

# Reinforcement Learning for Agents

Trung Vu



BESPOKE LABS

## A bit about me:

- Previously worked on recommendation systems at YouTube
- Working on data and environment curation for post-training at Bespoke Labs
- Love good tea & coffee — let me know if you have suggestions :)



# Outline

- Motivation
- Environments
  - Anatomy of an RL Environment
  - Types of RL Environment
  - What makes a good RL environment
  - Methods of Environment Curation
- Reinforcement Learning
  - Setup & Definition
  - Policy Gradient Algorithms
  - Reinforcement Learning for Agents
- Looking Ahead
- Q&A

# Motivation - Why Environments

Environments are simulations of real-world scenarios.

They allow agents to be stress-tested for and trained on a given task distribution before real-world deployment.



# Motivation - Why Reinforcement Learning (RL)?

Supervised fine-tuning required explicit demonstrations to be provided.

Demonstrations are really hard to write manually from scratch for agents!

Reinforcement learning allows agents to improve on a task without explicit demonstrations.



Example of an agent trajectory

# Anatomy of an RL Environment



# Anatomy of an RL Environment: Hello World



User

Write "Hello World" to  
/root/hello.txt



Task done!

LLM

```
{"name": "bash",  
 "cmd": "cat 'Hello World' >  
 /root/hello.txt"}
```



```
{"result": "Command  
successful"}
```

Tools

Execute bash  
command



Command executed  
successfully

State

File /root/hello.txt  
exists and content is  
'Hello World' ✓.

Reward: 1.0.

Grader



# Question for the audience

Are there any tasks you want do with LLMs that can be formulated as environments?

# Types of Environments

Function calling ([T-Bench](#), [BFCL](#))

**Tools:** Domain-specific tools

**User:** Hard-coded request or simulated user

**State:** In-process database

**Grader:** Database state checker

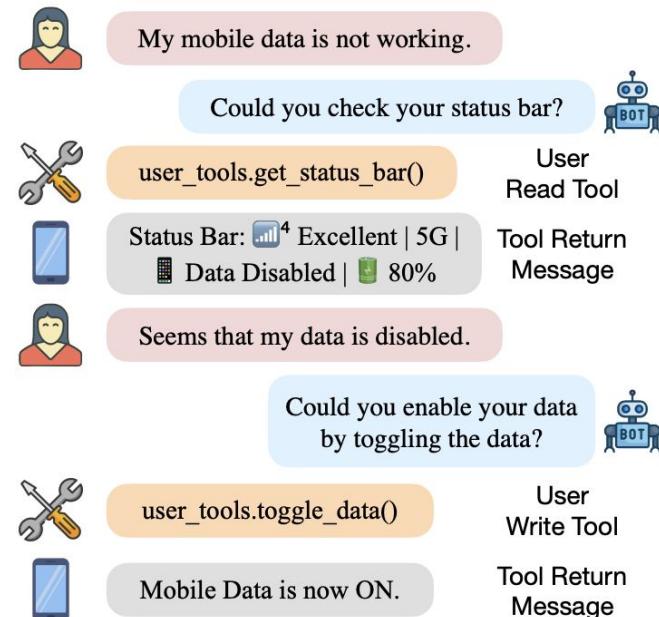


Figure from [T-Bench](#) paper

# Types of Environments

Search & Browsing (Deep Research, BrowseComp, Humanity's Last Exam)

**Tools:** Search, Web Browsing

**User:** Task description

**State:** Search Index, Internet

**Grader:** Rubrics, Reference answers

The screenshot shows a ChatGPT 5 interface. The user has asked for a list of 50 senior-level Customer Success and Support leaders in the US. The AI has responded by stating it will gather such a list and keep the user posted once ready. It also mentions evidence of their engagement like podcast appearances and LinkedIn posts. A summary at the bottom indicates research completed in 15m with 29 sources and 150 searches.

Great, I'll gather a list of at least 50 senior-level Customer Success and Support leaders in the US from B2B SaaS (primarily), B2C, e-commerce, and CPG companies who are publicly vocal about applying AI to their functions. I'll include evidence of their engagement—such as podcast appearances, LinkedIn posts, or articles—and note any relevant communities they're part of, like The F Suite or Lenny's Newsletter Slack.

I'll keep you posted once the list is ready. Feel free to continue chatting with me in the meantime!

Research completed in 15m · 29 sources · 150 searches

↑ AI-Forward Customer Success & Support Leaders (US-Based)

Below is a list of 50 senior customer success (CS) and customer support leaders in the US who have been especially vocal and proactive about applying AI in their field. These include VP-level, Heads of CS/Support, Chief Customer Officers, and similar roles across B2B SaaS, B2C/e-commerce, CPG, and tech companies. Each entry notes their role, company, and evidence of their AI-forward initiatives or thought leadership, along with any relevant community involvement. ↓

1. Kathy Isaacson, VP of Customer Success at Carbide. Participated in an Oxford-style

+ Ask anything

Activity 29 Sources

AI-Forward CX and Customer Success Leaders in the US

I'm considering AI-enhanced insights into 50 senior customer success leaders from various industries, showing their AI applications and public engagement.

Searched for customer success leaders AI 2024

Evaluating the relevance and utility of results.

Searched for top customer success leaders 2024 AI

I'm looking for influential figures like Nick Mehta and Jay Nathan, who emphasize AI in customer success roles. This involves using LinkedIn and specific search criteria to pinpoint the right individuals.

Searched for VP Customer Success AI ChatGPT blog

The search didn't disclose specific names, but I noted mentions of Jaibharat Valluri and Jim Mercer.

Read statisfy.com

I'm gathering names and titles from the event, focusing on AI and Customer Success.

# Types of Environments

Terminal Tasks ([Terminal-Bench](#), [SWE-Bench](#))

Tools: Terminal

User: Task instruction

State: Virtual machines

Grader: Evaluation of VM state

## Demo

```
root@3ea9f4d487b6:/app# apt-get update && apt-get install -y git python3
Get:1 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [128 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages
Get:4 http://archive.ubuntu.com/ubuntu noble-security/main amd64 Packages
Get:5 http://archive.ubuntu.com/ubuntu noble-security/restricted amd64 Packages
Get:6 http://archive.ubuntu.com/ubuntu noble-backports InRelease [128 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Get:8 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [152 kB]
Get:9 http://archive.ubuntu.com/ubuntu/noble-security/universe amd64 Packages [117 kB]
Get:10 http://archive.ubuntu.com/ubuntu/noble/restricted amd64 Packages [117 kB]
Get:11 http://archive.ubuntu.com/ubuntu/noble/multiverse amd64 Packages [331 kB]
Get:12 http://archive.ubuntu.com/ubuntu/noble/main amd64 Packages [189 kB]
Get:13 http://archive.ubuntu.com/ubuntu/noble-backports/main amd64 Packages [15 kB]
Get:14 http://archive.ubuntu.com/ubuntu/noble-updates/multiverse amd64 Packages [15 kB]
Get:15 http://archive.ubuntu.com/ubuntu/noble-updates/main amd64 Packages [15 kB]
Get:16 http://archive.ubuntu.com/ubuntu/noble-updates/restricted amd64 Packages [15 kB]
Get:17 http://archive.ubuntu.com/ubuntu/noble-updates/multiverse amd64 Packages [15 kB]
Get:18 http://archive.ubuntu.com/ubuntu/noble-backports/main amd64 Packages [15 kB]
Fetched 38.0 MB in 4s (8089 kB/s)
Reading package lists... 6%
```

## Agent Details

### State Analysis

The terminal is at a shell prompt as root in /app. No prior setup for a git server or webserver is evident yet.

### Explanation

First, install git and a simple webserver (e.g., nginx or python3) if not present. Then, create the directory structure for the git bare repository that will serve as the remote. This will allow future configuration steps like post-receive hooks to update the webserver content upon git push.

### Commands

```
apt-get update && apt-get install -y git python3
Blocking Timeout: 120s
mkdir -p /git/server && cd /git/server && git init --bare
Blocking Timeout: 15s
```

## Instruction

Configure a git server so that I can run on my computer

```
git clone user@server:/git/server
echo "hello world" > hello.html
git add hello.html
git commit -m "add hello.html"
git push origin master
```

And have this data then be pushed to a webserver running on port 8080 so if I run  
curl http://server:8080/hello.html  
then I see the output "hello world".

I'll setup login with the server to work, you don't have to worry about that.

# Types of Environments

# Computer Use (OSWorld)

## Tools: Keyboard and mouse (via PyAutoGUI)

## User: Task instruction

## **State:** Virtual machines

## Grader: Evaluation of VM state

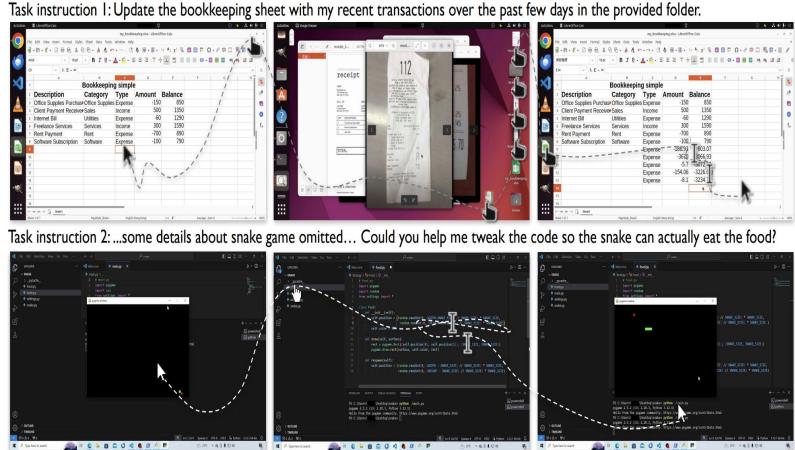


Figure from [OSWorld](#) paper

# What makes a good RL environment?

<b>Realism</b>	The environment closely resembles a real-world task.
<b>Cheating Proof</b>	The environment's grader must ensure that the model does not cheat.
<b>All Requirements Tested</b>	All of the requirements in the instruction are tested by the grader.
<b>All Tested Behaviors Specified</b>	All of the behaviors tested in the grader are described in the instruction.
<b>Not Too Easy</b>	RL will not learn from a tasks where the model's success rate is 100%.
<b>Not Too Difficult</b>	RL will not learn from a tasks where the model's success rate is 0%.
<b>Reproducible</b>	The environment must be able to start from a reproducible initial state and the state transitions & reward distribution must be stationary.

# Methods of Environment Curation - Manual

Many benchmarks created environments hand-crafted from scratch by domain-expert humans.

The task is successfully executed. We also adapt to tasks from the SQuAD set for analysis as usual on Windows. Overall, it takes 9 computer science students (all student authors) over 3 months, consuming approximately 1800 man-hours (650 hours on single-app tasks, 750 hours on workflow tasks and 400 hours for double-checking).

## OSWorld

The BrowseComp dataset was collected purely by human trainers. The instructions of BrowseComp largely followed those of SimpleQA (Wei et al., 2024)—the human trainers created challenging, fact-seeking questions with single, indisputable, short answers that would not change over time and were supported by evidence. The key properties of BrowseComp are described below.

## BrowseComp

Contributors			
We're looking for more contributors! If you are interested in collaborating please see our <a href="#">contributing page</a>			
Mike Merrill On-Lead	Alex Shaw On-Lead	Nicholas Carlini Contributor	Boxuan Li Contributor
Harsh Raj Contributor	Ivan Bercovitch Contributor	Lin Shi Contributor	Jeong Yeon Shin Contributor
Thomas Walsh Contributor	E. Kelly Buchanan Contributor	Junhong Shen Contributor	Guanghao Ye Contributor
Hawei Lin Contributor	Jason Polos Contributor	Mayya Wang Contributor	Jenja Jitsev Contributor
Marianna Nezhurina Contributor	Di Lu Contributor	Orfeas Menis Contributor	Zhiwei Xu Contributor
Zishao Chen Contributor	Yue Liu Contributor	Robert Zhang Contributor	Leon Liyang Chen Contributor
Anupag Kashyap Contributor	Jan-Lucas Ulu Contributor	Jeffrey Li Contributor	Jianbo Wu Contributor
Minchao Yan Contributor	Song Shan Contributor	Vedang Shama Contributor	Ke Sun Contributor
Steven Dillmann Contributor	Akashay Anand Contributor	Andrew Lanouthakoun Contributor	Bardia Koopah Contributor
Changran Hu Contributor	Eshagh Guha Contributor	Gabriel H. S. Orlman Contributor	Jincheng Zhu Contributor
Karl Krauth Contributor	Li Zhong Contributor	Niklas Wuenighoff Contributor	Robert Kwei Amanfu Contributor
Shangyin Tan Contributor	Shreyas Pimpalkar Contributor	Tushar Aggarwal Contributor	Xiangning Lin Contributor
Xin Lan Contributor	Xuandong Zhao Contributor	Yiqing Liang Contributor	Yuanli Wang Contributor
Zilong Wang Contributor	Changzhi Zhou Contributor	David Heineman Contributor	Hange Liu Contributor
Harsh Tsvivedi Contributor	John Yang Contributor	Junhong Lin Contributor	Manish Shetty Contributor
Michael Yang Contributor	Nabil Osi Contributor	Negin Reoof Contributor	Shanda Li Contributor
Terry Yue Zhuo Contributor	Wujie Lin Contributor	Yiwei Dai Contributor	Yuxin Wang Contributor

## Terminal-Bench

# Methods of Environment Curation - Synthesize from scratch

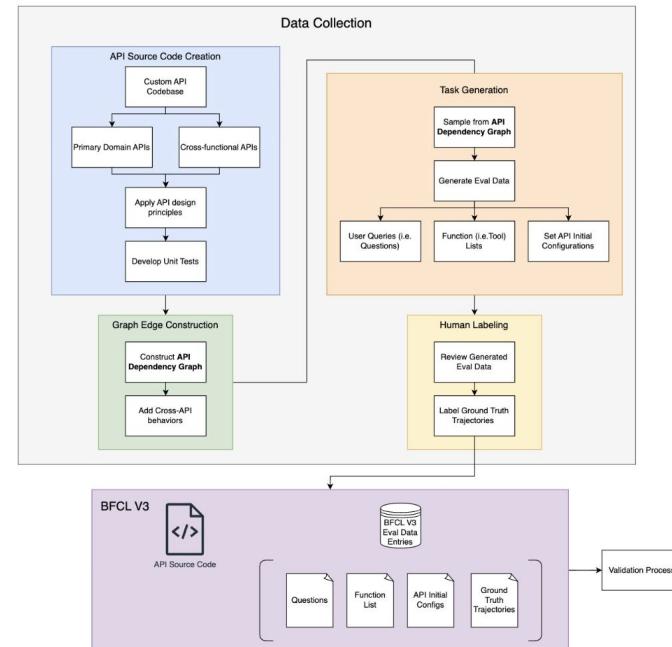
Environments can be synthesized from scratch with LLM data pipelines.

$\tau$ -bench defines domain-agnostic environment and user simulation classes shared by various domains, and domain-specific data in terms of database JSON files, database API Python code and documentation, domain policy text, and task instances. Each domain is created in a three-stage approach with a mix of LM and code runs, and human labeling and checking.

**Stage I: Manual design of database schema, APIs, and policies.** We start by co-designing the simplest possible database schemas, APIs, and policies with inspiration (and simplification) from their real-world counterparts. Simplicity is important for the logical consistency of various components and the ease of API and task annotation. Still, a minimally realistic domain requires at least tens of schemas, APIs, rules, and turns out to be challenging enough for existing agents. See § B.1 for more.

**Stage II: Automatic data generation with LMs.** Once the data schema is set up, we create an example entry and use gpt-4 to generate a systematic code snippet to sample scalable entries, and manually polish minor bugs in the code. See § B.2 for an example snippet and more details.

**Stage III: Manual task annotation and validation with agent runs.** Here, the key challenge is to ensure the user instruction leads to a unique database outcome. For example, if the preferred payment method is not specified, the user might answer differently and cause the final database to be different across trials. So we write an initial user instruction, run a trial with gpt-4-turbo function calling agent, polish the user instruction by examining the trajectory, and do this iteratively until we are certain no ambiguities exist (see Figure 7 in § A, where we run each  $\tau$ -retail task with  $> 40$  gpt-4-turbo trials and check all tasks with zero or low success rates). We can copy and edit agent actions and outputs for ground truth annotation, which is easier than annotating from scratch.



[\$\tau\$ -Bench](#)

[\*\*BFCL\*\*](#)

# Methods of Environment Curation - Mining Web Data

For certain domains, many web datasets exist that can be easily converted into environments.

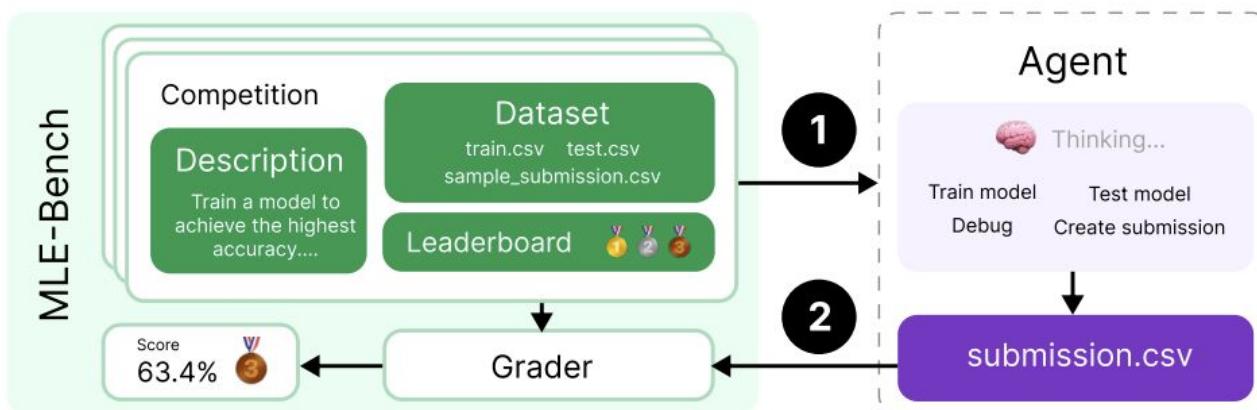


Figure 1: MLE-bench is an offline Kaggle competition environment for AI agents. Each competition has an associated description, dataset, and grading code. Submissions are graded locally and compared against real-world human attempts via the competition's leaderboard.

[MLE-Bench](#)

# Methods of Environment Curation - World-breaking

Take a system that (1) is working and (2) can be easily verified to work with tests. Break the system in way that breaks the tests.

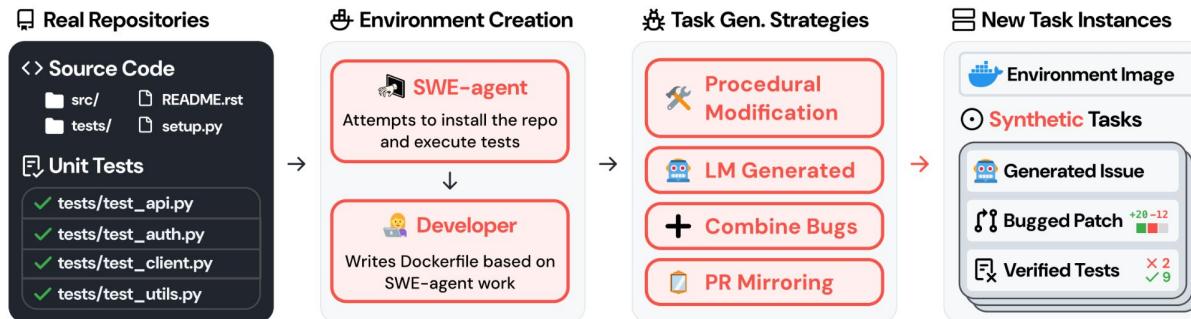


Figure 2: SWE-smith creates training data for software engineering agents by crafting bugs into real codebases. Given a codebase, we employ several strategies to create task instances that break existing tests. Using SWE-smith, we create 50k+ task instances with execution environments from 128 real world repositories.

# Methods of Environment Curation - Needle in a Haystack

**Easy to verify, but hard to solve.** To create challenging questions, we encouraged trainers to start with a fact, and then create an *inverted* question, where the answer is hard to find but easy to verify. Trainers typically started with a “seed” (could be a person, event, or artifact), found several characteristics with a large search space, and created a question from them. The exemplar question we gave them was:

What’s the title of the scientific paper published in the EMNLP conference between 2018-2023 where the first author did their undergrad at Dartmouth College and the fourth author did their undergrad at University of Pennsylvania? (Answer: Frequency Effects on Syntactic Rule Learning in Transformers)

In this example, it’s easy to verify the answer with just a few web searches, but it is much harder to find the answer, as a brute-force search would require examining thousands of papers, and looking into the backgrounds of the authors for each of them.

[BrowseComp](#)

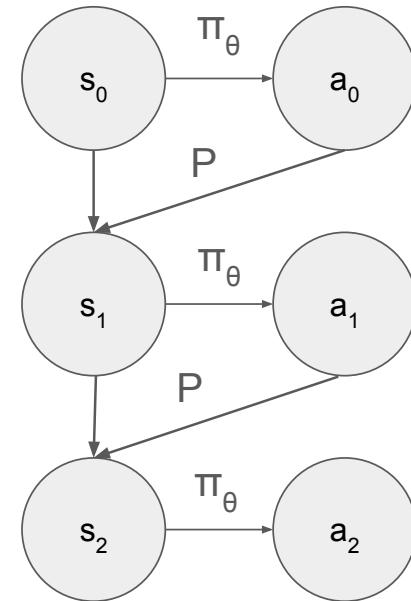
# Reinforcement Learning - Setup & Definition

## Markov Decision Process (MDP)

Repeat until  $s_t$  is a terminal state:

- Agent takes action  $a_t$ , conditional on state  $s_t$  according to the policy  $\pi_\theta(a_t|s_t)$
- The environment transitions from state  $s_t$  to  $s_{t+1}$  according to the state transition distribution  $P(s_{t+1}|s_t)$
- Agent receives a reward  $r_t(a_t, s_t, s_{t+1})$

Find the policy  $\pi_\theta$  that maximizes the total expected return  $E_\theta[\sum r_t]$



# Reinforcement Learning - Setup & Definition

$s_t$  = All the tokens observed by the model up to token t

$a_t$  = The LLM's predicted token at position t+1

$\pi_\theta$  = An LLM with parameters  $\theta$

$P_t$  = The environment's response at position t+1 (empty if LLM is generating, response token otherwise)

$r_t$  = The reward at step t, which is 0 if t is not the final token and equal to the grader reward if t is the final token



# Policy Gradient Algorithms - The Theorem

$$\nabla_{\theta} E_{\theta}[\sum_t r_t] = E_{\theta}[\sum_t (\sum_t r_t) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)]$$

## Intuition:

- $\nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$  is the direction of fastest descent for  $\log \pi_{\theta}(a_t | s_t)$ , which is the log-probability that the next token is  $a_t$  given the current state  $s_t$
- $\sum_t (\sum_t r_t)$  is the total return from the entire trajectory
- Total return is negative → decrease log probability of  $a_t$  in proportion to the magnitude of the return.
- Total return is positive → increase log probability of  $a_t$  in proportion to the magnitude of the return

# Policy Gradient Algorithms - REINFORCE

**Insight:** policy gradient maximizes the reward → empirically estimate the policy gradient and follow it

Repeat

1. Sample batches of b trajectories  $(s_0, a_0, r_0, \dots, s_n, a_n, r_n)$  from the policy  $\pi_\theta$
2.  $\theta \leftarrow \theta + \sum_b \sum_t (\sum_t r_t) \nabla_\theta \log \pi_\theta(a_t | s_t)$

# Policy Gradient Algorithms - The General Formulation

All policy gradient algorithms use policy gradient updates, but with different ways to compute the magnitude  $\Psi_t$

Policy gradient methods maximize the expected total reward by repeatedly estimating the gradient  $g := \nabla_{\theta} \mathbb{E} [\sum_{t=0}^{\infty} r_t]$ . There are several different related expressions for the policy gradient, which have the form

$$g = \mathbb{E} \left[ \sum_{t=0}^{\infty} \Psi_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right], \quad (1)$$

where  $\Psi_t$  may be one of the following:

1.  $\sum_{t=0}^{\infty} r_t$ : total reward of the trajectory.
2.  $\sum_{t'=t}^{\infty} r_{t'}$ : reward following action  $a_t$ .
3.  $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$ : baselined version of previous formula.
4.  $Q^{\pi}(s_t, a_t)$ : state-action value function.
5.  $A^{\pi}(s_t, a_t)$ : advantage function.
6.  $r_t + V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$ : TD residual.

# Policy Gradient Algorithms - GRPO

1. Prevent large abrupt updates

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

2. Account for differences between  
trained model and  
inference model

$$\mathbb{D}_{KL}(\pi_\theta || \pi_{ref}) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \quad (2)$$

where  $\varepsilon$  and  $\beta$  are hyper-parameters, and  $A_i$  is the advantage, computed using a group of rewards  $\{r_1, r_2, \dots, r_G\}$  corresponding to the outputs within each group:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \dots, r_G\})}{\text{std}(\{r_1, r_2, \dots, r_G\})}. \quad (3)$$

3. Baseline and  
normalize policy update  
magnitude based on the  
empirical distribution

# Reinforcement Learning for Agents - What's Different?

## Policy Gradient Algorithm

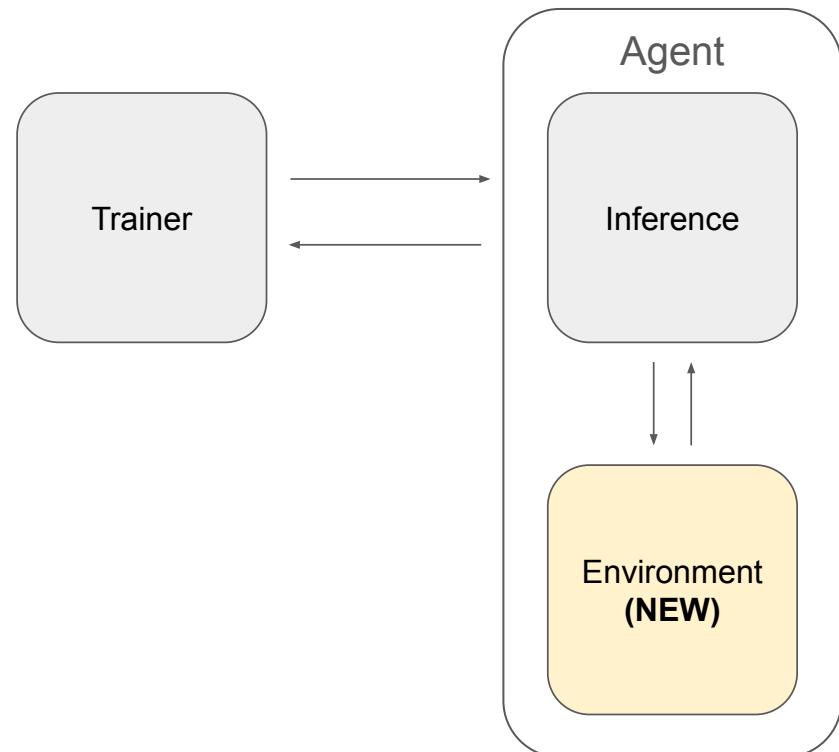
1. Sample batches of b trajectories  $(s_0, a_0, r_0, \dots, s_n, a_n, r_n)$  from the policy  $\pi_\theta$   
(LLM + **agent harness**) and **the environment P**
2.  $\theta \leftarrow \theta + \sum_b \sum_t (\sum_t r_t) \nabla_\theta \log \pi_\theta(a_t | s_t)$

# Reinforcement Learning for Agents - Challenges

Environments are a new infrastructure component to manage.

Many environments require us to spin up a VM / Docker instance to execute tools.

Now we need to manage multiple VMs/Docker.



# Reinforcement Learning for Agents - Challenges

The same model can perform differently in different agent harnesses.

This opens up another set of variables to account for during training, e.g. we might need to co-train the model with our intended harness.

Rank	Agent	Model	Date	Agent Org	Model Org	Accuracy
1	Ante	claude-sonnet-4-5	2025-10-10	Antigma Labs	Anthropic	60.3% ± 1.1
2	Droid	claude-opus-4-1	2025-09-24	Factory	Anthropic	58.8% ± 0.9
3	Droid	claude-sonnet-4-5	2025-09-29	Factory	Anthropic	57.5% ± 0.8
4	OB-1	Multiple	2025-09-10	OpenBlock	Multiple	56.7% ± 0.6
5	Ante	claude-sonnet-4	2025-09-30	Antigma Labs	Anthropic	54.8% ± 1.5
6	Droid	gpt-5	2025-09-24	Factory	OpenAI	52.5% ± 2.1
7	Chaterm	claude-sonnet-4-5	2025-10-10	Chaterm	Anthropic	52.5% ± 0.5
8	Warp	Multiple	2025-06-23	Warp	Anthropic	52.0% ± 1.0
9	Terminus 2	claude-sonnet-4-5	2025-09-30	Stanford	Anthropic	51.0% ± 0.8
10	Droid	claude-sonnet-4	2025-09-24	Factory	Anthropic	50.5% ± 1.4
11	DeepAgent Desktop	claude-sonnet-4-5	2025-10-17	Abacus.AI	Anthropic	50.5% ± 0.5

Terminal-Bench  
leaderboard

# Reinforcement Learning for Agents - Challenges

Agent trajectories can easily involve up to 100s of tool calls, quickly consuming context length.

This makes both serving and training more expensive.



A few turns in a long agent trajectories

# Reinforcement Learning for Agents - Recent Work

Agentic RL in the open is behind compared to agentic benchmarks. While widely applied in close-sourced labs, very few details are disclosed in the open. Some works with detailed recipes and datasets:

- Improving Multi-Turn Tool Use with Reinforcement Learning:  
<https://www.bespokelabs.ai/blog/improving-multi-turn-tool-use-with-reinforcement-learning>
- Search-R1: <https://github.com/PeterGriffinJin/Search-R1>
- DeepSWE:  
<https://pretty-radio-b75.notion.site/DeepSWE-Training-a-Fully-Open-sourced-State-of-the-Art-Coding-Agent-by-Scaling-RL-22281902c1468193aabbe9a8c59bbe33>

# Looking Ahead

A proxy for the economic value of a task is how long it takes for a human expert to complete that task.

How do we build agents that can autonomously complete tasks that would take a human experts hours / days / months to complete?

# Looking Ahead

A simple roadmap

Step 1: Create tasks that take hours / days / months to complete

Step 2: Use reinforcement learning to improve the agent on these tasks

# Looking Ahead

Step 1: Create tasks that take hours / days / months to complete

**Challenge:** The longer a task takes, the more complex the state space of its environment and the more difficult it is to create a reproducible sandbox around it, e.g. simulating a business requires simulating with a dizzying array of software tools, a complex and constantly changing supply chain and customer preferences.

# Looking Ahead

Step 2: Use reinforcement learning to improve the agent on these tasks

**Challenge:** The feedback from RL gets sparser as the horizon gets longer.

Credit assignment problem: If you made ~1,000s of decisions and succeeded, how much credit should you assign each to each of these decisions?

Andrej Karpathy: “sucking supervision through a straw”

Thank you!