南京航空航天大學
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

Reinforcement Learning for Self-Improving Agent with Skill Library

Jiongxiao Wang[1*] **Qiaojing Yan**[2] **Yawei Wang**[2] **Yijun Tian**[2] **Soumya Smruti Mishra**[2]
Zhichao Xu[2] **Megha Gandhi**[2] **Panpan Xu**[2†] **Lin Lee Cheong**[2]
[1]University of Wisconsin–Madison; [2]AWS Agentic AI
jwang2929@wisc.edu; {qiaojiny, yawenwan, yijunt, soumish, xzhichao, ganmegha, xupanpan, lcheong}@amazon.com

ACL 2026

Background

LLM 智能体是一个多轮“观察—决策—行动”系统，但是完成过一次任务，不等于下次已经学会

同时现有训练通常仍以单个任务为基本单元，但是当前任务的结果奖励不能直接评价某段行为是否对未来任务有帮助

现有 LLM Agent 的局限

- Agent 通常通过 推理 + Tool Calling / Code Execution 完成复杂任务
- 面对相似任务时，往往需要重复进行：任务分解、工具选择、参数生成、多步执行与结果判断
- 导致重复推理、Tool 调用冗余、Token 开销高、执行效率低

Methods

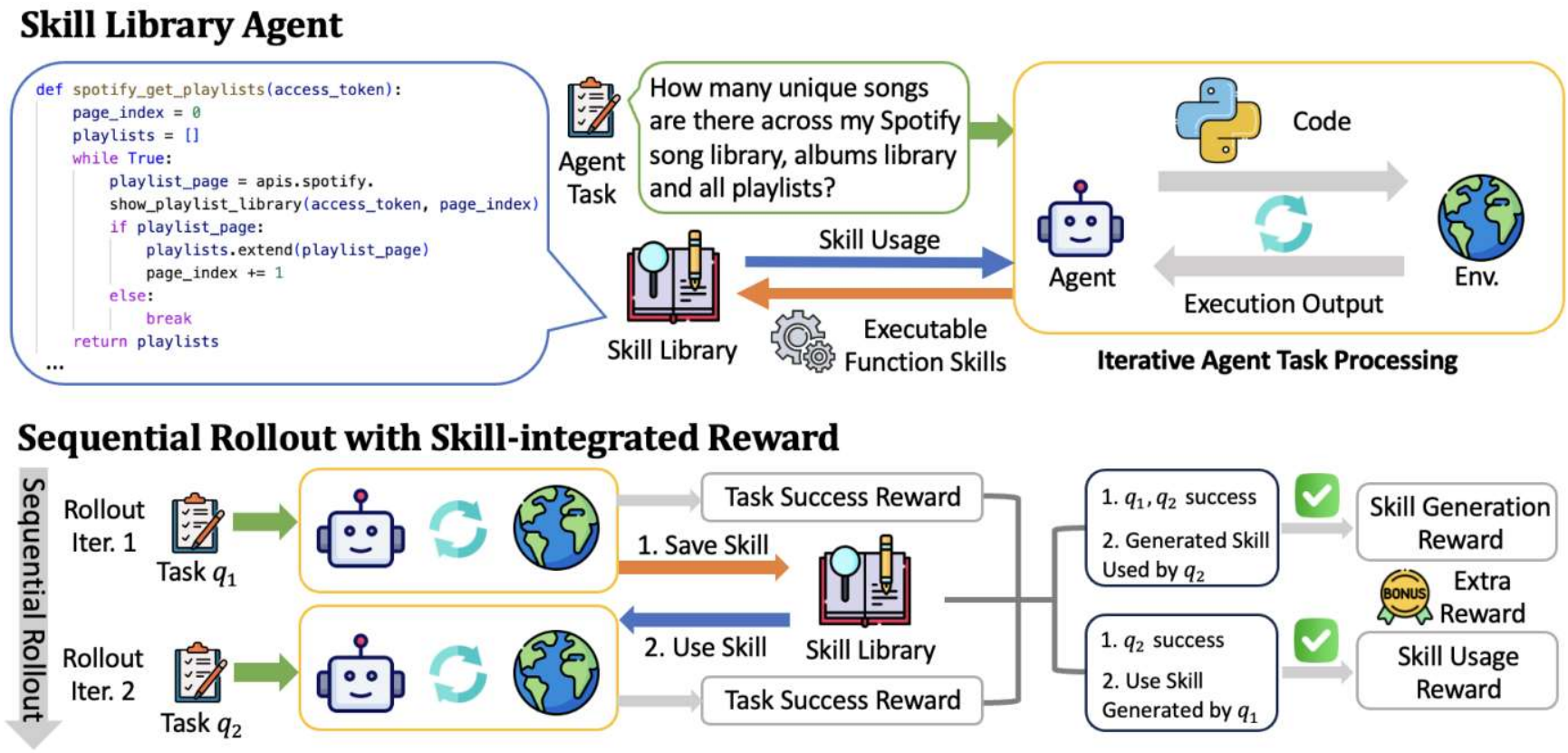
南京航空航天大學
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

SAGE（Skill Augmented GRPO for self-Evolution）核心价值：把技能库从提示组件变成强化学习的一部分，让 Agent 将一次任务中的有效经验沉淀为可执行能力，并在后续任务中复用。

Skill Library 的基本思想

- 将成功任务中的多步操作抽象为可复用能力，后续相似任务可直接调用 Skill，减少重复探索
- Low-level Actions → Reusable Skill

Methods



三个核心组成部分：

- Skill Library Agent：定义智能体如何生成、保存、使用和更新技能
- Sequential Rollout：让前序任务产生的技能在后续任务中得到验证
- Skill-integrated Reward：将技能复用结果转化为参数更新信号

Methods

技能的价值通常要到未来任务才能观察到，生成奖励和使用奖励具有不同触发条件
三个条件：
- (q^1) 是否成功
- (q^2) 是否成功
- (q^2) 是否实际使用了 (q^1) 生成的技能

需要避免的误解：任务一成功不代表生成的技能具有复用价值- 技能被调用不代表它真的帮助了任务- 后续任务成功且实际复用，才构成对前序技能生成的正向证据

任务一奖励：

$$R^1 = r^1 + \mathbf{1}[r^1 = 1]\mathbf{1}[r^2 = 1]\mathbf{1}_{\text{skill}}(q^2 \mid q^1)$$

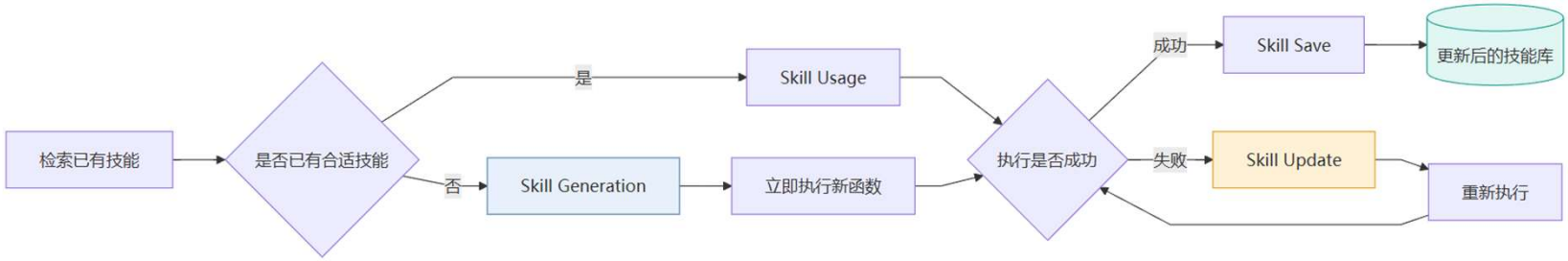
任务二奖励：

$$R^2 = r^2 + \mathbf{1}[r^2 = 1]\mathbf{1}_{\text{skill}}(q^2 \mid q^1)$$

精确解释：

- R^1 的技能生成奖励要求 $r^1 = 1$、$r^2 = 1$，且任务二使用任务一技能
- R^2 的技能使用奖励只要求任务二成功并实际使用技能
- 两项额外奖励的数值均为 1.0
- 未生成代码且直接终止时，论文设置 −1.0 惩罚

Methods



四类论文定义的操作:

- Skill Generation：定义新函数并立即执行
- Skill Save：函数成功执行后写入技能库
- Skill Usage：调用已有函数完成任务
- Skill Update：执行失败时修改技能并重试

Methods

SAGE 改变了智能体产生行为的方式，重要的不是语法变化，而是执行轨迹留下了有名字、可调用、可更新的能力接口

````
The Spotify password is an entry in the `passwords` list
with the account_name=spotify. I can use it to login now
with the provided email address.

```python
spotify_password = [account_password["account_name"] ==
"spotify" for account_password in passwords][0]["password"]

login_result = apis.spotify.login(username="jay@gmail.
com", password=spotify_password)

print(login_result)
```
````

基线交互格式

````
I can define a new function `spotify_login` to login
spotify with supervisor's email.

```python
def spotify_login(email: str) -> str:
    """
    Login to Spotify by automatically obtaining the
    supervisor's password and then proceeding with the
    login process.

    Args:
        email: supervisor's username presented as email.

    Returns:
        The login result of spotify.

    """
    passwords=apis.supervisor.show_account_passwords()

    spotify_password = [account_password["account_name"]
    == "spotify" for account_password in passwords][0]
    ["password"]

    login_result = apis.spotify.login(username=email,
    password=spotify_password)

    return login_result

login_result = spotify_login("jay@gmail.com")

print(login_result)
```
````

Skill Library Agent 交互格式

Methods

本文的 SAGE 构建在 GRPO（Shao et al., 2024）之上。对于每个查询 q，GRPO 首先从旧策略 $\pi_{\theta_{\text{old}}}$ 中采样一组输出 $\{o_1, \ldots, o_G\}$，然后最大化下列目标函数：

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim Q,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \left\{ \min\left[s_{i,t}\hat{A}_{i,t}, \text{clip}(s_{i,t}, 1-\epsilon, 1+\epsilon)\hat{A}_{i,t} \right] - \beta D_{\text{KL}} \right\} \right].$$

其中，

$$s_{i,t} = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})},$$

ϵ 为 clipping 比率，β 是策略模型 π_θ 与参考模型 π_{ref} 之间 KL 惩罚的系数。

$\hat{A}_{i,t}$ 是根据同一组输出之间的相对奖励计算得到的优势值。给定同一组输出对应的奖励

$$r = \{r_1, r_2, \ldots, r_G\},$$

优势函数定义为：

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

Methods

在收集 rollout 轨迹并计算相应奖励后，本文使用 **Skill Augmented GRPO for self-Evolution（SAGE）** 来训练技能库智能体。

遵循 Chen et al. (2025)，本文实际训练中：

- **不使用 KL divergence penalty**；
- 优势值**不再除以奖励标准差进行归一化。**

给定当前策略 π_θ 和旧策略 $\pi_{\theta_{\text{old}}}$，首先从原始数据分布中采样任务链：

$$(q^1, q^2) \sim Q.$$

对于每个任务对，执行 G 组 rollout：

$$\{\tau_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid (q^1, q^2)),$$

其中 τ_i 表示依次处理 (q^1, q^2) 时得到的 Sequential Rollout 轨迹。

为了区分轨迹 τ_i 中不同任务对应的输出，定义 $o_i^k \in \tau_i$ 为第 i 组中任务 q^k 的输出，其中 k 为任务链索引。

随后，通过最大化以下技能库集成 GRPO 目标来优化当前策略：

$$J_{\text{Agent}}(\theta) = \mathbb{E}_{(q^1,q^2)\sim Q,\{\tau_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|(q^1,q^2))} \left[\frac{1}{G} \sum_{i=1}^{G} \sum_{k=1}^{2} \frac{1}{|o_i^k|} \sum_{t=1}^{|o_i^k|} \min\left(s_{i,t}^k \hat{A}_i^k, \text{clip}(s_{i,t}^k, 1-\epsilon, 1+\epsilon) \hat{A}_i^k \right) \right].$$

其中：

$$s_{i,t}^k = \frac{\pi_\theta(o_{i,t}^k \mid q^k, M_i^k, o_{i,<t}^k)}{\pi_{\theta_{\text{old}}}(o_{i,t}^k \mid q^k, M_i^k, o_{i,<t}^k)},$$

$$\hat{A}_i^k = R_i^k - \text{mean}(\{R_i^k \mid i = 1, 2, \ldots, G\}).$$

$s_{i,t}^k$ 是重要性采样项；$\hat{A}_i^k$ 是根据 Skill-integrated Reward R_i^k 计算的优势；M_i^k 表示第 i 组中与任务 q_i^k 一起使用的技能库。

这里，M_i^1 为空集，而 M_i^2 包含智能体执行 q_i^1 时生成的技能。

由于技能库智能体需要通过多轮环境交互完成任务，$|o_i^k|$ 只计算 LLM 自身生成的内容；环境返回的 observation 在训练 loss 中被 mask 掉。

与 3.2.1 节的原始 GRPO 相比，本文方法最关键的差别是：由于引入 Sequential Rollout，期望是在**任务链层面**计算的。特别地，在同一组中，第二个任务的输出 o_i^2 实际来自不同的技能库 M_i^2，而原始 GRPO 中同组输出都来自相同查询条件。

Methods

SAGE 改变的不只是奖励，还改变了训练样本的时间跨度

维度	GRPO	SAGE
Rollout 单元	单个任务	相关任务链
策略条件	$q, o_{<t}$	$q, M, o_{<t}$
奖励	Outcome	Outcome + Skill
优化范围	当前任务	链内多个任务
技能状态	无	随任务变化

$$\text{Base LLM} \rightarrow \text{SFT} \rightarrow \underbrace{\text{Sequential Rollout} \rightarrow \text{Skill-integrated Reward} \rightarrow \text{GRPO Update}}_{\text{SAGE RL}}$$

论文使用 Claude Sonnet 3.5 V2 作为专家，在技能库智能体框架内执行 rejection sampling

Experiments

Table 1: Task Performance of SAGE compared with various baselines. Results of Methods marked with "*" are taken from (Chen et al., 2025). In these cases, Avg. Steps and Tokens were not reported, thus denoted by "- -".

Base Model	Methods	Test Normal				Test Challenge			
		TGC	SGC	Avg. Steps	Avg. Tokens	TGC	SGC	Avg. Steps	Avg. Token
Training Free Methods									
GPT-4o	ReAct*	48.8	32.1	- -	- -	30.2	13.0	- -	- -
OpenAI o1	ReAct*	61.9	41.1	- -	- -	36.7	19.4	- -	- -
Claude Sonnet 3.5 V2	ReAct	57.1	41.1	15.7	1,542	49.2	28.8	21.8	2,084
Qwen2.5 32B Instruct	ReAct*	39.2 ± 3.5	18.6 ± 2.0	- -	- -	21.0 ± 1.4	7.5 ± 1.2	- -	- -
RL without Skill Library									
Qwen2.5 32B Instruct	LOOP*	71.3 ± 1.3	53.6 ± 2.2	- -	- -	45.7 ± 1.3	26.6 ± 1.5	- -	- -
	GRPO	69.2 ± 2.7	51.8 ± 5.8	16.4 ± 0.2	3,613 ± 200	40.7 ± 1.5	26.9 ± 1.5	21.9 ± 0.1	5,211 ± 65
Our Approach									
Qwen2.5 32B Instruct	Skill Library Agent	30.7 ± 3.1	19.6 ± 1.4	13.4 ± 0.4	2,988 ± 73	15.3 ± 1.7	7.0 ± 1.2	18.7 ± 0.4	4,803 ± 117
	+ SFT	55.2 ± 1.5	41.7 ± 1.7	**11.4 ± 0.5**	**1,340 ± 65**	37.2 ± 1.2	20.9 ± 1.8	**16.2 ± 0.3**	1,909 ± 80
	+ SAGE	**72.0 ± 1.5**	**60.7 ± 1.5**	12.1 ± 0.2	1,475 ± 127	**50.1 ± 2.0**	**32.4 ± 3.7**	17.3 ± 0.3	**1,807 ± 29**

AppWorld 提供了高质量可执行环境，模拟了 9 个日常应用（例如 Spotify）、总计 457 个 API，并包含 100 多个模拟用户
AppWorld 共包含 750 个任务（250 个场景），分为：Train：105；Dev：60；Test-Normal：168；Test-Challenge：417。

所有训练步骤——包括 SFT 和之后的 SAGE——都只使用 Train 集。Dev 集用于选择最佳 checkpoint。最终在 Test-Normal 和 Test-Challenge 上评估。

Experiments

虽然 SAGE 已经在 AppWorld 达到很强的性能，但仍需进一步了解技能库在评估过程中究竟是如何被使用的。

本文使用四个指标：

- Skill Usage Rate：在拥有技能库的样本中，实际使用技能的比例；
- Success Skill Usage Rate：在已经使用技能的样本中，最终成功完成任务的比例；
- Skill Library Size：技能库中生成的技能总数；
- Used Skill Num：技能库中实际被使用过的技能数量。

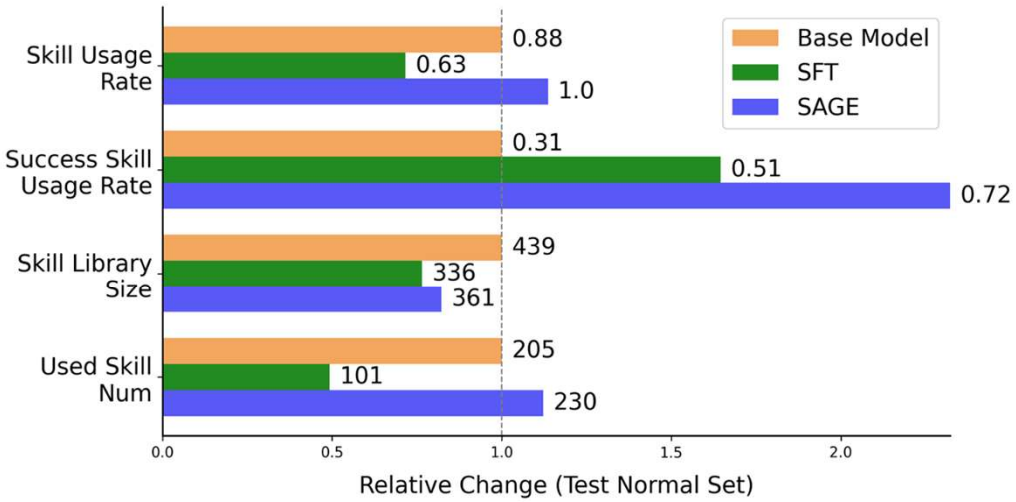


Figure 2: Analysis of Skill Usage Patterns. Performance metrics are shown as ratios relative to Base Model baseline, with numerical values annotated.

量化结果表明：

- SAGE 生成的 361 个技能中，有 **356 个（约 99%）** 属于包含多个 API 调用的复杂技能；
- Base Model 生成的 439 个技能中，有 **362 个（约 82%）** 属于复杂技能。

这进一步说明 SAGE 能生成更复杂、更高质量的技能，因此其技能库虽然更小，却显著更加有效。

Experiments

Table 2: Stepwise performance of skill library agent compared with no skills involved.

Step	With Skill	Test Normal			
		TGC	SGC	Avg. Steps	Avg. Tokens
Skill Library Agent	✓	30.7 ± 3.1	19.6 ± 1.4	13.4 ± 0.4	2,988 ± 73
	×	34.7 ± 3.0	14.9 ± 3.1	16.4 ± 0.5	3,704 ± 139
SFT	✓	55.2 ± 1.5	41.7 ± 1.7	**11.4 ± 0.5**	**1,340 ± 65**
	×	54.8 ± 2.1	39.9 ± 2.2	13.5 ± 0.1	1,611 ± 11
SAGE	✓	**72.0 ± 1.5**	**60.7 ± 1.5**	12.1 ± 0.2	1,475 ± 127
	×	71.4 ± 0.5	54.8 ± 0.8	16.0 ± 0.2	1,937 ± 81

所有阶段在使用技能时都能获得更高 SGC、更少平均步骤和更少 token。

这表明技能库即使在没有额外训练的情况下，也能提升跨任务场景性能和执行效率。

纯 Skill Library Agent 在使用技能后 TGC 反而下降。作者认为这是因为基础模型还不擅长准确使用技能，错误使用技能可能直接导致单任务失败

Table 3: SAGE with different retrieval methods.

Retrieval Methods	Test Normal			
	TGC	SGC	Avg. Steps	Avg. Tokens
Same Scenario	**72.0 ± 1.5**	**60.7 ± 1.5**	12.1 ± 0.2	1,475 ± 127
Query N-gram	**72.0 ± 2.0**	60.1 ± 1.7	12.7 ± 0.3	1,466 ± 101
Query Embedding	69.6 ± 1.6	59.5 ± 2.2	**11.8 ± 0.2**	**1,335 ± 63**
Skill Embedding	66.3 ± 0.7	56.0 ± 0.8	14.5 ± 0.3	1,692 ± 10

实验表明，技能检索策略对最终性能影响很大。经过合理设计的检索机制可以接近、甚至在某些效率指标上超过理想的 Same Scenario 设置

Experiments

Table 4: SAGE with different task chain constructions.

Task Chain Construction	Test Normal TGC	SGC	Avg. Steps	Avg. Tokens
Same Scenario	72.0 ± 1.5	60.7 ± 1.5	**12.1 ± 0.2**	**1,475 ± 127**
Similarity Search	**72.8 ± 2.3**	**62.5 ± 2.9**	13.0 ± 0.3	1,535 ± 114

Table 5: SAGE with different reward designs.

Reward Design	Test Normal TGC	SGC	Avg. Steps	Avg. Tokens
Skill-integrated	**72.0 ± 1.5**	**60.7 ± 1.5**	**12.1 ± 0.2**	1,475 ± 127
Outcome-based	69.8 ± 1.0	55.4 ± 1.4	13.1 ± 0.2	1,469 ± 61
Chain-based	67.9 ± 2.1	56.6 ± 1.6	15.7 ± 0.3	**1,361 ± 58**

Table 6: SAGE initialized with different methods.

Initialization Method	Extra Data	Test Normal TGC	SGC	Avg. Steps	Avg. Tokens
Base Model	×	40.7 ± 2.3	25.6 ± 0.7	**11.9 ± 0.1**	2,532 ± 93
Self Distillation	×	66.5 ± 1.6	53.6 ± 5.3	13.1 ± 0.4	2,321 ± 103
RL Warm-Up	×	68.3 ± 0.7	55.3 ± 3.8	16.0 ± 0.3	2,556 ± 62
SFT	✓	**72.0 ± 1.5**	**60.7 ± 1.5**	12.1 ± 0.2	**1,475 ± 127**

Table 7: Performance of SFT initialized GRPO.

Method	Test Normal TGC	SGC	Avg. Steps	Avg. Tokens
GRPO	69.2 ± 2.7	51.8 ± 5.8	16.4 ± 0.2	3,613 ± 200
SFT + GRPO	66.1 ± 1.3	51.2 ± 0.8	12.8 ± 0.1	**1,284 ± 18**
SAGE	**72.0 ± 1.5**	**60.7 ± 1.5**	**12.1 ± 0.2**	1,475 ± 127

Thanks