Progressive Mastery: Customized Curriculum Learning with Guided Prompting for Mathematical Reasoning

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have achieved remarkable performance across various reasoning tasks, yet post-training is constrained by inefficient sample utilization and inflexible difficulty samples processing. To address these limitations, we propose Customized Curriculum Learning (CCL), a novel framework with two key innovations. First, we introduce model-adaptive difficulty definition that customizes curriculum datasets based on each model's individual capabilities rather than using predefined difficulty metrics. Second, we develop "Guided Prompting," which dynamically reduces sample difficulty through strategic hints, enabling effective utilization of challenging samples that would otherwise degrade performance. Comprehensive experiments on supervised fine-tuning and reinforcement learning demonstrate that CCL significantly outperforms uniform training approaches across five mathematical reasoning benchmarks, confirming its effectiveness across both paradigms in enhancing sample utilization and model performance.

1 Introduction

Large Language Models (LLMs) have achieved breakthrough progress in the field of natural language processing (NLP) in recent years. With additional post-training optimization, these models have demonstrated exceptional performance on complex tasks such as code generation and mathematical reasoning(Guo et al., 2024; Shao et al., 2024; Yang et al., 2025; Kavukcuoglu, 2025; OpenAI, 2024). However, the current post-training process for LLMs still faces significant challenges. One notable limitation of conventional training is the uniform treatment of all examples, ignoring their varying difficulty levels or value. Consequently, challenging or high-quality samples are not strategically introduced at optimal points in the training process, thereby impeding effective knowledge acquisition and integration.

To address this limitation, Bengio et al. (2009) introduced the concept of curriculum learning to model training, inspired by human education's progression from simple to complex concepts. Several recent studies have also begun to explore the application of curriculum learning strategies based on heuristic rules to the LLM post-training pipeline. For instance, in logical reasoning task, Xie et al. (2025) measured example difficulty by input length to implement progressive training from simpler to more complex instances. Wen et al. (2025b) classified difficult examples as those incorrectly predicted by DeepSeek-R1(DeepSeek-AI et al., 2025),and prioritized these challenging cases during later training stages. Although these rule-based curriculum learning methods improved performance to some extent, they nonetheless present certain limitations.

First, these predefined difficulty metrics lack precision in measuring actual difficulty levels. As illustrated in Figure 2, our experiments on the MATH dataset demonstrate that model performance does not consistently decline with increasing predefined difficulty levels. Counterintuitively, models achieve higher accuracy on purportedly more difficulty Level 5 problems than on Level 4 problems. Second, defining difficulty using a uniform standard proves inadequate, as metrics that effectively gauge difficulty for one model often fail to appropriately characterize challenge levels for another model. As illustrated in Figure 3, samples that present significant challenges for Qwen2.5-MATH-7B are often solved with ease by DeepSeek-Math-7B-Instruct, and vice versa. To address these limitations, we propose a tailored curriculum learning approach that calibrates sample difficulty according to each model's individual capabilities, thereby customizing more appropriate training sequences for different models.

Another significant challenge in model training stems from the presence of extremely challenging examples in the training data. Previous research (Yu et al., 2025; Wen et al., 2025a) has demonstrated that forcing models to train on examples substantially beyond their current capabilities can lead to performance degradation. Consequently, a conventional approach has been to simply exclude such overly difficult samples from the training process to prevent negative impacts on model learning. However, this wholesale elimination of challenging data is inherently inefficient, as these difficult examples often contain valuable information that could potentially enhance model performance if leveraged appropriately. To overcome this limitation, we introduce "Guided Prompting," a technique that augments input examples with targeted hints to dynamically modulate their difficulty during the training process. This method effectively prevents performance deterioration while enabling the model to extract meaningful patterns from examples that would otherwise be discarded, thereby significantly improving overall data utilization efficiency.

The contributions of this study are summarized as follows:

- We tailored the course dataset based on the model's performance and proposed a novel post-training approach called Customized Curriculum Learning (CCL).
- We implement "Guided Prompting" for samples that significantly exceed current model capabilities, effectively controlling sample difficulty and substantially improving data utilization efficiency.
- We conduct comprehensive experiments across two mainstream post-training paradigms, namely supervised fine tuning and reinforcement learning, demonstrating significant performance improvements on our specific model.

2 Related Work

Curriculum Learning. Bengio et al. (2009) introduces the concept of curriculum learning, demonstrating that models learn more effectively when training examples are presented in a progressively harder order. Recent approaches in large language models build on this concept. Xie et al. (2025) designs curricula by adjusting task difficulty based on logical complexity, enhancing the model's reasoning abilities. Wen et al. (2025b) treats queries that DeepSeek-R1(DeepSeek-AI et al., 2025) struggles with as hard samples, deferring them to later training stages for focused learning. Team et al. (2025) refine training by filtering out noisy samples early, concentrating on high-quality examples for later stages. Huang et al. (2025) apply curriculum learning to retrieval-augmented generation (RAG), ordering tasks based on the number of distractors in retrieved passages. Shi et al. (2025) dynamically selects training samples whose predefined difficulty scores are closest to the model's current target difficulty level, which is adjusted during training based on reward feedback. Unlike these approaches that rely on heuristic rules to define a fixed difficulty hierarchy shared across all models, we propose a customized curriculum learning framework that tailors the training sequence to each model's reasoning ability, enabling a more adaptive and effective learning process.

Model-Adaptive Difficulty Awaring. In the model training process, different types of samples should be treated with varying degrees of emphasis, with difficulty-aware methods serving as a key approach to distinguishing data types. Team (2025); Xie et al. (2024); Lee and Song (2024) employ LLM-generated scoring to assess sample difficulty, while Wen et al. (2025b); Min et al. (2024); Yuan et al. (2025) heuristically treat long-form QA tasks as inherently challenging. Tong et al. (2024); Xue et al. (2025) take a more empirical approach by conducting multiple sampling iterations for each query, defining difficulty through incorrect response ratios and allocating more trials to challenging queries during synthesis. Similarly, Ma et al. (2024) implements multi-round query sampling but weights samples inversely proportional to accuracy, thereby giving higher weights to samples with lower accuracy scores.

Post Training. Supervised fine-tuning and reinforcement learning represent the two most prevalent methods in post-training. Fine-tuning pretrained models on high-quality datasets with step-by-step solutions markedly enhances problem-solving accuracy(Yue et al., 2023; Yuan et al., 2023; Hwang et al., 2024). Beyond supervised learning, reinforcement-based fine-tuning has been explored to further align LLMs with solution correctness and preferred reasoning styles. Luo et al. (2023); Luong et al. (2024); Yue et al. (2025) optimize policy networks using Proximal Policy Optimization (PPO). In contrast, Shao et al. (2024); DeepSeek-

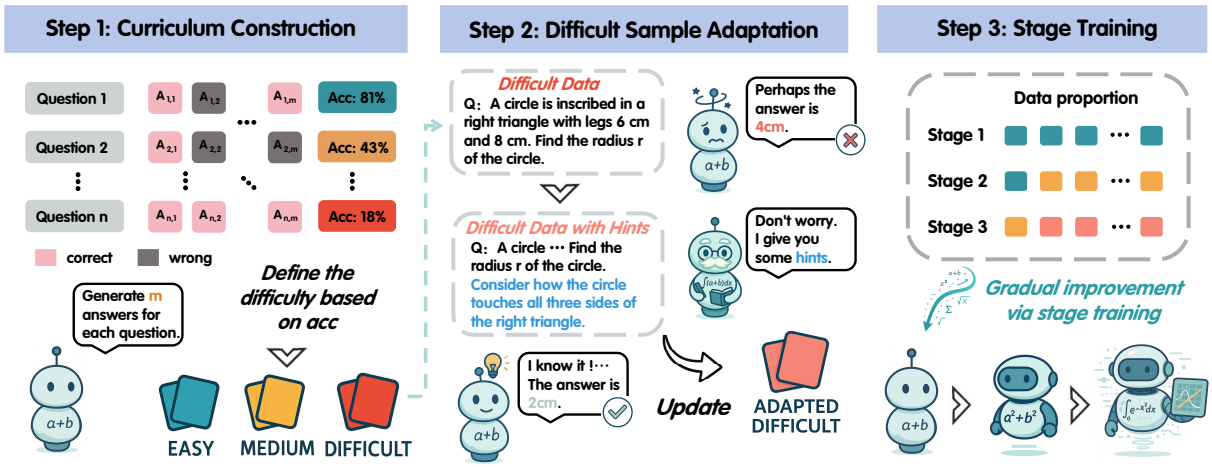

Figure 1: **Overall pipeline of our method. Step 1:** For each question in the training set, the model generates multiple responses and calculates the accuracy on that sample. Based on these accuracy scores, samples are ranked and organized into curriculum datasets. **Step 2:** Transforming difficult samples to reduce the answering difficulty for the model, bringing samples within the model's solvable range. **Step 3:** The model undergoes staged training sequentially on easy, medium, and difficult curriculum datasets, with its performance continuously enhanced.

AI et al. (2025); Yu et al. (2025) replace the critic model in PPO and optimize policy networks via Group Relative Policy Optimization (GRPO).

3 Method

Traditional training approaches treat all samples equally, failing to adequately leverage high-quality samples, which leads to suboptimal performance. To address this limitation, we propose the Customized Curriculum Learning (CCL) training framework, which enables models to learn progressively from easy to difficult samples. By structuring the training process to prioritize foundational concepts before advancing to more complex samples, CCL enables models to build extremely solid foundations during the early stages of training, upon which more sophisticated understanding can be constructed.

The CCL training framework consists of three key steps. First is curriculum construction. Since curriculum learning proceeds in stages across samples of varying difficulty, the original dataset must be partitioned accordingly. We propose an aptitude-based training approach that customizes curriculum for each model according to its inherent capabilities, applying the principle of individualized instruction to optimize learning progression. Second is difficult sample adaptation. Due to inherent model limitations, some samples remain consistently challenging regardless of the model's attempts. Training on such data can actually degrade model performance. We identify these problematic samples and implement a "Guided Prompting" method to reduce the answering difficulty, thereby improving sample utilization efficiency. The final step is multi-stage training. Utilizing the constructed and modified curriculum datasets from previous steps, we implement staged supervised fine-tuning and reinforcement learning, enabling the model to gradually adapt to samples of increasing difficulty, enhancing training stability and overall performance, the full algorithm is detailed in Algorithm 1.

3.1 Curriculum Construction

To implement training in an easy-to-difficult sequence, we first need to establish metrics for measuring sample difficulty, which will then be used to sort and segment the dataset. Ding et al. (2024) directly used the manually annotated difficulty levels in the MATH dataset as the standard for distinguishing sample complexity. However, this heuristic definition has certain limitations. As shown in Figure 2, we selected Qwen2.5-Math-1.5B(Yang et al., 2024), Qwen2.5-Math-7B, and Deepseek-Math-7B(Shao et al., 2024) as baseline models and tested them on the MATH dataset. The results indicate that model performance does not consistently decrease as the predefined difficulty level increases (for instance, models achieve higher accuracy on level 4 problems than on level 5 problems), suggesting that this heuristic definition lacks precision. Furthermore, Figure 3 demonstrates that defining difficulty using a uniform standard proves

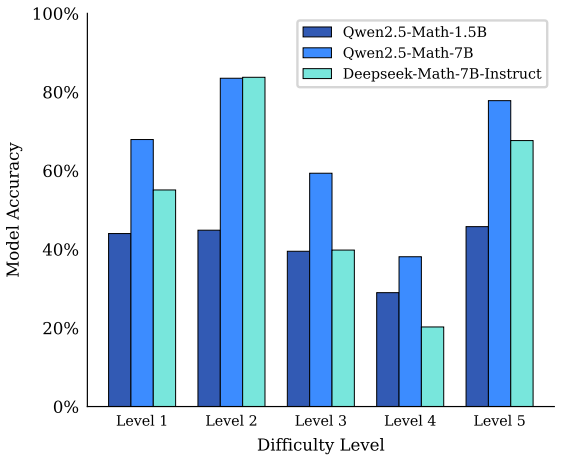

Figure 2: Performance of multiple models on MATH dataset subsets with predefined difficulty levels. As predefined difficulty increases from Level 1 to Level 5, model accuracy does not consistently decline but instead exhibits significant fluctuations, demonstrating that predefined difficulty standards may not correctly adapt to all models.

inadequate. Samples that are extremely simple for Qwen2.5-Math-1.5B may still challenge Qwen2.5-Math-7B, and vice versa, indicating that we cannot establish a single unified difficulty measurement standard that is suitable for all models.

To address these two issues, we propose a "teaching according to aptitude" curriculum construction approach, which customizes curriculum datasets based on each model's own performance. Specifically, we input all training samples into the model for inference and collect the model's responses through sampling. For each problem, the model provides N responses, and the model's accuracy rate for that sample is defined as following:

$$ACC_i = \frac{\sum_{j=1}^{n} \mathbf{1}\{A_{ij} = A_i^*\}}{n} \tag{1}$$

where ACC_i denotes the accuracy of i-th samples A_{ij} denotes the response generated by the model for the i-th sample in the j-th sampling iteration, and A_i^* denotes the golden answer of i-th samples. We define the difficulty level of a sample as the inverse of this accuracy rate and sort the samples according to their difficulty levels, as shown in Figure1 Step 1. Samples that the model can correctly answer multiple times are classified as simple samples and can be well mastered in the early stages of training. Samples that the model repeatedly fails to solve are classified as moderate or difficult samples, which are reserved for later stages of training after the model has acquired the relevant foundational knowledge in the domain, allowing it to more thoroughly learn these more complex samples.

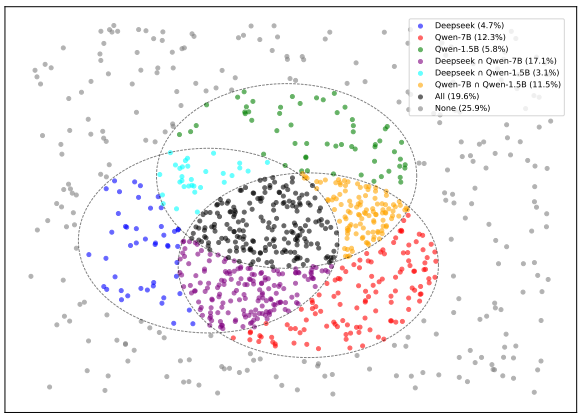

Figure 3: Visualization of solution correctness patterns for three mathematical reasoning models: Deepseek-Math-7B-Instruct (blue), Qwen2.5-Math-7B (red), and Qwen2.5-Math-1.5B (green). Each colored region represents a specific answering scenario; for example, the black region indicates questions all three models answered correctly. Approximately 55% of questions that are easy for one model prove difficult for another, demonstrating that using a unified standard to define sample difficulty across all models is unreasonable.

3.2 Difficult Sample Adaptation

Even after constructing curriculum data from easy to difficult as outlined in step 1, there remain samples that the model cannot solve correctly regardless of how many attempts it makes, due to limitations in the model's inherent capabilities. Yu et al. (2025); Wen et al. (2025a) have demonstrated that training on samples far beyond the model's current capabilities actually degrades performance, leading to the common practice of discarding such samples. However, we consider direct disposal of this data to be wasteful and propose a "guided prompting" method to transform and reclaim these difficult samples.

Similar to how teachers approach difficult concepts in education, we apply a pedagogical framework to model training. When students encounter obstacles while learning new material, effective teachers don't skip challenging concepts simply because they exceed the student's current understanding. Instead, they employ progressive, step-by-step guidance to facilitate knowledge absorption. By incorporating this educational philosophy into model training, we provide the model with hints that guide its solution generation process(Xi et al., 2024; Dou et al., 2025). This significantly reduces the difficulty of problem-solving and helps the model better

Algorithm 1 Customized Curriculum Learning for Enhancing Mathematical Reasoning

Require: training dataset $D = \{(Q_i, A_i)\}$ with questions Q_i and reference solution A_i
Require: pretrained model π_0, accuracy threshold τ, hint ratio α

Curriculum Construction:
for all question $Q_i \in D$ **do**
  Generate n response $\{A_{i1}, \ldots, A_{in}\}$ using model π_θ
  Calculate $ACC_i = \frac{\sum_{j=1}^{n} \mathbf{1}\{A_{ij}=A_i^*\}}{n}$, where A_i^* is the golden answer of Q_i
end for
Sort all samples in descending order based on their ACC_i values, then partition the dataset D into p disjoint subsets $\{D_1, D_2, ..., D_p\}$, where the p-th subset D_p contains samples with the lowest ACC_i values, representing the most challenging instances.
Difficult Sample Adaptive Processing:
for all difficult samples $(Q_i, A_i) \in D_p$ **do**
  Decompose the reference solution S_i into a sequence of problem-solving steps $\{s_{i1}, \ldots, s_{ik}\}$
  Gradually provide hints $P_i = \{s_{i1}, \ldots, s_{il}\}$ until either the ratio $\frac{|il|}{|ik|}$ reaches α, or the model's performance improves to the τ
  if model's performance improves to τ **then**
    Update question and answer: $Q_i \rightarrow [Q_i; P_i]$, $A_i \rightarrow \{s_{i(l+1)}, \ldots, s_{ik}\}$
  else
    Discard this overly difficult sample (Q_i, A_i) that the model still fails to handle effectively even with hints
  end if
end for
Multi-Stage Training Process:
for each stage $s \in \{1, \ldots, p\}$ **do**
  Fine-tune the previous stage's model π_{s-1} on current stage dataset D_s:

$$\pi_s = \arg \min_{(Q,A)\in D_s} \mathcal{L}(\pi_{s-1})$$

end for
Output: Enhanced model π_m with improved problem-solving skills

assimilate the current knowledge, as illustrated in Figure 1 Step 2.

Specifically, for a problem Q_i with corresponding reference answer S_i, we first decompose S_i into step-by-step solution components, such that $S_i = \{s_{i1}, s_{i2}, ..., s_{ik}\}$. We then extract a small prefix $P_i = \{s_{i1}, s_{i2}, ..., s_{ip}\}$ from S_i to serve as a hint, where p is smaller then k. This prefix P_i is concatenated with Q_i as input to guide the model toward generating the correct answer, that is

$$y_i \sim \pi_\theta(Y|[Q_i; P_i]) \tag{2}$$

where y_i denotes response generated by model θ. Through this approach, samples that previously exceeded the model's capabilities are transformed into manageable examples. The originally complex answer generation task becomes a simpler answer completion task, thereby enhancing the model's effective utilization of training samples.

3.3 Multi Stage Training

After constructing the curriculum dataset and modifying the difficult samples, we can utilize this partitioned data $D = \{D_1, D_2, ..., D_p\}$ for multi stage SFT and multi stage RL, as shown at Figure 1 Step 3. For model π_θ, dataset D_j has a higher difficulty level than dataset D_i, where j>i.

3.3.1 Multi Stage SFT

Let the pre-trained model be denoted as π_0. Based on the number of partitions in the curriculum dataset, multi-stage SFT necessitates m rounds of maximum likelihood optimization. The loss function for the i-th round of optimization can be formally expressed as follows:

$$\mathcal{L}_{SFT} = -\mathbb{E}_{(x,y)\sim D_i}[log(\pi_{i-1}(y|x))] \tag{3}$$

3.3.2 Multi Stage RL

Based on the constructed curriculum dataset, we introduce multi-stage reinforcement learning to enhance the model's generalizability. DeepSeek-AI et al. (2025) has demonstrated that reinforcement learning can be performed directly without supervised fine-tuning, known as DeepSeek-R1-Zero, which can also achieve excellent performance on reasoning tasks. We follow this setting to conduct our experiments.

The reward is the source of training signals in reinforcement learning and determines the optimization direction of the entire reinforcement learning process. To effectively train reasoning models through reinforcement learning, following the setup in (DeepSeek-AI et al., 2025), we adopt a rule-based reward function that includes two types of rewards. The first is the format reward, which measures whether the model outputs according to our required format. If the response contains special tokens such as "<think>,</think>,<answer>,</answer>", it is considered that the format is correct, formally represented as follows:

$$r_{format} = \begin{cases} 1.0 & \text{format is correct} \\ 0.0 & \text{others} \end{cases} \tag{4}$$

The second type is the accuracy reward, which measures whether the model's final prediction is correct. We extract the model's prediction from the generated response according to the rules and compare it with the golden answer, formally represented as follows:

$$r_{accuracy} = \begin{cases} 1.0 & \text{prediction is correct} \\ 0.0 & \text{others} \end{cases} \tag{5}$$

The final reward r equals the sum of both, that is $r = r_{format} + r_{accuracy}$.

We employ Group Relative Policy Optimization (GRPO) as our policy learning algorithm(Shao et al., 2024). The GRPO algorithm generates multiple candidate responses O for each question Q, where $O = \{o_1, o_2, ..., o_G\}$. These different responses to the same question form a group, and the reward of each response in this group is used to calculate the advantage A_i of that sample, as follows:

$$A_i = \frac{r_i - maen(\{r_1, r_2, ..., r_G\})}{std(\{r_1, r_2, ..., r_G\})} \tag{6}$$

GRPO adopts a clipped objective, together with a directly imposed KL penalty term:

$$\mathcal{L}_{GRPO} = \mathbb{E}_{(x)\sim D}[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}(min(r_{i,t}(\theta)A_i, clip(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)A_i) - \beta D_{KL}(\pi_\theta||\pi_{ref}))], \tag{7}$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})} \tag{8}$$

4 Experiments

4.1 Datasets

Train. Following the experimental setting of (Zeng et al., 2025), we selected the MATH dataset(Hendrycks et al., 2021) and extracted samples from level 3 to level 5 as training data, comprising a total of $9,255$ instances. To adapt our proposed CCL framework for creating a customized curriculum dataset for the model, we first needed to differentiate these samples based on their difficulty levels according to the model's performance on them. After completing inference on all samples, we ranked them according to the model's accuracy rate. Samples with higher accuracy rates were categorized as simple data for use in the early stages of model training, while samples with lower accuracy rates were designated as difficult data for later training stages. See Appendix A.1 for more detailed descriptions. Additionally, the data was processed into a conversational format. The prompts used in the SFT and GRPO processes can be found at Appendix C.

Test. We use five benchmark datasets to assess the model's performance across different levels of difficulty and mathematical reasoning. MATH 500(Lightman et al., 2023), is a subset of the MATH dataset, containing 500 representative problems designed to test a model's general mathematical capability. OlympiadBench (He et al., 2024) includes a collection of problems from Olympiad-level mathematics and physics competitions. Minerva Math (Lewkowycz et al., 2022) is a curated set of undergraduate-level math problems that assess complex mathematical reasoning and symbolic manipulation. AMC 23 and AIME 24 include problems from the 2023 American Mathematics Competitions and the 2024 American Invitational Mathe-

Model	Method	Learning Strategy	MATH 500	Minerva Math	Olympiad Bench	AIME24	AMC23	Average
Qwen2.5-Math-1.5B	SFT	*Uniform*	**48.60**	18.00	12.60	0.00	27.50	21.34
		CCL	48.00	**23.50**	**12.90**	0.00	27.50	**22.38**
	GRPO	*Uniform*	51.80	18.40	21.00	10.00	22.50	24.74
		CCL	**72.60**	**31.60**	**32.70**	**13.30**	**42.50**	**38.54**
Qwen2.5-Math-7B	SFT	*Uniform*	**68.80**	16.50	18.40	0.00	22.50	25.24
		CCL	63.00	**21.00**	**18.70**	**3.30**	**45.00**	**30.20**
	GRPO	*Uniform*	74.20	33.50	33.90	10.00	**62.50**	42.82
		CCL	**76.60**	**38.20**	**38.20**	**13.30**	60.00	**45.26**

Table 1: Evaluation results of different learning strategies on Math Datasets

matics Examination, respectively. Additionally, the data was processed into a conversational format.

4.2 Models

To effectively validate the efficacy of our CCL method across foundation models of varying capabilities, we selected two different-sized models for our experiments: Qwen2.5-MATH-1.5B(Yang et al., 2024) and Qwen2.5-MATH-7B.

4.3 Training Setup

We conducted our experiments using 8 NVIDIA A100 GPUs for the SFT experiments within the Llama-Factory framework(Zheng et al., 2024) and for the GRPO experiments within Hugging Face's Open R1 framework(Face, 2025). See Appendix A.2 for more detailed descriptions.

4.4 Evaluation Setup

We evaluated our models using the evaluation script from (Zeng et al., 2025). See Appendix B for more detailed descriptions.

4.5 Main Results

We implemented various strategies and training methods across multiple models of different scales and conducted extensive experiments on test sets of varying difficulty levels. As shown in Table 1, compared to uniform training that treats all data equally, our CCL strategy demonstrates significant advantages by customizing curriculum datasets and adopting an easy-to-hard training approach, yielding substantial performance improvements across multiple experimental settings.

Our CCL learning strategy demonstrates both method compatibility and model scalability. When applied to different-sized models under SFT settings, CCL improved performance by 1.04% and 4.96% for Qwen2.5-Math-1.5B and Qwen2.5-Math-7B respectively. Under GRPO settings, the improvements were 13.80% and 2.44% for the same models. Notably, our CCL training strategy yielded consistent performance gains across all test subsets, demonstrating its significant enhancement of model generalization capabilities. Appendix D also presents the overall performance changes of the CCL method on the test set during the multi-stage training process.

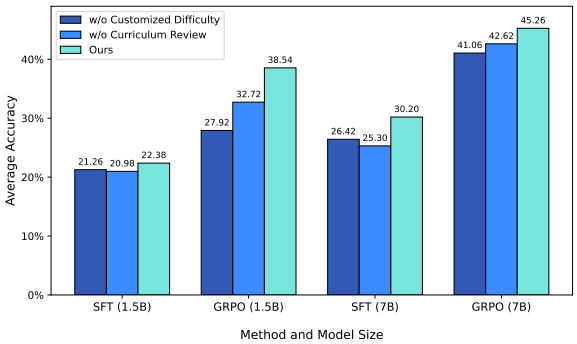

Figure 4: Comparison of model performance after training with data partitioned using different sample difficulty definition methods and across various data mixing strategies.

4.6 Ablation Study

To evaluate the effectiveness of our method's components, we perform ablation studies on three key aspects: sample difficulty definition, difficult sample processing, and data mixing strategies. These studies isolate each design choice's contribution to model performance and training stability.

4.6.1 Sample Difficulty Definition

Previous researches have relied on predefined difficulty labels from the MATH dataset to construct curriculum learning data. However, we propose that a more effective approach is to customize difficulty labels based on each model's actual performance on samples before constructing curriculum datasets. We conducted comparative experiments using both approaches to partition training data.

As shown in Figure 4, our customized difficulty definition yields superior performance across models of varying sizes and different post-training methods, demonstrating consistent advantages over the predefined difficulty categorization. This empirically validates that difficulty labels tailored to specific model capabilities lead to more effective curriculum learning than predefined difficulty metrics.

Model Size	Processing Strategy	Avg.
1.5B	Retaining Difficult Samples	26.36
	Discarding Difficult Samples	37.46
	Adapting Difficult Samples	**38.54**
7B	Retaining Difficult Samples	41.34
	Discarding Difficult Samples	44.86
	Adapting Difficult Samples	**45.26**

Table 2: Comparison of model performance across three difficult sample processing methods

4.6.2 Difficult Sample Processing

At this part, we conduct an ablation study to test whether continuous training on excessively difficult samples degrades model performance. We compare three difficult sample processing methods using GRPO: (1) retaining difficult samples, (2) discarding difficult samples, and (3) applying our "Guided Prompting" approach to adapt difficult samples with strategic hints. Results in Table 2 confirm that learning from overly challenging samples indeed harms model performance. Discarding difficult samples outperforms retaining them, validating this hypothesis. This effect is more severe for weaker models like Qwen2.5-Math-1.5B, which encounter more unsolvable samples and suffer greater performance degradation.

Rather than waste valuable data through direct removal, our "Guided Prompting" method repurposes difficult samples by providing hints that bring problems within the model's solvable range. Experimental results demonstrate this approach successfully recovers difficult samples while substantially improving model performance, establishing it as the optimal strategy for enhancing mathematical reasoning capabilities.

4.6.3 Data Mixing Strategy

Just as students need to periodically review previously mastered knowledge during their learning process, we believe that models undergoing staged curriculum learning require similar reinforcement of previously acquired content. Therefore, we design two distinct data mixing strategies for comparative analysis. The first approach, termed "Naive Curriculum," provides models with samples corresponding only to the current difficulty level at each training stage. The second approach, called "Curriculum Review," incorporates a small proportion of easier samples during later training stages, allowing the model to revisit previously learned material.

Experimental results in Figure 4 demonstrate that Curriculum Review data mixing strategy achieves superior performance, confirming that providing models with previously learned content during later training stages is crucial for preventing catastrophic forgetting. This finding underscores the importance of maintaining access to foundational knowledge throughout the curriculum learning process.

5 Conclusions

In conclusion, we presented Customized Curriculum Learning (CCL), a novel post-training framework that systematically constructs model-adaptive curriculum sequences and transforms difficult samples through guided prompting to enhance large language models' mathematical reasoning capabilities. Our comprehensive experiments demonstrated that models trained with CCL significantly outperform those using uniform training approaches across multiple mathematical reasoning benchmarks, with consistent improvements observed in both supervised fine-tuning and reinforcement learning paradigms. By effectively integrating curriculum learning into large language model training through model-specific difficulty customization and guided prompting, our work substantially improves sample utilization and model performance, advancing more effective training methodologies for large-scale language models.

Limitations

Despite the promising results of our work, several limitations warrant acknowledgment. While our study focuses on mathematical reasoning, we see great potential in extending the CCL framework to other domains such as logical reasoning, code generation, and natural language inference, allowing us to further investigate its generalizability across diverse task types. Furthermore, although our current study applies CCL within specific post-training paradigms, such as supervised fine-tuning and GRPO, we recognize that combining CCL with other post-training strategies—like PPO and broader reinforcement learning techniques—remains an open direction. Exploring these combinations may further reveal the full potential of the CCL framework in enhancing model learning dynamics.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *International Conference on Machine Learning*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiaoling Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 179 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv*, abs/2501.12948.

Yiwen Ding, Zhiheng Xi, Wei He, Zhuoyuan Li, Yitao Zhai, Xiaowei Shi, Xunliang Cai, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Mitigating tail narrowing in llm self-improvement via socratic-guided sampling. *ArXiv*, abs/2411.00750.

Shihan Dou, Muling Wu, Jingwen Xu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2025. Improving rl exploration for llm reasoning through retrospective replay.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. Deepseek-coder: When the large language model meets programming - the rise of code intelligence. *ArXiv*, abs/2401.14196.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Annual Meeting of the Association for Computational Linguistics*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Xiaodong Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *ArXiv*, abs/2103.03874.

Jerry Huang, Siddarth Madala, Risham Sidhu, Cheng Niu, J. Hockenmaier, and Tong Zhang. 2025. Rag-rl: Advancing retrieval-augmented generation via rl and curriculum learning. *ArXiv*, abs/2503.12759.

Hyeonbin Hwang, Doyoung Kim, Seungone Kim, Seonghyeon Ye, and Minjoon Seo. 2024. Self-explore: Enhancing mathematical reasoning in language models with fine-grained rewards. In *Conference on Empirical Methods in Natural Language Processing*.

Koray Kavukcuoglu. 2025. Gemini 2.5: Our most intelligent ai model. `https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/`. Accessed: 2025-05-17.

Jung X. Lee and Yeong-Tae Song. 2024. College exam grader using llm ai models. In *2024 IEEE/ACIS 27th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 282–289.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. *ArXiv*, abs/2206.14858.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *ArXiv*, abs/2305.20050.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jian-Guang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *ArXiv*, abs/2308.09583.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. *ArXiv*, abs/2401.08967.

Jingyuan Ma, Rui Li, Zheng Li, Lei Sha, and Zhifang Sui. 2024. Plug-and-play training framework for preference optimization. *ArXiv*, abs/2412.20996.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, and Jiahui Wen. 2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *ArXiv*, abs/2412.09413.

OpenAI. 2024. Openai o1 system card. https://arxiv.org/abs/2412.16720. Accessed: 2025-05-17.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junmei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/2402.03300.

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

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Feng Tang, Flood Sung, Guangda Wei, Guokun Lai, and 75 others. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *ArXiv*, abs/2501.12599.

NovaSky Team. 2025. Sky-t1: Train your own o1 preview model within 450. $https://novasky-ai.github.io/posts/sky-t1. Accessed: 2025-01-09$.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Rui Min Wu, and Junxian He. 2024. Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving. *ArXiv*, abs/2407.13690.

Cheng Wen, Tingwei Guo, Shuaijiang Zhao, Wei Zou, and Xiangang Li. 2025a. Sari: Structured audio reasoning via curriculum-guided reinforcement learning.

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. 2025b. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *ArXiv*, abs/2503.10460.

Zhiheng Xi, Wenxiang Chen, Boyang Hong, Senjie Jin, Rui Zheng, Wei He, Yiwen Ding, Shichun Liu, Xin Guo, Junzhe Wang, Honglin Guo, Wei Shen, Xiaoran Fan, Yuhao Zhou, Shihan Dou, Xiao Wang, Xinbo Zhang, Peng Sun, Tao Gui, and 2 others. 2024. Training large language models for reasoning through reverse curriculum reinforcement learning. *ArXiv*, abs/2402.05808.

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

Wenjing Xie, Juxin Niu, Chun Jason Xue, and Nan Guan. 2024. Grade like a human: Rethinking automated assessment with large language models. *ArXiv*, abs/2405.19694.

Boyang Xue, Qi Zhu, Hongru Wang, Rui Wang, Sheng Wang, Hongling Xu, Fei Mi, Yasheng Wang, Lifeng Shang, Qun Liu, and Kam-Fai Wong. 2025. Dast: Difficulty-aware self-training on large language models. *ArXiv*, abs/2503.09029.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *ArXiv*, abs/2409.12122.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, 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, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *ArXiv*, abs/2503.14476.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. 2025. Agent-r: Training language model agents to reflect via iterative self-training. *ArXiv*, abs/2501.11425.

Zheng Yuan, Hongyi Yuan, Cheng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *ArXiv*, abs/2308.01825.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *ArXiv*, abs/2309.05653.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, Tiantian Fan, Zhengyin Du, Xiang Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, and 8 others. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *Preprint*, arXiv:2503.18892.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of

100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

A Training Details

In this chapter, we will provide a detailed description of the construction of curriculum datasets in the multi-stage training process, while also presenting a comprehensive overview of the hyperparameters utilized during both training and testing procedures.

A.1 Constructing Dataset

In the process of constructing the curriculum dataset, we need to feed all training set samples into the pre-trained model for inference and evaluate the model's accuracy on each sample. To ensure that the evaluation results are as reliable as possible while not causing excessive computational overhead, for each question in the dataset, we use the VLLM framework to generate 16 responses from the model, extract predictions from these responses using appropriate scripts, and compare them with golden answers to determine the correctness of the generations. To fully harness the model's potential, we did not adopt a greedy decoding strategy to generate responses, but instead set the temperature to 0.7, generating responses through sampling.

After calculating the model's accuracy on the samples through the above steps, we sort the samples and divide them into 3 equal parts according to quantity. The top $1/3$ with the highest accuracy are classified as simple samples, used for the first stage of model training. The bottom $1/3$ with the lowest accuracy are classified as difficult samples, used for the final stage of model training.

In addition, for particularly challenging samples, we employed a "Guided Prompting" approach to reduce the difficulty for the model. Specifically, we first collected reference answers for these difficult samples, then segmented these reference answers into step-by-step reasoning processes, as illustrated in Figure 5. Finally, we selected a small portion of the prefix combined with the original question as input to assist the model in solving problems more effectively.

A.2 Training HyperParameters

SFT. We conducted our experiments using bf16 precision under the DeepSpeed framework with zero-2 configuration. We set per_device_train_batch_size to 1 and gradient_accumulation_steps to 4, employing a cosine lr_scheduler with warmup set to 0.1 and max length set to 2048. For the Qwen2.5-Math-1.5B model, we used a learning rate of $5e-6$ and trained for 3 epochs. For the Qwen2.5-Math-7B model, we used a learning rate of $1e-5$ and trained for 3 epochs.

GRPO. We conducted our experiments using bf16 precision under the DeepSpeed framework with zero-2 configuration. We set per_device_train_batch_size to 16 and gradient_accumulation_steps to 8, employing a cosine lr_scheduler with warmup set to 0.1 and beta to 0.04, num_generations to 7, max_prompt_lengtht to 512 and max_completion_length 1024. For the Qwen2.5-Math-1.5B model, we used a learning rate of $3e-6$ and trained for 6 epochs. For the Qwen2.5-Math-7B model, we used a learning rate of $3e-6$ and trained for 4 epochs.

B Evaluation Details

During the testing process, to ensure the stability of test results, all methods employed a greedy decoding strategy with top_p set to 0.95, and used "</answer>" as a stop word to truncate the generated content.

C Prompt Details

During both training and testing processes, the data was processed into a conversational format. Figure 7 and Figure 6 demonstrate the prompts we used during the SFT and GRPO processes respectively. After training the models using their respective methods, we employed the corresponding prompts during testing as well. Additionally, during the GRPO training process, besides adding the User's description, we also appended part of the Assistant's content prefixed with the special token "<think>". This approach helps the model quickly learn format compliance during the reinforcement learning process, greatly enhancing the stability of the model's reinforcement learning.

D Result Details

In this section, we demonstrate the overall performance changes on the test set when applying CCL to Qwen2.5-Math-1.5B and Qwen2.5-Math-7B using supervised fine-tuning and reinforcement learning methods for multi-stage training. As shown in Figure 8, our CCL method continuously improves in performance as training iterations progress.

Case Study

Problem
Let n be the smallest positive integer that satisfies the following conditions:
n divided by 2 is a perfect square
n divided by 3 is a perfect cube
n divided by 5 is a perfect fifth power
How many divisors of n are NOT multiples of 10?

Solution
The first condition implies that the power of each prime factor of n must be an even power (excluding 2, which must be an odd power). The second condition implies that the power of each prime factor of n must be divisible by 3 (excluding 3, which must leave a residue of 1 upon division by 3). The third condition implies that the power of each prime factor of n must be divisible by 5 (excluding 5, which must leave a residue of 1 upon division by 5).\nClearly, to minimize n, we want to just use the prime factors $2,3,5$. The power of 2 must be divisible by $3,5$, and 2^{15} works. Similarly, the powers of 3 and 5 must be 10 and 6, respectively, both of which leave a residue of 1 upon division. Thus, we need the number of factors of $2^{15} \\cdot 3^{10} \\cdot 5^{6}$ which are not multiples of 10.\nApplying the complement principle, there are a total of $(15+1)(10+1)(6+1) = 1232$ factors. We can draw a bijection between the number of divisors of $2^{15} \\cdot 3^{10} \\cdot 5^{6}$ that are divisible by 10 and the number of divisors of $2^{14} \\cdot 3^{10} \\cdot 5^{5}$ (as each of these divisors, when multiplied by 10, will provide a factor of the original number that is divisible by 10). There are $(14+1)(10+1)(5+1) = 990$. The answer is $1232-990 = \\boxed{242}$.

Step 1
The first condition implies that the power of each prime factor of n must be an even power (excluding 2, which must be an odd power).
Step 2
The second condition implies that the power of each prime factor of n must be divisible by 3 (excluding 3, which must leave a residue of 1 upon division by 3).
Step 3
The third condition implies that the power of each prime factor of n must be divisible by 5 (excluding 5, which must leave a residue of 1 upon division by 5).
Step 4
Clearly, to minimize n, we want to just use the prime factors $2,3,5$.
Step 5
The power of 2 must be divisible by $3,5$, and 2^{15} works.
Step 6
Similarly, the powers of 3 and 5 must be 10 and 6, respectively, both of which leave a residue of 1 upon division.
Step 7
Thus, we need the number of factors of $2^{15} \cdot 3^{10} \cdot 5^{6}$ which are not multiples of 10.
Step 8
Applying the complement principle, there are a total of $(15+1)(10+1)(6+1) = 1232$ factors.
Step 9
We can draw a bijection between the number of divisors of $2^{15} \cdot 3^{10} \cdot 5^{6}$ that are divisible by 10 and the number of divisors of $2^{14} \cdot 3^{10} \cdot 5^{5}$ (as each of these divisors, when multiplied by 10, will provide a factor of the original number that is divisible by 10).
Step 10
There are $(14+1)(10+1)(5+1) = 990$.
Step 11
The answer is $1232-990 = \boxed{242}$.

Figure 5: Decomposition of reference answers into step-by-step solution.

GRPO Prompt

You are a helpful AI Assistant that provides well-reasoned and detailed responses.
You first think about the reasoning process as an internal monologue and then provide the user with the answer.
Respond in the following format:
<think>
reasoning process here
</think>
<answer>
answer here
</answer>
User:
{Problem}
Assistant:
Let me solve this step by step.
<think>

Figure 6: Prompt Used in GRPO.

SFT Prompt

<|im_start|>system
You are a helpful assistant.
<|im_end|>
<|im_start|>user
{Problem}
Please reason step by step, and put your final answer within boxed{{}}.
<|im_end|>
<|im_start|>assistant

Figure 7: Prompt Used in SFT.

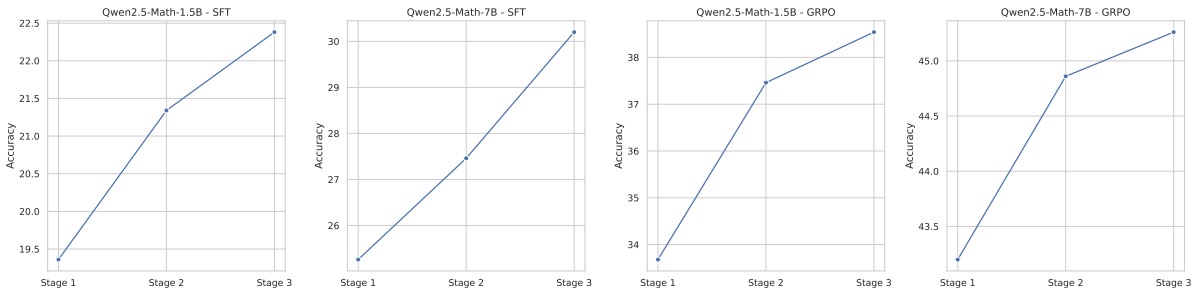

Figure 8: Performance Across Training Stages Using CCL.