Transactions of the Korean Nuclear Society Spring Meeting
Jeju, Korea, May 22-23, 2025

Enhancing Mathematical Reasoning Ability of Nuclear Domain-Specific Large Language Model using Reinforcement Learning

Byeongjae Kim[a], Solji Park[a], Yonggyun Yu[b,c,*]
[a]Korea University of Technology and Education (KOREATECH), 1600, Chungjeol-ro, Byeongcheon-myeon, Dongnam-gu, Cheonan-si, Chungcheongnam-do, Republic of Korea
[b]Artificial Intelligence Application Laboratory, Korea Atomic Energy Research Institute (KAERI), 111 Daedeok-daero 989 beon-gil, Yuseong-gu, Daejeon, 34057, Republic of Korea
[c] Korea National University of Science and Technology (UST), 217Gajeong-ro, Yuseong-gu, Daejeon, 34113, Republic of Korea
*Corresponding author: *ygyu@kaeri.re.kr*

***** Keywords: Large Language Model(LLM), Group Relative Policy Optimization(GRPO), Reinforcement Learning(RL), Mathematical reasoning, Nuclear engineering, Artificial Intelligence

1. Introduction

Large-scale language models (LLMs) have recently seen widespread development and application across various domains. Enhancing their reasoning capabilities is particularly valuable, as it enables them to assist in complex tasks such as operating nuclear reactor simulators, especially during critical scenarios like severe nuclear accidents. Among the recently developed LLMs, DeepSeek-R1 by DeepSeek-AI has demonstrated promising performance.

This model achieves high inference performance comparable to OpenAI's o1-1217 model [1] while requiring fewer GPU resources. One key factor contributing to DeepSeek-R1's efficiency is its adoption of Group Relative Policy Optimization (GRPO).

In this study, we apply the GRPO reinforcement learning method to improve the mathematical reasoning skills of AtomicGPT, a nuclear-domain-specific model developed by KAERI. Instead of traditional supervised fine-tuning methods, our approach combines reinforcement learning with LoRA(Low-Rank Adaptation). Specifically, we use GRPO, which does not require a separate critic model, making the training process more efficient than other methods like Proximal Policy Optimization (PPO).

The following sections present the GRPO training process, benchmark results, and a discussion of its effectiveness and implications. Through this research, we will explore the potential of leveraging LLMs for the autonomous operation of virtual reactor simulators.

2. Knowledge of GRPO and PPO

2.1 PPO (Proximal Policy Optimization)

Proximal Policy Optimization (PPO) is one of the most widely used methods for policy optimization in reinforcement learning. PPO is a policy-based algorithm that updates policies directly, allowing for more reliable learning than traditional policy gradient methods. To do this, PPO uses a surrogate objective function that updates the policy while maintaining a confidence region. [2]

However, there are some drawbacks to PPO. First, it requires a value function that is typically implemented as another model of comparable size as the policy model, which brings a substantial memory and computational burden. Second, during RL training, the value function is treated as a baseline in the calculation of the advantage for variance reduction. However, in the LLM context, usually only the last token is assigned a reward score by the reward model, which may complicate the training of a value function that is accurate at each token. [2]

2.2 GRPO (Group Relative Policy Optimization)

To address the shortcomings of PPO, we apply Group Relative Policy Optimization (GRPO). GRPO retains the core concept of PPO but optimizes policies through sample comparisons within a group without relying on a value function. Specifically, it generates multiple samples per prompt and updates the policy based on the highest-scoring output. Samples are collected for each question, after which rewards are assigned using a reward model, and policies are adjusted based on the scores. [2]

Consequently, GRPO requires fewer GPUs by training with a single model, offering significant advantages. It simplifies the learning process and accelerates convergence by eliminating the need for a separate value network, and it reduces VRAM usage, conserving GPU resources, which is particularly beneficial for training large language models (LLMs). GRPO also employs relative reward-based optimization, allowing effective policy updates in scenarios where defining absolute rewards is difficult. Additionally, GRPO achieves faster convergence and ensures more stable learning than traditional PPO.

3. The process of creating a model and evaluation

3.1 Model configuration

In this experiment, we used an A100 GPU. We used the opensource code provided by unsloth for loading the basic model and the reward function. [7] In the following, We will explain what we did differently from the code provided by unsloth.

To enhance our model's performance and efficiency, we implemented several configurations. We set the *max_sequence_length* to 2048 to support long contextual inference. For better memory efficiency, we switched from using *float* to *bfloat16* In preparation for constructing the model with LoRA, we configured additional parameters: *dropout*, *bias* parameter and *task_type*. To prevent overfitting, we applied a dropout value of 0.1 to the LoRA layer. We set the model's *bias* parameter to none to enhance its generalization ability. Additionally, we specified the *task_type* as *"CASUAL_LM"* to optimize the model for causal text generation tasks, which aids in predicting and generating contextual text effectively.

We configured the basic model and modified the config for GRPO training: we increased the batch size to improve training efficiency and specified *max_steps* and *save_steps* to find the optimal training state. After completing these settings, we loaded the model and continued with the next task.

3.2 Reward Algorithm

For effective GRPO reinforcement learning, it is important to carefully design the reward function. While PPO utilizes a separate value function (critic model) to evaluate the model output, GRPO updates the policy based on relative ranking alone, so the "reward" itself directly defines the objective the model should optimize for.

We modified the reward function based on the one provided by unsloth [7]. Since GRPO reinforcement learning does not use a critic model, we needed to implement an accurate answer extraction and comparison algorithm to correctly assign rewards. However, the initial algorithm was insufficient, as it failed to properly extract the answer in many cases, even though the correct answer was present in the solution process.

In Table I illustrates cases in the training process logs where the correct answer exists but is not extracted, resulting in missing rewards. Table I explains the reward function values for these cases; since the correct answer is not extracted, no reward is assigned, leading to a reward value of zero. The reward values shown in Table II include various functions: *correctness_reward_func*, *int_reward_func*, *soft_format_func*, *strict_format_func*, and *xmlcount_reward_func*, all of which output a reward of zero due to the failure in answer extraction.

```
---------------------- Step 3
Question:
Janet pays $40/hour for 3 hours per week of clarinet
lessons and $28/hour for 5 hours a week of piano
lessons. How much more does she spend on piano
lessons than clarinet lessons in a year?
Answer:
1040
Response:
Let's break down Janet's weekly expenses for both
lessons.

Clarinet lessons cost $40/hour for 3 hours/week, so
the weekly expense for clarinet lessons is $40 * 3 =
$120.
Piano lessons cost $28/hour for 5 hours/week, so the
weekly expense for piano lessons is $28 * 5 = $140.

The difference in weekly expenses between piano
and clarinet lessons is $140 - $120 = $20.

Since there are 52 weeks in a year, the annual
difference in expenses is $20 * 52 = $1040.

<answer> Janet spends $1040 more on piano lessons
than clarinet lessons per year. </answer>
Extracted:
Janet spends $1040 more on piano lessons than
clarinet lessons per year.
```

Table I: GRPO training logs showing missed rewards due to unextracted correct answers before modifications in the reward function.

```
"reward": -0.0698333826065063,
"reward_std": 0.3100970387458801,
"rewards/correctness_reward_func": 0.0,
"rewards/int_reward_func": 0.0,
"rewards/soft_format_reward_func": 0.0,
"rewards/strict_format_reward_func": 0.0,
"rewards/xmlcount_reward_func":
0.0698333826065063,
"step": 3
```

Table II: GRPO training reward function results before modifications in the reward function.

To address the challenges in answer extraction, we implemented a new function for accurately extracting the correct answer. Initially, we enhanced the algorithm to reliably extract numbers if they appear in XML format. Additionally, we refined the extraction process to handle instances where the correct answer exists within the solution process but is not provided in XML format. We further developed a pattern to identify and extract answers embedded in concluding sentences (e.g., phrases like "Therefore," "As a result," etc.) that were previously not captured.

Following these modifications, Table III presents the updated training logs, demonstrating successful answer extraction. Table Ⅳ illustrate the results of the reward function, showing increased reward values. In TableⅣ, rewards are correctly assigned upon successful answer extraction, and each function's reward values *correctness_reward_func*, *int_reward_func*, *soft_format_func*, *strict_format_func*, *xmlcount_reward_func*) are significantly higher compared to the values in Table II

Step 13
Question:
Janelle had 26 green marbles. Then she bought 6 bags of blue marbles. There were 10 marbles in each bag. She created a gift of 6 green marbles and 8 blue marbles and gave it to a friend. How many marbles does Janelle have now?
Expected Answer:
72
Full Response:
To find the total number of blue marbles Janelle bought, multiply the number of blue marbles per bag by the number of bags:
10 marbles/bag * 6 bags = 60 blue marbles.

The total number of marbles Janelle had initially was 26 green marbles + 60 blue marbles = 86 marbles.

After Janelle gave her friend a gift of 6 green marbles and 8 blue marbles, she had 86 - 6 - 8 = 72 marbles left.
Extracted Answer:
After Janelle gave her friend a gift of 6 green marbles and 8 blue marbles, she had 86 - 6 - 8 = 72 marbles left
Expected Number: 72.0, Extracted Number: 72.0
✓ CORRECT! Reward: 1.5

Table III: GRPO training logs showing successful extraction of correct answers and assignment of rewards after modifications in the reward function.

"reward": 2.041666667,
"reward_std": 0.5931,
"rewards/correctness_reward_func": 1.5,
"rewards/int_reward_func": 0.5,
"rewards/soft_format_reward_func": 0.0,
"rewards/strict_format_reward_func": 0.0,
"rewards/xmlcount_reward_func": 0.0416666679084301,
"step": 13

Table Ⅳ : GRPO training reward function results after modifications in the reward function.

This improvement in the reward extraction process allowed us to train the model more efficiently, using fewer GPU resources without the need for a critic model.

3.3 GRPO model development

One challenge in the GRPO model development was determining the optimal point during training to finalize the model. To address this, we continuously monitored the model's average reward to identify when it achieved its peak performance. The average reward was calculated by dividing the total accumulated reward by the number of training steps. This value served as a key indicator of the model's learning quality and was used to compare performance across different points in training. Step 750 showed the highest average reward, with a value of 2.6605. Based on this result, we selected the model from this step for final use, as it represented the most effective policy learned during training.

.

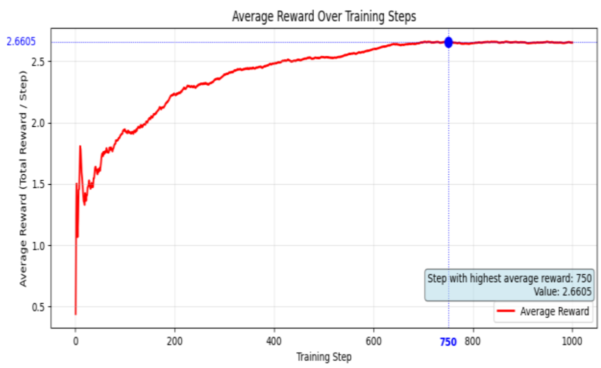

Fig. 1. illustrates the trend of average reward across training steps, with the maximum value observed at step 750.

3.4 Training and Evaluation Datasets

We used the gsm8k [6] dataset for training and the Aqua-rat [5] and Math-500 [4] datasets for performance evaluation. gsm8k consists of elementary-level math reasoning problems from the U.S. while Aqua-rat also contains elementary-level math problems from the US. Finally, Math-500 consists of US high school-level questions. Since gsm8k is composed of questions and answers, it is well-suited for enhancing reasoning ability through GRPO training.

To assess performance improvements, we evaluated the model on Aqua-rat, a dataset of similar difficulty. Finally, we will rigorously test reasoning improvements using Math-500, which consists of high-difficulty problems.

Since the solution process for Math-500 is complex and may involve multiple steps, extracting the exact answer can be difficult, particularly when there are variations in notation (such as LaTeX).

Aqua-rat, which consists of relatively simple problems, was constructed with the same answer extraction algorithm used for training to assess math reasoning skills. For both datasets, the evaluation metric was pass@1

3.5 benchmarking result

To address this, we utilized the Gemini-1.5 Pro [3] model. Gemini-1.5 Pro was used to evaluate the reasoning process and verify whether the correct answer was present. Even if the answer was expressed differently or not directly extracted, Gemini-1.5 Pro identified these as correct. This approach helped ensure that answers hidden within the solution process were correctly recognized and accounted for in the evaluation.

As shown in Fig. 2. and Table V, the original AtomicGPT scored 41 on Math-500 and the AtomicGPT with GRPO RL scored 47. ("Gemini-1.5 pro" checked the correct answer.) The More difficult math problems require greater reasoning ability, and the model with better math reasoning skills scored higher. On the Aqua-rat dataset, which is similar to the training data, the traditional AtomicGPT scored 51.5 and the model with GRPO RL achieved a higher score of 53.96.

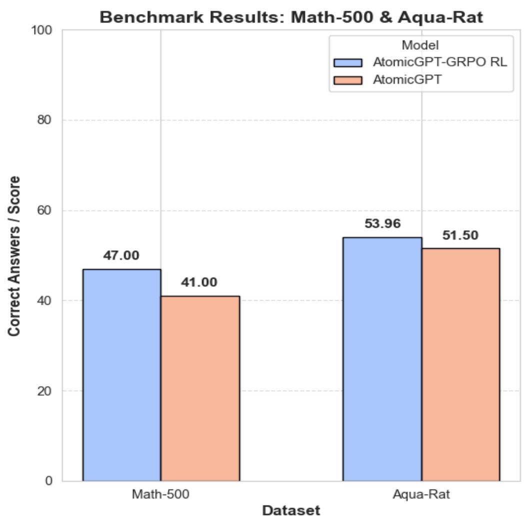

Fig. 2. Benchmarks of AtomicGPT- GRPO RL and AtomicGPT with Math-500 (100 questions with Gemini-1.5 pro to evaluate the correct answers), Aqua-rat dataset

	AtomicGPT-GRPO RL	AtomicGPT
Math-500 (pass@1)	47	41
Aqua-rat (pass@1)	53.96	51.5

Table V: AtomicGPT-GRPO RL, AtomicGPT benchmarks performed with Math-500 and Aqua-rat as evaluation datasets utilizing the pass@1 metric.

4. Conclusions

In conclusion, GRPO has been shown to improve the mathematical reasoning capabilities of LLMs while using fewer resources and datasets**.** These results suggest that the GRPO method can enhance the reasoning ability of LLMs, and if this improvement extends to fields beyond mathematics, it could be applied to controlling virtual reactor simulators.

If AtomicGPT, which is specialized in nuclear energy, enhances its reasoning ability for controlling a virtual reactor simulator, it will be able to handle more complex processes and perform delicate tasks more efficiently, leveraging its deeper knowledge of nuclear energy compared to general LLMs.

REFERENCES

[1] D. Guo, D. Yang, H. Zhang, et al, "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcemnt Learning," DeepSeek-AI, Jan. 2025.
[2] Z. Shao,P. Wang, Q. Zhu, et al, "DeepSeekMath: Pushing the Limits f Mathematical Reasoning in Open Language Models," DepSeek-AI, 2024.
[3] Google, Gemini 1.5 pro (gemini-1.5-pro), July 2024. URL https://cloud.google.com/vertex-ai/generative-ai/docs.
[4] HuggingFaceH4, 2024, "MATH-500," Hugging Face. [Online]. https://huggingface.co/datasets/HuggingFaceH4/MATH-500/tree/main.
[5] IdeaCode, 2025, "aquarat", Hugging Face. [Online]. Available: https://huggingface.co/datasets/ideacode/aquarat/tree/main.
[6] OpenAI, 2021, "gsm8k", Hugging Face. [Online]. Available: https://huggingface.co/datasets/openai/gsm8k.
[7] unsloth, 2024, "Llama3.1 (8B)-GRPO.ipynb", GitHub. [Online]. Available: https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/Llama3.1_(8B)-GRPO.ipynb.