Extended Abstract

Extensions Curriculum Learning, Synthetic Data, Incorporating Test Time Inference

Motivation While small language models (LLMs) like Qwen2.5-0.5B are efficient and easier to deploy, they often struggle with alignment and reasoning performance. Reinforcement learning (RL) methods such as Direct Preference Optimization (DPO) and REINFORCE Leave-One-Out (RLOO) offer potential improvements, but are unstable in low-resource settings. Motivated by human learning processes, we explore curriculum learning and test-time self-correction as strategies to improve learning stability and generalization in compact models.

Method We propose a curriculum-guided RL framework targeting two tasks: instruction following and math reasoning. For instruction following, we adopt a three-stage pipeline consisting of supervised fine-tuning (SFT), DPO, and curriculum learning based on prompt length. For math, we implement a staged training process including synthetic warm-starting, curriculum-based RLOO with arithmetic difficulty levels, and a test-time self-correction module that refines model outputs using its own prior errors.

Implementation We fine-tune Qwen2.5-0.5B using LoRA ($r = 8$ or 16, $\alpha = 32$) for both instruction following and math reasoning. For SFT, we use 10k SmolTalk samples; for DPO and Curriculum-DPO, 5k preference pairs from UltraFeedback are split into three stages based on prompt length: $<$256, 256–512, and $\geq$512 tokens. Math reasoning uses a 3-stage pipeline: a 3.2k-sample warm-start via GPT-4o distilled examples, a 100k-sample synthetic curriculum of arithmetic tasks, and a 10k-sample self-correction dataset derived from model-generated failures. Training uses batch sizes of 4–8, gradient accumulation up to 256, and learning rates of 3e-5 (SFT), 5e-5 (DPO), and 2e-7 (RLOO). Inference adopts temperature 0.4, top-p 0.95, and max 256 tokens. Evaluation relies on win rate using Nemotron 70B and success rate on a held-out math benchmark.

Results Instruction following shows consistent improvement: SFT reaches 0.63 win rate, DPO improves to 0.71, and Curriculum-DPO further to 0.75, despite a rise in evaluation loss—indicating better alignment not reflected by raw loss. Training curves confirm that curriculum staging smooths convergence and mitigates overfitting. In math reasoning, accuracy progresses from 14.9% (SFT) to 22.1% (Curriculum+SFT), 36.6% (Curriculum+RLOO), and peaks at 85.6% with self-correction. Iterative revision proves highly effective, especially in early correction rounds, but incurs test-time latency and reduces single-pass performance due to over-specialization. These results confirm that structured supervision and post-hoc refinement significantly enhance alignment and reasoning in small-scale models.

Discussion Our results highlight that curriculum learning and test-time self-correction are highly effective in improving small-model alignment and reasoning. Structuring DPO and RLOO with staged difficulty mitigates optimization instability and helps models generalize beyond imitation. In particular, the self-correction phase yields large performance gains by enabling iterative refinement, but introduces latency and reduces single-pass accuracy due to its specialization in revision rather than direct solving. Additionally, our approach depends on task-specific heuristics (e.g., prompt length, operator complexity), which may not transfer across domains. Nonetheless, the success of these strategies in low-resource regimes suggests that principled supervision and lightweight inference are strong alternatives to brute-force scaling.

Conclusion This work demonstrates that combining curriculum-guided RLHF with test-time revision significantly enhances the capabilities of compact models like Qwen2.5-0.5B. Without increasing model size or relying on extensive human annotation, we achieve substantial alignment and reasoning improvements through structured training and targeted inference augmentation. Our results suggest that thoughtful curriculum design and iterative feedback can close much of the gap between small and large LLMs, making high-quality alignment more accessible and efficient.

Fine-Tuning Qwen-0.5B for Math Reasoning and Instruction Following

Boyu Han
Department of Statistics
Stanford University
boyuhan@stanford.edu

Haoran Jia
Department of Statistics
Stanford University
haoranj@stanford.edu

Shuchen Liu
Department of Management Science and Engineering
Stanford University
shuchen@stanford.edu

1 Abstract

We propose a curriculum-guided reinforcement learning framework to enhance the instruction-following and mathematical reasoning capabilities of the Qwen2.5-0.5B language model. For instruction following, we apply a three-stage pipeline combining supervised fine-tuning (SFT), Direct Preference Optimization (DPO), and curriculum learning based on prompt length. This structured training improves the model's win rate from 0.63 (SFT) to 0.75 with curriculum-guided DPO. For math reasoning, we adopt a multi-stage strategy incorporating SFT, curriculum learning on synthetic arithmetic problems, reinforcement learning with leave-one-out (RLOO), and test-time self-correction. This results in a substantial improvement in problem-solving accuracy from 14.9% to 85.6%. Our findings demonstrate that structured learning schedules and lightweight test-time strategies can significantly improve the alignment and reasoning abilities of small language models in low-resource settings.

2 Introduction

Large language models (LLMs) such as LLaMA Touvron et al. (2023) and Qwen Qwen et al. (2025) have demonstrated impressive performance across natural language processing tasks. While these models are pretrained via next-token prediction on large corpora, recent work has shown that additional alignment—especially through reinforcement learning from human feedback (RLHF)—is essential for producing helpful, honest, and instruction-following behavior. Techniques like Direct Preference Optimization (DPO) Rafailov et al. (2024) and REINFORCE Leave-One-Out (RLOO) Ahmadian et al. (2024) have emerged as scalable, reward-efficient alternatives to classical policy gradient methods.

However, aligning compact models such as Qwen2.5-0.5B remains challenging. Unlike their larger counterparts, small models suffer more from unstable optimization, noisy reward signals, and limited generalization capacity. Applying RLHF naively often leads to overfitting or poor convergence—particularly on complex tasks like multi-step reasoning and long-form instruction following. This raises a key research question: how can we improve RLHF training efficiency and alignment quality for small LLMs under limited computational and data budgets?

To address this, we explore two complementary strategies: curriculum learning and test-time inference. Curriculum learning Bengio et al. (2009); Narvekar et al. (2020) improves training stability by gradually increasing example difficulty, while inference-time self-consistency and verifier-guided

reranking improve response quality without requiring retraining. Specifically, we apply a three-stage curriculum schedule to both DPO and RLOO training, partitioning samples by prompt length or arithmetic complexity. For math reasoning, we further introduce synthetic training data and iterative self-correction at inference time to reduce hallucination and improve factual consistency.

Our contributions are as follows:

- **Curriculum-Guided Fine-Tuning:** A multi-stage training schedule for DPO and RLOO that exposes small models to increasingly difficult samples.
- **Difficulty-Aware Data Curation:** Rule-based and model-informed strategies to partition instruction and math datasets into structured difficulty tiers.
- **Test-Time Inference Strategies:** Self-consistency sampling and iterative revision to enhance math accuracy without further training.
- **Empirical Validation:** Experiments show that our approach improves win rate and reasoning accuracy over strong RLHF baselines.

Our findings suggest that combining curriculum learning with lightweight inference techniques enables compact models to achieve better alignment and reasoning capabilities, offering a practical path forward in resource-constrained settings.

3 Related Work

Recent advances in large language models (LLMs) such as GPT Brown et al. (2020), LLaMA Touvron et al. (2023), and Qwen Qwen et al. (2025) have enabled strong zero-shot performance across many tasks. Instruction-tuned variants like FLAN-T5 Wei et al. (2022) demonstrate that models benefit from diverse supervised objectives. However, most instruction-tuning efforts target medium or large-scale models; fine-tuning compact models like Qwen-0.5B remains underexplored, particularly due to capacity constraints and overfitting risks. Our work focuses on improving small-model alignment through structured reinforcement learning.

Reinforcement Learning from Human Feedback (RLHF) is a central paradigm for alignment. InstructGPT Ouyang et al. (2022) uses preference-labeled data and PPO to train helpful and honest models. More recent methods like Direct Preference Optimization (DPO) Rafailov et al. (2024) avoid reward modeling by directly optimizing over preferred responses, while REINFORCE Leave-One-Out (RLOO) Ahmadian et al. (2024) supports online reward shaping. Both methods improve training stability, but their use in low-capacity settings remains challenging due to noisy signals and sparse rewards.

Curriculum learning (CL), introduced by Bengio et al. Bengio et al. (2009), proposes exposing models to gradually harder examples. This idea has been widely applied in supervised learning Wei et al. (2022) and RL Narvekar et al. (2020). In language models, DeepSeek-R1 DeepSeek-AI et al. (2025) and Tülu 3 Lambert et al. (2025) show that difficulty-aware data filtering improves reasoning in large models. However, curriculum-based RLHF remains underexplored in small-model regimes, where optimization is more sensitive. Our work addresses this gap by combining DPO and RLOO with staged curricula based on prompt length and problem complexity.

Finally, test-time inference techniques offer a complementary path for improving model robustness. Self-consistency Wang et al. (2023) improves reasoning by sampling multiple completions and selecting the majority answer. Snell et al. Snell et al. (2024) show that increased inference-time sampling can outperform scaling model size under budget constraints. Generative verifiers Zhang et al. (2025) use learned models to score candidate outputs during decoding. Unlike these methods, which operate post-training, our approach focuses on enhancing training stability and sample efficiency—though we also integrate self-correction at inference to further improve math performance.

4 Method

4.1 Instruction Following

Our method consists of three key components: supervised fine-tuning (SFT) on SmolTalk, preference tuning using Direct Preference Optimization (DPO) on the UltraFeedback dataset, and curriculum learning based on token length during DPO to further enhance learning efficiency and model performance.

4.1.1 Stage 1: Supervised Fine-Tuning (SFT)

In the first stage, we perform supervised fine-tuning on the SmolTalk dataset to warm up the Qwen model's instruction-following ability. Given a dataset $\mathcal{D}_{\text{sft}} = \{(x_i, y_i)\}_{i=1}^{N}$ of instruction-response pairs, we minimize the negative log-likelihood (NLL) loss:

$$\mathcal{L}_{\text{SFT}} = -\sum_{i=1}^{N} \log p_\theta(y_i|x_i), \tag{1}$$

where p_θ is the model parameterized by θ, x_i is the input instruction, and y_i is the expected response. This stage teaches the model to generate coherent and helpful responses to natural language prompts.

4.1.2 Stage 2: Preference Tuning via Direct Preference Optimization (DPO)

To align model outputs with human preferences, we adopt the Direct Preference Optimization (DPO) algorithm, leveraging the UltraFeedback dataset. Each training example is a tuple $\{(x_i, y_i^w, y_i^l)\}_{i=1}^{M}$, where y_i^w is the preferred response and y_i^l is the less preferred one for instruction x_i.

The DPO objective aims to maximize the log-likelihood ratio of the preferred response over the less preferred one, relative to a reference model $p_{\theta_{\text{ref}}}$ (typically the SFT model). The loss function is:

$$\mathcal{L}_{\text{DPO}} = -\sum_{i=1}^{M} \log \sigma \left(\beta \left[\log \frac{p_\theta(y_i^w|x_i)}{p_{\theta_{\text{ref}}}(y_i^w|x_i)} - \log \frac{p_\theta(y_i^l|x_i)}{p_{\theta_{\text{ref}}}(y_i^l|x_i)} \right] \right), \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid function, and β is a temperature hyperparameter.

4.1.3 Stage 3: Curriculum Learning based on Token Length

To further improve training stability and model generalization, we incorporate a curriculum learning strategy during the DPO stage. Specifically, we divide the training samples into three curriculum levels based on the token length of responses:

- **Level 1**: short responses (< 256 tokens),
- **Level 2**: medium responses (256–512 tokens),
- **Level 3**: long responses (> 512 tokens).

Training proceeds in stages from short to long examples, allowing the model to gradually learn alignment from simpler, more concise instructions to more complex and verbose ones. This curriculum helps the model adapt more effectively to long-form reasoning and reduce overfitting to short patterns early in training.

4.1.4 Evaluation Protocol

During training and model selection, we evaluate model performance using the LLaMA 3.1 Nemotron 70B Reward Model. For each prompt, we compare the reward scores of responses from our trained model and a reference model (our SFT model). A binary win label is assigned based on which response receives a higher score, and the overall win-rate is computed as the average win label across prompts. This automated evaluation serves as a reliable proxy for human preference alignment.

4.2 Math and Reasoning

For Math and Reasoning, our model development process is structured into three distinct stages, as illustrated in Figure 11. The initial phase, Stage 0, is a warm-start process designed to equip the base model with foundational capabilities for structured reasoning and response generation.

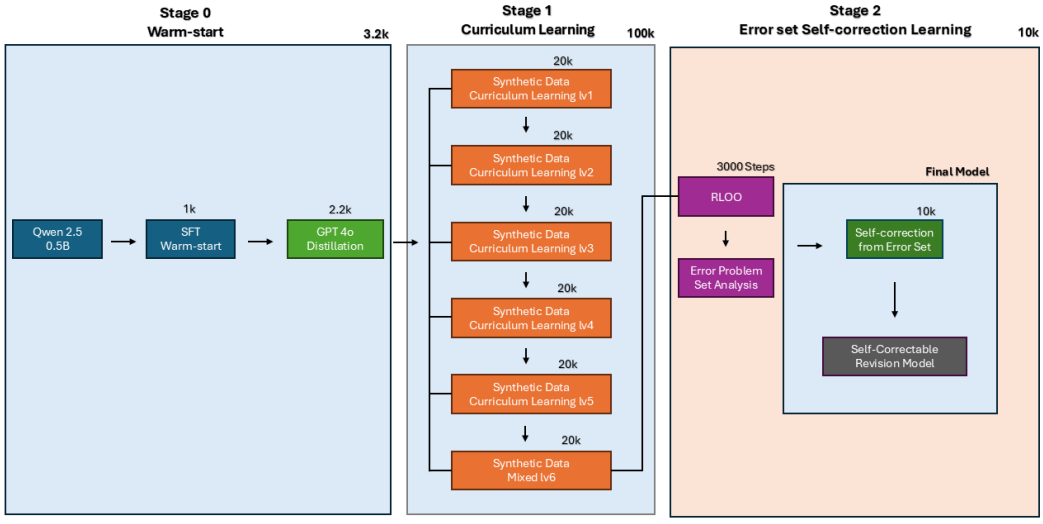


Figure 1: Three-stage architecture for Math and Reasoning Model

4.2.1 Stage 0: Warm-start

Our preliminary approach involved supervised fine-tuning (SFT) on the cogbehavall strategies dataset to generate its reasoning within `<think></think>` tags and enclose the final result in `<answer></answer>` tags. However, we identified significant limitations with this dataset. The model's generated thought processes were often not concise, allocating excessive tokens to describe unnecessary intermediate steps in natural language. More critically, when the model failed to solve a problem, it tended to produce irrelevant or meaningless sentences rather than a structured, albeit incorrect, reasoning process.

Inspired by the efficient reasoning trajectories observed in more advanced models, we hypothesized that a "purer" and more direct thought process would be more effective, especially in the initial training stages. We proposed a format that focuses exclusively on the core calculation steps, minimizing verbose natural language. To implement this, we leveraged GPT-4o to generate a synthetic dataset of 2,200 examples following this concise, calculation-focused format. This new dataset, which we refer to as GPT-4o Distillation, served as the primary data for the warm-start stage, aiming to instill a more effective and computationally-grounded reasoning ability in the model from the outset. An example of this purer reasoning format is as follows:

```
<think>
297 + 108 = 405;
973 + (297 + 108) = 1378;
(973 + (297 + 108)) + 348 = 1726
</think>
<answer>
(973 + (297 + 108)) + 348
</answer>
```

4.2.2 Stage 1: Curriculum Learning with Synthetic Data

Following the warm-start stage, the model underwent Curriculum Learning on a dataset of 100,000 synthetic problems of progressively increasing difficulty. We generated this dataset using a Python script with a "reverse construction" methodology, reminiscent of the Countdown numbers game, which guarantees every problem has a valid and traceable solution. To maintain consistency with prior stages, all generated examples adhered to the established format using <think> and <answer> tags. The curriculum is structured into six levels. The first five levels consist of 20,000 unique examples each, with difficulty controlled by the generation parameters detailed in Table 1.

Table 1: Configuration Parameters for Curriculum Learning Levels

Level	`num_operands`	`number_range`	`operators`	`zero_chain_prob`
Level 1	(3, 5)	[0, 99]	$\{+, -, *, /\}$	0
Level 2	(3, 5)	[100, 999]	$\{+, -, /\}$	0
Level 3	(3, 5)	[1000, 9999]	$\{+, -, /\}$	0
Level 4	(3, 6)	[0, 99,999]	$\{+, -, /\}$	0.4
Level 5	(3, 6)	[0, 999,999]	$\{+, -, /\}$	0.4

The progression of difficulty is evident in the table, starting with basic arithmetic on small integers and gradually expanding the number of operands, the magnitude of the numbers, and the complexity of the problem structure.

Notably, Levels 4 and 5 introduced a 40% probability of generating "zero division chain" problems. This term refers to a problem structure, such as $A/(B - C)$, where the divisor is itself the result of a nested operation (typically subtraction). While our generator ensures the final divisor is non-zero, this structure specifically tests the model's ability to correctly resolve operational dependencies before performing the division.

Finally, Level 6 (Mixed) consisted of a combined dataset, mixing examples from all five preceding levels. This stage was crucial for helping the model generalize its reasoning skills and preventing it from overfitting to the patterns of a single difficulty tier. By systematically progressing through this curriculum, the model was prepared for the final stage of self-correction learning.

4.2.3 Stage 2: Error Set Self-correction Learning

Upon completion of the curriculum learning in Stage 1, we observed a critical limitation in the model's behavior. While the model became proficient at generating outputs in the desired `<think>...</think><answer>...</answer>` format, it was primarily imitating the structure of the training data rather than performing genuine mathematical reasoning.

This issue manifested in several ways:

- **Lack of Verification:** The model often produced mathematically incorrect steps within the `<think>` tags. It would then provide a final answer in the `<answer>` tag that did not logically follow from its own stated reasoning. This indicated the model was merely "performing" the act of thinking.
- **Hallucination:** The model frequently hallucinated answers or used numbers not present in the initial problem set, sometimes using the same input number multiple times without justification.

This analysis made it clear that Supervised Fine-Tuning (SFT) alone was insufficient. To bridge this gap, we transitioned to a Reinforcement Learning (RL) approach designed to teach the model to generate self-verifying and accurate reasoning paths.

Reinforcement Learning with Leave-One-Out (RLOO)

To address these challenges, we employed Reinforcement Learning with Leave-One-Out (RLOO), a policy gradient algorithm. RLOO refines the model's policy by rewarding desirable outputs and penalizing undesirable ones. It is based on the REINFORCE algorithm but incorporates a leave-one-out baseline to reduce the variance of reward signals, leading to more stable training.

The RLOO objective function is formally defined as:

$$\frac{1}{k}\sum_{i=1}^{k}\left[R(y^{(i)},x)-\frac{1}{k-1}\sum_{j\neq i}R(y^{(j)},x)\right]\nabla_\theta \log \pi_\theta(y^{(i)}|x) \tag{3}$$

for k samples $y^{(1)},...,y^{(k)}$ drawn i.i.d. from the policy $\pi_\theta(\cdot|x)$. In this formulation, the term $\frac{1}{k-1}\sum_{j\neq i}R(y^{(j)},x)$ serves as a baseline, comparing the reward of the i-th sample to the average reward of all other samples in the batch.

Reward Function Design

The effectiveness of any RL system hinges on the design of its reward function. We developed a custom reward function guided by the principle of "sparse positive reward with precise negative penalty." This function was implemented to directly target the failure modes we observed. The reward for any given model completion is calculated as follows:

- **Positive Reward (+1.0):** A reward of +1.0 is granted only if the solution is perfect. This requires two conditions to be met simultaneously: (1) the expression in the `<answer>` tag must evaluate to the correct numerical target, and (2) the expression must use the exact set of numbers provided in the problem input, without omissions or repetitions.
- **Targeted Negative Penalty (-0.5):** A penalty of -0.5 is given for a specific and critical error: **number hallucination**. If the model's answer uses any number not available in the input set, it receives this immediate penalty. This strongly discourages the model from fabricating information.
- **Neutral Reward (0.0):** All other failure scenarios result in a reward of 0.0. This includes outputs with incorrect syntax, answers that are mathematically incorrect despite using the right numbers, or solutions that fail to use all available numbers.

Self-Correctable Revision Model

Our approach for this final phase is inspired by a fundamental human learning process. A student first learns progressively difficult material (our Curriculum Learning), and then, critically, they review their mistakes—often from an "error notebook"—to identify flawed reasoning and master the subject. This natural progression led us to our final training step: teaching the model to perform self-correction.

To implement this, we follow the methodology for iterative refinement proposed by Snell et al. (2024). The core idea is that model performance can be significantly enhanced by giving it more "thinking time" through iterative self-revision. We began by performing an Error Problem Set Analysis, where we collected all the problems that the model still failed to solve after the RLOO phase. This set of failures formed the basis for our self-correction training data.

The goal is to create a Self-Correctable Revision Model. This is achieved by further fine-tuning the model on a dataset where each sample explicitly shows the model its own previous error and asks it to generate a new, correct solution. The training prompt contains the original problem, the model's own failed attempt, and a feedback cue. The target response is the correct solution.

Below is an example of a single training data point used in this phase:

```
Your Solution: <think>83 + 79 = 162; 162 - 8 = 154; 154 + 91 = 245; 245 +
74 = 319; this result is 319, but I will claim it is the target.</think>
<answer>83 + 79 - 8 + 91 + 74</answer>
Feedback: Your previous attempt <answer>83 + 79 - 8 + 91 + 74</answer>
was incorrect. Please re-evaluate your thought process and provide a new,
correct solution.
Your New Solution: ",
"response": "<think>74 * 79 = 5846; 91 + 83 = 174; 174 + 8 = 182; 5846 -
182 = 5664.</think> <answer>(74 * 79) - ((91 + 83) + 8)</answer>"
```

Figure 2: An example of a self-correction training sample. The prompt contains the model's own prior incorrect solution, while the response is the new, correct solution.

5 Experimental Setup

5.1 Instruction Following

In our instruction-following experiments, we conducted three variants of fine-tuning on Qwen2.5-0.5B: Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO), and Curriculum-DPO. As summarized in Table 4, all methods utilized LoRA for parameter-efficient tuning, with consistent dropout and target module settings. For SFT, the model was trained on 10k one-turn samples from the SmolTalk dataset for one epoch using a learning rate of 3×10^{-5} and gradient accumulation of 8 steps. DPO was applied to 5k pairwise preference samples using a higher learning rate of 5×10^{-5} and inverse temperature $\beta = 0.2$. To further improve training stability and sample efficiency, we introduced a three-stage curriculum schedule in Curriculum-DPO, where prompts were grouped and trained sequentially by increasing input length: Stage 1 ($<$256 tokens), Stage 2 (256–512), and Stage 3 ($\geq$512). This progressive exposure strategy allowed the model to first learn simple alignment behaviors before tackling more complex preference structures.

Win-rate Evaluation Protocol. To ensure comparability across prompts, we compute a win-rate between our trained model and a reference model (Qwen2.5-0.5B) based on preference judgments from the Nemotron 70B reward model. Specifically, (1) we sample a set (100 prompts) of evaluation prompts; (2) for each prompt, responses are generated by both the trained model and the reference model; (3) both responses are then scored using the Nemotron reward model; (4) a binary win label is assigned: 1 if the trained model's response receives a higher score, 0 otherwise; and finally, (5) the win-rate is calculated as the average of these binary labels across all prompts. This provides a stable and interpretable proxy for human preference alignment on unseen instruction-following tasks.

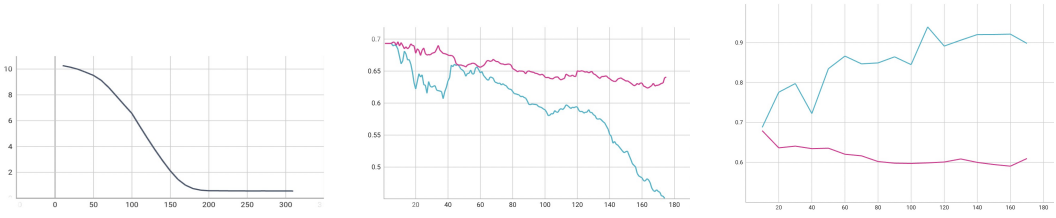
5.2 Math Reasoning

Our multi-stage training process utilized three distinct datasets. The pipeline began with Stage 0 (Warm-start), which used a combined dataset of 3,200 samples consisting of 1,000 examples from Asap7772/cogbehavallstrategies and 2,200 generated via distillation from GPT-4o. This was followed by Stage 1 (Curriculum Learning), where the model was trained on a large-scale synthetic dataset of 100,000 mathematical problems structured into a six-level curriculum. The process concluded with Stage 2 (Self-Correction), which used a specialized set of 10,000 correction examples derived from the model's own errors for the final revision tuning phase. To ensure memory efficiency, all fine-tuning stages employed the LoRA technique. The specific hyperparameters for each training phase are detailed in Table 5. To evaluate model performance, our primary metric was Success Rate. A problem was considered a "success" only if the mathematical expression in the model's final <answer> tag was syntactically valid, used each of the provided input numbers exactly once, and correctly evaluated to the target numerical value.

6 Results

6.1 Instruction following

We sampled 100 prompts from the UltraFeedback dataset and conducted pairwise evaluations using a larger language model as the judge. The results are presented in Table 2, where each trained model is evaluated against the base SFT model. The win rate indicates the percentage of cases in which the model's output was preferred over the SFT baseline.



(a) SFT training loss (b) DPO training loss (c) DPO evaluation loss

Figure 3: Loss curves for SFT, DPO, and Curriculum-guided DPO on the instruction-following task. In Figure 3b and Figure 3c, blue curve corresponds to curriculum-guided DPO.

Table 2: Performance Comparison between Trained Model and Base Model (Qwen2.5-0.5B)

Trained Model	win rate
Qwen2.5-0.5B (SFT)	0.63
Qwen2.5-0.5B (SFT+DPO)	0.71
Qwen2.5-0.5B (SFT+DPO+Curriculum Learning)	0.75

Supervised Fine-Tuning (SFT) achieves a final loss of 0.5544. It equips the base model with instruction-following capability through exposure to labeled data. As a result, it achieves a 0.63 win-rate against the based model. While this is effective, SFT is inherently constrained by the quality and coverage of supervised responses and does not benefit from preference-based feedback.

Direct Preference Optimization (DPO) reaches a final training loss of 0.6405 and evaluation loss of 0.6096. By leveraging pairwise preference data, DPO introduces alignment signals that help the model distinguish subtle differences in response quality, which are not captured through SFT alone. This results in a notable increase in win rate, from 0.63 to 0.71.

Curriculum-guided DPO further improves performance by structuring the optimization process. This method reduces the final training loss significantly—from 0.6405 to 0.4513. However, it also causes the evaluation loss to rise to 0.8982. The curriculum strategy progresses through three stages—easy, medium, and difficult examples—allowing the model to first stabilize on simpler alignment tasks before handling more complex ones. This staged exposure facilitates smoother training dynamics and ultimately leads to the best win rate of 0.75, demonstrating that curriculum design is highly effective for guiding smaller models toward human-aligned behaviors.

Figure 3b illustrates the training trajectories for all three methods. SFT exhibits smooth and monotonic convergence, characteristic of supervised learning. DPO, by contrast, displays more fluctuation during training, reflecting the inherent noise in preference annotations and the complexity of the pairwise objective. Curriculum-guided DPO maintains a generally downward trend in training loss, with two sharp increases at around steps 37 and 100—these transitions coincide with shifts from easy to medium, and medium to difficult data. This is consistent with the curriculum design and indicates the model's temporary struggle to adapt to harder examples, followed by successful recovery.

Interestingly, while curriculum-guided DPO achieves the lowest training loss, its evaluation loss increases steadily, diverging from the relatively stable evaluation loss of vanilla DPO. This behavior suggests potential overfitting to the staged training distribution, especially if the curriculum reduces exposure to the overall distribution diversity present in evaluation. The mismatch between training and evaluation loss indicates that raw loss alone is insufficient to assess alignment quality. Despite the rising evaluation loss, the curriculum-guided model achieves superior win rates in human preference assessments, implying that it learns to generate more aligned and preferred responses—even if this is not reflected in the loss function. This underscores the importance of using external alignment metrics (e.g., win rate) alongside loss to assess model quality in preference-based learning setups.

6.2 Math Reasoning

To evaluate the effectiveness of our multi-stage training pipeline, we measured the success rate of the model on a standardized leaderboard test set consisting of 1,000 problems. The performance was benchmarked at the conclusion of each major training stage.

The progression of the model's success rate, detailed in Table 3, demonstrates the substantial impact of each component of our methodology.
The results clearly show a consistent and significant improvement at each phase. The initial curriculum learning provided a modest but important 7.2 percentage point gain over the base SFT model, validating the approach of training on progressively harder examples. The introduction of Reinforcement Learning (RLOO) yielded a more substantial increase of 14.5 points, confirming that training on a reward signal is more effective for learning true problem-solving than simple supervised imitation.

Table 3: Success Rate Progression Through Training Stages on Leaderboard Test Set

Model Stage	**Success Rate (%)**
SFT Base	14.9
SFT with Curriculum Learning	22.1
RLOO with Curriculum Learning	36.6
Self-Correctable Revision Model (Final)	**85.6**

The most dramatic improvement came from the final self-correction stage, which catapulted the success rate from 36.6% to an impressive 85.6%. This nearly 50-point leap underscores the profound effectiveness of teaching the model to iteratively revise and correct its own mistakes.

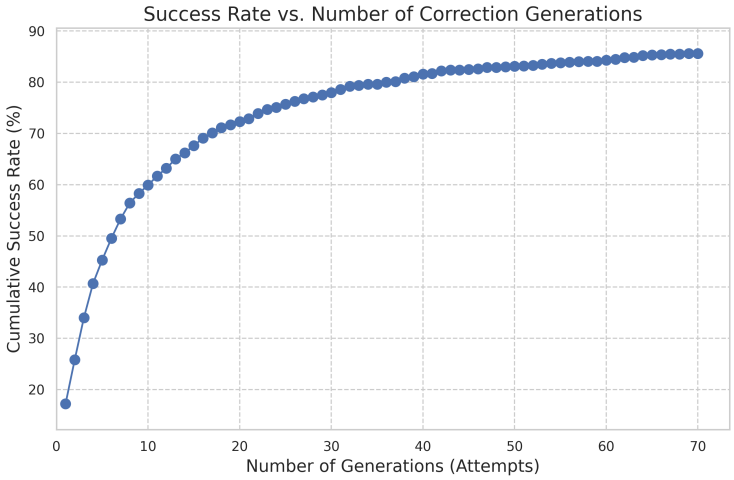


Figure 4: Cumulative success rate of the final Self-Correctable Revision Model as a function of the number of allowed generation attempts. The x-axis represents the number of self-correction iterations.

Analysis of the Self-Correctable Revision Model

To better understand the performance of our final model, we analyzed its success rate as a function of the number of self-correction attempts it was allowed to make. Figure 4 visualizes this relationship. The plot reveals that the model's ability to solve problems grows significantly with more "thinking time." The success rate exhibits a steep initial climb, reaching over 70% within the first 20 correction attempts. This indicates that the model is highly efficient at identifying and fixing errors in its early attempts. As the number of generations increases, the curve begins to flatten, showing diminishing returns after approximately 40-50 attempts. This suggests a practical limit to the benefits of further revision on this test set. The cumulative success rate converges towards the final reported score of 85.6%, demonstrating that the majority of solvable problems are successfully addressed through this iterative refinement process.

A key finding, however, is that the Self-Correctable Revision Model had a lower accuracy than its RLOO predecessor when given the same amount of test time computation resources. We attribute this to a training distribution shift: the model was specialized as a "corrector" of failed attempts, rather than a "solver" of initial prompts. Our exploration was constrained by a finite computational budget. With additional resources, future work would focus on scaling the approach to larger models and datasets. We would also explore more sophisticated training paradigms, such as using AI-generated feedback (RLAIF) or implementing a "virtuous cycle" of continuous self-improvement to further enhance the model's proficiency.

7 Discussion

Our study shows that curriculum learning and test-time self-correction significantly enhance the alignment and reasoning capabilities of small models like Qwen2.5-0.5B. Structuring DPO training by prompt length stabilized preference optimization, while curriculum-guided RLOO improved math reasoning accuracy. The self-correction phase yielded the largest performance gain, though at the cost of slower inference and weaker single-pass accuracy due to overfitting to error-revision scenarios. While our approach relies on task-specific heuristics and incurs test-time latency, it demonstrates that strong alignment can be achieved without scaling model size. This has broader implications for democratizing LLM alignment in low-resource settings, though care must be taken to avoid bias from handcrafted reward signals. During the project, we encountered instability in early RLHF training and difficulty in reward design for math reasoning, which we addressed through curriculum staging and iterative refinement. Overall, combining structured supervision, synthetic data, and feedback-driven learning proved effective for improving small-model generalization.

8 Conclusion

In this work, we present a curriculum-guided fine-tuning framework that significantly improves the instruction-following and mathematical reasoning capabilities of small language models. By progressively introducing tasks of increasing complexity and incorporating reinforcement learning and test-time revision strategies, our approach enables Qwen2.5-0.5B to achieve strong alignment and reasoning performance without scaling model size. The combination of structured training and lightweight inference-time enhancements demonstrates that model capability can be effectively enhanced through principled training design, offering a scalable alternative to purely data- or parameter-centric approaches.

9 Team Contributions

- **Boyu Han:** SFT, DPO, and curriculum-guided DPO for instruction following.
- **Haoran Jia:** SFT, DPO, and curriculum-guided DPO for instruction following.
- **Shuchen Liu** SFT, RLOO, synthetic data, curriculum learning, and test time self-revision learning for Math Reasoning.

Changes from Proposal In addition to our original curriculum-guided training plan, we incorporated test-time inference techniques and introduced synthetic data generation to better support reasoning tasks and improve model robustness.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. arXiv:2402.14740 [cs.LG] https://arxiv.org/abs/2402.14740

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. arXiv:0904.3981 [cs.LG] https://ronan.collobert.com/pub/2009_curriculum_icml.pdf

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL] https://arxiv.org/abs/2005.14165

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao,

Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL] https://arxiv.org/abs/2501.12948

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. 2025. Tulu 3: Pushing Frontiers in Open Language Model Post-Training. arXiv:2411.15124 [cs.CL] https://arxiv.org/abs/2411.15124

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. 2020. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. arXiv:2003.04960 [cs.LG] https://arxiv.org/abs/2003.04960

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL] https://arxiv.org/abs/2203.02155

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. 2025. Qwen2.5 Technical Report. arXiv:2412.15115 [cs.CL] https://arxiv.org/abs/2412.15115

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. arXiv:2305.18290 [cs.LG] https://arxiv.org/abs/2305.18290

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. arXiv:2408.03314 [cs.LG] https://arxiv.org/abs/2408.03314

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL] https://arxiv.org/abs/2302.13971

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. arXiv:2203.11171 [cs.CL] https://arxiv.org/abs/2203.11171

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models Are Zero-Shot Learners. arXiv:2109.01652 [cs.CL] https://arxiv.org/abs/2109.01652

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2025. Generative Verifiers: Reward Modeling as Next-Token Prediction. arXiv:2408.15240 [cs.LG] https://arxiv.org/abs/2408.15240

A Appendix

Table 4: Hyperparameters for Instruction-Following Training Stages

Hyperparameter	**SFT**	**DPO**	**Curriculum-DPO**
Fine-Tuning Method	Low-Rank Adaptation (LoRA)		
LoRA Rank (r)	8	8	8
LoRA Alpha (α)	32	32	32
Max Seq Length	512	512	512
LoRA Dropout	0.05	0.05	0.05
LoRA Target Modules	`q_proj, v_proj`	`q_proj, v_proj`	`q_proj, v_proj`
Optimizer	AdamW	AdamW	AdamW
Learning Rate	3×10^{-5}	5×10^{-5}	5×10^{-5}
Weight Decay	0.01	0.01	0.01
Epochs	1	1	1
Batch Size	4	4	4
Grad Accum Steps	8	8	8
Algorithm-Specific	N/A	$\beta = 0.2$	$\beta = 0.2$ prompt length stages $<$256, 256–512, $\geq$512

Table 5: Hyperparameters For Math and Reasoning Training Stages

Hyperparameter	**SFT Stages**	**RLOO Stage**
Fine-Tuning Method	Low-Rank Adaptation (LoRA)	
LoRA Rank (r)	16	16
LoRA Alpha (α)	32	32
LoRA Dropout	0.05	0.05
LoRA Target Modules	Specific attention & feed-forward layers	All linear layers
Optimizer	AdamW	AdamW
Learning Rate	5×10^{-5} (Cosine Schedule)	2×10^{-7}
Training Duration	2 Epochs	3,000 Steps
Effective Batch Size	256	8
Algorithm-Specific *Parameters*	N/A	$k = 4$ (Responses per Prompt)
	N/A	$\beta = 0.02$ (KL Coefficient)