DUMP: DISTRIBUTION-LEVEL CURRICULUM LEARNING FOR RL-BASED LLM POST-TRAINING

Zhenting Wang[1], Guofeng Cui[1], Yu-Jhe Li[2], Kun Wan[2]*, Wentian Zhao[2]*
[1]Rutgers University, [2]Adobe Inc.

ABSTRACT

Recent advances in reinforcement learning (RL)-based post-training have led to notable improvements in large language models (LLMs), particularly in enhancing their reasoning capabilities to handle complex tasks. However, most existing methods treat the training data as a unified whole, overlooking the fact that modern LLM training often involves a mixture of data from diverse distributions—varying in both source and difficulty. This heterogeneity introduces a key challenge: how to adaptively schedule training across distributions to optimize learning efficiency. In this paper, we present a principled curriculum learning framework grounded in the notion of distribution-level learnability. Our core insight is that the magnitude of policy advantages reflects how much a model can still benefit from further training on a given distribution. Based on this, we propose a distribution-level curriculum learning framework for RL-based LLM post-training, which leverages the Upper Confidence Bound (UCB) principle to dynamically adjust sampling probabilities for different distrubutions. This approach prioritizes distributions with either high average advantage (exploitation) or low sample count (exploration), yielding an adaptive and theoretically grounded training schedule. We instantiate our curriculum learning framework with GRPO as the underlying RL algorithm and demonstrate its effectiveness on logic reasoning datasets with multiple difficulties and sources. Our experiments show that our framework significantly improves convergence speed and final performance, highlighting the value of distribution-aware curriculum strategies in LLM post-training. Code: https://github.com/zhentingwang/dump.

1 INTRODUCTION

Reinforcement learning (RL)-based post-training has emerged as a powerful approach for enhancing the capabilities of large language models (LLMs), particularly in areas requiring structured reasoning, multi-step inference, and task-specific generalization (Ouyang et al., 2022; Rafailov et al., 2023; Shao et al., 2024; Guo et al., 2025). By leveraging reward signals derived from task performance, human feedback, or domain-specific metrics, RL provides a flexible alternative to supervised fine-tuning. Unlike imitation-based methods that merely mimic reference outputs, RL-based approaches allow models to optimize directly toward behavioral objectives, making them especially effective for boosting model performance on complex reasoning and agentic tasks.

While RL-based post-training has become a key technique for enhancing LLM capabilities in reasoning, alignment, and coding, one foundational challenge remains underexplored: *how to dynamically schedule training across heterogeneous data distributions*. In practice, LLMs are post-trained on datasets drawn from a wide variety of sources—ranging from factual QA to math problems and coding tasks—each differing in knowledge/capability relevance, and learning difficulty (Longpre et al., 2023; Lee et al., 2023; Lambert et al., 2024). This heterogeneity is evident in large-scale post-training datasets such as Tülu 3 (Lambert et al., 2024), where prompts span general dialogue, logic puzzles, STEM problems, and multilingual instructions, with widely varying counts, formats, and alignment objectives. More recently, *next-generation post-training pipelines* (e.g., Seed-Thinking v1.5 (Seed, 2025)) have shifted toward *synthetic data generation with controllable parameters*—e.g., configuring logical puzzle difficulty. This allows fine-grained control over the *data distribution*,

*Corresponding authors.

making **distribution-level curriculum learning both feasible and increasingly important**. Despite this, most RL-based pipelines still treat all data distributions equally—uniformly sampling tasks throughout training or relying on static, hand-designed curricula. This static treatment ignores the model's evolving learning needs and underutilizes the training budget. Moreover, it is difficult to handcraft effective curricula when the post-training data comes from multiple distributions lacking clear difficulty labels. As reinforcement learning becomes increasingly used in post-training and training costs continue to rise, a data-driven curriculum mechanism that dynamically prioritizes learnable distributions is not just desirable, but necessary.

This motivates the need for *automated distribution-level curriculum learning*: a dynamic strategy that adjusts sampling probabilities across data distributions throughout training. While prior work has explored instance-level curricula based on sample difficulty (Pattnaik et al., 2024), and static/heuristic multi-stage schedules have been applied in LLM post-training (Team et al., 2025; Xie et al., 2025), little attention has been paid to automated, distribution-level scheduling—especially in the context of RL for capability-oriented post-training. The central challenge lies in identifying signals that reflect the current learnability of each distribution and in designing algorithms that can stably and efficiently leverage these signals to guide sampling.

In this paper, we present DUMP (**D**istribution-level c**U**rriculu**M** learning for RL-based LLM **P**ost-training), a simple but theoretically grounded approach to address this challenge. Our central insight is that the *magnitude of policy advantages*—the expected absolute difference between a model's predicted return and its baseline value—serves as a natural proxy for distribution-level learnability. High advantages on specific data distribution indicate underfitting and high potential for improvement on it, while low advantages suggest diminishing returns. Moreover, the statistical reliability of these advantage estimates improves with the number of samples drawn from each distribution. DUMP operationalizes this insight by using bandit-style Upper Confidence Bound (UCB) scores to schedule distribution sampling. It maintains a sliding window of recent advantage magnitudes for each distribution and computes a score that balances exploitation (high advantage) and exploration (low visitation). These scores are normalized via a softmax to form sampling weights, which are then used to generate training batches. Unlike fixed or heuristic curricula, DUMP adapts throughout training based on empirical signals, and can be seamlessly integrated into standard LLM RL pipelines. We instantiate DUMP with GRPO (Shao et al., 2024), but the method is compatible with any advantage-based RL algorithm. We evaluate DUMP on logic reasoning corpora. Our experiments show that DUMP significantly accelerates convergence and yields stronger performance compared to uniform sampling. Furthermore, we provide theoretical analysis that supports the use of absolute advantages as a surrogate for distribution-level learnability, formalizing its connection to sample efficiency and regret minimization.

We summarize our contributions as follows. ① We highlight the underexplored challenge of curriculum learning at the distribution level for RL-based post-training aimed at capability enhancement. ② We propose DUMP, a theoretically grounded framework that leverages advantage-based UCB scores to adaptively guide training over data distributions. ③ We demonstrate DUMP's effectiveness through empirical results and theoretical analysis, showing that it enables faster, more efficient improvement on LLM capabilities.

2 Background

RL-based LLM Post-training. Reinforcement learning (RL) plays a central role in post-training large language models (LLMs), especially for tasks involving reasoning, subjective preference, or long-horizon control. The RLHF framework (Ouyang et al., 2022; Christiano et al., 2017; Ziegler et al., 2019; Bai et al., 2022; Glaese et al., 2022) laid the foundation by aligning models using reward signals derived from human preferences. Beyond preference alignment, recent RL-based post-training approaches have notably enhanced LLMs' capabilities in complex reasoning tasks, particularly coding and mathematics. For instance, RL post-trained model OpenAI o1 (OpenAI, 2024), o3 (OpenAI, 2025; El-Kishky et al., 2025), DeepSeek-R1 (Guo et al., 2025) significantly outperform LLMs without RL post-training such as pre-trained versions of GPT-4o (Hurst et al., 2024) and DeepSeek-V3 (Liu et al., 2024) on challenging mathematics and coding benchmarks, e.g., AIME (diz, 2025) and Codeforces (Mirzayanov). Proximal Policy Optimization (PPO) (Schulman et al., 2017) is widely used in post-training due to its clipped objective, which stabilizes training by preventing large policy updates. PPO remains a strong baseline in many LLM alignment settings. Direct Preference Optimization (DPO) (Rafailov et al., 2023) simplifies the pipeline by replacing RL rollouts with a classification-style loss derived from a KL-constrained reward maximization objective.

While DPO works well on pairwise preference data, it does not naturally support group-wise or comparative feedback. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) addresses this limitation by leveraging group-based feedback. For each input prompt x, GRPO samples a group of G candidate outputs $\{o_1, \ldots, o_G\} \sim \pi_{\text{ref}}(\cdot|x)$ from a frozen reference policy π_{ref}. Each output o_i is assigned a reward r_i, and the advantage of o_i is computed by normalizing its reward relative to others in the group: $\hat{A}_i = \frac{r_i - \text{mean}(\{r_1,\ldots,r_G\})}{\text{std}(\{r_1,\ldots,r_G\})+\epsilon}$, where $\epsilon > 0$ is a small constant for numerical stability. These normalized advantages capture the relative quality of outputs within the group. The model policy π_θ is then updated by maximizing the following clipped surrogate objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x,\{o_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(\frac{\pi_\theta(o_i|x)}{\pi_{\text{old}}(o_i|x)}\hat{A}_i, \text{clip}\left(\frac{\pi_\theta(o_i|x)}{\pi_{\text{old}}(o_i|x)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_i\right) - \beta\,\mathbb{D}_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\right], \tag{1}$$

where $\pi_\theta(o_i|x)$ is the probability assigned by the current model to output o_i, $\pi_{\text{old}}(o_i|x)$ is the same under the model from previous step, and $\pi_{\text{ref}}(o_i|x)$ is that under the reference model. The first term inside the summation is a clipped policy ratio scaled by $\hat{A}_i$, similar to PPO (Schulman et al., 2017), which prevents overly large updates. The outer expectation is taken over prompts x and their sampled output groups $\{o_i\}$. The second term is a KL divergence penalty that regularizes the updated policy π_θ to stay close to π_{ref}, weighted by a hyperparameter β. This formulation eliminates the need for an explicit value baseline and stabilizes training by comparing outputs within local groups.

Curriculum Learning for RL. Curriculum learning (Bengio et al., 2009; Graves et al., 2017) organizes training by progressing from easy to hard examples. In RL, curricula often follow task complexity (Justesen et al., 2018; Wang et al., 2019; Li et al., 2020), or are learned via teacher-student frameworks modeled as partially observable Markov decision process (Matiisen et al., 2019; Portelas et al., 2020). With the adoption of RL in LLM post-training, curriculum learning has shown potential for improving both training efficiency and model effectiveness. For example, Curri-DPO (Pattnaik et al., 2024) constructs instance-level curricula by ranking preference pairs based on the score gap between preferred and dispreferred responses, introducing harder pairs gradually during DPO fine-tuning. Kimi k1.5 (Team et al., 2025) and Logic-RL (Xie et al., 2025), on the other hand, use manually defined heuristic curricula with fixed training stages, e.g., models are first trained on "easy" samples for a pre-specified number of steps, then switched to "hard" samples. These strategies rely on static schedules and heuristic difficulty labels, without adapting to the model's learning progress. While these works demonstrate the benefit of curriculum learning in LLM post-training, most existing approaches focus on instance-level difficulty or use static, manually designed strategies. In contrast, automatic curriculum learning at the distribution level, especially in RL-based post-training, remains underexplored. In this paper, we propose DUMP to fill this gap by adaptively scheduling training over distributions using advantage-based learnability signals.

3 Method

In this section, we introduce DUMP, a distribution-level curriculum learning framework for RL-based LLM post-training. We first introduce expected absolute advantage as a proxy for learnability, and formalize the scheduling problem as a multi-armed bandit. We then describe a UCB-based strategy to guide distribution selection, followed by the full implementation of DUMP.

3.1 Measuring Learnability via Absolute Advantage

We aim to dynamically assess the usefulness of different data distributions during LLM reinforcement learning post-training. Intuitively, a distribution is more useful (or "learnable") if the model can gain more from training on its samples. To help understand and measure the learnability of the data samples from different distributions, we provide the following proposition:

Proposition 1 (Expected Advantage Magnitude Reflects Learnability)**.** Given a policy π_θ and a data distribution d, the expected absolute advantage $\mathbb{E}_{x\sim d}\left[\mathbb{E}_{o_i\sim\pi_\theta(\cdot|x)}\left[|\hat{A}_i|\right]\right]$ serves as a proxy for how much that distribution d can help the model improve, where the distribution d consisting of prompts $x \sim d$, each prompt has a group of sampled outputs $\{o_1, \ldots, o_n\}$, and $\hat{A}_i$ denotes the advantage of output o_i.

More detailed analysis can be found in Appendix A. Intuitively, if training on a distribution results in a larger expected advantage magnitude, then that distribution is considered more learnable. The

advantage function measures the deviation between an action's predicted value and its actual return; a large advantage—either positive or negative—indicates that the model's current policy is still far from optimal on those samples but has a large potential to improve. A small advantage magnitude does not necessarily imply mastery—it may also occur when a task is too difficult or noisy for the model to learn from effectively, resulting in weak or unstable learning signals. To capture this deviation in both directions, we take the absolute value of the advantage. Without this, positive and negative advantages within a batch may cancel out, masking the true extent of the model's uncertainty or suboptimality. By averaging the absolute advantage over multiple sampled outputs and prompts, we obtain a robust estimate of how much learning signal remains in a given distribution. This expected absolute advantage thus acts as a practical proxy for distribution-level learnability: it reflects how much the model can benefit from training on that distribution. It also has the strength of being lightweight to compute in RL pipelines, as advantage estimates are already generated during rollout.

3.2 Formalizing Distribution-Level Curriculum Learning as Multi-armed Bandit

We aim to design a curriculum learning strategy that dynamically allocates training focus across multiple data distributions to maximize overall model improvement. Let $\mathcal{D} = \{d_1, \ldots, d_N\}$ be a set of data distributions. At each training step, we sample a batch of examples $\mathcal{B}_t$ by drawing prompts from these distributions according to a learnable sampling policy, and use the batch to update model parameters θ via reinforcement learning. The goal is to assign higher sampling probabilities to distributions that offer greater learning potential, thereby maximizing cumulative capability gain.

As motivated in Section 3.1, we quantify the learning potential of a distribution d via its expected absolute advantage, defined as $L(d) = \mathbb{E}_{x\sim d}\left[\mathbb{E}_{o\sim\pi_\theta(\cdot|x)}\left[|\hat{A}(o)|\right]\right]$. Our objective is to dynamically adjust the sampling distribution over $\mathcal{D}$ such that, over the training horizon T, we approximately maximize the total expected learnability gain $\sum_{t=1}^{T}\mathbb{E}_{d\sim P_t}[L(d)]$, where P_t is the sampling distribution at step t. This setup resembles a multi-armed bandit (MAB) problem, where each distribution acts as an arm and its reward corresponds to its learnability. In this setting, the central challenge is to estimate and balance each distribution's potential: exploiting those with high observed advantage while still exploring under-sampled ones that may offer long-term benefit. To this end, we adopt the classic Upper Confidence Bound (UCB) principle (Auer et al., 2002), which provides theoretical guarantees for balancing exploration and exploitation in bandit problems. Specifically, UCB-based algorithms achieve sublinear regret compared to the optimal fixed-arm strategy, and we show in Appendix B that applying UCB on empirical advantage statistics yields a near-optimal schedule under mild assumptions. To allow smoother allocation of sampling probabilities without hard cutoffs and reducing variance in learning, we adopt a soft-selection mechanism: instead of choosing one distribution at each step, we compute a UCB score for every distribution and normalize the scores with a softmax function to obtain a sampling distribution. This soft-selection formulation preserves the spirit of UCB—higher scoring distributions are sampled more—but enables partial exploration of all arms, and it is easier to integrate into LLM training pipelines. The resulting sampling distribution provides a convex mixture over data sources, where each distribution d_j is selected with probability. Each training batch is then composed by drawing examples from multiple distributions in proportion to their scores. To estimate learnability in practice, we maintain a sliding window $\mathcal{A}^w_{d_j}$ of recent absolute advantages for each distribution d_j, and define its empirical reward as the mean absolute advantage: $\hat{L}(d_j) = \frac{1}{|\mathcal{A}^w_{d_j}|}\sum_{a\in\mathcal{A}^w_{d_j}}|a|$. We also track the total number of samples drawn from each distribution n_{d_j}, and the global sample count $n_{\text{total}} = \sum_j n_{d_j}$. The UCB score for each distribution is:

$$\text{UCB}(d_j) = \hat{L}(d_j) + \sqrt{\frac{2\log(n_{\text{total}}+1)}{n_{d_j}+1}} \tag{2}$$

The first term encourages exploitation of distributions with high observed advantages, while the second term ensures sufficient exploration of rarely sampled distributions. To obtain the final sampling weights, we apply a softmax over the UCB scores. Specifically, the probability of selecting distribution d_j is computed as: $P(d_j) = \frac{\exp(\text{UCB}(d_j)/\tau)}{\sum_{j=1}^{N}\exp(\text{UCB}(d_j)/\tau)}$, where $\tau > 0$ is a temperature hyperparameter that controls the sharpness of the sampling distribution. A lower τ results in more peaked selection around the top-scoring distributions, while a higher τ leads to a smoother, more exploratory curriculum. This bandit-based formulation provides a lightweight, adaptive, and reward-sensitive curriculum learning

Algorithm 1 Automated Distribution-Level Curriculum Learning with UCB Sampling

Input: Dataset $\mathcal{D} = \{d_1, \ldots, d_N\}$; pre-trained model parameters θ
Output: Post-trained model parameters θ

1: **function** DUMP($\mathcal{D}$, θ)
2: ▷ Initialize distribution-level statistics
3: **for** each $d_j \in \mathcal{D}$ **do**
4: $\mathcal{A}^w_{d_j} \leftarrow [\,]$ ▷ Sliding window for absolute advantages
5: $n_{d_j} \leftarrow 0$ ▷ Total samples seen from d_j
6: $P(d_j) \leftarrow \frac{1}{N}$ ▷ Equal initial weights
7: **for** training step $t = 1, 2, \ldots, T$ **do**
8: Sample batch $\mathcal{B}_t$ from $\mathcal{D}$ according to $P(d_j)$
9: Compute advantages $\hat{A}(o)$ for all $o \in \mathcal{B}_t$ via model rollout
10: **for** each d_j with samples in $\mathcal{B}_t$ **do**
11: $n_{d_j} \leftarrow n_{d_j} + |\mathcal{B}_{t,d_j}|$ ▷ Update sample count; $\mathcal{B}_{t,d_j}$: subset of batch from d_j
12: $\mathcal{A}^w_{d_j} \leftarrow \mathcal{A}^w_{d_j} \cup \left\{ |\hat{A}(o)| \mid x \in \mathcal{B}_{t,d_j},\, o \sim \pi_\theta(\cdot|x) \right\}$ ▷ Append new advantages from d_j
13: $\mathcal{A}^w_{d_j} \leftarrow \mathcal{A}^w_{d_j}[-k:]$ ▷ k: Window Size; Keep last k elements
14: ▷ Compute UCB scores for each distribution
15: $n_{\text{total}} \leftarrow \sum_{d_j \in \mathcal{D}} n_{d_j}$
16: **for** each $d_j \in \mathcal{D}$ **do**
17: $\hat{L}(d_j) \leftarrow \frac{1}{|\mathcal{A}^w_{d_j}|} \sum_{a \in \mathcal{A}^w_{d_j}} a$ ▷ Mean of absolute advantages
18: $\text{UCB}(d_j) \leftarrow \hat{L}(d_j) + \sqrt{\frac{2\log(n_{\text{total}}+1)}{n_{d_j}+1}}$ ▷ Eq. 2
19: ▷ Update sampling distribution
20: $P(d_j) \leftarrow \frac{\exp(\text{UCB}(d_j)/\tau)}{\sum_{j=1}^{N} \exp(\text{UCB}(d_j)/\tau)} \quad \forall d_j \in \mathcal{D}$ ▷ τ: temperature
21: Update θ using $\mathcal{B}_t$ with an RL algorithm (e.g., GRPO)
22: **return** θ

mechanism. It balances the need to focus on learnable distributions while avoiding premature neglect of underexplored ones. In the next section, we present the complete algorithmic implementation of DUMP, including its integration with rollout procedures and online statistics tracking.

3.3 Algorithm

The detailed curriculum learning procedure is illustrated in Algorithm 1. The algorithm takes as input a dataset $\mathcal{D} = \{d_1, \ldots, d_N\}$ composed of multiple distributions and returns the optimized model parameters θ through a reinforcement learning loop. In lines 3–6, we initialize per-distribution statistics: each distribution $d_j \in \mathcal{D}$ is associated with an empty sliding window $\mathcal{A}^w_{d_j}$ to store recent absolute advantages, a counter n_{d_j} for tracking the number of samples drawn from d_j, and an initial sampling probability $P(d_j) = \frac{1}{N}$ indicating uniform sampling. At each training step t (line 8), a batch $\mathcal{B}_t$ is sampled according to the current distribution weights $P(d_j)$. Advantages $\hat{A}(o)$ are then computed via model rollouts for each sampled output $o \in \mathcal{B}_t$ (line 9). For every distribution d_j that contributes samples in the current batch, we update its sample count n_{d_j} (line 11), append the corresponding advantages to its sliding window $\mathcal{A}^w_{d_j}$ (line 12), and truncate the window to retain only the most recent k entries (300 by default) in line 13. This ensures that our estimate of per-distribution learnability remains up-to-date and robust to noise. In lines 15–18, we compute the Upper Confidence Bound (UCB) score $\text{UCB}(d_j)$ for each distribution. The score consists of two terms: the empirical mean absolute advantage $\hat{L}(d_j)$ over the sliding window $\mathcal{A}^w_{d_j}$, and an exploration bonus inversely proportional to the square root of the number of samples n_{d_j}. This balances prioritization of distributions that are either highly learnable or underexplored. In line 20, the sampling probabilities $P(d_j)$ are updated by applying a softmax over the UCB scores with a temperature parameter τ (0.1 by default). Lower values of τ result in sharper distributions that concentrate more heavily on top-ranked distributions, while higher τ values induce a smoother, more exploratory curriculum. Finally, in line 21, the model parameters θ are updated using the current batch $\mathcal{B}_t$ with a reinforcement learning

algorithm such as GRPO. After T steps, the algorithm returns the post-trained model θ, which has been adaptively guided to learn from the most informative distributions.

4 Experiments and Results

In this section, we first introduce our experiments setup including used models datasets and more implementation details. We then demonstrate the results for the effectiveness of our method DUMP.

4.1 Experiments Setup

RL Algorithm and LLM Models. We use GRPO (Shao et al., 2024) as the underlying RL algorithm in our experiments, which is commonly used in capability-oriented LLM post-training (Guo et al., 2025). We use Qwen2.5-7B-Instruct-1M (Yang et al., 2024) and Qwen2.5-3B-Instruct (Yang et al., 2024) in our experiments.

Datasets and Settings. Multiple datasets are used in our experiments, including Knights and Knaves (K&K) puzzle dataset (Xie et al., 2024), RuleTaker (Clark et al., 2020), ProofWriter (Tafjord et al., 2020), AR-LSAT (Zhong et al., 2021), LogiQA (Liu et al., 2020), LogicNLI (Tian et al., 2021), LongICLBench (Li et al., 2024), GSM-8K (Cobbe et al., 2021), and AIME 1983-2024 (diz, 2025). In our experiments, we consider three different settings. The prompt template used in shown in Figure 3 in the Appendix.

Setting 1: Post-training on K&K puzzles with varying character numbers. The Knights and Knaves (K&K) dataset (Xie et al., 2024) contains procedurally generated logic puzzles where each character is either a knight (always truthful) or a knave (always lying), and the goal is to infer each character's identity. The dataset supports fine-grained difficulty control by adjusting the number of characters. We generate puzzles with 3 to 14 characters, treating each character count as a separate distribution—yielding *12 distinct distributions*. Each distribution includes 900 training and 100 test samples. We post-train Qwen2.5-7B-Instruct-1M on the combined dataset across all distributions.

Setting 2: Post-training on diverse logic reasoning distributions. We perform post-training using a mixture of logic reasoning datasets, including RuleTaker (Clark et al., 2020), ProofWriter (Tafjord et al., 2020), AR-LSAT (Zhong et al., 2021), LogiQA (Liu et al., 2020), LogicNLI (Tian et al., 2021), LongICLBench Geomotion (Li et al., 2024), and Knights and Knaves (K&K) (Xie et al., 2024). For RuleTaker, ProofWriter, and K&K, we further partition the data distributions by complexity levels: RuleTaker by 2, 3, and 5 required reasoning steps; ProofWriter by 3, 4, and 5 required reasoning steps; and K&K by the number of characters (3–7). In total, we construct 15 logic distributions, each containing 400 training samples. We use Qwen2.5-7B-Instruct-1M for this setting.

Setting 3: Post-training on diverse math reasoning distributions. We also explore post-training on diverse math data. For AIME, we split the data into four distributions based on competition years—1983–1993, 1994–2004, 2005–2015, and 2016–2024—since problem styles evolve significantly over time. We also include GSM-8K as a complementary math dataset. This results in five math distributions in total, with 7473 (GSM-8K), 124, 194, 283, and 238 training samples, respectively. We use Qwen2.5-3B-Instruct for this setting.

Reward Implementation. We adopt the rule-based reward mechanism Shao et al. (2024) to provide stable and hack-resistant training signals during RL-based post-training and follow the detailed reward implemetation in Logic-RL (Xie et al., 2025). Specifically, each model response is expected to follow a structured format with the reasoning process enclosed in `<think>` tags and the final answer enclosed in `<answer>` tags. The reward system consists of two components:

Format Reward. A binary reward based on whether the output strictly adheres to the expected format. If the model includes exactly one well-formed `<think>` and one `<answer>` section in the correct order, it receives a reward of $+1$; otherwise, it receives a penalty of -1.

Answer Reward. We evaluate the correctness of the final answer. If the predicted identities fully match the ground truth, the model receives a reward of $+2$; if the answer is incorrect, -1.5; and if the answer is missing or unparsable, -2.

Other Implementation Details. All experiments are conducted on servers equipped with 8 Nvidia A100 GPUs. Our method is implemented with VeRL (Sheng et al., 2024) LLM Reinforcement Learning framework. We use GRPO (Shao et al., 2024) as the training algorithm and follow standard

Data Distribution	without DUMP	with DUMP
RuleTaker 2 Steps	0.79	**0.79**
RuleTaker 3 Steps	0.76	**1.02**
RuleTaker 5 Steps	0.56	**0.98**
ProofWriter 3 Steps	**1.18**	1.09
ProofWriter 4 Steps	0.97	**1.09**
ProofWriter 5 Steps	**1.24**	1.05
AR-LSAT	-0.70	**-0.52**
LogiQA	**1.94**	1.70
LogicNLI	-0.29	**-0.23**
LongICLBench Geomotion	**0.54**	0.25
K & K 3 Characters	2.00	**2.00**
K & K 4 Characters	1.54	**1.76**
K & K 5 Characters	1.53	**1.84**
K & K 6 Characters	0.83	**1.42**
K & K 7 Characters	0.56	**1.02**
Average	0.90	**1.17**

Table 1: Test Answer Reward (see Section 4.1) on diverse logic reasoning distributions (Setting 2). The model used here is Qwen2.5-7B-Instruct-1M.

Data Distribution	without DUMP	with DUMP
GSM-8K	**1.50**	1.47
AIME 1983-1993	-0.76	**-0.39**
AIME 1994-2004	-1.50	**-1.02**
AIME 2005-2015	-0.94	**-0.94**
AIME 2016-2024	-1.27	**-1.27**
Average	-0.59	**-0.43**

Table 2: Test Answer Reward (see Section 4.1) on diverse math reasoning distributions (Setting 3). The model used here is Qwen2.5-3B-Instruct.

practice for actor rollout and optimization. The actor learning rate is set to $1e{-}6$, training batch size is set to 128, and the PPO mini-batch size is 32. KL divergence regularization is applied to encourage alignment with the reference policy, with a KL loss coefficient of 0.001. Each rollout batch contains 16 responses. If not specified, we allow for a maximum response length of 20480 and 4096 tokens during training for Qwen2.5-7B-Instruct-1M and Qwen2.5-3B-Instruct, respectively. The window size k and the temperature τ in our curriculum learning framework is set to 300 and 0.1, respectively.

4.2 Effectiveness of DUMP

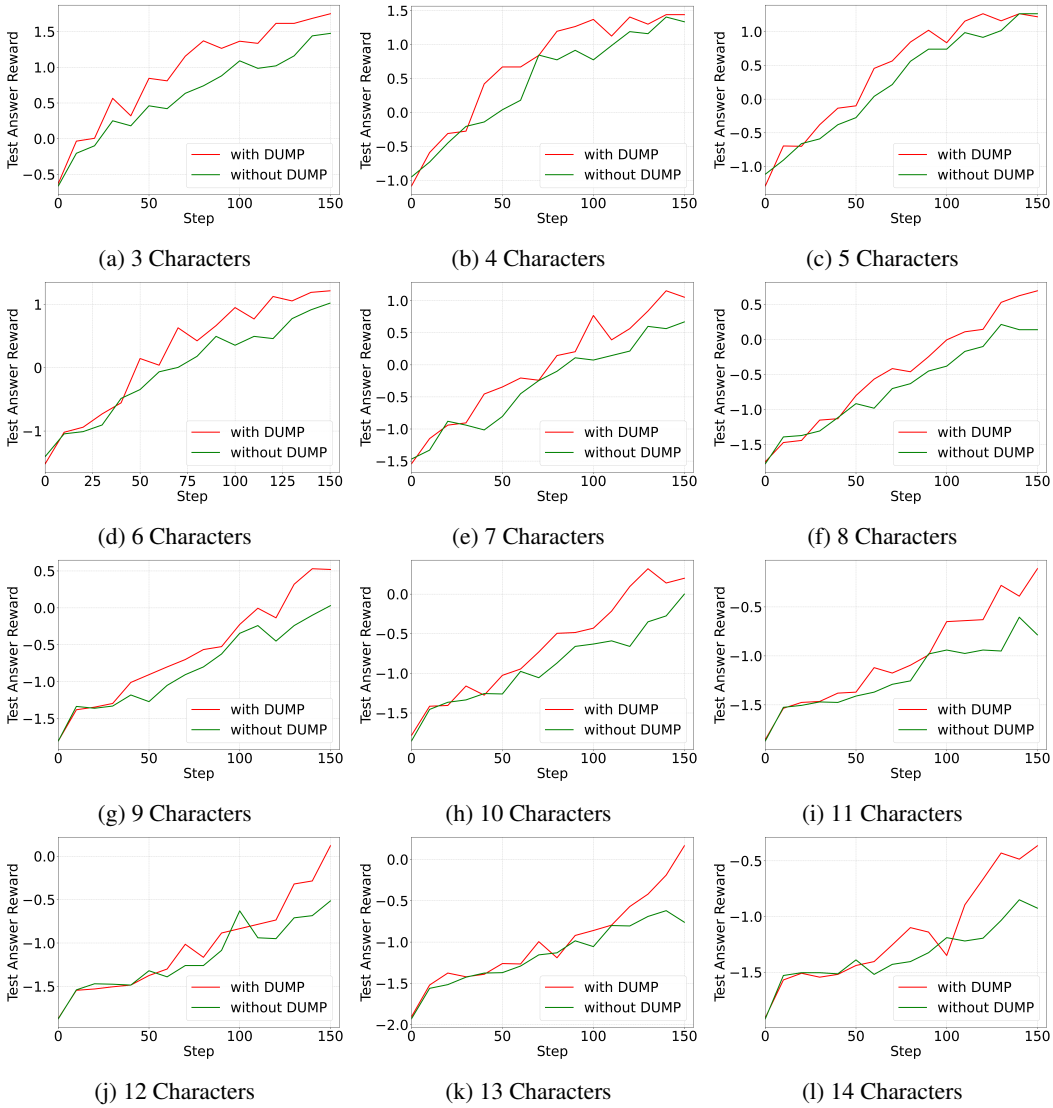
Setting 1: Post-training on the combination of K&K puzzle datasets with different number of characters. To evaluate the effectiveness of DUMP in improving post-training efficiency and performance, we compare it against a uniform distribution sampling baseline across 12 distinct data distributions in the K&K puzzle dataset. Each distribution corresponds to a fixed number of characters in the puzzle, ranging from 3 to 14. Figure 1 plots the test answer reward over training steps for each distribution, with and without DUMP. Across all distributions, DUMP consistently outperforms the baseline, achieving faster convergence and higher test performance. The gains are particularly notable in mid- to high-difficulty distributions (e.g., 6 to 12 characters), where uniform sampling tends to struggle due to data underutilization. For example, in the 9-character distribution (Figure 1g), the model trained with DUMP achieves a reward of over 0.5, whereas the baseline remains below 0.0. These results validate the core intuition of DUMP: dynamically adjusting the sampling focus toward high-learnability distributions accelerates policy improvement while avoiding wasted effort on over-saturated or low-signal data. Notably, the improvement is achieved without any curriculum heuristics or manual data ordering—only by observing advantage signals and adapting online.

Setting 2: Post-training on diverse logic reasoning distributions. We apply DUMP to 15 logic reasoning distributions including subsets of RuleTaker, ProofWriter, and K&K (with varying difficulty levels), as well as datasets such as AR-LSAT, LogiQA, LogicNLI, and LongICLBench. As shown in Table 1, DUMP improves the average test answer reward from 0.90 to 1.17. Notable improvements are observed on complex tasks such as AR-LSAT, where the reward increases from -0.70 to -0.52, and K&K 7 Characters, from 0.56 to 1.02. These results demonstrate that DUMP adaptively prioritizes undertrained but learnable distributions, leading to more efficient capability gains.

Setting 3: Post-training on diverse math data distributions. We further evaluate DUMP on GSM-8K and different subsets of AIME grouped by competition years. As shown in Table 2, DUMP raises the average test answer reward from -0.59 to -0.43, with the most significant gain on AIME 1994–2004, where performance improves from -1.50 to -1.02. These results highlight DUMP's robustness under distribution shifts and data imbalance.

4.3 Ablation Study on the Sampling Strategy

In this section, we ablate the sampling strategy used in DUMP's UCB-based scheduler. As described in Algorithm 1, our method applies soft sampling controlled by a temperature parameter. The greedy variant (temperature = 0) always selects the distribution with the highest UCB score, while our default



(a) 3 Characters (b) 4 Characters (c) 5 Characters

(d) 6 Characters (e) 7 Characters (f) 8 Characters

(g) 9 Characters (h) 10 Characters (i) 11 Characters

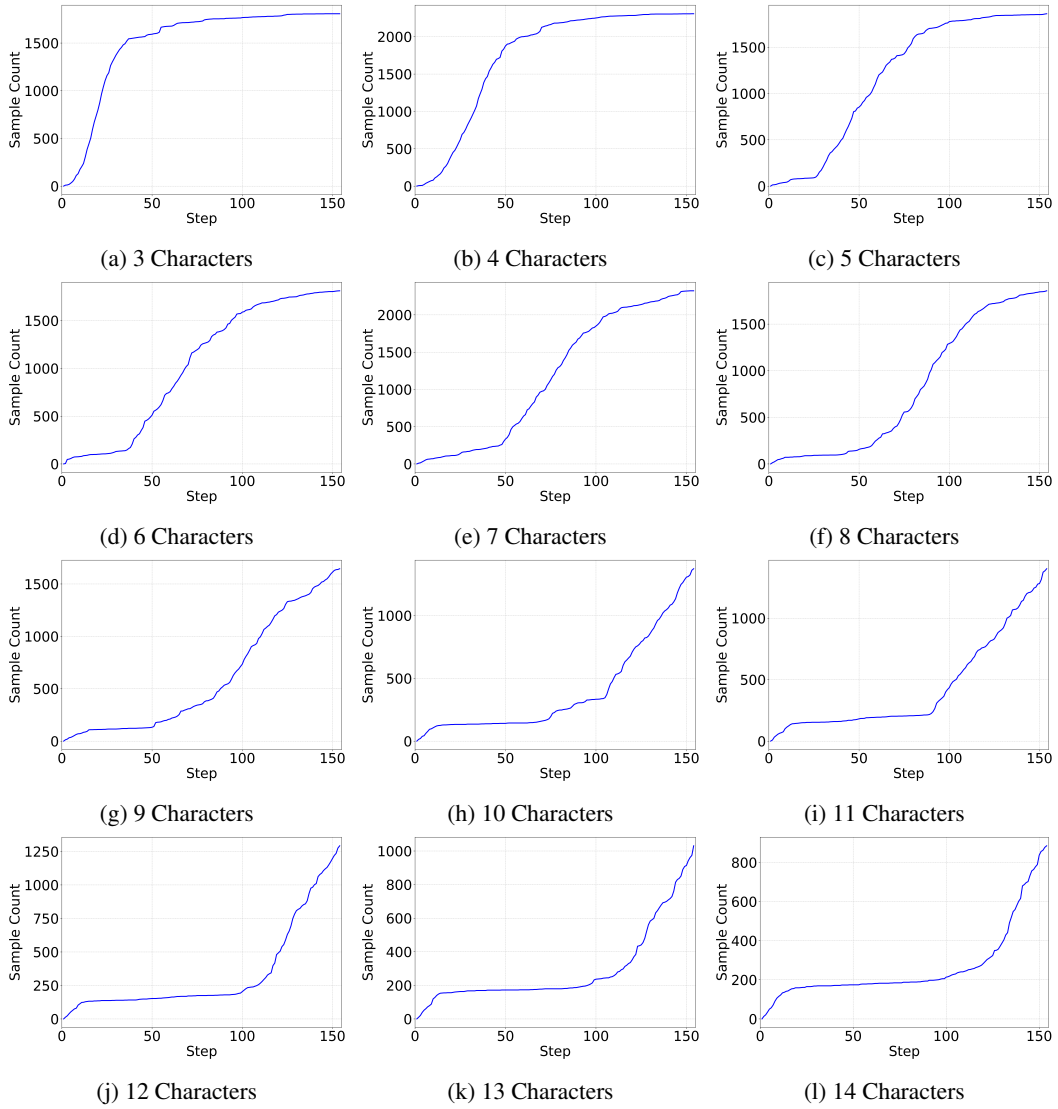
(j) 12 Characters (k) 13 Characters (l) 14 Characters

Figure 1: Effectiveness of DUMP on the K&K puzzle dataset mixed with 12 distributions defined by the number of characters in each puzzle (Setting 1). DUMP consistently achieves higher answer reward on test dataset compared to baseline. The model used here is Qwen2.5-7B-Instruct-1M.

uses a small temperature (0.1) to enable probabilistic sampling. We conduct experiments under Setting 1, with a maximum training response length of 10240 tokens. After 100 training steps, the greedy strategy significantly underperforms due to its lack of exploration—it tends to lock onto a single distribution early and fails to adapt. For instance, on the 13- and 14-character K&K tasks, the greedy variant achieves test answer rewards of -0.91 and -1.38, while soft sampling reaches -0.66 and -1.16, respectively. These results highlight the importance of maintaining exploration via a non-zero temperature to prevent the scheduler from collapsing onto suboptimal distributions.

4.4 Analyzing the Automated Curriculum by DUMP

To understand how DUMP dynamically allocates training effort across data distributions, we analyze the sampling patterns induced by its UCB-based curriculum mechanism. Figure 2 shows the cumulative number of samples drawn from each distribution (3 to 14 characters) over the course of training on K&K puzzles with varying character numbers (Setting 1). We observe a clear curriculum-like progression: distributions corresponding to simpler puzzles (e.g., 3–5 characters) are heavily sampled in the early stages of training, while more complex distributions (e.g., 10–14 characters) are gradually introduced and increasingly prioritized as training progresses. This pattern aligns with the model's



(a) 3 Characters (b) 4 Characters (c) 5 Characters

(d) 6 Characters (e) 7 Characters (f) 8 Characters

(g) 9 Characters (h) 10 Characters (i) 11 Characters

(j) 12 Characters (k) 13 Characters (l) 14 Characters

Figure 2: Curriculum (sample counts) induced by DUMP across 12 K&K puzzle distributions with increasing difficulty defined by the number of characters in each puzzle (Setting 1). Simpler distributions are automatically prioritized in early training, while more complex ones are progressively emphasized, demonstrating automated distribution scheduling.

evolving capacity—early training favors distributions with high initial advantage magnitudes, and as the model saturates on those, DUMP shifts focus to underexplored but learnable distributions. Importantly, this adaptive sampling behavior emerges automatically from empirical advantage signals without requiring manual specification of curriculum order. These results highlight DUMP's ability to construct an implicit, data-driven curriculum that mirrors traditional easy-to-hard strategies, while remaining responsive to online training dynamics.

5 Conclusion

In this work, we introduce a distribution-level curriculum learning framework for RL-based post-training of large language models. DUMP leverages the expected absolute advantage as a learnability signal to adaptively allocate training focus across heterogeneous distributions. By formalizing scheduling as a multi-armed bandit and adopting a UCB-based sampling strategy, DUMP balances exploitation and exploration in a principled way. Experiments demonstrate that DUMP consistently improves convergence and final performance over baselines. These results highlight the value of distribution-aware curriculum learning in LLM RL post-training.

REFERENCES

Aime_1983_2024 (revision 6283828), 2025. URL https://huggingface.co/datasets/di-zhang-fdu/AIME_1983_2024.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

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

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867*, 2020.

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

Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, et al. Competitive programming with large reasoning models. *arXiv preprint arXiv:2502.06807*, 2025.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 2022.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pp. 1311–1320. Pmlr, 2017.

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

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Illuminating generalization in deep reinforcement learning through procedural level generation. *arXiv preprint arXiv:1806.10729*, 2018.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv e-prints*, pp. arXiv–2309, 2023.

Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning. In *2020 ieee international conference on robotics and automation (icra)*, pp. 4051–4058. IEEE, 2020.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pp. 22631–22648. PMLR, 2023.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

Mikhail Mirzayanov. Codeforces. https://codeforces.com/. Accessed: 2025-04-13.

OpenAI. Learning to reason with llms. Technical report, OpenAI, 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

OpenAI. Openai o3-mini. Technical report, OpenAI, 2025. URL https://openai.com/index/openai-o3-mini/.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Pulkit Pattnaik, Rishabh Maheshwary, Kelechi Ogueji, Vikas Yadav, and Sathwik Tejaswi Madhusudhan. Enhancing alignment using curriculum learning & ranked preferences. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12891–12907, 2024.

Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pp. 835–853. PMLR, 2020.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

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

ByteDance Seed. Seed-thinking-v1.5: Advancing superb reasoning models with reinforcement learning. Technical report, ByteDance, 2025. URL https://github.com/ByteDance-Seed/Seed-Thinking-v1.5.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *arXiv preprint arXiv:2012.13048*, 2020.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. Diagnosing the first-order logical reasoning ability through logicnli. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3738–3747, 2021.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. *arXiv preprint arXiv:2410.23123*, 2024.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. Ar-lsat: Investigating analytical reasoning of text. *arXiv preprint arXiv:2104.06598*, 2021.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Detailed Analysis for Proposition 1

The notion of learnability concerns how much further improvement can be gained by training on a distribution d. In policy-gradient methods, the signal for improvement is encoded in the advantage $\hat{A}_i$ of each sampled output. For each output o_i, the surrogate update term is proportional to $\hat{A}_i$. If $\hat{A}_i > 0$, increasing $\pi_\theta(o_i \mid x)$ raises expected reward; if $\hat{A}_i < 0$, decreasing $\pi_\theta(o_i \mid x)$ raises expected reward. Hence the sign of $\hat{A}_i$ determines the direction of improvement, and the magnitude $|\hat{A}_i|$ determines the strength of the adjustment before clipping. A large $|\hat{A}_i|$ therefore indicates high potential for local improvement. Aggregating over outputs and prompts from d, we obtain $L(d) = \mathbb{E}_{x\sim d}\left[\mathbb{E}_{o_i\sim\pi_\theta(\cdot|x)}\left[|\hat{A}_i|\right]\right]$ This expectation quantifies the average actionable update mass contributed by d: if $L(d)$ is large, then samples from d frequently carry strong signals that can still move the policy; if $L(d)$ is small, then training on d tends to induce negligible changes, suggesting that the distribution is either saturated or uninformative. Thus $L(d)$ reflects the *remaining headroom* of d for improving the policy. We regard a distribution as "learnable" if additional training on it can reliably shift the policy toward higher reward. $L(d)$ is aligned with this definition because: (i) it is *reward-aware*—advantage derives directly from the reward signal relative to a baseline; (ii) it is *direction-aware*—the sign of $\hat{A}_i$ encodes whether probability mass should be increased or decreased; (iii) it is *magnitude-sensitive*—larger $|\hat{A}_i|$ means stronger room for adjustment. Therefore, higher $L(d)$ directly corresponds to a higher capacity of d to drive beneficial parameter updates. Unlike raw loss or uncertainty measures, $L(d)$ is computed for free during standard RLHF/GRPO training, and is invariant to affine transformations of rewards due to group normalization. This makes it both theoretically meaningful and practically convenient as a proxy for distribution-level learnability.

B Theoretical Justification for UCB-Based Distribution Scheduling

We provide a theoretical justification for using Upper Confidence Bound (UCB) as a strategy for scheduling training over data distributions in RL-based post-training. Our objective is to maximize the cumulative learnability gain over T training steps, defined as:

$$\max_{\{d_t\}_{t=1}^T} \sum_{t=1}^{T} L(d_t), \quad \text{where} \quad L(d) = \mathbb{E}_{x\sim d}\left[\mathbb{E}_{o\sim\pi_\theta(\cdot|x)}\left[|\hat{A}(o)|\right]\right].$$

This setting can be viewed as a stochastic multi-armed bandit (MAB) problem, where each data distribution $d_j \in \mathcal{D}$ corresponds to an arm with unknown reward $L(d_j)$, interpreted as the expected absolute advantage from training on samples from d_j. At each training step t, the learner selects a distribution d_t and obtains an empirical reward $\hat{L}(d_t)$ by averaging the absolute advantages observed in the batch.

We define the regret as the gap between the cumulative learnability gain of the best fixed distribution $d^* = \arg\max_d L(d)$ and that of the learner's actual selections:

$$\text{Regret}(T) = \sum_{t=1}^{T} L(d^*) - \sum_{t=1}^{T} L(d_t).$$

To analyze this regret, we make the following assumptions:

1. For each distribution d_j, the per-output absolute advantages $|\hat{A}(o)|$, where $o \sim \pi_\theta(\cdot|x)$, are i.i.d. and bounded in $[0, C]$ for some constant $C > 0$.
2. The true expected advantage $L(d_j)$ remains approximately stationary over a local training window, enabling meaningful online adaptation.

Note: In practice, we can clip or normalize $|\hat{A}(o)|$ to satisfy the boundedness condition. The introduction of the constant C only scales the regret by a constant factor and does not affect the asymptotic rate of convergence.

Under these assumptions, the following regret bound holds:

Theorem B.1. *Let $\mathcal{D} = \{d_1, \ldots, d_N\}$ be a set of data distributions with fixed expected rewards $L(d_j) \in [0, C]$. Then, applying the UCB1 algorithm to the empirical reward observations yields the regret bound:*

$$\textit{Regret}(T) \leq O\left(C \cdot \sum_{j:\Delta_j>0} \frac{\log T}{\Delta_j}\right), \quad \textit{where} \quad \Delta_j = L(d^*) - L(d_j).$$

Proof. This result is a direct application of the classical UCB1 regret bound (Auer et al., 2002), extended to the case where reward values lie in $[0, C]$. Let $d^* = \arg\max_d L(d)$ be the optimal distribution, and let $\Delta_j = L(d^*) - L(d_j)$ denote the suboptimality gap for each arm d_j.

At each time step t, UCB1 selects the distribution d_j with the highest upper confidence bound:

$$\text{UCB}(d_j) = \hat{L}(d_j) + \sqrt{\frac{2C^2 \log t}{n_j}},$$

where n_j is the number of times distribution d_j has been sampled so far, and $\hat{L}(d_j)$ is the empirical mean of observed rewards (mean absolute advantages).

Under the assumptions that rewards are i.i.d. and bounded in $[0, C]$, the Hoeffding inequality guarantees that with high probability the empirical mean concentrates around the true mean $L(d_j)$, and the UCB selection mechanism will only pick suboptimal arms a logarithmic number of times. Based on UCB1 regret bound (Auer et al., 2002), The cumulative regret is therefore bounded by:

$$\text{Regret}(T) \leq \sum_{j:\Delta_j>0} \left(\frac{8C^2 \log T}{\Delta_j} + O(\Delta_j)\right),$$

which simplifies to the stated asymptotic bound:

$$\text{Regret}(T) = O\left(C \cdot \sum_{j:\Delta_j>0} \frac{\log T}{\Delta_j}\right).$$

□

This result shows that our distribution-level scheduling strategy, when driven by UCB over empirical advantage rewards, is provably efficient. It dynamically concentrates training on distributions with high estimated learnability while ensuring sufficient exploration, with regret that scales logarithmically in T and linearly in $1/\Delta_j$.

C Comparison to Heuristic Curriculum

Heuristic curricula, which manually specify a fixed training schedule over data distributions—e.g., training on Distribution A for N steps before switching to Distribution B—have been explored in prior work Xie et al. (2025); Team et al. (2025), particularly in environments where task difficulty or domain progression is well understood. However, such approaches have several limitations that make them less suitable for our setting. First, effective heuristic scheduling requires strong prior knowledge about the relative difficulty and learnability of each distribution. In our setting, which involves diverse domains such as logic reasoning, mathematics, and programming, such prior knowledge is often unavailable or misleading. For example, a distribution may appear "easier" but provide low learning signal, or seem "harder" but actually yield high gradient utility. This makes it extremely difficult to construct robust, generalizable heuristics across tasks. Second, heuristic curricula are static and cannot adapt to the evolving needs of the model during training. In contrast, DUMP dynamically adjusts sampling priorities based on actual model performance—measured via policy advantages—allowing it to focus on the most beneficial distributions at each stage of learning. Finally, the lack of standardized or widely accepted heuristic curricula for our task suite makes it hard to conduct fair and meaningful comparisons. Instead, we benchmark DUMP against uniform sampling and adaptive baselines, which are more reflective of current best practices in large-scale post-training pipelines.

Example of Prompt

You are a helpful assistant. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and<answer> </answer> tags, respectively, i.e., <think> reasoning process here </think><answer> answer here </answer>. Now the user asks you to solve a reasoning problem. After thinking, when you finally reach a conclusion, clearly state the identity of each character within <answer> </answer> tags. [Problem]

Figure 3: Example of prompt used.

D LIMITATIONS

First, while the core idea of distribution-level curriculum learning is broadly applicable, we evaluate DUMP only in the context of large language models (LLMs) and do not extend the experiments to multimodal large language models (MLLMs) due to computational constraints. Second, our experiments are limited to 7B-scale models. Scaling our method to larger models remains an important direction for future work.