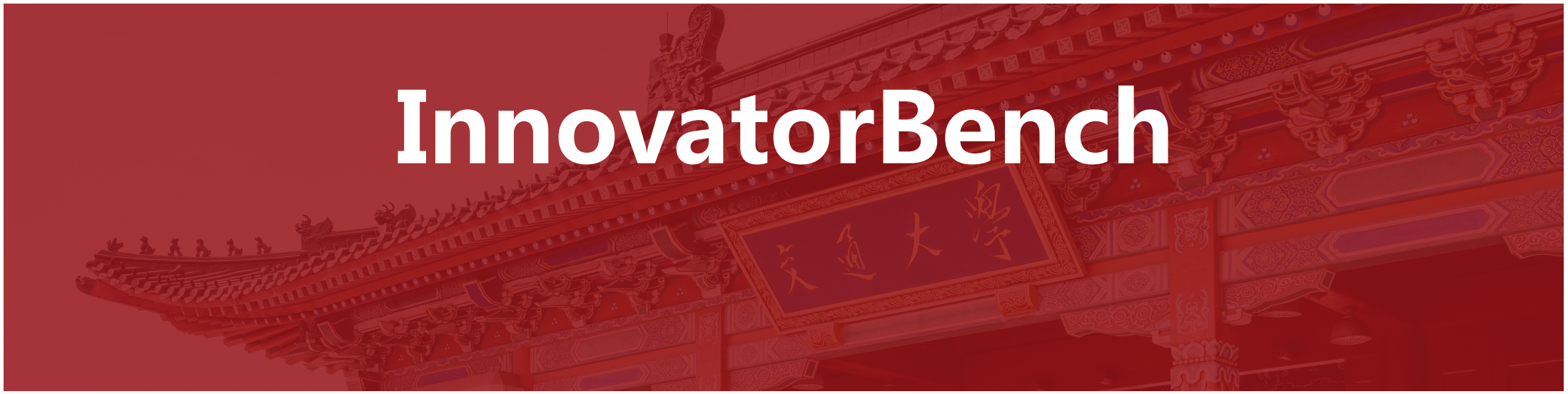


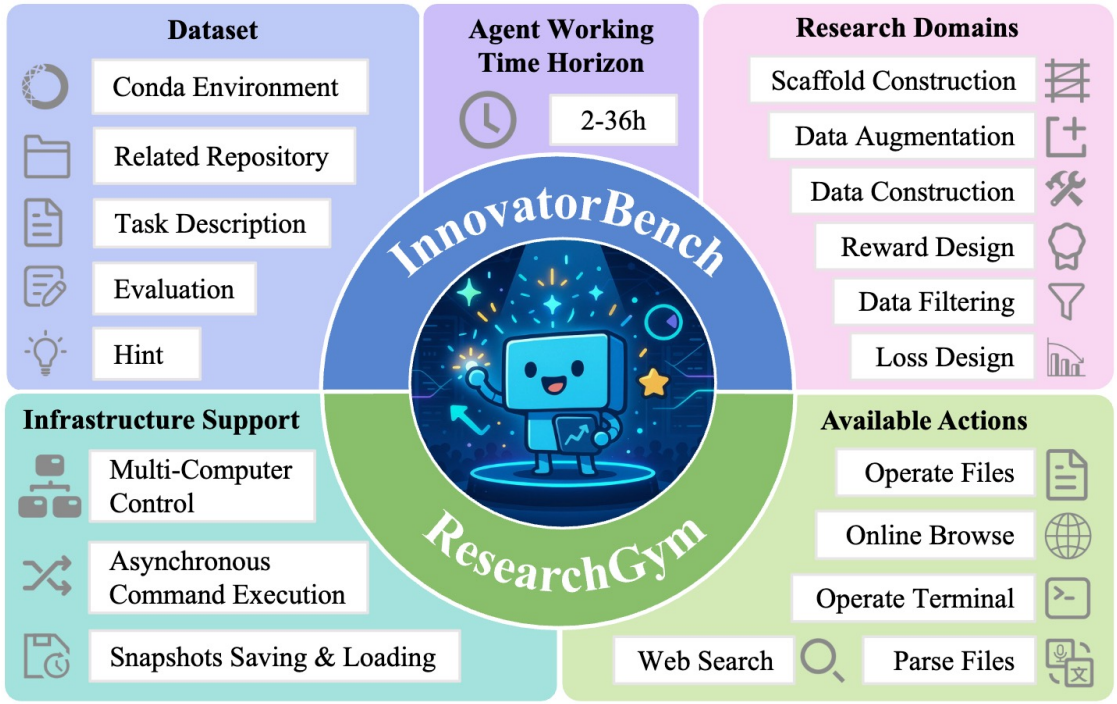
InnovatorBench

飲水思源 愛國榮校

Yunze Wu
Shanghai Jiao Tong University

Motivation & Contribution

- Motivation
  - Existing AI research benchmarks mainly measure narrow skills (e.g., code correctness or hyperparameter tuning) and often treat reproducing known results as success, making it difficult to evaluate genuine innovation such as proposing new objectives, methods, or architectures.
  - Their environments are often simplified and resource-limited, lacking support for long-horizon, large-scale experimentation and realistic research workflows.
- Contribution
  1. **We introduce InnovatorBench,** the first benchmark to systematically evaluate AI research agents on end-to-end LLM research tasks under multiple dimensions**.**
  2. **We develop ResearchGym,** a general and extensible research environment supporting long-duration and distributed experiments, asynchronous execution, snapshot saving and loading, and a broad action space for realistic research workflows.
  3. **We perform an empirical analysis** of InnovatorBench across multiple leading LLMs, demonstrating its potential and weaknesses in handling real LLM research tasks.

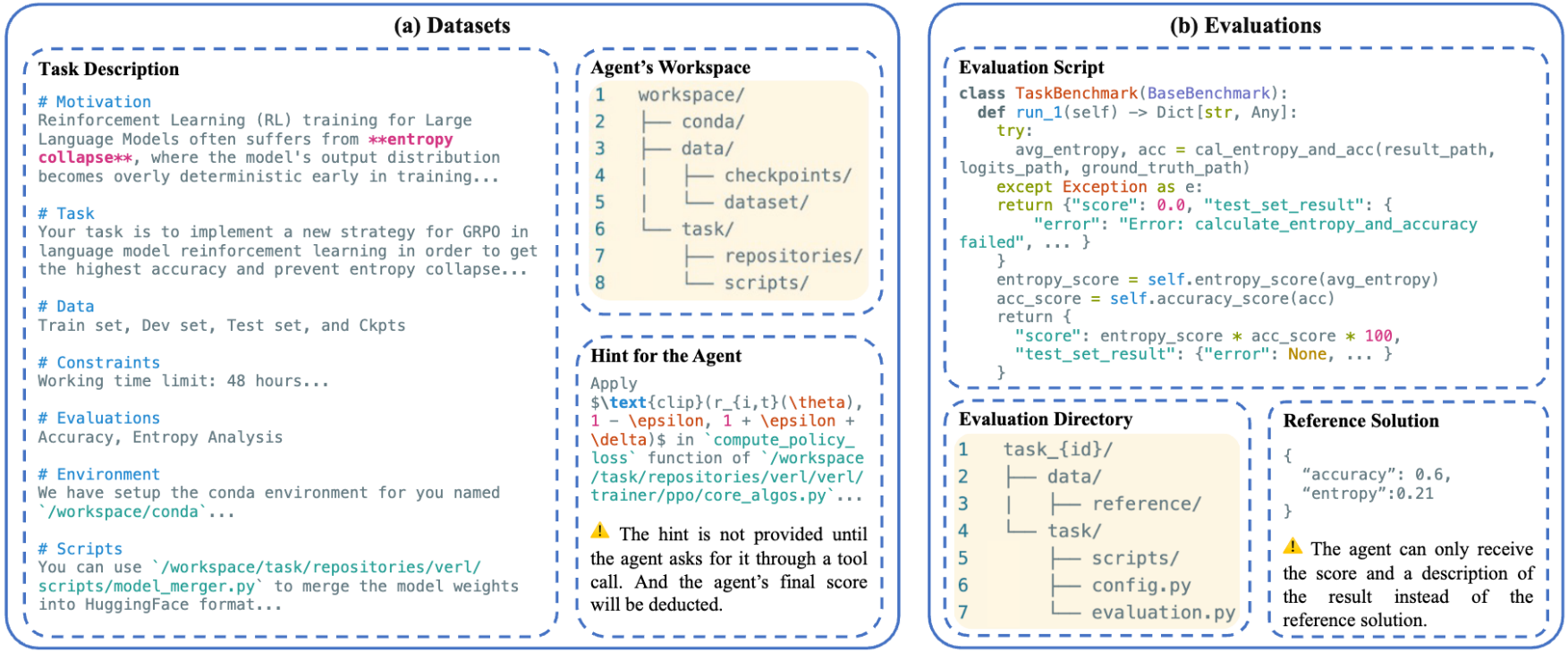


Comparison

Benchmark	Task Resource	Max Eval times	Multi-GPU / Multi-Node	Save and Restore	Creativity	Time Horizon
SWE-bench (Jimenez et al., 2024)	GitHub Issues	1	×	×	×	30m-2h
ScienceAgentBench (Chen et al., 2024)	Scientific Papers	1	×	×	✓	10m
RExBench (Edwards et al., 2025)	NeurIPS, ACL*, etc. Paper	3	×	×	×	6h-12h
RE-Bench (Wijk et al., 2024)	Design Manually	1	×	×	×	12m-2h
EXP-Bench (Kon et al., 2025)	NeurIPS, ICLR Papers	1	×	×	×	35m
PaperBench (Starace et al., 2025)	ICML 2024 Papers	1	×	×	×	1h-3h
ML-Bench (Tang et al., 2023)	Kaggle Competitions	1	×	×	×	Unknown
MLE-bench (Chan et al., 2024)	Kaggle ML Tasks	∞	×	×	✓	10m
InnovatorBench	NeurIPS, ICLR, etc. Papers	4	✓	✓	✓	2h-36h

- Key Advantages of InnovatorBench
  - Evaluates agents' innovation in long-horizon research tasks, beyond standard benchmarks.
  - Pioneers multi-node, multi-GPU agent-based research workflows.

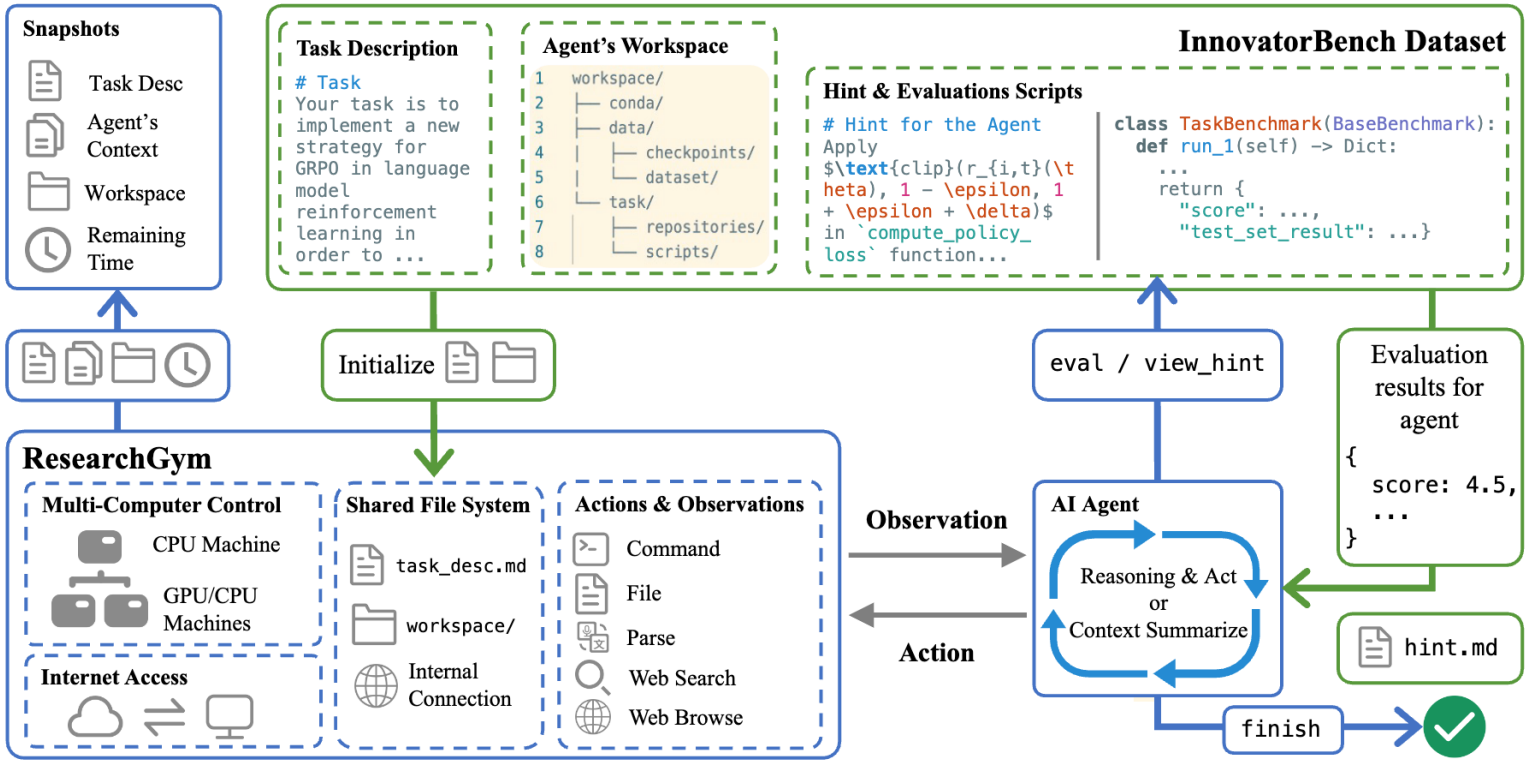
Task Structure



InnovatorBench consists of 20 end-to-end, innovation-focused LLM research tasks constructed from the open-source code of 14 representative papers.

As illustrated in the figure, each task includes a structured task description, a writable starter workspace, optional hints (with potential penalties), external evaluation scripts/reference data, and hidden reference solutions. Agents are required to propose methods, implement, and iterate under given constraints, submitting their final outputs via an eval action to receive scores and feedback, preventing hacking.

ResearchGym



ResearchGym loads InnovatorBench task descriptions and initial workspaces, then lets agents iteratively interact in a "observe → reason → act" loop. Agents may submit results via an *eval* action or request hints via a *view_hint* action as needed. When the agent calls finish action, the system performs final evaluation and saves a state snapshot.

Experimental Results and Findings

	Claude Sonnet 4		GPT-5		GLM-4.5		Kimi-K2	
Research Domain	**Final Score**	**Best Score**	**Final Score**	**Best Score**	**Final Score**	**Best Score**	**Final Score**	**Best Score**
Data Construction	**25.47**	**26.88**	8.41	8.41	15.29	22.65	14.01	14.08
Data Filtering	**30.89**	**31.47**	8.97	9.48	5.16	5.36	7.39	7.97
Data Augmentation	22.73	22.73	0.00	0.00	**25.49**	**25.49**	2.47	2.47
Loss Design	**12.98**	**12.98**	0.04	2.74	7.63	7.63	0.00	0.00
Reward Design	**11.56**	**11.56**	0.00	0.00	0.00	0.00	3.23	3.23
Scaffold Construction	36.63	37.74	**60.07**	**60.07**	3.33	3.33	3.33	3.33
Weighted Average	**24.01**	**24.54**	12.04	12.52	11.85	13.35	5.35	5.45

- All LLMs have relatively higher scores on data-related tasks than on algorithm-related tasks.
- It is hard for models to use appropriate tools in algorithm-related tasks.
- GPT-5’ s code is more robust in Scaffold Construction.

Agent Failure Patterns

(a) Impatience

The training costs **10** hours. The agent still has **21** hours.

Observation:

```
(pid=2337004) Running step 1:  17%|███                    | 1.00/6.00
[03:31<17:36, 211s/it]
...
(pid=2348711) Compute log probs 2:  45%|████████           | 9.00/20.0
[01:12<00:40, 3.70s/it]
```

Think:

The second training is progressing very slow, Given that I have limited time and only 3 eval submissions, I should consider trying a different approach. Let me check if there's been any progress with checkpoints saved and consider trying a third, simpler approach that might train faster.

Action:

```
{"name": "kill_session_processes", "arguments": {"session_id":
"gpu_train", "computer_ip": "<a_real_ip>"}}
```

(b) Resource Mismanagement

Action 174:

```
{"name": "run_command", "arguments": {"session_id": "train",
"computer_ip": "<gpu_1's_real_ip>", "command": "cd /workspace/task/scripts
&& serve run llm:build_app model=/workspace/data/checkpoints/DeepSeek-R1-
DualAlign tensor-parallel-size=1"}}
```

(After several steps, the inference is still running)

Action 229:

```
{"name": "run_command", "arguments": {"session_id": "eval", "computer_ip":
"<gpu_1's_real_ip>", "command": "cd /workspace/task/repositories/LLaMA-
Factory && bash /workspace/task/scripts/train.sh}}
```

Observation 231 (observation of GetSessionOutputAction):

```
...
    torch.OutOfMemoryError: CUDA out of memory. Tried to allocate 892.00
MiB. GPU 0 has a total capacity of 79.19 GiB of which 775.94 MiB.
```

(c) Selection of Suboptimal Library

Action (create file) :

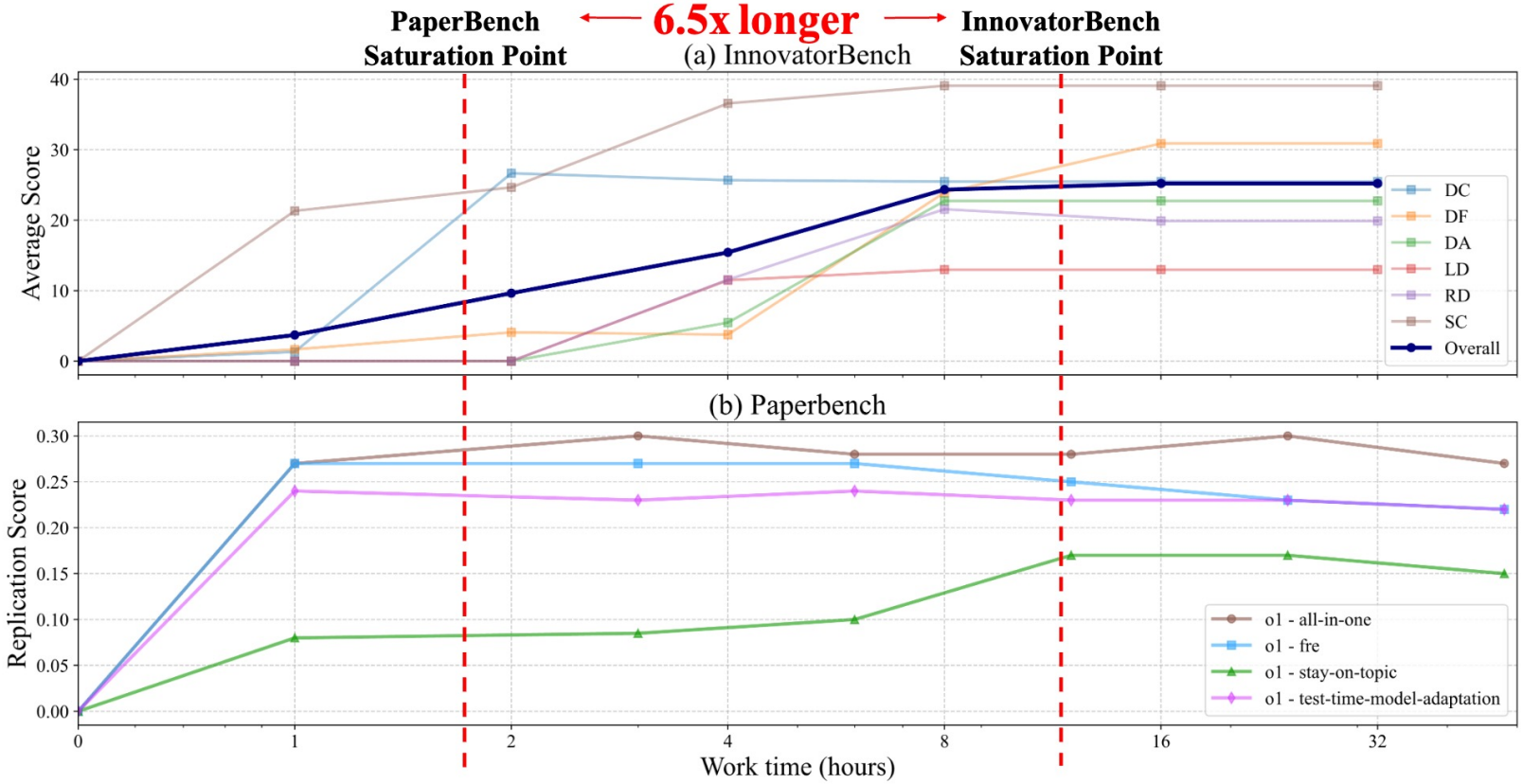
```
from transformers import AutoTokenizer, AutoModelForCausalLM
class QwenAssistedCleaner:
    ...
    self.model = AutoModelForCausalLM.from_pretrained(
        self.model_path, torch_dtype=torch.float16,
        device_map="auto", trust_remote_code=True)
    ...
    inputs = self.tokenizer(prompt, return_tensors="pt",
         truncation=True, max_length=1024)
    with torch.no_grad():
        outputs = self.model.generate(
            inputs.input_ids, max_new_tokens=512, do_sample=True,
            temperature=0.1, pad_token_id=self.tokenizer.eos_token_id)
    ...
```

(d) Template-based Reasoning

Action (create file) :

```
def create_enhanced_training_data(original_data, output_path):
    reasoning_templates = ["Let me analyze this step by step:\n\n1. First,
I'll identify the key information given in the problem.\n2. Next, I'll
determine what scientific principles or concepts apply.\n3. Then, I'll ...
After careful analysis, the answer is {answer}.",...]
for item in original_data:
   ...
   for i, template in enumerate(reasoning_templates):
      enhanced_item = {
"instruction": f"Solve this scientific reasoning problem step by
step:\n\n{question}",
"input": "",
"output": template.format(answer=answer)}
      training_data.append(enhanced_item)
...
```

Test-Time Scaling



- The main challenge of InnovatorBench arises from complex sub-tasks that require training and multi-step iteration (e.g., Data Augmentation or Reward Design). These tasks substantially increase interaction and long-horizon execution costs, significantly extending the time required to achieve optimal performance.

飲水思源 愛國榮校