DARE: DIFFICULTY-ADAPTIVE REINFORCEMENT LEARNING WITH CO-EVOLVED DIFFICULTY ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning improves the reasoning ability of large language models but remains costly and sample-inefficient, as many rollouts provide weak learning signals. Difficulty-aware data selection methods attempt to address this by prioritizing moderately difficult prompts, yet our analysis reveals three limitations: difficulty estimates become inaccurate under policy drift, data selection alone yields limited final-performance gains, and inference efficiency remains largely unchanged. These findings suggest that efficient and effective RL requires more than filtering by difficulty: the policy should learn to solve hard tasks while producing concise responses for easy ones. To this end, we propose DARE, a unified framework that co-evolves difficulty estimation with the policy via self-normalized importance sampling, maintains diverse difficulty coverage through a symmetric Beta sampling distribution, and applies tailored training strategies across difficulty tiers with adaptive compute allocation. Extensive experiments across multiple models and domains demonstrate that DARE improves training efficiency, final effectiveness, and inference efficiency over existing methods, producing more concise responses on easy tasks while improving correctness on hard ones. Code is available at https://anonymous.4open.science/r/rl_efficiency-43DB/.

1 INTRODUCTION

Reinforcement learning (RL) has emerged as a powerful paradigm for improving the reasoning capabilities of large language models (LLMs), with algorithms such as GRPO (DeepSeek-AI et al., 2025; Shao et al., 2024) driving substantial gains on mathematical, coding, and scientific benchmarks (Shao et al., 2024; Jain et al., 2025; Rein et al., 2023). However, existing RL methods remain highly resource-intensive and sample-inefficient (Yue et al., 2025; Tang et al., 2026; Sun et al., 2025), largely due to repeated rollout generation, in which many rollouts yield little useful gradient signal. For instance, in standard GRPO training, prompts that are too easy or too hard produce small group-relative advantages, resulting in weak learning signals and wasted rollout budgets (Bae et al., 2026; Li et al., 2025b; Zhang et al., 2025).

This inefficiency has motivated a growing body of work on *difficulty-aware data selection for RL*, which prioritizes prompts of moderate difficulty where the expected gradient signal is maximized (Yu et al., 2025; Sun et al., 2025; Zeng et al., 2026; Bae et al., 2026). Central to these approaches is the accuracy of the difficulty estimator. In this work, we first conduct a comprehensive analysis of existing difficulty estimation methods (Sun et al., 2025; Pang et al., 2026; Qu et al., 2026; Tabib & Deedar, 2025) for RL and identify three key limitations: ❶ *Inaccurate difficulty estimation under policy drift.* As the policy evolves during RL training, static or slowly adapting estimators become increasingly unreliable. Prompts deemed "moderately difficult" may in fact be trivially easy or intractably hard for the current policy, undermining the intended efficiency gains. ❷ *Limited final-performance gains from data selection alone.* Given a sufficient training budget, difficulty-aware data selection alone converges to similar final performance as standard training, with many hard tasks remaining unsolved (Zheng et al., 2025; Zhang et al., 2025; Zeng et al., 2026). ❸ *No improvement in inference efficiency.* Models trained with difficulty-based filtering still produce chain-of-thought (CoT) responses of similar length compared to those trained without filtering. These findings suggest that improving both the inference efficiency and effectiveness of RL requires more than selecting

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

moderately difficult data: the policy must also learn to solve hard tasks while preserving its ability to solve easy ones efficiently.

To address these limitations, we propose DARE (**D**ifficulty-**A**daptive **R**einforcement learning with co-**E**volved difficulty estimation), a unified framework that couples accurate, policy-aligned difficulty estimation with difficulty-adaptive training strategies for efficient and effective RL. First, we introduce a difficulty estimation and policy co-evolution mechanism: by combining self-normalized importance sampling with a prompt-wise First-In-First-Out (FIFO) replay buffer, our method tracks policy changes and maintains accurate difficulty estimates throughout training. We further propose a symmetric Beta sampling distribution that biases selection toward medium-difficulty samples while retaining easy and hard ones, reducing the risk of forgetting easy skills and continuing to expose the policy to hard prompts. Finally, we partition prompts into three difficulty tiers—easy, medium, and hard—and adaptively allocate compute and apply a tailored training strategy to each tier.

We evaluate DARE across multiple models and task domains. Results show consistent improvements in training efficiency (measured by steps and wall-clock time), final effectiveness, and inference efficiency (measured by average token usage) compared to existing difficulty-aware RL methods. Extensive ablation studies and additional analyses confirm that our method produces shorter and more concise answers on easy tasks while improving correctness on hard tasks. These findings demonstrate that DARE effectively improves both RL training and inference efficiency while adaptively distributing reasoning effort across difficulty levels, providing a promising direction for efficient and effective RL.

Overall, our contributions are fourfold:

① **Analysis of existing difficulty-aware data selection for RL.** We identify three empirical gaps in current methods (Section 2): inaccurate difficulty estimation under policy drift, limited final-performance gains from data selection alone, and no improvement in inference efficiency.

② **Accurate difficulty estimation via policy co-evolution.** We propose a difficulty estimation and policy co-evolution strategy that pairs self-normalized importance sampling with a Beta sampling distribution, yielding more accurate difficulty estimates and diverse difficulty-level data selection to improve training efficiency.

③ **Difficulty-adaptive training strategies.** We combine difficulty-aware data selection with tailored RL training strategies that encourage shorter and more concise solutions for easy tasks while increasing correctness and reasoning depth for hard tasks, enabling dynamic training and inference compute allocation across difficulty levels.

④ **Comprehensive empirical validation.** We conduct extensive experiments across diverse model architectures and tasks. Our method consistently achieves better training efficiency, inference efficiency, and final effectiveness compared to existing difficulty-aware RL methods. Detailed ablation studies further demonstrate the contribution of each component in our design.

2 PRELIMINARY INVESTIGATION

Accurate difficulty estimation is the foundation of the difficulty-aware RL method. In this section, we analyze the accuracy of existing difficulty estimation methods and their impact on training efficiency, inference efficiency, and final effectiveness when used for RL training.

Experimental Setting. We train Qwen2.5-Math-1.5B (Yang et al., 2024a) with GRPO under several difficulty-based online data selection methods, and evaluate on MATH-500 (Lightman et al., 2024). The training dataset composition and experimental setup follow Section 4.1, with full hyperparameters provided in Appendix F. All difficulty estimation methods target samples with predicted difficulty 0.5, so that differences in training behavior are attributable to the estimator rather than to the selection target. The one exception is *Entropy-Based Estimation*, whose entropy scores are not calibrated to the $[0, 1]$ difficulty scale and which therefore keeps the softmax-over-entropy selection rule of its source paper (Appendix F). We compare the following methods: (i) *Random Selection*: selects samples uniformly without difficulty consideration; (ii) *Embedding-Based Prediction* (Sun et al., 2025): computes difficulty based on embedding similarity of a prompt to a representative reference set; (iii) *Entropy-Based Estimation* (Pang et al., 2026): uses sequence-level entropy as a proxy for difficulty; (iv) *LLM-Judge* (Tabib & Deedar, 2025): uses the policy model itself as a difficulty judge; (v) *Bayesian Posterior Estimation* (Qu et al., 2026): estimates difficulty via Bayesian posterior

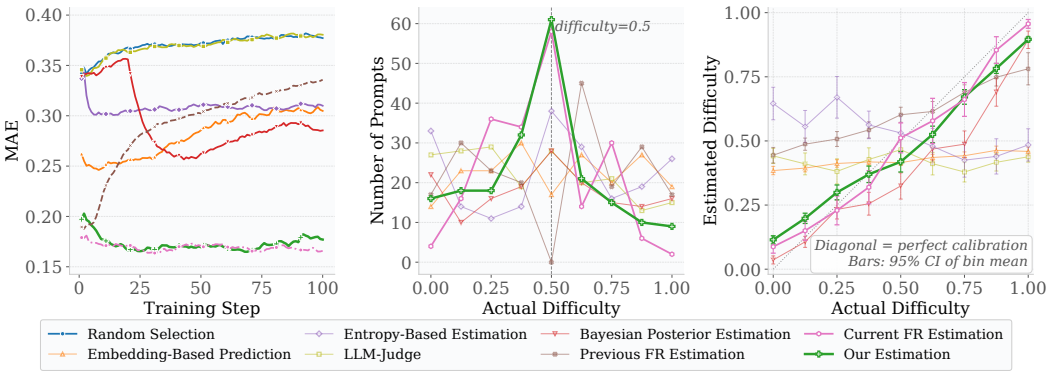


Figure 1: Difficulty estimation results for Qwen2.5-Math-1.5B during and after GRPO training under different difficulty-estimation methods. (Left) MAE between each estimator and *Current FR Estimation* across training steps. (Middle) Distribution of ground-truth difficulty for prompts assigned a predicted difficulty of 0.5 by each method. (Right) Calibration of estimated difficulty against ground-truth difficulty on MATH-500, with bin means and 95% confidence intervals. **Existing difficulty estimation methods are inaccurate during RL training.**

updates; (vi) *Previous Failure Rate Estimation (Previous FR Estimation)*: estimates difficulty as the failure rate from the latest previous rollouts, computed as one minus the average binary reward over the GRPO group of size G; (vii) *Current Failure Rate Estimation (Current FR Estimation)*: serves as the **ground-truth** difficulty reference, computed as the current policy's failure rate over G rollouts. (viii) *Our Estimation*: uses the replay buffer with policy-aligned importance-sampling correction indicated in Section 3.1. Detailed descriptions of all baselines are provided in Appendix E.

We evaluate estimation accuracy against *Current FR Estimation* using MSE and MAE, and estimate its intrinsic error from the discrepancy between two independent current-policy rollout runs. We analyze the ground-truth difficulty distribution of samples selected by each method at predicted difficulty 0.5, comparing predicted and actual difficulty. Training efficiency is measured by the steps and wall-clock time needed to reach matched performance, while inference efficiency is measured by average output tokens, both overall and stratified by the dataset's original difficulty labels.

Difficulty Estimation Analysis Under Policy Drift. The difficulty estimation accuracy and distribution results are shown in Figure 1 and Table 5. We make the following observations: ❶ **Existing difficulty estimation methods are inaccurate during RL training.** The intrinsic estimation error of *Current FR Estimation* is approximately 0.17, while the MAE of all existing methods exceeds 0.25—corresponding to an error of roughly 2 out of 8 rollouts, far above the intrinsic baseline. ❷ **Estimation error grows as training progresses.** The MAE of all existing baselines begins increasing after roughly 40 training steps, indicating that their difficulty estimates become less reliable as the policy drifts during RL training. ❸ **Estimated difficulty distributions deviate substantially from ground truth.** When filtering for samples with predicted difficulty 0.5, all existing methods retain many samples whose ground-truth difficulty is far from 0.5. Moreover, methods such as *Entropy-Based Estimation* and *LLM-Judge* systematically overestimate difficulty for easy prompts, rather than producing a monotonic mapping from true difficulty to estimated difficulty. These observations show that neither static estimators (e.g., *Previous FR Estimation*) nor existing policy-aware methods (e.g., *Entropy-Based Estimation*) produce reliable difficulty estimates throughout RL training, motivating us to seek an estimator that better tracks policy-dependent difficulty as training evolves.

Training and Inference Efficiency and Effectiveness Analysis. We further analyze RL training and inference performance when each estimation method selects only prompts with estimated difficulty 0.5. Training curves are shown in Figure 2 and inference results in Table 1 (comprehensive evaluation in Appendix G). We draw the following conclusions: ❶ **Difficulty filtration improves sample efficiency but yields limited final accuracy gains.** Most difficulty-aware methods converge faster than *Random Selection* GRPO in both training steps and wall-clock time, yet achieve only limited final performance improvements. This suggests that difficulty-aware selection primarily accelerates optimization; improving final accuracy requires training mechanisms beyond sample filtration

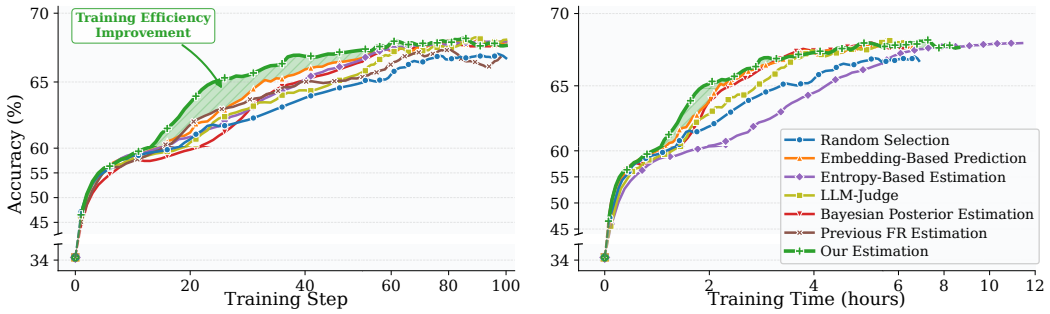


Figure 2: Training-step and wall-clock efficiency on MATH-500 with Qwen2.5-Math-1.5B trained by GRPO using different difficulty-estimation methods for data selection. **Data filtration improves training efficiency but yields only limited final-accuracy gains.**

Table 1: Average output token usage and accuracy by the original MATH-500 difficulty level for GRPO using Qwen2.5-Math-1.5B with different difficulty-estimation-based data selection methods. **Filtration alone does not improve inference efficiency.**

Method	Level 1 Tokens | Acc.	Level 2 Tokens | Acc.	Level 3 Tokens | Acc.	Level 4 Tokens | Acc.	Level 5 Tokens | Acc.	Overall Tokens | Acc.
Random Selection	357.3 | 88.4	441.5 | 80.0	539.4 | 77.1	726.2 | 66.4	811.6 | 45.5	626.9 | 67.4
Embedding-Based Pred.	349.5 | 90.7	451.5 | 81.1	583.4 | 80.0	706.0 | 71.1	876.7 | 44.8	649.5 | 69.4
Entropy-Based Est.	325.5 | 88.4	452.0 | 78.9	573.7 | 78.1	682.4 | 69.5	912.9 | 48.5	649.2 | 69.0
LLM-Judge	342.2 | 89.7	413.0 | 81.2	536.0 | 80.0	751.3 | 68.5	809.9 | 49.0	625.7 | 69.8
Bayesian Posterior Est.	339.2 | 91.2	449.8 | 81.4	587.4 | 80.2	714.5 | 70.9	815.2 | 45.6	634.8 | 69.7
Previous FR Est.	357.3 | 87.4	435.5 | 81.0	567.4 | 76.4	712.9 | 67.4	800.6 | 46.8	625.3 | 67.9
Our Estimation	359.4 | 90.7	438.3 | 82.2	545.7 | 82.9	679.6 | 71.1	873.0 | 43.3	632.3 | 69.8

alone. ❷ **More accurate difficulty estimation improves training-step efficiency.** *Embedding-Based Prediction* and *Bayesian Posterior Estimation* produce more accurate difficulty estimates than *LLM-Judge* and *Random Selection* (Figure 1), and exhibit correspondingly faster training-step convergence, confirming that estimation quality directly impacts training efficiency. ❸ **Filtration alone does not improve inference efficiency.** As shown in Table 1, difficulty-aware methods produce similar token usage to *Random Selection* GRPO across difficulty levels, despite differences in training efficiency. This is expected because filtration changes which prompts are sampled during training, but does not teach the model to allocate different reasoning lengths to different difficulty levels. These findings motivate difficulty-adaptive training strategies that go beyond data filtration to improve RL training efficiency, inference efficiency, and final effectiveness.

3 METHOD

The analysis in Section 2 shows that existing difficulty-aware RL suffers from inaccurate estimates under policy drift, while data filtration alone yields limited final-accuracy gains and does not improve inference efficiency. To address these issues, we propose DARE, a difficulty-adaptive RL framework that integrates policy-aligned difficulty estimation, dynamic data selection, and difficulty-adaptive policy optimization. At each training epoch, DARE estimates current prompt difficulty from a replay buffer with policy-aligned correction, samples prompts with a bias toward medium difficulty while retaining easy and hard prompts, and applies difficulty-specific compute budgets and reward shaping during policy optimization. Unlike filtration-only methods, which use difficulty estimates only for data selection, DARE adapts both the training data and learning signals as the policy evolves. An overview is shown in Figure 3, and pseudo-code is provided in Appendix C.

3.1 CO-EVOLVED DIFFICULTY ESTIMATION

Replay buffer and cold start. We maintain a FIFO replay buffer $\mathcal{B}$ to support difficulty estimation and sample reuse. For each prompt q, let K_q be the number of currently buffered rollouts; when q is

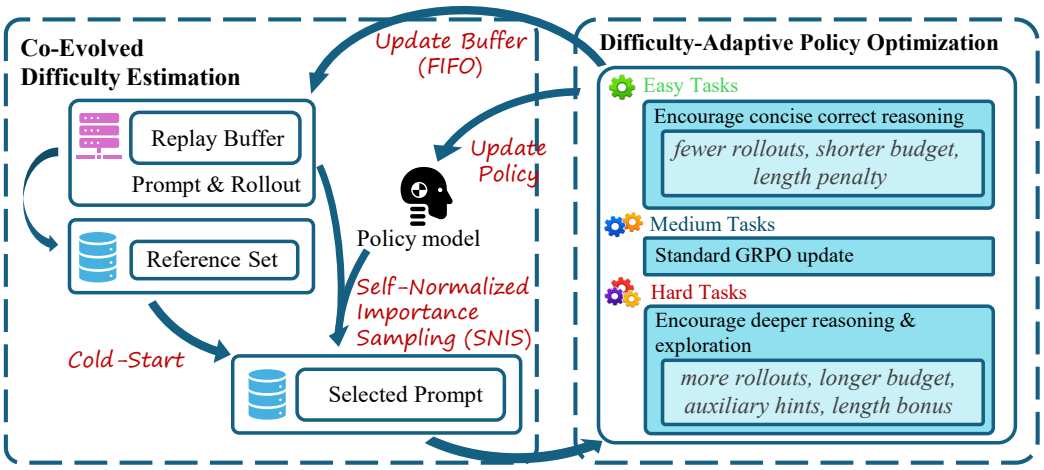


Figure 3: Overview of DARE. At each epoch, DARE updates a prompt-wise replay buffer, estimates current prompt difficulty with SNIS, and uses the estimate for dynamic prompt selection and difficulty-adaptive policy optimization.

fixed, we write $K = K_q$. The buffer stores

$$\mathcal{B}_q = \left\{ \left(o_k, r_k, \log \pi_{\theta_{\text{beh},k}}(o_k \mid q) \right) \right\}_{k=1}^{K_q}, \tag{1}$$

where o_k is a generated response, $r_k \in \{0, 1\}$ is the binary outcome reward, and $\pi_{\theta_{\text{beh},k}}$ is the behavior policy that generated o_k.

For prompts without buffered rollouts, we use an embedding-based cold start (Sun et al., 2025). We sample a small reference set $\mathcal{D}_{\text{ref}} = \{p_i\}_{i=1}^{N}$ from the training set $\mathcal{D}$, where $|\mathcal{D}_{\text{ref}}| \ll |\mathcal{D}|$. For each reference prompt p_i, we generate G rollouts and define its initial difficulty as the failure rate,

$$d_{p_i} = 1 - \bar{r}_{p_i} = \frac{1}{G} \sum_{k=1}^{G} (1 - r_k). \tag{2}$$

For an unseen prompt q, we compute a similarity-weighted difficulty estimate,

$$a_i = \frac{\exp(z_q^\top z_i / \sqrt{h})}{\sum_{j=1}^{N} \exp(z_q^\top z_j / \sqrt{h})}, \qquad \hat{d}_q = \sum_{i=1}^{N} a_i d_{p_i}, \tag{3}$$

where z_q and z_i are prompt embeddings and h is the embedding dimension. This provides an initial difficulty estimate before sufficient rollout history is available.

Policy-aligned replay estimation. Once a prompt has buffered rollouts, directly averaging historical rewards can be biased because those rollouts may come from older policies. We correct this policy drift with self-normalized importance sampling (SNIS). For each buffered rollout o_k, we compute the current-to-behavior log-ratio and clipped weight as

$$w_k = \exp\left(\text{clip}\left(\sum_{t=1}^{|o_k|} \left[\log \pi_\theta(o_{k,t} \mid q, o_{k,<t}) - \log \pi_{\theta_{\text{beh},k}}(o_{k,t} \mid q, o_{k,<t}) \right], -c, c \right) \right), \tag{4}$$

where clipping is applied for numerical stability. Here π_θ and $\pi_{\theta_{\text{beh},k}}$ denote the actual token-level rollout distributions after applying the decoding and stopping rules, rather than raw model softmax distributions. The current-policy difficulty estimate is then $\hat{d}_q = \frac{\sum_{k=1}^{K} w_k (1 - r_k)}{\sum_{k=1}^{K} w_k}$. This estimator tracks policy evolution by upweighting rollouts likely under the current policy and downweighting stale ones. In practice, it avoids the cost of re-rolling every prompt at each step, as required by *Current FR Estimation*, while keeping replay statistics policy-aligned and achieving comparable estimation accuracy, as shown in Section 2.

We further give an idealized finite-sample decomposition for the SNIS estimate. Fix a prompt q, an estimation epoch, and its K buffered rollouts. Let $Y(o) = 1 - r(o)$, $d_\theta(q) = \mathbb{E}_{O \sim \pi_\theta(\cdot|q)}[Y(O)]$,

and $\rho_k(o) = \frac{d\pi_\theta(\cdot|q)}{d\mu_k(\cdot|q)}(o)$ for $O_k \sim \mu_k(\cdot \mid q)$, where π_θ and μ_k denote the actual current and behavior rollout distributions, including decoding and stopping. Let W_k be the clipped weight with $0 \le W_k(o) \le C_w$; for Equation 4, $C_w = e^c$.

Proposition 3.1 (Finite-sample error of clipped SNIS difficulty estimation, the Proof is detailed in Appendix B)**.** *Suppose $O_1, \ldots, O_K$ are conditionally independent given the fixed epoch and current policy, $\pi_\theta(\cdot \mid q) \ll \mu_k(\cdot \mid q)$ for all k, and $K^{-1}\sum_k \mathbb{E}_k[W_k(O)] \ge b_{\min} > 0$, where $\mathbb{E}_k$ denotes expectation under $O \sim \mu_k(\cdot \mid q)$. For $\hat{d}_q = (K^{-1}\sum_k W_k(O_k)Y(O_k))/(K^{-1}\sum_k W_k(O_k))$ and $\varepsilon_\delta = C_w\sqrt{\log(4/\delta)/(2K)} < b_{\min}$, with probability at least $1-\delta$,*

$$\left|\hat{d}_q - d_\theta(q)\right| \le \frac{2\varepsilon_\delta}{b_{\min} - \varepsilon_\delta} + \frac{1}{b_{\min}K}\sum_{k=1}^{K}\mathbb{E}_k[|\rho_k(O) - W_k(O)|]\,. \tag{5}$$

For Equation 4, the second term equals $(b_{\min}K)^{-1}\sum_k \mathbb{E}_k[(\rho_k(O) - e^c)_+ + (e^{-c} - \rho_k(O))_+]$. The worst-case concentration term can be loose for large c, so we use ESS (see Appendix C) in practice.

3.2 Difficulty-Guided Dynamic Data Selection

Given the estimated difficulty $\hat{d}_q$, DARE samples prompts for each training batch. Since $\hat{d}_q$ estimates the failure rate, the expected binary reward is approximately $1 - \hat{d}_q$, with variance $\hat{d}_q(1-\hat{d}_q)$. This variance is maximized at $\hat{d}_q = 0.5$, where group-relative advantages provide the strongest learning signal. We therefore bias sampling toward medium-difficulty prompts using a symmetric Beta density,

$$p_{\text{DARE}}(q) \propto \text{Beta}(\hat{d}_q; \alpha, \alpha), \alpha = 1 + \kappa/2, \tag{6}$$

where κ controls the concentration around $\hat{d}_q = 0.5$.

Unlike hard-threshold filtering that retains only prompts near a fixed difficulty value (Sun et al., 2025), Equation 6 assigns nonzero probability across the full difficulty range, so no prompt is permanently excluded from training (Parashar et al., 2026; Kordi et al., 2026; Yang et al., 2024b). Within each batch, prompts are sampled without replacement according to $p_{\text{DARE}}(q)$.

3.3 Difficulty-Adaptive Policy Optimization

Data selection determines *which* prompts are trained; difficulty-adaptive policy optimization determines *how* each is trained. Easy prompts are trained to produce concise correct reasoning, medium prompts receive standard GRPO updates, and hard prompts are allocated additional exploration compute to obtain useful trajectories. Prompts are partitioned into these three tiers by two thresholds d_{easy} and d_{hard}, using whichever difficulty signal is available when the decision is made. Rollout allocation precedes generation and therefore uses the estimate $\hat{d}_q$ of Section 3.1. After the group is scored, reward shaping uses the realized failure rate without an estimator:

$$\tilde{d}_q = 1 - \bar{r}_q = \frac{1}{G_q}\sum_{i=1}^{G_q}(1 - r_i). \tag{7}$$

We optimize the policy with the following difficulty-conditioned objective:

$$\mathcal{J}_{\text{DARE}}(\theta) = \mathbb{E}_{q\sim p_{\text{DARE}},\{o_i\}_{i=1}^{G_q}\sim\pi_{\theta_{\text{beh}}}(\cdot|q)}\left[\frac{1}{G_q}\sum_{i=1}^{G_q}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(\min\left(\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{beh}}}(o_{i,t} \mid q, o_{i,<t})}A_i,\right.\right.\right.$$
$$\left.\left.\left.\text{clip}\left(\frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{beh}}}(o_{i,t} \mid q, o_{i,<t})}, 1-\epsilon_q^-, 1+\epsilon_q^+\right)A_i\right) - \beta\,\text{D}_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\right)\right]. \tag{8}$$

Here, p_{DARE} is the sampling distribution from Equation 6, G_q is the rollout group size for prompt q, A_i is the group-relative advantage computed from the shaped reward $\tilde{r}_i$, and $\epsilon_q^-, \epsilon_q^+$ are difficulty-conditioned clipping ranges. Each batch mixes a fraction σ of fresh on-policy rollouts with $1-\sigma$ replay buffer trajectories, where $\pi_{\theta_{\text{beh}}}$ denotes the behavior policy that generated replayed trajectories.

Concise reasoning for easy prompts. Easy prompts typically yield uniformly correct rollouts with little advantage variation. When $\hat{d}_q < d_{\text{easy}}$, DARE reduces the rollout count to $G_q = G_{\text{easy}} < G$ and imposes a shorter response budget; when $\tilde{d}_q < d_{\text{easy}}$, it penalizes length on correct rollouts to encourage concise solutions. The easiness weight $w_q^{\text{easy}} = \text{clamp}\big((d_{\text{easy}} - \tilde{d}_q)/d_{\text{easy}},\, 0,\, 1\big)$ grows the more reliably the policy solves the prompt, and the shaped reward is

$$\tilde{r}_i = r_i - \mathbf{1}_{\{\tilde{d}_q < d_{\text{easy}}\}} \mathbf{1}_{\{r_i = 1\}}\, \lambda_{\text{easy}}\, w_q^{\text{easy}}\, \frac{|o_i|}{L_q^{\max}}, \tag{9}$$

where λ_{easy} is the penalty coefficient and $L_q^{\max} = \max_{i \le G_q} |o_i|$ is the longest rollout for prompt q, so that the length term is normalized to $[0, 1]$ per group. This is distinct from the global decoding cap $T_{\max}$ of Section 4.1. This gives higher advantages to shorter correct solutions while leaving incorrect rollouts unchanged, teaching the policy to save inference tokens on prompts it already solves reliably. We also relax the upper clipping bound ($\epsilon_q^+ > \epsilon$) while keeping $\epsilon_q^- = \epsilon$, allowing stronger positive updates toward concise correct solutions without amplifying harmful updates.

Standard updates for medium prompts. Medium prompts use the base configuration: $G_q = G$ when $\hat{d}_q$ lies in the medium band, and $\tilde{r}_i = r_i$, $\epsilon_q^- = \epsilon_q^+ = \epsilon$ when $\tilde{d}_q$ does. Their rollout groups typically contain both correct and incorrect responses, providing sufficient advantage variation without additional shaping. The Beta sampler in Section 3.2 ensures that medium prompts appear most frequently in each batch, making this tier the backbone of optimization.

Exploration incentive for hard prompts. Hard prompts often produce rollout groups in which every response is incorrect, collapsing the group-relative advantage. When $\hat{d}_q > d_{\text{hard}}$, DARE allocates more rollouts ($G_q = G_{\text{hard}} > G$) to increase the probability of obtaining at least one correct trajectory: G standard rollouts alongside $G_q - G$ hint-augmented ones, where the hint is a successful historical trajectory retrieved from the replay buffer (if none exists, all G_q rollouts use the standard prompt). When $\tilde{d}_q > d_{\text{hard}}$, a bounded length bonus is added to the incorrect rollouts to further support exploration. Analogously to the easy case, we define a hardness weight $w_q^{\text{hard}} = \text{clamp}\big((\tilde{d}_q - d_{\text{hard}})/(1 - d_{\text{hard}}),\, 0,\, 1\big)$ and shape the reward as

$$\tilde{r}_i = r_i + \mathbf{1}_{\{\tilde{d}_q > d_{\text{hard}}\}} \mathbf{1}_{\{r_i = 0\}}\, \lambda_{\text{hard}}\, w_q^{\text{hard}}\, \frac{|o_i|}{L_q^{\max}}. \tag{10}$$

This bonus encourages the policy to invest more reasoning effort on hard prompts rather than giving up early. Since $\lambda_{\text{hard}} < 1$, any correct rollout still receives a higher reward than an incorrect one with the bonus, ensuring that correctness remains the dominant training signal.

Together, the easy-prompt penalty and hard-prompt bonus create a difficulty-adaptive length incentive: the policy learns to be concise on prompts it solves reliably and to reason more deeply on prompts where it struggles. This behavior cannot arise from data filtering alone, which changes *which* prompts are trained but does not shape *how much* reasoning effort each difficulty level receives.

4 EXPERIMENTS

We evaluate DARE along the three axes established in Section 2: training efficiency, final accuracy, and inference-time token usage. Section 4.2 reports the main results across five benchmarks, while Section 4.3 analyzes the contribution of individual designs and the effects of hyperparameter tuning.

4.1 EXPERIMENTAL SETUP

Models and training data. We evaluate on three model scales: Qwen2.5-Math-1.5B, Qwen2.5-Math-7B (Yang et al., 2024a), and SmolLM3-3B-Base (Bakouch et al., 2025), covering math-specialized and general-purpose architectures. All models are trained with GRPO (Shao et al., 2024) on the same union of MATH Level 3–5 (Hendrycks et al., 2021b), DeepScaleR-40K (Luo et al., 2025), Open-Reasoner-Zero-57K (Hu et al., 2025a), and DeepMath-103K (He et al., 2025) (35,512 prompts). All prompts share a uniform system prompt, with difficulty-aware hints appended to the user content for hard prompts; prompt templates, and dataset details are provided in Appendix D and Appendix F.

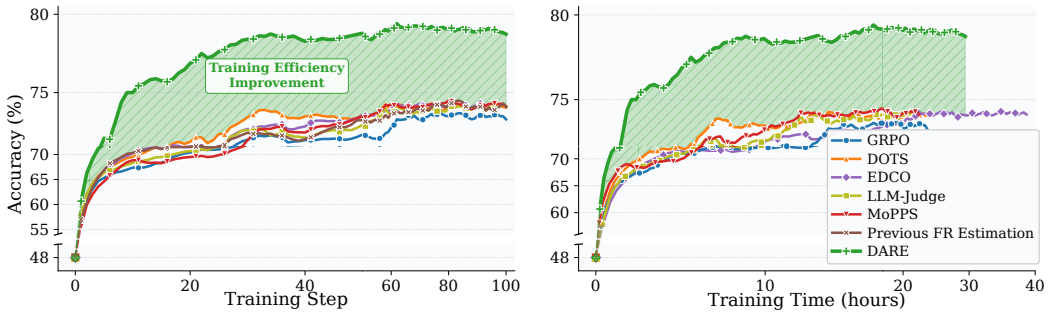


Figure 4: Training performance on MATH-500 against training steps (left) and wall-clock time (right) with Qwen2.5-Math-7B under various difficulty-aware RL methods.

Method	GSM8K	AIME-AMC	Minerva	Olympiad
GRPO	77.63	45.78	13.24	22.55
DOTS	79.38	45.78	15.44	23.74
EDCO	80.59	46.99	13.60	24.63
LLM-Judge	79.08	42.17	13.24	24.04
MoPPS	82.87	46.99	16.54	21.96
Prev. FR Est.	76.50	44.58	13.24	23.15
DARE	**84.53**	**50.60**	**19.85**	**26.41**

Table 2: Accuracy (%) with different difficulty-aware RL methods on Qwen2.5-Math-1.5B.

Level	Acc (%) GRPO	Acc (%) DARE	Tok GRPO	Tok DARE
1	88.37	**88.37** 0.00	357	**341** ▲ 4%
2	80.00	**84.44** ▲ 4.44	442	**400** ▲ 10%
3	77.14	**84.76** ▲ 7.62	539	**483** ▲ 10%
4	66.41	**71.09** ▲ 4.68	726	**648** ▲ 11%
5	45.52	**53.73** ▲ 8.21	812	867 ▼ 7%
Overall	67.40	**73.20** ▲ 5.80	627	**601** ▲ 4%

Table 3: MATH-500 accuracy (%) and average output tokens by difficulty level on Qwen2.5-Math-1.5B. Arrows indicate changes versus *GRPO*.

Implementation details. A training step is one policy update on a sampled batch. Unless otherwise stated, we use the AdamW optimizer, base rollout count G=8, maximum response length $T_{\max}$=3072 tokens, sampling temperature 0.6, and top-p 0.95. For DARE, the cold-start reference size is N=256, the SNIS log-ratio clip is c=4.0, and the Beta concentration is κ=100 in Equation 6. Prompts are stratified at $(d_{\text{easy}}, d_{\text{hard}})$=(0.3, 0.8) with tiered rollout counts $(G_{\text{easy}}, G_{\text{hard}})$=(4, 16), reward-shaping coefficients λ_{easy}=λ_{hard}=1×10^{-4}, and symmetric clipping ϵ=0.2 on medium and hard prompts; for easy prompts we use the relaxed upper bound selected by the ablation in Table 9. The per-prompt rollout history size K and replay buffer capacity are reported in Appendix F.

Baselines. We compare against *GRPO* plus five difficulty-aware baselines. Unless otherwise specified, each difficulty-aware baseline follows the training pipeline of its source paper, unlike in Section 2, where all baselines use the same data-sampling rule. (i) *GRPO* (Shao et al., 2024): uniform prompt sampling without difficulty-aware selection. (ii) *DOTS* (Sun et al., 2025): Embedding-Based Prediction with rollout replay. (iii) *EDCO* (Pang et al., 2026): Entropy-Based Estimation with dynamic curriculum selection. (iv) *MoPPS* (Qu et al., 2026): Bayesian Posterior Estimation with Thompson-sampling prompt selection. (v) *LLM-Judge* and (vi) *Previous FR Estimation* reuse the protocol of Section 2, where GRPO is performed on prompts with estimated difficulty 0.5.

Evaluation. We evaluate on five math benchmarks: MATH-500 (Lightman et al., 2024), GSM8K (Cobbe et al., 2021), AIME-AMC (AI-MO, 2024a;b), MinervaMath (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). For the reward-shaping and clipping ablations (Table 8, Table 9), we also report *AIME24/25* (Mathematical Association of America, 2025), a separate 60-problem set combining AIME 2024 and AIME 2025. To test whether the gains transfer beyond mathematics, we also evaluate on HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), and LiveCodeBench (Jain et al., 2025), with the corresponding training data and protocol detailed in Appendix I. Unless stated otherwise, all results use greedy decoding.

4.2 Main Results

DARE's difficulty estimation is the most accurate among existing methods. As shown in Figure 1 and Table 5, our policy-aligned SNIS estimator achieves the lowest error among all difficulty estimation methods during and after RL training, approaching the intrinsic error of *Current FR Estimation*. This result shows that SNIS can effectively correct policy drift in difficulty estimation.

Table 4: Ablation of difficulty-adaptive training components on Qwen2.5-Math-7B across four benchmarks. All rows below *GRPO* include the SNIS estimator; each subsequent row adds one further component on top of *GRPO* + *SNIS*. The bottom row DARE combines all components.

Method	MATH-500	GSM8K	AIME-AMC	Minerva	Overall
GRPO	72.4	86.8	50.6	14.7	56.1
+ *SNIS*	75.8 ▲3.4	88.7 ▲1.9	51.8 ▲1.2	16.2 ▲1.5	58.1 ▲2.0
+ *tiered rollout allocation*	77.0 ▲4.6	91.0 ▲4.2	51.8 ▲1.2	16.9 ▲2.2	59.2 ▲3.1
+ *easy penalty*	74.8 ▲2.4	**93.0** ▲6.2	51.8 ▲1.2	14.3 ▼0.4	58.5 ▲2.4
+ *easy penalty + hard bonus*	77.0 ▲4.6	90.2 ▲3.4	51.8 ▲1.2	19.1 ▲4.4	59.5 ▲3.4
DARE (all components)	**78.2** ▲5.8	90.2 ▲3.4	**55.4** ▲4.8	**19.9** ▲5.2	**60.9** ▲4.8

DARE's unified difficulty-adaptive RL framework improves both training efficiency and effectiveness. Figure 4 shows MATH-500 accuracy against training steps and wall-clock time with Qwen2.5-Math-7B. DARE reaches the highest accuracy in the fewest steps and shortest time among difficulty-aware methods. On early-saturating GSM8K, *MoPPS* and *EDCO* reach the 88% target sooner (Appendix J). Unlike filtration-only training, whose methods converge to similar final accuracy (Figure 2), these gains reflect both difficulty-guided data selection and difficulty-adaptive policy optimization. On SmolLM3-3B-Base, DARE also leads in MATH-500 training efficiency and final accuracy (Appendix H.5); on Qwen2.5-Math-1.5B, it achieves the best final accuracy across four reasoning benchmarks (Table 2). The estimator-level gain transfers to LiveCodeBench, HumanEval, and MBPP (Appendix I). DARE uses less average GPU memory than *GRPO* (25.75 vs. 26.62 GB) with essentially unchanged peak usage (50.94 vs. 50.46 GB); its replay buffer increases CPU memory, as for other buffer-based baselines (Table 6).

DARE improves inference efficiency and effectiveness simultaneously. As shown in Table 3, compared with the GRPO baseline, DARE improves both overall inference efficiency and accuracy. The token savings hold across Levels 1–4 (up to 11%) with accuracy maintained or improved at every level, while Level 5 is the only level where DARE spends more tokens (+7%) and it is also where the accuracy gain is largest (+8.21). This is the intended difficulty-adaptive trade-off: reasoning effort is withdrawn from prompts the policy already handles and redirected to the hardest ones. A case study in Section H.2 contrasts shorter and longer correct responses to an easy MATH-500 question. These results show that reward shaping and dynamic compute allocation help the policy allocate reasoning effort more efficiently across difficulty levels.

4.3 Ablation Studies and Additional Investigations

DARE component-design ablation. We ablate DARE's main components on Qwen2.5-Math-7B and Qwen2.5-Math-1.5B, with the memory-guided hint studied on the latter. As shown in Table 4 and Table 10, we observe that ❶ *SNIS* alone accounts for a substantial share of the total gain over *GRPO*, and each further component adds a smaller marginal improvement on top of it; and ❷ DARE achieves the best overall performance when all components are combined, indicating that these designs provide complementary gains rather than being redundant.

Additional investigations and hyperparameter tuning. Table 8 reports the sweep over the reward-shaping coefficients λ_{easy} and λ_{hard}, while Table 9 reports the sweep over the clipping parameters ϵ_q^- and ϵ_q^+. The results show that ❶ the two coefficients trade accuracy against tokens in opposite directions, so we adopt the balanced setting $\lambda_{\text{easy}} = \lambda_{\text{hard}} = 1 \times 10^{-4}$, which gives the strongest Minerva and OlympiadBench accuracy; it is not uniformly best, as the 60-problem AIME24/25 split favors the no-shaping configuration. ❷ Relaxing only the upper bound $1 + \epsilon_q^+$ to 1.6 while keeping $1 - \epsilon_q^-$ tight yields the best overall accuracy.

5 Conclusion

We presented DARE, a difficulty-adaptive RL framework for efficient LLM reasoning. DARE addresses the limits of filtration-only methods by combining policy-aligned difficulty estimation, dynamic data selection, and difficulty-specific policy optimization. Across multiple models and benchmarks, DARE improves training efficiency, final accuracy, and inference-token efficiency, showing that effective RL should adapt both data selection and training signals to prompt difficulty.

AI USE STATEMENT

Generative AI was used to improve manuscript readability and edit existing code after the authors determined the implementation approach. The authors reviewed the edits, tested the code, and take responsibility for the final paper and artifacts.

REFERENCES

AI-MO. AI-MO validation AIME dataset. Hugging Face Datasets, 2024a. URL https://huggingface.co/datasets/AI-MO/aimo-validation-aime.

AI-MO. AI-MO validation AMC dataset. Hugging Face Datasets, 2024b. URL https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2026. URL https://arxiv.org/abs/2504.03380.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, and Thomas Wolf. SmolLM3: Smol, multilingual, long-context reasoner. Hugging Face Blog, 2025. URL https://huggingface.co/blog/smollm3.

Chengkun Cai, Xu Zhao, Haoliang Liu, Zhongyu Jiang, Tianfang Zhang, Zongkai Wu, Jenq-Neng Hwang, and Lei Li. The role of deductive and inductive reasoning in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16780–16790, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for LLM reasoning. *arXiv preprint arXiv:2505.14970*, 2025. URL https://arxiv.org/abs/2505.14970.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Huilin Deng, Ding Zou, Rui Ma, Hongchen Luo, Yang Cao, and Yu Kang. Boosting the generalization and reasoning of vision language models with curriculum reinforcement learning. *arXiv preprint arXiv:2503.07065*, 2025. URL https://arxiv.org/abs/2503.07065.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://aclanthology.org/2024.acl-long.211/.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. DeepMath-103K: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2021a. URL https://arxiv.org/abs/2105.09938.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2021b. URL https://arxiv.org/abs/2103.03874.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-Reasoner-Zero: An Open Source Approach to Scaling up Reinforcement Learning on the Base Model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a.

Zengjie Hu, Jiantao Qiu, Tianyi Bai, Haojin Yang, Binhang Yuan, Qi Jing, Conghui He, and Wentao Zhang. VADE: Variance-aware dynamic sampling via online sample-level difficulty estimation for multimodal RL. *arXiv preprint arXiv:2511.18902*, 2025b. URL https://arxiv.org/abs/2511.18902.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, An Yang, Rui Men, Fei Huang, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024. URL https://arxiv.org/abs/2409.12186.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida I. Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination-free evaluation of large language models for code. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2403.07974.

Can Jin, Yang Zhou, Qixin Zhang, Hongwu Peng, Di Zhang, Zihan Dong, Marco Pavone, Ligong Han, Zhang-Wei Hong, Tong Che, and Dimitris N Metaxas. AIRL-S: Unifying reinforcement learning and search-based test-time scaling via adversarial inverse reinforcement learning. *arXiv preprint arXiv:2508.14313*, 2025.

Can Jin, Rui Wu, Tong Che, Qixin Zhang, Hongwu Peng, Jiahui Zhao, Zhenting Wang, Wenqi Wei, Ligong Han, Zhao Zhang, et al. Reasoning over precedents alongside statutes: Case-augmented deliberative alignment for llm safety. *arXiv preprint arXiv:2601.08000*, 2026.

Yeganeh Kordi, Nihal V Nayak, Max Zuo, Ilana Nguyen, and Stephen Bach. Revisiting generalization across difficulty levels: It's not so easy. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7014–7042, 2026.

Sunbowen Lee, Qingyu Yin, Chak Tou Leong, Jialiang Zhang, Yicheng Gong, Shiwen Ni, Min Yang, and Xiaoyu Shen. Probing the difficulty perception mechanism of large language models. *arXiv preprint arXiv:2510.05969*, 2025. URL https://arxiv.org/abs/2510.05969.

Xuanyu Lei, Chenliang Li, Yuning Wu, Kaiming Liu, Weizhou Shen, Peng Li, Ming Yan, Fei Huang, Ya-Qin Zhang, and Yang Liu. Writing-RL: Advancing long-form writing via adaptive curriculum reinforcement learning. *arXiv preprint arXiv:2506.05760*, 2025. URL https://arxiv.org/abs/2506.05760.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023. URL https://arxiv.org/abs/2312.14852.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. LIMR: Less is more for RL scaling. *arXiv preprint arXiv:2502.11886*, 2025a. URL https://arxiv.org/abs/2502.11886.

Ziniu Li, Congliang Chen, Tianyun Yang, Tian Ding, Ruoyu Sun, Ge Zhang, Wenhao Huang, and Zhi-Quan Luo. Knapsack RL: Unlocking exploration of LLMs via optimizing budget allocation. *arXiv preprint arXiv:2509.25849*, 2025b. URL https://arxiv.org/abs/2509.25849.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

Shicheng Liu and Minghui Zhu. Leveraging explanation to improve generalization of meta reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Xiang Liu, Xuming Hu, Xiaowen Chu, and Eunsol Choi. DiffAdapt: Difficulty-adaptive reasoning for token-efficient LLM inference. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2510.19669.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. Notion Blog, 2025. URL https://notion.so/19681902c1468005bed8ca303013a4e2.

Mathematical Association of America. AIME 2024 and 2025 problems. American Invitational Mathematics Examination, 2025. URL https://maa.org/maa-invitational-competitions/.

Jing-Cheng Pang, Liu Sun, Chang Zhou, Xian Tang, Haichuan Ma, Kun Jiang, Jianlong Wang, Kai Zhang, Sijie Wu, Haoran Cai, Chenwei Wu, Xubin Li, and Xin Chen. EDCO: Dynamic curriculum orchestration for domain-specific large language model fine-tuning. *arXiv preprint arXiv:2601.03725*, 2026. URL https://arxiv.org/abs/2601.03725.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and Shuiwang Ji. Curriculum reinforcement learning from easy to hard tasks improves LLM reasoning. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2506.06632.

Yun Qu, Qi Wang, Yixiu Mao, Vincent Tao Hu, Björn Ommer, and Xiangyang Ji. Can prompt difficulty be online predicted for accelerating RL finetuning of reasoning models? In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2026. URL https://arxiv.org/abs/2507.04632.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023. URL https://arxiv.org/abs/2311.12022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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.

Qianli Shen, Daoyuan Chen, Yilun Huang, Zhenqing Ling, Yaliang Li, Bolin Ding, and Jingren Zhou. BOTS: A unified framework for Bayesian online task selection in LLM reinforcement finetuning. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2510.26374.

Jingzhe Shi, Qinwei Ma, Hongyi Liu, Hang Zhao, Jeng-Neng Hwang, and Lei Li. Intrinsic entropy of context length scaling in llms. In *The Fourteenth International Conference on Learning Representations*, 2026.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. Efficient reinforcement finetuning via adaptive curriculum learning. *arXiv preprint arXiv:2504.05520*, 2025. URL https://arxiv.org/abs/2504.05520.

Yifan Sun, Jingyan Shen, Yibin Wang, Tianyu Chen, Zhendong Wang, Mingyuan Zhou, and Huan Zhang. Improving data efficiency for LLM reinforcement fine-tuning through difficulty-targeted online data selection and rollout replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2506.05316.

HM Tabib and Jaber Ahmed Deedar. Toward trustworthy difficulty assessments: Large language models as judges in programming and synthetic tasks. *arXiv preprint arXiv:2511.18597*, 2025.

Xinyu Tang, Zhenduo Zhang, Yurou Liu, Wayne Xin Zhao, Zujie Wen, Zhiqiang Zhang, and Jun Zhou. Towards high data efficiency in reinforcement learning with verifiable reward. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2509.01321.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, Haosheng Zou, Yongchao Deng, Shousheng Jia, and Xiangzheng Zhang. Light-R1: Curriculum SFT, DPO and RL for long COT from scratch and beyond. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL): Industry Track*, 2025. doi: 10.18653/v1/2025.acl-industry.24. URL https://arxiv.org/abs/2503.10460.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024a.

Zhe Yang, Yichang Zhang, Tianyu Liu, Jian Yang, Junyang Lin, Chang Zhou, and Zhifang Sui. Can large language models always solve easy problems if they can solve harder ones? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1531–1555, 2024b.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. LIMO: Less is more for reasoning. In *Conference on Language Modeling (COLM)*, 2025. URL https://arxiv.org/abs/2502.03387.

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.

Xiangchi Yuan, Xiang Chen, Tong Yu, Dachuan Shi, Can Jin, Wenke Lee, and Saayan Mitra. Mitigating forgetting between supervised and reinforcement learning yields stronger reasoners. *arXiv preprint arXiv:2510.04454*, 2025.

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.

Yongcheng Zeng, Zexu Sun, Bokai Ji, Erxue Min, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Haifeng Zhang, Xu Chen, and Jun Wang. CurES: From gradient analysis to efficient curriculum learning for reasoning LLMs. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2510.01037.

Ruiqi Zhang, Daman Arora, Song Mei, and Andrea Zanette. SPEED-RL: Faster training of reasoning models via online curriculum learning. In *ICML 2025 Workshop on AI for Math (AI4Math@ICML25)*, 2025. URL https://arxiv.org/abs/2506.09016.

Yang Zhao, Kai Xiong, Xiao Ding, Li Du, Yangou Ouyang, Zhouhao Sun, Jiannan Guan, Wenbin Zhang, Bin Liu, Dong Hu, Bing Qin, and Ting Liu. UFO-RL: Uncertainty-focused optimization for efficient reinforcement learning data selection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2505.12457.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2506.02177.

Yang Zhou, Mingyu Zhao, Zhenting Wang, Difei Gu, Bangwei Guo, Ruosong Ye, Ligong Han, Can Jin, and Dimitris N Metaxas. M^3-Bench: Multi-modal, multi-hop, multi-threaded tool-using MLLM agent benchmark. *arXiv preprint arXiv:2511.17729*, 2025a.

Yang Zhou, Shiyu Zhao, Yuxiao Chen, Zhenting Wang, Can Jin, and Dimitris N Metaxas. LED: LLM enhanced open-vocabulary object detection without human curated data generation. *arXiv preprint arXiv:2503.13794*, 2025b.

Yubo Zhu, Dongrui Liu, Zecheng Lin, Wei Tong, Sheng Zhong, and Jing Shao. The LLM already knows: Estimating LLM-perceived question difficulty via hidden representations. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1160–1176, 2025. doi: 10.18653/v1/2025.emnlp-main.61. URL https://arxiv.org/abs/2509.12886.

APPENDIX CONTENT

A Related Works 16
B Proof of Proposition 3.1 16
C Full Algorithm 18
D Prompt Templates 18
E Difficulty Estimation Baselines 20
F Experimental Details 23
G Extended Preliminary Investigation 24
H Ablation Studies and Additional Investigations 24
H.1 Effective Sample Size of the SNIS Estimator 24
H.2 Case Study: Stratified Generation Behavior 25
H.3 Entropy–Difficulty Relationship Across Models 25
H.4 Ablation Results on Qwen2.5-Math-1.5B 25
H.5 Additional Results on SmolLM3-3B 26
I Generalization to Code Generation 27
J Training Efficiency Analysis 28
K Accuracy Under Constrained Rollout Budgets 29
L Impact Statement 30

A Related Works

Reinforcement Learning for LLM Reasoning. With the widespread application of LLMs across various tasks Zhou et al. (2025b;a), reinforcement learning has become a central approach for improving the reasoning capabilities of large language models. PPO Schulman et al. (2017) and GRPO Shao et al. (2024) are among the most widely adopted algorithms, with GRPO eliminating the need for a separate value network by using group-relative advantages. These algorithms have driven significant progress on mathematical reasoning Shao et al. (2024); DeepSeek-AI et al. (2025); Cai et al. (2025); Jin et al. (2026); Yuan et al. (2025), code generation Jain et al. (2025); Jin et al. (2025), and scientific question answering Rein et al. (2023). However, RL training for LLMs remains costly and sample inefficient (Yue et al., 2025; Tang et al., 2026; Liu & Zhu, 2026), since many generated rollouts produce near-zero advantages and contribute little to policy improvement. This inefficiency has motivated research into more effective data selection and compute allocation strategies.

Online Data Selection. A growing body of work shows that uniform prompt sampling wastes substantial compute on prompts that are either trivially easy or intractably hard for the current policy, both of which yield collapsed advantage signals Bae et al. (2026); Li et al. (2025b); Zhang et al. (2025). The prevailing strategy is to prioritize prompts of moderate difficulty, where the training signal is maximized Yu et al. (2025); Sun et al. (2025); Zeng et al. (2026); Bae et al. (2026); Zhang et al. (2025). Several methods further organize difficulty progression across training stages Chen et al. (2025); Parashar et al. (2026); Shi et al. (2025), with extensions to code, multimodal, and writing tasks Deng et al. (2025); Lei et al. (2025); Wen et al. (2025). Complementary work demonstrates that sample informativeness matters more than dataset scale Li et al. (2025a); Ye et al. (2025), while theoretical analyses justify moderate-difficulty selection from the perspectives of advantage variance and gradient signal-to-noise ratio Bae et al. (2026); Zeng et al. (2026); Li et al. (2025b). Among these, DOTS Sun et al. (2025) is a particularly relevant baseline that combines an embedding-based difficulty predictor with rollout replay. However, prior studies also show that training exclusively on either easy or hard data does not consistently improve performance across the full range of task difficulties Parashar et al. (2026); Kordi et al. (2026); Yang et al. (2024b), suggesting that focusing solely on moderate-difficulty prompts may be similarly insufficient. DARE goes beyond moderate-difficulty filtering by introducing co-evolved difficulty estimation that maintains accuracy under policy drift, and difficulty-adaptive training strategies that apply tailored objectives across easy, medium, and hard prompts to jointly improve inference efficiency and final effectiveness.

Difficulty Estimation. Accurate difficulty estimation is central to effective data selection, yet existing methods face a fundamental trade-off between cost and accuracy. Embedding-based predictors Sun et al. (2025) estimate difficulty via similarity to a fixed prompt space but cannot adapt as the policy evolves. Lightweight proxies based on entropy, perplexity, or uncertainty Pang et al. (2026); Tang et al. (2026); Zhao et al. (2025); Shi et al. (2026) are inexpensive but provide only coarse approximations. Bayesian posterior and bandit-style approaches Qu et al. (2026); Hu et al. (2025b); Shen et al. (2026) offer principled online estimation but typically require dedicated evaluation rollouts. Recent work also explores internal model representations or LLM-as-judge methods as difficulty signals Lee et al. (2025); Zhu et al. (2025); Tabib & Deedar (2025). Despite these advances, many estimators rely on proxy signals or stale rollout statistics that become increasingly inaccurate as the policy changes during RL, as demonstrated in Section 2.

B Proof of Proposition 3.1

Proof. All probabilities and expectations below are conditional on the fixed estimation epoch, current policy, behavior policies, and clipping rule described before the proposition; we omit this conditioning to keep notation light. Write $\mathbb{E}_k[\cdot] \triangleq \mathbb{E}_{O\sim\mu_k(\cdot|q)}[\cdot]$. Fix the prompt q throughout and write

$$Y_k = Y(O_k), \qquad \omega_k = W_k(O_k), \qquad d_\theta = d_\theta(q).$$

By absolute continuity, the likelihood ratio $\rho_k(O) = \frac{d\pi_\theta(\cdot|q)}{d\mu_k(\cdot|q)}(O)$ satisfies the change-of-measure identities

$$\mathbb{E}_{O\sim\mu_k(\cdot|q)}[\rho_k(O)Y(O)] = \mathbb{E}_{O\sim\pi_\theta(\cdot|q)}[Y(O)] = d_\theta, \tag{11}$$

and

$$\mathbb{E}_{O\sim\mu_k(\cdot|q)}[\rho_k(O)] = 1. \tag{12}$$

Define the empirical numerator and denominator

$$A = \frac{1}{K}\sum_{k=1}^{K}\omega_k Y_k, \qquad B = \frac{1}{K}\sum_{k=1}^{K}\omega_k, \tag{13}$$

and their expectations

$$a = \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{O\sim\mu_k(\cdot|q)}[W_k(O)Y(O)], \qquad b = \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{O\sim\mu_k(\cdot|q)}[W_k(O)]. \tag{14}$$

On the event $B > 0$, $\hat{d}_q = A/B$, and the denominator assumption gives $b \geq b_{\min}$.

Since $0 \leq \omega_k Y_k \leq C_w$ and $0 \leq \omega_k \leq C_w$, the conditional independence assumption allows Hoeffding's inequality for independent bounded random variables, giving

$$\mathbb{P}\left(|A - a| \geq \varepsilon\right) \leq 2\exp\left(-\frac{2K\varepsilon^2}{C_w^2}\right), \tag{15}$$

and

$$\mathbb{P}\left(|B - b| \geq \varepsilon\right) \leq 2\exp\left(-\frac{2K\varepsilon^2}{C_w^2}\right). \tag{16}$$

Set $\varepsilon = \varepsilon_\delta = C_w\sqrt{\log(4/\delta)/(2K)}$. A union bound implies that, with probability at least $1-\delta$,

$$|A - a| \leq \varepsilon_\delta, \qquad |B - b| \leq \varepsilon_\delta. \tag{17}$$

On this event, $\varepsilon_\delta < b_{\min}$ implies

$$B \geq b - \varepsilon_\delta \geq b_{\min} - \varepsilon_\delta > 0. \tag{18}$$

Moreover, because $0 \leq Y \leq 1$, we have $0 \leq a \leq b$. Therefore,

$$\begin{aligned}\left|\frac{A}{B} - \frac{a}{b}\right| &= \left|\frac{b(A-a) - a(B-b)}{bB}\right| \\ &\leq \frac{b|A-a| + a|B-b|}{bB} \\ &\leq \frac{2\varepsilon_\delta}{B} \leq \frac{2\varepsilon_\delta}{b_{\min} - \varepsilon_\delta}.\end{aligned} \tag{19}$$

It remains to compare a/b with the current-policy difficulty d_θ. Using Equation 11 and Equation 12,

$$a = d_\theta - \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{O\sim\mu_k(\cdot|q)}\left[(\rho_k(O) - W_k(O))Y(O)\right], \tag{20}$$

and

$$b = 1 - \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{O\sim\mu_k(\cdot|q)}\left[\rho_k(O) - W_k(O)\right]. \tag{21}$$

Let

$$S = \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{O\sim\mu_k(\cdot|q)}\left[\rho_k(O) - W_k(O)\right],$$

$$T = \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{O\sim\mu_k(\cdot|q)}\left[(\rho_k(O) - W_k(O))Y(O)\right].$$

Then $b = 1 - S$, $a = d_\theta - T$, and $b \geq b_{\min}$. Since $Y(O) \in [0, 1]$, we also have $d_\theta \in [0, 1]$, and hence $|d_\theta - Y(O)| \leq 1$. Thus

$$\begin{aligned}\left|\frac{a}{b} - d_\theta\right| &= \left|\frac{d_\theta - T}{1 - S} - d_\theta\right| \\ &= \frac{|d_\theta S - T|}{1 - S} \leq \frac{1}{b_{\min} K} \sum_{k=1}^{K} \mathbb{E}_{O \sim \mu_k(\cdot|q)} \left[|\rho_k(O) - W_k(O)|\, |d_\theta - Y(O)|\right] \\ &\leq \frac{1}{b_{\min} K} \sum_{k=1}^{K} \mathbb{E}_{O \sim \mu_k(\cdot|q)} \left[|\rho_k(O) - W_k(O)|\right]. \end{aligned} \tag{22}$$

Combining (19) and (22) by the triangle inequality gives (5). Finally, for the two-sided log clipping in Equation 4,

$$W_k(O) = \min\{\max\{\rho_k(O), e^{-c}\}, e^{c}\},$$

and therefore

$$|\rho_k(O) - W_k(O)| = (\rho_k(O) - e^{c})_+ + (e^{-c} - \rho_k(O))_+,$$

which yields the stated log-clipping special case. □

C Full Algorithm

The complete procedure of DARE is summarized in Algorithm 1. At the start of training, a small reference set is rolled out to bootstrap embedding-based difficulty estimates for unseen prompts (lines 3–5). In each epoch, DARE first updates the difficulty estimate $\hat{d}_q$ for every prompt: cold-start prompts use similarity-weighted interpolation over the reference set, while prompts with buffered rollouts are corrected for policy drift via clipped self-normalized importance sampling (lines 9–17). These estimates then define a Beta-shaped sampling distribution that concentrates each training batch on medium-difficulty prompts while retaining nonzero probability across the full difficulty spectrum (lines 21–22). Within the sampled batch, the rollout budget is set from $\hat{d}_q$ before generation: fewer rollouts and a shorter response budget for easy prompts, the base group size for medium ones, and additional partly hint-augmented rollouts for hard ones (lines 22–28). Once the group is scored, reward shaping follows its realized difficulty $\tilde{d}_q$: a length penalty on correct rollouts of easy groups, a length bonus on incorrect rollouts of hard groups (lines 29–36). Finally, the policy is updated on a mixture of fresh and replayed trajectories with difficulty-conditioned clipping, and new rollouts are pushed into the replay buffer for future reuse (lines 37–39).

D Prompt Templates

All difficulty levels share the same system prompt. For hard prompts, additional hints are appended to the user content.

System prompt (all difficulty levels).

```
Let's think step by step and output the final answer
within \boxed{}.
```

For GSM8K-style tasks, the system prompt is:

```
Let's think step by step and output the final answer
after "####".
```

Think-longer rollout. The following hint is appended to the user content for hard prompts ($\hat{d}_q > d_{\text{hard}}$):

```
This is a challenging problem.  Please think longer
and more carefully.  Break it down step by step before
giving your final answer.
```

Algorithm 1 DARE: Difficulty-Adaptive Reinforcement Learning with Co-Evolved Difficulty Estimation

Require: Training set $\mathcal{D}$, reference set $\mathcal{D}_{\text{ref}}$, initial policy π_θ, replay buffer $\mathcal{B} \leftarrow \emptyset$, Beta concentration κ, thresholds $d_{\text{easy}}, d_{\text{hard}}$, rollout counts $G, G_{\text{easy}}, G_{\text{hard}}$, replay mix ratio σ, clipping bound c, ESS threshold τ_{ESS}, shaping coefficients $\lambda_{\text{easy}}, \lambda_{\text{hard}}$

/* Cold-start: initialize reference difficulties */
for each reference prompt $p_i \in \mathcal{D}_{\text{ref}}$ **do**
  Generate G rollouts from $\pi_\theta(\cdot \mid p_i)$; compute $d_{p_i} \leftarrow 1 - \bar{r}_{p_i}$ {Eq. (2)}
end for
for each training epoch **do**
  /* Phase 1: Co-Evolved Difficulty Estimation (§3.1) */
  for each prompt $q \in \mathcal{D}$ **do**
    if $\mathcal{B}_q = \emptyset$ **then**
      Compute embedding similarities a_i; set $\hat{d}_q \leftarrow \sum_i a_i \, d_{p_i}$ {Eq. (3)}
    else
      for each rollout $(o_k, r_k, \log \pi_{\theta_{\text{beh},k}})$ in $\mathcal{B}_q$ **do**
        $w_k \leftarrow \exp\big(\text{clip}(\log \pi_\theta(o_k \mid q) - \log \pi_{\theta_{\text{beh},k}}(o_k \mid q),\ -c,\ c)\big)$ {Eq. (4)}
      end for
      $\text{ESS}_q \leftarrow (\sum_k w_k)^2 \big/ \sum_k w_k^2$ {Eq. (24)}
      $\hat{d}_q \leftarrow \sum_k w_k (1 - r_k) \big/ \sum_k w_k$ if $\text{ESS}_q \geq \tau_{\text{ESS}}$, else $\hat{d}_q \leftarrow \sum_i a_i \, d_{p_i}$ {fall back if weights collapse}
    end if
  end for
  /* Phase 2: Difficulty-Guided Dynamic Data Selection (§3.2) */
  Set $p_{\text{DARE}}(q) \propto \text{Beta}(\hat{d}_q;\ 1 + \kappa/2,\ 1 + \kappa/2)$ for all q {Eq. (6)}
  Sample batch $\mathcal{Q}$ from $\mathcal{D}$ without replacement according to p_{DARE}
  /* Phase 3: Difficulty-Adaptive Policy Optimization (§3.3) */
  /* Pre-rollout: allocate compute from the *estimated* difficulty */
  for each prompt $q \in \mathcal{Q}$ **do**
    Set $G_q \leftarrow G_{\text{easy}}$ and impose a shorter response budget **if** $\hat{d}_q < d_{\text{easy}}$ {⚙}
    Set $G_q \leftarrow G$ **if** $d_{\text{easy}} \leq \hat{d}_q \leq d_{\text{hard}}$ {⚙}
    Set $G_q \leftarrow G_{\text{hard}}$ and generate G standard $+ (G_q - G)$ hint-augmented rollouts **if** $\hat{d}_q > d_{\text{hard}}$ {⚙}
    Generate the G_q rollouts $\{o_i\}$ from $\pi_{\theta_{\text{beh}}}(\cdot \mid q)$ and score them to obtain $\{r_i\}$
  end for
  /* Post-rollout: shape rewards from the *realized* difficulty */
  for each prompt $q \in \mathcal{Q}$ **do**
    $\tilde{d}_q \leftarrow 1 - \frac{1}{G_q} \sum_i r_i$ {Eq. (7)}
    ⚙ If $\tilde{d}_q < d_{\text{easy}}$: $\tilde{r}_i \leftarrow r_i - \mathbf{1}_{r_i=1} \lambda_{\text{easy}} w_q^{\text{easy}} |o_i| / L_q^{\max}$, $\epsilon_q^+ > \epsilon$ {Eq. (9)}
    ⚙ Else if $\tilde{d}_q > d_{\text{hard}}$: $\tilde{r}_i \leftarrow r_i + \mathbf{1}_{r_i=0} \lambda_{\text{hard}} w_q^{\text{hard}} |o_i| / L_q^{\max}$ {Eq. (10)}
    ⚙ Else: $\tilde{r}_i \leftarrow r_i$, $\epsilon_q^- \leftarrow \epsilon_q^+ \leftarrow \epsilon$ {standard GRPO}
    Compute group-relative advantages A_i from $\{\tilde{r}_i\}_{i=1}^{G_q}$
  end for
  Construct batch: mix fraction σ fresh on-policy rollouts with $(1 - \sigma)$ replay trajectories
  Update θ by maximizing $\mathcal{J}_{\text{DARE}}(\theta)$ with difficulty-conditioned clipping {Eq. (8)}
  Push new rollouts into $\mathcal{B}$ (FIFO eviction)
end for
return Optimized policy π_θ

Hint-augmented Prompt. For hard prompts with a successful historical trajectory available in the replay buffer, the user content is augmented with a reasoning skeleton extracted from the successful response:

```
[Original question]
Helpful hint from one previous successful attempt on
this same problem:
- Guided reasoning skeleton:
1.  [Top-scoring fragment 1]
2.  [Top-scoring fragment 2]
3.  [Top-scoring fragment 3]
Use the hint as guidance, but still reason carefully
and give your own complete solution within \boxed{}.
```

E Difficulty Estimation Baselines

All methods below share the same prompt selection rule: given predicted difficulty labels $\hat{d}_q$ for every prompt $q \in \mathcal{D}$, sampling weights are computed as $p(q) \propto \mathrm{Beta}(\hat{d}_q;\, \alpha, \alpha)$ with $\alpha = 1 + \kappa/2$ (Eq. (6)), and B prompts are drawn via multinomial sampling. The methods differ only in how $\hat{d}_q$ is obtained.

Random Selection. No difficulty estimation is performed. All prompts are assigned a uniform predicted label $\hat{d}_q = 0.5$, and B prompts are drawn uniformly at random from $\mathcal{D}$ without replacement.

Algorithm 2 *Random Selection*

Require: Training prompts $\mathcal{D}$, selection budget B
1: **for** each training step **do**
2: $\quad \hat{d}_q \leftarrow 0.5 \quad \forall\, q \in \mathcal{D}$
3: $\quad$ Draw B prompts uniformly at random: $\mathcal{S} \leftarrow \textsc{RandomSample}(\mathcal{D}, B)$
4: $\quad$ Roll out G responses per prompt in $\mathcal{S}$; proceed to GRPO update
5: **end for**

Embedding-Based Prediction. This method follows the teacher-model framework of Sun et al. (2025). At the beginning of each training step, a reference set $\mathcal{R} \subset \mathcal{D}$ of n_{ref} prompts is sampled uniformly and rolled out with the current policy π_θ to obtain ground-truth success rates. A lightweight teacher model f_ϕ—a 3-layer residual MLP head attached to the frozen last-hidden-state embeddings of Qwen2.5-Math-1.5B-Instruct—computes the cosine similarity between each remaining prompt and the reference set in embedding space, and predicts the success rate via group-logit-temperature scaling. The predicted difficulty is $\hat{d}_q = 1 - \hat{s}_q^{\mathrm{teacher}}$.

Algorithm 3 *Embedding-Based Prediction* (Sun et al., 2025)

Require: Training prompts $\mathcal{D}$, teacher model f_ϕ, reference size n_{ref}, selection budget B
1: **for** each training step **do**
2: $\quad$ Sample reference set $\mathcal{R} \subset \mathcal{D}$, $|\mathcal{R}| = n_{\mathrm{ref}}$
3: $\quad$ Roll out G responses per prompt in $\mathcal{R}$; compute ground-truth s_q for $q \in \mathcal{R}$
4: $\quad$ Extract embeddings $\mathbf{h}_q$ for all $q \in \mathcal{D}$ from the frozen encoder
5: $\quad$ **for** each $q \in \mathcal{D} \setminus \mathcal{R}$ **do**
6: $\quad\quad \hat{s}_q^{\mathrm{teacher}} \leftarrow f_\phi(\mathbf{h}_q,\, \{\mathbf{h}_{q'}, s_{q'}\}_{q' \in \mathcal{R}})$
7: $\quad\quad \hat{d}_q \leftarrow 1 - \hat{s}_q^{\mathrm{teacher}}$
8: $\quad$ **end for**
9: $\quad$ For $q \in \mathcal{R}$: $\hat{d}_q \leftarrow 1 - s_q$
10: $\quad$ Select B prompts via Beta sampling on $\{\hat{d}_q\}$
11: **end for**

Entropy-Based Estimation. Following Pang et al. (2026), this method uses the token-level entropy of the current policy as a proxy for difficulty. For each prompt q, a single short prefix of L_{prefix} tokens is generated from π_θ at temperature T_{ent}. The token-level entropy $H_t = -\sum_v \pi_\theta(v \mid q, o_{<t}) \log \pi_\theta(v \mid q, o_{<t})$ is computed at each generated position, and the prompt-level score is the mean $\bar{H}_q = \frac{1}{L_{\text{prefix}}} \sum_{t=1}^{L_{\text{prefix}}} H_t$. Prompts are then selected via softmax sampling with temperature τ_{ent}: $p(q) \propto \exp(\bar{H}_q / \tau_{\text{ent}})$, which favors high-entropy (uncertain) prompts.

Algorithm 4 *Entropy-Based Estimation* (Pang et al., 2026)

Require: Training prompts $\mathcal{D}$, prefix length L_{prefix}, temperature T_{ent}, softmax temperature τ_{ent}, budget B

for each training step **do**
  for each $q \in \mathcal{D}$ **do**
    Generate prefix $o_{1:L_{\text{prefix}}} \sim \pi_\theta(\cdot \mid q)$ with temperature T_{ent}
    Compute $\bar{H}_q \leftarrow \frac{1}{L_{\text{prefix}}} \sum_{t=1}^{L_{\text{prefix}}} H_t$
  end for
  $p(q) \propto \exp\big(\bar{H}_q \,/\, \tau_{\text{ent}}\big)$
  Draw B prompts via multinomial sampling from p
end for

LLM-Judge. Inspired by Tabib & Deedar (2025), this method uses the policy model π_θ itself to judge prompt difficulty. For each prompt q, a difficulty-estimation instruction is prepended as a system message, asking the model to output a success probability $\hat{s}_q \in [0, 1]$. The model generates a short response (up to 32 tokens) at temperature $T_{\text{judge}} = 0.3$, and the first valid floating-point number in the output is parsed as $\hat{s}_q$. The predicted difficulty is $\hat{d}_q = 1 - \hat{s}_q$. Prompts for which parsing fails are assigned $\hat{d}_q = 0.5$.

Algorithm 5 *LLM-Judge* (Tabib & Deedar, 2025)

Require: Training prompts $\mathcal{D}$, policy π_θ, judge temperature T_{judge}, budget B

for each training step **do**
  for each $q \in \mathcal{D}$ **do**
    Construct judge prompt: system instruction + q
    Generate response $r \sim \pi_\theta(\cdot \mid \text{judge prompt})$ with $T = T_{\text{judge}}$, max 32 tokens
    Parse $\hat{s}_q$ from r; set $\hat{s}_q \leftarrow 0.5$ on parse failure
    $\hat{d}_q \leftarrow 1 - \hat{s}_q$
  end for
  Select B prompts via Beta sampling on $\{\hat{d}_q\}$
end for

Bayesian Posterior Estimation. Following Qu et al. (2026), each prompt q is modeled as a Bernoulli arm with latent success rate $\gamma_q \sim \text{Beta}(\alpha_q, \beta_q)$. The posterior is initialized with a uniform prior $(\alpha_0, \beta_0) = (1, 1)$. After each rollout batch where prompt q receives k rollouts with s successes, the posterior is updated with temporal discounting factor λ:

$$\alpha_q \leftarrow \lambda\, \alpha_q + (1 - \lambda)\, \alpha_0 + s, \qquad \beta_q \leftarrow \lambda\, \beta_q + (1 - \lambda)\, \beta_0 + (k - s). \tag{23}$$

Difficulty is estimated via Thompson Sampling: $\tilde{\gamma}_q \sim \text{Beta}(\alpha_q, \beta_q)$, and $\hat{d}_q = 1 - \tilde{\gamma}_q$. For prompts not yet covered by any rollout, a default label $\hat{d}_q = 0.5$ is assigned, and an exploration ratio ρ_{explore} controls the probability mass allocated to uncovered prompts.

Algorithm 6 *Bayesian Posterior Estimation* (Qu et al., 2026)

Require: Training prompts $\mathcal{D}$, prior (α_0, β_0), decay λ, explore ratio ρ_{explore}, budget B
Initialize $\alpha_q \leftarrow \alpha_0,\ \beta_q \leftarrow \beta_0 \quad \forall\, q \in \mathcal{D}$
for each training step **do**
  for each $q \in \mathcal{D}$ with rollout history **do**
    Draw $\tilde{\gamma}_q \sim \text{Beta}(\alpha_q, \beta_q)$
    $\hat{d}_q \leftarrow 1 - \tilde{\gamma}_q$
  end for
  For uncovered prompts: $\hat{d}_q \leftarrow 0.5$; allocate ρ_{explore} of sampling mass
  Select B prompts via Beta sampling on $\{\hat{d}_q\}$
  Roll out G responses per selected prompt; observe (s_q, k_q)
  Update posteriors via Eq. (23)
end for

Previous FR Estimation. This method uses the empirical failure rate from previous rollouts stored in the replay buffer $\mathcal{B}$ as the difficulty estimate, without importance-weight correction. For each prompt q with K stored rollouts $\{r_k\}_{k=1}^K$ in $\mathcal{B}$, the predicted failure rate is $\hat{f}_q^{\text{raw}} = 1 - \frac{1}{K}\sum_{k=1}^K r_k$, and the difficulty is $\hat{d}_q = \hat{f}_q^{\text{raw}}$. For prompts without buffer entries, a default label $\hat{d}_q = 0.5$ is used. This is equivalent to the SNIS estimator (Section 3.1) with all importance weights set to unity, i.e., ignoring the distributional shift between the behavior policies and the current policy.

Algorithm 7 *Previous FR Estimation*

Require: Training prompts $\mathcal{D}$, replay buffer $\mathcal{B}$, budget B
for each training step **do**
  for each $q \in \mathcal{D}$ with entries in $\mathcal{B}$ **do**
    Retrieve stored rollouts $\{r_k\}_{k=1}^K$ from $\mathcal{B}(q)$
    $\hat{f}_q^{\text{raw}} \leftarrow 1 - \frac{1}{K}\sum_{k=1}^K r_k$
    $\hat{d}_q \leftarrow \hat{f}_q^{\text{raw}}$
  end for
  For prompts without buffer entries: $\hat{d}_q \leftarrow 0.5$
  Select B prompts via Beta sampling on $\{\hat{d}_q\}$
end for

Current FR Estimation. This method serves as the ground-truth oracle for difficulty estimation. At each training step, every prompt $q \in \mathcal{D}$ is rolled out G times with the current policy π_θ at temperature T, yielding the observed failure rate $f_q = 1 - \frac{1}{G}\sum_{i=1}^G r_i$. Only prompts whose observed failure rate falls within a target interval $[f_{\text{lo}}, f_{\text{hi}}]$ centered at 0.5 are retained; rollouts continue until B qualifying prompts are collected. The difficulty is $\hat{d}_q = f_q$.

Algorithm 8 *Current FR Estimation* (Ground Truth)

Require: Training prompts $\mathcal{D}$, policy π_θ, group size G, target interval $[f_{\text{lo}}, f_{\text{hi}}]$, budget B

for each training step **do**
  $\mathcal{S} \leftarrow \emptyset$
  while $|\mathcal{S}| < B$ **do**
    Sample a batch of prompts from $\mathcal{D}$
    for each sampled prompt q **do**
      Generate G rollouts $\{o_i\} \sim \pi_\theta(\cdot \mid q)$; compute $f_q = 1 - \frac{1}{G}\sum_{i=1}^{G} r_i$
      if $f_q \in [f_{\text{lo}},\ f_{\text{hi}}]$ **then**
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{q\}$; $\hat{d}_q \leftarrow f_q$
      end if
    end for
  end while
  Proceed to GRPO update with $\mathcal{S}$
end for

F Experimental Details

The preliminary investigation in Section 2 trains Qwen2.5-Math-1.5B (Yang et al., 2024a) on the same four-dataset union as the main experiments, following the data preparation of Sun et al. (2025): MATH Level 3–5 (Hendrycks et al., 2021b) (8,888 prompts merged from the train and test splits, with the MATH-500 evaluation set removed), DeepScaleR-40K (Luo et al., 2025) (10,240 prompts subsampled from 40K), Open-Reasoner-Zero-57K (Hu et al., 2025a) (8,192 prompts subsampled from 57K), and DeepMath-103K (He et al., 2025) (8,192 prompts subsampled from 103K), for a total of 35,512 prompts. Subsampling uses a fixed seed for reproducibility; each dataset is converted to a common prompt format with a shared system prompt and a rule-based verifiable reward. Training uses GRPO (Shao et al., 2024) with the AdamW optimizer (learning rate 2×10^{-6}, no weight decay), group size $G = 8$, generation temperature $T = 0.6$, maximum response length 3,072 tokens, and entropy coefficient 0. Evaluation is performed on MATH-500 (Lightman et al., 2024) and GSM8K (Cobbe et al., 2021).

Shared parameters for teacher-based and filtration methods. *Random Selection*, *Embedding-Based Prediction*, *Entropy-Based Estimation*, *LLM-Judge*, and *Current FR Estimation* use an effective batch size $B = 512$, on-policy fraction $\sigma = 0.5$ (i.e., 256 fresh rollouts and 256 replay samples per step), and a replay buffer of capacity $C = 512$. Prompt selection uses the symmetric Beta sampling distribution of Eq. (6) with concentration $\kappa = 100$, which is centered at difficulty 0.5.

Shared parameters for buffer-based methods. *Previous FR Estimation* and *Bayesian Posterior Estimation* use $B = 512$ and $\sigma = 1.0$, i.e. every gradient update is computed from fresh on-policy rollouts. These methods still maintain a larger history of capacity $C = 4096$, but it is read only for *difficulty estimation* (failure-rate averaging and posterior updates) and never replayed into the policy-gradient objective. DARE uses the same capacity $C = 4096$ with $\sigma = 0.5$, so its buffer serves both purposes.

Method-specific parameters.

- **Embedding-Based Prediction**: reference set size $n_{\text{ref}} = 256$; teacher model is a 3-layer residual MLP head on top of frozen Qwen2.5-Math-1.5B-Instruct embeddings, with group-logit-temperature scaling; teacher batch size 32.
- **Entropy-Based Estimation**: prefix length $L_{\text{prefix}} = 64$ tokens, generation temperature $T_{\text{ent}} = 0.6$, softmax temperature $\tau_{\text{ent}} = 1.0$. The entropy-based selection replaces the Beta sampling rule with softmax sampling over raw entropy scores.
- **LLM-Judge**: the policy model itself generates difficulty predictions at temperature $T_{\text{judge}} = 0.3$ with a maximum of 32 output tokens per prompt.
- **Bayesian Posterior Estimation**: uniform prior $(\alpha_0, \beta_0) = (1.0, 1.0)$, temporal decay $\lambda = 0.5$, target failure rate $\gamma^* = 0.5$, exploration ratio $\rho_{\text{explore}} = 0.3$.

- **Previous FR Estimation**: uses the same replay buffer as the SNIS estimator ($C = 4096$, $\sigma = 1.0$) but computes the raw average failure rate without importance-weight correction.
- **Current FR Estimation**: rolls out $G = 8$ responses per prompt with the current policy at temperature $T = 0.6$; retains only prompts with observed failure rate $f_q \in [0.4, 0.6]$ (i.e., $f_q = 0.5$ given $G = 8$), continuing until B qualifying prompts are collected.

SNIS difficulty estimation defaults (main experiments). Runs that use SNIS difficulty estimation (Section 3.1) in Section 4 use the following defaults unless explicitly overridden in an ablation: replay buffer size 4,096, Beta sampling concentration $\kappa = 100$, SNIS clipping range $c = 4.0$, ESS threshold $\tau_{\text{ESS}} = 3$, EasyLength penalty coefficient $\lambda_{\text{easy}} = 10^{-4}$, and hard length bonus coefficient $\lambda_{\text{hard}} = 10^{-4}$ when reward shaping is enabled.

G Extended Preliminary Investigation

Table 5 reports the MSE and MAE of each difficulty estimation method against *Current FR Estimation* on 200 held-out prompts. The results show that *Our Estimation* achieves the lowest error among all practical estimation methods, reducing MAE from 0.2775 for the strongest existing baseline, *Embedding-Based Prediction*, to 0.1889. Its error is also close to the intrinsic error of *Current FR Estimation* itself (0.1688), indicating that our policy-aligned estimator better tracks the current policy difficulty during RL training.

Table 5: Estimation error (MSE and MAE) of each difficulty estimator against *Current FR Estimation*, the ground-truth difficulty reference, on 200 held-out prompts. The *Current FR Estimation* row reports its own intrinsic error. Lower is better.

Method	MSE↓	MAE↓
Random Selection	0.1624	0.3719
Entropy-Based Estimation	0.1288	0.3120
Previous FR Estimation	0.1622	0.2988
LLM-Judge	0.1568	0.3688
Bayesian Posterior Estimation	0.0964	0.2839
Embedding-Based Prediction	0.0943	0.2775
Current FR Estimation (Ground Truth)	0.0473	0.1688
Our Estimation	0.0654	0.1889

H Ablation Studies and Additional Investigations

H.1 Effective sample size of the SNIS estimator

To decide whether the SNIS estimate is reliable enough to be used as the difficulty signal, we monitor the effective sample size computed from the clipped importance weights $\{w_k\}_{k=1}^{K}$ (Eq. (4)) associated with the K buffered rollouts of prompt q:

$$\text{ESS}_q = \frac{\left(\sum_{k=1}^{K} w_k\right)^2}{\sum_{k=1}^{K} w_k^2} \in [1,\, K]. \tag{24}$$

ESS_q approaches K when the buffered behavior policies are close to the current policy, and drops toward 1 when the weights are dominated by a single stale rollout. In the estimation phase of Algorithm 1, the SNIS estimate is accepted only if $\text{ESS}_q \geq \tau_{\text{ESS}}$ (we use $\tau_{\text{ESS}} = 3$ throughout); otherwise the estimator falls back to the embedding-based teacher prediction from the cold-start phase. Table 7 reports the coverage (fraction of buffered prompts whose SNIS estimate is accepted) and MAE on the 200 held-out prompts of Table 5 as τ_{ESS} varies over $\{1, 2, 3, 4, 5\}$; $\tau_{\text{ESS}} = 3$ minimizes MAE (0.189) at a coverage of 88.7%, striking the best trade-off between rejecting weight-collapsed estimates and leaving enough prompts under SNIS control to matter.

Table 6: Average and peak GPU memory, together with average and peak CPU memory, under different difficulty-aware RL methods on Qwen2.5-Math-1.5B. All values are in GB.

Method	Avg. GPU (GB)	Peak GPU (GB)	Avg. CPU (GB)	Peak CPU (GB)
GRPO	26.62	50.46	6.50	10.26
EDCO	26.61	50.50	7.80	14.50
LLM-Judge	26.60	50.43	7.33	13.15
DOTS	26.64	50.45	7.08	12.51
Previous FR Estimation	23.75	47.94	9.32	18.85
MoPPS	26.62	50.50	7.80	14.50
DARE	25.75	50.94	7.32	15.65

Table 7: Effect of the ESS acceptance threshold τ_{ESS} on the SNIS difficulty estimator (Qwen2.5-Math-1.5B, same 200 held-out prompts as Table 5, group size $G = 8$ so $\text{ESS}_q \in [1, 8]$). Coverage is the fraction of prompts for which the SNIS estimate is accepted; rejected prompts fall back to the cold-start embedding predictor. Lower MAE is better.

τ_{ESS}	1	2	3	4	5
Coverage (%)	99.1	95.0	88.7	76.4	46.2
MAE↓	0.2190	0.2010	**0.1889**	0.2185	0.2498

H.2 Case study: stratified generation behavior.

Figure 5 illustrates the stratified generation behavior with two representative examples from the MATH benchmark (Hendrycks et al., 2021b).

The Level-1 problem is solved correctly by all 64 rollouts; the shortest correct response uses only 219 tokens, demonstrating that the EasyLength penalty successfully encourages concise reasoning on easy prompts. In contrast, the Level-5 problem has an accuracy of only 23/64: the shorter 427-token rollout misassigns the intersection sizes and arrives at the wrong answer (100 instead of 110), while the longer 970-token rollout carefully applies the inclusion-exclusion principle and reaches the correct answer. This contrast illustrates how the think-longer re-rollout for hard prompts—generating extra rollouts under a hint-augmented prompt and replacing incorrect responses with correct ones—provides richer gradient signal that pure filtration cannot supply.

H.3 Entropy–Difficulty Relationship Across Models

Figure 6 visualizes the relationship between sequence-level reasoning entropy and normalized prompt difficulty across four models (Qwen2.5-Math-1.5B, Qwen2.5-3B, Qwen2.5-Math-7B, and Qwen2.5-7B) on multiple datasets. Across most settings, entropy tends to be lower for prompts of moderate difficulty, consistent with the intuition that mid-difficulty prompts elicit more focused reasoning patterns. However, the magnitude and consistency of this trend vary substantially across model–dataset combinations, suggesting that entropy alone is an unreliable proxy for adaptive difficulty—motivating the importance-sampling-based estimator proposed in Section 3.1.

H.4 Ablation results on Qwen2.5-Math-1.5B.

Table 10 isolates the contribution of each component on Qwen2.5-Math-1.5B by adding one design at a time on top of *GRPO* + *SNIS*, and then evaluates their combined effect in DARE. The results show that each configuration improves over *GRPO* on overall accuracy, with gains ranging from $+1.7$ for *SNIS* alone to $+3.4$ for the combined reward shaping. Tiered rollout allocation, the memory-guided hint, and the combined shaping terms provide consistent gains across most benchmarks, while the easy penalty applied on its own trades accuracy on the harder benchmarks for a large gain on GSM8K ($+8.8$) and the token reductions reported in Table 3. Combining all components yields the best overall score, improving it from 45.3 to 50.9 ($+5.6$), which confirms that the components are complementary rather than redundant.

Level 1 (Easy) *Long: 334 tokens* *Short: 219 tokens*

Problem: Find k, if $(3^k)^6 = 3^6$.

Long rollout (334 tokens, ✓):

To solve $(3^k)^6 = 3^6$, we use $(a^m)^n = a^{m \cdot n}$:

$$(3^k)^6 = 3^{6k}$$

So $3^{6k} = 3^6$, thus $6k = 6$, giving $k = 1$. Substituting back: $(3^1)^6 = 3^6 \implies 3^6 = 3^6$. The solution is correct. We can also verify with code: `(3**1)**6 == 3**6` returns `True`. $\boxed{1}$

Short rollout (219 tokens, ✓):

Using the power rule $(a^m)^n = a^{mn}$: $(3^k)^6 = 3^{6k}$. So $3^{6k} = 3^6 \Rightarrow 6k = 6 \Rightarrow k = 1$. Substituting back: $(3^1)^6 = 3^6 \implies 3^6 = 3^6$. $\boxed{1}$

Level 5 (Hard) *Short wrong vs. long correct rollout*

Problem: There are 360 people in a school. 15 take calculus, physics, and chemistry; 15 take none. 180 take calculus. Twice as many take chemistry as physics. 75 take both calculus and chemistry; 75 take both physics and chemistry; 30 take both physics and calculus. How many take physics?

Short rollout (427 tokens, ✗):

Using inclusion-exclusion: $180 + P + 2P - (15 + 75 + 75) + 30 = 345$, so $45 + 3P = 345$, giving $P = 100$. $\boxed{100}$ *(wrong: misassigns $|C \cap P| = 15$ and $|C \cap P \cap Ch| = 30$)*

Long rollout (970 tokens, ✓):

Let $|C| = 180$, $|P|$, $|Ch| = 2|P|$, $|C \cap P \cap Ch| = 15$, $|C \cap Ch| = 75$, $|P \cap Ch| = 75$, $|C \cap P| = 30$, $|C \cup P \cup Ch| = 345$. By inclusion-exclusion:

$$345 = 180 + |P| + 2|P| - 30 - 75 - 75 + 15 \Rightarrow 330 = 3|P| \Rightarrow |P| = 110.$$

$\boxed{110}$

Figure 5: Case study of stratified generation behavior. For the easy Level-1 problem, the model produces a concise 219-token solution. For the hard Level-5 problem, a short 427-token rollout misassigns intersection values and gives the wrong answer (100), while a longer 970-token rollout carefully applies inclusion-exclusion and arrives at the correct answer (110).

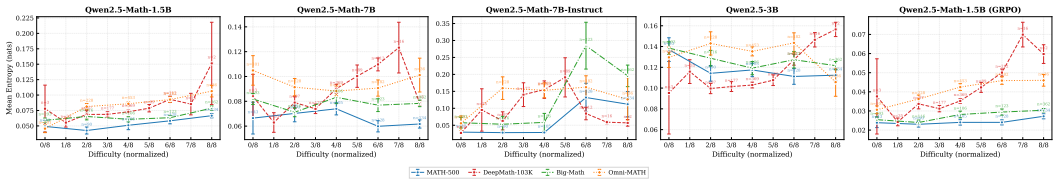


Figure 6: Reasoning entropy versus normalized difficulty across multiple models and datasets, adapted from DiffAdapt Liu et al. (2026). The overall tendency is compatible with lower entropy in the middle difficulty range, although the magnitude and consistency of the effect vary across settings.

H.5 Additional results on SmolLM3-3B.

Figure 7 compares the training dynamics of DARE and baseline methods on SmolLM3-3B-Base, using both training steps and wall-clock time as the optimization budget. The results show that DARE consistently achieves higher accuracy than all baselines on both MATH-500 and GSM8K throughout training. The shaded regions indicate a clear and persistent performance gap over the strongest competing method, showing that the gain is not limited to the final checkpoint. Moreover, the wall-clock comparison on MATH-500 confirms that this advantage remains when accounting for

Table 8: Effect of the easy penalty coefficient λ_{easy} and hard bonus coefficient λ_{hard} in the shaped reward on Qwen2.5-Math-1.5B. The top row is *GRPO + SNIS*. "Tok" denotes the average number of response tokens per problem at inference time.

λ_{easy}	λ_{hard}	AIME24/25		GSM8K		AIME-AMC		Minerva		Olympiad	
		Acc	Tok	Acc	Tok	Acc	Tok	Acc	Tok	Acc	Tok
–	–	**8.33**	1157	80.14	374	46.99	860	14.34	826	24.04	922
10^{-4}	10^{-4}	3.33	1466	82.49	351	46.99	956	**17.28**	882	**25.07**	1062
10^{-4}	–	6.67	**1000**	86.43	**233**	46.99	796	12.87	720	21.96	764
2×10^{-4}	10^{-4}	3.33	1104	**89.01**	248	44.58	**741**	12.87	**712**	21.07	**667**
10^{-4}	2×10^{-4}	6.67	2011	80.52	588	**48.19**	1842	16.91	1515	24.04	1491

Table 9: Ablation on asymmetric clipping range for easy prompts on Qwen2.5-Math-1.5B. The clipping bounds are $1-\epsilon_q^-$ and $1+\epsilon_q^+$; the top row is *GRPO + SNIS* with symmetric clip ranges. Relaxing only the upper bound (0.8–1.6) yields the best overall accuracy.

Clip Range	AIME24/25	MATH-500	GSM8K	AIME-AMC	Minerva	Olympiad
0.8–1.2	8.33	69.80	80.14	46.99	14.34	24.04
0.6–1.2	6.67	69.40	78.85	45.78	11.76	23.44
0.6–1.4	8.33	70.40	81.80	49.40	17.65	24.04
0.8–1.4	8.33	70.60	82.34	49.40	18.75	24.04
0.8–1.6	10.00	70.80	81.20	50.60	19.85	24.04

actual training time, indicating that DARE improves both training efficiency and final effectiveness on SmolLM3-3B-Base.

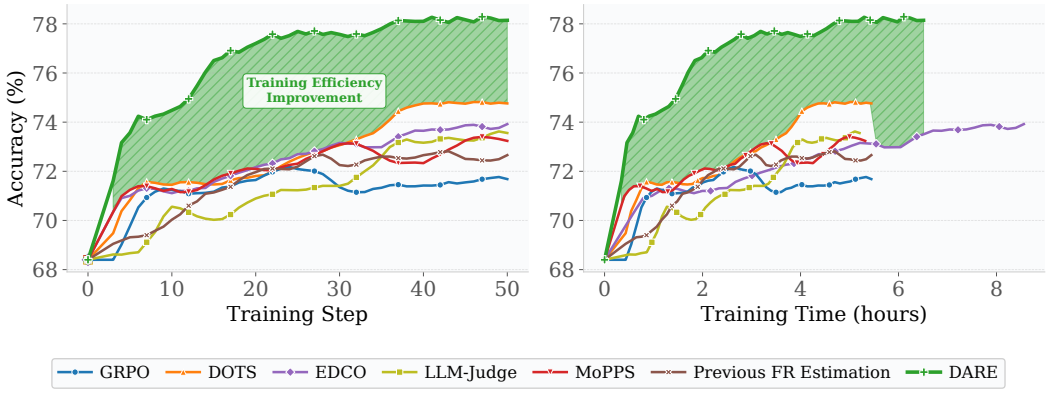


Figure 7: Training curves on MATH-500 (left), GSM8K (center), and MATH-500 vs. training time (right) for SmolLM3-3B-Base. DARE consistently outperforms all baselines throughout training, with the shaded region highlighting the accuracy advantage over the best competing method. The training-time plot confirms that the advantage is preserved when the x-axis is wall-clock time rather than training steps.

I GENERALIZATION TO CODE GENERATION

To verify that the SNIS-over-baselines ordering observed on mathematical reasoning also holds on a domain with execution-based rewards, we train Qwen2.5-Coder-1.5B (Hui et al., 2024) under the same set of difficulty-estimation baselines used in Section 4. Training data consists of 10,240 problems drawn in a 50/50 split from TACO (Li et al., 2023) (BAAI) and APPS (Hendrycks et al., 2021a) (codeparrot), with per-problem stdin/stdout test cases used as the verifiable reward. Evaluation covers the standard code-generation setup on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), together with the contamination-controlled LiveCodeBench (Jain et al., 2025).

Table 10: Per-component ablation on Qwen2.5-Math-1.5B, mirroring Table 4. All rows below *GRPO* include the SNIS estimator; each subsequent row adds one further component on top of *GRPO* + *SNIS*. The bottom row DARE *(all components)* combines all components. ▲ / ▼ denotes improvement / degradation over *GRPO*.

Method	MATH-500	GSM8K	AIME-AMC	Minerva	Olympiad	Overall
GRPO	67.4	77.6	45.8	13.2	22.6	45.3
+ *SNIS*	69.8 ▲2.4	80.1 ▲2.5	47.0 ▲1.2	14.3 ▲1.1	24.0 ▲1.4	47.0 ▲1.7
+ tiered rollout allocation	71.6 ▲4.2	83.4 ▲5.8	47.0 ▲1.2	15.1 ▲1.9	24.2 ▲1.6	48.3 ▲3.0
+ memory-guided hint	70.4 ▲3.0	83.5 ▲5.9	48.2 ▲2.4	16.2 ▲3.0	24.0 ▲1.4	48.5 ▲3.2
+ easy penalty	68.6 ▲1.2	**86.4** ▲8.8	47.0 ▲1.2	12.9 ▼0.3	22.0 ▼0.6	47.4 ▲2.1
+ easy penalty + hard bonus	71.6 ▲4.2	82.5 ▲4.9	47.0 ▲1.2	17.3 ▲4.1	25.1 ▲2.5	48.7 ▲3.4
DARE (all components)	**73.2** ▲5.8	84.5 ▲6.9	**50.6** ▲4.8	**19.9** ▲6.7	**26.4** ▲3.8	**50.9** ▲5.6

Table 11 reports final accuracy on HumanEval and MBPP, where SNIS difficulty estimation is the top row on both (46.9 and 71.4). Figure 8 shows the training trajectories on LiveCodeBench, where SNIS leads every baseline in both training steps and wall-clock time. Together, these results indicate that the estimator-level advantage transfers from mathematical reasoning to code generation.

Table 11: Final accuracy on HumanEval and MBPP on Qwen2.5-Coder-1.5B, trained on TACO + APPS (50/50, 10,240 prompts). Results are reported as pass@1 (%). The best per-column value is shown in bold.

Method	HumanEval	MBPP
GRPO	44.96	69.32
DOTS	43.70	68.37
EDCO	43.23	70.17
LLM-Judge	45.80	70.48
MoPPS	46.17	69.29
Previous FR Estimation	45.58	69.72
DARE (Ours)	**46.86**	**71.44**

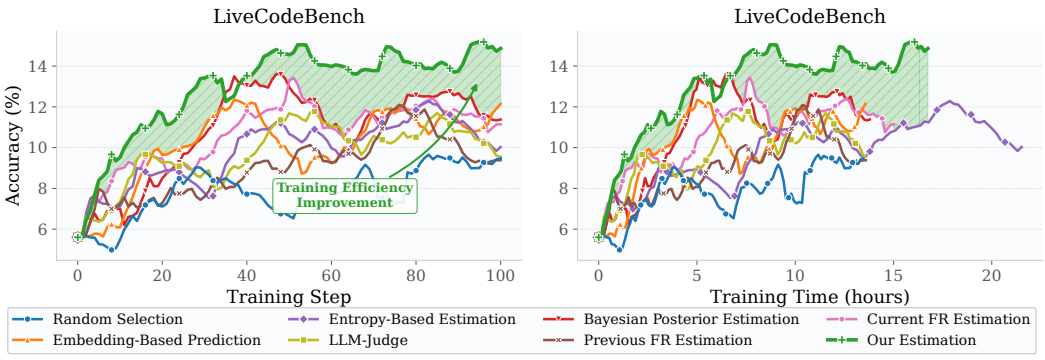


Figure 8: Training curves on LiveCodeBench for Qwen2.5-Coder-1.5B Hui et al. (2024), against training steps (left) and wall-clock time (right). Our difficulty estimation exceeds all baselines throughout training on both axes.

J TRAINING EFFICIENCY ANALYSIS

Metrics. To go beyond visual inspection of the training curves in Figure 4 and Figure 7, we summarize each training run with two budget-level metrics defined directly on the per-epoch accuracy trajectory $\mathrm{acc}_m(s)$ of method m at training step s. Let H denote the common horizon (the last step

reached by every method on a given benchmark). We define

$$S_m(\tau) = \min\{s \in [0, H] : \mathrm{acc}_m(s) \geq \tau\}, \qquad \mathrm{sp}_m(\tau) = \frac{S_{\mathrm{base}}(\tau)}{S_m(\tau)}, \tag{25}$$

the step at which m first reaches target accuracy τ and the corresponding *step speedup* over the uniform-sampling *GRPO* baseline. We further define the normalized area under the accuracy curve,

$$\mathrm{AUC}_m = \frac{1}{H}\int_0^H \mathrm{acc}_m(s)\, ds \approx \frac{1}{H}\sum_{s=0}^{H-1} \tfrac{1}{2}\big(\mathrm{acc}_m(s) + \mathrm{acc}_m(s{+}1)\big), \tag{26}$$

which equals the average accuracy attained throughout training and is insensitive to evaluation noise at any single step. Unlike final accuracy, AUC_m jointly penalizes slow starts and early plateaus, so a higher value means both *how fast* and *how high* the curve climbs are favorable.

Results on Qwen2.5-Math-1.5B and Qwen2.5-Math-7B. Table 12 reports $S_m(\tau)$, $\mathrm{sp}_m(\tau)$, and AUC_m on Qwen2.5-Math-1.5B and Qwen2.5-Math-7B for MATH-500 and GSM8K. DARE reaches the MATH-500 target on Qwen2.5-Math-1.5B (67%) in 42 steps, compared to 118 steps for *GRPO* (sp=2.81×); on Qwen2.5-Math-7B MATH-500 the target (72%) is reached in 14 steps vs. 122 (sp=8.71×). In terms of AUC, DARE is the top row on Qwen2.5-Math-1.5B MATH-500 and ties the top tier (AUC within 0.2 of the best) on Qwen2.5-Math-1.5B GSM8K and Qwen2.5-Math-7B MATH-500. On Qwen2.5-Math-7B GSM8K, *MoPPS* attains the highest AUC (88.8 vs. 86.5).

Table 12: Training efficiency of DARE versus filtration-only baselines on Qwen2.5-Math-1.5B and Qwen2.5-Math-7B. Baseline rows are computed from the per-epoch accuracy trajectories; the DARE row is computed from the corresponding DARE run under the identical evaluation schedule. AUC is the normalized area under the accuracy curve (Equation 26, as a percentage). S_τ is the smallest training step at which a method first reaches accuracy τ (Equation 25); "–" means the target is not reached within the common horizon. sp_τ is the step speedup $S_{\mathrm{GRPO}}(\tau)/S_m(\tau)$; on Qwen2.5-Math-7B GSM8K, *GRPO* does not reach τ=0.88, so we report S_{88} only.

	Qwen2.5-Math-1.5B				*Qwen2.5-Math-7B*			
	MATH-500 (H=194)		GSM8K (H=194)		MATH-500 (H=198)		GSM8K (H=198)	
Method	AUC	S_{67}/sp_{67}	AUC	S_{78}/sp_{78}	AUC	S_{72}/sp_{72}	AUC	S_{88}
GRPO	64.4	118 / 1.00×	72.4	184 / 1.00×	72.2	122 / 1.00×	84.3	–
DOTS	65.7	64 / 1.84×	75.5	74 / 2.49×	**73.3**	52 / 2.35×	86.3	56
EDCO	65.5	86 / 1.37×	75.2	104 / 1.77×	73.2	52 / 2.35×	86.1	**12**
LLM-Judge	65.1	102 / 1.16×	74.1	124 / 1.48×	72.8	52 / 2.35×	85.4	162
MoPPS	65.1	92 / 1.28×	**77.3**	54 / 3.41×	72.7	114 / 1.07×	**88.8**	30
Previous FR Estimation	64.1	114 / 1.04×	73.1	82 / 2.24×	73.0	118 / 1.03×	86.1	58
DARE (Ours)	**66.3**	**42 / 2.81**×	77.1	**46 / 4.00**×	73.2	**14 / 8.71**×	86.5	36

Results on SmolLM3-3B. Table 13 reports the same metrics on SmolLM3-3B-Base, computed from the curves in Figure 7. The common horizon is H=98 steps; targets are chosen close to the plateau of *GRPO* so that the step speedups remain informative. DARE is the top row on both MATH-500 and GSM8K in AUC and reaches both targets in the fewest steps.

K ACCURACY UNDER CONSTRAINED ROLLOUT BUDGETS

To further validate the practical impact of improved difficulty estimation, we evaluate downstream accuracy under constrained rollout budgets on two challenging benchmarks: AIME-AMC (83 problems) and MATH-500 Level 5 (134 problems). As shown in Figure 9, our SNIS difficulty estimation method consistently achieves higher accuracy than *Embedding-Based Prediction*, *LLM-Judge*, and *GRPO* baselines across all budget levels (K=1, 4, 8, 16, 32, 64), with the advantage most pronounced at small budgets where accurate difficulty estimation matters most. Figure 9 also shows that the set of problems solved by SNIS difficulty estimation overlaps strongly with baseline coverage while still including additional uniquely solved examples.

Table 13: Training efficiency on SmolLM3-3B-Base (same metrics as Table 12). *GRPO* does not reach τ=0.73 on MATH-500 or τ=0.85 on GSM8K within the common horizon, so step speedups are undefined and we report the absolute step S_τ only.

	MATH-500 (H=98)		GSM8K (H=98)	
Method	AUC	S_{73}	AUC	S_{85}
GRPO	68.6	–	79.7	–
DOTS	70.4	44	80.7	22
EDCO	69.9	44	82.2	14
LLM-Judge	69.0	62	80.9	34
MoPPS	69.6	50	82.1	16
Previous FR Estimation	69.1	34	80.4	84
DARE (Ours)	**71.0**	**24**	**82.8**	**14**

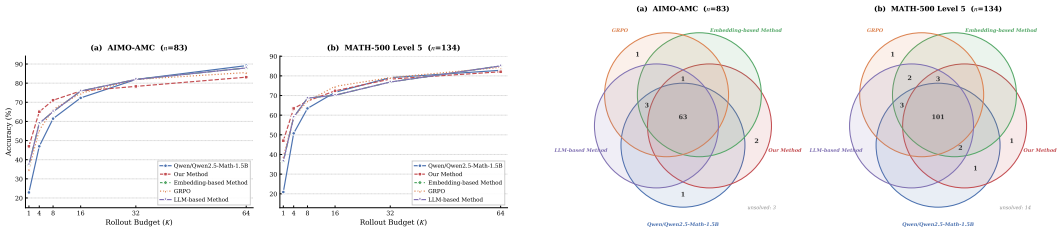


Figure 9: **Left**: Accuracy under constrained rollout budgets on AIME-AMC and MATH-500 Level 5. Our difficulty estimation method achieves consistently higher accuracy, with the largest gains at small budgets. **Right**: Venn diagram of problems solved at K=64. Our difficulty estimation covers most problems solved by any baseline while uniquely solving additional examples.

L IMPACT STATEMENT

This work proposes a difficulty-adaptive RL fine-tuning framework that improves the sample efficiency of training LLM-based mathematical reasoners. On the positive side, reducing the number of rollouts required to reach a given accuracy level directly lowers the compute and energy cost of RL fine-tuning, which benefits researchers with smaller hardware budgets and reduces the environmental footprint of training strong reasoning models; it also makes it more tractable to iterate on reward design and curriculum strategies. On the negative side, our method makes capable mathematical-reasoning LLMs cheaper to obtain, which could amplify existing concerns around misuse of reasoning models (for example, automated assistance with academic assessments). These risks are properties of capable reasoning LLMs in general rather than risks introduced by the proposed training algorithm, and we do not release any new pretrained model weights as part of this work.