DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

Dooyoung Kim
Natural Language Processing Lab, SKKU

성균관대학교
SUNG KYUN KWAN UNIVERSITY(SKKU)

Contents

DeepSeekMath Training Pipeline

- **Pretraining**
  - **일반적인 Language Modeling 능력을 학습**

- **Math Domain Pretraining**
  - **수학 데이터셋을 구축하는 파이프라인 제시**
    - Seed 수학 데이터 구축 -> FastText Model 학습 -> Common Crawl 데이터셋에서 수학 관련 페이지 추출 -> 수학 관련 도메인 탐지 -> 수학 관련 URL을 Tagging -> Seed 수학 데이터 구축

- **SFT (Instruction tuning)**
  - **수학 문제 + 풀이 데이터로 instruction tuning**

- **RL (GRPO)**
  - **추론 능력 강화를 위한 강화학습 수행**
    - PPO는 LLM에서 잘 동작하지 않음
    - 이를 해결하기 위한 새로운 알고리즘 제안

DeepSeekMath Training Pipeline

Pretraining (General + Math Domain)

- High Quality 사전학습 데이터셋 구축 (Table 1)
- 다양한 수학 벤치마크에서 좋은 성능을 달성 (Table 2)
- 자연어 이해 능력 역시 준수한 성능을 달성 (Table 4)

Math Corpus	Size	English Benchmarks					Chinese Benchmarks		
		GSM8K	MATH	OCW	SAT	MMLU STEM	CMATH	Gaokao MathCloze	Gaokao MathQA
No Math Training	N/A	2.9%	3.0%	2.9%	15.6%	19.5%	12.3%	0.8%	17.9%
MathPile	8.9B	2.7%	3.3%	2.2%	12.5%	15.7%	1.2%	0.0%	2.8%
OpenWebMath	13.6B	11.5%	8.9%	3.7%	31.3%	29.6%	16.8%	0.0%	14.2%
Proof-Pile-2	51.9B	14.3%	11.2%	3.7%	43.8%	29.2%	19.9%	5.1%	11.7%
DeepSeekMath Corpus	**120.2B**	**23.8%**	**13.6%**	**4.8%**	**56.3%**	**33.1%**	**41.5%**	**5.9%**	**23.6%**

Table 1 | Performance of DeepSeek-LLM 1.3B trained on different mathematical corpora, evaluated using few-shot chain-of-thought prompting. Corpus sizes are calculated using our tokenizer with a vocabulary size of 100K.

Model	Size	English Benchmarks					Chinese Benchmarks		
		GSM8K	MATH	OCW	SAT	MMLU STEM	CMATH	Gaokao MathCloze	Gaokao MathQA
Closed-Source Base Model									
Minerva	7B	16.2%	14.1%	7.7%	-	35.6%	-	-	-
Minerva	62B	52.4%	27.6%	12.0%	-	53.9%	-	-	-
Minerva	540B	58.8%	33.6%	17.6%	-	63.9%	-	-	-
Open-Source Base Model									
Mistral	7B	40.3%	14.3%	9.2%	71.9%	51.1%	44.9%	5.1%	23.4%
Llemma	7B	37.4%	18.1%	6.3%	59.4%	43.1%	43.4%	11.9%	23.6%
Llemma	34B	54.0%	25.3%	10.3%	71.9%	52.9%	56.1%	11.9%	26.2%
DeepSeekMath-Base	7B	**64.2%**	**36.2%**	**15.4%**	**84.4%**	**56.5%**	**71.7%**	**20.3%**	**35.3%**

Table 2 | Comparisons between DeepSeekMath-Base 7B and strong base models on English and Chinese mathematical benchmarks. Models are evaluated with chain-of-thought prompting. Minerva results are quoted from Lewkowycz et al. (2022a).

Model	Size	MMLU	BBH	HumanEval (Pass@1)	MBPP (Pass@1)
Mistral	7B	**62.4%**	55.7%	28.0%	41.4%
DeepSeek-Coder-Base-v1.5†	7B	42.9%	42.9%	40.2%	52.6%
DeepSeek-Coder-Base-v1.5	7B	49.1%	55.2%	**43.2%**	**60.4%**
DeepSeekMath-Base	7B	54.9%	**59.5%**	40.9%	52.6%

Table 4 | Evaluation on natural language understanding, reasoning, and code benchmarks. DeepSeek-Coder-Base-v1.5† is the checkpoint right before learning rate decay, which is used to train DeepSeekMath-Base. On MMLU and BBH, we use few-shot chain-of-thought prompting. On HumanEval and MBPP, we evaluate model performance under the zero-shot setting and a few-shot setting, respectively.

Policy Optimization

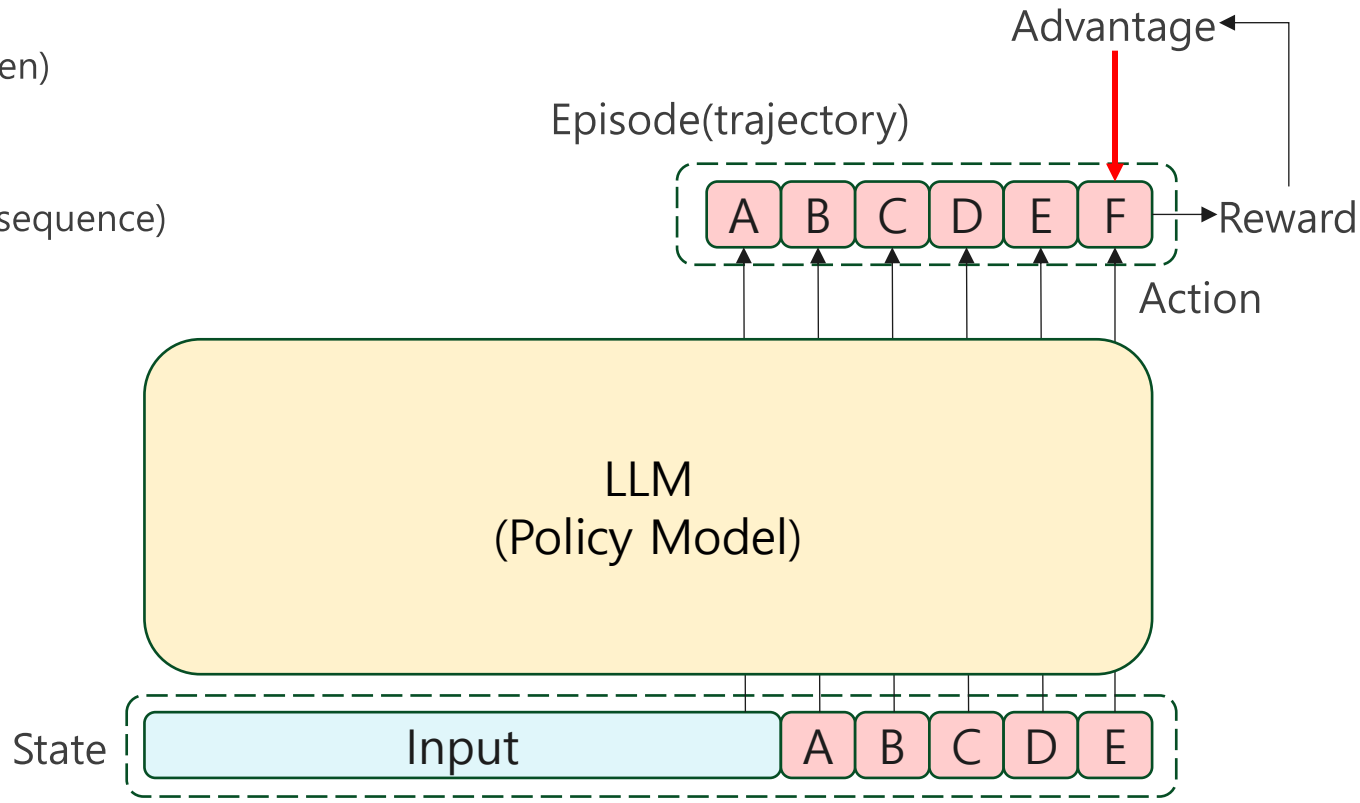
Policy Optimization in LLM

Definition

- π : Policy (LLM)
- s_t : State (Input prompt + 지금까지 생성한 token)
- a_t : Action (다음 token 생성)
- $\hat{A}_{i,t}$: Advantage (Action에 대한 보정된 reward)
- τ_i *or* o_i : Episode or Trajectory (생성된 output sequence)
- r_t : Reward (정답 여부 or 선호도 등)

RL Framework

1. 특정 Trajectory를 생성
2. Trajectory에 대한 Reward 계산
3. 생성된 각 token에 대해서 Advantage를 계산
4. 계산된 Advantage를 가중치로 하여 학습



Proximal Policy Optimization Algorithms (PPO)

Weakness of PPO

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}$$
$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

reward Value Model

Value Model의 필요성

- PPO 및 기존의 RL 방법론은 **Policy Model과 Value Model**이 필요함
- Policy Model = 실제 학습하고자 하는 모델
- Value Model = Policy 모델의 학습에 필요한 보조 모델 (Advantage 계산)
- 일반적으로 **Value Model은 Policy Model과 비슷한 크기**를 가짐 (메모리 비효율적)

Reward의 구조적 문제

- 기존의 강화학습 알고리즘은 action sequence 중간에 부분적인 reward가 발생하는 상황을 가정
- 일반적인 LLM에서는 **reward가 output이 완성되는 순간에 집중**됨 (정답 영역)
- Value Model은 중간 step에 대한 reward 값을 마지막 토큰의 reward를 기반으로 예측 (학습 난이도)

Group Relative Policy Optimization (GRPO)

- **GRPO**
  - **Value Model 제거**
  - **Sampling된 group의 평균 점수를 baseline으로 사용**

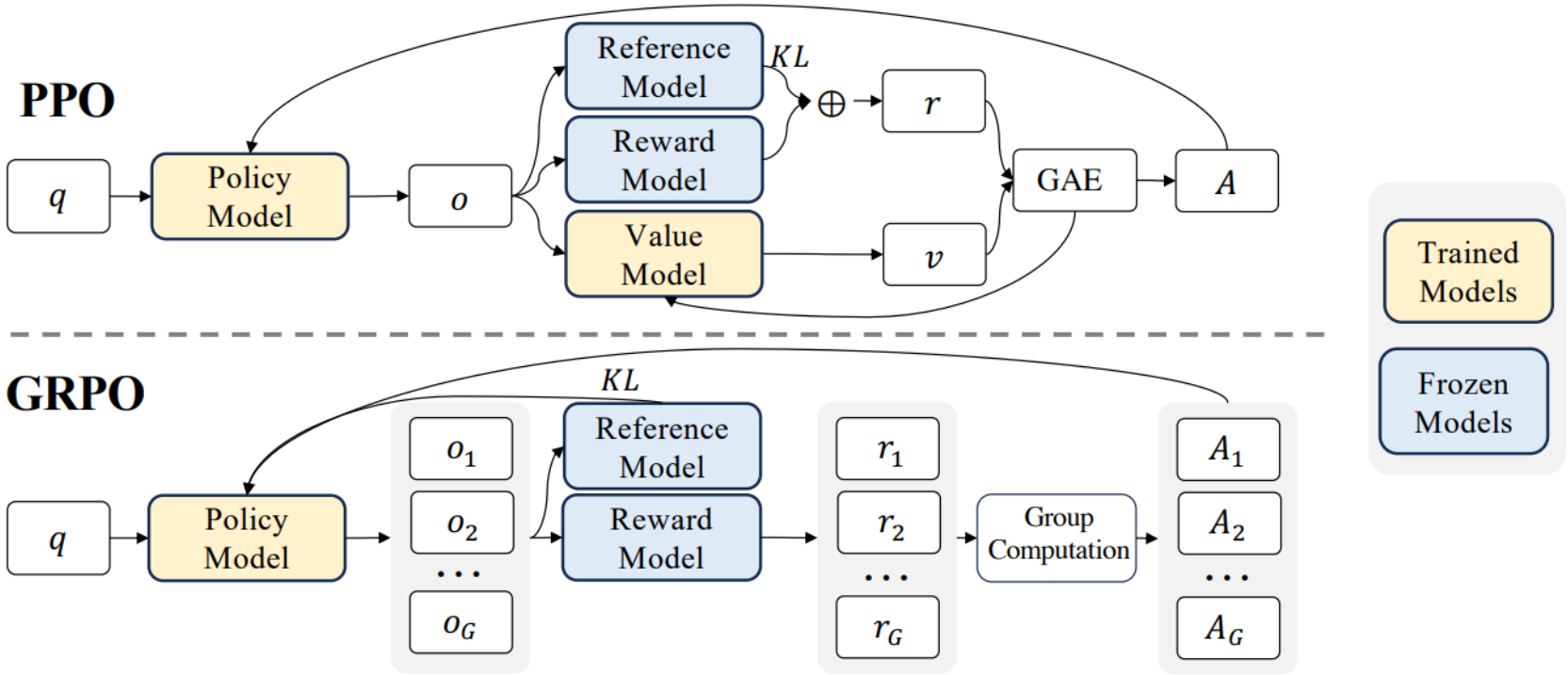


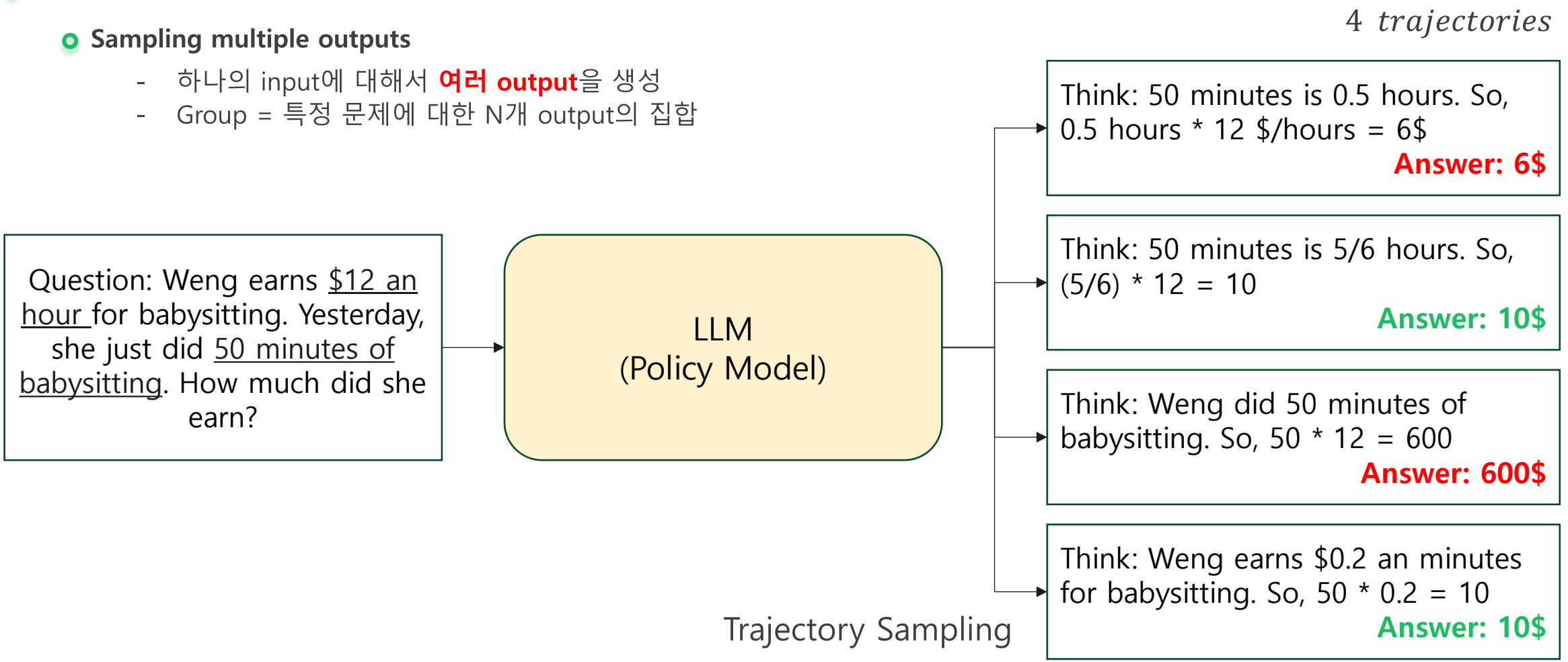
Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

Group Relative Policy Optimization (GRPO)

GRPO

Sampling multiple outputs

- 하나의 input에 대해서 **여러 output**을 생성
- Group = 특정 문제에 대한 N개 output의 집합



Group Relative Policy Optimization (GRPO)

GRPO

Calculate Advantage

- 각 Trajectory에 대한 Reward를 계산
- Group 내 reward를 기반으로 Advantage 계산 (**group 내 reward 정규화**)

$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)}$$

Think: 50 minutes is 0.5 hours. So, 0.5 hours * 12 $/hours = 6$
Answer: 6$

Think: 50 minutes is 5/6 hours. So, (5/6) * 12 = 10
Answer: 10$

Think: Weng did 50 minutes of babysitting. So, 50 * 12 = 600
Answer: 600$

Think: Weng earns $0.2 an minutes for babysitting. So, 50 * 0.2 = 10
Answer: 10$

$r_0 = 0$
$r_1 = 1$
$r_2 = 0$
$r_3 = 1$

$\text{mean}(r) = 0.5$
$\text{std}(r) = 0.5$

$\hat{A}_{0,t} = -1$
$\hat{A}_{1,t} = 1$
$\hat{A}_{2,t} = -1$
$\hat{A}_{3,t} = 1$

Group Relative Policy Optimization (GRPO)

GRPO

Objective Function

- PPO에서 사용한 Clipping + min 함수 구조를 사용
- KL divergence term을 reward와 분리
  - InstructGPT 등 일반적으로 LLM의 PPO에서 reward에 KL divergence term을 추가
  - Reward signal 왜곡
  - 복잡한 advantage 계산 과정

$$\mathcal{J}_{GRPO} = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{old}}(O|q)} \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min[r_t(\theta)\hat{A}_{i,t}, clip(r_t(\theta),1-\epsilon,1+\epsilon)\hat{A}_{i,t}] - \beta\mathbb{D}_{KL}[\pi_\theta||\pi_{ref}]\right\}$$

PPO의 목적 함수와 동일 (Advantage 계산이 다름)

질문 q에 대해서

output group O을 샘플링 (G개)

Token별 평균

Unbiased KL divergence estimator (Schulman, 2020)

- unbiased / positive / low variance

$$where \quad \hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)},$$

$$\mathbb{D}_{KL}[\pi_\theta||\pi_{ref}] = \frac{\pi_{ref}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})} - \log\frac{\pi_{ref}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})} - 1$$

Experiments Setting

Dataset

SFT

- GSM8K + MATH (en) / Chinese K-12 (ch)

RL

- Policy Model: GSM8K and MATH
- Reward Model: (Wang et al,, 2023)

Model

DeepSeekMath-Instruct 7B, DeepSeekMath-Instruct 1.3B

Hyperparameters

General hyperparameters

- Learning rate: 1e-6 / max_length: 1024 / training batch size: 1024

GRPO hyperparameters

- 64 samples / KL coefficient β: 0.04

Code Affects Mathematical Reasoning

Mathematical reasoning under different training settings

Training Setting	Training Tokens			w/o Tool Use			w/ Tool Use	
	General	Code	Math	GSM8K	MATH	CMATH	GSM8K+Python	MATH+Python
No Continual Training	–	–	–	2.9%	3.0%	12.3%	2.7%	2.3%
Two-Stage Training								
Stage 1: General Training	400B	–	–	2.9%	3.2%	14.8%	3.3%	2.3%
Stage 2: Math Training	–	–	150B	19.1%	14.4%	37.2%	14.3%	6.7%
Stage 1: Code Training	–	400B	–	5.9%	3.6%	19.9%	12.4%	10.0%
Stage 2: Math Training	–	–	150B	**21.9%**	**15.3%**	**39.7%**	17.4%	9.4%
One-Stage Training								
Math Training	–	–	150B	20.5%	13.1%	37.6%	11.4%	6.5%
Code & Math Mixed Training	–	400B	150B	17.6%	12.1%	36.3%	**19.7%**	**13.5%**

Code vs Normal Text

Training method

Effect of Code data

Table 6 | Investigation of how code affects mathematical reasoning under different training settings. We experiment with DeepSeek-LLM 1.3B, and evaluate its mathematical reasoning performance without and with tool use via few-shot chain-of-thought prompting and few-shot program-of-thought prompting, respectively.

Code & Math Training Affects Model Capability

- **Language understanding, reasoning, coding under different training settings**

Training Setting	Training Tokens			MMLU	BBH	HumanEval (Pass@1)	MBPP (Pass@1)
	General	Code	Math				
No Continual Training	–	–	–	24.5%	28.1%	12.2%	13.0%
Two-Stage Training							
Stage 1: General Training	400B	–	–	25.9%	27.7%	15.2%	13.6%
Stage 2: Math Training	–	–	150B	33.1%	32.7%	12.8%	13.2%
Stage 1: Code Training	–	400B	–	25.0%	31.5%	25.0%	**40.0%**
Stage 2: Math Training	–	–	150B	**36.2%**	35.3%	12.2%	17.0%
One-Stage Training							
Math Training	–	–	150B	32.3%	32.5%	11.6%	13.2%
Code & Math Mixed Training	–	400B	150B	33.5%	**35.6%**	**29.3%**	39.4%

Code vs Normal Text

Training method

Effect of Code data

Table 7 | Investigation of how different settings of code and math training affect model performance of language understanding, reasoning, and coding. We experiment with DeepSeek-LLM 1.3B. We evaluate the models on MMLU and BBH using few-shot chain-of-thought prompting. On HumanEval and MBPP, we conduct zero-shot and few-shot evaluations, respectively.

Performance of Open- and Closed-Source models

- **Chain-of-Thought (CoT) Reasoning**
  - Reasoning + Answer
- **Tool-Integrated Reasoning**
  - Using Tool ex) python interpreter

Model	Size	English Benchmarks		Chinese Benchmarks	
		GSM8K	MATH	MGSM-zh	CMATH
Tool-Integrated Reasoning					
Closed-Source Model					
GPT-4 Code Interpreter	-	97.0%	69.7%	-	-
Open-Source Model					
InternLM2-Math	20B	80.7%	54.3%	-	-
DeepSeek-LLM-Chat	67B	86.7%	51.1%	76.4%	85.4%
ToRA	34B	80.7%	50.8%	41.2%	53.4%
MAmmoTH	70B	76.9%	41.8%	-	-
DeepSeekMath-Instruct	7B	83.7%	57.4%	72.0%	84.3%
DeepSeekMath-RL	7B	86.7%	58.8%	78.4%	87.6%

Model	Size	English Benchmarks		Chinese Benchmarks	
		GSM8K	MATH	MGSM-zh	CMATH
Chain-of-Thought Reasoning					
Closed-Source Model					
Gemini Ultra	-	94.4%	53.2%	-	-
GPT-4	-	92.0%	52.9%	-	86.0%
Inflection-2	-	81.4%	34.8%	-	-
GPT-3.5	-	80.8%	34.1%	-	73.8%
Gemini Pro	-	86.5%	32.6%	-	-
Grok-1	-	62.9%	23.9%	-	-
Baichuan-3	-	88.2%	49.2%	-	-
GLM-4	-	87.6%	47.9%	-	-
Open-Source Model					
InternLM2-Math	20B	82.6%	37.7%	-	-
Qwen	72B	78.9%	35.2%	-	-
Math-Shepherd-Mistral	7B	84.1%	33.0%	-	-
WizardMath-v1.1	7B	83.2%	33.0%	-	-
DeepSeek-LLM-Chat	67B	84.1%	32.6%	74.0%	80.3%
MetaMath	70B	82.3%	26.6%	66.4%	70.9%
SeaLLM-v2	7B	78.2%	27.5%	64.8%	-
ChatGLM3	6B	72.3%	25.7%	-	-
WizardMath-v1.0	70B	81.6%	22.7%	64.8%	65.4%
DeepSeekMath-Instruct	7B	82.9%	46.8%	73.2%	84.6%
DeepSeekMath-RL	7B	88.2%	51.7%	79.6%	88.8%

RFT vs GRPO

- **Rejection Sampling Fine-tuning (RFT)**
  - 여러 output을 생성하여, 정답인 trajectory만 학습 (틀린 시퀀스 가중치 = 0)
- **OS (only 정답) vs PS (풀이 과정)**
  - PS가 step-wise로 reward를 계산하기 때문에 더 유의미한 학습 signal을 제공 (credit assignment)
- **Offline (고정된 샘플) vs Online (매번 샘플링)**
  - Online RL이 현재 모델 분포에서 샘플을 생성하므로 더 유의미한 학습 signal 제공 (Reward 모델 역시 업데이트 필요 Figure 6)

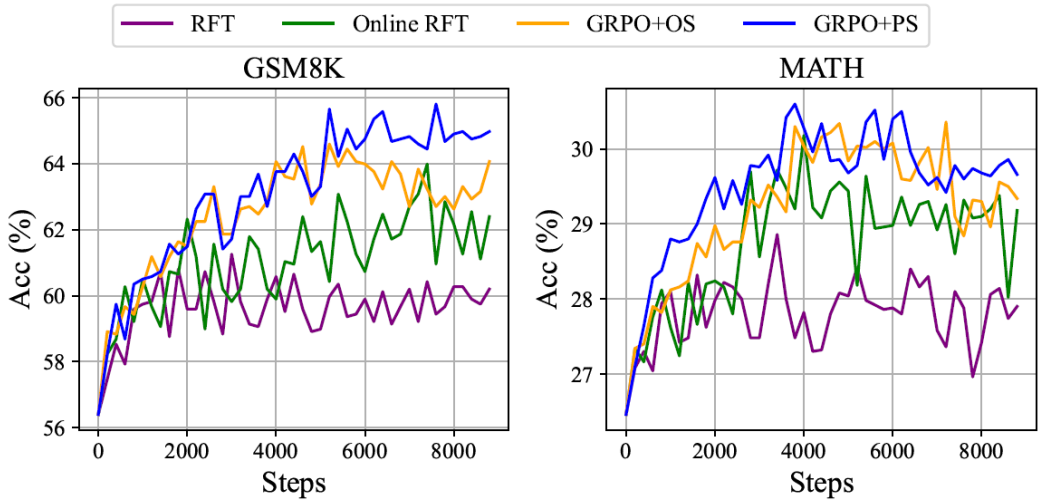


Figure 5 | Performance of the DeepSeekMath-Instruct 1.3B model, which was further trained using various methods, on two benchmarks.

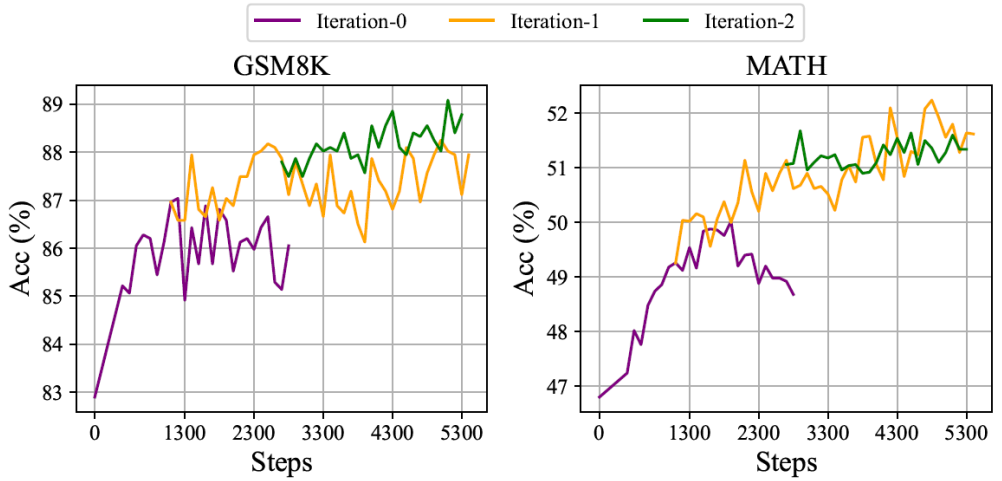
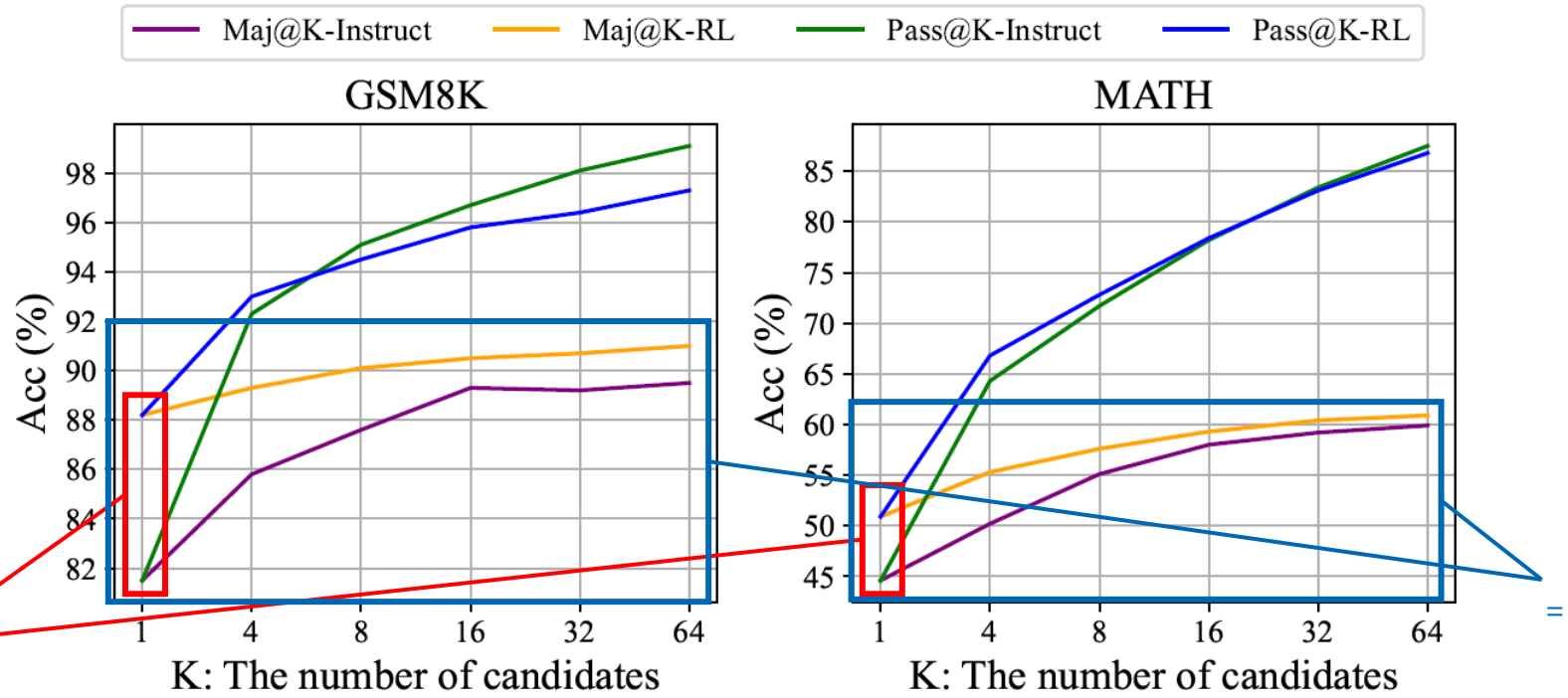


Figure 6 | Performance of iterative reinforcement learning with DeepSeekMath-Instruct 7B on two benchmarks.

RL is Mass Reallocation

- **Pass@K vs Maj@K**
  - **Pass@K**: K개의 output 중에서 정답이 있는가?
  - **Maj@K**: K개의 output 중에서 Majority한 답이 정답인가?

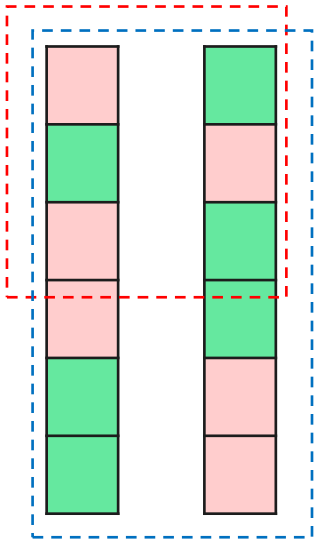


Pass@1에서 RL > SFT
=> 정답의 확률을 올린다

ΔMaj@K
=> 정답들이 높은 rank에 있다

Ranking A
Maj@3 = 0
Maj@6 = 1

Ranking B
Maj@3 = 1
Maj@6 = 1



<ΔMaj@K의 의미>

Figure 7 | The Maj@K and Pass@K of SFT and RL DeepSeekMath 7B on GSM8K and MATH (temperature 0.7). It was noted that RL enhances Maj@K but not Pass@K.

Conclusion

- **Code + Math Domain에서의 사전학습 필요성**

- **GRPO 알고리즘을 제안**
  - PPO의 Value 모델 제거
  - 여러 output을 Sampling하여 Group 구성, 그룹 내 리워드를 정규화하여 Advantage로 사용

- **강화학습에 대한 분석**
  - RFT vs GRPO : 틀린 시퀀스에 대한 확률 감소로 유의미한 성능 향상
  - RL은 Mass Reallocation으로 볼 수 있다
  - Online RL > Offline RL : 현재 정책에 맞는 분포와 리워드를 활용하여 유의미한 개선 (effectiveness vs efficiency tradeoff)
  - 가능하면 Process 단위의 Reward를 제공 (output score만 사용하는 환경 = RLVR)

Thank you

Dooyoung Kim

(kdysunleo98@gmail.com)

질문 https://forms.gle/LtvyMJ7BFwMKpWtz8 **피드백** https://forms.gle/PAmxLQnRBZVhAMaw8

응답 https://docs.google.com/spreadsheets/d/1uWyc0pUfQOwTTZUDyY3gxlmU5xcFro90kKJKOcDitnk/edit?usp=sharing