Dynamic Adaptive Curriculum for Reinforcement Fine-Tuning of Large Language Models

Anonymous ACL submission

Abstract

Reinforcement learning (RL) alignment improves the reasoning capabilities of large language models (LLMs), but its sample efficiency heavily depends on the sequence of training data. While curriculum learning methods attempt to order tasks by difficulty, they typically rely on static difficulty labels that become obsolete as the model's policy evolves. We introduce Dynamic Adaptive Curriculum (DAC), a framework that continuously updates task difficulty estimates during training. DAC operates by balancing exploitation (sampling tasks at the boundary of the model's current competence) and exploration (revisiting tasks to prevent catastrophic forgetting). The method can leverage prior difficulty labels or operate entirely end-to-end from a label-free initialization. We evaluate DAC on mathematical reasoning benchmarks to assess its impact on sample efficiency and final accuracy compared to static curriculum baselines.

1 Introduction

Reinforcement Fine-Tuning (RFT) algorithms, such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024), have proven effective for improving the reasoning capabilities of Large Language Models (LLMs). However, RFT requires substantial computational resources. A major bottleneck is data efficiency: training on tasks that are either too easy or too difficult for the current policy yields weak learning signals and slows down optimization. Curriculum learning addresses this by ordering the training distribution, but scaling it to dynamic LLM fine-tuning presents specific challenges.

Prior work in active learning suggests that models learn most efficiently from tasks at their boundary of competence, where the probability of success is near 0.5. Static curricula attempt to target this zone using fixed initial difficulty labels. However, because the model's policy π_θ changes continuously during fine-tuning, initial labels become stale. As training progresses, heavily sampled complex tasks become trivial, while unvisited tasks may be forgotten. Relying on fixed initial rankings makes the curriculum vulnerable to this breakdown in relative task ordering.

To address these limitations, we propose Dynamic Adaptive Curriculum (DAC). DAC updates task difficulties dynamically during training using an Exponential Moving Average (EMA) based on real-time rewards, ensuring the curriculum reflects the current policy state. Task selection is formulated as a mixture distribution that allows the model to exploit high-signal tasks while exploring the task pool to update stale metrics and prevent catastrophic forgetting.

DAC can operate using prior labels or entirely end-to-end. By starting all tasks at a neutral baseline, the algorithm discovers the task difficulty space during training. We evaluate our approach on mathematical reasoning benchmarks to benchmark its performance against uniform and static curriculum strategies.

2 Related Work

Reinforcement Fine-Tuning. Reinforcement Learning from Human Feedback (RLHF) has become the standard paradigm for aligning language models (Ouyang et al., 2022; Bai et al., 2022). Recently, Reinforcement Fine-Tuning (RFT) has emerged as a powerful extension for domains requiring rigorous logical consistency, such as mathematics and code generation, where sparse, rule-based rewards are available (Yuan et al., 2023; Luo et al., 2023). Traditional RFT relies on Proximal Policy Optimization (PPO) (Schulman et al., 2017), which requires maintaining a parameterized critic network, imposing significant memory overhead. Group Relative

Policy Optimization (GRPO) (Shao et al., 2024) mitigates this by eliminating the critic and instead utilizing the mean and standard deviation of rewards across a group of sampled responses. Beyond memory efficiency, the group average reward in GRPO provides a natural, zero-cost estimator of the model's current competence on specific tasks, which we leverage in our dynamic difficulty metric.

Curriculum Learning and Task Prioritization. Curriculum Learning (CL) (Bengio et al., 2009) hypothesizes that neural networks train more effectively when presented with examples in an order of increasing difficulty. In reinforcement learning, Unsupervised Environment Design (UED) extends CL by dynamically generating or sampling environments that maximize the agent's learning potential. A seminal approach in this area is Prioritized Level Replay (PLR) (Jiang et al., 2021), which prioritizes the replay of training levels based on learning progress (e.g., temporal difference error) and integrates a staleness mechanism to prevent the starvation of previously visited states. Our work adapts the principles of PLR to the domain of LLM fine-tuning, replacing TD-error with policy competence metrics.

Curriculum Learning in LLMs. Curriculum learning for LLMs aims to maximize the useful learning signal per training step by presenting tasks in an order aligned with the model's current capability. Task difficulty can be estimated heuristically (prompt length, number of reasoning steps, or reference-model loss) or derived from model feedback during training. AdaRFT (Shi et al., 2025) samples tasks whose success rate lies near a target threshold; DUMP (Wang et al., 2025), SEC (Chen et al., 2025), and VCRL (Jiang et al., 2025) score items using advantage-derived statistics; and MR-ACL (Liu et al., 2026) learns a sampling policy over a multi-reward space.

Static Curricula in LLMs and their Limitations. More closely related to our setting is AdaRFT (Shi et al., 2025), which adapts curriculum learning for LLM reasoning tasks. AdaRFT proposes sampling tasks from a sliding window where the model's success probability is approximately 0.5, aiming to maximize the learning signal. However, the sliding window mechanism suffers from an aggregation fallacy: an average score of 0.5 within a window does not guarantee that individual tasks are near the 0.5 threshold. A window might contain a mixture of trivial tasks (success probability near 1.0) and impossible tasks (success probability near 0.0), leading to wasted computational resources. Our dynamic curriculum addresses this by continuously updating difficulty metrics based on model outputs and replacing the heuristic window with a probabilistic mixture distribution.

3 Dynamic Adaptive Curriculum

3.1 Problem Formulation

In Reinforcement Fine-Tuning (RFT), we aim to optimize a language model policy π_θ to maximize the expected reward on a given task dataset $\mathcal{D} = \{x_1, \ldots, x_N\}$. For a given prompt x_i, the model generates a response $y \sim \pi_\theta(\cdot|x_i)$ and receives a rule-based reward $r(x_i, y) \in [0, 1]$. The standard RFT objective is to maximize the expected reward over a uniform sampling of the dataset:

$$\max_\theta \mathbb{E}_{x\sim\mathcal{U}(\mathcal{D}),y\sim\pi_\theta(\cdot|x)}[r(x, y)] \tag{1}$$

However, optimizing this objective uniformly is highly sample-inefficient. If a task x_i is too easy ($r \approx 1$) or too hard ($r \approx 0$) for the current policy π_θ, the resulting gradient provides a near-zero learning signal. Curriculum learning modifies this objective by replacing the uniform distribution $\mathcal{U}(\mathcal{D})$ with a dynamic sampling distribution $P(x|\pi_\theta)$ that prioritizes tasks yielding the highest learning signal (typically where the expected reward is near 0.5). Our goal is to design a sampling distribution P that continuously adapts to the evolving policy π_θ without relying on static, pre-computed difficulty labels.

To resolve the limitations of static curricula, DAC continuously calibrates task difficulty and balances the exploitation of high-signal tasks with the exploration of the broader task pool.

3.2 Dynamic Difficulty Calibration

We utilize GRPO not only as the optimization algorithm but also as the mechanism for difficulty estimation. For each prompt in a training batch, GRPO generates a group of k responses (typically $k = 8$). The average reward across this group, $r_{avg} \in [0, 1]$, serves as a Monte Carlo approximation of the model's current capability to solve the specific task.

Methods relying on static labels, such as AdaRFT, face a fundamental issue: these labels at some moment stop reflecting the real difficulty

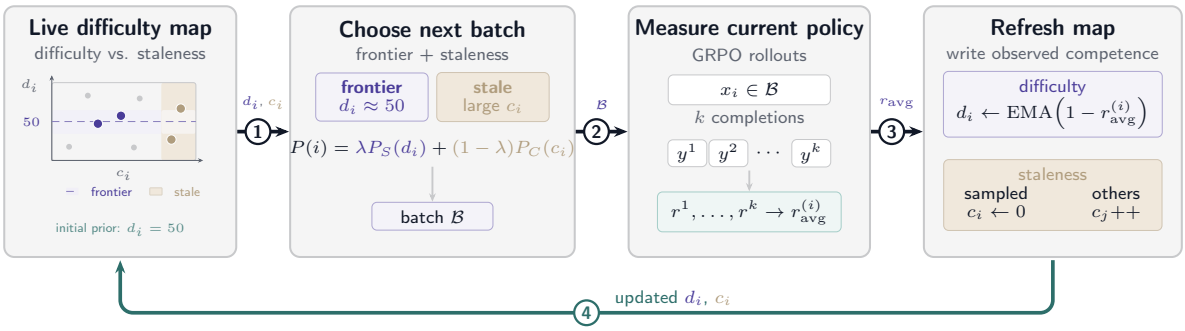

Figure 1: The DAC Framework. Tasks are sampled using a mixture of exploitation (P_s, targeting the $p \approx 0.5$ success zone) and exploration (P_c, preventing starvation). After each GRPO step, task difficulties are updated via an EMA mechanism based on the observed rewards.

of tasks. As the model learns, initially hard tasks become easy. If the curriculum continues to trust the initial labels, it will sample tasks based on outdated information, slowing down training.

To ensure the difficulty metrics reflect the evolving policy, the difficulty d_i for each sampled task is dynamically updated after every forward pass using an Exponential Moving Average (EMA):

$$d_i \leftarrow \alpha d_i + (1-\alpha)(1 - r_{avg}) \cdot 100 \qquad (2)$$

where $\alpha \in [0, 1]$ is the momentum coefficient. This smoothing protects the difficulty distribution from the high variance of individual batches while adapting the metric map to the model's current state.

3.3 Exploration-Exploitation Sampling

Exploitation Distribution (P_s). The highest sampling probability is assigned to tasks where $d_i \approx 50$ (corresponding to a success probability of 0.5). We use an L1 deviation (`score_type=l1`, $q = 1$) to form a Laplace-like distribution, applying a strict penalty to tasks that are either too easy or too hard:

$$P_s(i) = \frac{\exp\left(-\beta \left|\frac{d_i}{100} - 0.5\right|\right)}{\sum_{j=1}^{N} \exp\left(-\beta \left|\frac{d_j}{100} - 0.5\right|\right)} \qquad (3)$$

For a typical dataset of ~8,000 tasks, computing this distribution introduces negligible computational overhead.

Exploration Distribution (P_c). Another critical vulnerability of sliding-window methods like AdaRFT is window stagnation. As training progresses and the model's learning rate plateaus, the overall success rate stabilizes. Consequently, the sliding window stops moving and gets stuck around a specific difficulty level. The model ends up repeatedly sampling from the same narrow subset of tasks. This may lead to overfitting on that specific subset and forgetting of tasks outside the window (e.g., easier tasks seen early in training).

To address this, we draw inspiration from Prioritized Level Replay (PLR) (Jiang et al., 2021), a method from reinforcement learning that faced similar starvation issues when prioritizing environments based on learning progress. PLR solved this by tracking how long an environment had been ignored. We adapt this principle to LLM fine-tuning. If a task with an initially high difficulty metric becomes easy for the updated policy, the algorithm remains unaware because its P_s is near zero, preventing it from being sampled and updated. To mitigate this and prevent stagnation, we introduce a mechanism to force the model to periodically explore unvisited states using staleness counters c_i. The sampling probability based on staleness is calculated as:

$$P_c(i) = \frac{c_i}{\sum_{j=1}^{N} c_j} \qquad (4)$$

where c_i is incremented at each global training step and resets to zero when task i is sampled. The longer a task remains untouched, the higher its priority in the exploration distribution.

3.4 Final Task Distribution

The final probability of selecting task i for the next training batch is a linear combination of the exploitation and exploration distributions:

$$P(i) = \lambda P_s(i) + (1-\lambda) P_c(i) \qquad (5)$$

where $\lambda \in [0, 1]$ controls the trade-off. For example, with $\lambda = 0.8$, the majority of computational

Algorithm 1 Dynamic Adaptive Curriculum (DAC)

Input: task pool $\mathcal{D} = \{x_i\}_{i=1}^N$, policy model π_θ, reward function $R(\cdot,\cdot)$
Input: batch size B, group size k, EMA momentum α, frontier temperature β, mixture weight λ
Input: boolean flag FIRSTTOUCH (optional; default **True**)
$d_i \leftarrow 50,\ c_i \leftarrow 0,\ \text{seen}_i \leftarrow \text{FALSE} \quad \forall i \in \{1,\dots,N\}$ ▷ Initialize difficulty, staleness, and coverage maps
while training is not finished **do**
  $c_i \leftarrow c_i + 1 \quad \forall i$ ▷ Age unsampled tasks
  $w_s(i) \leftarrow \exp(-\beta\,|d_i/100 - 0.5|) \quad \forall i$ ▷ Score frontier tasks
  $P_s(i) \leftarrow w_s(i)/\sum_{j=1}^N w_s(j); \quad P_c(i) \leftarrow c_i/\sum_{j=1}^N c_j$ ▷ Normalize exploitation and exploration
  $P(i) \leftarrow \lambda P_s(i) + (1-\lambda)P_c(i)$ ▷ Mixed sampler
  $\mathcal{B} \leftarrow \text{Sample}_B(\mathcal{D}; P)$ ▷ Draw next training batch
  for each $x_i \in \mathcal{B}$ **do**
    $Y_i \leftarrow \{y_i^{(m)} \sim \pi_\theta(\cdot \mid x_i)\}_{m=1}^k$ ▷ Generate GRPO rollouts
    $r_{\text{avg}}^{(i)} \leftarrow \frac{1}{k}\sum_{m=1}^k R(x_i, y_i^{(m)})$ ▷ Estimate current competence
    if FIRSTTOUCH **and** $\text{seen}_i = \text{FALSE}$ **then**
      $d_i \leftarrow \big(1 - r_{\text{avg}}^{(i)}\big)\cdot 100$ ▷ First-touch replace (no EMA)
    else
      $d_i \leftarrow \alpha d_i + (1-\alpha)\big(1 - r_{\text{avg}}^{(i)}\big)\cdot 100$ ▷ Refresh difficulty via EMA
    end if
    $\text{seen}_i \leftarrow \text{TRUE}; \quad c_i \leftarrow 0$ ▷ Mark task as seen; reset staleness
  end for
  $\pi_\theta \leftarrow \mathcal{A}_{\text{GRPO}}(\pi_\theta; \mathcal{B})$ ▷ Update policy from rollout rewards
end while

resources are directed toward boundary tasks, while the remaining fraction is dedicated to exploration, ensuring the global accuracy of the difficulty map and preventing catastrophic forgetting.

3.5 Frontier Sufficiency Condition

The mixture sampling above is well-defined only if a sufficient pool of tasks remains inside the target frontier $W = [D_{\min}, D_{\max}]$ at stationarity. Since each estimate d_i is noisy, some true-frontier tasks drift out of the corridor, and if there are too many of those tasks, DAC sampler must then fill the batch from outside this interval. In Appendix E we give an intuitive AR(1) argument for this trade-off, which yields the following **Frontier Sufficiency** rule of thumb:

$$\bar{P}_{\text{remain}} \;\geq\; \bar{P}_{\text{crit}} \;:=\; \frac{100\,B}{N\,\Delta}, \tag{6}$$

where $\bar{P}_{\text{remain}}$ is the stationary probability that a true-frontier task is retained inside the corridor, B is the batch size, N is the pool size, and $\Delta = D_{\max} - D_{\min}$ is the corridor width. When (6) is violated, the in-corridor pool is smaller than a single batch, and the curriculum is forced to resample or pull off-target items – a regime we observe empirically as schedule collapse.

4 Experiments

Our experimental evaluation is organized around the following research questions:

- **RQ1 (Stagnation).** Does the sliding-window heuristic of AdaRFT suffer from window stagnation during real LLM fine-tuning, trapping the curriculum in a narrow subset of tasks? (subsection 4.1)
- **RQ2 (Robustness).** Under shifting task difficulties, can DAC reliably track the optimal difficulty frontier where static curricula fail? (subsection 4.2)
- **RQ3 (Performance without labels).** Can the label-free DAC-E2E configuration match the final accuracy of label-dependent static baselines on mathematical reasoning benchmarks? (subsection 4.4)
- **RQ4 (Curriculum dynamics).** How do the training reward and the estimated difficulty distribution evolve over the course of fine-tuning, and does DAC autonomously converge

toward the optimal learning zone? (subsection 4.5, subsection 4.6)

4.1 Curriculum Dynamics and Stagnation

We first examine the training dynamics to verify the window stagnation problem of AdaRFT described earlier. For this specific analysis, both AdaRFT and DAC are provided with static initial difficulty labels d_i (i.e., we evaluate the DAC configuration with initial labels). These prior dataset labels were pre-computed using a larger annotator model (`Qwen2.5-7B`). Figure 2 visualizes the tasks sampled by AdaRFT during the real fine-tuning of the `Qwen2.5-7B` model with a batch size of 64.

The empirical results confirm our hypothesis. While the sliding window initially moves as the model's performance improves, it quickly gets stuck once the average reward stabilizes. From that point on, AdaRFT repeatedly samples from the exact same narrow slice of the dataset. This stagnation prevents the model from revisiting earlier tasks (risking catastrophic forgetting) and completely ignores tasks outside the stalled window.

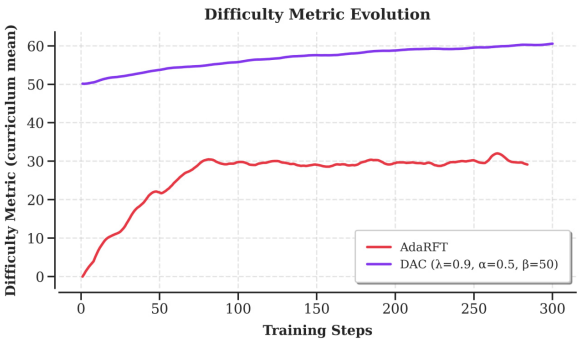

Figure 2: Sampling distribution of AdaRFT over time (`Qwen2.5-7B`, batch size 64). The sliding window stagnates, trapping the curriculum in a narrow subset of tasks and starving the rest of the dataset.

4.2 Robustness Analysis via Simulation

Evaluating the robustness of curriculum algorithms across multiple LLM fine-tuning runs is computationally prohibitive. To overcome this, we design a simulated environment to test how well the algorithms handle changing task difficulties.

In this simulation, the model's true competence on a task i is governed by a hidden variable, the true difficulty $d_{i,true} \in [0, 100]$. The probability of the model successfully solving the task is simply $p_i = (100 - d_{i,true})/100$. This is the true difficulty that our dynamic curriculum tries to estimate using GRPO rewards.

To simulate the model learning new skills or forgetting old ones, we apply a random drift to the true difficulties at each training step: $d_{i,true}^{(t)} = d_{i,true}^{(t-1)} + \mathcal{N}(\mu, \sigma^2)$. By running curriculum algorithms in this shifting environment, we can measure the Target Acquisition Rate (the proportion of sampled tasks that fall within the optimal difficulty zone) and empirically prove that DAC avoids the window stagnation problem. We provide a detailed breakdown of this simulation, including the exact parameters and justification for the noise models, in Appendix B.

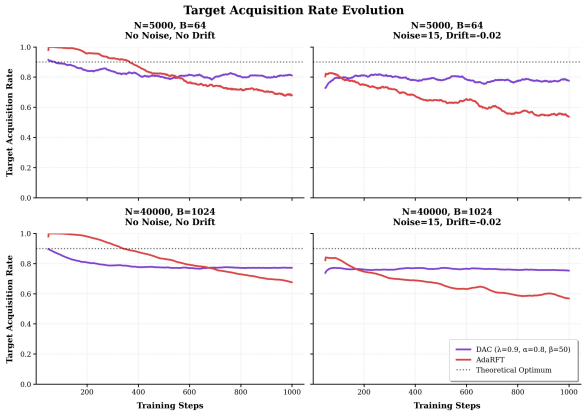

Figure 3: Target Acquisition Rate in the simulated environment across stationary and dynamic scenarios. DAC consistently tracks the optimal difficulty frontier. Detailed parameters are provided in Appendix B.

4.3 LLM Fine-Tuning Setup

To isolate the effect of sampling strategies, we fix the base model, training data, and optimization budget across all runs. Full hyperparameters and reproducibility details are provided in Appendix A. The dataset is available at `https://huggingface.co/datasets/lime-nlp/DeepScaleR_Difficulty`.

Model and Optimization. We use `Qwen/Qwen2.5-Math-1.5B` as our base model. Policy optimization is performed via Group Relative Policy Optimization (GRPO) with a group size of $k = 8$, estimating advantages directly from intra-group statistics. A low-variance KL divergence penalty (coefficient 10^{-3}) is applied within the actor loss. Models are fine-tuned via FSDP on 4× H100 GPUs for 100 optimizer steps, utilizing a global batch size of 1024, a learning rate of 10^{-6}, and the AdamW optimizer.

Training Data and Uniform Constraint. All configurations are trained on a fixed dataset of ∼8,000 mathematical problems sourced from the DeepScaleR-Difficulty collection (Luo et al., 2025). Crucially, we enforce a strict **uniform distribution** across all difficulty deciles. By strictly uniformly distributing the initial pool, we ensure that any performance gains stem entirely from the curriculum's active sampling mechanism.

Compared Methods We evaluate our approach against two primary baselines: a Uniform lower bound with no active curriculum, and AdaRFT, a state-of-the-art sliding-window method that relies on static dataset labels. For our method, we specifically evaluate the label-free **DAC-E2E** configuration. In this setup, the algorithm is initialized entirely uniformly (all tasks start with a neutral difficulty of 50) and must autonomously discover the difficulty landscape. We chose this end-to-end configuration to demonstrate that DAC can match the performance of label-dependent methods like AdaRFT without requiring any prior difficulty annotations.

Evaluation Protocol During training, we monitor progress by performing validation exclusively on the GSM8K test set (∼1,319 problems) every 5 optimization steps. After the completion of 100 training steps, FSDP shards are merged into native HuggingFace formats for evaluation via vLLM.

Final post-hoc performance is reported on six held-out mathematical benchmarks: MATH-500, GSM8K, OlympiadBench, Minerva Math, AIME 2024, and AMC 2023. We append the instruction *"Please reason step by step, and put your final answer within \boxed{}."* to all evaluation prompts. For the first four standard benchmarks, we use greedy decoding (pass@1, $T = 0$). For the competition-level datasets (AIME 2024, AMC 2023), we sample 8 responses per problem at $T = 1.0$ and report the average accuracy (Avg@8) to ensure stable estimates. We employ strict rule-based extraction (matching #### for GSM8K or parsing `\boxed{}`) followed by Hendrycks-style symbolic equivalence checking.

Post-hoc evaluation is reported for Uniform, AdaRFT, and DAC-E2E.

4.4 Main Results: Mathematical Reasoning

Table 1 presents the final accuracy of the fine-tuned models across the evaluated mathematical reasoning benchmarks. To ensure statistical reliability, the reported scores for DAC-E2E and AdaRFT represent the average performance across three independent training runs with different random seeds, accompanied by confidence intervals calculated via a t-test. The Uniform baseline was run once to establish a standard empirical lower bound, as previous work has already demonstrated that AdaRFT outperforms uniform sampling. For the DAC configuration, the curriculum was driven by $k = 8$ rollouts per prompt for difficulty estimation and an EMA momentum of $\alpha = 0.5$. These parameters were chosen to keep the curriculum within the stability regime discussed in Appendix E.

The results show that the label-free DAC-E2E method achieves an improvement in average accuracy (37.29%) compared to both the AdaRFT baseline (36.86%) and the Uniform strategy (35.96%). DAC performs best on three out of the six individual benchmarks (GSM8K, MATH500, and AMC23).

Importantly, DAC reaches this performance without requiring prior difficulty labels, bypassing the need for an external annotation pipeline. By dynamically discovering task difficulties from a neutral initialization, DAC avoids the stagnation risks associated with static sliding-window methods. This provides a more compute-efficient approach that can be readily applied to unannotated reasoning datasets.

4.5 Curriculum Reward Evolution

Figure 4 illustrates the mean training reward across the fine-tuning process. The AdaRFT algorithm, which utilizes prior dataset labels, maintains a reward of ∼0.5 from the very first step. In contrast, the label-free DAC-E2E configuration starts with zero prior knowledge, initially matching the Uniform baseline's random sampling reward of ∼0.25. The plot demonstrates that DAC requires a brief initial exploration phase to map the difficulty landscape. Within approximately 20 steps, the dynamic difficulty updates autonomously drive the training reward toward the optimal learning zone (∼0.4).

4.6 Curriculum Difficulty Evolution

For the **DAC-E2E (First-Touch)** configuration, the difficulty curve exhibits a distinct "hump" (Figure 5). During the first 8–10 steps, the average batch difficulty rises monotonically from the neutral initialization ($d = 50$) to a peak of ∼75, followed by a steady decay toward a stable state of $d \approx 60$. This initial rise is an artifact of the "first

Method	GSM8K	MATH500	AMC23	Olympiad	Minerva	AIME24	Avg
Uniform	77.79	64.40	39.06	21.96	8.82	3.75	35.96
AdaRFT	77.76±1.75	64.80±1.31	41.25±3.88	23.24±0.85	7.97±0.53	6.11±2.15	36.86±1.06
DAC	78.19±0.76	65.27±2.74	43.33±7.04	22.90±1.49	8.21±1.40	5.83±1.04	37.29±1.87

Table 1: Final accuracy on mathematical reasoning benchmarks. Scores for DAC-E2E and AdaRFT represent the mean and confidence intervals (t-test) across 3 independent runs with different random seeds. The Uniform baseline is reported from a single run. DAC configurations utilize $k = 8$ rollouts and an EMA momentum of $\alpha = 0.5$.

Figure 4: Mean training reward over 100 steps. DAC requires an initial exploration phase from a random start but autonomously converges toward the optimal difficulty zone.

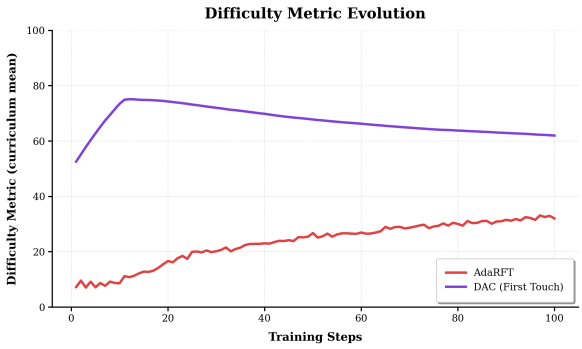

Figure 5: Evolution of the average batch difficulty over 100 training steps. The DAC-E2E (First-Touch) configuration exhibits a distinct "hump" due to initial dataset exploration, while AdaRFT slowly plateaus.

touch" exploration phase interacting with formatting mismatches between the annotator and the base model; a detailed mechanical explanation of this phenomenon is provided in Appendix F.

In contrast, the **AdaRFT** baseline ($\eta = 50$) demonstrates a gradual increase that begins to plateau early. Starting from a low average difficulty ($\sim$10), it only reaches $\sim$30 by step 100, with the rate of increase noticeably slowing. This trajectory highlights a limitation of its sliding-window heuristic: within the constrained 100-step budget, the window advances incrementally through the easier portion of the dataset.

5 Conclusion

In this paper, we introduced the Dynamic Adaptive Curriculum (DAC) to improve the sample efficiency of Reinforcement Fine-Tuning. We designed DAC to address specific issues in existing static curriculum methods: the rapid obsolescence of initial difficulty labels, the stagnation of sliding windows, and the aggregation fallacy where average window scores mask a mix of trivial and impossible tasks.

DAC addresses these problems by continuously updating task difficulties using real-time GRPO rewards. By balancing the exploitation of boundary tasks with a staleness-based exploration mechanism, DAC maintains a steady learning signal and prevents the curriculum from stalling on a narrow subset of tasks.

Our experiments show that the label-free DAC-E2E configuration achieves a higher average accuracy on mathematical reasoning benchmarks than the AdaRFT baseline. Because DAC can operate from a neutral initialization, it eliminates the need for prior dataset annotation. By dynamically discovering task difficulties, DAC provides a compute-efficient framework that can be applied to unannotated datasets.

6 Limitations

While the proposed Dynamic Curriculum effectively mitigates schedule collapse and provides robust frontier tracking, the stability argument in Appendix E highlights several inherent limitations and structural trade-offs that must be considered when deploying the algorithm in practice.

The Responsiveness-Variance Trade-off. The core mechanism relies on an Exponential Moving Average (EMA) to update task difficulties. This creates a strict trade-off controlled by the momentum parameter α. If α is small, the curriculum reacts quickly to the model's learning progress. However, because the GRPO rewards are inherently noisy (based on a small group of k rollouts), a small α

causes the estimated difficulties to oscillate wildly. This noise can misrepresent the true value of tasks and artificially shrink the pool of suitable tasks. In severe cases, this variance can cause the algorithm to run out of tasks in the target zone, leading to the very stagnation it was designed to prevent.

Hyperparameter Sensitivity. The framework introduces a complex, tightly coupled hyperparameter space. The stability of the curriculum depends delicately on the interplay between the EMA momentum α, the GRPO group size k, the exploitation temperature β, and the mixing coefficient λ. This entanglement makes the algorithm non-trivial to tune for novel datasets or model architectures. Debugging curriculum instability is challenging because a failure to maintain the target frontier can stem from multiple entangled sources–e.g., α being too low, or the model's capabilities drifting faster than the EMA can track. This imposes an additional computational burden for hyperparameter calibration prior to full-scale training.

Sensitivity to Dataset Skewness. DAC assumes that there are always enough tasks available near the model's current boundary of competence. If the underlying dataset is pathologically skewed (e.g., the vast majority of tasks are either overwhelmingly hard or entirely trivial for the specific model), this assumption breaks down. In such regimes, the curriculum struggles to find a sufficient volume of tasks in the optimal difficulty zone. If the model's frontier moves into a sparse region of the dataset, the algorithm is forced to either sample off-target tasks or repeatedly sample the same small subset, inadvertently degrading into the very stagnation it was intended to solve.

7 Ethical Considerations

We use only publicly released research artifacts: the DeepScaleR-Difficulty mathematical reasoning dataset, public mathematical reasoning benchmarks, and the Qwen2.5-Math-1.5B checkpoint. Our use is consistent with their intended purpose: training and evaluating methods for mathematical reasoning and reinforcement fine-tuning. Any derived artifacts produced by our experiments, including checkpoints, curriculum traces, task difficulty estimates, and rollout statistics, are intended for research use and should not be used for high-stakes decision-making or decisions about individual people.

The data used in this work consists of mathematical problem statements, answers, and difficulty-related metadata. We do not collect private user data. Our training pipeline stores only prompts, model completions, rewards, and aggregate curriculum statistics.

We used AI writing assistance to polish language and improve clarity, while all technical content, experimental design, analysis, and final claims were reviewed and controlled by the authors.

References

Yuntao Bai and 1 others. 2022. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

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

Guochao Jiang, Wenfeng Feng, Guofeng Quan, Chuzhan Hao, Yuewei Zhang, Guohua Liu, and Hao Wang. 2025. Vcrl: Variance-based curriculum reinforcement learning for large language models. *arXiv preprint arXiv:2509.19803*.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. 2021. Prioritized level replay. In *International Conference on Machine Learning (ICML)*, pages 4940–4950. PMLR.

Wenxuan Liu, Liangyu Huo, Yi Jing, Xiyuan Zhang, and Jian Xie. 2026. Mracl: Multi-reward space guided adaptive curriculum reinforcement learning for llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 37663–37672.

Haipeng Luo and 1 others. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Michael Luo, Xinyang Shi, Chi Cai, Tianyi Zhang, Jiahui Wong, Yiyang Wang, Chi Wang, and Yuxuan Huang. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. *arXiv preprint arXiv:2502.13965*.

Long Ouyang and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

John Schulman and 1 others. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng and 1 others. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. 2025. Adarft: Adaptive reward-driven reinforcement fine-tuning. *arXiv preprint arXiv:2504.05520*.

Zhenting Wang, Guofeng Cui, Yu-Jhe Li, Kun Wan, and Wentian Zhao. 2025. Dump: Automated distribution-level curriculum learning for rl-based llm post-training. *arXiv preprint arXiv:2504.09710*.

Zheng Yuan and 1 others. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

A Detailed Training Hyperparameters

To ensure complete reproducibility of our GRPO training pipeline, we provide the comprehensive list of global optimization parameters in Table 2. These settings remain strictly constant across all experimental setups to isolate the impact of the curriculum samplers.

Table 2: Comprehensive GRPO and system hyperparameters.

Parameter	Value
Model Architecture	Qwen/Qwen2.5-Math-1.5B
Optimization Strategy	Full fine-tune (FSDP)
Hardware	$4\times$ NVIDIA H100, 1 node
Optimizer	AdamW
Learning Rate	10^{-6}
Train Batch Size	1024 prompts per step
GRPO Group Size (k)	8
KL Divergence Coef	10^{-3} (actor loss, low-variance)
Max Prompt Length	1024 tokens
Max Response Length	3000 tokens
Total Training Steps	100
Dynamic Batching	Enabled (`use_dynamic_bsz=True`)
Rollout Engine	vLLM (`tp=1`)
GPU Memory Util	0.7
GPU compute	about 100 hours

B Simulation Details and Trajectory Analysis

To empirically validate the robustness of DAC against the window stagnation problem observed in AdaRFT, we designed the simulation environment described in Section 4.2.

Figure 3 presents the results of this simulation across four different configurations. The top row simulates a smaller pool of 5,000 tasks with a batch size of 64, while the bottom row scales to 40,000 tasks with a batch size of 1024. The left column represents a stationary baseline where initial difficulty labels perfectly match the true difficulty (mean shift of 0) and the drift is zero-centered ($\mu = 0$).

The right column introduces a more realistic, challenging scenario: initial labels are systematically shifted (mean shift of 15), and the true difficulty drifts downwards with $\mu = -0.02$. We introduce a mean shift of 15 to the initial difficulty labels to realistically model the noise in static dataset annotations. In practice, human-annotated or heuristic-based difficulty labels are often systematically misaligned with a specific LLM's actual competence. This shift represents the initial calibration error. Furthermore, a drift of -0.02 means that over 100 training steps, the average true difficulty

of a task decreases by exactly 2 points, accurately modeling the gradual improvement of the LLM's capabilities during fine-tuning. Across all settings, DAC successfully maintains a high target acquisition rate near the theoretical optimum, whereas AdaRFT inevitably stagnates.

C Extended Curriculum Configurations and Ablations

Table 3 expands upon the core configurations presented in the main text, detailing the exact sampling hyperparameters (α, β, λ) for all six experimental setups, including our ablation studies.

The **DAC (Alt Temp)** setup serves as an ablation for the exploitation distribution, utilizing a lower temperature ($\beta = 25$) to create a broader, softer probability peak around the 0.5 difficulty threshold compared to the primary DAC configuration ($\beta = 50$). The **DAC-E2E (First-Touch)** ablation modifies the initial difficulty assignment in label-free settings: instead of smoothly blending the neutral prior ($d_i = 50$) with the first observed reward via EMA, it completely replaces the initial difficulty with $(1 - r_{\text{avg}}) \cdot 100$ upon the first encounter. This forces a rapid initial dispersion of tasks across the difficulty space.

D Implementation and Reproducibility

An anonymized implementation is available at https://anonymous.4open.science/r/DAC-Dynamic-Adaptive-Curriculum-B465. The training stack is built upon the `verl` framework, which is based on the HybridFlow architecture (Sheng et al., 2024), and orchestrated via Ray.

The training stack is built upon the `verl` framework, orchestrated via Ray. The actor weights are partitioned using Fully Sharded Data Parallel (FSDP) to fit the optimizer states and gradients within the 80GB VRAM limits during the backward pass, while vLLM handles the high-throughput forward rollout generation. The global random seed is fixed to 42 for all runs.

To maintain a globally consistent curriculum state during training, the DAC mechanism is implemented via a custom trainer hook. When DAC is active, the training DataLoader is strictly instantiated with `num_workers=0` to prevent in-place state desynchronization across spawned processes. We enforce `val_before_train=True` to capture a strict zero-shot baseline prior to curriculum activation. In-training validation evaluates strictly on the GSM8K test split. The KL divergence penalty (`use_kl_in_reward=False`) is computed directly within the actor loss rather than modifying the environment reward signal. Following the completion of the 100 optimization steps, FSDP shards are consolidated into standard HuggingFace checkpoint formats to facilitate evaluation.

E Why the Frontier Can Run Dry: A Stability Intuition

This appendix gives an informal, back-of-the-envelope argument for why DAC needs enough tasks near the model's competence boundary, and how the group size k, EMA momentum α, and corridor width Δ trade off against one another. The argument is deliberately heuristic: it relies on simplifying assumptions — independent rollouts, a roughly uniform spread of task difficulties, static true difficulties on a short horizon, and a Gaussian approximation of the estimator noise — that do *not* hold exactly during real fine-tuning. We use it only to build intuition for the hyperparameter choices in section 3, not as a formal guarantee.

E.1 The difficulty estimate is a noisy moving average

Each task i has an unknown true difficulty $d_{\text{true}} \in [0, 100]$, which we read as a failure rate $p = 1 - d_{\text{true}}/100$. A single GRPO group of k rollouts yields a difficulty observation $U = (1 - X/k) \cdot 100$ with $X \sim \text{Binomial}(k, p)$, so $\mathbb{E}[U] = d_{\text{true}}$ and $\text{Var}(U) = d_{\text{true}}(100 - d_{\text{true}})/k$. The EMA update $d^{(t+1)} = \alpha d^{(t)} + (1 - \alpha)U^{(t)}$ is just an AR(1) recursion. If successive rollouts are treated as independent, the estimate fluctuates around the true value with stationary variance

$$\sigma_\infty^2 \approx \frac{1-\alpha}{1+\alpha} \cdot \frac{d_{\text{true}}(100 - d_{\text{true}})}{k}. \quad (7)$$

The takeaway is simply that d_i never sits exactly on d_{true} — it jitters with standard deviation σ_∞. Two cheap levers shrink that jitter: more rollouts per task (larger k, since the binomial variance scales as $1/k$) and stronger smoothing (larger α). Increasing α is the cheaper of the two — it costs no extra rollouts — but it slows the estimate down: starting from the neutral value $d_i = 50$, the EMA needs on the order of $1/(1 - \alpha)$ updates to forget its initialization and track a moved target.

Table 3: Complete hyperparameter grid for all curriculum setups, mapping to exact repository scripts. For AdaRFT, parameters denote the sliding window heuristic (α: tanh steepness, β: target reward, η: step size). For DAC variants, they denote the probabilistic sampler (α: EMA momentum, β: softmax temperature, λ: exploitation weight, q: L1 score type).

ID	Method Name	Curriculum Mechanism	Initial Difficulty (d_i)	Key Hyperparameters	Role / Notes
1	AdaRFT	Sliding-window	Static (dataset)	β=0.5, α=2, η=50	AdaRFT logic baseline
2	DAC (Dataset Init)	Probabilistic (DAC)	Static (dataset)	β=50, α=0.5, λ=0.9, q=1	Primary DAC with prior labels
3	DAC (Alt Temp)	Probabilistic (DAC)	Static (dataset)	β=25, α=0.6, λ=0.8, q=1	Ablation: softer exploitation peak
4	Uniform	Shuffle	N/A	—	Empirical lower bound
5	DAC-E2E	Probabilistic (DAC)	Uniform ($\forall i, d_i = 50$)	β=40, α=0.5, λ=0.85, q=1	Label-free map discovery
6	DAC-E2E + First-Touch	Probabilistic (DAC)	Uniform ($\forall i, d_i = 50$)	β=50, α=0.5, λ=0.9, q=1	Ablation: replaces d_i on first sample

E.2 Jitter leaks tasks out of the corridor

DAC concentrates sampling on a difficulty corridor $W = [D_{\min}, D_{\max}]$ of width Δ. Because of the jitter in (7), a task whose *true* difficulty lies inside W will, on any given step, sometimes be *estimated* outside it. Approximating the jitter as Gaussian, the fraction $\bar{P}_{\text{remain}}$ of genuinely on-frontier tasks that are still seen as on-frontier is a decreasing function of the noise-to-width ratio σ_∞/Δ: noisier estimates (small k, small α) or a narrower corridor leak more tasks out. Table 4 lists representative values for the corridor used in our runs.

k	α	σ_∞	$\bar{P}_{\text{remain}}$	In-corridor pool
2	0.3	25.9	41.6%	~1000 (below B)
8	0.5	10.2	72.9%	~1750 (default)
8	0.8	5.9	84.3%	~2020
16	0.5	7.2	80.8%	~1940
16	0.8	4.2	88.9%	~2130

Table 4: Illustrative retained fraction $\bar{P}_{\text{remain}}$ and resulting in-corridor pool for N=8000, batch B=1024, corridor W=[35, 65] (Δ=30), computed under a Gaussian approximation of the jitter. Numbers are meant as a qualitative guide, not a measured quantity.

E.3 A sufficiency rule of thumb

If task difficulties are spread roughly uniformly, about $N_{\text{true}} \approx N\Delta/100$ tasks genuinely belong in the corridor. Of those, only a fraction $\bar{P}_{\text{remain}}$ are estimated inside it at any moment. To assemble a full batch of B tasks from the corridor, this retained pool must hold at least one batch, $N\,(\Delta/100)\,\bar{P}_{\text{remain}} \geq B$, which rearranges to

$$\bar{P}_{\text{remain}} \;\geq\; \bar{P}_{\text{crit}} \;:=\; \frac{100\,B}{N\,\Delta}. \qquad (8)$$

We call (8) the *Frontier Sufficiency* rule of thumb. When it is violated, fewer than B tasks look on-target and the sampler must repeat tasks or pull off-corridor items — the schedule collapse discussed in the main text. This is only a rough guide: it ignores the exploration term P_c (which deliberately samples outside the corridor), assumes a uniform spread of difficulties, and treats the jitter as Gaussian. In Table 4 the stress setting (k=2, α=0.3) falls below $\bar{P}_{\text{crit}}$, whereas all settings we actually use stay well above it.

E.4 Reading it as hyperparameter guidance

For our experiments ($N \approx 8000$, $B = 1024$, $\Delta = 30$) the rule gives $\bar{P}_{\text{crit}} \approx 43\%$. The same picture explains the three knobs: k shrinks the per-group noise but with diminishing returns once $\sigma_\infty \ll \Delta$; α is the cheapest way to satisfy (8) since it adds no rollouts, at the cost of a slower warm-up; and widening Δ relaxes *both* sides of the inequality at once. Our default $(k, \alpha) = (8, 0.5)$ leaves the estimated retained fraction (~73%) comfortably above $\bar{P}_{\text{crit}}$, which is consistent with not observing schedule collapse in practice. We stress again that real training violates the idealized assumptions here (e.g. the difficulty "hump" of subsection 4.6), so these numbers should be read as intuition for parameter selection rather than as a tight prediction.

F Mechanics of the Difficulty "Hump"

The distinct "hump" observed in the difficulty evolution of the DAC-E2E (First-Touch) configuration (Figure 5) can be explained by the interaction between the batch size, the dataset size, and the discrepancy between the annotating model and the training model.

With a global batch size of 1024 and a strictly uniform dataset of ~8,000 tasks, it takes approximately 8 steps to sample the entire task pool. Initially, a random batch of unseen tasks averages to a difficulty of 50. However, the annotating Qwen-2.5-7B model frequently fails to format answers within the correct tags compared to our base model (Qwen-2.5-Math-1.5B). Consequently, many tasks appear artificially harder to the 7B annotator than they actually are for the 1.5B Math model.

When our algorithm samples a task and observes a win rate of $\sim$0.5, its updated difficulty remains near 50, making it highly likely to be resampled by the exploitation distribution (P_s). Because of the aforementioned mismatch, these resampled tasks are genuinely harder relative to the 7B annotator's baseline. By continuously mixing these systematically harder, resampled tasks with the remaining unseen uniform tasks in the batch, the average batch difficulty is driven upward until the entire dataset has been explored.

After step 10, the difficulty map is fully populated, and the algorithm begins converging to the true difficulty labels. Because the 1.5B Math model is fundamentally more capable at solving and correctly formatting these specific mathematical tasks than the 7B annotator's initial labels imply, the true average difficulty of the targeted boundary tasks is lower than the artificial peak. Thus, the curve slowly decays toward a stable state of $d \approx 60$.

G Evaluation under Stochastic Decoding

The main results in Table 1 rely on greedy decoding for the four standard benchmarks. To verify that the observed ranking is not an artifact of a single deterministic decoding pass, we additionally evaluate all methods under stochastic decoding. For every problem in each of the six benchmarks, we draw 10 independent samples at temperature 0.6 and report the mean accuracy across these samples (Mean@10). As in the main text, the DAC and AdaRFT scores are averaged over three independent training runs with different random seeds, with confidence intervals computed via a t-test, while the Uniform baseline is reported from a single run. Bold entries mark the best score in each column.

Table 6 reports the results. The overall picture is consistent with the greedy evaluation: DAC attains the highest average accuracy (36.87%), slightly ahead of AdaRFT (36.73%), and both curriculum methods improve over the Uniform baseline (36.28%). The per-benchmark scores of DAC and AdaRFT largely overlap within their confidence intervals, indicating that the two methods are statistically comparable under this protocol; DAC's advantage lies in achieving this competitive accuracy without prior difficulty labels. The wide intervals on AMC23 and AIME24 reflect the small size of these competition-level sets, where Mean@10 estimates remain inherently high-variance.

Mean @10 samples per problem

Method	**GSM8K**	**MATH500**	**AMC23**	**Olympiad**	**Minerva**	**AIME24**	**Avg**
Uniform	70.33	**62.74**	47.00	21.29	9.34	7.00	36.28
AdaRFT	70.52±0.69	62.11±0.83	**48.83±7.20**	**21.49±0.35**	**9.73±0.73**	7.67±2.19	36.73±1.46
DAC	**71.49±1.36**	62.70±0.75	48.17±5.90	20.95±0.56	9.58±0.86	**8.33±3.79**	**36.87±0.57**

Figure 6: Accuracy under stochastic decoding (Mean@10, temperature 0.6) on the six mathematical reasoning benchmarks. Scores for DAC and AdaRFT are the mean and confidence intervals (t-test) over 3 independent runs with different random seeds; the Uniform baseline is reported from a single run. Bold marks the best score per column.