

Reinforcement Learning beyond Reward Maximization

Yuda Song, Frontiers in Online Reinforcement Learning Workshop, April 1st 2026

RL with Reward Maximization

Protocol

$$s_h, a_h, r_h, s_{h+1} \sim P^\pi$$

Objective

$$\arg \max_{\pi} \mathbb{E}^\pi \left[\sum_{h=1}^H r_h \right]$$

Works pretty well in LLM post-training: reasoning, agent training with e.g., GRPO.

Are we done? No:

- Diversity collapse (restrained capability within base model)
- Overoptimization
- Stuck in local minimum

RL beyond Reward Maximization

Protocol

$$s_h, a_h, r_h, s_{h+1} \sim P^\pi$$

Part 2: why do we only use reward as the only **signal?**

Expanding the Capabilities of RL via Text Feedback

Objective

$$\arg \max_{\pi} \mathbb{E}^{\pi} \left[\sum_{h=1}^H r_h \right]$$

Part 1: are we using the right **objective?**

Maximum Likelihood Reinforcement Learning

RL beyond Reward Maximization

Protocol

$$s_h, a_h, r_h, s_{h+1} \sim P^\pi$$

Part 2: why do we only use reward as the only **signal**?

Expanding the Capabilities of RL via Text Feedback

Objective

$$\arg \max_{\pi} \mathbb{E}^{\pi} \left[\sum_{h=1}^H r_h \right]$$

Part 1: are we using the right **objective**?

Maximum Likelihood Reinforcement Learning

Binary Reward Tasks

Correctness-based tasks (reasoning, navigation, program synthesis) share similar structure.

Stochastic Generation

Okay, I have thisTherefore the answer is **3**.

So, my goal is ... I think the answer is **6**.

Consider the expansion of ..., answer is **3**.

Okay, so Finally, number of roots: **3**.

Let me think ... Final answer: **8**.

Binary Verifier

Ground Truth
 $y^* = 3$



Reward = 1

Reward = 0

Model-Induced Success Probability

✗ 6

✓ 3

✗ 8

20%

60%

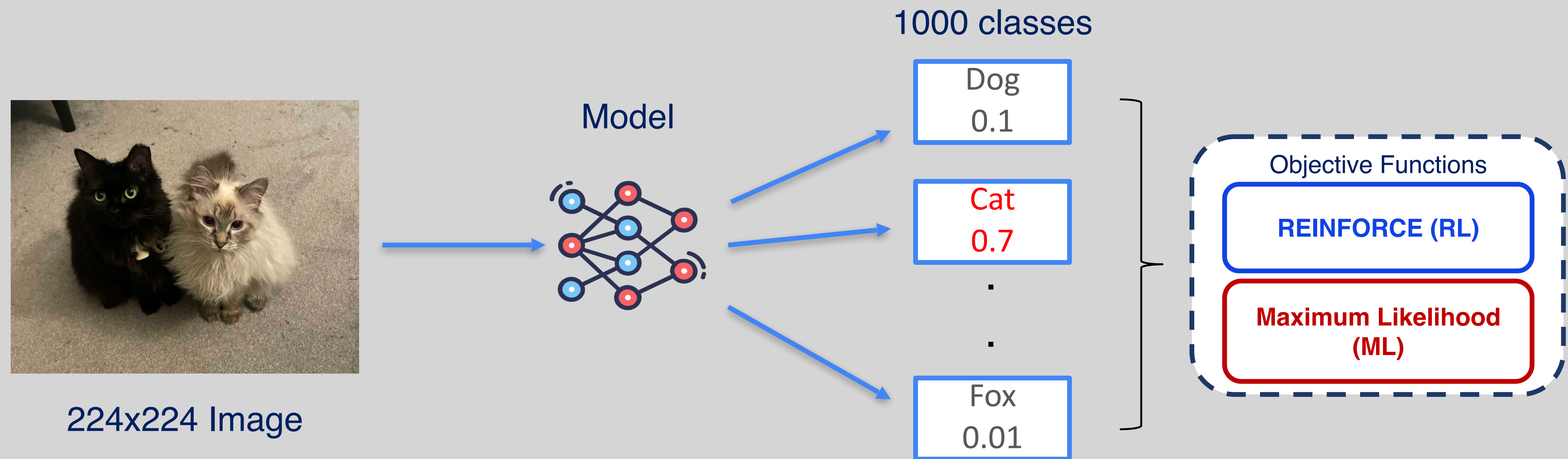
20%

Output Distribution
 $p_{\theta}(\mathbf{x})$

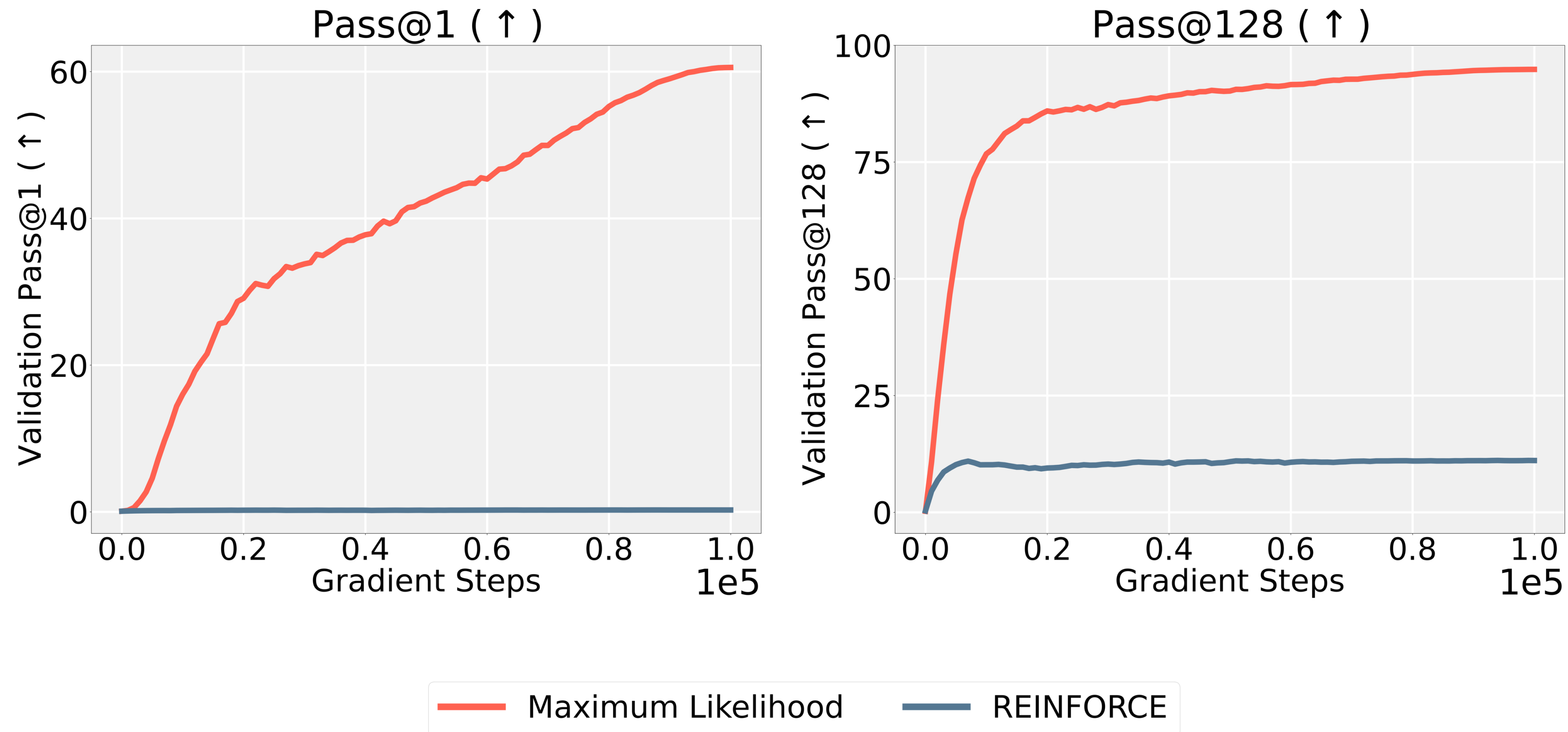
Can RL Solve Image Classification?

Image classification:

- It is a binary reward task.
- No non-differentiable sampling required.
- The gold standard objective is maximum likelihood, but we can also use RL.



REINFORCE Barely Works on Image Classification



Unpacking REINFORCE and ML Objectives

Let $p_{\theta}(y \mid x)$ denote the model probability for input x belonging to class y .

Let $y^*(x)$ be the correct class for input x . Can define everything similarly with $z \sim p_{\theta}(\cdot \mid x)$ and extract $y := f(z)$.

$$r(x, y) = \mathbb{I}(y = y^*)$$

$$p_{\theta}(x) := p_{\theta}(y^*(x) \mid x) \text{ aka } \text{pass@1}(x)$$

REINFORCE (RL)

$$\begin{aligned} J_{\text{RL}} &= \mathbb{E}_x[\mathbb{E}_{y \sim p_{\theta}(\cdot \mid x)}[r(x, y)]] \\ &= \mathbb{E}_x[\mathbb{E}_{y \sim p_{\theta}(\cdot \mid x)}[\mathbb{I}(y = y^*)]] \\ &= \mathbb{E}_x[p_{\theta}(x)] \end{aligned}$$

Maximum Likelihood (ML)

$$\begin{aligned} J_{\text{ML}} &= \mathbb{E}_x[\mathbf{log} p_{\theta}(x)] \\ \nabla_{\theta} J_{\text{ML}} &= \mathbb{E}_x \left[\frac{1}{p_{\theta}(x)} \nabla_{\theta} p_{\theta}(x) \right] \end{aligned}$$

REINFORCE: First-order Approximation of ML

REINFORCE (RL)

$$\nabla_{\theta} J_{\text{RL}}(x) = \nabla_{\theta} \text{pass}@1(x)$$

Maximum Likelihood (ML)

$$\nabla_{\theta} J_{\text{ML}}(x) = \nabla_{\theta} \sum_{k=1}^{\infty} \text{pass}@k(x)$$

When pass rate is very small, higher order pass@k gradients provide the required learning signal.

But we can not estimate the gradient in practice with finite sample unbiasedly?

Constructing MaxRL

REINFORCE

$$\nabla_{\theta} J_{\text{RL}}(x) = \nabla_{\theta} \text{pass@1}(x)$$

N = 1

MaxRL

$$\nabla_{\theta} J_{\text{maxRL}}^{(N)}(x) = \nabla_{\theta} \sum_{k=1}^N \text{pass@k}(x)$$

Truncation order, N

Maximum Likelihood

$$\nabla_{\theta} J_{\text{ML}}(x) = \nabla_{\theta} \sum_{k=1}^{\infty} \text{pass@k}(x)$$

N = ∞

This gives us a general framework for optimizing ML via policy gradients:

Estimate pass@k gradients via any estimator, plug them in!

But can we do better?

Constructing MaxRL

REINFORCE

$$\nabla_{\theta} J_{\text{RL}}(x) = \nabla_{\theta} \text{pass@1}(x)$$

N = 1

MaxRL

$$\nabla_{\theta} J_{\text{maxRL}}^{(N)}(x) = \nabla_{\theta} \sum_{k=1}^N \text{pass@k}(x)$$

Truncation order, N

Maximum Likelihood

$$\nabla_{\theta} J_{\text{ML}}(x) = \nabla_{\theta} \sum_{k=1}^{\infty} \text{pass@k}(x)$$

N = ∞

The maximum likelihood objective can be seen as an expectation under the success-conditioned distribution.

Proposition

$$\nabla_{\theta} J_{\text{ML}}(x) = \mathbb{E}[\nabla_{\theta} \log p_{\theta}(z | x) | f(z) = y^*(x)]$$

Learner's score function

Conditioning on success

- Maximum likelihood is learning from the success conditioned current policy.
- We can estimate MaxRL objective empirically in this form.

Constructing MaxRL

REINFORCE

$$\nabla_{\theta} J_{\text{RL}}(x) = \nabla_{\theta} \text{pass@1}(x)$$

T = 1

MaxRL

$$\nabla_{\theta} J_{\text{MaxRL}}^{(T)}(x) = \sum_{k=1}^T \frac{1}{k} \nabla_{\theta} \text{pass@k}(x)$$

Truncation order, T

Maximum Likelihood

$$\nabla_{\theta} J_{\text{ML}}(x) = \sum_{k=1}^{\infty} \frac{1}{k} \nabla_{\theta} \text{pass@k}(x)$$

T = ∞

Proposition

$$\nabla_{\theta} J_{\text{ML}}(x) = \mathbb{E}[\nabla_{\theta} \log p_{\theta}(z | x) | f(z) = y^*(x)]$$

Learner's score function

Conditioning on success

Score function of each sample

If we only have N samples, then let's just estimate the above directly with those samples:

Proposition

$$\mathbb{E}[\hat{g}_{\text{N}}] = \nabla_{\theta} J_{\text{MaxRL}}^{(\text{N})}$$

$$\hat{g}_{\text{N}}(x) := \begin{cases} \frac{1}{K} \sum_{i=1}^N r_i S_i, & K \geq 1, \\ 0, & K = 0. \end{cases}$$

Number of success

Implementing MaxRL

Algorithm 1 On-Policy Implementation of MAXRL.

```
require Batch of inputs  $B$ , number of rollouts  $N$ , latent policy  $m_\theta(\cdot \mid \cdot)$ 
1: for each input  $x \in B$  do
2:   Sample  $z_1, \dots, z_N \sim m_\theta(\cdot \mid x)$ 
3:   for  $j = 1$  to  $N$  do
4:      $r_j \leftarrow \mathbb{I}\{f(z_j) = y^*(x)\}$ 
5:      $S_j \leftarrow \nabla_\theta \log m_\theta(z_j \mid x)$ 
6:   end for
7:    $\hat{r}(x) \leftarrow \frac{1}{N} \sum_{j=1}^N r_j$ 
8:    $\hat{g}(x) \leftarrow \begin{cases} \frac{1}{N \hat{r}(x)} \sum_{j=1}^N (r_j - \hat{r}(x)) S_j, & \hat{r}(x) > 0, \\ 0, & \text{otherwise} \end{cases}$ 
9: end for
10:  $\hat{g} \leftarrow \frac{1}{|B|} \sum_{x \in B} \hat{g}(x)$ 
11: return  $\hat{g}$ 
```

Average
reward

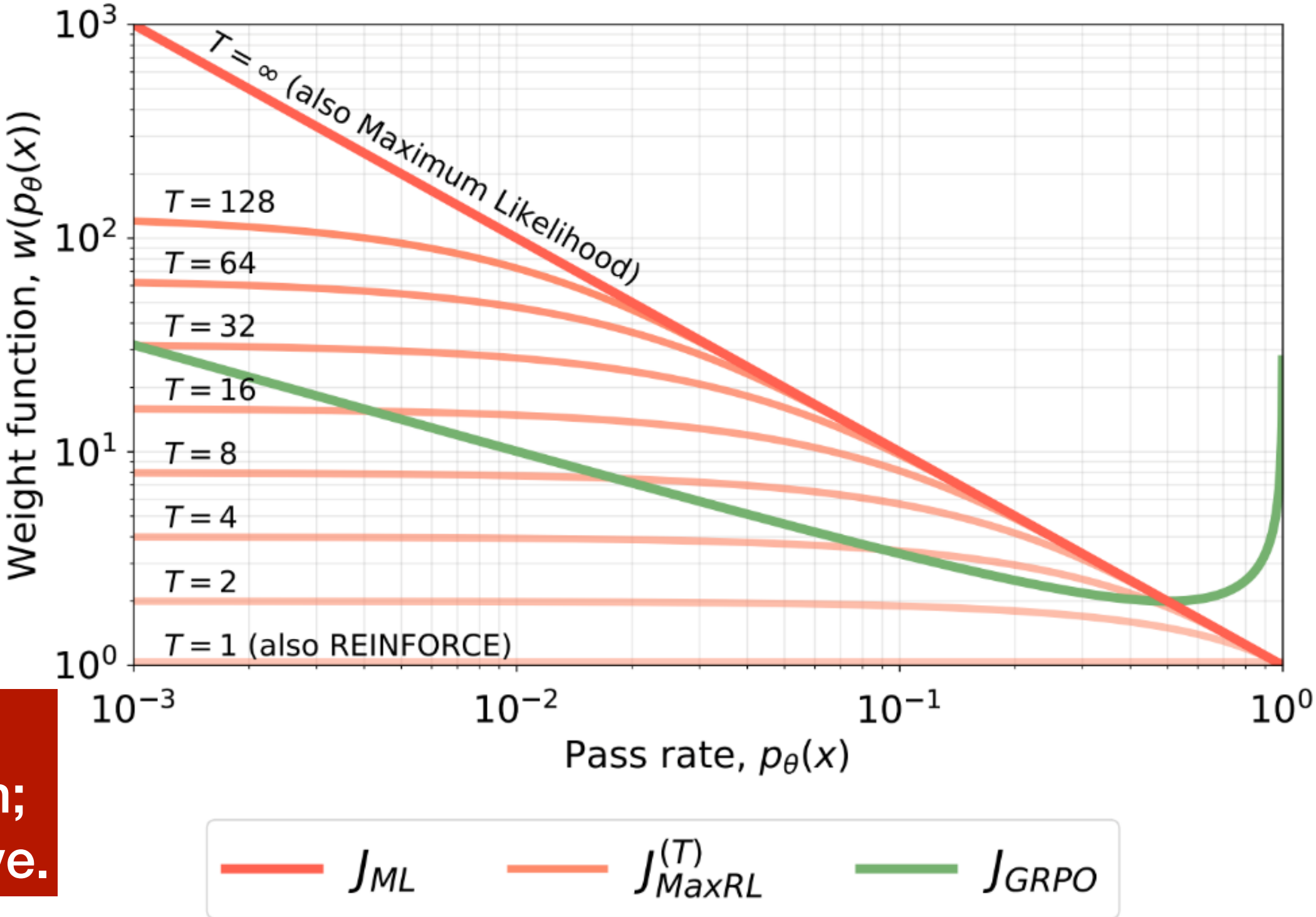
Number of
success

With more samples:
GRPO only improves the estimation;
MaxRL approaches a better objective.

$$\nabla_\theta J = \mathbb{E}_{x \sim \rho} [w(p_\theta(x)) \nabla_\theta p_\theta(x)]$$

Objective-specific
weight function

Standard policy gradient
(e.g., REINFORCE)



How Does MaxRL Perform in-Practice?

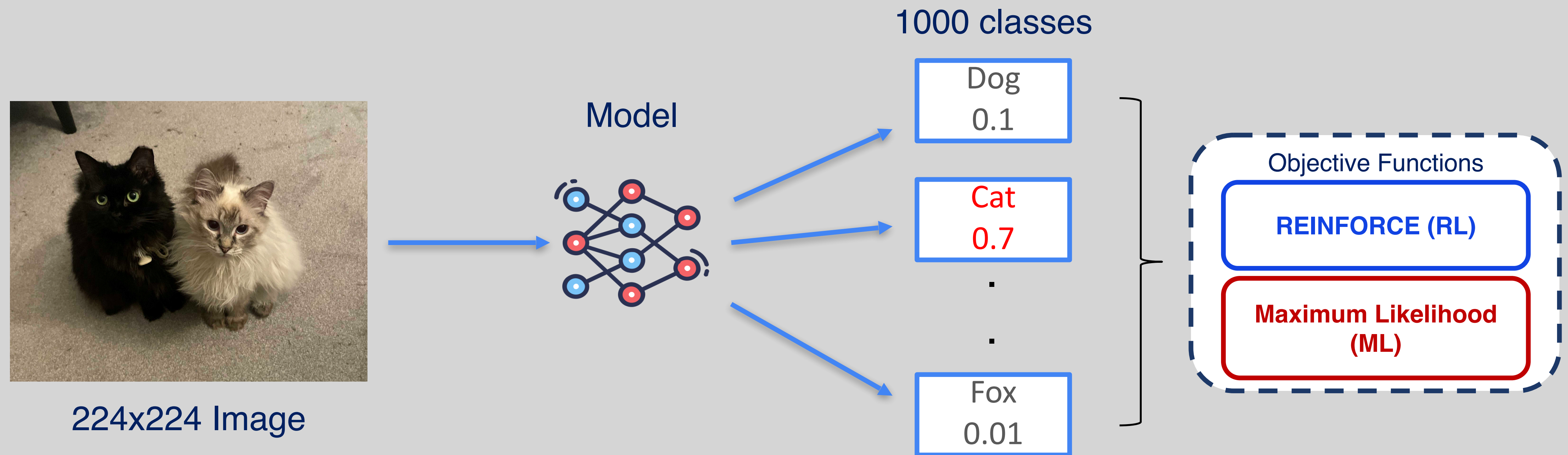
4 Settings:

- Didactic image classification
- Infinite training data regime
- Data-scarce regime
- Larger scale LLM reasoning problems

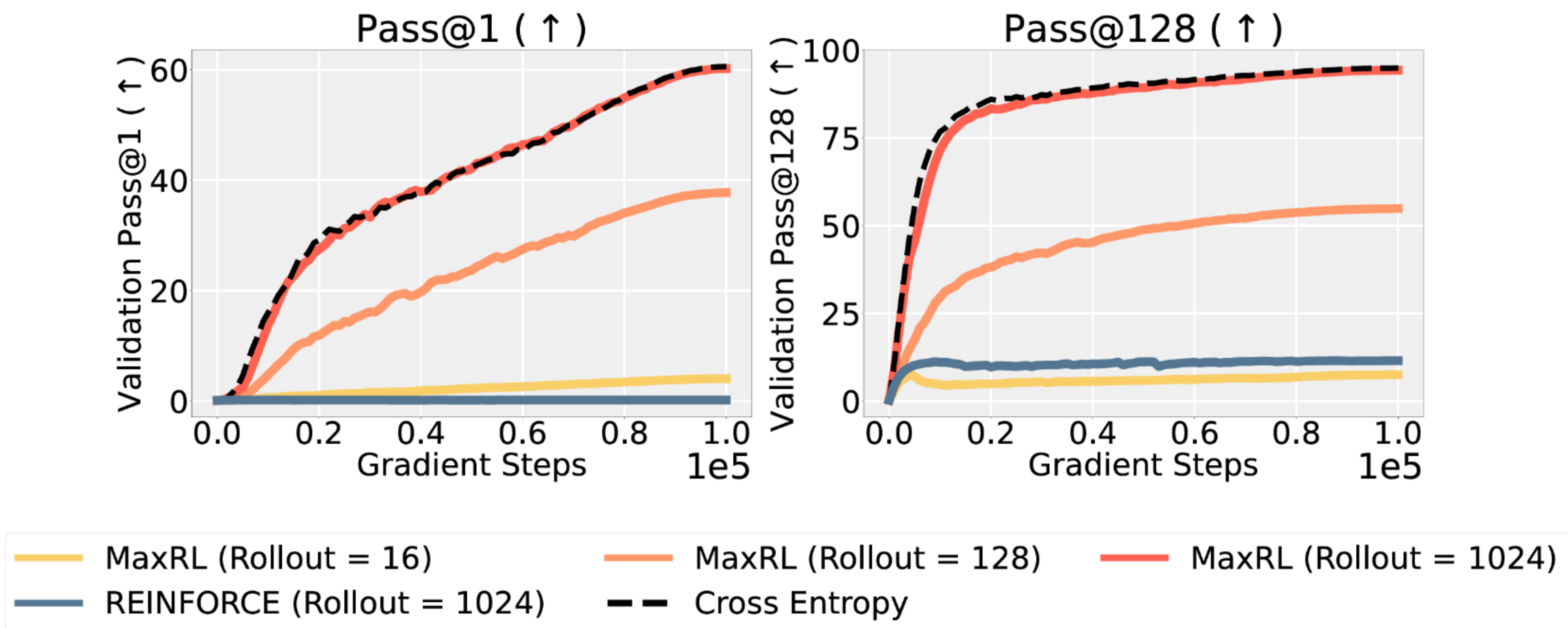
Image Classification

Consider the same image classification setup from before.

Except now we study sampling-based RL objectives.



MaxRL approaches maximum likelihood with more compute

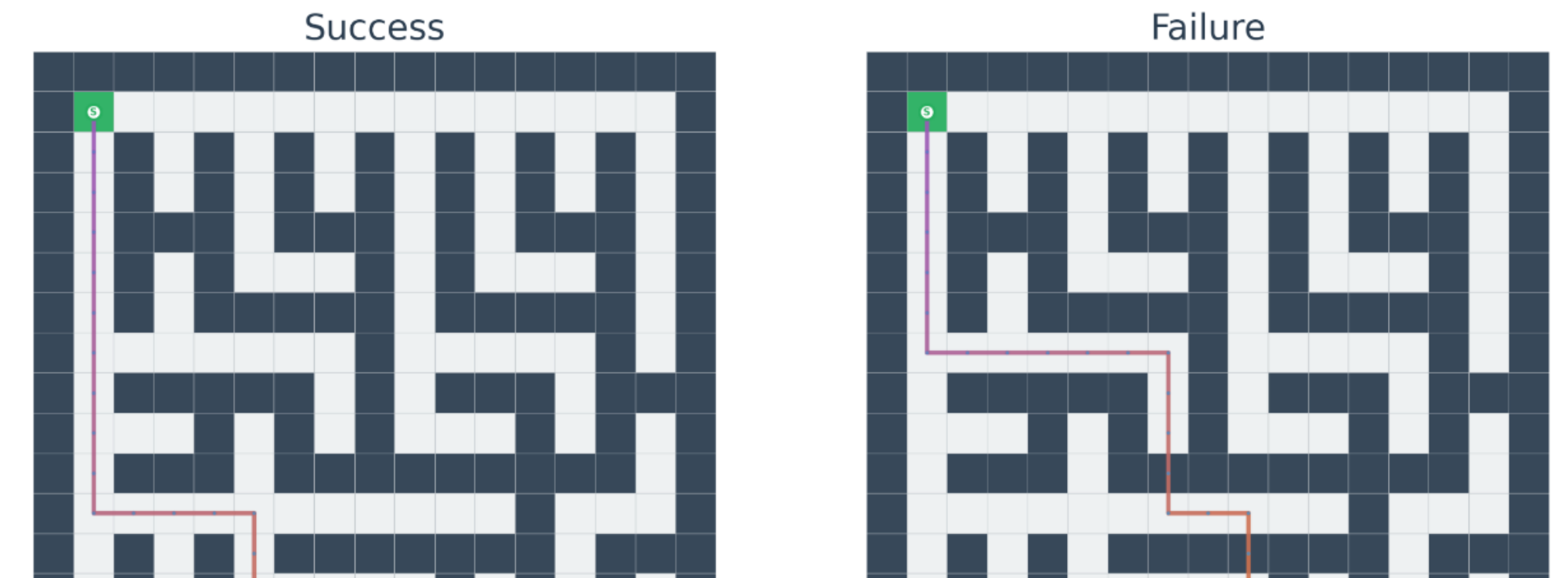


What Happens if We Had Infinite Training Data?

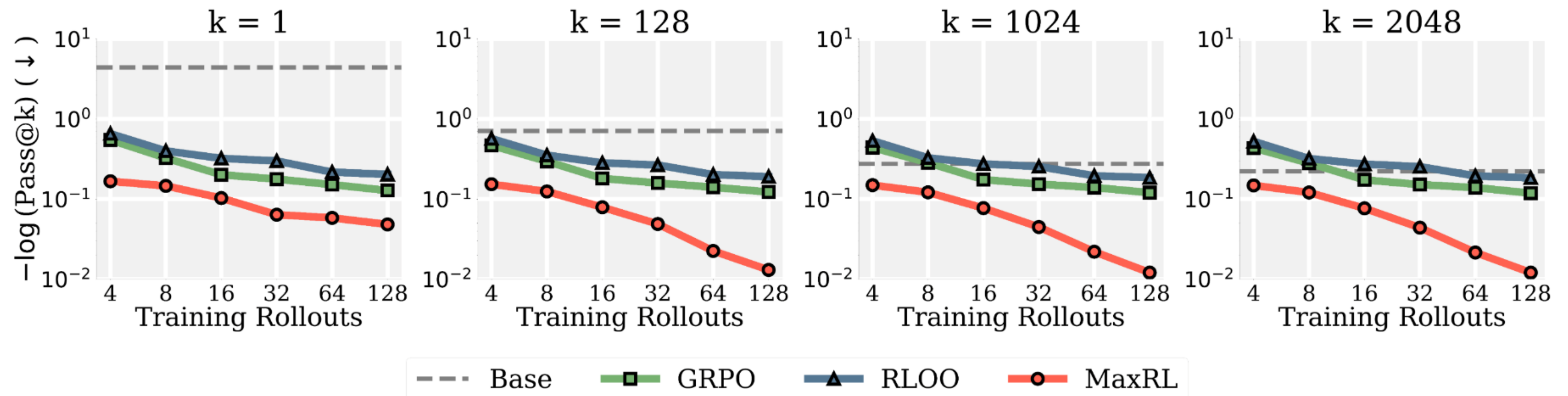
With sampling, but more training data than compute

Training data: 1M unique mazes

Evaluation: 1000 held-out mazes



As we increase training compute (i.e., sampling budget), MaxRL scales much better, specially at higher pass@k!



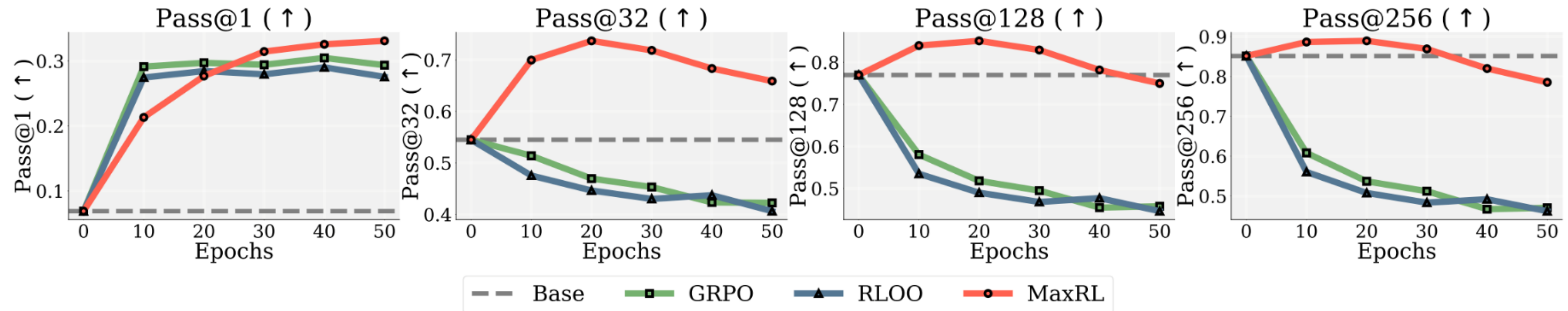
RL under Infinite Compute

Model: SmoLM2-360M-Instruct

Training dataset: GSM8K

Evaluation dataset: GSM8K-Platinum

What if we have more compute than training samples?
(Something that will very likely to happen in the future).



MaxRL grows slower at pass@1, but eventually converges to a higher performance level.
MaxRL also exhibits much lower pass@k degradation (no degradation until 30 epochs!)

(Some new findings that MaxRL might not necessarily be slower)

Half Time

Protocol

$$s_h, a_h, r_h, s_{h+1} \sim P^\pi$$

Part 2: why do we only use reward as the only **signal?**

Expanding the Capabilities of RL via Text Feedback

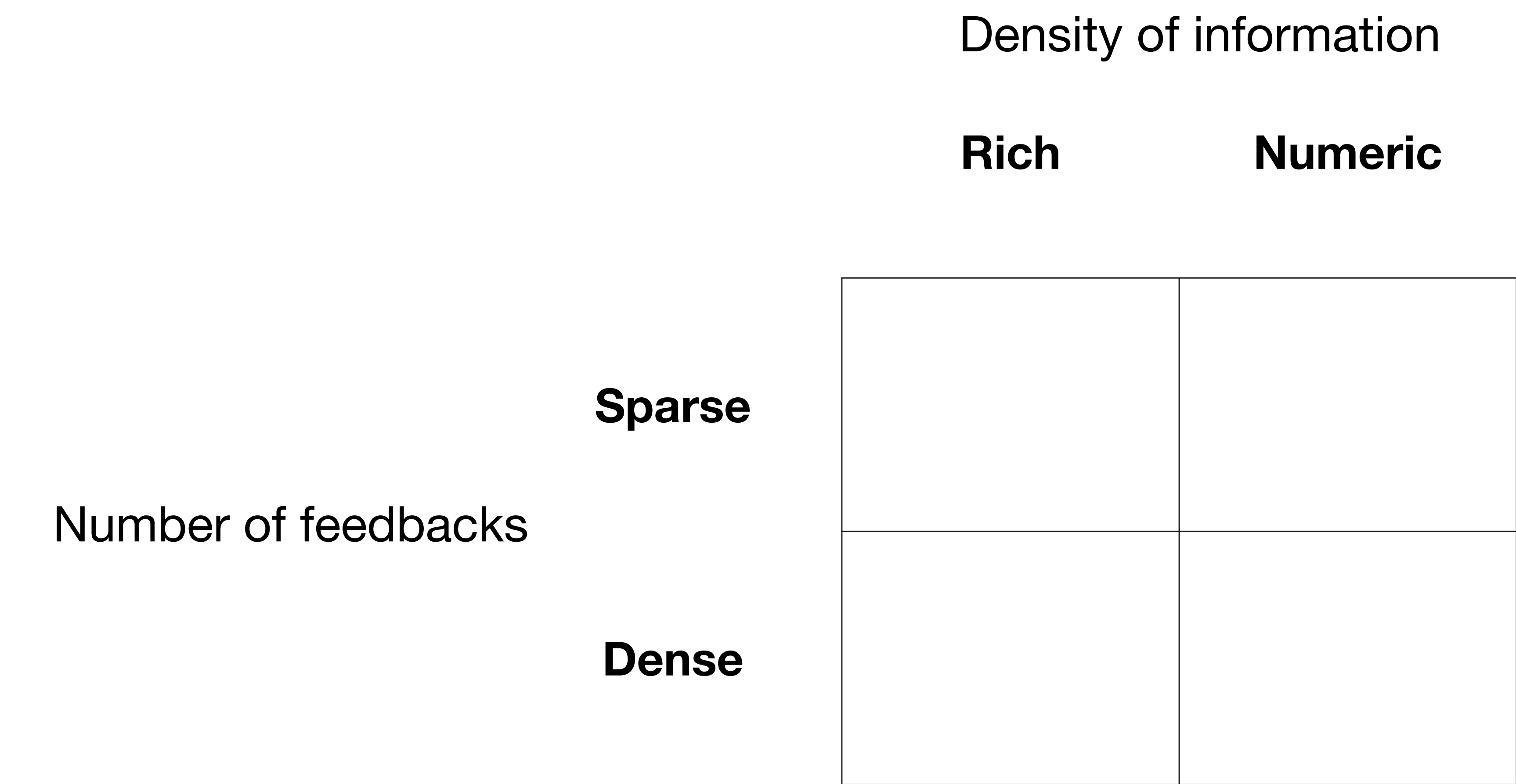
Objective

$$\arg \max_{\pi} \mathbb{E}_x [\log p_{\pi}(x)]$$

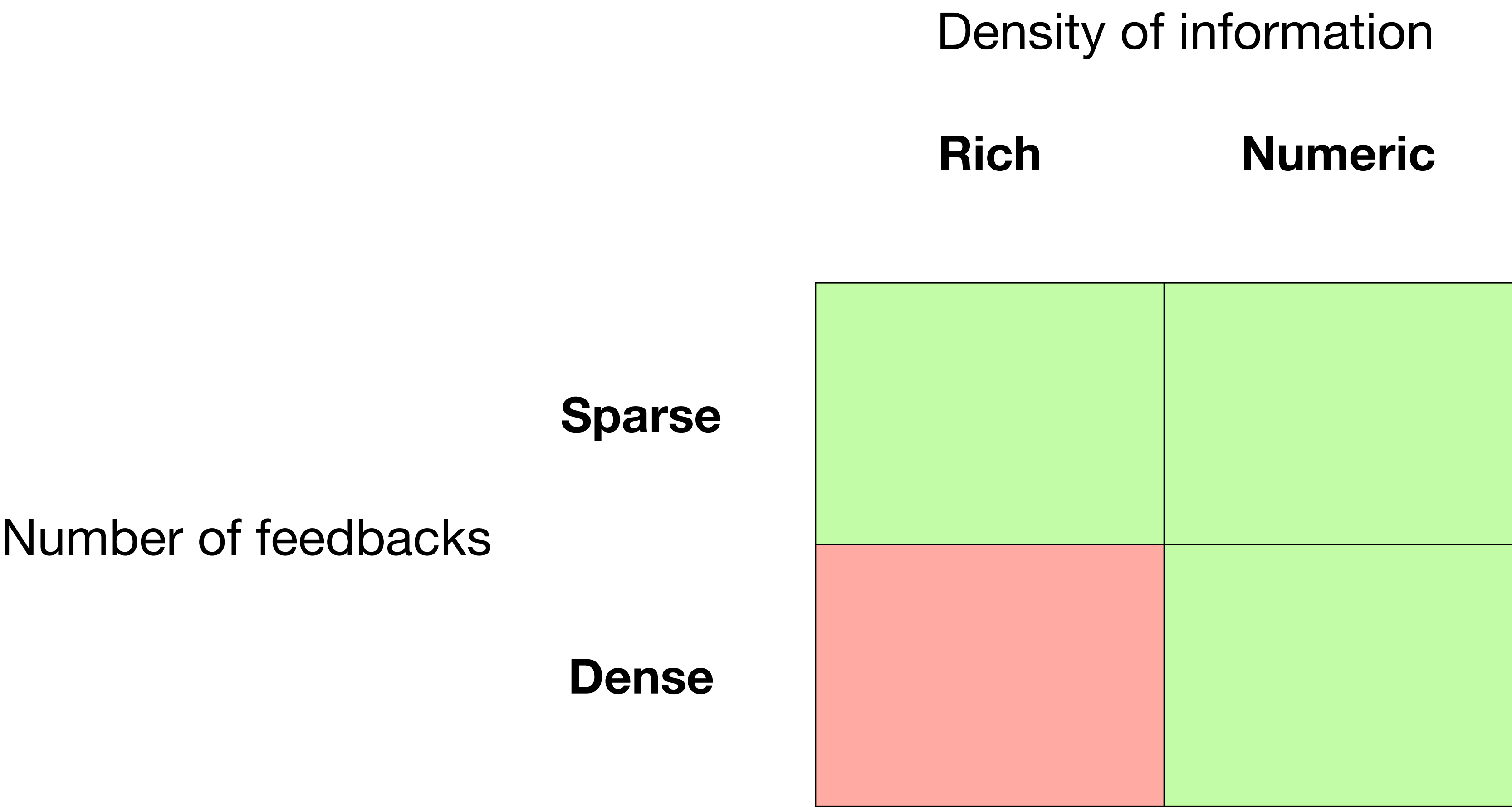
Part 1: are we using the right **objective?**

- Reward maximization may not always be the most suitable objective.
- MaxRL: optimize maximum likelihood in sampling-based settings with binary rewards.
- MaxRL scales better and shows less diversity collapse.

A Taxonomy for Feedback Signal



A Taxonomy for Feedback Signal



RLVR

Density of information

Rich

Numeric

Sparse

RLVR

Number of feedbacks

Dense

- Scalable but inefficient:
- One bit signal per thousands of tokens

Supervised Learning

Density of information

Rich

Numeric

Sparse

Number of feedbacks

Dense

	RLVR
Supervise Learning	

- Not scalable in training frontier models — require human demonstration.
- Most of time, full demonstration is unnecessary.

Process Reward Model

Density of information

Rich

Numeric

Sparse

RLVR

Number of feedbacks

Dense

Supervise Learning

Process Reward Models

Process Reward Model

4.2. Unsuccessful Attempts

In the early stages of developing DeepSeek-R1, we also encountered failures and setbacks along the way. We share our failure experiences here to provide insights, but this does not imply that these approaches are incapable of developing effective reasoning models.

Process Reward Model (PRM) PRM is a reasonable method to guide the model toward better approaches for solving reasoning tasks ([Lightman et al., 2023](#); [Uesato et al., 2022](#); [Wang et al., 2023](#)). However, in practice, PRM has three main limitations that may hinder its ultimate success. First, it is challenging to explicitly define a fine-grain step in general reasoning. Second, determining whether the current intermediate step is correct is a challenging task. Automated annotation using models may not yield satisfactory results, while manual annotation is not conducive to scaling up. Third, once a model-based PRM is introduced, it inevitably leads to reward hacking ([Gao et al., 2022](#)), and retraining the reward model needs additional training resources and it complicates the whole training pipeline. In conclusion, while PRM demonstrates a good ability to rerank the top-N responses generated by the model or assist in guided search ([Snell et al., 2024](#)), its advantages are limited compared to the additional computational overhead it introduces during the large-scale reinforcement learning process in our experiments.

Density of informa

Rich

Nun

Sparse

RLVR

Number of feedbacks

Dense

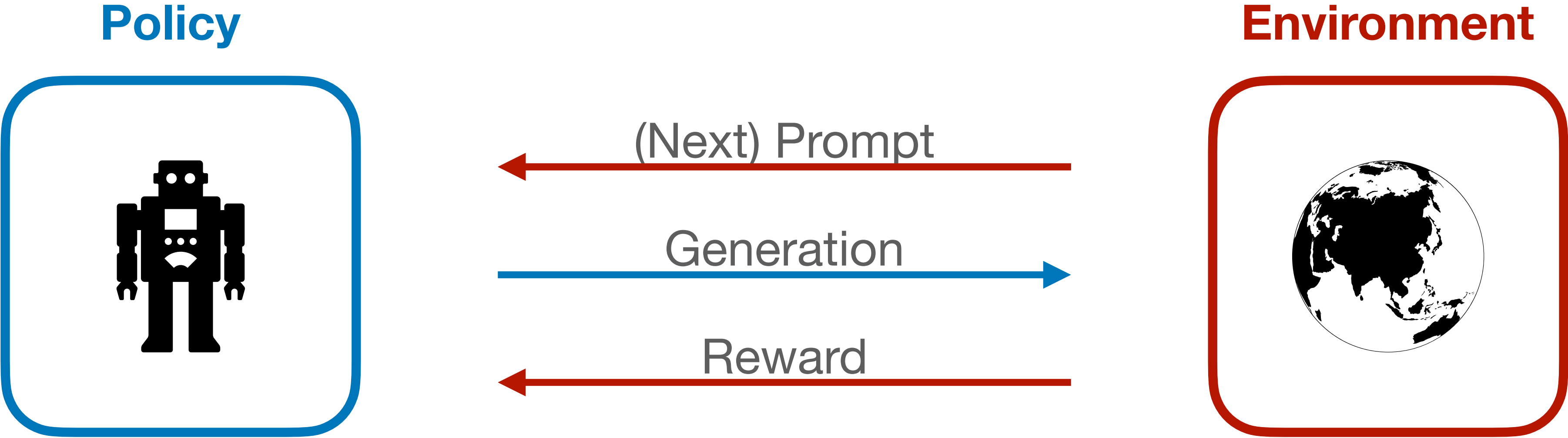
Supervise Learning

Process Reward Models

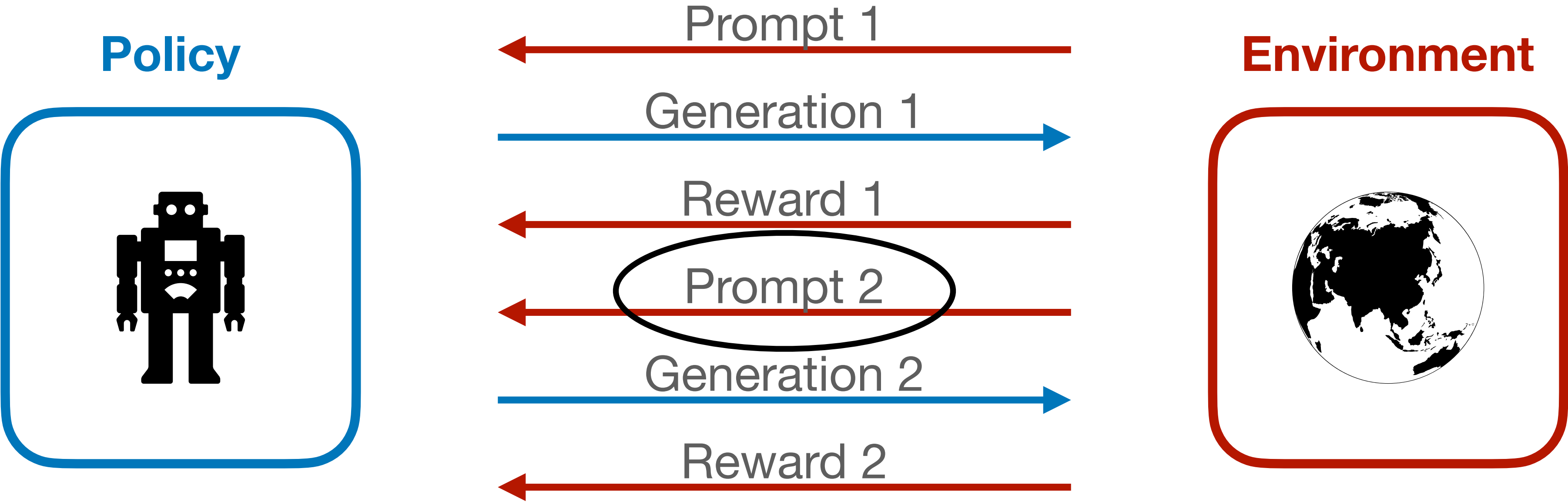
RL from Text Feedback

		Density of information	
		Rich	Numeric
Number of feedbacks	Sparse	We can get this signal “for free”! RLTF	RLVR
	Dense	Supervise Learning	Process Reward Models

RL Interaction Protocol

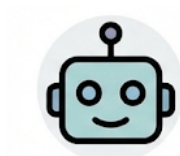


RL Interaction Protocol



Next Prompt Contains Rich Information

Human-AI Interaction

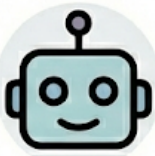


Here is my proof:
Since x is an even number, we can write $x = 2k$ for some integer k .
Therefore, $x + 1 = 2k + 1$.

Your proof is incomplete. You haven't shown that $2k + 1$ is an odd number.



Interpreter



Can you write a Python function to calculate the factorial of a number?



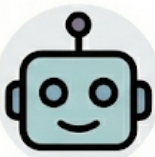
Environment:
Code Interpreter

Here is the factorial code:

```
```python
def factorial(n):
 if n == 0:
 return 1
 else:
 return n * factorial(n-1)
```
```

> Executing: factorial(5)

Output:
120



> Executing: factorial(-1)

Error:
RecursionError:
maximum recursion depth
exceeded in comparison



Question: Solve the system of linear equations:

1. $x + y + z = 6$
2. $2x - y + 3z = 9$
3. $3x + 2y - z = 4$

Generation:

To find the solution for the variables x , y , and z , we can use the method of substitution.

Step 1: Isolate z in the first equation...

Step 2: Substitute z into the second equation...

Step 3: Substitute z into the third equation.

Next, substitute the expression for z into the third equation:

$$3x + 2y - (6 - x - y) = 4$$

Distribute the negative sign:

$$3x + 2y - 6 + x + y = 4$$

Combine like terms ($3x - x$ and $2y + y$):

$$2x + y - 6 = 4$$

Add 6 to both sides:

$$2x + y = 10 \quad \text{(Equation B)}$$

Step 4: Solve the system of Equation A and Equation B...

Step 5: Solve for x ...

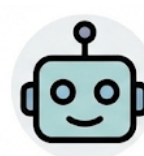
Step 6: Solve for z

Final Answer:

The solution to the system of equations is $x = \frac{31}{7}$, $y = \frac{8}{7}$, and $z = \frac{3}{7}$.

Rule of Prompt 2

Human-AI Interaction

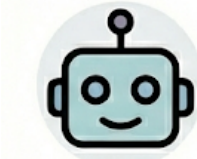


Here is my proof:
Since x is an even number, we can write $x = 2k$ for some integer k .
Therefore, $x + 1 = 2k + 1$.



Your proof is incomplete. You haven't shown that $2k + 1$ is an odd number.

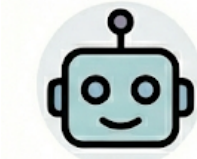
Interpreter



Can you write a Python function to calculate the factorial of a number?



Environment:
Code Interpreter



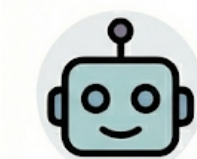
Here is the factorial code:

```
```python
def factorial(n):
 if n == 0:
 return 1
 else:
 return n * factorial(n-1)
```
```

> Executing: factorial(5)

Output:
120

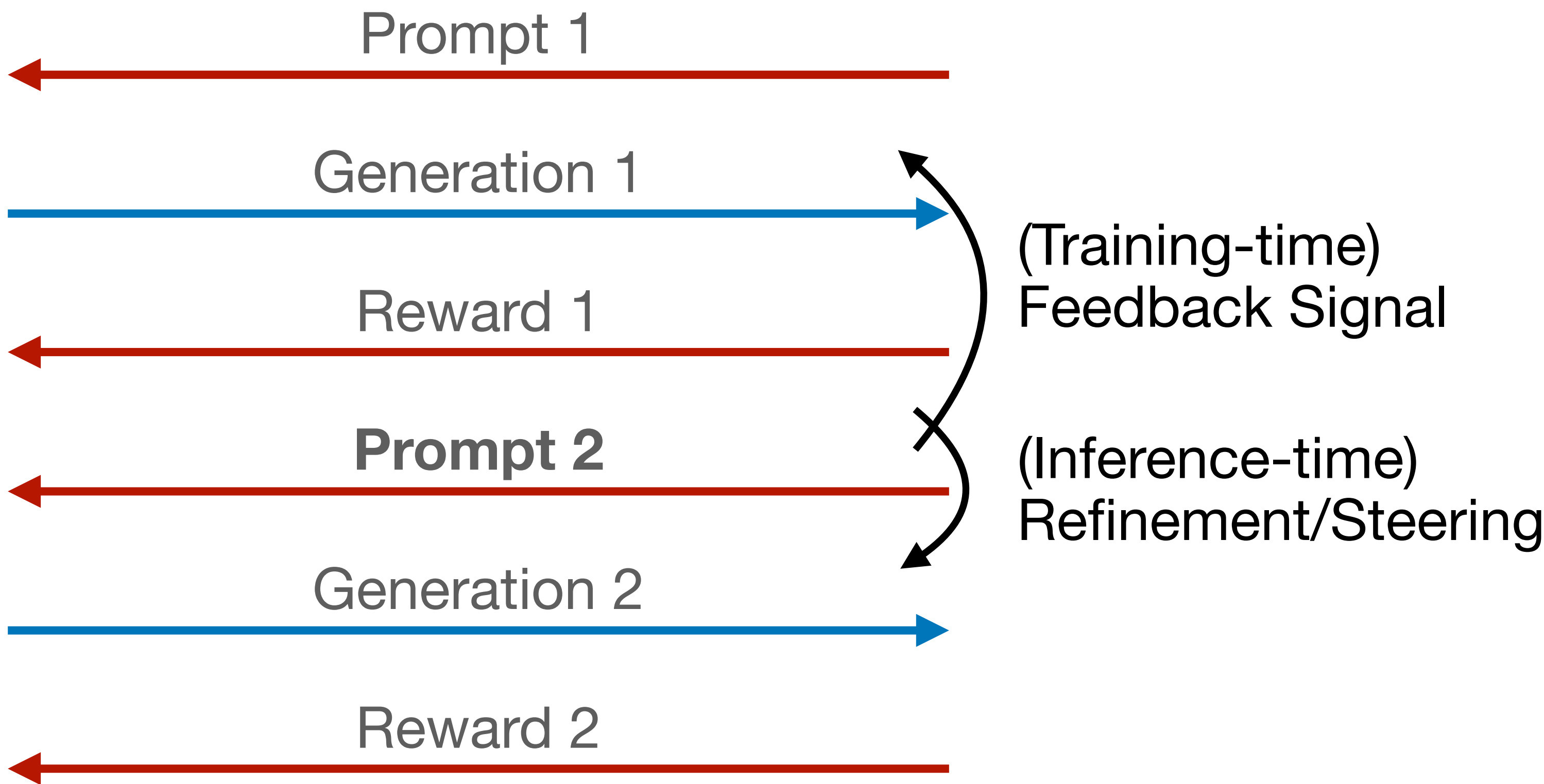




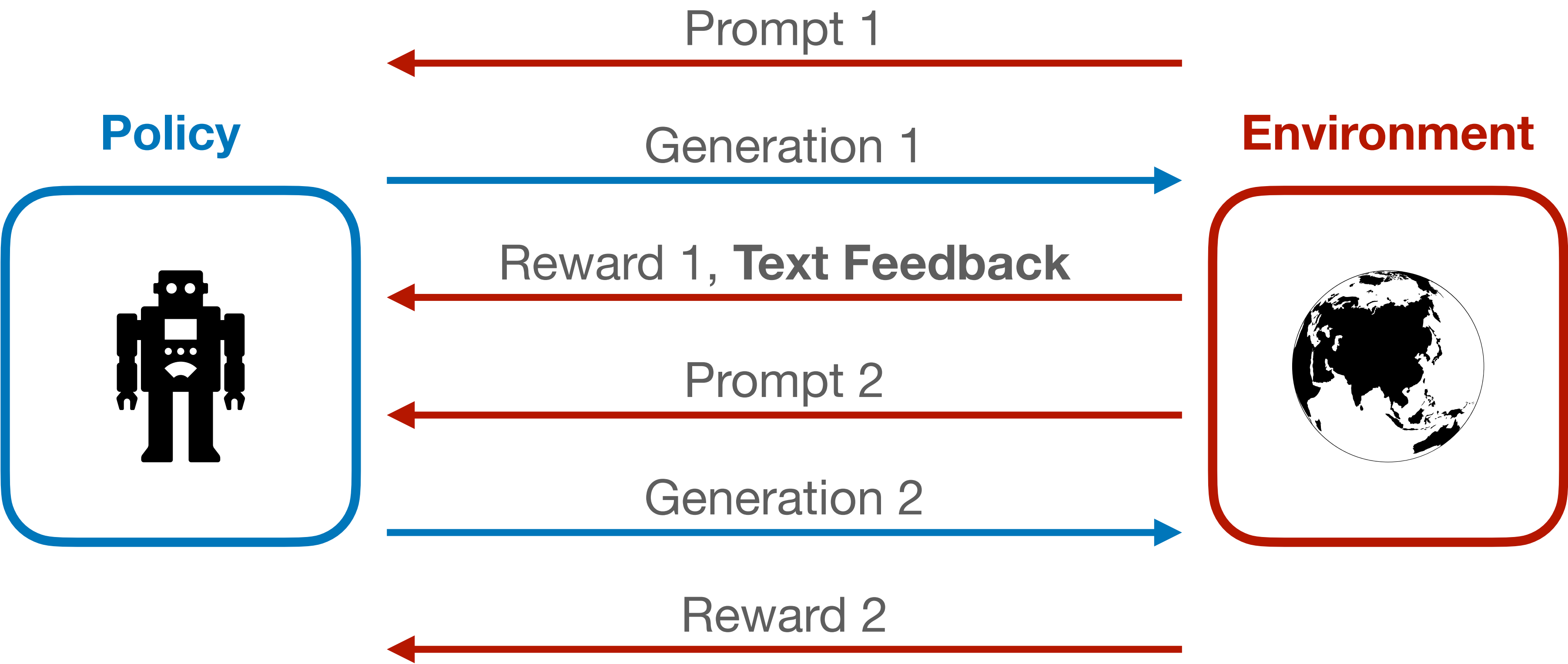
> Executing: factorial(-1)

Error:
RecursionError:
maximum recursion depth
exceeded in comparison

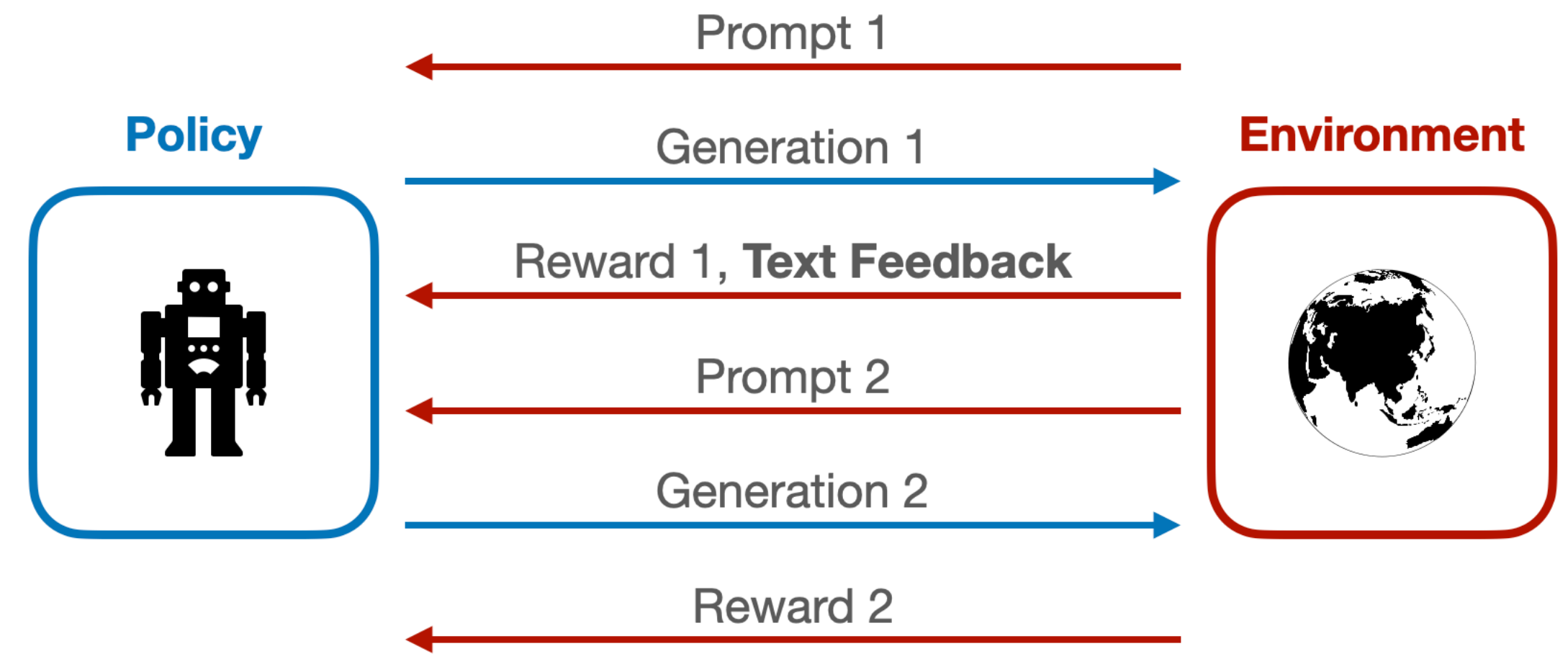




RL from Text Feedback

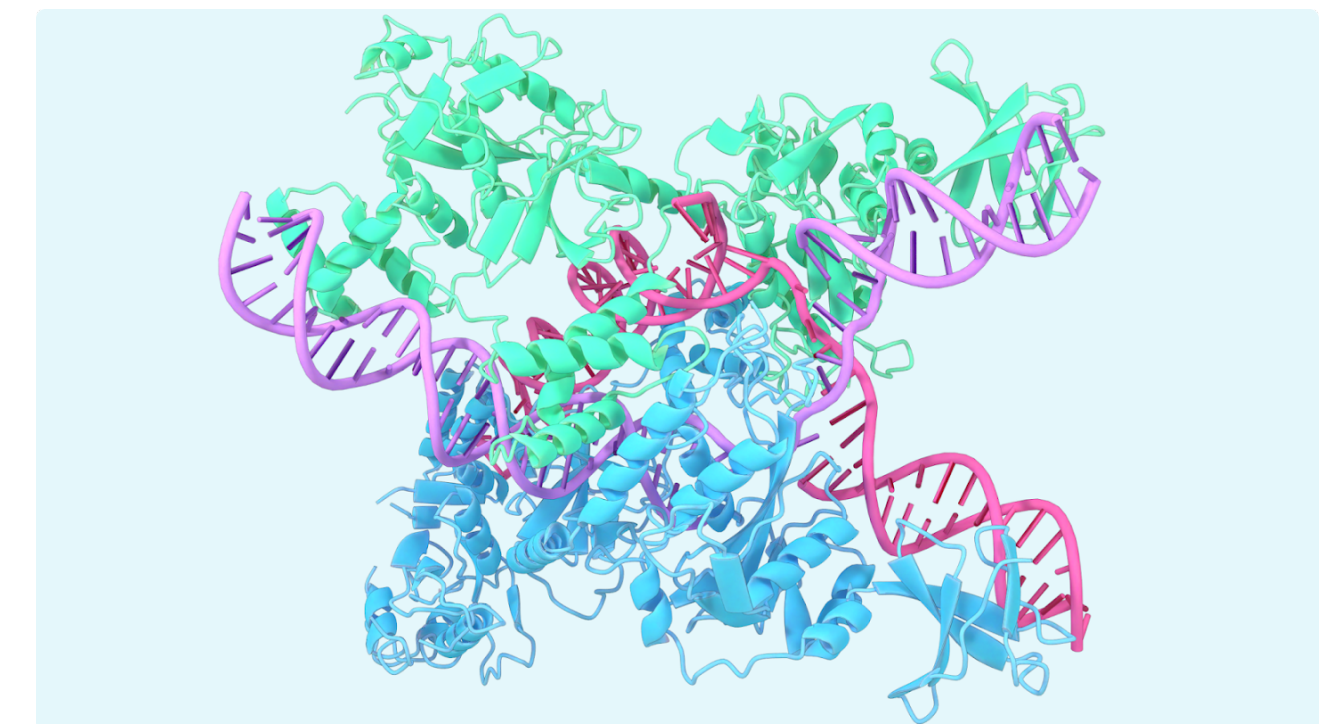


RLTF is the Paradigm towards Continual Learning



Deploy a post-trained model and continue adapting:

- Local customization by learning from user preferences.
- Adapting to company's internal data and tools while interacting with users.
- Scientific discovery by learning from interpreter output or experiment outcomes.



RL from Text Feedback

Main Learning Goal:

$$\max_{\pi} \mathbb{E}_{x_1, y_1}^{\pi} [R(x_1, y_1)]$$

Feedback may not exist at final evaluation

Can't we just run RL with regular objective

$$\mathbb{E}^{\pi} \left[\sum_{h=1}^H R(x_h, y_h) \right] ?$$

$x_1 \sim \mu$ initial prompt

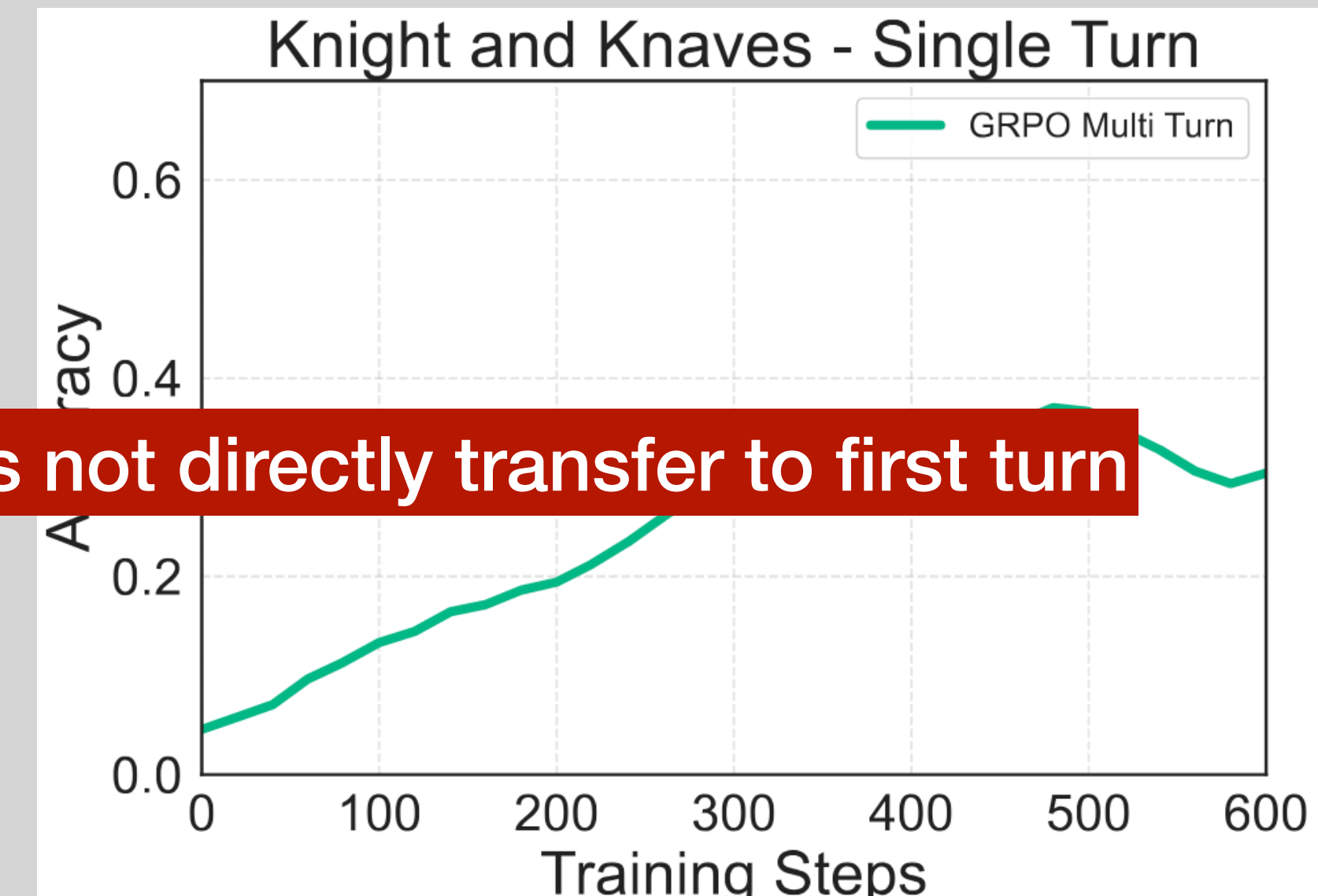
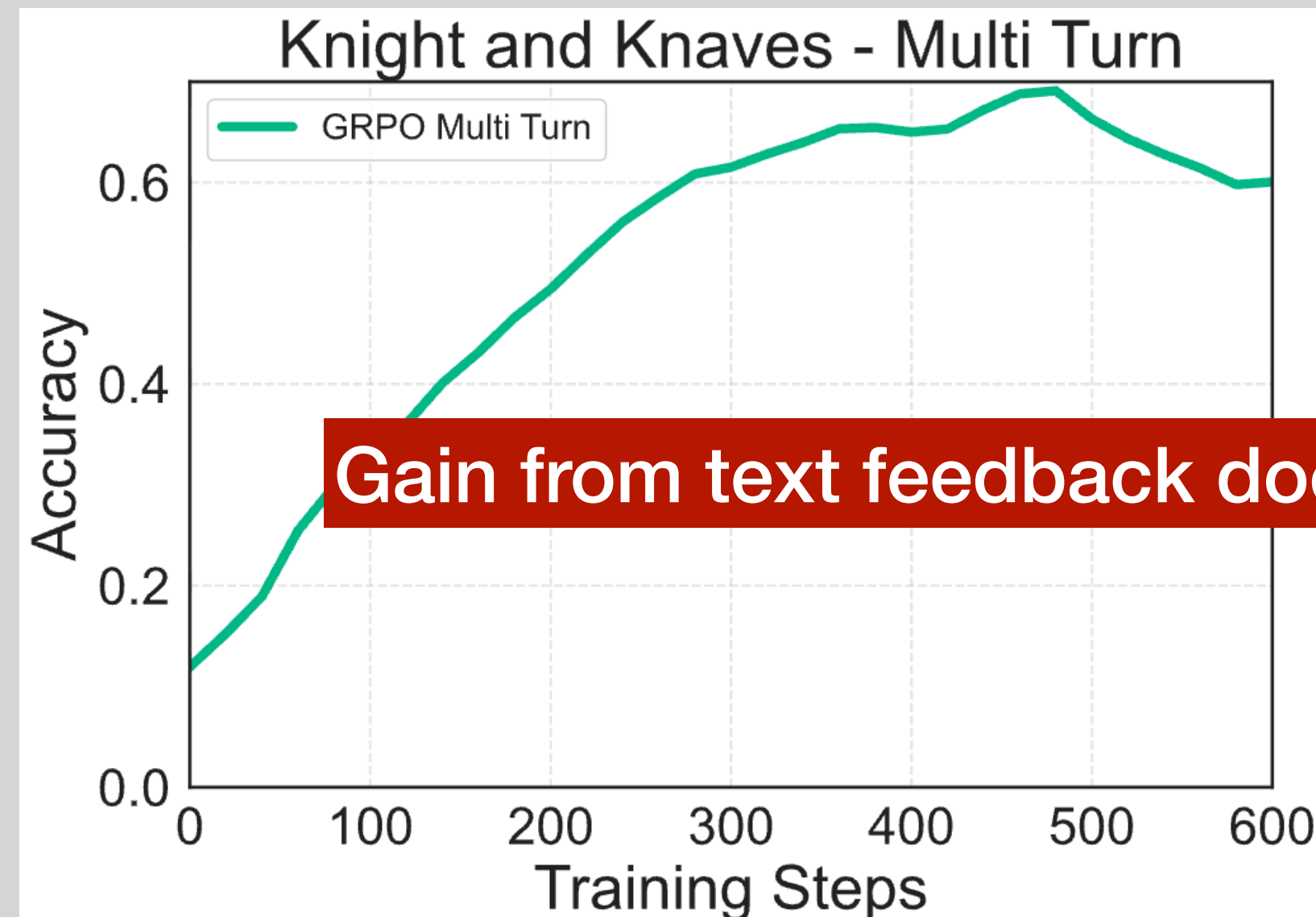
$y_1 \sim \pi(\cdot | x_1)$

feedback $\sim \text{Env}(\cdot | x_1, y_1)$, $r_1 = R(x_1, y_1)$

$x_2 = f(x_1, y_1, \text{feedback})$

$y_2 \sim \pi(\cdot | x_2)$

$r_2 = R(x_1, y_2)$



Gain from text feedback does not directly transfer to first turn

Two Ways of Feedback Internalization

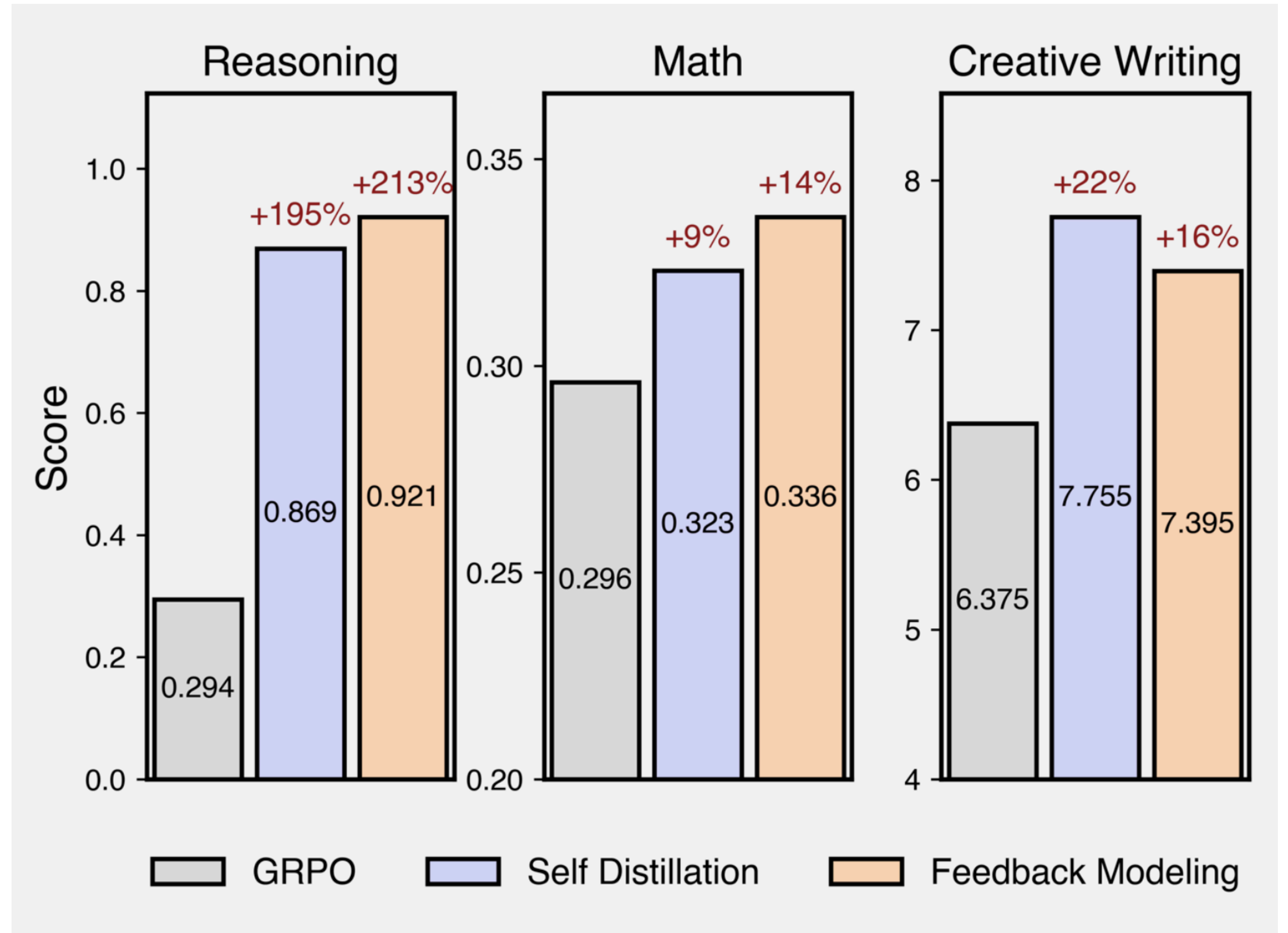
- Observation 1: Directly learning the feedback already provides a rich learning signal
 - Feedback Modeling
- Observation 2: The feedback conditioned policy usually has a better generation distribution than the original policy.
 - Distill the feedback conditioned policy as a teacher policy.

- Combine these objective with the RL objective $\mathbb{E}^{\pi} \left[\sum_{h=1}^H R(x_h, y_h) \right]$

Result Overview

- Observation 1: Directly learning the feedback already provides a rich learning signal
 - **Feedback Modeling**
- Observation 2: The feedback conditioned policy usually has a better generation distribution than the original policy.
 - **Distill the feedback conditioned policy as a teacher policy.**
- Combine these objective with the RL

objective $\mathbb{E}^{\pi} \left[\sum_{h=1}^H R(x_h, y_h) \right]$



Feedback Modeling

- Auxiliary loss: predicting the feedback given the prompt and generation:

$$\log \pi(\text{feedback} \mid x_1, y_1)$$

- Combined objective:

$$\mathbb{E}^{\pi}[r_1 + r_2] + \mathbb{E}^{\text{sg}[\pi]}[\log \pi(\text{feedback} \mid x_1, y_1)]$$

- Not learning better generation directly, but intuitively:
 - Self-consistency
 - Representation Learning (shown in simple settings in the paper)

$$x_1 \sim \mu$$

$$y_1 \sim \pi(\cdot \mid x_1)$$

$$\text{feedback} \sim \text{Env}(\cdot \mid x_1, y_1), \\ r_1 = R(x_1, y_1)$$

$$x_2 = f(x_1, y_1, \text{feedback})$$

$$y_2 \sim \pi(\cdot \mid x_2)$$

$$r_2 = R(x_1, y_2)$$

Test-time Scaling with Feedback Modeling

We almost always do not have the feedback provider at time-time

$$x_1 \sim \mu$$

$$y_1 \sim \pi(\cdot | x_1)$$

$$\text{feedback} \sim \text{Env}(\cdot | x_1, y_1), r_1 = R(x_1, y_1)$$

$$x_2 = f(x_1, y_1, \text{feedback})$$

$$y_2 \sim \pi(\cdot | x_2)$$

$$r_2 = R(x_1, y_2)$$

$$x_1 \sim \mu$$

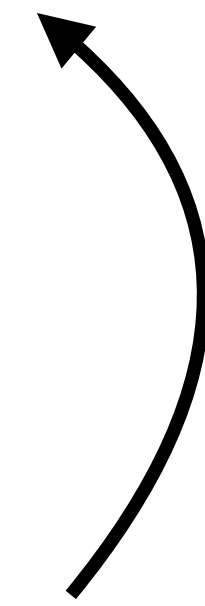
$$y_1 \sim \pi(\cdot | x_1)$$

$$\widehat{\text{feedback}} \sim \pi(\cdot | x_1, y_1)$$

$$\widehat{x}_2 = f(x_1, y_1, \widehat{\text{feedback}})$$

$$\widehat{y}_2 \sim \pi(\cdot | \widehat{x}_2)$$

$$\widehat{\text{feedback}} \sim \pi(\cdot | x_2, y_2)$$



Test-time Scaling with Feedback Modeling

Feedback Modeling enables test-time scaling

$$x_1 \sim \mu$$

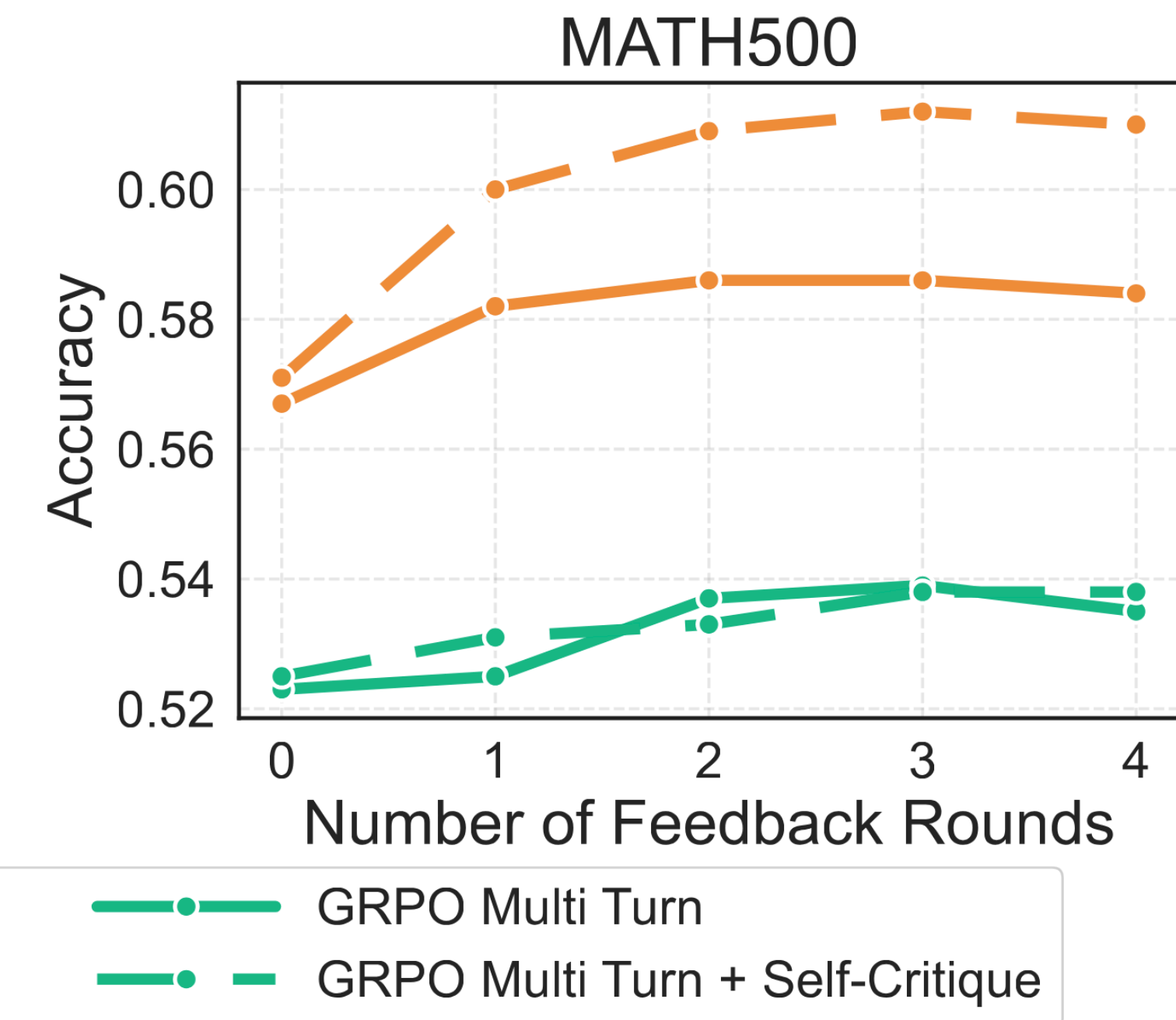
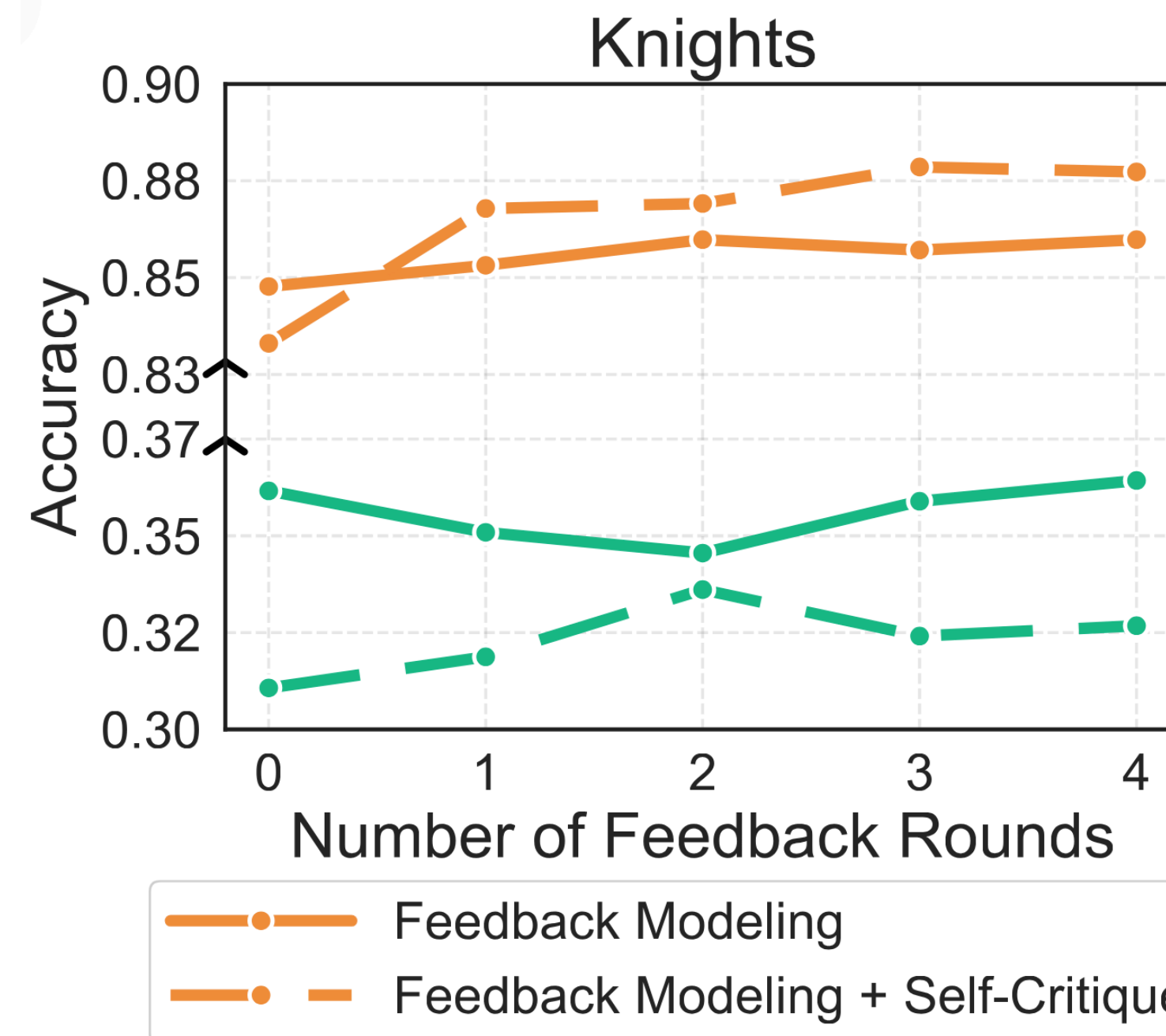
$$y_1 \sim \pi(\cdot | x_1)$$

$$\widehat{\text{feedback}} \sim \pi(\cdot | x_1, y_1)$$

$$\widehat{x}_2 = f(x_1, y_1, \widehat{\text{feedback}})$$

$$\widehat{y}_2 \sim \pi(\cdot | \widehat{x}_2)$$

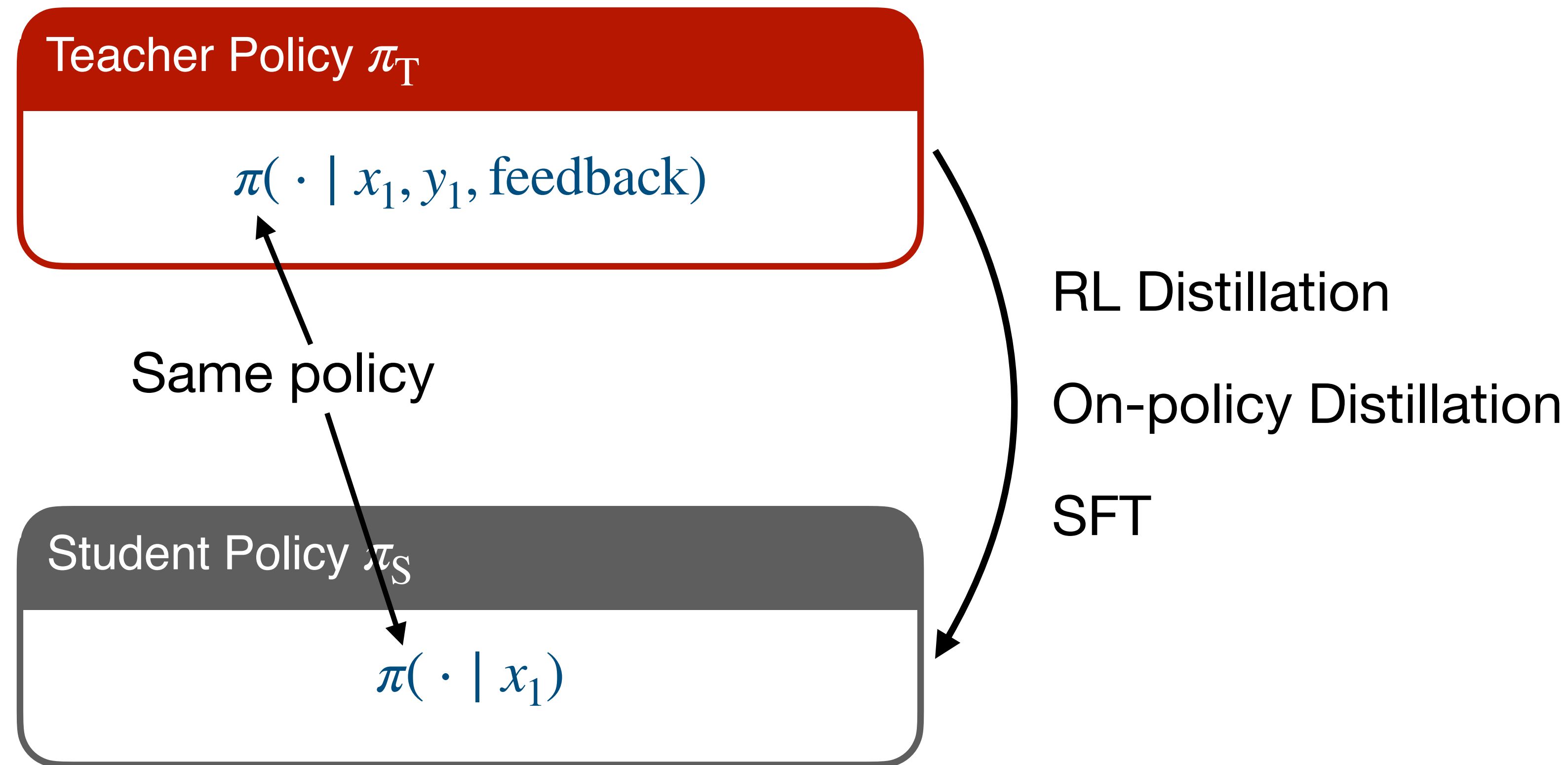
$$\widehat{\text{feedback}} \sim \pi(\cdot | x_2, y_2)$$



Two Ways of Feedback Internalization

- Observation 1: Directly learning the feedback already provides a rich learning signal
 - Feedback Modeling
- Observation 2: The feedback conditioned policy usually has a better generation distribution than the original policy.
 - Distill the feedback conditioned policy as a teacher policy.

Self-distillation



$$x_1 \sim \mu$$

$$y_1 \sim \pi(\cdot \mid x_1)$$

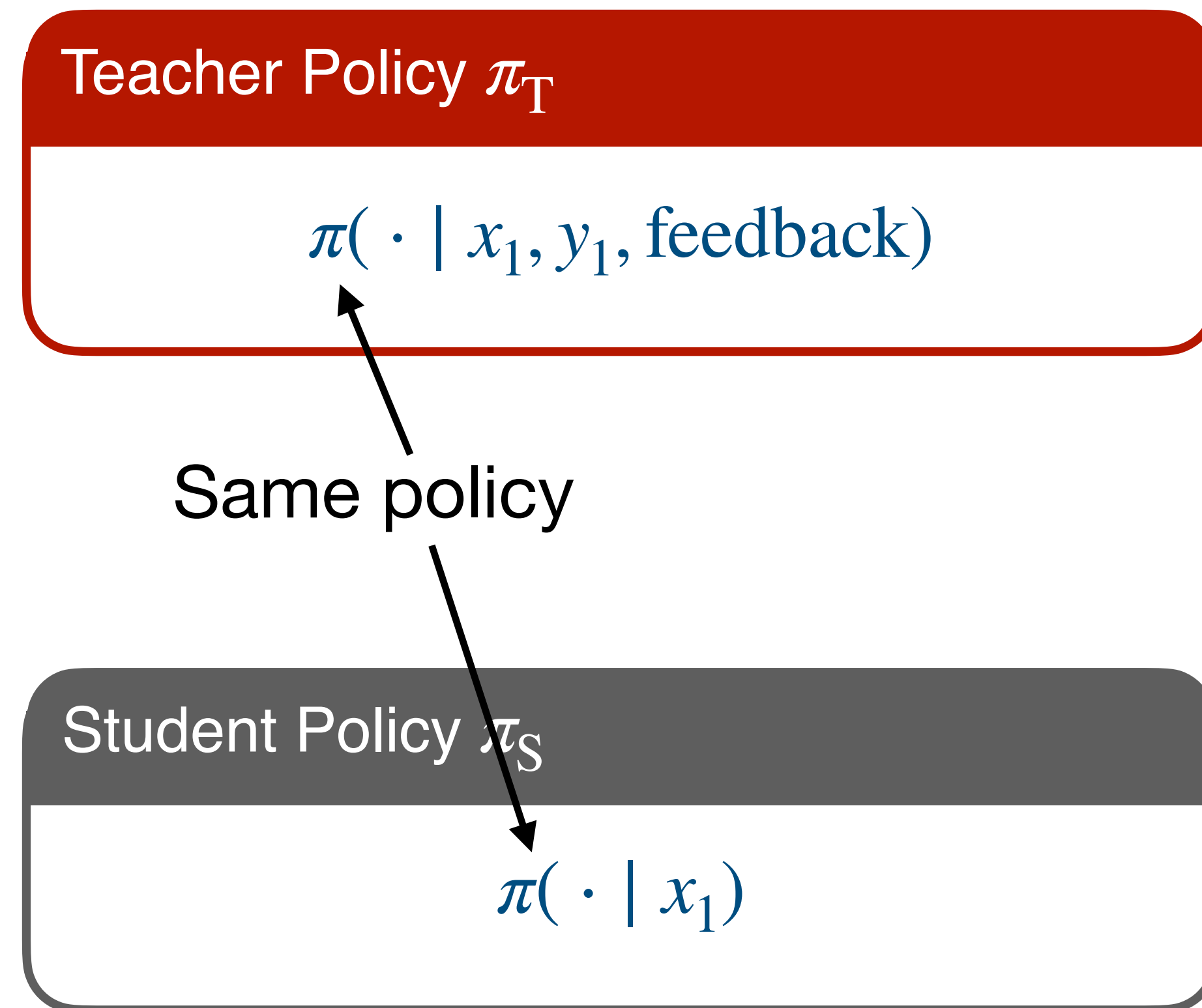
$$\text{feedback} \sim \text{Env}(\cdot \mid x_1, y_1), \\ r_1 = R(x_1, y_1)$$

$$x_2 = f(x_1, y_1, \text{feedback})$$

$$y_2 \sim \pi(\cdot \mid x_2)$$

$$r_2 = R(x_1, y_2)$$

Self-distillation



RL Distillation

On-policy Distillation

SFT

$$x_1 \sim \mu$$

$$y_1 \sim \pi(\cdot | x_1)$$

$$\text{feedback} \sim \text{Env}(\cdot | x_1, y_1), \\ r_1 = R(x_1, y_1)$$

$$x_2 = f(x_1, y_1, \text{feedback})$$

$$y_2 \sim \pi(\cdot | x_2)$$

$$r_2 = R(x_1, y_2)$$

RL Distillation

Idea: using teacher sampling distribution to train the student policy.

RL Distillation Objective:

$$x_1 \sim \mu$$

$$y_1 \sim \pi(\cdot | x_1)$$

$$\text{feedback} \sim \text{Env}(\cdot | x_1, y_1)$$

$$\mathbb{E}_{y_2 \sim \pi_T} \left[\frac{\pi_S(y_2)}{\pi_{\text{ref}}(y_2)} \widehat{A}(y_2) \right]$$

Candidates:
 π_T or π_S

Return estimation with
Variance Reduction

Clipping on importance weighting
E.g., PPO, CISPO



Teacher Policy π_T

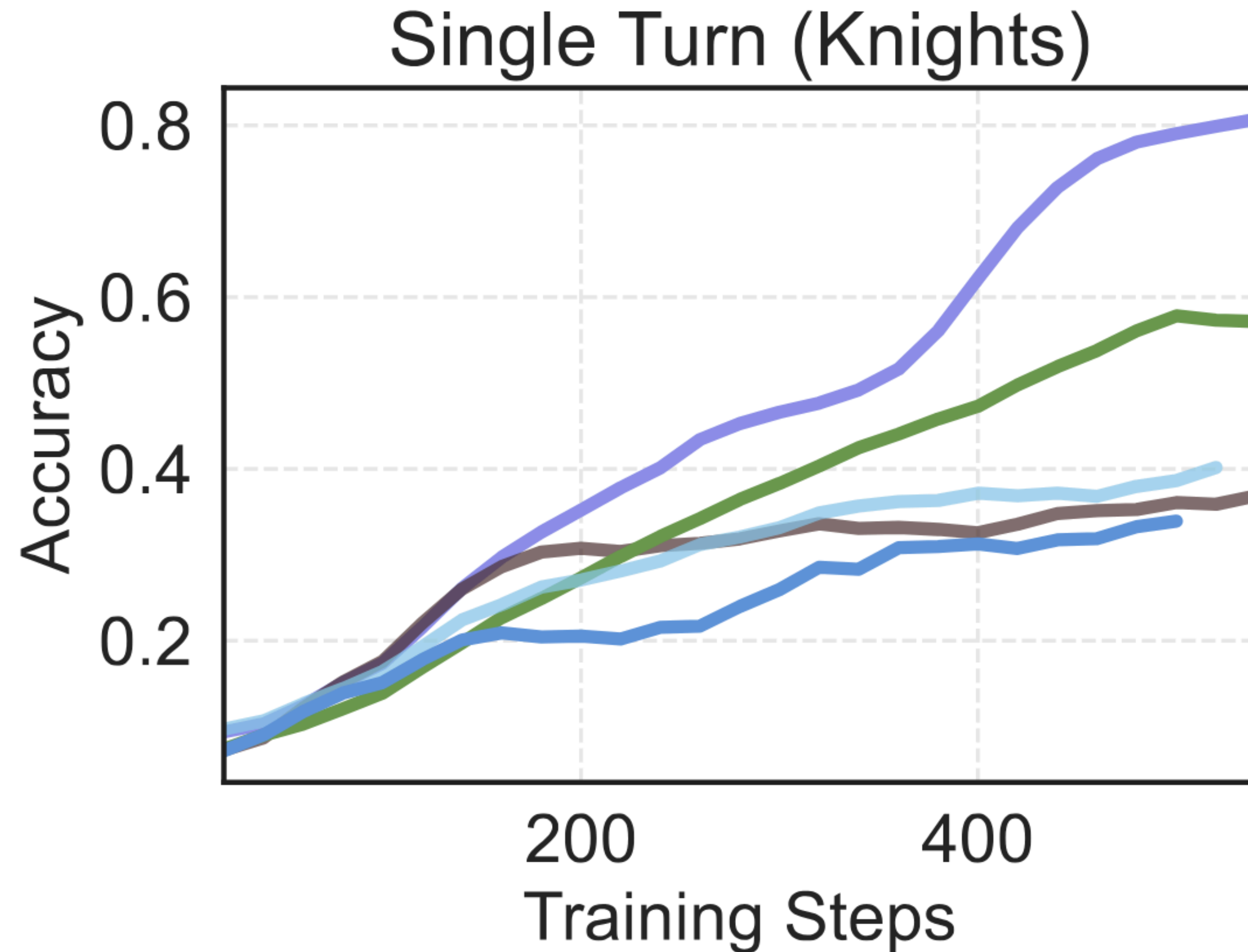
$$\pi(\cdot | x_1, y_1, \text{feedback})$$

Student Policy π_S

$$\pi(\cdot | x_1)$$

- GRPO-style: $\widehat{A}(y_2) = r_2 - \text{mean}(r_2)$
- Ours: $\widehat{A}(y_2) = r_2 - \text{mean}(r_1)$

Ablation



$$\mathbb{E}_{y_2 \sim \pi_T} \left[\frac{\pi_S(y_2)}{\pi_{\text{ref}}(y_2)} \widehat{A}(y_2) \right]$$

$$\pi_{\text{ref}} = \pi_S, \widehat{A}(y_2) = r_2 - \text{mean}(r_1)$$

$$\pi_{\text{ref}} = \pi_S, \widehat{A}(y_2) = r_2 - \text{mean}(r_2)$$

$$\pi_{\text{ref}} = \pi_T, \text{PPO}, \widehat{A}(y_2) = r_2 - \text{mean}(r_1)$$

Rejection sampling

$$\pi_{\text{ref}} = \pi_T, \text{CISPO}, \widehat{A}(y_2) = r_2 - \text{mean}(r_1)$$

$$\pi_{\text{ref}} = \pi_T$$

RL Self-Distillation: Full Objective

RL Distillation Objective:

$$\mathbb{E}_{x_1, y_1, x_2, y_2}^{\pi} \left[r_1 + r_2 + \frac{\pi(y_2 | x_1)}{\text{sg}[\pi(y_2 | x_1)]} [r_2 - \text{mean}(r_1)] \right]$$

RL objective

Self-distillation objective

$$x_1 \sim \mu$$

$$y_1 \sim \pi(\cdot | x_1)$$

$$\text{feedback} \sim \text{Env}(\cdot | x_1, y_1), \\ r_1 = R(x_1, y_1)$$

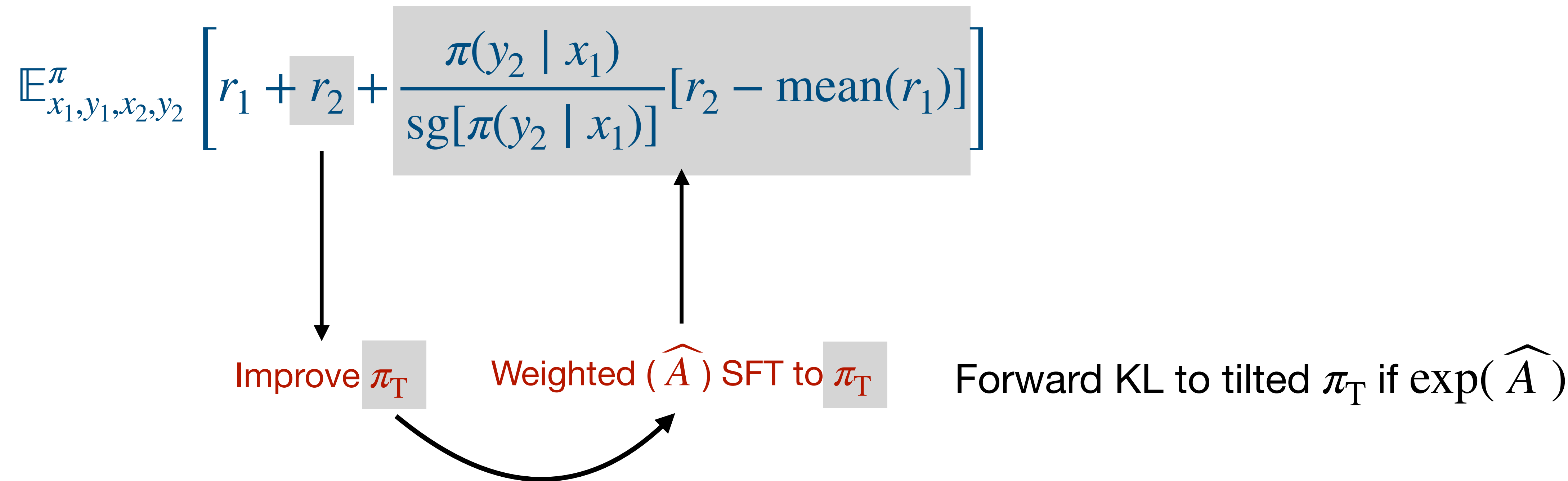
$$x_2 = f(x_1, y_1, \text{feedback})$$

$$y_2 \sim \pi(\cdot | x_2)$$

$$r_2 = R(x_1, y_2)$$

RL Self-Distillation: Interpretation

RL Distillation Objective:



Second Half Time

Protocol

$$s_h, a_h, r_h, \text{feedback}_h, s_{h+1} \sim P^\pi$$

Part 2: why do we only use reward as the only **signal?**

- Text feedback: richer than reward, cheaper than demonstrations, abundant in practice.
- SD and FM enables feedback internalization.
- A paradigm for continual learning⁴⁶

Objective

$$\arg \max_{\pi} \mathbb{E}_x [\log p_{\pi}(x)]$$

Part 1: are we using the right **objective?**

- Reward maximization may not always be the most suitable objective.
- MaxRL: optimize maximum likelihood in sampling-based settings with binary rewards.
- MaxRL scales better and shows less diversity collapse.

Experiment: Full Results

| | Base Model | GRPO
Single turn | GRPO
Multi turn | Feedback
Descent | RLTF-SD | RLTF-FM |
|-------------------------|------------|---------------------|--------------------|---------------------|---------|---------|
| Reasoning | | | | | | |
| Knights and Knaves | 0.058 | 0.373 | 0.352 | 0.055 | 0.802 | 0.880 |
| Binary Matrix | 0.001 | 0.125 | 0.950 | 0.005 | 0.976 | 0.978 |
| Shortest Path | 0.034 | 0.385 | 0.384 | 0.035 | 0.830 | 0.905 |
| Math | | | | | | |
| MATH500 (DAPO) | 0.376 | 0.526 | 0.523 | 0.415 | 0.548 | 0.567 |
| AIME24 (DAPO) | 0.025 | 0.058 | 0.025 | 0.045 | 0.088 | 0.083 |
| MATH500 (DeepMath) | 0.376 | 0.558 | 0.578 | 0.424 | 0.598 | 0.636 |
| AIME24 (DeepMath) | 0.025 | 0.042 | 0.050 | 0.054 | 0.058 | 0.058 |
| Creative Writing | | | | | | |
| LitBench | 4.20 | 6.83 | 6.41 | 8.25 | 8.80 | 8.40 |
| WritingBench | 5.71 | 5.92 | 6.29 | 5.30 | 6.71 | 6.39 |

Expanding the Capabilities of RL via Text Feedback

RLTF



Protocol

$$s_h, a_h, r_h, \text{feedback}_h, s_{h+1} \sim P^\pi$$

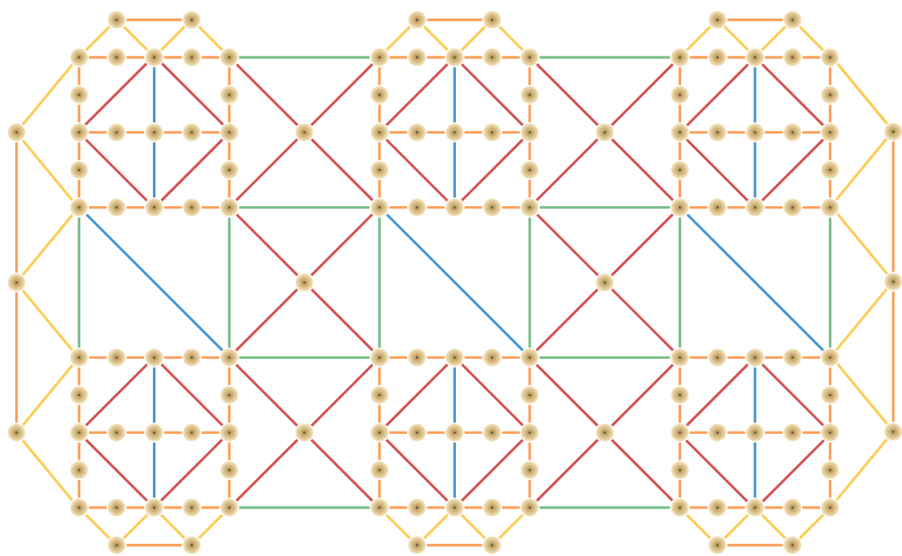
Maximum Likelihood Reinforcement Learning

MaxRL



Objective

$$\arg \max_{\pi} \mathbb{E}_x \left[\log p_{\pi}^{\text{pass}}(x) \right]$$



TinkerToy Computer invented by Daniel Hillis and Brian Silverman

Tinker is a training API for *researchers*

Control every aspect of model training and fine-tuning
while we handle the infrastructure.

[Sign up](#)

[Sign in](#)

[Docs](#)