

## Extended Abstract

**Motivation** Large language models (LLMs) demonstrate impressive capabilities in language understanding and reasoning, but still face challenges in faithfully following complex instructions and performing multi-step mathematical reasoning. Improving the alignment and reasoning ability of LLMs requires effective training strategies and diverse evaluation. In this project, we aim to enhance LLM performance on two key tasks: instruction following and math reasoning, by exploring deep learning, reinforcement learning-based methods and leveraging AI-generated feedback, as well as external tools.

**Method** We adopt a sequential reinforcement learning pipeline, using Supervised Fine-Tuning (SFT) as a foundational step for both tasks. For instruction following, we incorporate Direct Preference Optimization (DPO) and Reinforcement Learning from AI Feedback (RLAIF). For math reasoning, we introduce REINFORCE Leave-One-Out (RLOO) and an external calculator tool that systematically explores arithmetic expressions. These methods are designed to optimize for model alignment, performance, and generalizability, and we evaluate them on diverse benchmarks.

**Implementation** We use the Qwen-2.5-0.5B language model as the backbone. The SFT stage trains on high-quality preference and verifier-based datasets (SmolTalk and WarmStart, respectively). DPO and RLAIF further optimize the instruction-following model using filtered preference data. The math reasoning pipeline utilizes RLOO, supported by the Countdown dataset, and an external calculator tool that provides solution strings. All models train on a single Nvidia L40S GPU and are evaluated on leaderboard-provided held-out prompts.

**Results** Our experiments show that SFT, DPO, and RLOO each improves model performance compared to the base model on their respective tasks. DPO achieves the 6th leaderboard ranking, with score 0.32, reward -21.07, and win rate 72% for the instruction following task. The tool-augmented method achieves near-perfect accuracy and the highest leaderboard rank for the math reasoning task. The leaderboard score is 0.99 and the win rate is 99%. RLAIF demonstrates scalability and competitive alignment, though with some trade-offs in output fluency and diversity. Qualitative analysis further confirms that each reinforcement learning stage produces increasingly accurate and interpretable responses.

**Discussion** While our reinforcement learning strategies and extensions show strong quantitative and qualitative gains, several challenges remain. The instruction model performs poorly on foreign language prompts. This is likely due to the fact that most of the training data we use is mostly in English. Moreover, Out-of-distribution and non-standard prompts also lead to degraded responses, showing the need for more diverse data and robust handling of unfamiliar queries. In the meantime, while RLAIF offers scalable alternatives, it may reduce output diversity or fluency. The external tool use is highly effective for structured reasoning but relies on accurate invocation and does not address all reasoning challenges. Future work may include developing a larger model that can solve both tasks at the same time.

**Conclusion** This work demonstrates that combining deep learning and reinforcement learning techniques significantly enhances instruction following and math reasoning capabilities. Extensions such as RLAIF and external tool integration provide scalable and robust improvements, though further research is needed to expand generalization to multilingual and out-of-distribution scenarios. Future directions include exploring larger models, integrating additional external tools, and improving data diversity to create LLMs that are both more robust and more generally capable.

---

# Galaxy: Fine-Tuning Large Language Models with Reinforcement Learning

---

**Feiyang Zhu**

Department of Computer Science  
Stanford University  
zfy@stanford.edu

**Shutong Zhang**

Department of Computer Science  
Stanford University  
tonyzst@stanford.edu

**Siyuan Wu**

Department of Computer Science  
Stanford University  
wusiyuan@stanford.edu

## Abstract

Large language models (LLMs) have demonstrated strong performance across a range of tasks, including text generation, image classification, and reasoning. To better align these base models with specific user needs, fine-tuning plays a crucial role. In this project, we aim to improve a large language model’s instruction following and math reasoning abilities through fine-tuning. Specifically, we work on Qwen-2.5-0.5B model, apply Supervised Fine-Tuning (SFT) and incorporate reinforcement learning techniques, including Direct Preference Optimization (DPO) and REINFORCE Leave-One-Out (RLOO). For the instruction following task, we use SmolTalk dataset and UltraFeedback dataset for SFT and DPO, respectively. We also incorporate Reinforcement Learning from AI Feedback (RLAIF) for preference scoring. For the math reasoning task, we use WarmStart dataset and Countdown dataset for SFT and RLOO, respectively. To further boost the accuracy, we use a calculator tool during both training and inference time.

## 1 Introduction

Recent advances in large language models (LLMs) have made them essential across a wide range of fields, including computer vision, natural language processing, and robotics. Applications include object detection, autonomous driving, text autocompletion, and emotion recognition. Large language models excel at providing clear and accurate answers to general questions, but often struggle with domain-specific or user-specific queries. Thus, in such a transfer learning problem, fine-tuning is important to align the large language model with user needs. In this project, we are interested in fine-tuning a large language model using deep learning and reinforcement learning techniques to improve its performance in downstream tasks, including instruction following and math reasoning.

Meanwhile, state-of-the-art (SOTA) models often have extremely large parameter counts. For example, Meta’s Llama 3 has 70 billion parameters. Fine-tuning models of this scale is highly challenging due to their computational and memory demands. Also, SOTA models already have excellent instruction following and math reasoning abilities, so the improvement through fine-tuning will not be obvious. Thus, we choose Qwen-2.5-0.5B model as our base model. On one hand, a model with 0.5 billion parameters is relatively manageable for fine-tuning. On the other hand, Qwen-2.5 models have reasonable instruction following and math capabilities. We believe Qwen-2.5-0.5B is a promising model for us to explore.

For both the instruction following and math reasoning tasks, we will use Supervised Fine-Tuning (SFT) as the first step to provide the model with basic knowledge through deep learning. We will then incorporate reinforcement learning methods. For the instruction following task, we will use Direct Preference Optimization (DPO) and Reinforcement Learning from AI Feedback (RLAIF) techniques. For the math reasoning task, we will use REINFORCE Leave-One-Out (RLOO) and a calculator tool to help improve the performance. Eventually, fine-tuned models on both tasks achieve better performance than the base model.

## 2 Related Work

### 2.1 Reinforcement learning from AI feedback (RLAIF)

The paper **RLHF** Ouyang et al. (2022) proposes an approach to incorporate human preference in reinforcement learning. Reward functions, the traditional approach in reinforcement learning, can be challenging to design, especially in complex environments. Besides, incorporating human feedback can help the model better align with human preferences.

In addition, the paper **RLAIF** Lee et al. (2024) explores a compelling alternative to traditional Reinforcement Learning from Human Feedback (RLHF) by replacing costly and limited human preference data with synthetic feedback generated by AI assistants. In particular, they propose Reinforcement Learning with AI Feedback (RLAIF) that uses LLM for response generation and preference scoring. Moreover, this paper also adapts the Direct-RLAIF that directly uses LLM feedback as the reward signal instead of the traditional reward model in RL, which efficiently mitigates the issue of reward staling. Experiment results show that the RLAIF is a scalable and efficient approach, achieving comparable or even better performance on alignment tasks compared with RLHF, and reduces hallucination that is prevalent in RLHF. Notably, a small-scale experiment conducted by the authors shows that the difference between RLHF and RLAIF in terms of accuracy, coverage, and coherence is not statistically significant. It significantly reduces the reliance on human annotators, enabling an easier and faster data generation process.

However, despite its advantages in scalability and label generation, RLAIF still has several drawbacks. First of all, RLAIF sometimes produces less fluent summaries than RLHF, which leads to a performance drop in the final model. Moreover, RLAIF has been observed to produce repetitive responses that fail to accurately capture the intent of the original input. It is also unknown whether RLAIF generalizes well across diverse domains such as code generation, question answering, or instruction-following since the paper only evaluates summarization and dialogue generation tasks. Though RLAIF enables data generation at scale, it does not explicitly address the reduction in diversity that occurs as RL training progresses, which can lead to model capacity challenges. These limitations may suggest that simply increasing feedback volume, even synthetically, may not be sufficient to resolve deeper inefficiencies in the learning process.

### 2.2 Effective Tool Use

Tools are functions that allow LLMs to interact with external applications, providing an interface for accessing and retrieving information beyond the model’s static training data. This capability is particularly valuable for queries requiring high precision (e.g., mathematical calculations) or up-to-date information (e.g., live weather data).

A typical tool-use workflow involves the LLM identifying when a tool is needed based on the query and available tool descriptions Hao et al. (2024); Lu et al. (2025). Upon detecting a relevant tool, the model halts text generation and emits a structured response—commonly a JSON object—specifying the tool’s name and parameter values. The external function is then executed, and its output is returned to the LLM to complete the response. This architecture allows LLMs to augment their reasoning with reliable, external computation or retrieval.

Optimizing tool use with RL presents unique challenges and opportunities. RL-based approaches can train models not only to recognize when a tool is necessary but also to select the most appropriate tool and parameters for a given context. Recent research explores how RL can be used to maximize the utility of tool calls, balancing the benefits of external information against the costs of unnecessary or incorrect tool invocation.

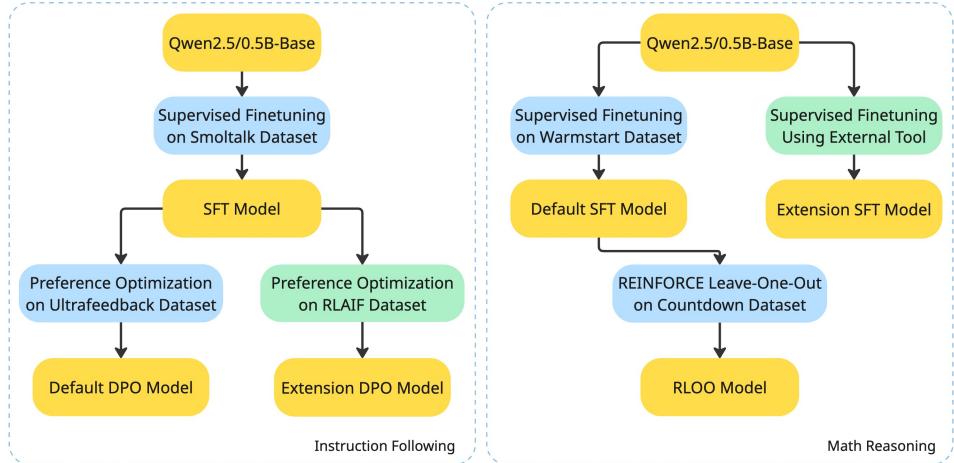


Figure 1: **Overview of our training pipeline for instruction following and math reasoning tasks.** For instruction following (left), we apply supervised fine-tuning on the SmolTalk dataset, followed by preference optimization using either human feedback (UltraFeedback) or AI-generated feedback (RLAIF), resulting in default and extension DPO models. For math reasoning (right), we fine-tune on the WarmStart dataset and further optimize using REINFORCE Leave-One-Out (RLOO) on the Countdown dataset, while the extension incorporates supervised fine-tuning with an external calculator tool.

Recent advances in this area include several notable works. Schick et al. (Schick et al., 2023) introduced Toolformer, a framework in which large language models (LLMs) teach themselves to use tools by generating synthetic demonstrations of tool use and fine-tuning on these examples. This method allows models to autonomously learn when and how to invoke tools during text generation. Jin et al. (Jin et al., 2025) proposed Search-RL, a reinforcement learning approach that trains LLMs to reason effectively and leverage search engines by issuing queries and integrating retrieved information, thereby enhancing factual accuracy. Meanwhile, Gehring et al. (Gehring et al., 2025) developed RLEF, a technique that grounds code LLMs in execution feedback through reinforcement learning, enabling models to iteratively refine their code outputs by interacting with execution environments and learning from the outcomes. Together, these works highlight promising directions for enabling LLMs to use external tools more effectively.

### 3 Method

In this section, we present the methods we use for the instruction following and math reasoning tasks, as well as our extensions. Figure 1 provides an overview of our method pipeline.

#### 3.1 Supervised Fine-Tuning (SFT)

Supervised Fine-Tuning (SFT) is the initial step in our RL pipeline for **both instruction following and math reasoning tasks**, adapting a pre-trained language model to specific tasks using high-quality expert demonstrations.

**Implementation Details.** The model is trained to maximize the likelihood of preferred responses for each query, applying the next-token prediction loss only to the completion portion of the data. Formally, for queries  $x$  and completions  $y$  from an expert dataset, the SFT objective is:

$$\max_{\theta} \mathbb{E}_{x,y \in \mathcal{D}} \sum_{t=1}^{|y|} \log \pi_{\theta}(y_t | x, y_{<t})$$

where  $\pi_{\theta}$  denotes the model with parameters  $\theta$ .

### 3.2 Direct Preference Optimization (DPO)

In the second stage of the **instruction following task**, we apply Direct Preference Optimization (DPO) (Rafailov et al., 2024) to fine-tune a large language model for improved alignment with human preferences.

**Implementation Details.** The model is trained to increase the log-probability of generating the chosen response relative to the rejected response, following the DPO objective function:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[ \log \sigma \left( \beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right].$$

In the above equation,  $\pi_\theta$  donates the fine-tuned model,  $\pi_{\text{ref}}$  donates the reference model used to prevent the model from diverging from the original model, and  $\beta$  is the hyperparameter that controls the sharpness of the optimization.

### 3.3 REINFORCE Leave One-Out (RLOO)

In the second stage of the **math reasoning task**, we applied the REINFORCE Leave One-Out (RLOO) (Ahmadian et al., 2024) to fine-tune a large language model for improved reasoning abilities, including planning, decomposing, and executing mathematical steps.

**Implementation Details.** RLOO is an online RL algorithm. During training, the model will generate  $k$  responses, and we will use a reward model or a reward function to assign a reward value for each response. RLOO aims to reduce the variance of samples by leave one-out compared with the baseline. The objective function is as follows:

$$\frac{1}{k} \sum_{i=1}^k \left[ R(y_{(i)}, x) - \frac{1}{k-1} \sum_{j \neq i} R(y_{(j)}, x) \right] \nabla \log \pi(y_{(i)} | x) \quad \text{for } y_{(1)}, \dots, y_{(k)} \stackrel{i.i.d.}{\sim} \pi_\theta(\cdot | x)$$

In the above equation,  $\pi_\theta$  donates the fine-tuned model,  $y_{(1)}, \dots, y_{(k)}$  denotes the  $k$  responses generated by the model, where  $k$  is the hyperparameter that decides how many responses the model generates for each prompt.  $\log \pi(y_{(i)} | x)$  denotes the log-probability of generating the response  $y_{(i)}$  given the prompt  $x$ .  $R(y_{(i)}, x)$  denotes the reward for a specific response  $y_{(i)}$  with the prompt  $x$ . In our case, since we use RLOO for the math reasoning task, we know the ground truth value. Thus, instead of training a reward model, we use Countdown Reward Function Gandhi et al. (2025) to get the reward value for each response by evaluating its answer format and correctness.

### 3.4 Extension

#### 3.4.1 Reinforcement learning from AI feedback (RLAIF)

Building on the concept of Reinforcement Learning with AI Feedback (RLAIF), we generate synthetic data for reward modeling in the **instruction following task** by leveraging the UltraFeedback dataset (Cui et al., 2024), which is also used for Direct Preference Optimization (DPO). Instead of relying on human annotations, we employ the Llama 3.1 Nemotron 70B Reward Model (Wang et al., 2024) to assign reward values to both the chosen and rejected completions.

Similar to the preprocessing for the DPO dataset, we re-rank the chosen and rejected responses and remove samples in which the chosen response receives a reward score lower than  $-10$ . We generated 4,000 pairs of good and bad prompts for the DPO training. This approach enables us to construct a new dataset containing high-quality reward signals produced by advanced AI models rather than human evaluators.

#### 3.4.2 Effective Tool Use

We introduce an effective tool use strategy for the **math reasoning task** by integrating a dedicated calculator tool. This tool is designed to provide closed-form solutions by exhaustively searching for valid arithmetic expressions that reach the target number.

Our calculator tool employs a depth-first search (DFS) algorithm to systematically explore all possible combinations of the provided numbers, arithmetic operations  $(+, -)$ , and the use of parentheses.

By traversing every valid arrangement, the tool guarantees a solution whenever one exists for the given Countdown instance. After computing the answer using the calculator tool, we concatenate the original query with the tool-generated answer and provide this combined input to the language model as shown in Figure 2. The model is then prompted to generate a detailed reasoning process that leads to the correct answer.

**Efficient Tool Use Example**

**System:** A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer.

**User:** Using the numbers [95, 36, 32], create an equation that equals 91. You can use basic arithmetic operations (+, -, \*, /) and each number can only be used once. Show your work in <think> </think> tags. And return the final answer in <answer> </answer> tags, for example <answer> (1 + 2) / 3 </answer>. The correct answer is <answer>95 - (36 - 32)</answer>.

**Assistant:** <think>If we do  $95 - (36 - 32) = 95 - 4 = 91$  This works perfectly! Let's verify the solution working forward: First:  $36 - 32 = 4$  Then:  $95 - 4 = 91$  This gives us our target of 91. </think> <answer>95 - (36 - 32)</answer>

Figure 2: A Sample Entry for the Efficient Tool Use Extension.

This setup allows us to assess the model’s ability to not only reproduce the correct final answer but also to generate a coherent and accurate reasoning chain, leveraging the external tool’s output as an intermediate step. This approach demonstrates how external tool use can enhance the factual accuracy and interpretability of language model outputs on mathematically challenging tasks.

## 4 Experimental Setup

### 4.1 Task Overview

In this project, we use **Qwen-2.5-0.5B** as our base model. We explore two independent downstream tasks: instruction following and math reasoning, with corresponding datasets and evaluation metrics.

#### 4.1.1 Instruction Following Task.

This task evaluates the model’s ability to generate responses based on natural language instructions. The goal is to ensure that the large language model can produce coherent, relevant, and helpful responses to unseen instructions while maintaining factual accuracy. For the instruction following task, we sequentially apply Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO) in our training pipeline. Given the nature of this task, we employ preference-based datasets at each stage of training.

#### 4.1.2 Math Reasoning Task.

This task evaluates the basic arithmetic ability of the model. The goal is to ensure that the large language model can solve multi-step mathematical problems with correct solutions using natural language. For the math reasoning task, we sequentially apply Supervised Fine-Tuning (SFT) and REINFORCE Leave One-Out (RLOO) in our training pipeline. Given the nature of this task, we employ verifier-based datasets at each stage of training.

## 4.2 Data Processing

### 4.2.1 SmolTalk Dataset.

- **Overview.** We use the SmolTalk dataset Ben Allal et al. (2025) during the SFT stage of the instruction following task. SmolTalk contains 460k examples in the training split and 24.2k examples in the test split. Each data entry is a multi-turn conversation between a user and an assistant, covering a wide range of topics including math, programming, daily conversation, and natural geology.

- **Data Processing.** we use a filtered subset of the SmolTalk dataset containing high-quality GPT-4o chat responses. Filtering focuses on the first round of multi-turn conversations and limits queries to under 128 words and completions to under 512 words, balancing training efficiency and data quality.

#### 4.2.2 UltraFeedback Dataset.

- **Overview.** After SFT, we use the UltraFeedback dataset Cui et al. (2024) during the DPO stage of the instruction following task. UltraFeedback contains 61.1k examples in the training split and 2k examples in the test split. Each data entry consists of four parts: system instructions, user prompt, chosen assistant response, and rejected assistant response.
- **Data Processing.** To further improve the quality and consistency of the preference data, we introduce an additional filtering step using a pre-trained reward model. Specifically, we use the Llama 3.1 Nemotron 70B Reward Model (Wang et al., 2024) to re-rank the chosen and rejected responses and remove samples in which the chosen response received a reward score lower than  $-10$ . After this filtering step, 13,434 samples remain in the dataset. A processed example can be found in Figure 3.

#### UltraFeedback Example

**System:** You are a helpful assistant.

**User:** Which animal has two hands, a hyrax or a dog?

**Assistant Chosen:** Neither a hyrax nor a dog has hands. Hyraxes have four legs with feet adapted for climbing and gripping. Similarly, dogs have four legs with paws. The term "hands" is usually reserved for primates, such as humans.

**Assistant Rejected:** Thank you for your question. I'm happy to help you with that! However, the question may not be factually coherent. A hyrax is a small mammal of the family Procaviidae, while a dog is a domesticated member of the Canidae. Neither has hands.

Figure 3: A Sample Entry from the Processed UltraFeedback Dataset.

#### 4.2.3 WarmStart Dataset.

- **Overview.** We use the WarmStart dataset Gandhi et al. (2025) during the SFT stage of the math reasoning task. WarmStart contains 1k examples in the training split and 200 examples in the test split. Each data entry consists of three parts: system instructions, user prompt, assistant response.
- **Data Processing.** We filter the training data by limiting query lengths to under 128 words and completion lengths to under 512 words. A processed example can be found in Figure 4.

#### 4.2.4 Countdown Dataset.

- **Overview.** After SFT, we use the Countdown dataset Pan et al. (2025) during the RLOO stage of the math reasoning task. Countdown contains 490k examples in total. Each data entry consists of two parts: a list of numbers and a target number. The model needs to use basic arithmetic operations to create an equation that equals to the target number.
- **Data Processing.** We adopt the same prompt structure used in the WarmStart dataset by incorporating the candidate numbers and the target into a formatted prompt. Due to the large size of the dataset (490k examples), we sample only a subset of the data for training. After preprocessing, we obtain a set of 2,000 prompt-formatted examples.

### 4.3 Metrics

We use the following evaluation metrics to evaluate the performance of our fine-tuned model against the baseline.

1. **Nemotron Reward:** We measure the reward score of the responses generated by our model. Specifically, we used the Llama 3.1 Nemotron 70B Reward Model for the instruction following task and the countdown score Gandhi et al. (2025) for the math reasoning task.

### WarmStart Example

**System:** A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer.

**User:** Using the numbers [95, 36, 32], create an equation that equals 91. You can use basic arithmetic operations (+, -, \*, /) and each number can only be used once. Show your work in `<think></think>` tags. And return the final answer in `<answer></answer>` tags, for example `<answer>(1 + 2) / 3</answer>`.

**Assistant:** `<think>If we do 95 - (36 - 32) = 95 - 4 = 91 This works perfectly! Let's verify the solution working forward: First: 36 - 32 = 4 Then: 95 - 4 = 91 This gives us our target of 91. </think>`  
`<answer>95 - (36 - 32)</answer>`

Figure 4: A Sample Entry from the Processed WarmStart Dataset.

2. **Win Rate:** We compare rewards of responses generated by the fine-tuned model against those generated by the SFT model and report the win rate.
3. **Leaderboard Score:** The score our model obtained on the leaderboard.
4. **Leaderboard Rank:** The rank our model obtained on the leaderboard.

## 4.4 Training Details

In this section, we provide the training details for all experiments. All training is conducted on a single Nvidia L40S GPU.

### 4.4.1 Instruction Following Task Training Setup

- **SFT:** We train the model using the AdamW optimizer with default parameters, a batch size of 8 with a gradient accumulation step of 8, and a learning rate of  $2 \times 10^{-5}$ . We train the model for 3 epochs.
- **DPO:** We train the model using the AdamW optimizer with default parameters, a  $\beta$  of 0.1, a batch size of 4 with a gradient accumulation step of 8, and a learning rate of  $1 \times 10^{-6}$  for three epochs.
- **RLAIF Extension:** We train the model using AI-generated data, following the same settings as in DPO training.

### 4.4.2 Math Reasoning Task Training Setup

- **SFT:** We train the model using the AdamW optimizer with default parameters, a batch size of 8 with a gradient accumulation step of 8, and a learning rate of  $2 \times 10^{-5}$ . We train the model for 50 epochs.
- **RLOO:** We train the model using the AdamW optimizer with default parameters, a  $k$  of 8, a batch size of 1 with a gradient accumulation step of 16, and a learning rate of  $1 \times 10^{-6}$  for three epochs.
- **External Tool Extension:** We train the model under the same settings as SFT training, using only data from the WarmStart dataset and training for 50 epochs.

## 4.5 Inference Details

We test the instruction following task using the 400 held-out prompts provided by the leaderboard, and the math reasoning task using the 1,000 held-out prompts provided by the leaderboard. For both tasks, we use a maximum sequence length of 1,024, a temperature of 0, a top\_p of 1.0, and a repetition penalty of 1.1 based on vllm inference settings.

## 5 Results

### 5.1 Quantitative Evaluation

Tables 1 and Table 2 present the quantitative evaluation results for the instruction following and math reasoning tasks. On the instruction following task, both SFT and DPO outperform the base model, with DPO achieving a leaderboard rank of 6th, a leaderboard score of 0.32, and a win rate of 72%. The RLAIF extension also performs competitively, reaching 8th place with a leaderboard score of 0.30 and a win rate of 69%, demonstrating that it is possible to approach DPO-level performance using much cheaper AI-generated feedback.

For the math reasoning task, all enhanced methods outperform the base model. The extension with external tool use achieves the best performance, ranking 1st on the leaderboard with a reward win rate of 98% and a leaderboard score of 0.98. Both SFT and RLOO also lead to substantial improvements over the base model. SFT achieves a 50% win rate, while RLOO attains a 73% win rate, a leaderboard score of 0.35, and a rank of 49. In contrast, the base model only achieves a 0% win rate and does not place on the leaderboard.

Methods	Reward	Win Rate	Leaderboard Score	Leaderboard Rank
Base	-27.52	-	-	-
With SFT	-23.13	50%	-	-
With DPO	<b>-21.07</b>	<b>72%</b>	<b>0.32</b>	<b>6</b>
With RLAIF	-21.52	69%	0.30	8

Table 1: **Quantitative evaluation on the instruction following task.** Our implementation achieves better performance than the base model. ‘-’ denotes not eligible.

Methods	Reward	Win Rate	Leaderboard Score	Leaderboard Rank
Base	0.01	0%	-	-
With SFT	0.23	50%	-	-
With RLOO	0.35	73%	0.35	60
With Tool	<b>0.99</b>	<b>99%</b>	<b>0.99</b>	<b>1</b>

Table 2: **Quantitative evaluation on the math reasoning task.** Our implementation achieves better performance than the base model. ‘-’ denotes not eligible.

Overall, these results demonstrate that our proposed methods consistently and substantially outperform the base model in both tasks, as reflected by improvements in reward, win rate, and leaderboard ranking.

### 5.2 Qualitative Analysis

To further assess our models beyond quantitative metrics, we present a qualitative comparison of model outputs in Figure 6. The two examples illustrate the progression from the base model to SFT and DPO. The base model either copies the input or provides minimal output. SFT improves the relevance and correctness of the response, while DPO leads to the most detailed, step-by-step answers that reflect both correctness and explainability. These qualitative results support our quantitative findings, showing that both SFT and DPO produce more helpful and interpretable outputs than the base model.

We observe a similar trend in the Countdown math reasoning task. The base model struggles to generate correct or even format-correct answers, while models trained with SFT and RLOO show clear improvements in accuracy and reasoning. Notably, when using the external tool extension, the model is not able to generate responses with the correct answer, but also with the correct reasoning steps, as shown in Appendix A. This demonstrates that combining reinforcement learning with tool augmentation enables the model to reason more reliably and transparently about complex

mathematical problems. It also provides a quicker and cheaper way to generate datasets like the Warmstart Dataset.

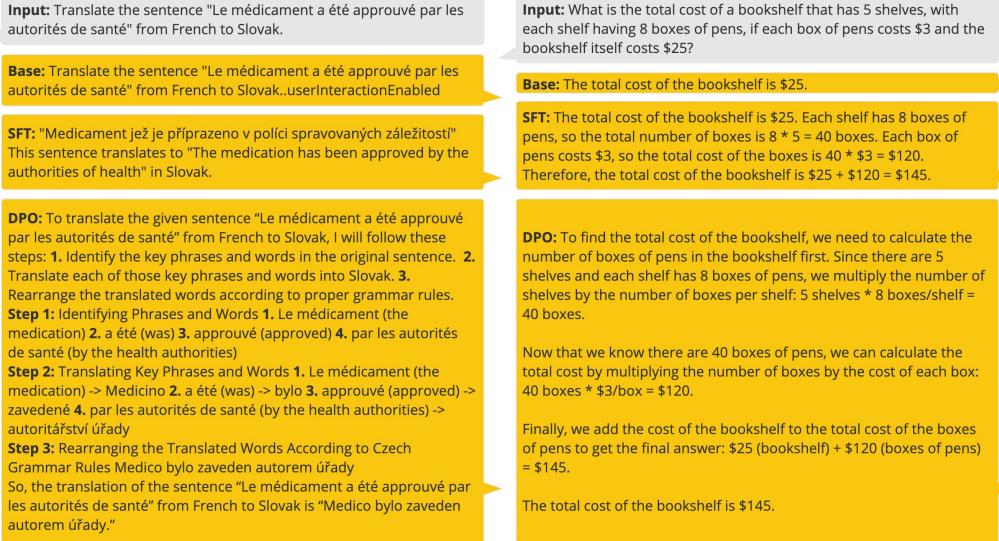


Figure 5: **Qualitative result of our model on the ultrafeedback dataset.** Result shows that models after SFT and DPO outperform the base model and the model after SFT, respectively.

## 6 Discussion

Our results show that reinforcement learning methods like SFT, DPO, and RLOO lead to clear improvements over the base model for both instruction following and math reasoning. The tool-use extension is particularly helpful for math problems, allowing the model to compute accurate answers and provide step-by-step explanations. RLAIF, which relies on AI-generated feedback, makes it easier to scale up data collection.

**"prompt":**"Q: 据报道，日本政府已决定将福岛核电站处理过的废水排入太平洋，这一决定引起了周边国家的强烈反对。\\n\\nA: It has been reported that the Japanese government has decided to release treated wastewater from the Fukushima nuclear power plant into the Pacific Ocean, a decision that has sparked strong opposition from neighboring countries."

**"prompt":**"[EX Q]: And surely I am with you always, to the very end of the age.\\n\\nA: جون که خدا جهان را اینقدر محبت نمود که پسر یگانه خود را داد تا هر که بر او ایمان آورد هلاک نگردد بلکه حیات جاودانی یابد:\\n\\n[EX Q]: \\n\\n[EX Q]: زیرا خداوند خدا آفتاب را برای حکومت روز و ماه و سtarگان را برای حکومت شب قرار داد." [EX A]:"

Figure 6: **Examples of out-of-distribution prompts.** Prompts in foreign languages or non-standard question formats tend to cause the model to generate low-quality responses.

In analyzing failure cases, we find that the instruction-following model struggles with prompts involving foreign languages. This is likely because most of the training data is in English. We also notice that some prompts in the held-out set appear to be out-of-distribution, which leads to lower-quality or off-topic responses. These limitations suggest that our model needs more diverse training data to better handle unfamiliar inputs. This can be done by incorporating additional external tools, such as translators, that could help the model generalize to multilingual scenarios.

## 7 Conclusion

In this project, we focus on improving large language model performance on instruction following and math reasoning tasks by exploring several reinforcement learning strategies. Our primary

implementation uses SFT, DPO, and RLOO, and these methods show clear improvements over the base model. We also explore two extensions: RLAIF, which replaces human feedback with scalable AI-generated preference signals, and external tool integration, which allows the model to leverage calculator functions for mathematical problem solving. Both approaches further enhance the performance of the base models on both tasks. Overall, our findings suggest that combining preference-based reinforcement learning with effective tool use can substantially improve model alignment and reasoning abilities. In future works, we plan to further investigate the generalization of RLAIF beyond our current tasks, refine tool integration for more complex applications, and explore the use of larger models capable of solving both tasks simultaneously.

## 8 Team Contributions

- **Feiyang Zhu:** REINFORCE Leave One-Out Dataset & Algorithm, Extension, Final Report
- **Shutong Zhang:** Preference Optimization Dataset & Algorithm, Extension, Final Report
- **Siyuan Wu:** Supervised Fine-tuning Dataset & Algorithm, Extension, Final Report

**Changes from Proposal** In addition to the proposed "Synthetic Data Augmentation" extension using RLAIF, we have also implemented the "Effective Tool Use" extension, specifically tailored for the math reasoning task.

## 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>

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajaráin, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. SmolLM2: When Smol Goes Big — Data-Centric Training of a Small Language Model. <https://doi.org/10.48550/arXiv.2502.02737> arXiv:2502.02737 [cs.CL]

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. UltraFeedback: Boosting Language Models with Scaled AI Feedback. arXiv:2310.01377 [cs.CL] <https://arxiv.org/abs/2310.01377>

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. 2025. Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective STaRs. <https://doi.org/10.48550/arXiv.2503.01307> arXiv:2503.01307 [cs.CL]

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneaux, Taco Cohen, and Gabriel Synnaeve. 2025. RLEF: Grounding Code LLMs in Execution Feedback with Reinforcement Learning. arXiv:2410.02089 [cs.CL] <https://arxiv.org/abs/2410.02089>

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2024. ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings. arXiv:2305.11554 [cs.CL] <https://arxiv.org/abs/2305.11554>

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. arXiv:2503.09516 [cs.CL] <https://arxiv.org/abs/2503.09516>

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2024. RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. arXiv:2309.00267 [cs.CL] <https://arxiv.org/abs/2309.00267>

Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. 2025. OctoTools: An Agentic Framework with Extensible Tools for Complex Reasoning. arXiv:2502.11271 [cs.LG] <https://arxiv.org/abs/2502.11271>

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>

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. 2025. TinyZero. <https://github.com/Jiayi-Pan/TinyZero>. Accessed: 2025-04-25.

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>

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv:2302.04761 [cs.CL] <https://arxiv.org/abs/2302.04761>

Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. 2024. HelpSteer2-Preference: Complementing Ratings with Preferences. arXiv:2410.01257 [cs.LG] <https://arxiv.org/abs/2410.01257>

## A Math Reasoning Task Examples

In this section, we present several example prompts of the cmath reasoning task.

### Model's Reasoning Example 1

**Numbers:** 4383, 2004, 1525

**Target:** 4862

**Response:**

<think>

Let me analyze this step by step:

1. First, let's look at the numbers: 4383, 2004, 1525 and target 4862.

2. Let's try working backwards from 4862:

- It's close to 4383, so maybe we can add something
- We need to add 482 to 4383

3. Looking at 2004 and 1525:

- $2004 - 1525 = 479$
- This is close to 482!

4. Let's try:

$$4383 + (2004 - 1525) = 4383 + 479 = 4862$$

5. Verifying:

- Second step:  $2004 - 1525 = 479$
- Final step:  $4383 + 479 = 4862$

This works! We've added all the numbers correctly.

</think>

<answer>4383 + 2004 - 1525</answer>

## Model's Reasoning Example 2

**Numbers:** 133, 818, 807

**Target:** 144

**Response:**

<think>

Let me analyze this step by step:

1. First, let's look at factors of 144:

$$144 = 2^4 * 3$$

Factors include: 1, 2, 3, 4, 6, 8, 9, 12, 16, 24, 32, 48, 144

2. Looking at the numbers:

133, 818, 807

These are all relatively large numbers.

3. Let's try to get close to 144:

$$818 - 807 = 11$$

$$133 + 11 = 144$$

4. This works! We can reach 144 by:

First subtract 807 from 818 to get 11

Then add 133 to get 144

</think>

<answer>133 + 818 - 807</answer>

## Model's Reasoning Example 3

**Numbers:** 82, 5, 10

**Target:** 32

**Response:**

<think>

Let me try to reach 32 using these numbers.

First, let me see if I can work backwards from 32.

32 is  $8 * 4$ , so maybe I can try to get 8 first.

82 is much larger than what I need, but maybe I can divide it.

$82/10$  would be 8.2, not helpful.

Let me try a different approach:

$$5 * 10 = 50$$

$$82 - 50 = 32!$$

And I can put this result back into the expression...

So working forward:

1. First get 50:  $5 * 10 = 50$

2. Then subtract from 82:  $82 - 50 = 32$

</think>

<answer>82 - 5 \* 10</answer>