arXiv:2508.05015v1 [cs.LG] 7 Aug 2025

SPaRFT: Self-Paced Reinforcement Fine-Tuning for Large Language Models

Van Dai Do
Applied Artificial Intelligence Initiative
Deakin University
Australia
v.do@deakin.edu.au

Manh Nguyen
Applied Artificial Intelligence Initiative
Deakin University
Australia
manh.nguyen@deakin.edu.au

Svetha Venkatesh
Applied Artificial Intelligence Initiative
Deakin University
Australia
svetha.venkatesh@deakin.edu.au

Hung Le
Applied Artificial Intelligence Initiative
Deakin University
Australia
thai.le@deakin.edu.au

Abstract

Large language models (LLMs) have shown strong reasoning capabilities when fine-tuned with reinforcement learning (RL). However, such methods require extensive data and compute, making them impractical for smaller models. Current approaches to curriculum learning or data selection are largely heuristic-driven or demand extensive computational resources, limiting their scalability and generalizability. We propose **SPaRFT**, a self-paced learning framework that enables efficient learning based on the capability of the model being trained through optimizing which data to use and when. First, we apply *cluster-based data reduction* to partition training data by semantics and difficulty, extracting a compact yet diverse subset that reduces redundancy. Then, a *multi-armed bandit* treats data clusters as arms, optimized to allocate training samples based on model current performance. Experiments across multiple reasoning benchmarks show that SPaRFT achieves comparable or better accuracy than state-of-the-art baselines while using up to $100\times$ fewer samples. Ablation studies and analyses further highlight the importance of both data clustering and adaptive selection. Our results demonstrate that carefully curated, performance-driven training curricula can unlock strong reasoning abilities in LLMs with minimal resources.

1 Introduction

Large Language Models (LLMs) have achieved remarkable progress in tasks requiring reasoning, problem-solving, and generalization, driven largely by scaling trends in model size, data, and compute [8, 21]. As the cost and complexity of pretraining continue to rise, research attention has increasingly shifted toward post-training techniques, which aim to improve LLM capabilities more efficiently. Among these, Reinforcement Fine-Tuning (RFT) has emerged as a promising method that aligns model behavior with outcome-based reward signals, often relying on lightweight supervision without elaborate reward engineering or inference-time computation [5, 15, 19].

Standard RFT trains on uniformly sampled batches from the full dataset [5]. While being simple, this approach ignores each example's difficulty, informativeness, and uncertainty, wasting limited reward feedback on trivial or noisy instances and slowing convergence [6, 22]. Thus, this raises two key underexplored dimensions: how to select which examples to train on, and how to present them to the LLMs over time.

Data reduction methods prioritize informativeness by estimating example difficulty or uncertainty. For instance, variance-based filtering techniques score and select examples based on multiple forward passes through a reference model [34]. While effective in reducing noise, these approaches incur significant computational overhead, making them impractical for resource-constrained models. Moreover, the method is sensitive to both the selected training examples and the uncertainty estimation model, which can negatively impact its generalization to new LLMs.

In parallel, the design of curriculum—the order of examples being trained—plays a central role in guiding the learning trajectory [2]. As the model improves throughout fine-tuning, its ability to benefit from more difficult examples evolves dynamically. Static curricula or random example orders often fail to reflect this progression. Some recent methods attempt to introduce adaptivity by filtering examples using heuristic thresholds [28], but these threshold-based mechanisms can be fragile. When applied to small or weak models, these methods often rely heavily on predefined difficulty metrics, which can be inaccurate or misaligned with the model's evolving capabilities. As a result, they may prematurely exclude sufficiently challenging examples during early training stages, limiting exposure to informative data and stalling learning progress.

In this paper, we introduce **SPaRFT**, a self-paced RFT framework that automatically selects informative training examples and designs an adaptive training progress to improve the efficiency of LLM training. In particular, we formulate the RFT process as a Multi-Armed Bandit (MAB) problem [30], where each arm corresponds to a cluster with similar semantics and difficulty. We optimize the MAB to determine which data to present to LLMs at each training step, overcoming the rigidity and heuristic-driven data assignment of existing training methods. First, we begin with the data reduction stage to form the clusters (arms). We perform clustering using both semantic representations of data sample and its difficulty scores, calculated from its solve rate using a moderate LLM model. Motivated by recent findings that much training data is redundant [34], we retain only a fixed number of samples per cluster. As such, within each cluster, representative examples are chosen iteratively by maximizing their distance from previously selected samples until the quota is reached. In the second stage, we dynamically optimize the cluster selection based on the model's training performance. Pulling an arm corresponds to sampling training examples from the associated cluster. The inverse of the current model's solve rate on these examples serves as the reward signal, which updates the bandit's selection strategy. Our approach is guided by a simple intuition: *cluster selection should adapt to the model's performance to prioritize currently challenging examples*. This prevents unnecessary retraining on examples the LLM already solves easily, which is particularly important for low-resource settings, involving tiny LLMs with smaller parameter counts, and tight resource budgets [17]. For such settings, curriculum design and data reduction must be efficient, performance-aware, and free from costly heuristics, essential for making RFT practical in the small-model regime.

To evaluate our approach, we conduct extensive experiments on mathematical problem-solving tasks using several tiny LLMs of less than 1 billion parameters. Results show that SPaRFT significantly improves reasoning accuracy and robustness compared to both reinforcement learning and curriculum learning baselines, while reducing the number of training examples by a factor of 100. Notably, SPaRFT outperforms methods that rely on exhaustive search to select single or few training examples. Our analysis reveals how poorly designed curricula can get stuck in easy example regions, failing to leverage the diversity of the dataset.

In summary, our contributions are threefold: (1) We propose a novel two-stage framework that reduces the number of training examples and optimizes the RFT progress for LLMs using MAB. (2) Our method is lightweight, significantly reducing the number of training examples while adding minimal computational overhead, making it well-suited for RFT of tiny LLMs. (3) We show through extensive experiments that SPaRFT consistently outperforms existing curriculum and data reduction strategies.

2 Related Work

2.1 Reinforcement Fine-Tuning for Language Models.

Language models can be formulated as sequential decision-making agents, enabling the application of RL techniques for fine-tuning. Proximal Policy Optimization (PPO) [27] has been widely adopted in early RLHF pipelines. More recent work introduced actor-only alternatives such as REINFORCE++ [12] and Group Relative Policy Optimization (GRPO) [5], which eliminate the need for value networks and have shown strong performance on large language models, particularly in reasoning tasks. GRPO has been used in large-scale instruction tuning setups for alignment without reliance on learned reward models.

2.2 Data Reduction for Reinforcement Fine-tuning of Language Models.

The role of data in RFT for LLMs remains an open area of research. Several works attempt to curate high-quality mathematical datasets [20, 36], but they do not explicitly explore which data is most effective for fine-tuning. More recently, alternative approaches have proposed using heuristic-based scores, such as Learning Impact Measurement [18] or variance-based selection methods [34]. These techniques alleviate data constraints by enabling training on only a small subset—sometimes as few as one or two examples—while still achieving strong reasoning capabilities. However, these methods typically require significant pre-computation, limiting their practicality in real-world deployment, and remain untested on small models with limited reasoning capabilities.

2.3 Curriculum Learning for LLMs.

Humans and animals learn more effectively when examples are presented in a meaningful order that gradually increases in complexity. Curriculum learning [2] and performance-guided training progression [16] have been applied to supervised and RL training. For LLMs, recent studies have explored how to organize training data to reduce computational cost and improve sample efficiency, though this area remains underdeveloped. Existing approaches include hand-crafted difficulty tiers [35, 20, 29], which often require task-specific insights and manual tuning. More adaptive methods, such as AdaRFT [28], learn a training curriculum by dynamically adjusting a difficulty threshold to select examples. While promising, these methods still face limitations: repeated training on easy examples can lead to overfitting or poor generalization; difficulty heuristics may not transfer across tasks; and fixed sampling strategies may fail to adapt to evolving model capabilities. There remains a need for more principled and generalizable curriculum strategies tailored to the scale and dynamics of LLM training.

3 Self-paced Reinforcement Fine-Tuning

We aim to improve the performance of a policy model π_θ through adaptive training progress while minimizing the amount of training data required. Training directly on tasks that are either too easy or too difficult often leads to inefficient learning, as the model receives limited challenge or feedback. Instead, the data assignment should adapt to the model's evolving capabilities—presenting examples that it is ready to learn from at each stage. Threshold methods for curriculum control require manual tuning and can be suboptimal, especially when the model struggles early in training [28]. To overcome these limitations, we introduce **SPaRFT**, a two-phase approach for self-paced training optimization: (1) cluster-based data reduction and (2) bandit-based data assignment. SPaRFT integrates seamlessly with common RFT algorithms, and we use GRPO [5] as the default method. A detailed description of the algorithm is provided in Appendix A.1's Algorithm 1.

3.1 Cluster-based Data Reduction

3.1.1 Data Clustering

RFT often relies on large quantities of high-quality labeled training data. However, acquiring such data can be expensive, time-consuming, or infeasible in many real-world scenarios, especially low-resource settings. To address this limitation, we propose a clustering-based data reduction method that aims to reduce data requirements while maintaining or even enhancing training efficiency. Our

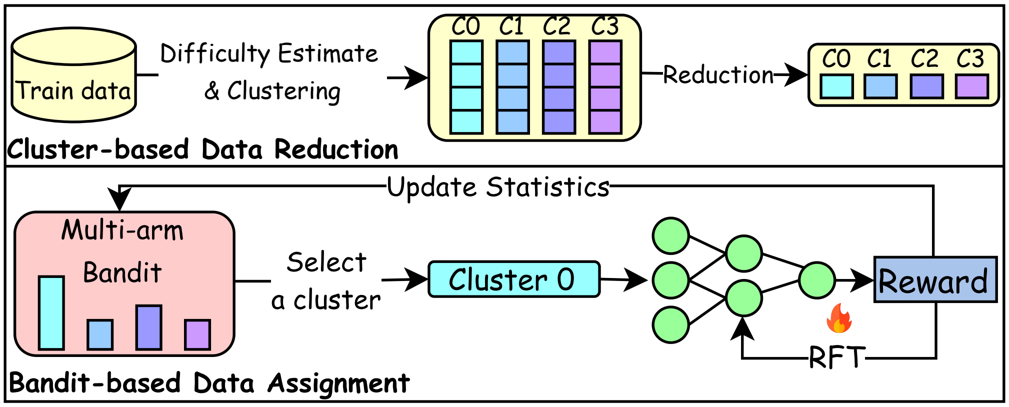

Figure 1: SPaRFT Architecture. Top: Initial training data is annotated with difficulty, and each example's feature vector is formed by concatenating its semantic embedding with its difficulty score. These vectors are clustered into K groups, and from each cluster we select representative samples to ensure both coverage and diversity. Bottom: Each cluster is treated as an arm in a multi-armed bandit. At each step, Thompson Sampling is used to select a cluster, its representative examples are fed to the LLM to obtain rewards for RFT, and those rewards are used to update the bandit statistics.

approach leverages two key signals for each training example: its semantic content and its estimated difficulty. Intuitively, grouping examples with similar semantics and comparable difficulty levels enables a more coherent and effective curriculum design.

The first component is *semantic representation*. We use embeddings from a pretrained Sentence-BERT model [24], which produces high-dimensional vectors that encode both surface-level and contextual semantics. However, directly using these high-dimensional embeddings for clustering may lead to issues related to the curse of dimensionality [1], which can negatively impact the effectiveness of distance-based clustering. To address this, we apply Principal Component Analysis (PCA) to reduce the dimensionality of the semantic representations in $\mathcal{D}$. Let x_i denote i^{th} training example, and let s_i be its PCA-reduced semantic embedding.

The second component is *difficulty estimation*. Each example in $\mathcal{D}$ is assigned a scalar difficulty score d_i. In curriculum learning, it is common practice to begin with an estimate of example difficulty [37, 33]. We adopt an automatic approach to estimate difficulty, following the formulation proposed by [28], which defines the difficulty of a question x_i based on its empirical success rate across multiple attempts:

$$d_i = 100 \times \left(1 - \frac{\#\text{successful attempts on question } x_i}{n}\right), \tag{1}$$

where n is the total number of attempts. Following the common practice, we set $n = 128$ to reduce estimation variance and avoid bias. To compute difficulty scores, we use the **Qwen2.5-Math-7B** model [23]. A strong model (e.g., OpenAI o1 [21], DeepSeek R1[5]) solves most problems with ease, making it hard to distinguish easy from hard examples. A weak model (e.g., LLaMA 3.3-1B [9]) fails too often, resulting in noisy estimates. Qwen2.5-Math-7B offers a middle ground, solving moderately difficult problems while still failing on harder ones, which produces informative and well-calibrated difficulty scores.

To incorporate the two aforementioned components into the final representation for clustering, we first standardize each element of the PCA-reduced semantic embedding and the difficulty score so that they each have zero mean and unit variance. This ensures neither modality dominates the distance metric. We then concatenate the standardized semantic embedding with the standardized difficulty score:

$$e_i = \hat{s}_i \oplus \hat{d}_i, \tag{2}$$

where $\hat{s}_i$ and $\hat{d}_i$ denote the standardized semantic vector and difficulty scalar, respectively, and $\oplus$ denotes vector concatenation. Finally, we apply K-means clustering on these combined embeddings to partition the dataset into K clusters.

3.1.2 Data Reduction

After clustering the training set into K clusters, we select a fixed number of representative examples from each cluster to ensure coverage and diversity. Let l denotes the number of examples for each cluster, C_k denote the set of indices assigned to cluster k^{th}, and let μ_k be the corresponding cluster centroid in the embedding space. For each example $i \in C_k$, we compute its Euclidean distance to the cluster centroid:

$$\delta_i = \|e_i - \mu_k\|_2, \tag{3}$$

We then sort all examples in C_k by ascending δ_i. To construct a diverse and representative training subset from each cluster, we employ a greedy farthest-point sampling strategy: starting from the cluster centroid, we iteratively select examples that are maximally distant from those already chosen. This balances centrality and coverage in the embedding space. The full procedure is outlined in Phase 1 of Algorithm 1 in Appendix A.1.

3.2 Bandit-based Data Assignment

Our core intuition is simple: *at any point during training, we should prioritize examples that are currently challenging for the model.* If we were to fix a static set of "challenging" examples upfront, the model's capabilities would evolve, and some initially difficult examples would become trivial over time, resulting in inefficient training on examples the model already masters.

At each training step t, our policy network π_θ selects a cluster $c_t \in \{1, \ldots, K\}$ from which to sample training data. This setup resembles a multi-armed bandit problem, where π_θ acts as the policy and each arm corresponds to a cluster c_t. The reward for pulling an arm changes based on policy π_θ over time. The objective in our framework is performance-driven: we aim to select examples that are currently hard for π_θ, so that the model can learn valuable insights from such challenging instances.

To implement this performance-driven curriculum, we adopt a multi-arm bandit approach with Thompson sampling [31, 26]. Let μ_k denote the expected reward of cluster k, and let $\hat{\mu}_k^{(t)}$ and $n_k^{(t)}$ represent the empirical mean and the number of observed rewards for cluster k up to step t, respectively. Thompson sampling maintains a posterior belief over μ_k by drawing samples from a Gaussian distribution:

$$\tilde{\mu}_k^{(t)} \sim \mathcal{N}\left(-\frac{R_k^{(t)}}{n_k^{(t)} + \epsilon}, \frac{1}{n_k^{(t)} + \epsilon}\right), \tag{4}$$

where $R_k^{(t)}$ is the cumulative reward for cluster k, given by the sum of its past solve rates, and ϵ is a small constant to prevent division by zero. The *solve rate* is the proportion of examples in a cluster that the current model solves correctly. Since we aim to prioritize more challenging examples, we negate the mean reward. A lower solve rate corresponds to higher difficulty and thus yields higher reward. Note that this solve rate is computed online from the model's performance during training. It differs from the *difficulty* score, which is estimated beforehand as described in Section 3.1.1. Therefore, the MAB is optimized to select the data cluster based on the current performance of the model. At training step t, the cluster is selected greedily based on the sampled posterior:

$$c_t = \arg\max_k \tilde{\mu}_k^{(t)}. \tag{5}$$

This formulation naturally balances exploration and exploitation: clusters with high uncertainty (i.e., fewer observations) are more likely to be explored, while clusters with consistently lower solve rates (i.e., harder samples) are exploited more frequently. A batch of size B is sampled from cluster c_t to train π_θ. The reward for each sample in the batch is computed based on the correctness of the model's output:

$$r_i = \begin{cases} 1, & \text{if the response is correct} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

The average reward over the batch is calculated as:

$$r_{\text{avg}} = \frac{1}{B} \sum_{i=1}^{B} r_i \tag{7}$$

r_{avg} is used in two ways. First, it provides a scalar learning signal to update the policy network π_θ via any RL algorithm (e.g., GRPO). The training objective is to maximize the expected total reward:

$$\max_\theta \mathbb{E}_{q \sim D_{train}, a \sim \pi_\theta} [r_{avg}] \tag{8}$$

where a, sampled from π_θ is the answer for question q.

Second, r_{avg} is used to update the internal statistics of the multi-armed bandit by adjusting both the cumulative reward and the count of interactions for the selected cluster:

$$R_{c_t}^{(t+1)} = R_{c_t}^{(t)} + r_{\text{avg}}, \quad n_{c_t}^{(t+1)} = n_{c_t}^{(t)} + 1. \tag{9}$$

This dynamic update process ensures that the sampling distribution adapts online to the evolving state of the model. Clusters that temporarily present higher error rates receive increased sampling probability, while those that become easy are sampled less frequently. Crucially, because rewards are derived from model predictions themselves, this approach tightly couples curriculum scheduling with real-time model capacity—resulting in a self-regulating training signal that continuously steers focus toward maximally informative examples. Full algorithm of SPaRFT is provided in Algorithm 1, Appendix A.1. In addition, we formally analyze the convergence of the MAB used in SPaRFT, as stated in the following Proposition.

Proposition 1. *Under assumptions: (i) bounded rewards, LLM training with (ii) gradient clipping and (iii) decayed learning rate, the Thompson Sampling scheduler in SPaRFT satisfies sublinear reward variation up to step T: $V_T = O(\log T)$. Consequently, as $t \to \infty$, the sampling distribution concentrates on clusters with maximal expected reward.*

Proof. See Appendix A.2. □

4 Experimental Setup

As we are interested in examining training of small models (<1B parameters) with a single GPU, we experiment with four tiny LLMs, each containing at most 1 billion parameters: *Qwen3-0.6B*, *Qwen2.5-0.5B-Instruct*, *Falcon3-1B-Instruct*, and *Llama3.2-1B-Instruct*. Despite their smaller parameter footprint, models like Qwen3-0.6B achieve performance on par with much larger counterparts, such as *Qwen2.5-Math-7B-Instruct*. All models are fine-tuned using a single NVIDIA H100 GPU. Training is conducted on four math datasets: (1) DeepScaleR-uniform, (2) DeepScaleR-easy and (3) DeepScaleR-difficult, each comprising 10,000 uniformly, low-difficulty, or high-difficulty samples drawn from the DeepScaleR dataset [20]; and (4) GSM8K [4], using all 7,473 available training problems. If not otherwise stated, we use *Qwen3-Embedding-0.6B* to obtain semantic embeddings for clustering. Each experiment is run with three random seeds to account for variability in reinforcement learning, ensuring that results are stable and not reliant on a particular initialization. All training is implemented and executed using the Open-R1 codebase [14].

For DeepScaleR and GSM8K training, our method selects as many as **100 training examples** using the reduction method described in Section 3. Our selection process is lightweight and introduces minimal computational overhead compared to standard reinforcement learning baselines such as R1. For example, when training Qwen3-0.6B on an H100 GPU, our training time is comparable to the R1 baseline trained on a uniformly sampled subset of DeepScaleR (15h37m vs. 15h23m 1×GPU H100 runtime, respectively), with only a slight increase due to the bandit-based selection.

Evaluation We use four benchmarks that span different reasoning types and difficulty levels: **GSM8K** [4], consists of diverse grade school math problems; **MATH500**, a 500-sample subset of the MATH dataset [11]; **AIME24** and **AIME25**, comprising problems from the 2024 and 2025 American Invitational Mathematics Examination, respectively. We report the extractive match scores for all settings, following Lighteval's evaluation framework [10].

Method	**GSM8K**	**MATH500**	**AIME24**	**AIME25**
Base	78.0	75.4	10.0	<u>16.7</u>
π_1	78.1	75.4	<u>13.3</u>	<u>16.7</u>
π_2	<u>79.0</u>	74.2	6.7	10.0
Ordered	77.9	74.8	<u>13.3</u>	13.3
SFT	$77.6_{0.5}$	$74.8_{0.4}$	$7.8_{1.9}$	$14.4_{5.1}$
R1	$77.9_{1.0}$	$71.6_{1.3}$	$8.9_{5.1}$	$12.2_{5.1}$
AdaRFT	$78.9_{0.3}$	<u>$75.9_{1.5}$</u>	$12.2_{3.9}$	$14.4_{3.9}$
SPaRFT	$\mathbf{79.5}_{0.6}$	$\mathbf{78.0}_{1.3}$	$\mathbf{14.4}_{1.9}$	$\mathbf{18.9}_{5.1}$

Table 1: Results using Qwen3-0.6B across different datasets. We report extractive match scores (mean_{std}) at the final training checkpoint, averaged over 3 seeds (except for the Base, π_1, π_2 and Ordered baselines). Best results are highlighted in bold, and the second-best are underlined.

Baselines **Base** refers to the pretrained model without any fine-tuning. π_1 and π_2 are models trained on one and two examples from the DeepScaleR dataset, respectively, following the 1-shot RLVR protocol [34]. **SFT** denotes the supervised fine-tuning baseline. **Ordered** [2] is a curriculum baseline in which training begins with easier examples and gradually progresses to harder ones. **R1** is the RL baseline trained with the standard GRPO algorithm without an SFT cold start, as in DeepSeek-R1 [5]. **AdaRFT** is a curriculum learning approach that selects examples based on a difficulty threshold [28].

5 Experimental Results

5.1 Benchmarking SPaRFT with Qwen3-0.6B

5.1.1 Setup

We fine-tune *Qwen3-0.6B* in a zero-shot setup. Despite its small size, it is a strong baseline and supports a *thinking mode* using `<think>` tags. Following [34], we use one seed for π_1 and π_2. For the *Ordered* baseline, we also use one seed, as the order of training examples is fixed. Further details on training hyperparameters are provided in Appendix A.6.

5.1.2 Main Results

Table 1 presents test accuracies from the final training checkpoint using *Qwen3-0.6B*. SPaRFT consistently achieves the highest performance across all benchmarks, despite using only 100 training examples—*two orders of magnitude fewer than other baselines*. On GSM8K, SPaRFT reaches $79.5 \pm 0.6\%$, outperforming all baselines including AdaRFT ($78.9 \pm 0.3\%$) and R1 ($77.9 \pm 1.0\%$). On the MATH500 benchmark, SPaRFT yields $78.0 \pm 1.3\%$, improving over the next-best result ($75.9 \pm 1.5\%$) by a margin of 2.1 percentage points. Notably, SPaRFT also shows strong gains in harder math datasets: it reaches $18.9 \pm 5.1\%$ on AIME25, outperforming AdaRFT ($14.4 \pm 3.9\%$) by 4.5 percentage points, and achieves the best score on AIME24 with $14.4 \pm 1.9\%$. These results highlight SPaRFT's data efficiency and robustness across tasks of varying difficulty.

5.2 SPaRFT Can Work With Various Datasets

5.2.1 Setup

We evaluate our method on three distinct training sets: (1) GSM8K; (2) DeepScaleR–Easy, a subset of DeepScaleR with primarily low-difficulty questions; and (3) DeepScaleR–Difficult, a subset with mainly high-difficulty questions. All experiments use *Qwen3-0.6B* as the backbone, and we adopt the same reward functions and hyperparameters as in Section 5.1. We compare with the top-3 baselines, excluding π_1 and π_2 as they are unavailable in GSM8K or always present in both DeepScaleR subsets.

5.2.2 Main Results

We report results in Table 2. Across all settings, SPaRFT consistently outperforms standard SFT, achieving gains of 2.9–5.5 % on GSM8K and up to 7.8 % on the AIME benchmarks. R1 generally ranks second, especially on the easier splits, while AdaRFT falls 1–2 % behind in most cases. Notably,

Set	Method	GSM8K	MATH.	AIME24	AIME25
1	SFT	$74.1_{0.6}$	$70.4_{2.4}$	$6.7_{0.0}$	$11.1_{1.9}$
	R1	$\underline{78.6_{1.4}}$	$\mathbf{75.0}_{1.8}$	$\underline{16.7_{3.3}}$	$\underline{15.6_{5.1}}$
	AdaRFT	$77.4_{0.5}$	$74.7_{0.7}$	$8.9_{1.9}$	$13.3_{5.8}$
	SPaRFT	$\mathbf{79.9}_{0.8}$	$\mathbf{75.0}_{0.5}$	$\mathbf{17.8}_{1.9}$	$\mathbf{18.9}_{1.9}$
2	SFT	$77.2_{1.0}$	$\underline{75.0_{0.6}}$	$7.8_{1.9}$	$13.3_{3.4}$
	R1	$\underline{77.7_{0.7}}$	$74.1_{0.2}$	$11.1_{3.8}$	$11.1_{1.9}$
	AdaRFT	$77.6_{0.5}$	$72.1_{0.2}$	$\mathbf{13.3}_{0.0}$	$\underline{14.4_{5.1}}$
	SPaRFT	$\mathbf{78.7}_{1.9}$	$\mathbf{75.4}_{0.9}$	$\mathbf{13.3}_{3.3}$	$\mathbf{15.6}_{5.1}$
3	SFT	$77.9_{0.4}$	$73.8_{0.5}$	$8.9_{5.1}$	$\underline{15.5_{1.9}}$
	R1	$77.3_{0.3}$	$\underline{75.0_{1.2}}$	$11.1_{10.2}$	$14.4_{1.9}$
	AdaRFT	$\mathbf{78.9}_{0.4}$	$73.6_{0.7}$	$\underline{14.4_{5.1}}$	$12.2_{7.7}$
	SPaRFT	$\mathbf{78.9}_{0.7}$	$\mathbf{76.1}_{1.2}$	$\mathbf{16.7}_{3.3}$	$\mathbf{18.9}_{5.1}$

Table 2: Results (mean_{std}) averaged over 3 training seeds on four datasets using Qwen3-0.6B. MATH. refers to the MATH500 benchmark. (1) corresponds to training on GSM8K, (2) on DeepScaleR–Easy, and (3) on DeepScaleR–Difficult. Bold values indicate the best result; underlined values indicate the second-best.

	Method	GSM8K	MATH500	AIME24	AIME25
1	Base	$\underline{37.9}$	$\mathbf{17.2}$	0.0	0.0
	AdaRFT	$37.6_{1.8}$	$\underline{17.1_{2.2}}$	$0.0_{0.0}$	$0.0_{0.0}$
	SPaRFT	$\mathbf{39.7}_{2.3}$	$16.3_{1.5}$	$0.0_{0.0}$	$\mathbf{2.2}_{1.9}$
2	Base	27.9	17.0	0.0	0.0
	AdaRFT	$\underline{29.0_{0.5}}$	$\underline{17.4_{1.2}}$	$0.0_{0.0}$	$0.0_{0.0}$
	SPaRFT	$\mathbf{30.1}_{0.6}$	$\mathbf{18.3}_{0.8}$	$\mathbf{1.1}_{1.9}$	$\mathbf{1.1}_{1.9}$
3	Base	26.3	$\underline{20.0}$	0.0	0.0
	AdaRFT	$\underline{30.5_{0.7}}$	$18.9_{1.1}$	$0.0_{0.0}$	$0.0_{0.0}$
	SPaRFT	$\mathbf{32.9}_{0.9}$	$\mathbf{22.0}_{0.7}$	$\mathbf{2.2}_{1.9}$	$\mathbf{1.1}_{1.9}$

Table 3: Results using other LLMs across datasets. We report extractive match scores (mean_{std}) at the final checkpoint, averaged over 3 seeds (1 seed for Base). (1) corresponds to Falcon3-1B-Instruct, (2) corresponds to Llama-3.2-1B-Instruct, and (3) corresponds to Qwen2.5-0.5B-Instruct. Best results are in bold, while second-best results are underlined.

when trained on the difficult subset (3), SPaRFT attains a 2.3 % improvement on MATH500 and more than doubles AIME24 accuracy relative to SFT. These results confirm that SPaRFT not only enhances overall accuracy but also yields the greatest benefits on the most challenging training set.

5.3 SPaRFT Can Help Diverse LLM Learners

5.3.1 Setup

We evaluate SPaRFT on additional LLMs with $\leq$1B parameters: *Qwen2.5-0.5B-Instruct*, *Falcon3-1B-Instruct*, and *Llama3.2-1B-Instruct*—compact models with strong reasoning, language, code, and math capabilities. We compare against the base model and AdaRFT, the most consistent and second-best method in Section 5.1. All models use the same zero-shot setup, except *Llama3.2-1B-Instruct*, which requires a single in-context example per training instance to yield valid correctness rewards [17].

5.3.2 Main Results

Table 3 reports the performance of various tiny LLMs across multiple datasets. SPaRFT emerges as the clear winner, achieving the top rank in 10 out of 16 cases. By contrast, the next best baseline, AdaRFT, attains the highest score in only one case. Overall, SPaRFT outperforms all baselines by margins of up to 6% on GSM8K, 3% on MATH500, and 2% on both AIME24 and AIME25. Moreover, neither the Base model nor AdaRFT succeeds on the more challenging AIME24 and AIME25 test sets,

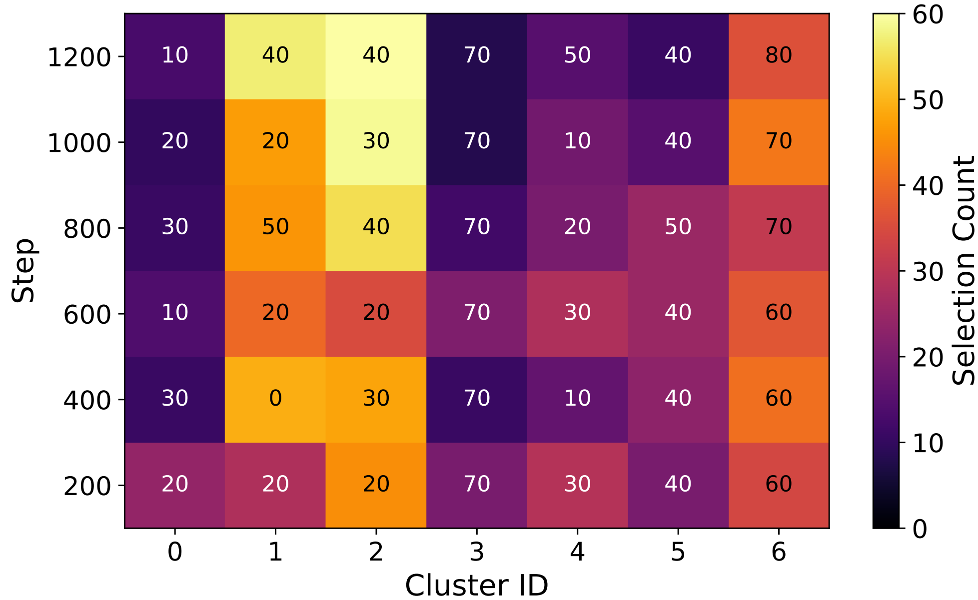

Figure 2: Multi-arm Bandit Cluster Selection and its Impact on Cluster Solve Rates During Training.

underscoring the importance of selecting appropriate training samples and employing an optimized curriculum.

6 Ablation Studies and Model Analyses

6.1 Multi-arm Bandit Analysis

6.1.1 Empirical Convergence

SPaRFT uses MAB to guide curriculum learning, making it crucial to understand its behavior during training. Figure 2 shows a heatmap of cluster solve rates and bandit selections over time using *Qwen3-0.6B* with 7 clusters. At step 200, selection is nearly uniform, with all clusters sampled at similar frequencies. As training progresses, clusters 1 and 2 become increasingly favored, indicating higher utility for learning. From step 600 onward, this preference becomes pronounced.

Solve Rate Trends. Clusters 1 and 2 show the largest solve rate gains, increasing from 20% at step 200 to 40% by step 1200. These clusters appear moderately difficult and provide strong learning signals. In contrast, cluster 3, which is rarely picked, starts high at 70% and remains flat, suggesting it is too easy to help the model improve. Other clusters, such as 4 and 5, show modest gains with continued exploration, except for cluster 0. By the end of training, the bandit converges to favor clusters 1 and 2, which consistently yield the most benefit.

6.2 Data Reduction Analysis

6.2.1 Selected Training Samples Difficulty

We analyze the selected training examples to understand how SPaRFT constructs its curriculum. Figure 3 shows the difficulty distribution of questions selected on the DeepScaleR-uniform dataset. SPaRFT consistently chooses examples across the full difficulty range—from easy (near 0) to hard (near 100)—ensuring balanced coverage. This diversity enables training on a broad range of problems, avoiding overfitting to simple or complex cases. Notably, this balance emerges without

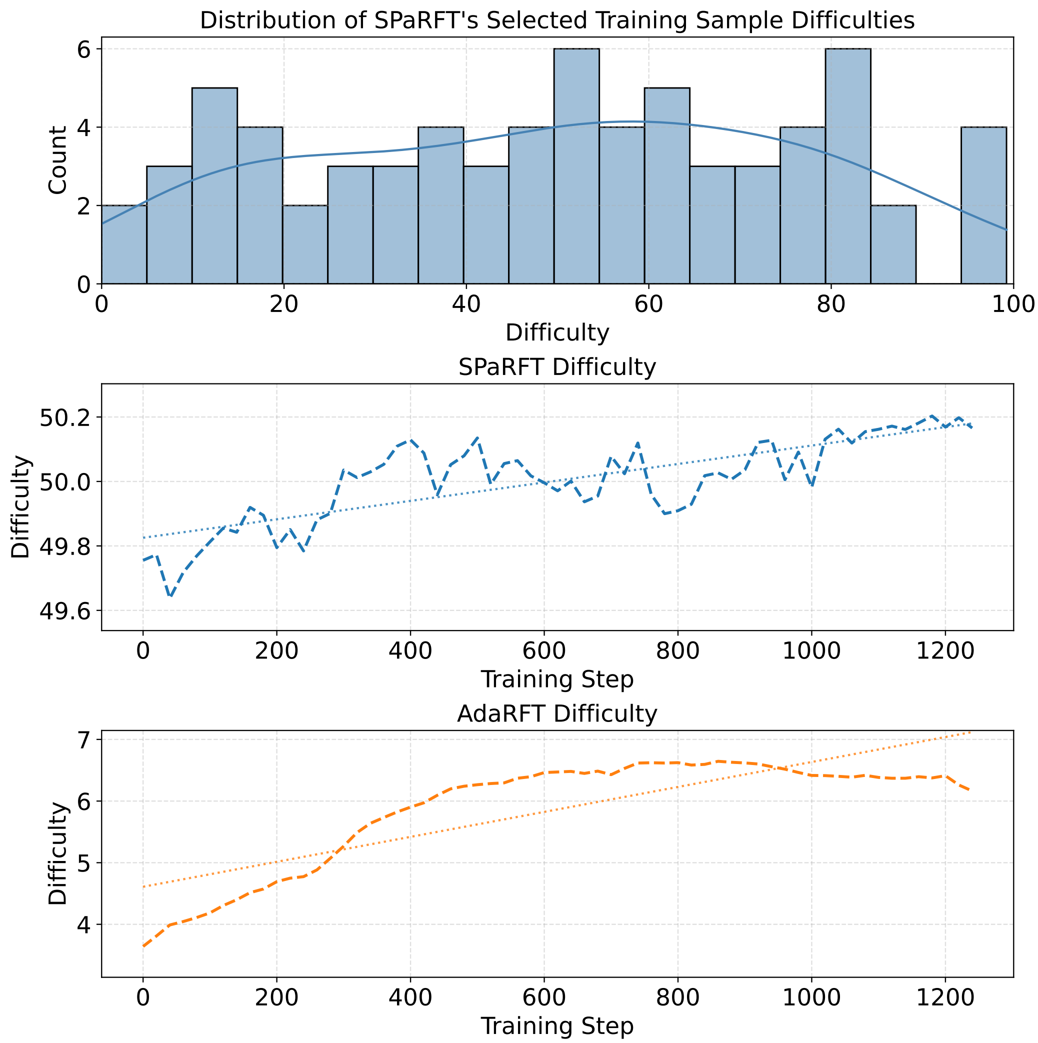

Figure 3: Top: Difficulty distribution of training examples in SPaRFT. Middle: Difficulty of training examples selected by SPaRFT over time. Bottom: Difficulty of training examples selected by AdaRFT over time. Note: The Middle and Bottom plots are shown separately due to the large difference in their difficulty scales.

manual difficulty constraints, highlighting the effectiveness of SPaRFT's clustering and selection strategy.

6.2.2 Average Sample Difficulty Comparison

Figure 3 (middle and bottom) compare the average difficulty of training examples selected over time by SPaRFT and AdaRFT. Although AdaRFT has access to the entire training dataset, its performance with tiny LLMs struggles to reach medium and hard examples during training, due to its heavy reliance on a threshold mechanism that is highly sensitive and affects sample selection. In contrast, SPaRFT tends to favor medium to hard instances, avoiding overemphasis on easy examples as seen in AdaRFT. We believe this advantage stems from our clustering strategy, which effectively captures both semantic diversity and difficulty. Each cluster contains a broad mixture of examples, allowing our selection process to explore harder examples without compromising variety. This contributes to the superior performance of SPaRFT compared to other curriculum or selection methods.

Method	GSM8K	MATH500	AIME24	AIME25
SPaRFT	**79.5±0.6**	**78.0±1.3**	**14.4±1.9**	**18.9±5.1**
SPaRFT^{-}	78.8±0.4	76.4±0.7	12.2±3.9	11.1±1.9

Table 4: Mean $\pm$ standard deviation over 3 seeds on the DeepScaleR-uniform dataset using Qwen3-0.6B as the base model. SPaRFT^{-} denotes the variant without data reduction. Best results are in bold.

6.2.3 Impact of Data Reduction

To evaluate the effectiveness of the data reduction phase in our method, we conduct ablation experiments where data reduction is disabled—that is, all examples within each cluster are retained without filtering. Results are shown in Table 4. Without data reduction, each cluster contains significantly more examples than the batch size B. As shown, SPaRFT without data reduction still outperforms the Base baseline but underperforms the full version with selection. We attribute this to increased randomness when sampling from large clusters without filtering. Specifically, the difficulty of sampled examples may vary widely, weakening the consistency and effectiveness of the learning signal during training.

6.2.4 Effect Of Diverse Sample Selection

We show the impact of sample selection strategies for the data reduction on the performance of different LLMs in Appendix's Figure 4. Specifically, we compare our method against two baselines: *random*, which randomly selects training examples for the cluster, and *closest*, which selects the closest examples to the cluster center. As observed, selecting diverse examples with our method consistently yields the highest performance across four datasets. Interestingly, the *closest* strategy performs worse than *random* in most cases. We hypothesize that this is because the examples nearest to the cluster center tend to be overly similar, thus failing to provide sufficient coverage and variation for effective learning.

6.3 Clustering Effects

6.3.1 Number Of Clusters

We illustrate how SPaRFT's performance varies with the number of clusters K in Appendix's Figure 5. Performance is highest with a moderate number of clusters (7 to 10), while both lower (1) and higher (20) cluster counts lead to decreased accuracy. This trend suggests that moderate clustering strikes a better balance between specialization and generalization, whereas too few clusters limit differentiation and too many may over-fragment the data. Careful selection of the cluster count is therefore important for maximizing performance.

6.3.2 Number Of Samples In Each Cluster

We vary the number of samples per cluster l to assess its impact, and present the results in Figure 6. As shown, performance peaks at our default setting of $l = 10$. In contrast, using $l = 1$ yields lower performance, likely because the selected examples lack sufficient diversity to provide a strong learning signal. Larger values of $l \in [100, 300]$ also lead to degraded performance, which we attribute to increased randomness when sampling too many examples per arm, especially when $l \gg B$. Due to compute limits, we couldn't extensively tune l between 10–100, where better results might be possible. Overall, setting $l \sim B$ achieves the most reasonable results.

6.3.3 Importance Of Concatenating Difficulty In Clustering

To quantify the contribution of initial difficulty estimates to cluster quality, we repeat our clustering pipeline after ablating the difficulty signal (i.e., omitting the *Qwen2.5-7B*–derived difficulty scores described in Section 3.1.1). Appendix's Figure 7 shows the downstream performance of *Qwen3-0.6B* on each dataset under this "no-difficulty" condition. Across all benchmarks, we observe a consistent drop in accuracy and reasoning metrics when difficulty information is excluded from the clustering stage. This degradation confirms that task difficulty is a key signal for grouping examples into meaningful clusters, and that its inclusion materially improves the effectiveness of our curriculum selection.

7 Conclusion

We introduced **SPaRFT**, a lightweight framework that enables efficient reasoning in small language models through clustering and adaptive curriculum learning. SPaRFT selects compact, diverse training subsets and dynamically adapts training focus based on model performance. Experiments show that SPaRFT achieves competitive accuracy with significantly fewer samples. These results highlight the effectiveness of combining semantic clustering with performance-driven curricula to unlock reasoning in small models using minimal resources.

References

[1] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery.

[3] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. *Advances in neural information processing systems*, 27, 2014.

[4] 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.

[5] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[6] Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, KaShun SHUM, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[8] Google. Introducing gemini 2.0: Our new ai model for the agentic era, 2024. Accessed: 2025-03-05.

[9] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz

Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ

Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024.

[10] Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. Lighteval: A lightweight framework for llm evaluation, 2023.

[11] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

[12] Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025.

[13] Tianjin Huang, Ziquan Zhu, Gaojie Jin, Lu Liu, Zhangyang Wang, and Shiwei Liu. SPAM: Spike-aware adam with momentum reset for stable LLM training. In *The Thirteenth International Conference on Learning Representations*, 2025.

[14] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025.

[15] Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Salman Khan, and Fahad Shahbaz Khan. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*, 2025.

[16] Hung Le, Majid Abdolshah, Thommen K George, Kien Do, Dung Nguyen, and Svetha Venkatesh. Episodic policy gradient training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7317–7325, 2022.

[17] Hung Le, Dai Do, Dung Nguyen, and Svetha Venkatesh. Reasoning under 1 billion: Memory-augmented reinforcement learning for large language models. *arXiv preprint arXiv:2504.02273*, 2025.

[18] Xuefeng Li, Haoyang Zou, and Pengfei Liu. Limr: Less is more for rl scaling, 2025.

[19] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[20] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. `https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca3030` 2025. Notion Blog.

[21] OpenAI. Hello gpt-4o, 2024.

[22] 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.

[23] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[24] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[25] Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 3rd edition, 1976.

[26] Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on thompson sampling, 2020.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[28] Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. Efficient reinforcement finetuning via adaptive curriculum learning, 2025.

[29] Mingyang Song, Mao Zheng, Zheng Li, Wenjie Yang, Xuan Luo, Yue Pan, and Feng Zhang. Fastcurl: Curriculum reinforcement learning with stage-wise context scaling for efficient training r1-like reasoning models, 2025.

[30] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[31] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[32] Guoxia Wang, Shuai Li, Congliang Chen, Jinle Zeng, Jiabin Yang, Tao Sun, Yanjun Ma, Dianhai Yu, and Li Shen. Adagc: Improving training stability for large language model pretraining, 2025.

[33] 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, 2019.

[34] Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example, 2025.

[35] Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, Haosheng Zou, Yongchao Deng, Shousheng Jia, and Xiangzheng Zhang. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond, 2025.

[36] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025.

[37] Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

A Appendix

A.1 Algorithm for SPaRFT

In this section, we provide the pseudo-code for SPaRFT in Algorithm 1.

A.2 Convergence of the Thompson Sampling Scheduler

We analyze the convergence of the Thompson Sampling scheduler used in SPaRFT. Each data cluster is treated as an arm in a multi-armed bandit. At step t, let $\pi_{\theta^{(t)}}$ denote the model with parameters $\theta^{(t)}$. The expected reward (solve rate) of cluster $\mathcal{C}_k$ is defined as:

$$\mu_k^{(t)} = \mathbb{E}_{x\sim\mathcal{C}_k}\left[\Pr(\pi_{\theta^{(t)}}(x) = \text{ correct})\right].$$

where $\Pr\left(\pi_{\theta^{(t)}}(x) = \text{correct}\right)$ denotes the probability that the model produces a correct answer for input x.

We already have: (1) the model is trained using gradient clipping with threshold $G_{\max}$; (2) the learning rate α_t follows a cosine decay schedule with warmup and is therefore non-increasing and vanishes as $t \to \infty$; and (3) the expected rewards satisfy $\mu_k^{(t)} \in [0, 1]$ for all clusters k. These three properties are directly enforced in the SPaRFT implementation.

To complete the convergence analysis, we now bound the drift of each cluster's expected reward. Define

$$f_k(\theta) \;=\; \mathbb{E}_{x\sim\mathcal{C}_k}\big[\Pr\big(\pi_\theta(x) \;=\text{correct}\big)\big],$$

and note that since each layer of our base model is continuously differentiable, so is $f_k(\theta)$ [7]. We assume that, in practice, gradient clipping at norm $G_{\max}$ together with a bounded initialization prevents the parameters $\{\theta^{(t)}\}$ from diverging excessively, effectively keeping them in some large but fixed ball $\{\|\theta\| \le R\}$. Empirical studies on LLM training have repeatedly observed that clipped updates under cosine-decay schedules yield stable trajectories without catastrophic parameter growth (e.g., [32, 13]).

Under this assumption, the extreme-value theorem [25] guarantees the existence of a constant $H < \infty$ such that

$$\|\nabla_\theta f_k(\theta)\| \;\le\; H \quad \forall\, \|\theta\| \le R.$$

Moreover, each gradient step with cosine-decay learning rate α_t and gradient clipping satisfies

$$\|\theta^{(t+1)} - \theta^{(t)}\| \;\le\; \alpha_t\, G_{\max}.$$

Applying the mean-value theorem [25] then yields

$$\begin{aligned} \big|\mu_k^{(t+1)} - \mu_k^{(t)}\big| &= \big|f_k(\theta^{(t+1)}) - f_k(\theta^{(t)})\big| \\ &\le H\,\|\theta^{(t+1)} - \theta^{(t)}\| \\ &\le H\,G_{\max}\,\alpha_t \;=\; \varepsilon_t, \end{aligned} \tag{10}$$

where $\varepsilon_t \to 0$ as $\alpha_t \to 0$. Thus, even though a formal boundedness proof is elusive, we obtain the desired vanishing drift $\big|\mu_k^{(t+1)} - \mu_k^{(t)}\big| \le \varepsilon_t$ for every cluster k.

Let $V_T = \sum_{t=1}^{T-1} \max_k |\mu_k^{(t+1)} - \mu_k^{(t)}|$ denote the total reward variation up to step T. The bound above implies

$$V_T \le \sum_{t=1}^{T-1} \varepsilon_t.$$

Since $\alpha_t = O(1/t)$ after warmup, we have $\sum_{t=1}^{T-1} \varepsilon_t = O(\log T)$, and hence $V_T = O(\log T)$. This sublinear variation is sufficient to ensure convergence of Thompson Sampling in the non-stationary bandit setting, as shown in prior work [3]. Therefore, the Thompson Sampling scheduler concentrates on the cluster(s) with the highest current expected reward as $t \to \infty$. This ensures that the bandit-based curriculum used in SPaRFT converges to the most informative training distribution over time.

Algorithm 1 SPaRFT

Input: Policy π_θ, Dataset $\mathcal{D}$, Embedding model ϕ, Clusters K, Batch size B, RL algorithm $\mathcal{A}$, Small positive constant ϵ
Output: Trained policy π_θ

```
// Phase 1: Cluster-based Data Reduction
for each x_i ∈ D do
    s_i ← PCA(φ(x_i)),    d_i ← difficulty(x_i)
    e_i ← s_i ⊕ d_i
end for
Run K-means on {e_i} to form {C_k}_{k=1}^K
Select l diverse examples S_k ⊂ C_k via farthest sampling
D_train ← ⋃_k {x_i | i ∈ S_k}
// Phase 2: Bandit-driven Curriculum
Initialize Multi-arm Bandit
for each k do
    Initialize R_k ← 0, n_k ← 0
end for
while training not finished do
    for each cluster k = 1, ..., K do
        Sample reward estimate: μ̃_k ~ N(−R_k/(n_k+ϵ), 1/(n_k+ϵ))
    end for
    Select cluster: c_t ← argmax_k μ̃_k
    Sample batch X ⊂ C_{c_t}
    Generate responses G ← π_θ(X)
    Calculate correctness r_i, i = 1, ..., K
    Compute average reward r_avg ← (1/B) Σ_{i=1}^B r_i
    Update policy: π_θ ← A(π_θ, X, G, r_avg)
    Update bandit stats:
        R_{c_t} ← R_{c_t} + r_avg,    n_{c_t} ← n_{c_t} + 1
end while
return π_θ
```

Embedding	GSM8K	MATH500	AIME24	AIME25
Qwen3[1]	32.9±0.9	22.0±0.7	2.2±1.9	1.1±1.9
Qwen2.5[2]	31.6±0.6	21.0±1.2	1.1±1.9	0.0±0.0

Table 5: Results using DeepScaleR as training data for Qwen2.5-0.5B-Instruct, evaluated with two embedding models: Qwen3[1] and Qwen2.5[2].

A.3 Impact Of Embedding Model Choice

To assess SPaRFT's robustness to different semantic embedding backbones, we compare its default sentence encoder with an alternative based on *Qwen2-1.5B-Instruct* (*Alibaba-NLP/gte-Qwen2-1.5B-instruct*). We denote Qwen3[1] as the baseline using *Qwen3-Embedding-0.6B*, which is adopted in SPaRFT, and Qwen2.5[2] as the baseline using *Alibaba-NLP/gte-Qwen2-1.5B-instruct*. Table 5 reports zero-shot performance on four math reasoning benchmarks. Results show that SPaRFT yields nearly identical performance across both embedding models, with the Qwen3-based encoder showing a slight advantage. This plug-and-play flexibility demonstrates that any high-quality pre-trained encoder can be seamlessly integrated into our framework without retraining.

A.4 Impact Of The Number Of PCA Component

In SPaRFT, we first apply PCA to reduce the dimensionality of the latent vectors extracted from the pretrained Sentence-BERT model. By default we use 50 principal components; here, we vary this number to study its effect on final performance. Using Qwen3-0.6B as the base model and training on the DeepScaleR-uniform dataset, the results are shown in Table 6. We observe that smaller to

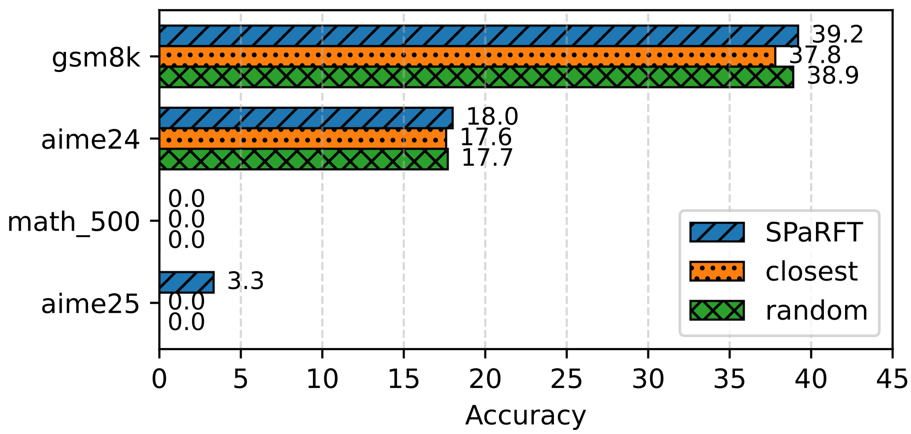

Figure 4: Comparison of selection strategies across datasets. Selecting diverse examples with SPaRFT outperforms both random and closest baselines. Closest examples perform worse, likely due to reduced variety within each cluster.

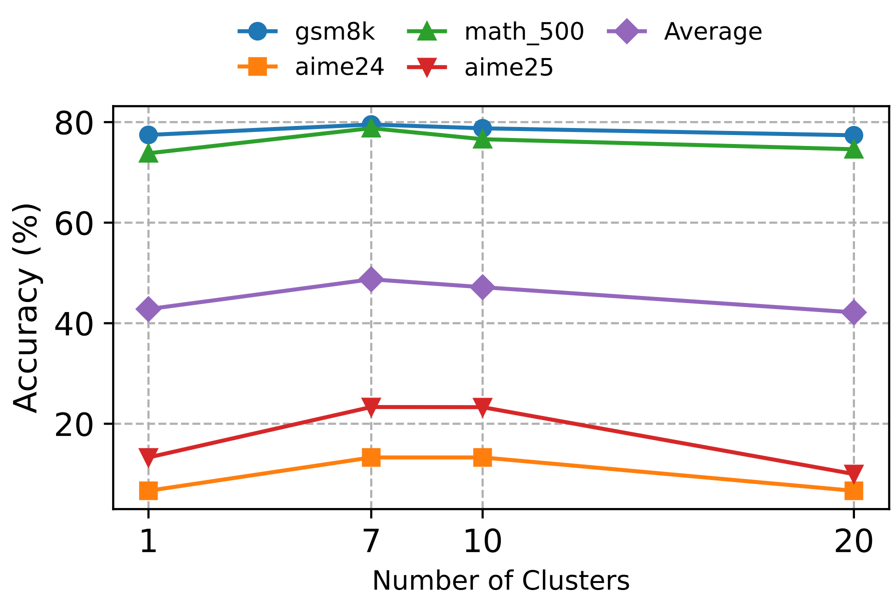

Figure 5: Results for 1 seed with Qwen3-0.6B and different number of clusters.

moderate numbers of components (10 or 50) yield the best performance, whereas larger values (100 or 300) lead to a decline. We hypothesize that very high-dimensional embeddings overwhelm the difficulty signal prior to clustering, resulting in poorer downstream performance.

A.5 DeepScaleR difficulty distribution

We provide the difficulty score distributions of three DeepScaleR subsets: DeepScaleR Uniform, DeepScaleR Easy, and DeepScaleR Difficult, as shown in Figure 8. Each subset exhibits distinct difficulty characteristics, reflecting the varying levels of challenge present in the data. The distributions are grouped into bins of size 10, allowing for clear comparison of how problem difficulty varies across these subsets.

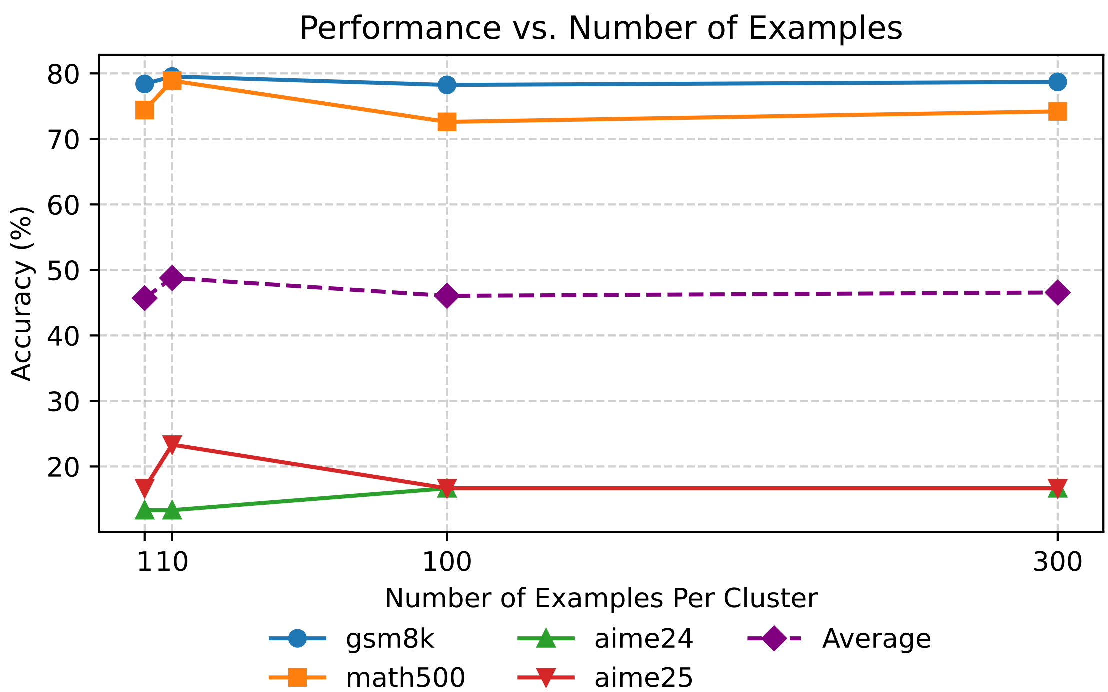

Figure 6: Results for 1 seed with Qwen3-0.6B and different number of samples per cluster.

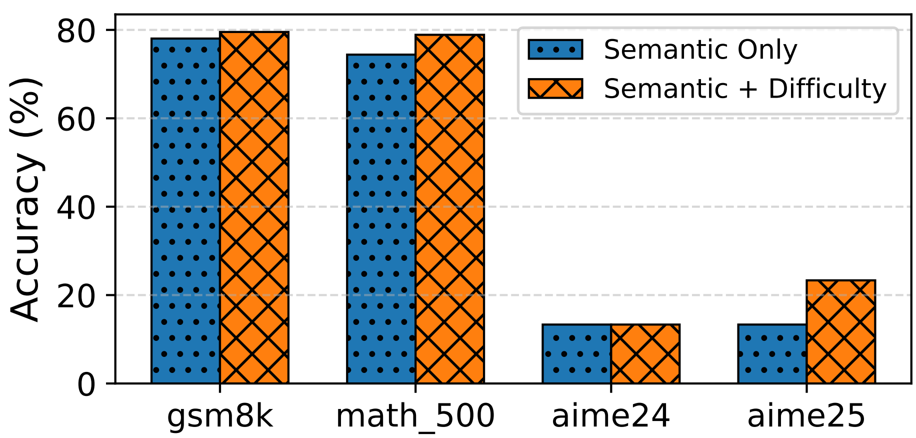

Figure 7: Effect of Removing Difficulty in Clustering on Performance.

A.6 SPaRFT training details

System Prompt

Following [14], the system prompt asks the model to generate the answer with clear requirements, with reasoning and answer following the format, as described in Figure 9.

Training Hyperparameters

In this section, we provide the training details of SPaRFT in Table 8.

Number of PCA Components	GSM8K	MATH500	AIME24	AIME25	Average
10	79.2	74.4	13.3	20.0	46.7
50	79.5	78.9	13.3	23.3	48.8
100	79.8	76.6	6.7	13.3	44.1
300	79.0	75.8	16.7	10.0	45.4

Table 6: Results on 1 same seed on the DeepScaleR-uniform dataset using Qwen3-0.6B as the base model with different number of PCA components.

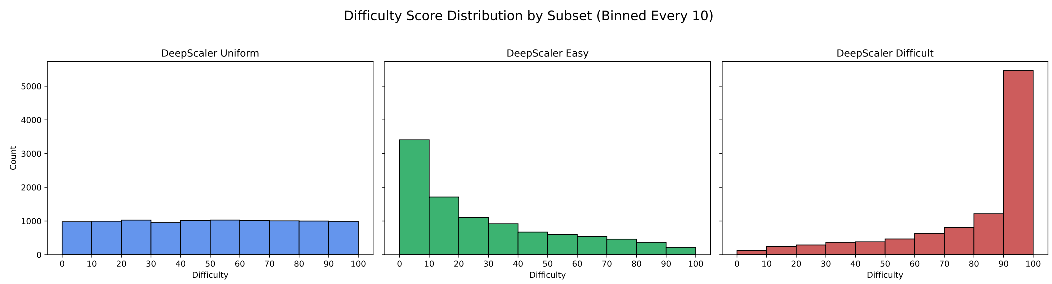

Figure 8: DeepScaleR subsets' difficulty distributions.

Number of clusters

We provide the number of clusters used for different settings of SPaRFT in Table 9. While the optimal number of clusters varies across datasets, it remains within a moderate range, consistent with our observations in Section 6.3.

Model and data references

We list the links to the LLM models and datasets in Table 7.

A.7 Response Examples

We present several sampled responses of SPaRFT in Table 10 and 11.

A.8 Selected Example Analysis

In this section, we provide details on the selected examples by SPaRFT. For each cluster, we show 2 selected examples selected by our method using *Qwen3-0.6B*, with the DeepScaleR-uniform dataset. The examples are shown from Table 12 to Table 18

SYSTEM PROMPT

You are a helpful assistant. Please reason step by step, and put your final answer within \boxed{}.

Figure 9: System prompt used in our experiments.

Models/Datasets	URL
Qwen3-Embedding-0.6B	`https://huggingface.co/Qwen/Qwen3-Embedding-0.6B`
Qwen2.5-0.5B-Instruct	`https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct`
Llama3.2-1B-Instruct	`https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct`
Falcon3-1B-Instruct	`https://huggingface.co/tiiuae/Falcon3-1B-Instruct`
Alibaba-NLP/gte-Qwen2-1.5B-instruct	`https://huggingface.co/Alibaba-NLP/gte-Qwen2-1.5B-instruct`
Qwen/Qwen2.5-Math-7B-Instruct	`https://huggingface.co/Qwen/Qwen2.5-Math-7B-Instruct`
DeepScaleR	`https://huggingface.co/datasets/agentica-org/DeepScaleR-Preview`
GSM8K	`https://huggingface.co/datasets/openai/gsm8k`
MATH-500	`https://huggingface.co/datasets/HuggingFaceH4/MATH-500`
AIME24	`https://huggingface.co/datasets/math-ai/aime24`
AIME25	`https://huggingface.co/datasets/math-ai/aime25`

Table 7: Models and Datasets Details.

Parameters	Value
Number of examples per cluster (l)	10
Number of PCA components	50
Batch size (B)	8
Number of generation per step (G)	8
Maximum completion length (L)	1200
Initial learning rate (α)	$5e^{-6}$
Weight Decay	0.1
Warmup Ratio	0.1
lr_scheduler_type	cosine
Adam β_1	0.9
Adam β_2	0.99
bf16	True
Per device train batch size	8
Gradient accumulation steps	8
Max grad norm (G_{norm})	0.1
ϵ	$1e^{-6}$

Table 8: Parameters used in SPaRFT.

Model	Train dataset	Number of clusters
Qwen3-0.6B	DeepScaleR-uniform	7
	DeepScaleR-easy	8
	DeepScaleR-difficult	10
	GSM8K	10
Falcon3-1B-Instruct	DeepScaleR-uniform	6
Llama-3.2-1B-Instruct	DeepScaleR-uniform	8
Qwen2.5-0.5B-Instruct	DeepScaleR-uniform	7

Table 9: Number of clusters used for different settings of our method.

Table 10: Qwen3-0.6B fine-tuned with SPaRFT responses.

Question

Let f be the function defined by $f(x) = ax^2 - \sqrt{2}$ for some positive a. If $f(f(\sqrt{2})) = -\sqrt{2}$ then $a =$. Please reason step by step, and put your final answer within `\boxed{}`.

Answer

To solve for a, we need to find the value of a such that $f(f(\sqrt{2})) = -\sqrt{2}$, where $f(x) = ax^2 - \sqrt{2}$. First, let's compute $f(\sqrt{2})$:

$$f(\sqrt{2}) = a(\sqrt{2})^2 - \sqrt{2} = a(2) - \sqrt{2} = 2a - \sqrt{2}$$

Now, we need to compute $f(f(\sqrt{2}))$:

$$f(f(\sqrt{2})) = f(2a - \sqrt{2}) = a(2a - \sqrt{2})^2 - \sqrt{2}$$

Let's expand the square term:

$$(2a - \sqrt{2})^2 = (2a)^2 - 2(2a)(\sqrt{2}) + (\sqrt{2})^2 = 4a^2 - 4a\sqrt{2} + 2$$

...

So the discriminant is zero, which means there is a repeated root:

$$a = \frac{2\sqrt{2}}{4} = \frac{\sqrt{2}}{2}$$

Therefore, the value of a is $\frac{\sqrt{2}}{2}$.

Let me double-check the calculations to ensure there are no errors. Starting from the beginning:

1. $f(\sqrt{2}) = a(2) - \sqrt{2} = 2a - \sqrt{2}$
2. Then $f(f(\sqrt{2})) = f(2a - \sqrt{2}) = a(2a - \sqrt{2})^2 - \sqrt{2}$
3. Expanding $(2a - \sqrt{2})^2$ gives $4a^2 - 4a\sqrt{2} + 2$
4. Multiplying by a and subtracting $\sqrt{2}$ gives $4a^3 - 4a^2\sqrt{2} + 2a - \sqrt{2}$
5. Setting equal to $-\sqrt{2}$ leads to the equation $4a^3 - 4a^2\sqrt{2} + 2a = 0$
6. Factoring gives $2a(2a^2 - 2a\sqrt{2} + 1) = 0$, leading to the quadratic with discriminant zero.

Yes, the steps seem correct. The discriminant being zero implies a repeated root, which gives $a = \frac{\sqrt{2}}{2}$. Therefore, the answer is $\boxed{\dfrac{\sqrt{2}}{2}}$.

Final Answer

$$\boxed{\dfrac{\sqrt{2}}{2}}$$

Table 11: Qwen3-0.6B fine-tuned with SPaRFT responses.

Question

A set S of points in the xy-plane is symmetric about the origin, both coordinate axes, and the line $y = x$. If $(2, 3)$ is in S, what is the smallest number of points in S?. Please reason step by step, and put your final answer within `\boxed{}`.

Answer

To solve this problem, we need to determine the smallest number of points in the set S, given that the set is symmetric about the origin, both coordinate axes, and the line $y = x$. Additionally, the point $(2, 3)$ is in S.

—

Step 1: Understand the Symmetry Conditions

A set of points that is symmetric about the **origin** means that for every point (x, y) in S, $(-x, -y)$ is also in S.
Symmetry about the **coordinate axes** means that if (x, y) is in S, then (y, x) is also in S.
Symmetry about the **line $y = x$** means that if (x, y) is in S, then (y, x) is also in S.

...

Is there a point that we might be missing?
Let's check if $(3, 2)$ is symmetric to itself. Yes, it is.
So, with these three points, we have covered all the symmetries.
But wait! What about the point $(2, 3)$? Is there a point symmetric to it that we haven't considered?
No, because we already considered all the points under the given symmetries.
So, the minimal number of points in S is 3.

—

Final Answer:

$$\boxed{3}$$

Table 12: Cluster 0 examples. Only 2 examples are shown for illustration.

Example 1

Prompt

(In the preliminaries of optimal method and experimental design) When using the 0.618 method to find the optimal amount to add in an experiment, if the current range of excellence is $[628, 774]$ and the good point is 718, then the value of the addition point for the current experiment is ______. Please reason step by step, and put your final answer within `\boxed{}`.

Answer

684

Example 2

Prompt

Calculate the probability that in a deck of 52 cards, the second card has a different suit than the first, and the third card has a different suit than the first and second. Please reason step by step, and put your final answer within `\boxed{}`.

Answer

$\frac{169}{425}$

Table 13: Cluster 1 examples. Only 2 examples are shown for illustration.

Example 1 **Prompt** Calculate: $\frac{\cos 190^\circ(1+\sqrt{3}\tan 10^\circ)}{\sin 290^\circ\sqrt{1-\cos 40^\circ}} =$ ______. Please reason step by step, and put your final answer within `\boxed{}`. **Answer** $2\sqrt{2}$
Example 2 **Prompt** Let a_n be the number of n-digit numbers formed using only digits 1 and 2 such that no two adjacent digits are both 2. Find a_5. Please reason step by step, and put your final answer within `\boxed{}`. **Answer** 13

Table 14: Cluster 2 examples. Only 2 examples are shown for illustration.

Example 1 **Prompt** You have 5 red balls and 5 blue balls in a box. You randomly draw 4 balls without replacement. What is the probability that exactly 2 red balls are drawn? Please reason step by step, and put your final answer within `\boxed{}`. **Answer** $\frac{25}{63}$
Example 2 **Prompt** If a and b are real numbers such that $a^2 + b^2 = 1$, what is the maximum value of ab? Please reason step by step, and put your final answer within `\boxed{}`. **Answer** $\frac{1}{2}$

Table 15: Cluster 3 examples. Only 2 examples are shown for illustration.

Example 1 **Prompt** Solve for x: $\log_3(x^2 - 1) = 2$. Please reason step by step, and put your final answer within `\boxed{}`. **Answer** 4
Example 2 **Prompt** Evaluate the integral $\int_0^1 xe^x\, dx$. Please reason step by step, and put your final answer within `\boxed{}`. **Answer** $e - 2$

Table 16: Cluster 4 examples. Only 2 examples are shown for illustration.

Example 1 **Prompt** If $\sin x + \cos x = \sqrt{2}$, find the value of $\sin^4 x + \cos^4 x$. Please reason step by step, and put your final answer within `\boxed{}`. **Answer** $\frac{3}{4}$
Example 2 **Prompt** Find the sum of the series $\sum_{n=1}^{\infty} \frac{1}{n(n+1)}$. Please reason step by step, and put your final answer within `\boxed{}`. **Answer** 1

Table 17: Cluster 5 examples. Only 2 examples are shown for illustration.

Example 1 **Prompt** How many 4-digit numbers are there such that no two adjacent digits are the same? Please reason step by step, and put your final answer within `\boxed{}`. **Answer** 5832
Example 2 **Prompt** If $A = \{1, 2, 3, 4\}$ and $B = \{3, 4, 5, 6\}$, what is $A \cup B$? Please reason step by step, and put your final answer within `\boxed{}`. **Answer** $\{1, 2, 3, 4, 5, 6\}$

Table 18: Cluster 6 examples. Only 2 examples are shown for illustration.

Example 1 **Prompt** What is the value of the determinant of the matrix $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$? Please reason step by step, and put your final answer within `\boxed{}`. **Answer** -2
Example 2 **Prompt** Simplify: $(2x - 3)^2 - (x + 1)^2$. Please reason step by step, and put your final answer within `\boxed{}`. **Answer** $3x^2 - 14x + 8$