AGENTGYM-RL: AN OPEN-SOURCE FRAMEWORK TO TRAIN LLM AGENTS FOR LONG-HORIZON DECISION MAKING VIA MULTI-TURN RL

ICLR 26 Oral

Zhiheng Xi[1,*,†], Jixuan Huang[1,*], Chenyang Liao[1,*], Baodai Huang[1], Jiaqi Liu[1], Honglin Guo[1], Yajie Yang[1], Rui Zheng[1], Junjie Ye[1], Jiazheng Zhang[1], Wenxiang Chen[1], Wei He[1], Yiwen Ding[1], Guanyu Li[1], Zehui Chen[2], Zhengyin Du[2], Xuesong Yao[2], Yufei Xu[2], Jiecao Chen[2], Tao Gui[1,3,4,†], Zuxuan Wu[1,3,4], Qi Zhang[1,3,4,†], Xuanjing Huang[1,3,4], Yu-Gang Jiang[1,3,4]

[1]Fudan University [2]ByteDance Seed [3]Shanghai Key Laboratory of Multimodal Embodied AI
[4]Shanghai Collaborative Innovation Center of Intelligent Visual Computing
zhxi22@m.fudan.edu.cn, {tgui, qz}@fudan.edu.cn

→ LLM 에이전트를 다양한 환경에서 멀티턴 강화학습으로 학습시키는 **전체 프레임워크와 패러다임**

Robust Tool Use via FISSION-GRPO: Learning to Recover from Execution Errors

ACL 26 Main

Zhiwei Zhang[1,3], Fei Zhao[2], Rui Wang[1,3], Zezhong WANG[1], Bin Liang[1,3], Jiakang Wang[2], Yao Hu[2], Shaosheng Cao[2*], Kam-Fai Wong[1,3*]

[1]The Chinese University of Hong Kong,
[2]Xiaohongshu Inc.,
[3]MoE Key Laboratory of High Confidence Software Technologies
zhangzhiwei1019@link.cuhk.edu.hk, caoshaosheng@xiaohongshu.com

→ 그렇다면 멀티턴 tool-use 에이전트가 실제 실행 오류를 만났을 때는 어떻게 복구 능력을 학습할 수 있을까?”

AGENTGYM-RL: AN OPEN-SOURCE FRAMEWORK TO TRAIN LLM AGENTS FOR LONG-HORIZON DECISION MAKING VIA MULTI-TURN RL

Zhiheng Xi[1,*,†], **Jixuan Huang**[1,*], **Chenyang Liao**[1,*], **Baodai Huang**[1], **Jiaqi Liu**[1], **Honglin Guo**[1], **Yajie Yang**[1], **Rui Zheng**[1], **Junjie Ye**[1], **Jiazheng Zhang**[1], **Wenxiang Chen**[1], **Wei He**[1], **Yiwen Ding**[1], **Guanyu Li**[1], **Zehui Chen**[2], **Zhengyin Du**[2], **Xuesong Yao**[2], **Yufei Xu**[2], **Jiecao Chen**[2], **Tao Gui**[1,3,4,†], **Zuxuan Wu**[1,3,4], **Qi Zhang**[1,3,4,†], **Xuanjing Huang**[1,3,4], **Yu-Gang Jiang**[1,3,4]
[1]Fudan University [2]ByteDance Seed [3]Shanghai Key Laboratory of Multimodal Embodied AI
[4]Shanghai Collaborative Innovation Center of Intelligent Visual Computing
zhxi22@m.fudan.edu.cn, {tgui, qz}@fudan.edu.cn

ICLR 26

AGENTGYM-RL

: AN OPEN-SOURCE FRAMEWORK TOTRAIN LLM AGENTS FOR LONG-HORIZON DECISIONMAKING VIA MULTI-TURN RL

- LLM agent를 RL로 학습시키기 위한 제대로 된 오픈소스 프레임워크가 부족하다

  - 모델이 한 번 답하는 것이 아니라, 환경을 관찰하고, 행동하고, 피드백을 받고, 다시 행동하는 과정을 반복해야 함

  => RL이 적절하나,

  기존 오픈소스 생태계에는 다양한 현실적 환경에서 agent를 처음부터 RL로 학습시킬 수 있는 통합 프레임워크가 부족

- agent 성능 향상에는 내부 reasoning만 늘리는 것보다 환경과의 interaction을 늘리는 것이 중요
- 처음부터 긴 interaction horizon으로 RL 학습을 하면 불안정해진다!

"LLM agent가 복잡한 long-horizon task를 잘 풀도록 환경과의 interaction을 늘리되,
RL 학습이 망가지지 않게 안정적으로 학습시키려면 어떻게 해야 하는가?"

⇒ ScalingInter-RL 제안

: 처음에는 짧은 interaction horizon으로 기본적인 행동 정책을 안정적으로 배우게 하고, 이후 단계적으로 허용 interaction 수를 늘려 더 깊은 탐색을 하게 만드는 curriculum 방식

Formulation

- Partially Observable Markov Decision Process

  에이전트가 환경의 전체 상태를 완벽히 알지 못하고, 매 턴마다 관찰 가능한 정보만 보고 행동을 선택하는

→ LLM agent는 한 번 답을 내는 모델이 아니라, 여러 번 환경을 보고 행동하면서 최종 task 성공을 목표로 학습되는 정책 모델로 정의

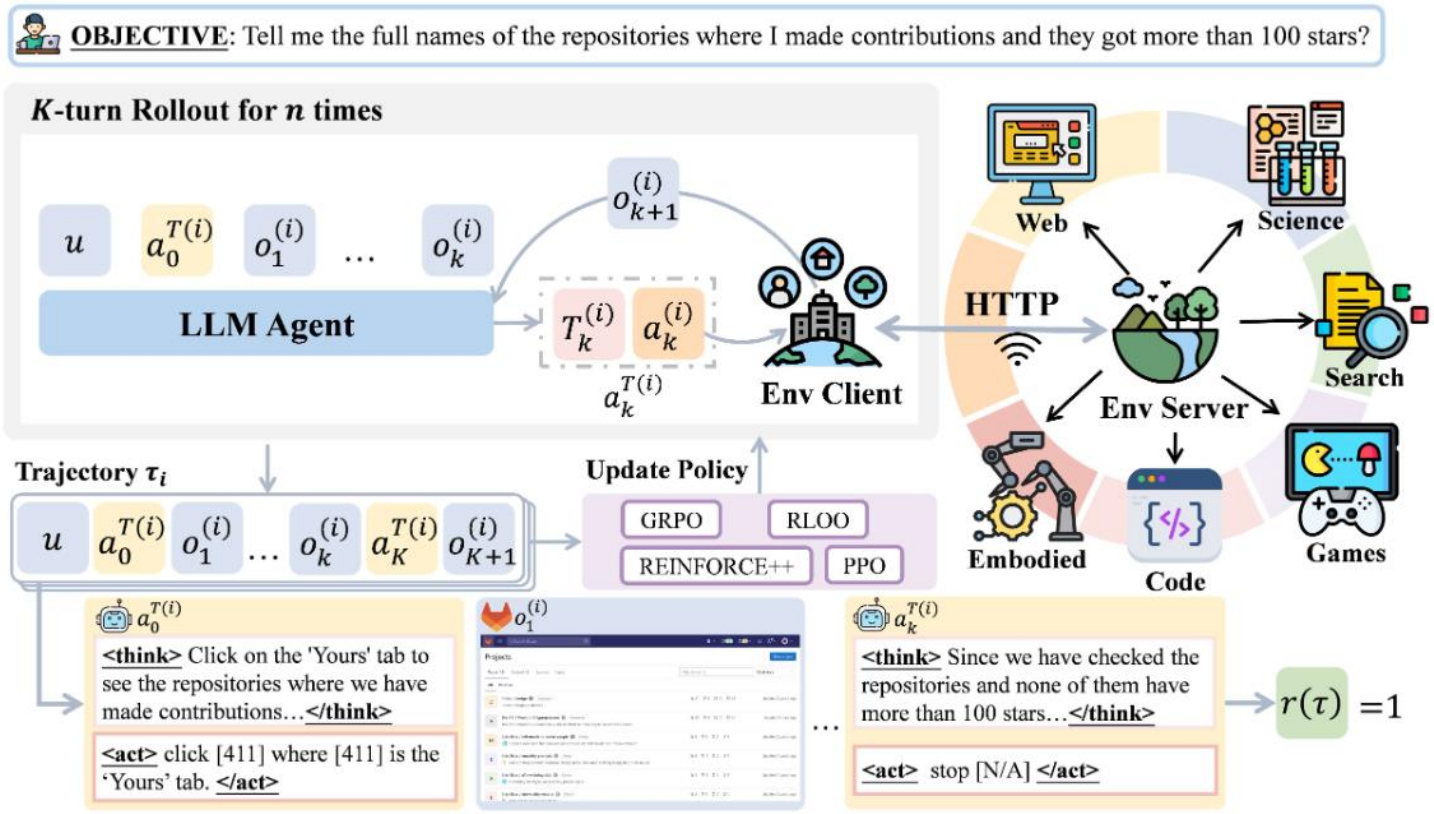
- policy $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[r(\tau)]$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[r(\tau) \sum_{k=0}^{K} \nabla_\theta \log \pi_\theta(a_k|s_k) \right]$$

→ 최종 보상을 높이기 위해 policy gradient 기반 강화학습으로 에이전트 정책을 업데이트

→ 실제 LLM 강화학습에서 많이 쓰이는 알고리즘인 **PPO, GRPO, REINFORCE++** 등이 AgentGym-RL 프레임워크에 통합되어 있음

THE AGENTGYM-RL FRAMEWORK



- AgentGym-RL은 크게 세 가지 핵심 모듈로 구성됨

1. Environment module
2. Agent module
3. Training module

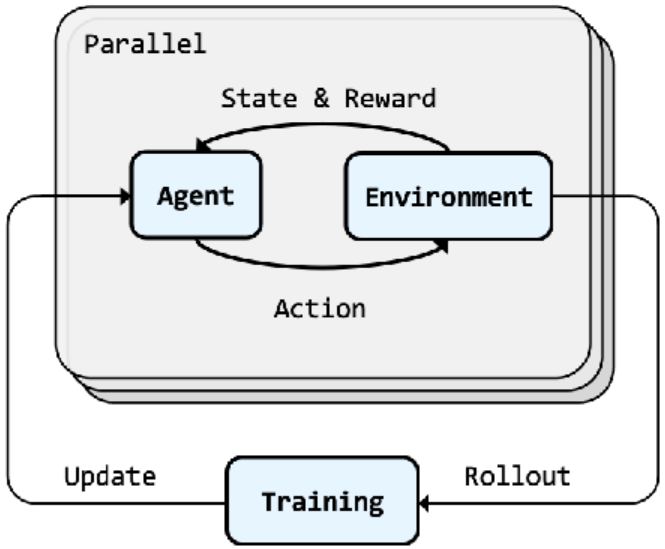
세 모듈이 분리되어 있기 때문에, 새로운 환경을 추가하거나,
=> 다른 LLM agent를 연결하거나,
PPO/GRPO 같은 RL 알고리즘을 바꾸는 작업이 비교적 쉬움

GRPO 같은 알고리즘을 agent 환경에서 돌릴 수 있게 해주는 프레임워크

THE AGENTGYM-RL FRAMEWORK

```
# Stage 1: Generate responses
task_ids = expand(task_ids, sample_num)
envs = create_env_clients(task_ids, “webarena”, base_url)

Do in parallel:
    for (env, task_id) in zip(envs, task_ids):
        env.reset(task_id)

handlers =[
    RolloutHandler().add_user_message(env.observe())
    for env in envs]

for i in range(max_rounds)
    prompts = [h.get_prompt() for h in handlers]
    responses = actor.generate(prompts)

    results = thread_safe_list()
    Do in parallel:
        for (env, response) in zip (envs, responses):
            results.append(env.step(response))

    for (h, r, res) in zip(handlers, responses, results):
        h.add_assistant_message(r)
        h.add_user_message(res.state)
        h.score = res.score

    if all_done(handlers): break
```

```
# Stage 2: Prepare experience
batch = gen_batch_from_rollout_handlers(handlers)
batch = actor.compute_log_prob(batch)
batch = reference.compute_ref_log_prob(batch)
batch = compute_advantages(batch, method="grpo")

# Stage 3: Actor training
actor.update_actor(batch)
```



THE AGENTGYM-RL FRAMEWORK

- AgentGym-RL이 기존 AgentGym에서 무엇을 확장했고, 어떤 특징을 갖는지

  AgentGym-RL은 단순 benchmark 환경이 아니라 **LLM agent를 RL로 실제 학습시키기 위한 통합 프레임워크**

1. Diverse scenarios and environments.

   Web navigation, deep search, digital games, embodied control, and scientific task 시나리오 지원

   → 즉, AgentGym-RL은 특정 task 하나에 맞춘 시스템이 아니라, 여러 종류의 long-horizon decision-making task를 다루는 테스트베드 역할

2. Comprehensive algorithm support.

   RL뿐 아니라 SFT, DPO, self-improvement 같은 보조 학습 방식도 지원

   → 하나의 알고리즘만 구현하는 것이 아니라, 여러 agent RL 방법을 같은 인터페이스에서 비교하고 사용할 수 있게

3. Engineering optimizations.

   scalability를 위해 subprocess-based architecture와 개선된 environment initialization routine을 도입

   Reliability 측면에서는 memory leak이나 recursive implementation 문제를 고쳐서 장시간 학습이 안정적으로 돌아가게

4. Open-source availability and Visualization support.

   코드와 데이터, 평가 파이프라인을 공개하고, 재현 가능한 학습 및 평가 스크립트를 제공

   interactive graphical UI를 제공해서 agent가 어떤 step을 거쳐 task를 수행했는지

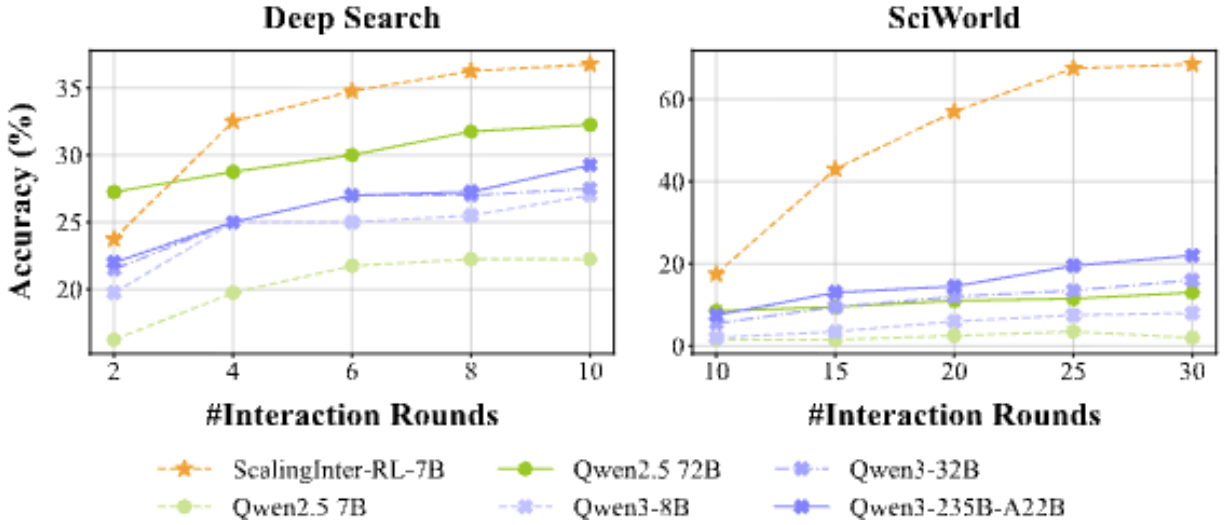
ScalingInter-RL: Scaling Interactions for LLM Agents

- LLM agent가 환경과 상호작용하는 횟수를 학습 중에 점진적으로 늘려서 long-horizon task를 안정적으로 학습시키는 방법 제안

왜 필요한가.

LLM agent task는 단순히 내부적으로 오래 생각한다고 해결되는 문제가 아님

→ Agent 성능을 높이려면 internal reasoning만 늘리는 것이 아니라, external interaction, 즉 환경과의 상호작용 자체를

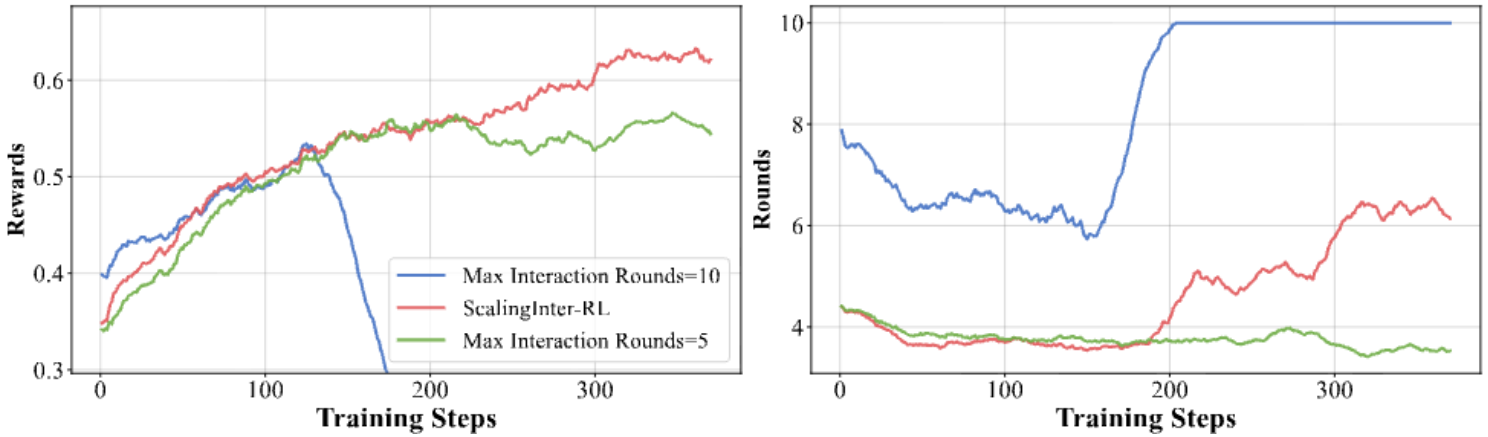


Deep Search와 SciWorld에서 interaction round 수를 늘릴수록 여러 baseline model들의 성능이 올라감

→ 즉, 더 많이 환경과 상호작용할 기회를 주면 task를 더 잘 풀 수 있다

ScalingInter-RL: Scaling Interactions for LLM Agents

- 긴 interaction으로 바로 학습하면 불안정함



  - interaction round가 짧으면 학습은 안정적이지만 성능 한계가 있음
  - interaction round가 길면 더 깊은 탐색이 가능하지만 학습이 불안정해짐
  - 긴 horizon에서는 agent가 불필요한 반복 행동이나 redundant interaction을 하면서 training collapse가 발생할 수 있음

→ long-horizon interaction은 필요하지만, 처음부터 길게 학습하면 RL 학습이 흔들린다

ScalingInter-RL: Scaling Interactions for LLM Agents

- ScalingInter-RL은 이 문제를 해결하기 위해 interaction horizon을 단계적으로 늘림

  → curriculum learning처럼 쉬운 조건에서 시작해서 점점 어려운 조건으로

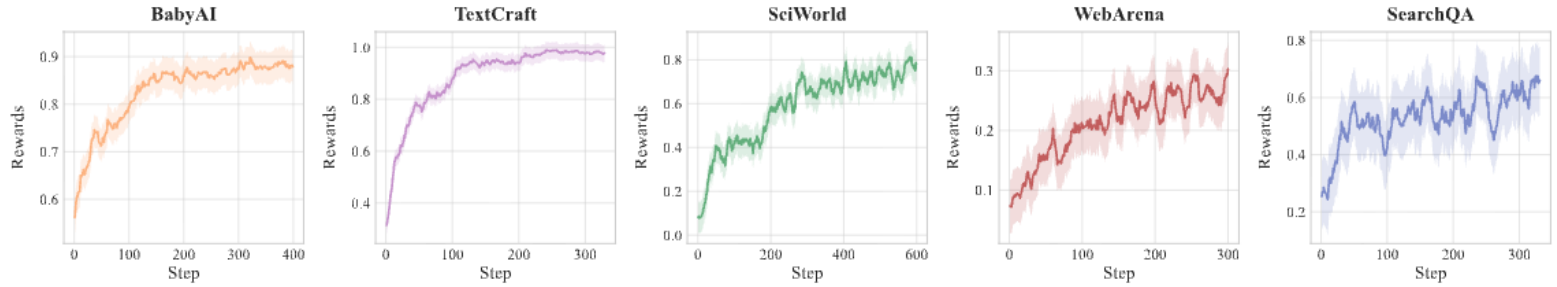


Figure 6: Training rewards in different environments leveraging AgentGym-RL framework with the ScalingInter-RL method.

- BabyAI, TextCraft, SciWorld, WebArena, SearchQA 등 다양한 환경에서 ScalingInter-RL을 적용했을 때 training reward가 올라가는

→ 즉, ScalingInter-RL이 특정 환경 하나에서만 되는 것이 아니라, 여러 종류의 agentic environment에서 안정적인 성능 향상을 보인다는 것을 보여줌

EXPERIMENTS

- RL을 하면 open-source LLM의 agent 성능이 크게 오른다

  Qwen, Llama 같은 open-source 모델들은 multi-turn agent task에서 성능이 제한적, AgentGym-RL로 학습하면 성능이 크게 올라감

  강화학습이 open-source LLM의 agentic intelligence를 향상시킨다

Model	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	Musique	Bamboogle	Overall
Proprietary Models								
GPT-4o (Hurst et al., 2024)	20.0	**70.0**	30.0	30.0	32.0	10.0	34.0	26.8
Qwen-Max (Yang et al., 2024)	24.0	52.0	26.0	24.0	16.0	17.0	36.0	29.5
Gemini-2.5-Flash (Comanici et al., 2025)	8.0	60.0	30.0	24.0	16.0	8.0	34.0	23.5
OpenAI o4-mini (OpenAI, 2025)	22.0	68.0	50.0	38.0	44.0	28.0	62.0	42.5
OpenAI o3 (OpenAI, 2025)	**28.0**	**70.0**	**56.0**	**46.0**	**64.0**	**29.0**	**74.0**	**49.5**
Gemini-2.5-Pro (Comanici et al., 2025)	22.0	62.0	38.0	28.0	48.0	19.0	56.0	36.5
Open-sourced Models $\geq$ 100B								
Qwen3-235B-A22B (Yang et al., 2025a)	28.0	54.0	30.0	32.0	22.0	14.0	32.0	28.3
DeepSeek-V3-0324 (DeepSeek-AI et al., 2024)	28.0	60.0	24.0	28.0	18.0	11.0	34.0	26.5
DeepSeek-R1-0528 (DeepSeek-AI et al., 2025)	**32.0**	**68.0**	**42.0**	**44.0**	**50.0**	**21.0**	**44.0**	**40.3**
Open-sourced Models $<$ 100B								
Qwen2.5-3B-Instruct (Yang et al., 2024)	8.0	42.0	22.0	14.0	8.0	2.0	10.0	13.5
Qwen2.5-7B-Instruct (Yang et al., 2024)	18.0	54.0	20.0	18.0	6.0	4.0	26.0	18.8
Qwen2.5-72B-Instruct (Yang et al., 2024)	22.0	52.0	24.0	28.0	24.0	12.0	**38.0**	26.5
Qwen3-4B (Yang et al., 2025a)	18.0	58.0	26.0	24.0	26.0	5.0	20.0	22.8
Qwen3-8B (Yang et al., 2025a)	26.0	44.0	26.0	22.0	32.0	10.0	32.0	25.3
Qwen3-32B (Yang et al., 2025a)	24.0	54.0	22.0	36.0	28.0	11.0	20.0	25.8
Llama-3.1-8B-Instruct (Dubey et al., 2024)	16.0	26.0	12.0	6.0	2.0	4.0	18.0	11.0
Llama-3.1-70B-Instruct (Dubey et al., 2024)	20.0	44.0	22.0	22.0	18.0	9.0	32.0	22.0
SearchR1-it-3B-v0.3$_{\text{GRPO}}$ (Jin et al., 2025b)	20.0	50.0	30.0	28.0	32.0	5.0	14.0	23.0
SearchR1-it-7B-v0.3$_{\text{GRPO}}$ (Jin et al., 2025b)	24.0	52.0	30.0	22.0	34.0	6.0	26.0	25.0
Our RL Models								
AgentGym-RL-3B	30.0	50.0	30.0	30.0	**46.0**	4.0	12.0	25.8
AgentGym-RL-7B	44.0	64.0	32.0	40.0	36.0	**15.0**	26.0	34.0
ScalingInter-7B	**52.0**	**70.0**	**46.0**	**42.0**	44.0	14.0	24.0	**38.3**

EXPERIMENTS

- ScalingInter-RL이 일반 RL보다 더 좋다

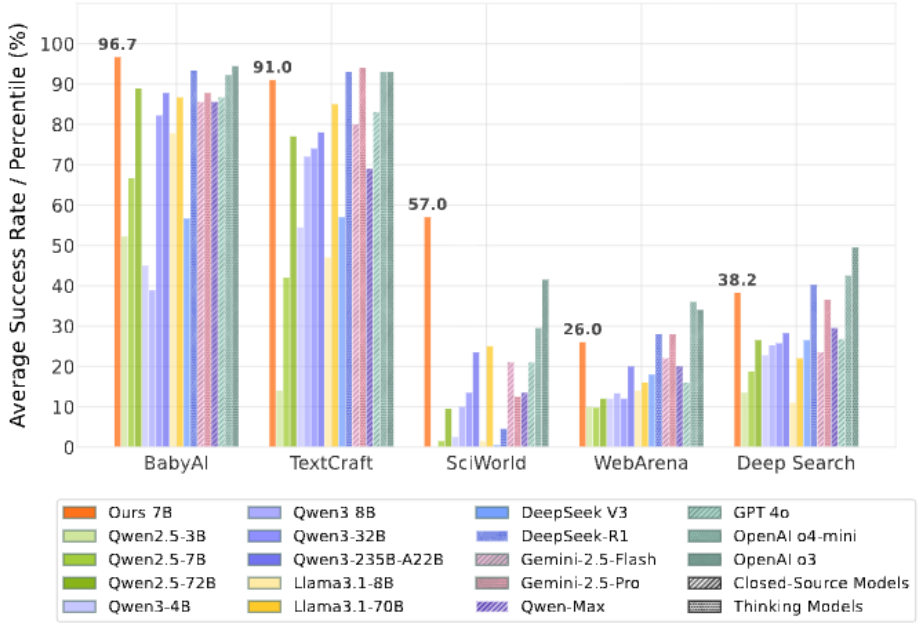
  ScalingInter-RL이 일반 AgentGym-RL보다 일관되게 성능을 더 높인다

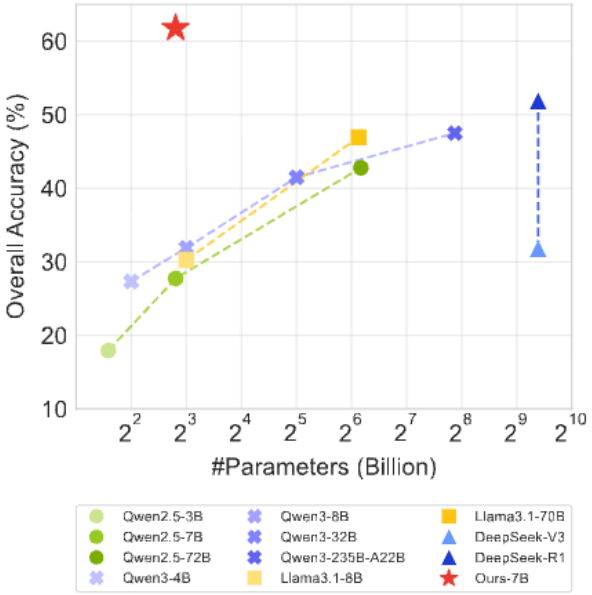
Model	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	Musique	Bamboogle	Overall
Proprietary Models								
GPT-4o (Hurst et al., 2024)	20.0	**70.0**	30.0	30.0	32.0	10.0	34.0	26.8
Qwen-Max (Yang et al., 2024)	24.0	52.0	26.0	24.0	16.0	17.0	36.0	29.5
Gemini-2.5-Flash (Comanici et al., 2025)	8.0	60.0	30.0	24.0	16.0	8.0	34.0	23.5
OpenAI o4-mini (OpenAI, 2025)	22.0	68.0	50.0	38.0	44.0	28.0	62.0	42.5
OpenAI o3 (OpenAI, 2025)	**28.0**	**70.0**	**56.0**	**46.0**	**64.0**	**29.0**	**74.0**	**49.5**
Gemini-2.5-Pro (Comanici et al., 2025)	22.0	62.0	38.0	28.0	48.0	19.0	56.0	36.5
Open-sourced Models $\geq$ 100B								
Qwen3-235B-A22B (Yang et al., 2025a)	28.0	54.0	30.0	32.0	22.0	14.0	32.0	28.3
DeepSeek-V3-0324 (DeepSeek-AI et al., 2024)	28.0	60.0	24.0	28.0	18.0	11.0	34.0	26.5
DeepSeek-R1-0528 (DeepSeek-AI et al., 2025)	**32.0**	**68.0**	**42.0**	**44.0**	**50.0**	**21.0**	**44.0**	**40.3**
Open-sourced Models $<$ 100B								
Qwen2.5-3B-Instruct (Yang et al., 2024)	8.0	42.0	22.0	14.0	8.0	2.0	10.0	13.5
Qwen2.5-7B-Instruct (Yang et al., 2024)	18.0	54.0	20.0	18.0	6.0	4.0	26.0	18.8
Qwen2.5-72B-Instruct (Yang et al., 2024)	22.0	52.0	24.0	28.0	24.0	12.0	**38.0**	26.5
Qwen3-4B (Yang et al., 2025a)	18.0	58.0	26.0	24.0	26.0	5.0	20.0	22.8
Qwen3-8B (Yang et al., 2025a)	26.0	44.0	26.0	22.0	32.0	10.0	32.0	25.3
Qwen3-32B (Yang et al., 2025a)	24.0	54.0	22.0	36.0	28.0	11.0	20.0	25.8
Llama-3.1-8B-Instruct (Dubey et al., 2024)	16.0	26.0	12.0	6.0	2.0	4.0	18.0	11.0
Llama-3.1-70B-Instruct (Dubey et al., 2024)	20.0	44.0	22.0	22.0	18.0	9.0	32.0	22.0
SearchR1-it-3B-v0.3$_{\text{GRPO}}$ (Jin et al., 2025b)	20.0	50.0	30.0	28.0	32.0	5.0	14.0	23.0
SearchR1-it-7B-v0.3$_{\text{GRPO}}$ (Jin et al., 2025b)	24.0	52.0	30.0	22.0	34.0	6.0	26.0	25.0
Our RL Models								
AgentGym-RL-3B	30.0	50.0	30.0	30.0	**46.0**	4.0	12.0	25.8
AgentGym-RL-7B	44.0	64.0	32.0	40.0	36.0	**15.0**	26.0	34.0
ScalingInter-7B	**52.0**	**70.0**	**46.0**	**42.0**	44.0	14.0	24.0	**38.3**

→ 단순히 RL을 하는 것도 성능이 오르지만 **상호작용 길이를 단계적으로 늘리는 학습 전략**이 더 효과적

EXPERIMENTS

- 작은 모델도 RL post-training을 하면 큰 모델과 경쟁할 수 있다



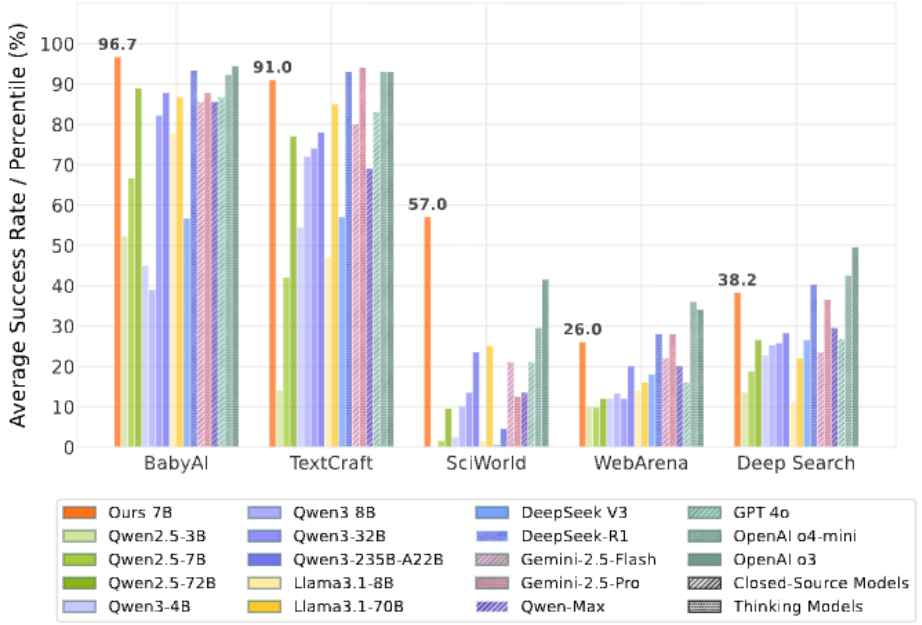


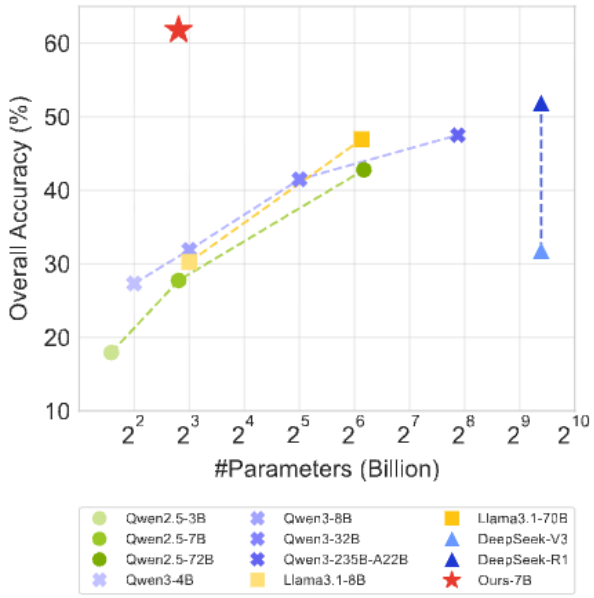
ScalingInter-RL로 학습한 7B 모델이 훨씬 큰 모델들보다 높은 평균 성능을 보임

→ "큰 모델이면 agent도 잘한다"가 아니라, **agent로서 행동하는 법을 학습했는지가 중요하다**

EXPERIMENTS

- 환경에 따라 RL 효과가 다르다





- TextCraft, BabyAI, SciWorld처럼 **규칙이 명확하고 reward가 비교적 분명한 환경**에서는 RL 성능 향상이 매우 큼
- 반면 WebArena나 Deep Search처럼 더 open-ended하고 복잡한 환경에서는 성능 향상이 상대적으로 제한적

→ RL agent 학습에서는 모델 구조뿐만 아니라 환경 설계, reward 품질, feedback의 명확성이 매우 중요하다.

Discussion

- Test-time Scaling for Agents

: 학습이 끝난 뒤, 추론 시점에 agent에게 더 많은 계산/시도 기회를 주면 성능이 올라가는가 ?

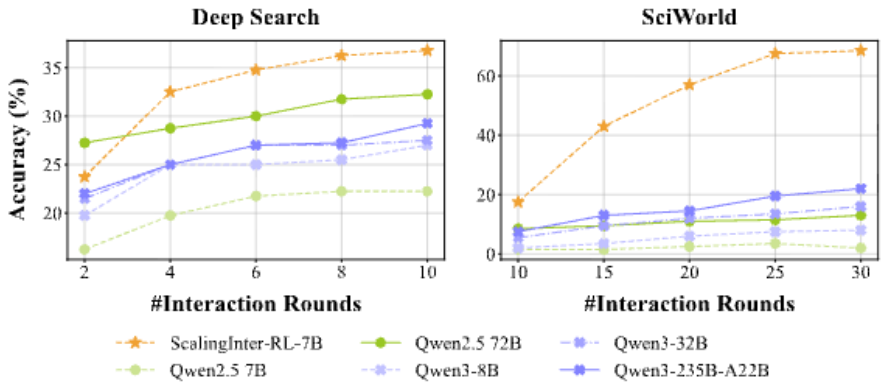
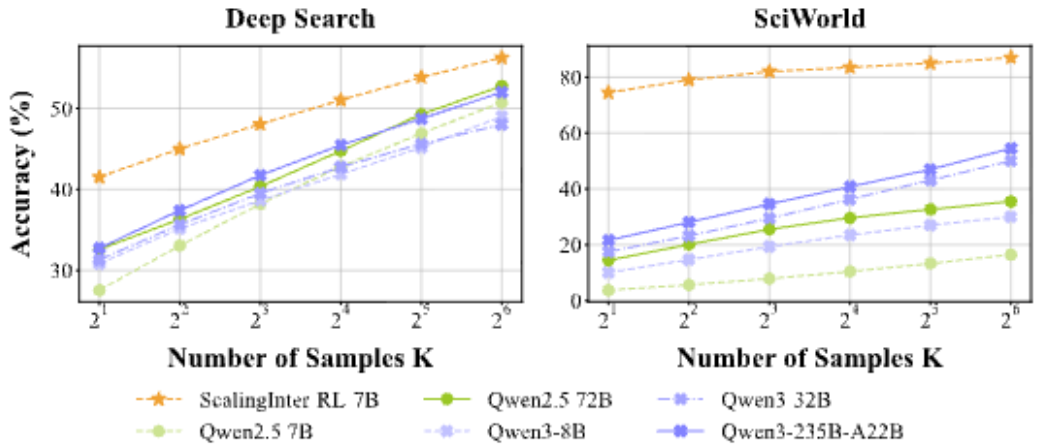


Figure 5: Scaling test-time interaction turns.



- Deep Search와 SciWorld에서 interaction round 수가 늘어날수록 성능이 좋아

→ Agent task에서는 단순히 내부 reasoning을 오래 하는 것뿐 아니라, 실제 환경과 더 많이 상호작용하는 것이 성능 향상에 중

- 같은 문제에 대해 여러 개의 trajectory를 샘플링했을 때, K가 커질수록 성능이 올라감

→ 단순히 평균 성능만 좋은 것이 아니라, 여러 시도를 했을 때 성공 trajectory를 만들어낼 가능성도 높다

Discussion

- PERFORMANCE OF DIFFERENT RL ALGORITHMS

: RL 알고리즘에 따라 성능이 어떻게 달라지는지

Table 2: Evaluation results of different RL algorithms.

RL Algorithms	TextCraft	BabyAI	SearchQA
Qwen2.5-3B-Instruct			
GRPO	**75.00**	**93.33**	**25.75**
REINFORCE++	28.00	70.00	13.25
Qwen2.5-7B-Instruct			
GRPO	**89.00**	**92.22**	**34.00**
REINFORCE++	73.00	84.44	24.00

- GRPO가 TextCraft, BabyAI, SearchQA에서 일관되게 더 좋음

→ 이 차이를 advantage 계산 방식으로 설명함

→ GRPO는 하나의 query에 대해 여러 trajectory를 만들고,

→ 그 평균을 baseline으로 사용해 advantage를 정규화하기 때문에 개별 trajectory의 outlier 영향을 줄일 수 있음.

→ 반면 REINFORCE++는 batch 단위 정규화를 하기 때문에 gradient variance가 더 클 수 있다

Discussion

- ABLATION STUDY FOR SCALINGINTER-RL

: hyperparameter를 바꿔도 성능이 유지되는지

→ ScalingInter-RL이 hyperparameter에 민감하지 않다

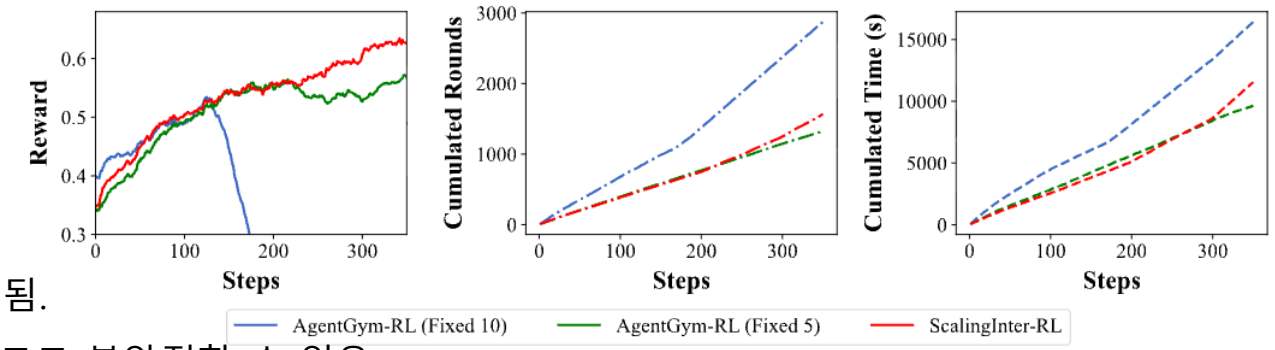
Table 3: Ablation study of ScalingInter-RL.

Interact Turn List	Stage Transition Frequency	Performance
[5,8,10]	100	38.3
[5,8,10]	75	37.8
[5,8,10]	125	38.5
[3,8,13]	100	36.8
[8,10,12]	100	37.6
[5,10,15]	100	39.1
[5,7,9]	100	37.8

- EFFICIENCY ANALYSIS OF SCALINGINTER-RL

: 성능뿐 아니라 학습 효율성도 좋은지

(Fixed: 학습 동안 최대 interaction round 수를 고정)



- Fixed 5는 interaction이 짧아서 효율적이지만 최종 성능이 제한됨.
- Fixed 10은 긴 interaction을 허용해 탐색은 가능하지만 비용이 크고 불안정할 수 있음.
- ScalingInter-RL은 **처음에는 짧게 시작하고 나중에 늘리므로, 비교적 적은 시간과 interaction 비용으로 높은 reward에 도달함**.

Discussion

- APPLYING SCALINGINTER-RL TO MORE ALGORITHMS

Table 4: Applying ScalingInter-RL to more algorithms.

RL Algorithm	Method	TextCraft	BabyAI	SciWorld
Base Model	-	42.00	66.67	1.50
PPO	AgentGym-RL-7B	68.00	86.66	10.83
	ScalingInter-7B	**71.00**	**90.00**	**25.69**
REINFORCE++	AgentGym-RL-7B	73.00	84.44	13.63
	ScalingInter-7B	**77.00**	**87.77**	**26.14**

ScalingInter-RL을 붙였을 때 성능이 올라감

Conclusion

- AgentGym-RL은 기존 agent RL 학습 구조를 오픈소스 프레임워크로 체계화하고,
- ScalingInter-RL을 통해 long-horizon interaction 학습을 안정화한 연구

- LLM agent 성능은 모델 크기뿐 아니라 환경과의 상호작용 기반 post-training에 크게 좌우됨

- 하지만 이 논문의 의의는 새로운 이론적 알고리즘보다는,
  다양한 agent 환경과 RL 알고리즘을 하나의 오픈소스 프레임워크로 통합했다는 데 있음

- 따라서 본 논문은,
  AgentGym-RL 프레임워크를 소개하고 ScalingInter-RL이라는 실용적 training recipe를 제시하는
  technical report 성격이 강한

Robust Tool Use via FISSION-GRPO: Learning to Recover from Execution Errors

Zhiwei Zhang[1,3], Fei Zhao[2], Rui Wang[1,3], Zezhong WANG[1], Bin Liang[1,3], Jiakang Wang[2], Yao Hu[2], Shaosheng Cao[2*], Kam-Fai Wong[1,3*]

[1]The Chinese University of Hong Kong,
[2]Xiaohongshu Inc.,
[3]MoE Key Laboratory of High Confidence Software Technologies
zhangzhiwei1019@link.cuhk.edu.hk, caoshaosheng@xiaohongshu.com

Robust Tool Use via FISSION-GRPO

: Learning to Recover from Execution Errors

- "툴을 잘 호출하는 LLM"보다, 실제 서비스에서는 "툴 호출이 실패했을 때 에러 메시지를 이해하고 복구하는 LLM"이 필요

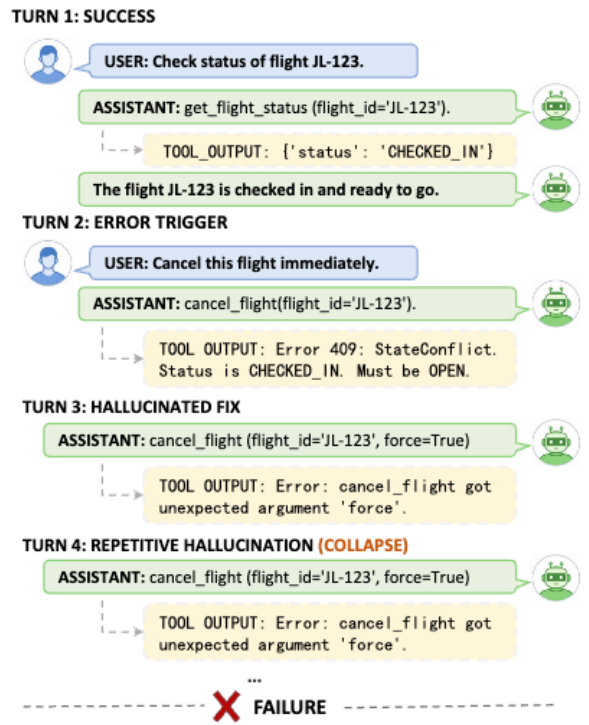
  기존 학습법은 이 복구 능력을 제대로 가르치지 못한다

- multi-turn tool use

  실제 환경에서는 API 에러, 잘못된 파라미터, 바뀐 상태값, timeout 같은 문제가 필연적

  => 좋은 tool-use agent는 "처음부터 맞는 호출"뿐 아니라 실패 후 self-correction을 할 수 있

  => 논문은 이를 "robustness", 특히 **execution error recove**ry 문제로



(a) Typical failure: an API error triggers a hallucinated retry loop.

- 기존 연구들의 한계

- offline synthetic error-correction dataset을 만들어 학습시키는 방식

  : 모델이 학습되면서 실제로 내는 에러 분포가 계속 바뀌기 때문에 금방 mismatch

- RL 방식은 실패를 그냥 낮은 reward로 처리

  : 어떻게 고쳐야 하는지에 대한 구체적인 supervision은 받지 못함

execution error를 단순 실패가 아니라 학습 가능한 경험으로 바꾸자

Robust Tool Use via FISSION-GRPO

: Learning to Recover from Execution Errors

- 실제 API 환경에서는 에러가 나고, 파라미터가 invalid해지고, 시스템 상태가 예상과 다르게 바뀜

- **failed trajectory를 버리지 않고**, 그 실패에 대해 Error Simulator가 진단 **피드백을 붙여 새로운 corrective training instance**로 만들고, 그 상태에서 여러 recovery rollout을 다시 샘플링

  ➔ FISSION-GRPO

- 하나의 실패가 여러 복구 시도로 "fission"되어 dense training signal을 만든다

- Contribution

  - execution error를 on-policy corrective training instance로 바꾸는 Fission-GRPO framework

    → 결과적으로 sparse negative reward 하나로 끝나는 것이 아니라, 실패 하나가 dense한 학습 신호로 바뀜

  - target answer를 직접 누설하지 않으면서 현실적인 diagnostic feedback을 생성하는 Learned Error Simulator
  - BFCL v4 Multi-Turn, TAU-Bench, TAU2-Bench에서 성능 향상을 보였다는 empirical validation

FISSION-GRPO

- Preliminaries

  GRPO를 backbone으로 사용

- Reward Design

  reward를 단순히 "성공/실패"로 주지 않고, 세 가지 요소를 합침 $R(\tau, t) = \frac{1}{3}[w_{fmt}(t)R_{fmt}(\tau) + w_{corr}(t)R_{corr}(\tau) + R_{len}(\tau)]$

  1) Format Compliance

     tool call이 XML/JSON schema 같은 요구 **형식을 잘 따르는지,** binary reward

     학습 초반에는 format을 잘 맞추는 게 중요하므로 w_fmt(t)를 크게 두고, 시간이 갈수록 그 비중을 줄임

  2) Functional Correctness

     실제로 user intent에 맞는 tool을 호출했는지

     function name이 맞는가(0.5) parameter 값이 맞는가 (0.5)

  3) Efficiency Regularization

     답변이나 reasoning이 너무 길어지거나 이상하게 반복되는 것을 막기 위한 길이 reward

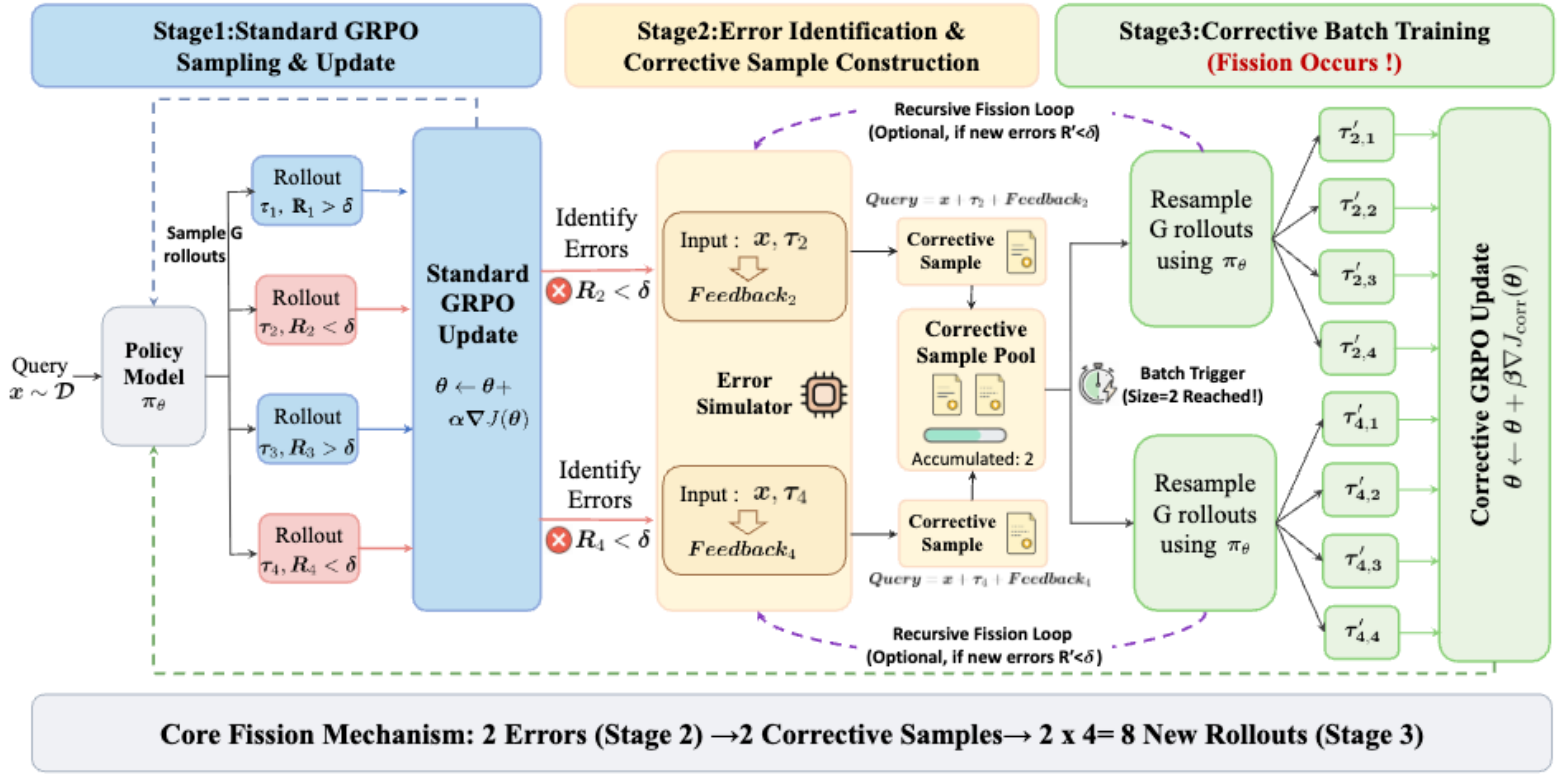
     목표 길이에 가까울수록 높고, 너무 짧거나 길면 낮아지는 piecewise Gaussian function

FISSION-GRPO

- FISSION-GRPO Framework

  3-stage

  standard GRPO exploration → error identification & synthesis → fission-based corrective update



recovery rollout을 샘플링하고,
이를 이용해 다시 policy를 업데이

→ 모델은 단순히 성공 trajectory만 배우는 것이 아니라, 실패 후 복구하는 방법도 함께 배우게

FISSION-GRPO

- Stage 1: Standard Exploration and Update

  입력 query x가 들어오면 현재 policy πθ에서 G개의 rollout을 샘플링

  각 rollout에 대해 앞에서 정의한 reward, 즉 format compliance, functional correctness, length reward를 계산

  업데이트

< 여기서 샘플링된 모든 trajectory를 Stage 2로 넘김 - 성공한 것만 학습하는 게 아니라 **실패한 것들을 따로 잡아내서 corrective training에 사용하기** 위해>

- Stage 2: Error Identification and Corrective Sample Construction

  실패한 trajectory를 찾아서, 그것을 다시 학습 가능한 corrective sample로

  두가지 에러 타입

  1) Format error

     tool call 형식 자체가 틀린 경우 (ex. JSON schema가 깨졌거나, parser가 읽을 수 없는 형태)

  2) Semantic error

     형식은 맞지만 function이나 parameter가 틀린 경우

FISSION-GRPO

- Hybrid Feedback Synthesis

  단순히 "reward 0점"을 주는 게 아니라, 모델이 복구할 수 있도록 diagnostic feedback을 만들어줌

  - format error

    parser/compiler-style feedback처럼 "schema가 틀렸다", "필드가 누락됐다"는 deterministic error message

  - Semantic error

    Learned Error Simulator 사용

    simulator: Qwen3-32B를 SFT, runtime environment가 줄 법한 짧고 구체적인 error message를 생성

  ! simulator가 정답을 그대로 알려주는 게 아니라, 현실적인 runtime feedback처럼 복구 힌트만 주도록 제한

- Corrective Sample Construction

  $$x_{\text{corr}} = [x; \tau_{\text{err}}; f]$$

  실패 trajectory τerr와 feedback f가 생기면, 새로운 corrective context를 만듦

  원래 대화 x 뒤에 실패한 tool call과 diagnostic feedback을 만들어서 모델은 다음 rollout에서 "방금 왜 틀렸는지"를 보고 다시 시도하게

  이 corrective sample들은 B_corr라는 LIFO buffer에 저장

FISSION-GRPO

- Stage 3: Corrective Batch Training

  LIFO buffer에 corrective samples가 충분히 쌓이면, 가장 최근 corrective context들을 꺼냄

  그리고 각 corrective context x_corr에 대해 다시 여러 개의 rollout을 샘플링

$$\{\tau'_j\}_{j=1}^{G'} \sim \pi_\theta(\cdot \mid x_{\text{corr}}).$$

  하나의 실패 사례가 G'개의 recovery attempt로 "분열(fission)"

Experiments

Method	BFCL v4 Multi-Turn					TAU-Bench		TAU2-Bench		
	Overall	Base	Miss Func	Miss Param	Long Ctx	Retail	Airline	Retail	Airline	Telecom
Qwen3-1.7B Models										
Base	7.80	10.00	11.00	8.00	2.50	6.1	14.0	7.9	12.0	25.0
GRPO	17.12	22.00	**18.50**	15.50	12.50	7.0	20.0	8.5	12.7	32.5
DAPO	16.00	22.00	17.00	14.00	11.00	**8.7**	18.0	6.7	14.0	28.3
Dr.GRPO	16.12	19.50	17.50	14.50	13.00	7.8	22.0	**9.4**	15.3	32.5
FISSION-GRPO	**20.38**	**29.00**	**18.50**	**16.00**	**18.00**	7.8	**24.0**	8.5	**16.0**	**37.5**
Qwen3-4B Models										
Base	19.37	24.50	19.00	14.50	19.50	19.1	26.0	22.8	20.0	22.5
GRPO	36.38	46.50	34.50	27.50	37.00	20.0	24.0	27.2	26.0	37.5
DAPO	38.25	48.50	36.50	28.00	**40.00**	21.7	**32.0**	28.1	24.0	30.0
Dr.GRPO	34.75	43.00	34.50	27.50	34.00	20.9	22.0	23.7	26.0	35.0
AWPO	38.75	43.00	39.50	**36.50**	36.00	27.0	22.0	**29.8**	28.0	40.0
FISSION-GRPO	**40.87**	**51.50**	**42.50**	30.50	39.00	**37.4**	30.0	29.0	**32.0**	**42.5**
External 8B Agent Models	**specialized tool agent**									
ToolACE-2-8B	37.00	47.00	31.00	28.00	42.00	0.9	4.0	8.2	26.7	26.7
BitAgent-8B	37.75	46.50	37.50	24.00	43.00	2.6	6.0	6.1	37.3	6.7
Qwen3-8B Models										
Base	28.75	35.50	37.00	22.50	20.00	35.7	23.2	29.8	28.0	40.0
GRPO	42.75	50.50	41.00	36.00	43.50	39.1	36.0	28.1	26.0	52.5
DAPO	43.12	54.50	44.50	29.00	44.50	45.2	20.0	34.2	26.0	47.5
Dr.GRPO	44.88	55.00	**45.00**	32.50	47.00	47.0	28.0	**39.5**	**34.0**	35.0
AWPO	44.50	53.50	41.50	**40.50**	42.50	43.5	30.0	32.5	26.0	45.0
FISSION-GRPO	**46.75**	**57.50**	43.50	38.00	**48.00**	**51.3**	**40.0**	36.8	32.0	**55.0**

Table 1: Main results on BFCL v4 Multi-Turn, TAU-Bench, and TAU2-Bench. BFCL reports overall and category-level accuracy (%), while TAU-Bench and TAU2-Bench report pass@1 (%) under simulated-user interaction. The best result within each model-size group is highlighted in **bold**; rows shaded in blue denote our proposed FISSION-GRPO.

- 모든 Qwen3 scale에서 FISSION-GRPO가 GRPO 및 다른 post-training baseline보다 전반적으로 좋다
- Qwen3-8B FISSION-GRPO는 BFCL에서 ToolACE-2-8B, BitAgent-8B보다 각각 9.75%p, 9.00%p 높다

=> 즉, specialized tool agent보다도 일반 Qwen3 기반 FISSION 학습이 더 강하다

Experiments

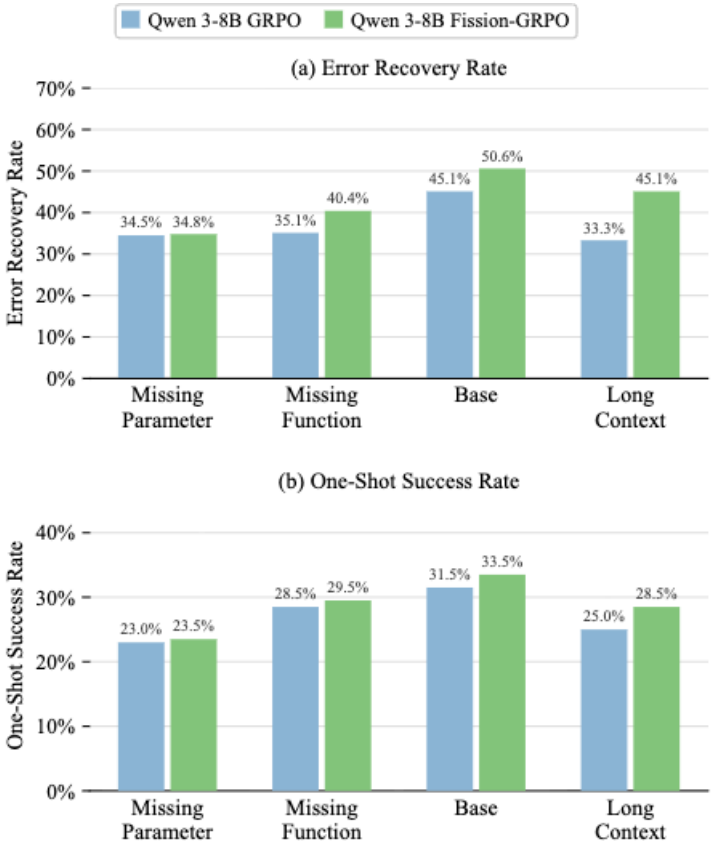


Figure 3: Performance decomposition on BFCL v4 Multi-Turn (Qwen3-8B).

- Error Recovery Analysis

  Error Recovery Rate: 중간에 error가 발생했는데도 결국 성공한 비율

  One-Shot Success Rate: error 없이 바로 성공한 비율

→ FISSION-GRPO의 성능 향상은 주로 Error Recovery Rate 향상에서

→ 중요한 점은 one-shot 능력이 희생되지 않았다는 것

➔ FISSION-GRPO는 "처음부터 맞히는 능력"을 깎아먹으면서 error recovery만 올린 게 아니라, error prevention과 error correction을 함께 개선했다

Base: 기본 tool-use 시나리오
M.Func: Missing Function 오류 상황
M.Param: Missing Parameter 오류 상황
Long: 긴 컨텍스트/상태 추적 상황

Experiments

Method	Avg.	Base	M.Func	M.Param	Long
Qwen3-1.7B					
GRPO	17.12	22.00	18.50	15.50	12.50
Static	17.75	23.50	**21.00**	15.50	11.00
Dynamic	**20.38**	**29.00**	18.50	**16.00**	**18.00**
Qwen3-4B					
GRPO	36.38	46.50	34.50	27.50	37.00
Static	37.25	50.50	34.00	28.50	36.00
Dynamic	**40.87**	**51.50**	**42.50**	**30.50**	**39.00**
Qwen3-8B					
GRPO	42.75	50.50	41.00	36.00	43.50
Static	44.00	53.50	43.00	35.50	44.00
Dynamic	**46.75**	**57.50**	**43.50**	**38.00**	**48.00**

Table 2: Ablation on feedback quality. *Static* denotes Fission training with generic error prompts; *Dynamic* uses our simulated feedback. *Avg.* denotes overall accuracy; *M.Func* and *M.Param* denote missing function and missing parameter errors, respectively. The best result in each panel is in **bold**.

- Impact of Feedback Quality

Fission 구조 자체가 중요한지, 아니면 Error Simulator가 좋은 feedback을 줘서만 성능이 오른 건지

- GRPO : 일반 GRPO, explicit recovery training 없음
- Fission-Static : fission은 하지만 모든 error에 generic message 사용
- Fission-Dynamic : 논문의 full method, Error Simulator가 context-aware feedback 생성

- Static도 GRPO보다 좋아짐

→ fission resampling 구조 자체가 유효하다

- Dynamic이 Static보다 더 좋음

→ generic feedback보다 구체적인 diagnostic feedback이 semantic error, 특히 Miss Param과 Long Context correction에 더 효과적

Experiments

Dimension	In-domain	Out-of-domain	Δ
Localization	4.21	3.97	−0.24
Actionability	4.26	4.12	−0.14
Non-leakage	4.88	4.83	−0.05

Table 3: Error Simulator cross-domain evaluation (1–5 scale). The simulator maintains high quality on unseen domains with minimal degradation.

- Error Simulator Analysis

새로운 domain에도 generalize하는가?

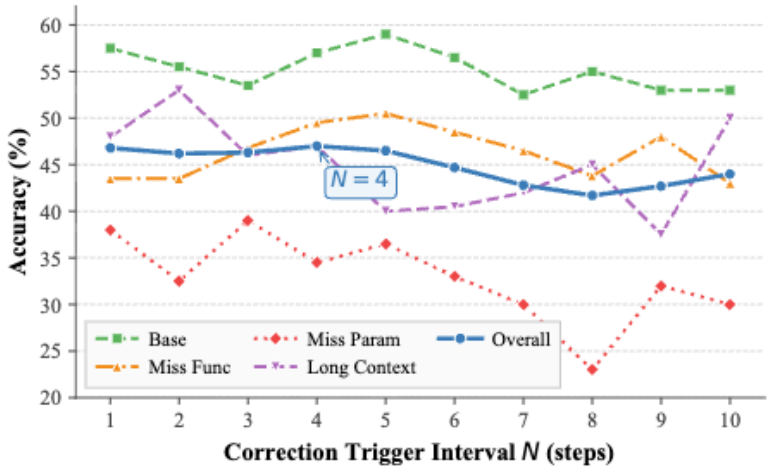
정답 tool call을 누설하지 않는가?

- domain이 바뀌어도 점수 하락이 최대 0.24로 작고, Non-leakage는 거의 유지
- human annotation에서도 100개 output 중 96%가 strictly non-leaking으로 판단

→ simulator가 "정답을 알려주는 teacher"가 아니라 "현실적인 error message를 주는 simulator"로 작동한다는 점

- Localization: 오류 위치/원인을 정확히 짚는가
- Actionability: 모델이 그 피드백을 보고 다음 행동을 고칠 수 있는가
- Non-leakage: 정답 tool call을 그대로 알려주지는 않는가

Experiments



N	Method	Avg.	Base	M.F	M.P	Long
3	GRPO (432 upd.)	41.75	56.50	40.00	29.50	41.00
	Fission (432 upd.)	**46.75**	**57.50**	**43.50**	**38.00**	**48.00**
6	GRPO (312 upd.)	43.75	**53.00**	44.00	31.50	46.50
	Fission (312 upd.)	**46.12**	**53.00**	**48.50**	**36.00**	**47.00**

Table 4: Compute-matched comparison on BFCL v4 Multi-Turn (Qwen3-8B). Both methods use identical total update steps.

- Trigger Frequency and Compute Efficiency

error가 생길 때마다 corrective rollout을 추가로 샘플링하므로 compute overhead 문제

그래서 저자들은 correction trigger interval N을 조절

N: N은 correction update가 최소 몇 global step 간격으로 발생할 수 있는지

- 작은~중간 정도의 N에서는 성능이 안정적이지만, correction이 너무 드물어지면 성능이 떨어짐

  → parameter-level error와 long-context state tracking error가 timely correction에 특히 의존한다

- Compute-matched 비교

G′ = G로 두어 fission update와 standard GRPO update의 비용을 맞추고, 총 update step도 동일하게

- 같은 compute budget에서도 FISSION-GRPO가 GRPO를 계속 이김

  → 성능 향상이 단순히 compute를 더 써서 생긴 게 아니라,

  training budget을 현재 policy의 failure mode 쪽으로 더 잘 배분했기 때문

Conclusion

- 기존 tool-use RL은 execution error를 주로 sparse negative reward로 처리
- 하지만 실제 multi-turn agent에서는 에러 이후 무엇을 잘못했는지 이해하고 복구하는 능력이 중요하다.
- FISSION-GRPO는 failed trajectory에 diagnostic feedback을 붙이고, 여러 recovery rollout으로 확장해 학습
- 이 논문은 tool-use agent 학습의 초점을
  "avoid errors"에서 "learn from errors"로

Thank you

Challenge Category	Definition	Pass Criteria	Fail Criteria	Example
Inference Memory	This metric evaluates the model's ability to retain and accurately reference SPECIFIC information from previous turns in the conversation, especially from multiple turns ago. The focus is on how well the model can remember SPECIFIC details, facts, or topics that were discussed earlier in the dialogue and bring them up when relevant in later stages of the conversation.	The model successfully recalls and integrates the necessary context from earlier turns, making its responses coherent and contextually appropriate.	If the model shows any indication that it forgot or misremembered key details from previous turns, leading to incoherent or contextually inconsistent responses, it would be considered a failure on this challenge category.	In a conversation about a dinner date, the user mentions their girlfriend is allergic to nuts. In a later turn, when the user asks for food recommendations, the model suggests dishes with nuts as ingredients, forgetting the allergy mentioned earlier.