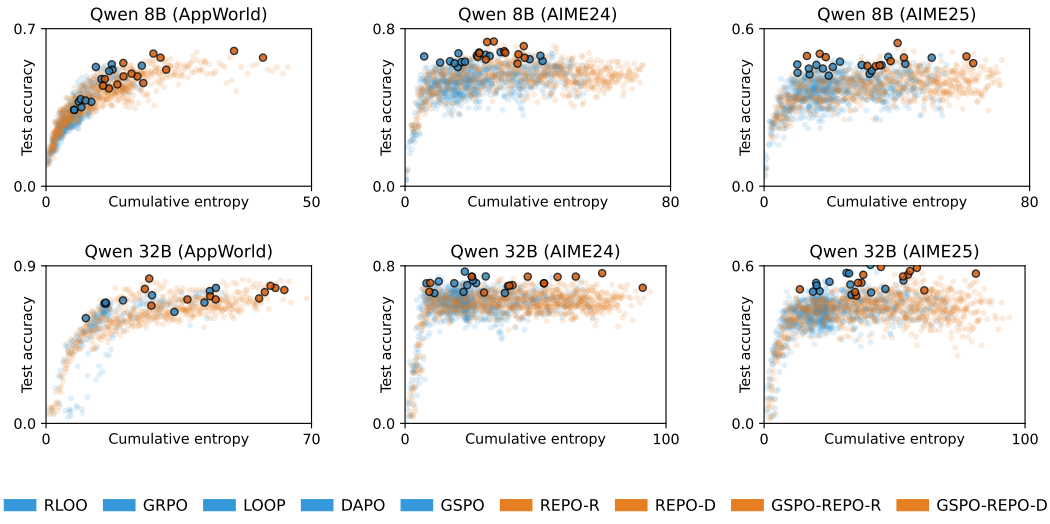

Figure 12: Accuracy on test set versus cumulative per-token entropy (sum of average per-token entropies during the training run up to that point) for all training checkpoints. Dark points show checkpoints with peak test accuracy. AppWorld test accuracy is measured on the Test-Normal set.

F QWEN 2.5 EXPERIMENTS

In previous publications, methods like LOOP performed well with "non-thinking" models such as Qwen 2.5 32B (Chen et al., 2025a). In our Qwen3 experiments however, LOOP (and very similarly, GRPO) experienced early entropy collapse and underperformed compared to other RL methods.

We conducted additional experiments (see Fig. 13 and Tab. 7) to determine whether this discrepancy arises from differences in model behavior or from implementation details. Key observations:

- Qwen 2.5 32B exhibits a significantly higher *initial* success rate (before the first training iteration) compared to Qwen 3 models. For example, on Test Normal the initial success rate is close to 40% versus under 10% for Qwen 3.
- The best results on the hardest test split (Test Challenge) are substantially lower for Qwen 2.5 compared to Qwen 3, most likely reflecting the limitations of the respective base models.
- We were able to replicate and exceed results reported in previous work for Qwen 2.5 32B: the success rate of our best-performing LOOP checkpoints surpasses those in Chen et al. (2025a) by approximately 7% on Test Normal and 9% on Test Challenge. This improvement is most likely attributable to the numerical changes described in App. A, as our setup and hyperparameters for Qwen 2.5 closely match those in Chen et al. (2025a) in all other respects.
- Unlike in our Qwen 3 experiments, LOOP/GRPO do not experience rapid entropy collapse, whereas RLOO does, suggesting that base-model characteristics play a major role in entropy dynamics during training irrespective of the RL algorithm.

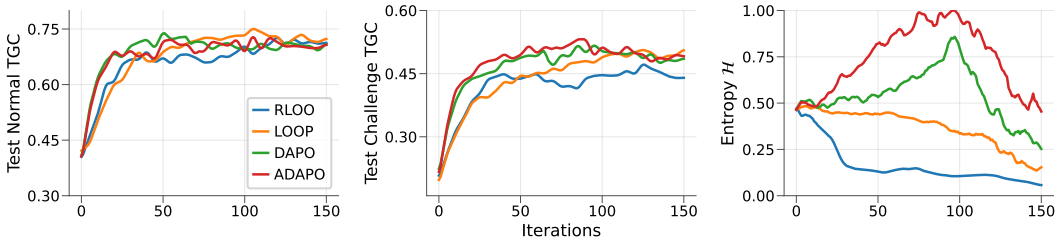

Figure 13: Qwen 2.5 32B test performance and token entropy on AppWorld vs. training iterations. Curves show mean across three independent seeds for each algorithm.

Algorithm	Test Normal	Best TN	Test Challenge	Best TC
RLOO	0.72	0.78	0.47	0.50
LOOP	0.75	0.78	0.50	0.54
DAPO	0.74	0.77	0.50	0.56
ADAPO	0.73	0.78	0.51	0.59

Table 7: Task-goal completion scores for AppWorld `Qwen-2.5-32B` by training algorithm. For each test split, we report the best average score across three seeds and the highest score among all seeds and training iterations.