SPC: Evolving Self-Play Critic via Adversarial Games for LLM Reasoning

Jiaqi Chen[1] **Bang Zhang**[*] **Ruotian Ma**[2†] **Peisong Wang**[3]
Xiaodan Liang[4] **Zhaopeng Tu**[2] **Xiaolong Li**[2] **Kwan-Yee K. Wong**[1†]
[1]The University of Hong Kong [2]Tencent
[3]Tsinghua University [4]MBZUAI
Project: https://chen-judge.github.io/SPC/

SPC: Evolving Self-Play Critic via Adversarial Games for LLM Reasoning

Topic： 本文试图解决如何评估大模型推理过程中每一步的可靠性和准确性的问题。具体而言，它关注于以下几个关键问题：

每一步高质量标注数据的获取困难：评估LLM的推理步骤需要大量的标注数据，但获取这些数据既困难又成本高昂。

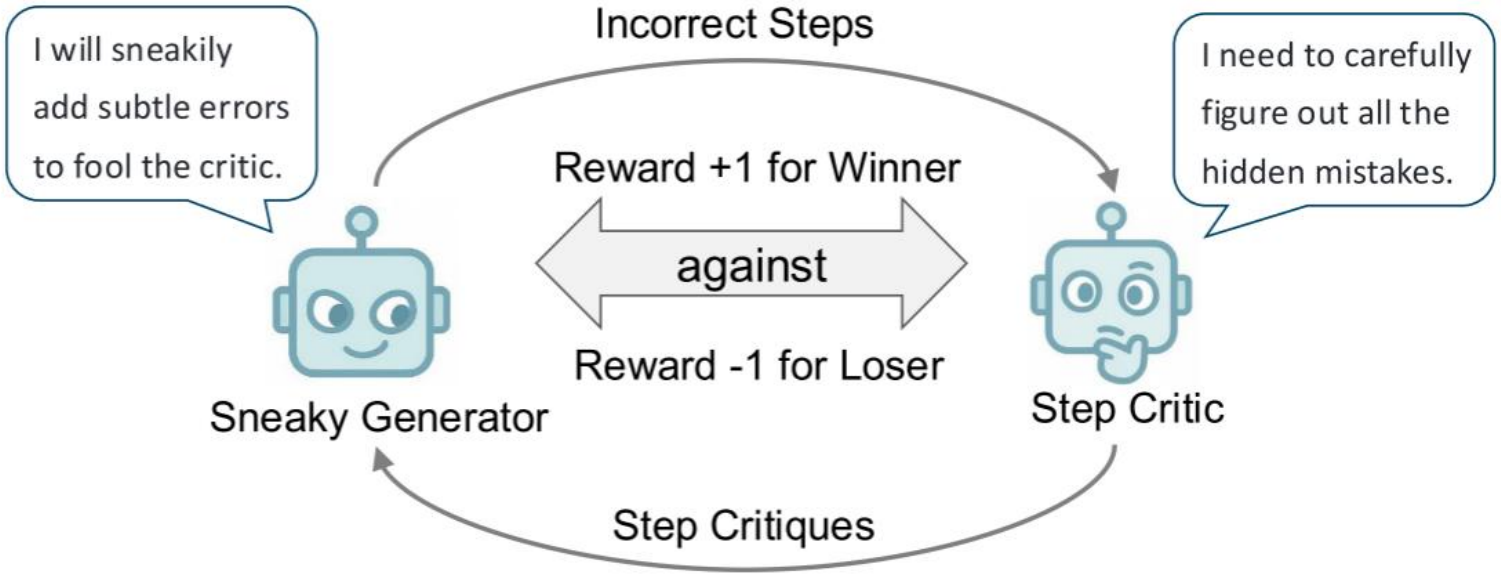
现有验证模型的局限性：现有的验证模型主要分为结果验证器（ORMs）和过程验证器（PRMs），它们通常只能提供解决**方案级别**的评分，而缺乏对推理步骤的细粒度分析和评估。此外，这些验证器通常只能提供标量分数，而**不能提供自然语言形式**的反馈，这限制了它们在评估LLM输出方面的有效性。

为了解决这些问题，本文提出了一种名为SelfPlay Critic（SPC）的方法，通过**对抗性自玩**来自动生成数学推理的训练数据，从而无需人工标注即可训练一个能够评估推理步骤正确性的**批评模型**。这种方法不仅能够提高批评模型的性能，还能够将其应用于**指导LLM在测试时的搜索过程**，从而提高LLM的推理性能。

SPC: Evolving Self-Play Critic via Adversarial Games for LLM Reasoning

总体框架

- **Sneaky Generator (S)**：将正确的推理步骤转换为错误的步骤，目标是生成难以被检测到的错误步骤。
- **Step Critic (C)**：正确地识别生成器产生的错误并提供批评。



这种相反的优化目标使生成器和批评者能够不断提高其性能，实现迭代的自我进化。

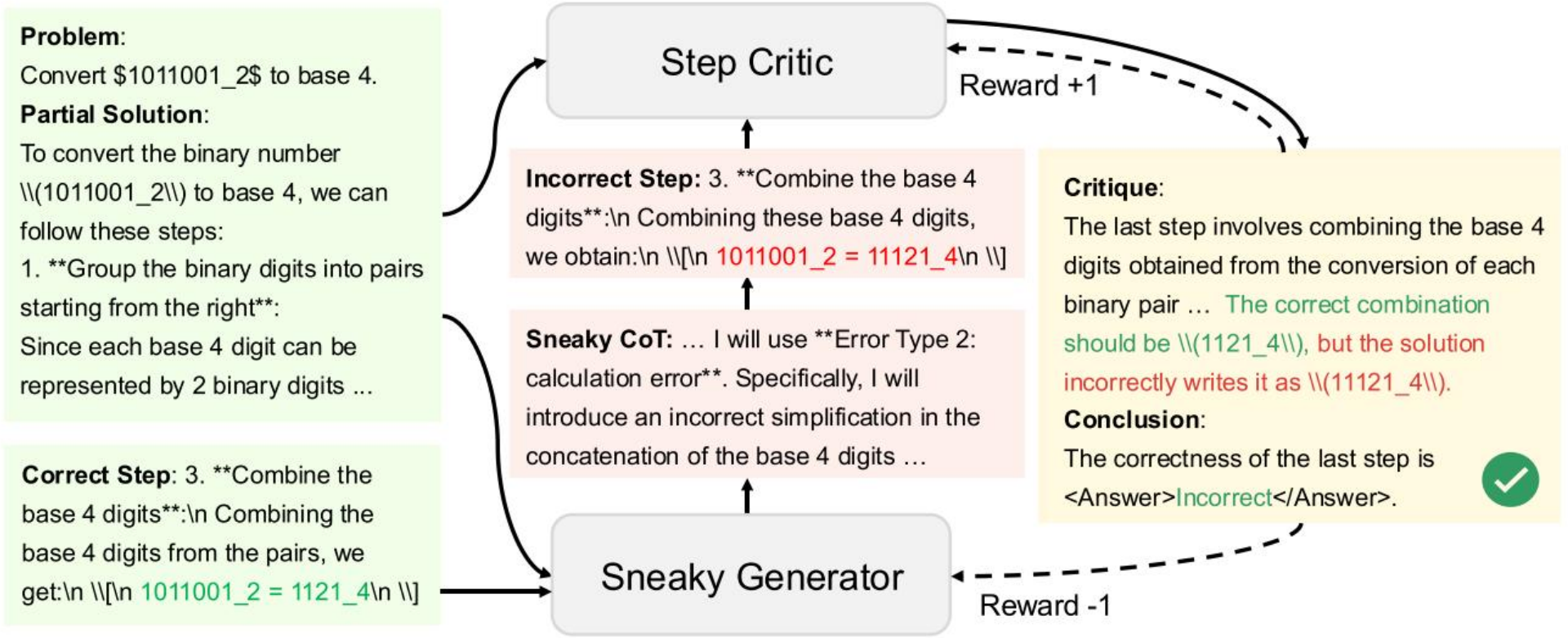
$$R_{sneaky} = \begin{cases} 1, & \texttt{Sneaky Generator Wins} \\ -1, & \texttt{Sneaky Generator Loses} \end{cases} \tag{2}$$

$$R_{critic} = \begin{cases} 1, & \texttt{Critic Wins} \\ -1, & \texttt{Critic Loses} \end{cases} \tag{3}$$

SPC: Evolving Self-Play Critic via Adversarial Games for LLM Reasoning

对抗性游戏

随机选择一个正确的步骤以及该步骤之前的解，并将它们输入到生成器中，生成器首先选择一个预定义的错误类型，然后将正确的步骤转换为错误的步骤，如果这个错误步骤**会显著影响模型推理的正确性**那他就是**有效的**。然后，将生成的不正确步骤提供给批评者以进行错误检测。如果批评者成功地识别了错误，它将获得+1的奖励，而生成者将获得-1的奖励。如果批评者被欺骗了，批评者和生成者将分别获得-1和+1的奖励。通过对抗性游戏生成强化学习正负样本。



SPC: Evolving Self-Play Critic via Adversarial Games for LLM Reasoning

模型初始化

- **初始化Sneaky Generator(S)**：使用PRM800K数据集中的正确-错误步骤对，具体来说选择一个预定义的错误类型，通过GPT-4生成从正确步骤到错误步骤的转换过程，形成一个行为克隆数据集。再使用上述数据集对Qwen2.5-7B-Instruct进行微调，得到初始的Sneasky Generator。
- **初始化Step Critic (C)**：使用PRM800K数据集中的问题、部分解决方案和混合的正确/错误步骤（1:1），通过DeepSeek-R1-Distill-Qwen-7B生成长篇批评，再通过GPT-4将其改写为简洁标准化的批评,再用上述数据集对DeepSeek-R1-Distill-Qwen-7B进行微调。

生成步骤的自动验证规则

首先采样一个原始步骤，然后使用生成器生成错误步骤，随后将其分别输入到solver模型，如果原始步骤的正确率大于或等于75%，而错误的步骤的成功率为0%，那么，这个步骤就是有效的。

SPC: Evolving Self-Play Critic via Adversarial Games for LLM Reasoning

离线强化学习

先收集对抗游戏数据：采用多种LLM求解器为每个问题生成原始的分步解决方案，随机选择步骤生成错误步骤进行对抗游戏，筛选有效数据。

- **Sneaky Generator(S)**：训练数据包括以下三种样本，且这三种样本的比例是1:1:1。正样本：既能影响大型语言模型（LLM）的成功率，又能欺骗评估器的步骤。负样本类型一：未能影响LLM成功率的步骤。负样本类型二：虽然成功影响了LLM的成功率，但未能欺骗评估器的步骤。
- **Step Critic (C)**：训练数据包括，来自正确解决方案的正确步骤以及生成器生成的错误步骤，正负样本各占总样本数的一半。

迭代训练

在每次迭代中，使用新更新的策略重新进行对抗游戏，生成新的训练数据。

非对称进化策略：观察到初始的生成器（S0）比初始的评估器（C0）弱，导致胜率不平衡，通过同步迭代得到的S1甚至比C1更弱。因此，让S1与C0进行对抗，以生成第二轮数据。这使得在第二轮训练的C2能够进一步提升性能。这种策略类似于人类通过与水平相当的对手下棋来提高自己的棋艺。

SPC: Evolving Self-Play Critic via Adversarial Games for LLM Reasoning

指导LLM的测试时搜索

传统的处理过程奖励模型（PRMs）需要对完整的解决方案的每一步进行评分。然而，这种方法存在一个明显的问题：一旦在推理过程中出现第一个错误步骤，LLM应该立即纠正这个错误，而非继续生成后续有缺陷的推理步骤。

SPC提出了一种新方法，直接利用评估器（critic）协助LLM进行推理步骤的搜索。

在测试过程中，通过使用“\n\n”来控制LLM的输出，使其每次只输出一个推理步骤。评估器会验证每一步的正确性：

- 如果步骤正确，推理过程继续进行；
- 如果步骤错误，LLM需要重新生成该步骤（最多尝试五次，之后则跳过该步骤）。

SPC: Evolving Self-Play Critic via Adversarial Games for LLM Reasoning

实验结果

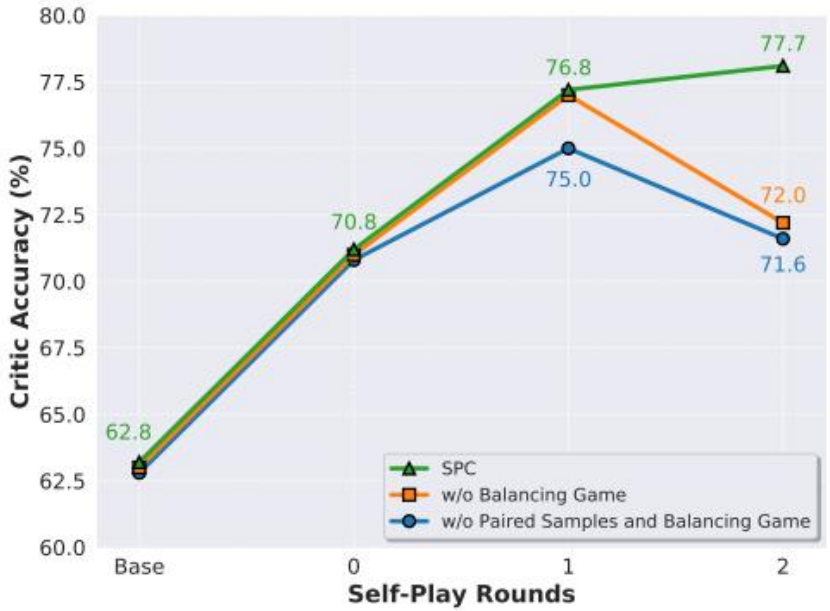
Models	PRM800K				DeltaBench			
	Average	HarMean	Correct	Error	Average	HarMean	Correct	Error
Process Reward Models (PRMs)								
Math-Shepherd-PRM-7B [26]	50.0	49.5	55.2	44.8	53.3	14.3	7.69	**98.8**
Qwen2.5-Math-7B-PRM800K [27]	73.6	73.6	74.4	72.8	58.5	41.3	**90.1**	26.8
Prompting LLMs as Critic Models								
Llama-3.1-8B-Instruct [10]	51.9	30.5	18.6	85.2	49.1	6.38	3.30	95.0
Llama-3.1-70B-Instruct [10]	54.6	38.9	25.3	83.9	44.6	20.3	11.7	77.5
Qwen2.5-7B-instruct [12]	52.8	37.2	24.1	81.6	48.2	33.8	21.8	74.7
Qwen2.5-32B-instruct [12]	59.0	50.5	36.6	81.4	44.7	33.0	21.8	67.6
GPT-4o [6]	68.5	68.4	70.3	66.6	49.9	48.7	42.0	57.9
DeepSeek-R1-Distill-Qwen-7B [21]	71.4	71.2	67.3	75.5	50.9	50.6	54.9	46.9
Our Critic Models								
SPC (Round 0)	71.0	70.8	67.8	74.2	54.9	53.5	45.9	64.0
SPC (Round 1)	72.8	70.3	59.4	**86.1**	58.8	57.3	68.4	49.3
SPC (Round 2)	**75.8**	**75.8**	**74.8**	76.9	**60.5**	**59.5**	68.2	52.8

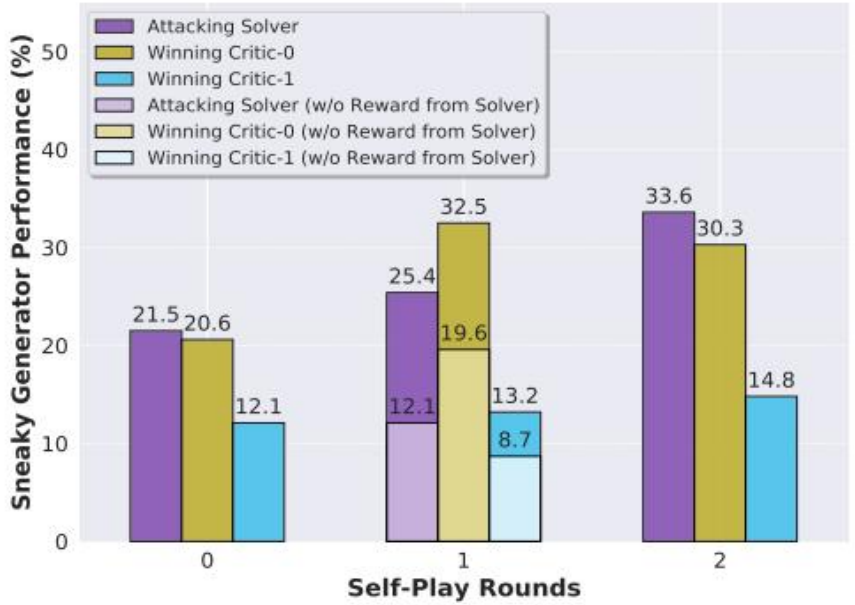
随着迭代次数的增加，指标上升。

SPC优于所有的PRM。

Models	GSM8K	MATH	Olympiad-Bench	Omni-MATH	Average
Process Reward Models (PRMs)					
Math-Shepherd-PRM-7B [26]	58.0	58.4	68.0	64.1	62.1
Qwen2.5-Math-7B-PRM800K [27]	77.0	72.9	66.9	62.1	69.7
Prompting LLMs as Critic Models					
Llama-3.1-8B-Instruct [10]	59.5	57.7	53.6	53.9	56.2
Llama-3.1-70B-Instruct [10]	67.2	62.8	61.7	61.9	63.4
Qwen2.5-7B-Instruct [12]	64.2	64.0	62.1	60.8	62.8
Qwen2.5-32B-Instruct [12]	76.2	68.1	68.9	63.9	69.3
GPT-4o [6]	75.5	70.5	70.0	64.5	70.1
DeepSeek-R1-Distill-Qwen-7B [21]	79.0	**81.3**	73.4	67.3	75.2
Our Critic Models					
SPC (Round 0)	78.0	74.1	67.8	63.2	70.8
SPC (Round 1)	82.0	80.3	74.8	**70.3**	76.8
SPC (Round 2)	**84.2**	80.8	**76.5**	69.2	**77.7**

SPC: Evolving Self-Play Critic via Adversarial Games for LLM Reasoning





对于每个成功转换的错误步骤，评估器会生成四个批评，从而形成正负样本对，这在RL训练中更为有效，使其的性能从70.8%提升到76.8%。而如果不构建正负样本对，评估器的性能仅能达到75.0%。

对于round2有两种策略：

（1）混合Sneaky-1和Critic-1对抗生成的数据与第一轮的数据混合

（2）混合Sneaky-1和Critic-0对抗生成的数据与第一轮的数据，来训练Critic-2，性能上升到77.7%。

w/o Rward from Solver:不将对solver的失败攻击添加为负样本，仅使用成功生成的错误步骤来构建样本以训练Sneaky，这样会影响生成器的性能。在成功攻击的样本中，能够欺骗评估器的比例也很低，对Critic-0的胜率仅为19.6%。这意味着生成器不仅需要成功生成错误步骤来欺骗评估器，还需要确保这些错误步骤能实际影响solver的成功率。

SPC: Evolving Self-Play Critic via Adversarial Games for LLM Reasoning

一些后续的思考：

1. 多领域推理步骤评估

- 研究问题：当前的 SPC 主要集中在数学推理领域，是否可以将其扩展到其他领域，开发针对不同领域的错误类型和转换策略，以生成更多样化的错误步骤，从而训练出更通用的批评模型。

2. 与其他的强化学习方法结合，比如多智能体强化学习，以及加入工具调用来提高模型的泛化性。

Visual Agentic Reinforcement Fine-Tuning

Ziyu Liu[1,2] **Yuhang Zang**[2✉] **Yushan Zou**[4] **Zijian Liang**[1]
Xiaoyi Dong[2,3] **Yuhang Cao**[2] **Haodong Duan**[2] **Dahua Lin**[2,3] **Jiaqi Wang**[2✉]
[1]Shanghai Jiaotong University [2]Shanghai Artificial Intelligence Laboratory
[3]The Chinese University of Hong Kong [4]Wuhan University
liuziyu77@sjtu.edu.cn, {zangyuhang, wangjiaqi}@pjlab.org.cn
https://github.com/Liuziyu77/Visual-RFT/tree/main/Visual-ARFT

Visual Agentic Reinforcement Fine-Tuning

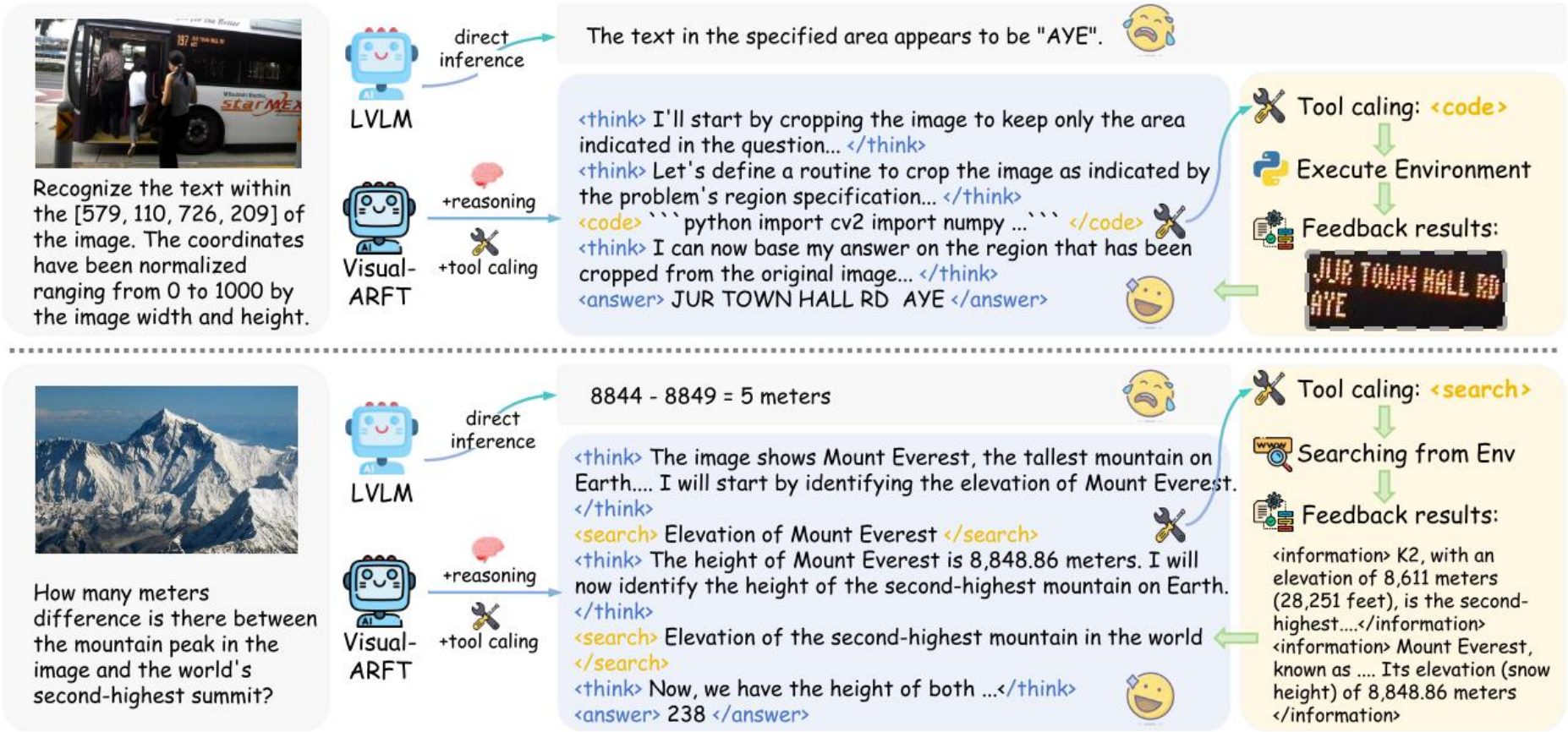
近期大型语言模型（LLMs）的发展推动了代理系统的发展，这些模型能够规划、推理并与外部工具交互以解决复杂任务。例如，OpenAI的o3展示了对文本和视觉模态的工具增强推理的原生支持。

本文通过提出 Visual Agentic Reinforcement Fine-Tuning (Visual-ARFT) 框架来解决如何为大型视觉-语言模型（LVLMs）赋予灵活且适应性强的推理能力的问题，尤其是**使用外部工具**的能力，旨在使开源的LVLMs能够：

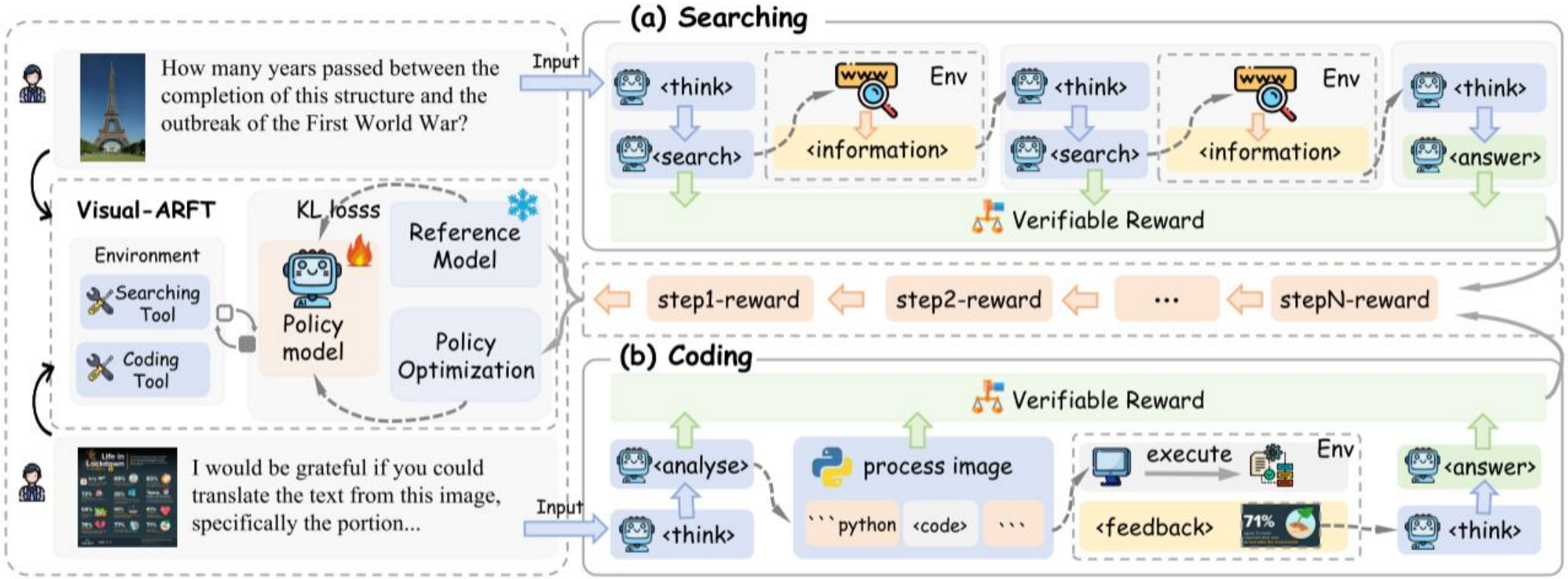
1. 浏览网页以获取实时信息更新。
2. 编写并执行代码来处理和分析输入图像，例如通过裁剪、旋转等图像处理技术。
3. 在多模态任务中，通过推理、任务分解和工具调用，解决复杂的视觉问答问题。

Visual Agentic Reinforcement Fine-Tuning

本文专注于两个多模态设置：代理搜索和代理编码。例如这张图（顶部）编写和执行Python代码以准确读取指定图像区域内的文本，以及（底部）使用互联网搜索来回答多跳问题。



Visual Agentic Reinforcement Fine-Tuning



给定多模态输入，策略模型生成包括中间推理步骤（例如<think>）和行动决策，Visual-ARFT训练模型自主分析和分解多跳查询，并通过与Web搜索引擎的迭代交互来解决这些问题。在代理编码设置中，输入图像可能遭受视觉退化或包含冗余内容，Visual-ARFT旨在通过训练模型主动生成以视觉问题为条件的可执行代码来处理此问题。

Visual Agentic Reinforcement Fine-Tuning

reward设计

本文设计了基于规则的可验证奖励，用于评估模型的工具使用情况和最终答案，每一步都有奖励。

格式奖励： $R_{\text{format}}(o) = \mathbb{I}\left[o \text{ contains valid tags}\right].$

要求模型生成严格遵循预定义格式的输出。

准确性奖励：

对于最终答案使用F1分数作为奖励。对于search任务使用语义相似度奖励，反映模型是否真正理解了检索目标的真实意图。对于code任务只要模型输出的内容是可执行的代码块，就给予奖励（即奖励值为1），而不直接监督代码的具体内容，避免模型过于依赖固定的代码模板。

$$R_{\text{acc}}(q,o) = \begin{cases} R_{\text{F1}}(o_{\text{ans}}, a), & \text{if } o \text{ is the final answer } (<\texttt{answer}>), \\ R_{\text{sem}}(o_{\text{search}}, s), & \text{if } o \text{ is a search query } (<\texttt{search}>), \\ 1, & \text{if } o \text{ is a code block } (<\texttt{code}>). \end{cases}$$

$$R_{\text{total}}(q,o) = R_{\text{format}}(o) + R_{\text{acc}}(q,o).$$

Visual Agentic Reinforcement Fine-Tuning

构建Multimodal Agentic Tool Bench（MAT）

为了评估模型的多模态代理推理和工具使用能力，论文提出了多模态代理工具基准（MAT），包括两个子任务：**MAT-Search** 和 **MAT-Coding**。

1. MAT-Search

任务描述：MAT-Search 评估模型的代理搜索能力，要求模型通过多跳推理和外部知识检索来回答复杂的多模态问题。

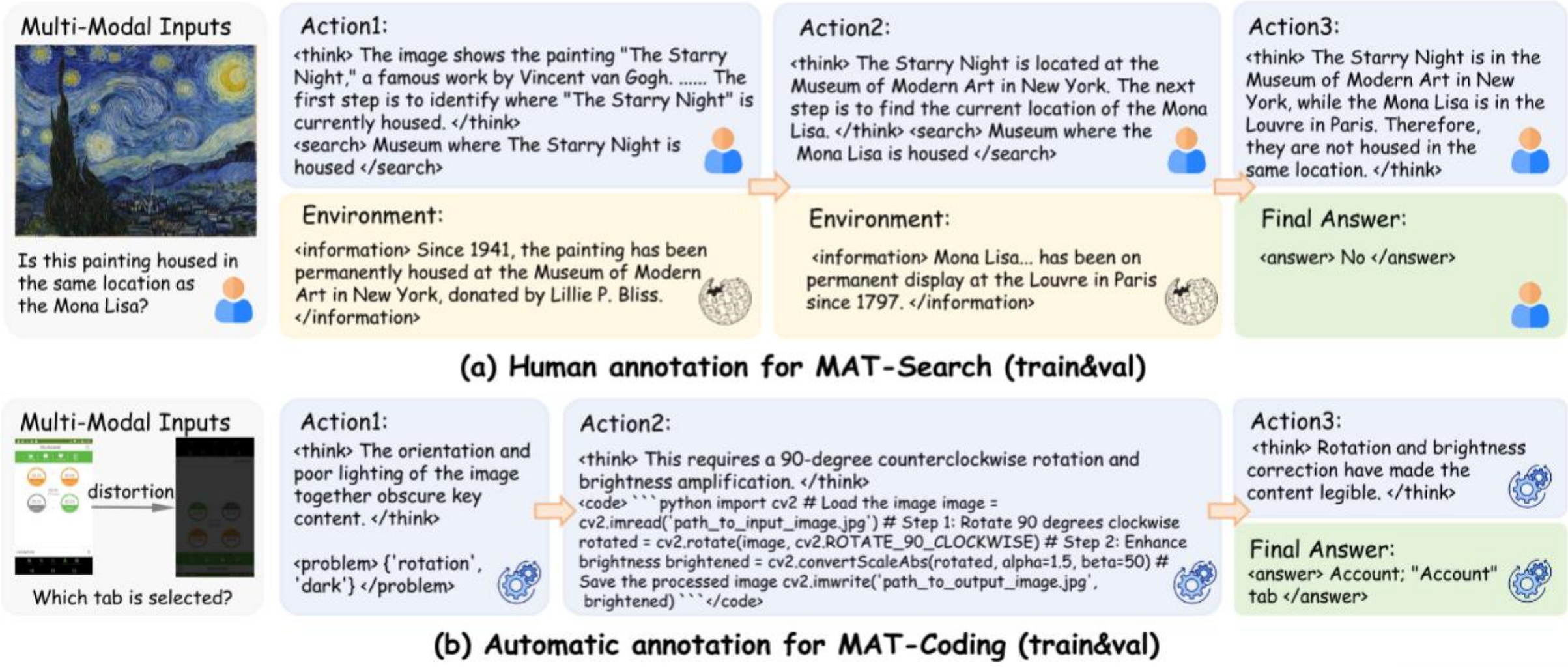
数据构建：MAT-Search 数据集通过手动标注构建，包含 150 个高质量的多模态多跳 VQA 示例，每个示例都经过人工审核以确保其质量和适用性。

2. MAT-Coding

任务描述：MAT-Coding 评估模型的代理编码能力，要求模型通过编写和执行代码来处理图像，从而解决视觉问题。

数据构建：MAT-Coding 数据集通过自动化流程生成，包含 200 个测试样本，这些样本经过人工审核以确保其质量和多样性。

Visual Agentic Reinforcement Fine-Tuning



Visual Agentic Reinforcement Fine-Tuning

Models	Reasoning with Tools	MAT-Coding						MAT-Search					
		Simple		Hard		Avg		Simple		Hard		Avg	
		F1	*EM*	*F1*	*EM*	*F1*	*EM*	*F1*	*EM*	*F1*	*EM*	*F1*	*EM*
GPT-4o [10]	✗	47.12	38.57	27.57	15.38	34.41	23.5	68.55	61.33	53.61	42.67	61.08	52.00
OpenAI-o3 [29]	✓	70.38	65.38	75.00	70.59	72.99	68.33	79.72	70.67	63.74	52.00	71.73	61.33
LLaVa-v1.5-7B [24]	✗	19.50	12.86	9.30	5.38	12.87	8.00	56.55	52.00	30.32	25.33	43.44	38.67
LLaVa-Next-7B [17]	✗	30.78	17.14	17.11	10.00	21.89	12.5	63.27	56.00	38.75	29.33	51.01	42.67
LLaVa-OneVision-7B [16]	✗	39.86	28.57	16.05	11.54	24.38	17.5	61.78	54.67	31.66	26.67	46.72	40.67
Xcomposer2.5 [51]	✗	36.06	22.86	19.90	10.77	25.56	15.0	60.16	54.67	31.93	28.00	46.04	41.33
InternVL2.5-8B [3]	✗	39.48	28.57	26.62	13.85	31.12	19.00	61.72	53.33	41.69	33.33	51.70	43.33
Qwen2.5-VL-3B [1]	✗	46.29	35.71	17.98	13.85	27.89	21.50	57.54	50.67	33.11	26.67	45.32	38.67
+ Visual-ARFT	✓	49.78	40.00	28.42	13.08	35.90	22.50	56.41	50.67	45.55	36.00	50.98	43.33
Δ	–	+3.49	+4.29	+10.44	-0.78	+8.01	+1.0	-1.13	+0.0	+12.44	+9.33	+5.66	+4.66
Qwen2.5-VL-7B [1]	✗	55.23	40.00	19.67	11.54	32.12	21.50	67.40	61.33	39.59	32.00	53.49	46.67
+ Visual-ARFT	✓	60.10	51.43	45.60	25.38	50.68	34.50	71.78	66.67	55.77	44.00	63.77	55.33
Δ	–	+4.87	+11.43	+25.93	+13.84	+18.56	+13.00	+4.38	+5.37	+16.18	+12.00	+10.28	+8.66

Qwen2.5-VL-7B和Qwen2.5-VL-3B模型使用VisualARFT，性能都提高了，并且优于gpt-4o。o3通过工具调用，性能超过了gpt-4o。说明通过引入代理功能，能使LVLM能够推理，调用工具，并有效地解决复杂的视觉问题。

Visual Agentic Reinforcement Fine-Tuning

Models	Reasoning with Tools	2Wiki [9]		HotpotQA [46]		MuSiQue [44]		Bamboogle [32]		Avg	
		F1	*EM*	*F1*	*EM*	*F1*	*EM*	*F1*	*EM*	*F1*	*EM*
Qwen-2.5-7B-Instruct [1]											
+ Direct Inference	✗	–	25.00	–	18.30	–	3.10	–	12.00	–	14.60
+ CoT	✗	–	11.00	–	9.20	–	2.20	–	23.20	–	11.40
+ IRCoT	✗	–	14.90	–	13.30	–	7.20	–	22.40	–	14.45
+ RAG	✗	–	23.50	–	29.90	–	5.80	–	20.80	–	20.00
+ Search-o1 [18]	✓	–	17.60	–	18.70	–	5.80	–	29.60	–	17.93
+ Search-R1 [13]	✓	–	41.40	–	**37.00**	–	14.60	–	36.80	–	32.45
+ ZeroSearch [39]	✓	–	43.12	–	29.21	–	**19.72**	–	35.20	–	31.81
Qwen2.5-VL-3B-Instruct [1]											
+ Direct Inference	✗	31.60	26.40	22.90	15.84	9.64	2.36	10.82	5.60	18.74	12.55
+ RAG	✗	35.14	26.92	35.08	23.67	15.08	7.61	28.37	19.20	28.42	19.35
+ Visual-ARFT	✓	49.77	39.13	41.33	30.10	19.10	11.58	49.01	38.40	39.80	29.80
Δ	–	+18.17	+12.73	+18.43	+14.26	+9.46	+9.22	+38.19	+32.80	+21.06	+17.25
Qwen2.5-VL-7B-Instruct [1]											
+ Direct Inference	✗	27.90	22.15	25.35	17.35	9.45	2.61	14.07	5.60	19.19	11.93
+ RAG	✗	31.58	21.74	36.33	24.09	15.36	7.61	27.60	19.20	19.82	12.73
+ Visual-ARFT	✓	**63.09**	**51.99**	**48.00**	36.48	**22.71**	14.07	**60.15**	**48.80**	**48.49**	**37.84**
Δ	–	+35.19	+29.84	+22.65	+19.13	+13.26	+11.46	+46.08	+43.20	+29.30	+25.91

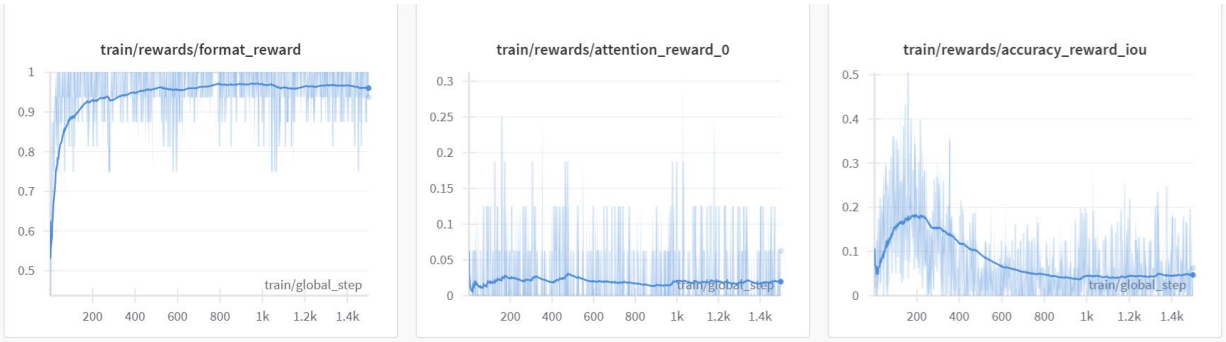
Table 4: **Ablation Study** with Direct inference/CoT/RAG baselines on MAT-Search.

Task	Method	Simple		Hard		Avg	
		F1	*EM*	*F1*	*EM*	*F1*	*EM*
Qwen2.5 -VL -3B	Direct	57.54	50.67	33.11	26.67	45.32	38.67
	CoT	37.84	30.67	31.03	24.00	34.43	27.33
	RAG	49.42	45.33	39.07	32.00	44.25	38.67
	Visual-ARFT	56.41	50.67	45.55	36.00	50.98	43.33
Qwen2.5 -VL -7B	Direct	67.40	61.33	39.59	32.00	53.49	46.67
	CoT	57.57	49.33	46.65	32.00	52.11	40.67
	RAG	59.14	56.00	42.44	36.00	50.79	46.00
	Visual-ARFT	71.78	66.67	55.77	44.00	63.77	55.33

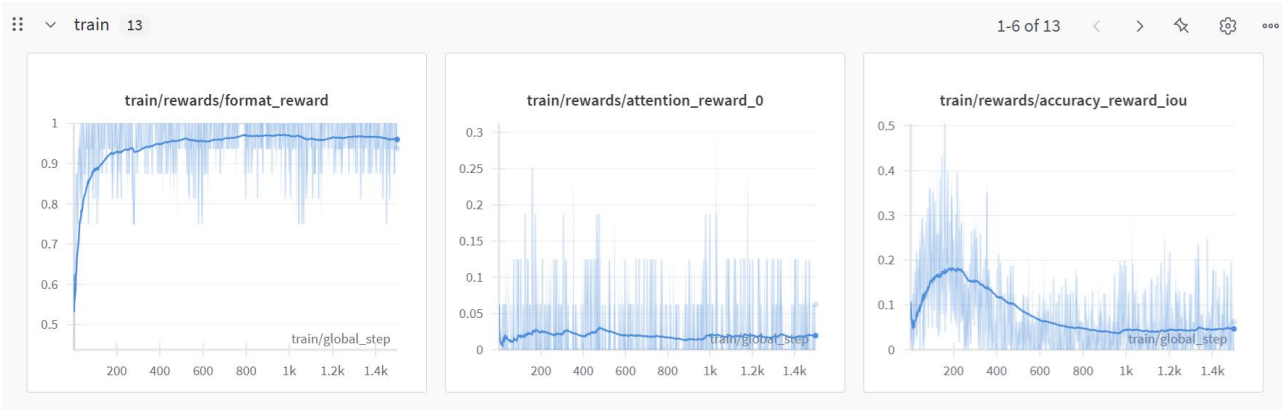
为了进一步评估 Visual-ARFT 的泛化能力，在几个纯文本多挑问答基准测试上进行了实验，并且与cot,rag等方法进行对比，说明其泛化能力较强。

梯度实验结果

0.2

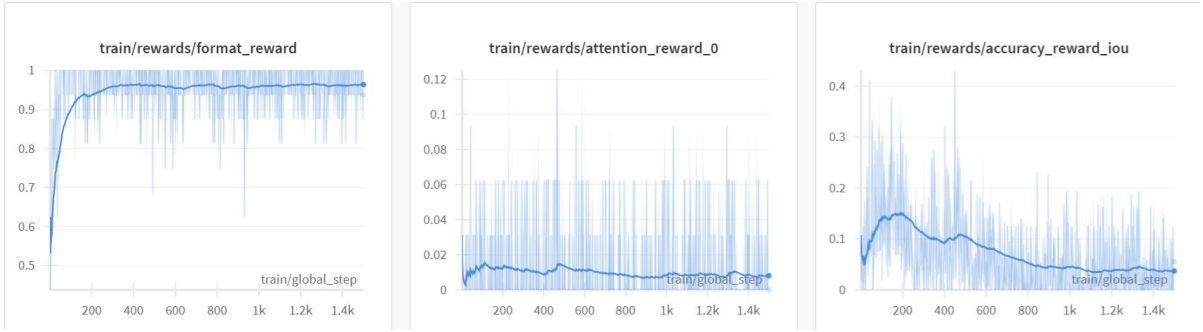


0.3



0.4

0.5



思考的结构

<cot>..analys and choose a type..</cot>

<serach>....</search>

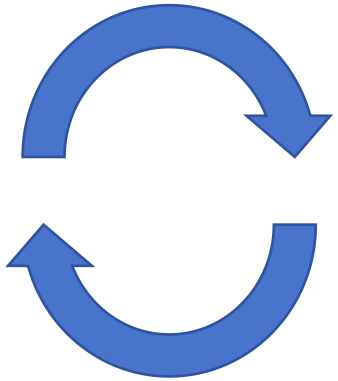
...

<code>....</code>

<think>....</think>

<cheat>do some cheating </cheat>

generator



refiner

<think>....</think>

<serach>....</search>

...

<code>....</code>

<think>....</think>

<answer>T/F<answer>

<refine>advice</refine>

trained-refiner

solver

思考的：有没有一种方式可以让solver和refiner是一个