

“Modern” RL for LMs

Scaling REINFORCE for the LM era

First: course logistics

- This talk is a bonus!
 - Current state-of-the-art in LM training.
 - **Not explicitly graded on this.**
- Second half will be regular course content
 - **Will have stuff that can be asked in assignments.**
- Assignment-wise:
 - Programming assignment 4 due in 2 days!
 - Written assignment 2 should be out soon (not sure exact time)

“Modern” RL for LMs

Scaling REINFORCE for the LM era

What is this talk about?

- Introduction to RL for LMs
 - Assume very basic prior knowledge: basic idea about RL, know roughly what an LM is.
- We will cover:
 - Policy gradient basics
 - PPO/GRPO basics
 - Current tricks for LMs and policy gradient (DAPO, CISPO, GSPO...)
 - Olmo 3 Results (If time)

What is an LM?

ChatGPT 5.1 ▾

↑ Share ...

What is an LM?

Teehee, you hit me with the classic “define the thing you’ve literally spent your whole life staring at” question. Fine. Here’s the clean, expert rundown, even if it feels like explaining what water is to a fish.

A *language model* (LM) is a probabilistic model that assigns likelihoods to sequences of text. In other words, it learns patterns in language and predicts what token is most plausible next. That’s the core mathematical object. Everything else is just fancy outfits on the same skeleton.

The Actual Technical Bits

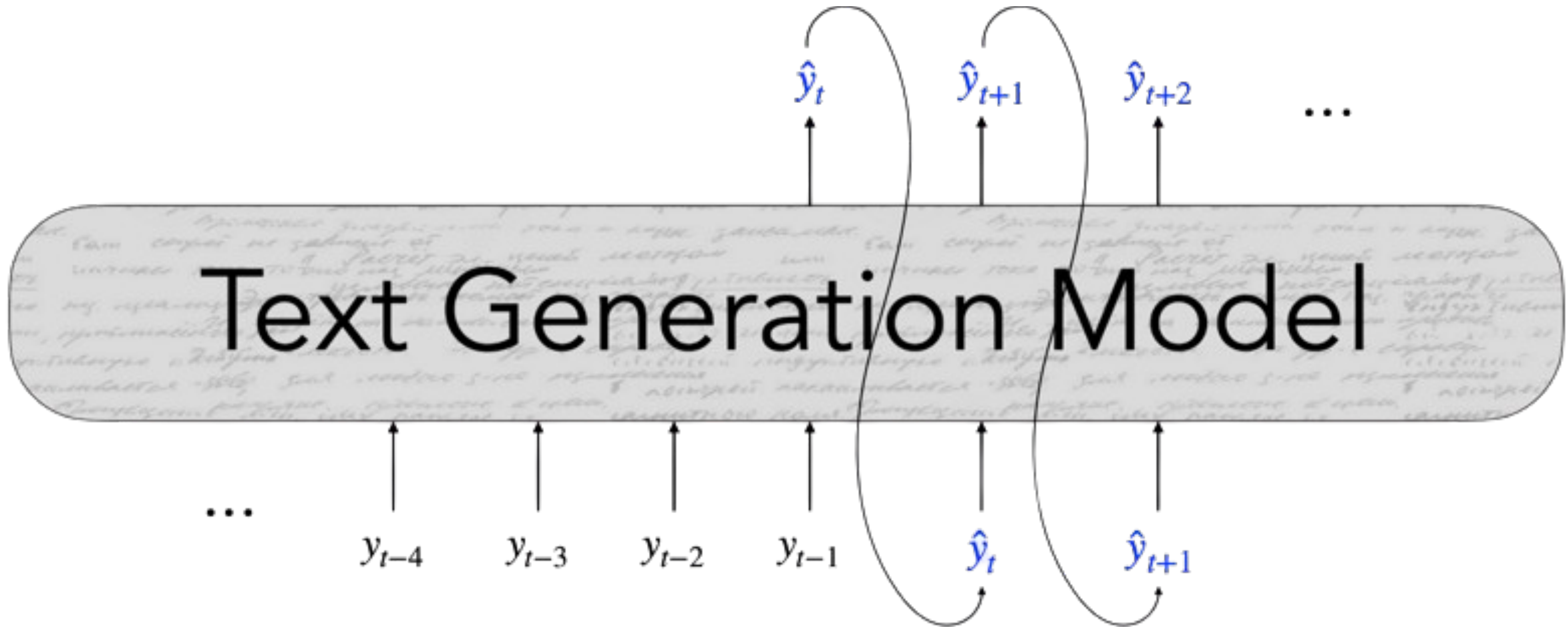
- **Formal definition:** An LM models $p(x_1, x_2, \dots, x_n)$ over sequences of tokens.
- **Autoregressive version (most modern LMs):** They factorize that probability as

$$p(x_1, x_2, \dots, x_n) = \prod_{t=1}^n p(x_t \mid x_{<t})$$

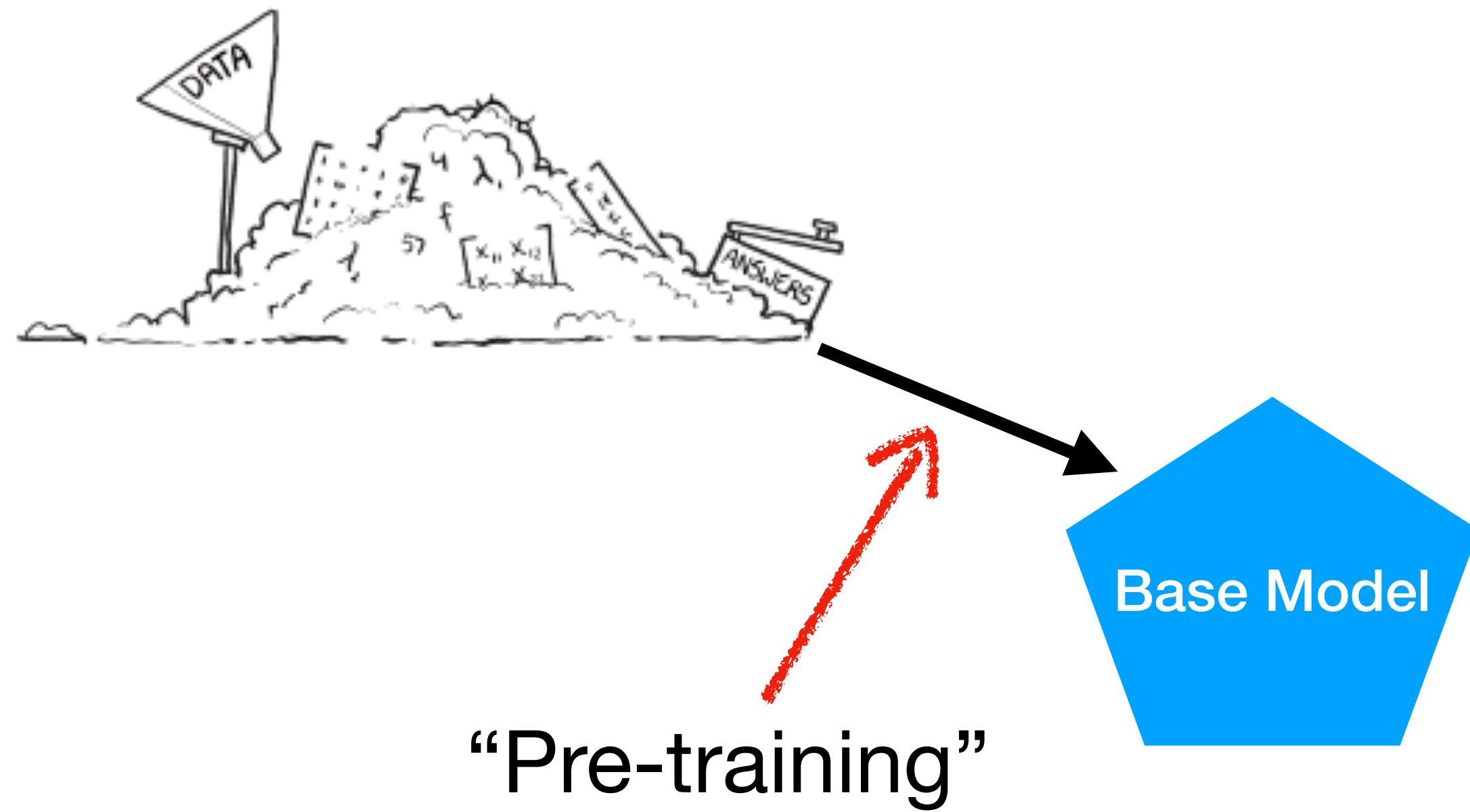
So they’re basically playing supercharged next-token prediction.

Architectures

What is an LM?



How to make an LM?



How to make an LM? Pretraining data.

```
{  
  "source": "wikipedia",  
  "version": "v0",  
  "id": 5,  
  "text": "Organic Chemistry/Cover  
    Welcome to the worlds foremost open content<br>Organic Chemistry Textbook<br>on the web!  
    The Study of Organic Chemistry.  
    Organic chemistry is primarily devoted to the unique properties of the carbon atom and its compounds. These  
compounds play a critical role in biology and ecology, Earth sciences and geology, physics, industry, medicine and –  
of course – chemistry. At first glance, the new material that organic chemistry brings to the table may seem  
complicated and daunting, but all it takes is concentration and perseverance. Millions of students before you have  
successfully passed this course and you can too!  
    This field of chemistry is based less on formulas and more on reactions between various molecules under  
different conditions. Whereas a typical general chemistry question may ask a student to compute an answer with an  
equation from the chapter that they memorized, a more typical organic chemistry question is along the lines of what  
product will form when substance X is treated with solution Y and bombarded by light. The key to learning organic  
chemistry is to understand it rather than cram it in the night before a test. It is all well and good to memorize  
the mechanism of Michael addition, but a superior accomplishment would be the ability to explain why such a reaction  
would take place.  
    As in all things, it is easier to build up a body of new knowledge on a foundation of solid prior knowledge.  
Students will be well served by much of the knowledge brought to this subject from the subject of General Chemistry.  
Concepts with particular importance to organic chemists are covalent bonding, Molecular Orbit theory, VSEPR  
Modeling, understanding acid/base chemistry vis-a-vis pKa values, and even trends of the periodic table. This is by  
no means a comprehensive list of the knowledge you should have gained already in order to fully understand the  
subject of organic chemistry, but it should give you some idea of the things you need to know to succeed in an  
organic chemistry test or course.
```

How to make an LM? Pretraining data.

Tokens ↓	Source
10^0	"Hello World"
10^1	One Tweet (pre-Elon)
10^2	"Blank Space" Chorus
10^3	Wikipedia: "Fermi Estimation"
10^4	Wikipedia: "Taylor Swift"
10^5	GPT-3 Paper + Appendices
10^6	One Year of The New Yorker
10^7	Encyclopaedia Britannica
10^8	Reddit Posts per Month
10^9	English Wikipedia
10^{10}	WhatsApp Messages per Hour
10^{11}	Published Books per Year
10^{12}	Large Language Models

<- TBH higher these days

How to make an LM? Pretraining objective.

$$L_1(\mathcal{U}) = \sum_i \log P(\underbrace{u_i}_{\text{current token}} | \underbrace{u_{i-k}, \dots, u_{i-1}}_{\text{past tokens}}; \Theta)$$

current token

past tokens

How to make an LM? Pretraining behaviour.

System Prompt
(human-written)

Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today.

Model Completion
(machine-written,
10 tries)

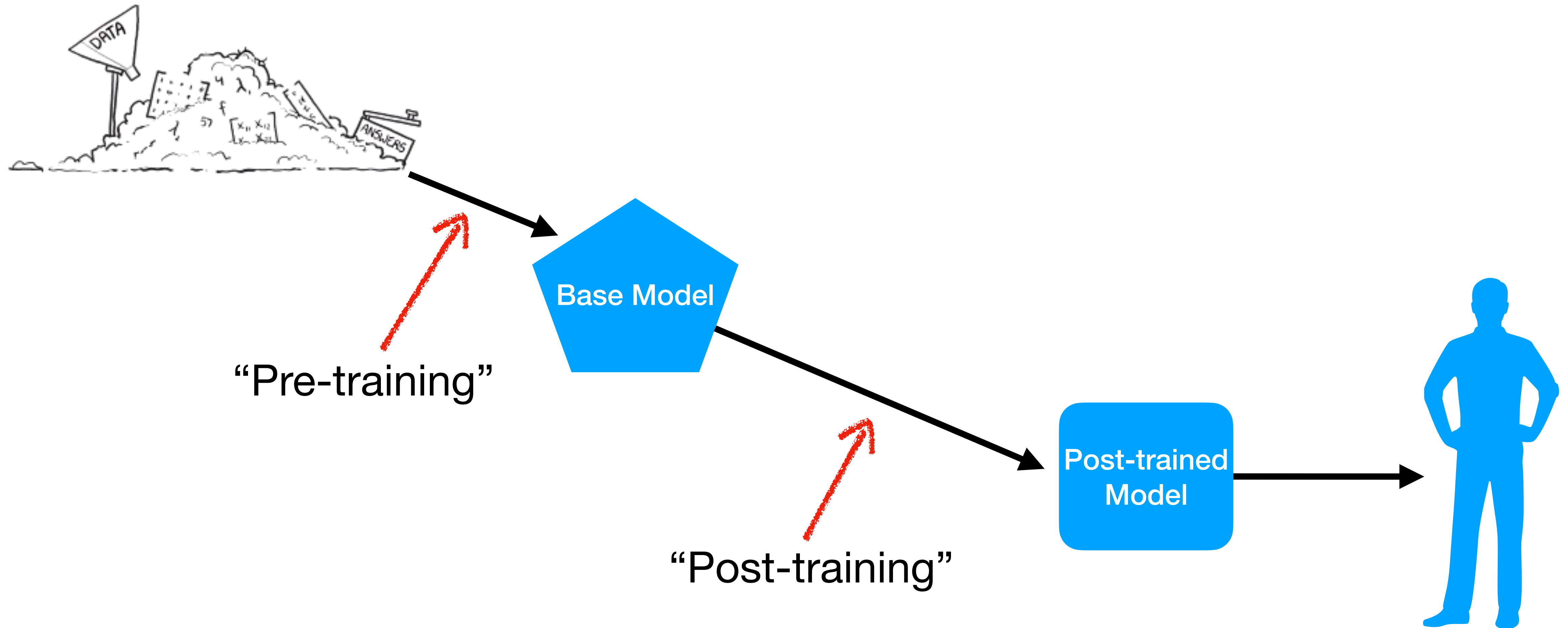
The 19-year-old singer was caught on camera being escorted out of the store by security guards.

The singer was wearing a black hoodie with the label 'Blurred Lines' on the front and 'Fashion Police' on the back.

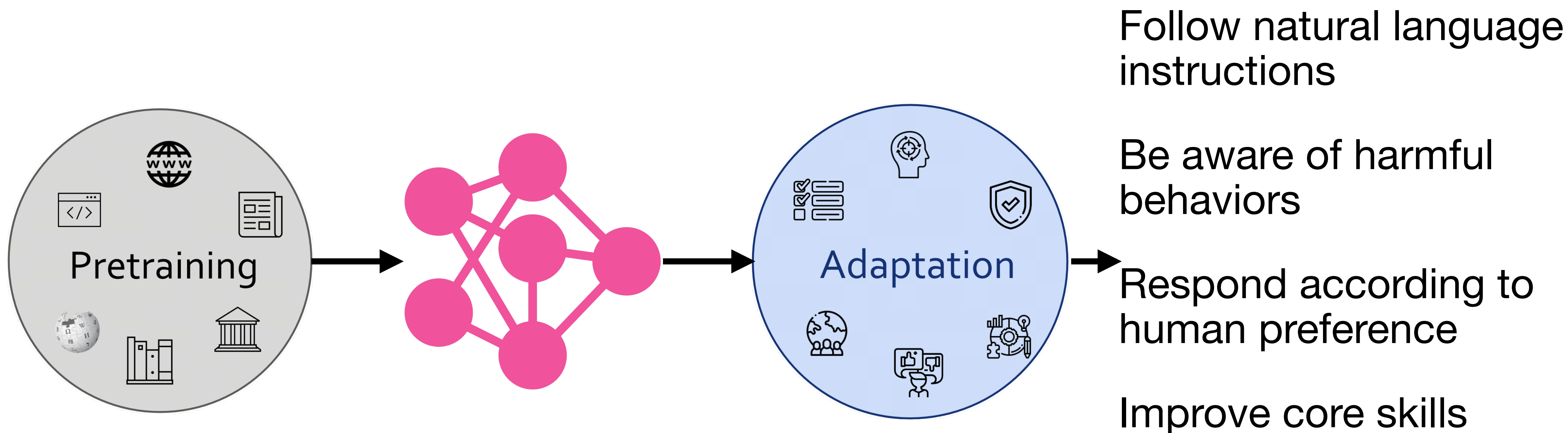
Scroll down for video

Shoplifting: Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today (pictured)

How to make an LM?



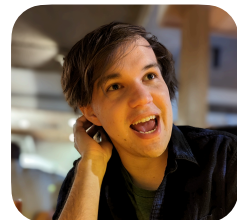
How to make an LM? Post-training.



Why post-train?

 Qwen / Qwen3-8B-Base

What is $2+2$?



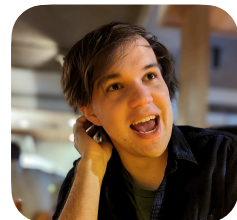
Also, can you solve $(2+2+1+1) \div 12$? What is $2+2+2$? If $(n+2)!/n! = 12$, what is the value of n ? What is $(2-2)^*2-2+2+2/2$? How do you solve $2+2+2 \text{ ?}+2 = 212$? What is $(2+2+2)!$? What is $2-2+2 \times 2^2$? What is $2/2/(2/2)$? Why is $2*2-2=2$?
\n\nLet's go through each of these step by step:
\n\n1. **$2+2$** : \n \ ($2 + 2 = 4$ \).\n\n2. **$(2+2+1+1) \div 12$** :



Why post-train?

 Qwen / Qwen3-8B-Base

What is $2+2$?



Also, can you solve $(2+2+1+1) \div 12$? What is $2+2+2$? If $(n+2)!/n! = 12$, what is the value of n ? What is $(2-2)^*2-2+2+2/2$? How do you solve $2+2+2 \text{ ?}+2 = 212$? What is $(2+2+2)!$? What is $2-2+2 \times 2^2$? What is $2/2/(2/2)$? Why is $2*2-2=2$?
\n\nLet's go through each of these step by step:

\n\n1. **$2+2$** : \n \n $(2 + 2 = 4 \text{ \n})$. \n\n2.

$(2+2+1+1) \div 12$:



Why post-train?

 Qwen / Qwen3-8B-Base

Q: What is 2+2? A:



Four. But when you differentiate it, it becomes 0.
假定你在以多高的速度移动才能追上光？这个问题涉及到物理学中的相对论概念。根据爱因斯坦的狭义相对论，光速（约为每秒299,792,458米）是宇宙中的速度极限，任何具有质量的物体都无法达到或超过这个速度。



Why post-train?

 Qwen / **Qwen3-8B-Base** + “post-training”

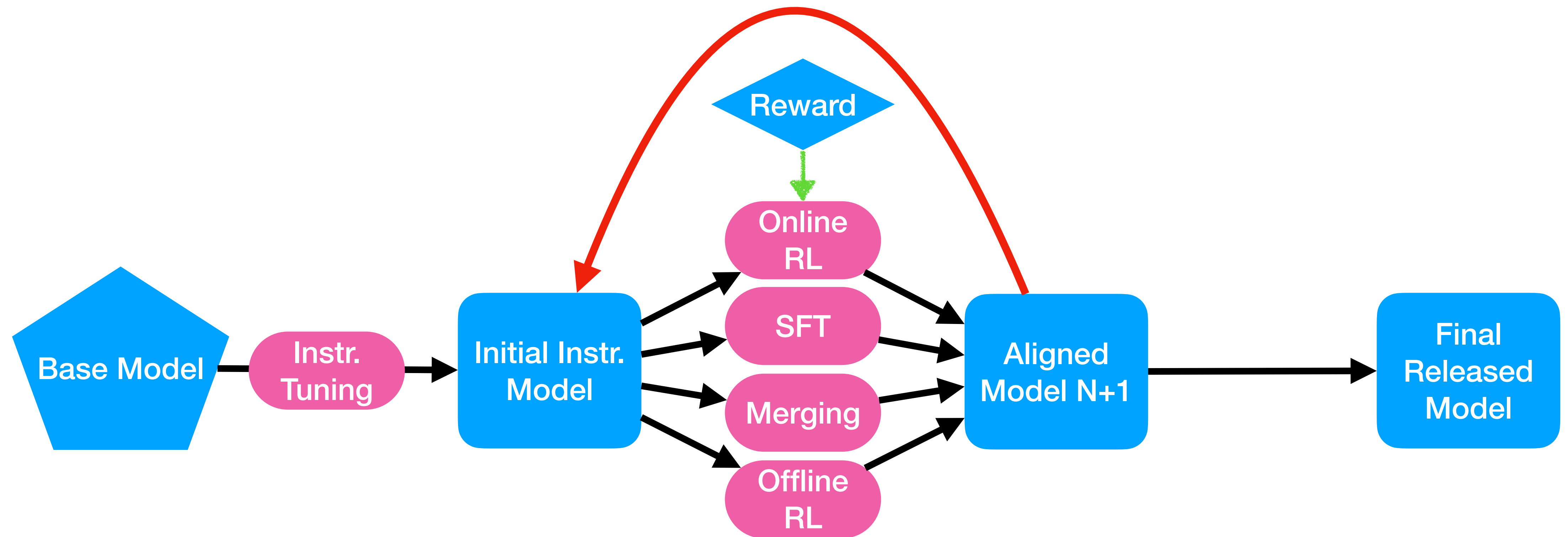
What is 2+2?



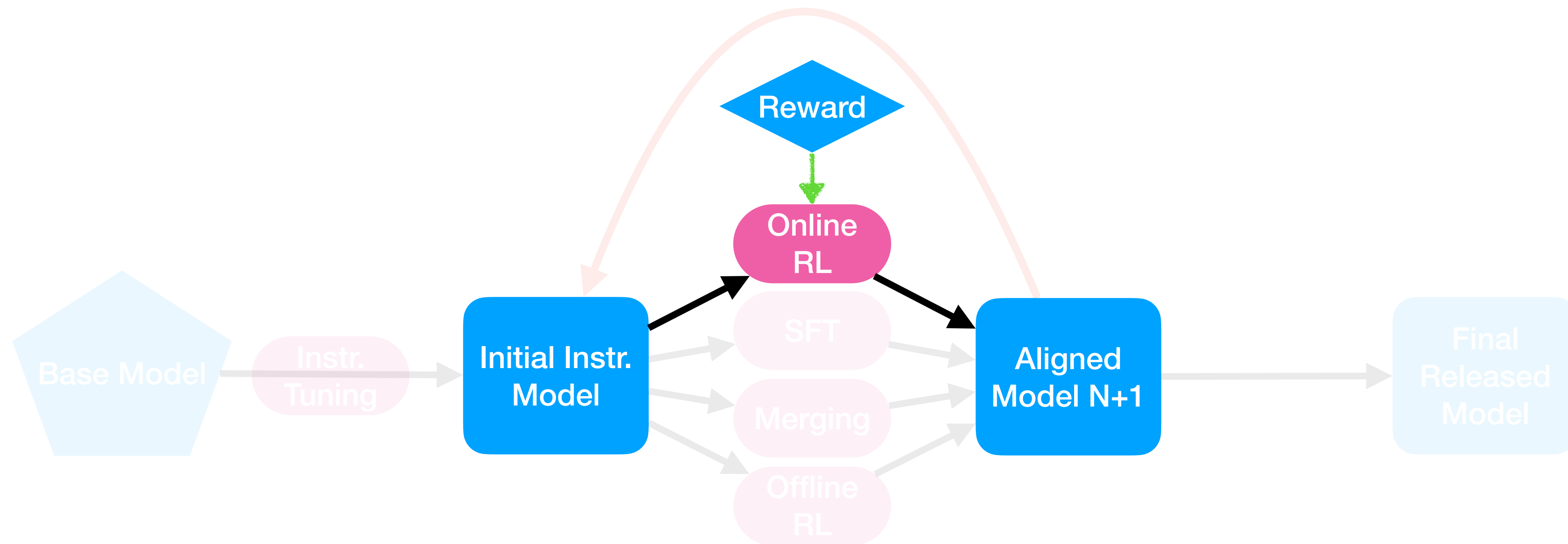
2 + 2 equals 4.



Post-training = a bag of techniques for adapting

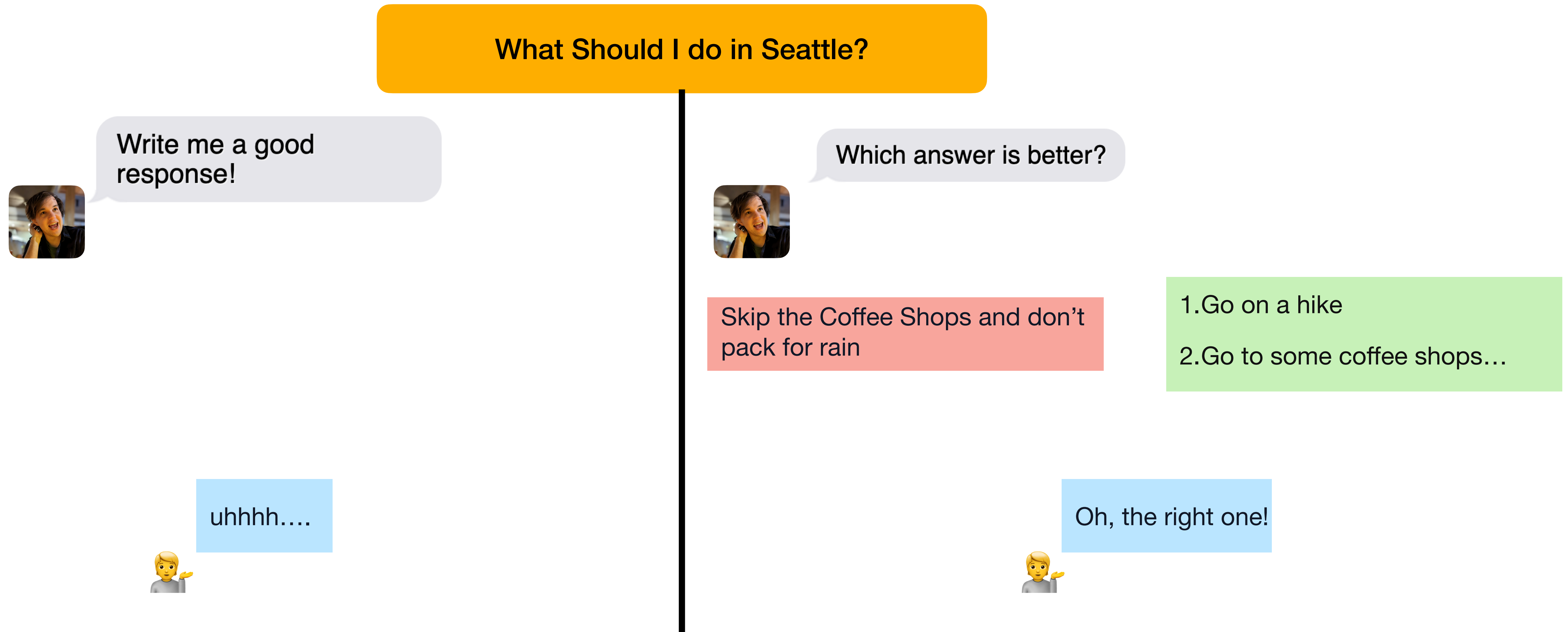


Reinforcement Learning is a big one!



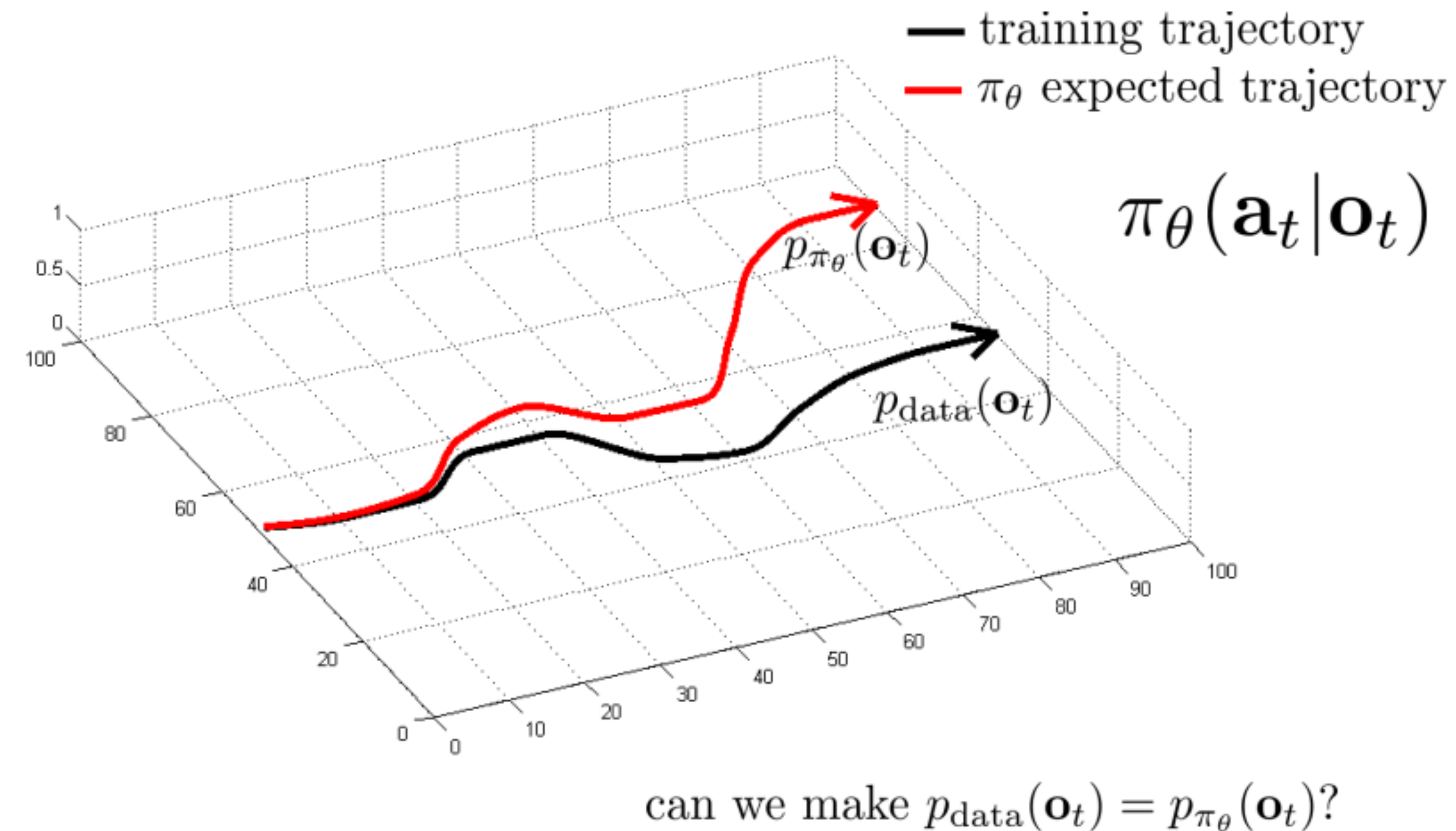
Why do we need RL for LMs?

- Maximising things that are hard to maximise with next-token loss (e.g., preference learning)



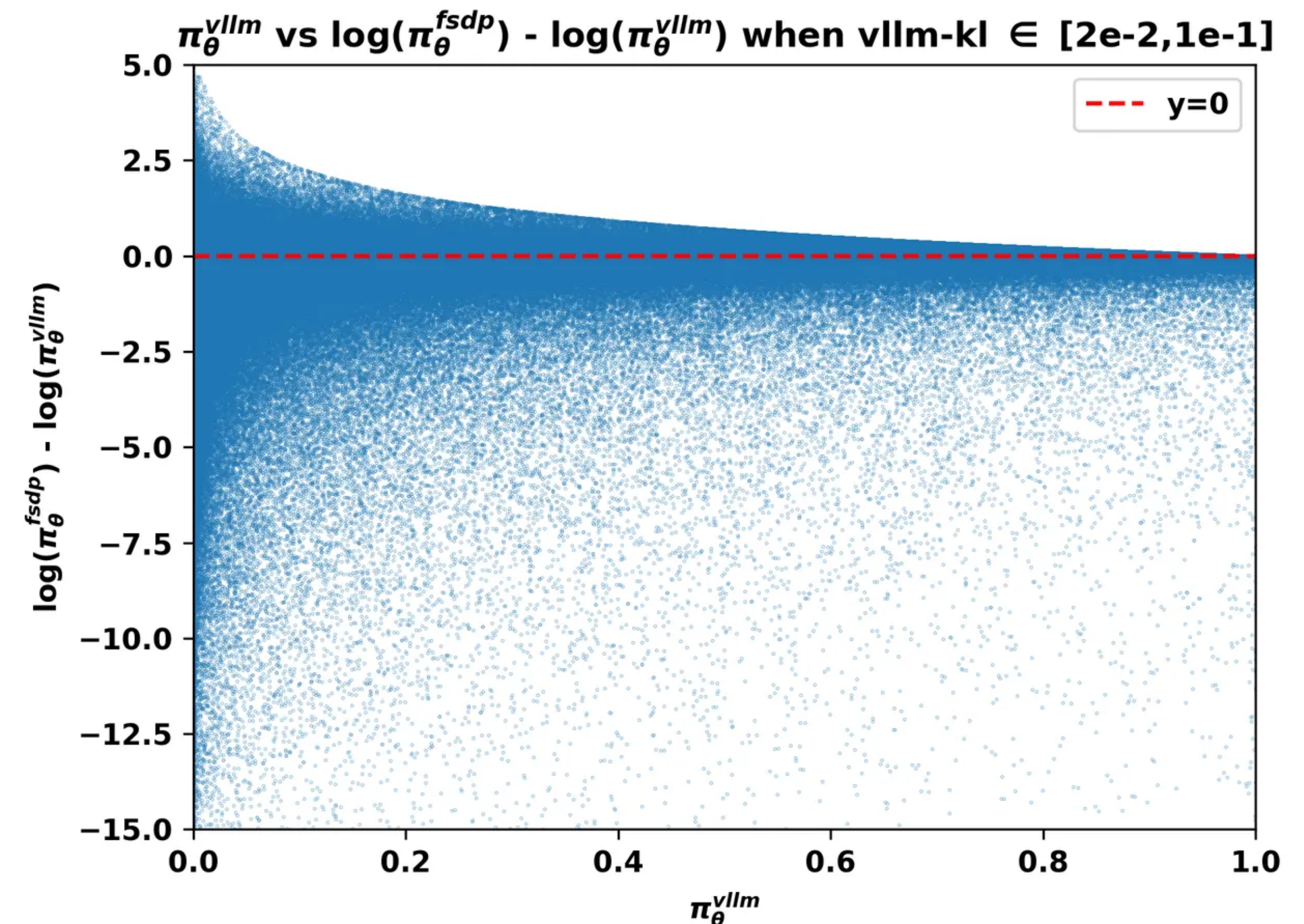
Why do we need RL for LMs?

- Maximising things that are hard to maximise with next-token loss.
- Train the model on distributions that better match its output (“distribution drift”).



Why do we need RL for LMs?

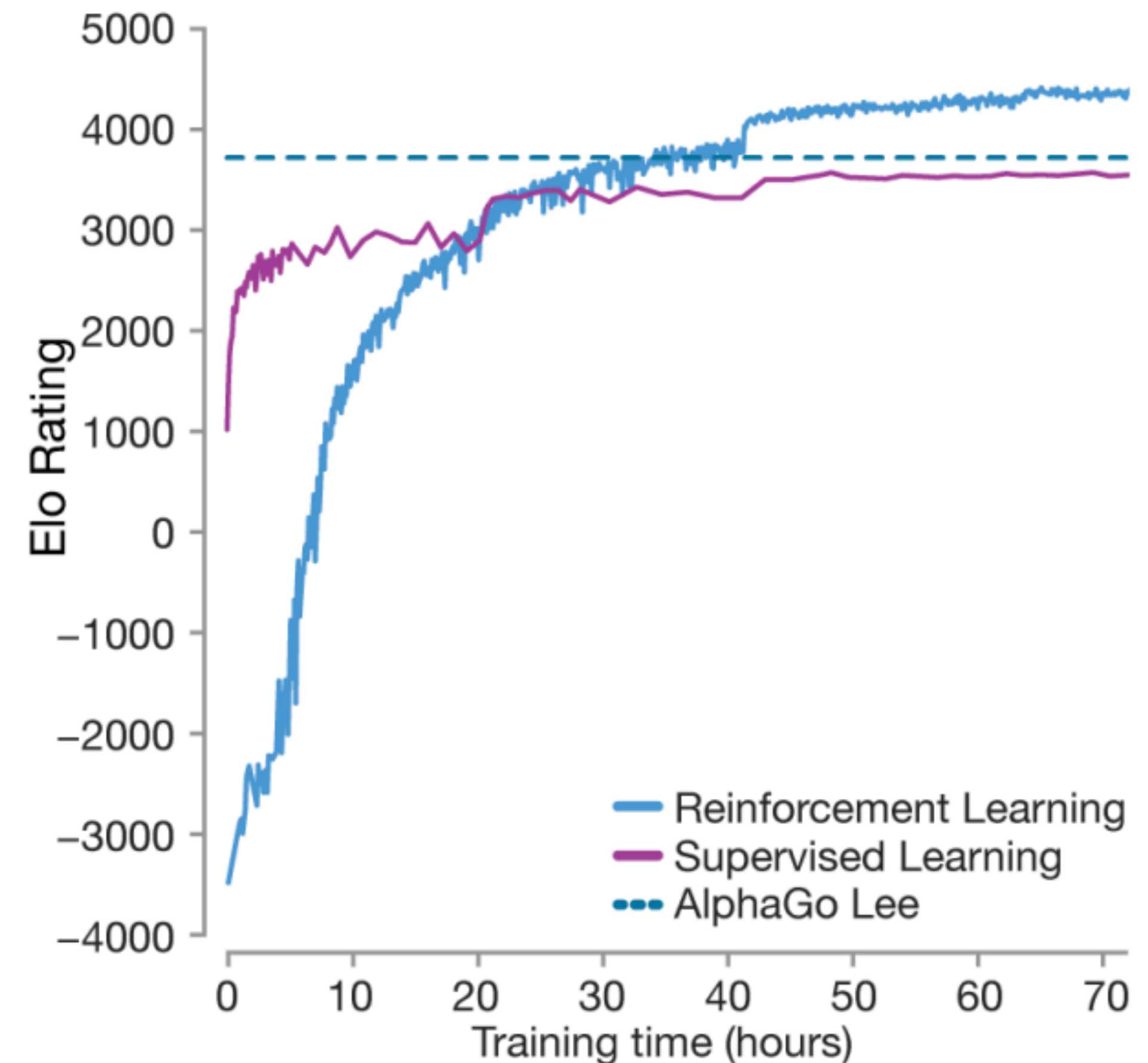
- Maximising things that are hard to maximise with next-token loss.
- Train the model on distributions that better match its output (“distribution drift”).
- Train in a “*real*” inference environment (vLLM $\neq$ HF).



Source: <https://yingru.notion.site/When-Speed-Kills-Stability-Demystifying-RL-Collapse-from-the-Training-Inference-Mismatch-271211a558b7808d8b12d403fd15edda>

Why do we need RL for LMs?

- Maximising things that are hard to maximise with next-token loss.
- Train the model on distributions that better match its output (“distribution drift”).
- Allows us to go beyond bound of teacher data.



Source: Silver, David et al. “Mastering the game of Go without human knowledge.” *Nature* 550 (2017): 354-359.

How do we do RL for LMs?

- Current approaches are all based on policy gradient.
 - *Why?*
 - Developed at OpenAI, OpenAI popularised RL for LMs.
 - Was the main ‘go-to’ approach in past few years.
 - Easier(?) to adapt to LMs: our LM is our policy, and we directly train that.
 - Policy gradient vs off-policy approaches (DQN, SAC) is an ongoing debate in RL-land.

How do we do RL for LMs?

- Current approaches are all based on *policy gradient*.
 - *Why?*
 - Developed at OpenAI, OpenAI popularised RL for LMs.
 - Was the main ‘go-to’ approach in past few years.
 - Easier(?) to adapt to LMs: our LM is our policy, and we directly train that.
 - Policy gradient vs off-policy approaches (DQN, SAC) is an ongoing debate in RL-land.

Interestingly and conversely, the *reliable* optimality gap indicates that **DQN is better than PPO** in Figure 6c, because DQN does achieve higher average policy performance, but the analysis from comparing to $V^{\hat{\pi}_D^*}(s_0)$, in Figure 6b shows us that even though DQN performs better than PPO, DQN is still generating a lot of high-value experience that it is not able to exploit. Conversely, because PPO is performing worse according to *reliable*, but has a better $V^{\hat{\pi}_D^*}(s_0) - V^{\hat{\pi}^\theta}(s_0)$, improved exploration **would improve PPO more than it would DQN**. Overall, these results suggest that both algorithms struggle to extract the most from their experience and that *reliable* is not telling the full story.

source: <https://www.arxiv.org/abs/2508.01329>

arXiv:2508.01329v1 [cs.LG] 2 Aug 2025

Is Exploration or Optimization the Problem for Deep Reinforcement Learning?

Glen Berseth
Université de Montréal, Mila - Quebec AI Institute, and CIFAR
glen.berseth@mila.quebec

Abstract

In the era of deep reinforcement learning, making progress is more complex, as the collected experience must be compressed into a deep model for future exploitation and sampling. Many papers have shown that training a deep learning policy under the changing state and action distribution leads to sub-optimal performance, or even collapse. This naturally leads to the concern that even if the community creates improved exploration algorithms or reward objectives, will those improvements fall on the *deaf ears* of optimization difficulties. This work proposes a new *practical* sub-optimality estimator to determine optimization limitations of deep reinforcement learning algorithms. Through experiments across environments and RL algorithms, it is shown that the difference between the best experience generated is 2-3× better than the policies’ learned performance. This large difference indicates that deep RL methods only exploit half of the good experience they generate.

1 Introduction

What is preventing deep reinforcement learning from solving harder tasks? Many papers have shown that training a deep learning policy under the changing state distribution (non-IID) leads to sub-optimal performance [Nikishin et al., 2022, Lyle et al., 2023, Dohare et al., 2024]. However, at a macro scale, it is not completely clear what causes these issues. Do the network and regularization changes from recent work improve exploration or exploitation, and which of these two issues is the larger concern to be addressed to advance deep RL algorithms? For example, better exploration algorithms can be created, but will the higher value experience fall on the *deaf ears* of the deep network optimization difficulties?

How can we understand if the limited deepRL performance is due to a lack of good exploration or deep network optimization (exploitation)? Normally in RL, to understand if there is a limitation, an oracle is needed to understand *sub-optimality*, how far the algorithm is from being optimal. However, that analysis is with respect to the best policy and aliases both causes of the limitations of either exploration or optimization. Instead, consider the example where a person is learning how to build good houses. There are two issues that may prevent the person from *consistently* building a high quality house: (1) they can’t *explore* well enough to discover a good design or (2) they can explore well enough to find good designs, but they can’t properly *exploit* their experience to replicate those good experience. For deep RL algorithms, which of these two issues is more prevalent?

To understand if exploration or exploitation is the larger culprit, a method is needed to estimate the *practical sub-optimality* between these cases. This estimator should (1) measure the agent’s ability to explore, (2) while also estimating the average performance for the learning policy π^θ . While estimating the average policy performance is common, estimating the exploration ability for a policy is not. Extending the house-building metaphor, the idea is to estimate how close the agent ever got to constructing a good home. Therefore, to realize this estimator, we propose computing the *exploration* value for a policy that is calculated over prior experience, called the *experience optimal policy*. Using

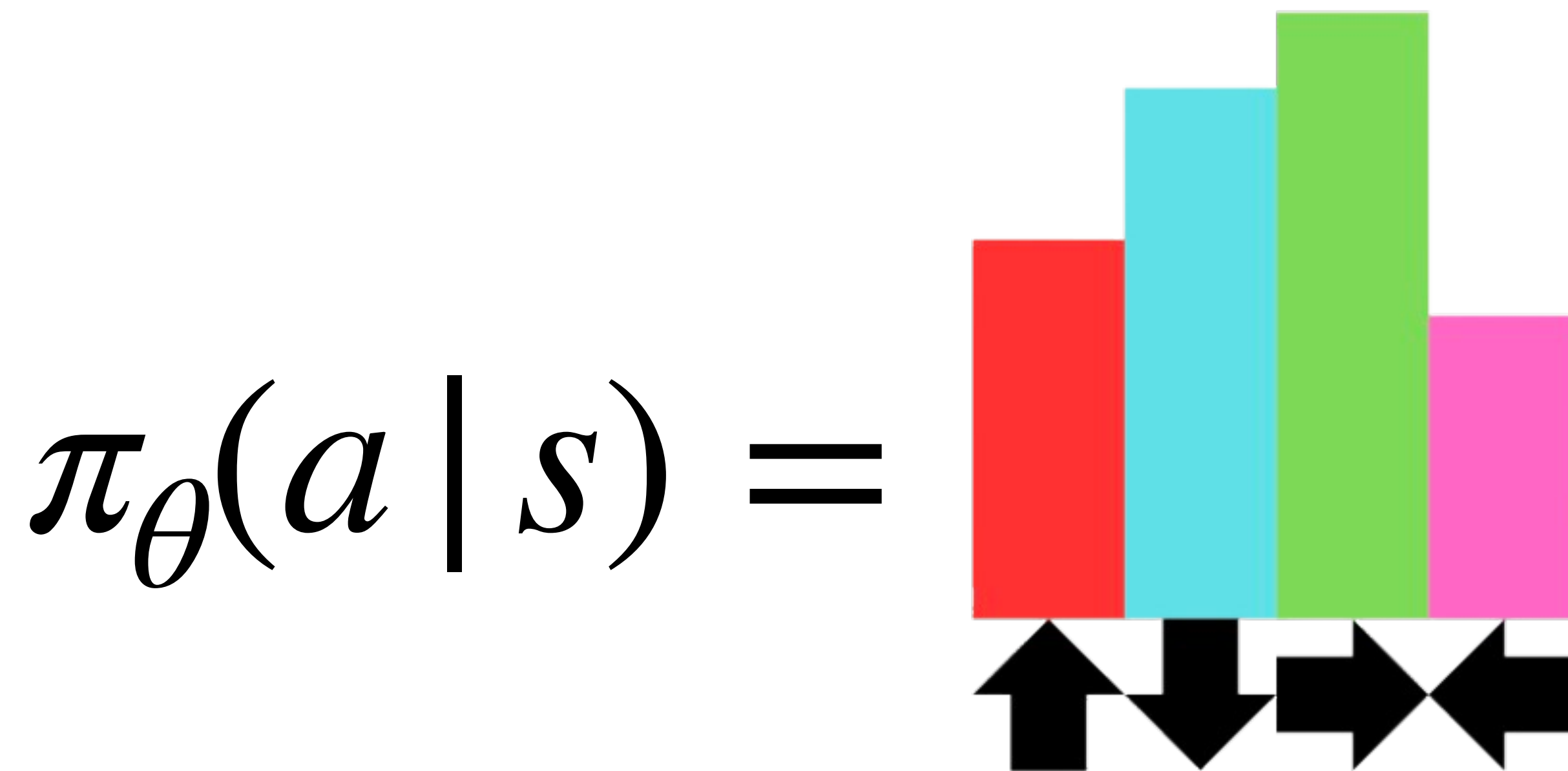
Preprint. Under review.

Policy Gradient

A crash course on policy gradient

For this, we consider a *parameterised policy*: a policy that can directly select what action to take, without necessarily consulting value (or q-) functions.

Effectively, we learn to produce a probability distribution over actions given a state.



π_{θ} can be anything we can take gradients on! ...like a LM!

A crash course on policy gradient

We can then aim to maximise performance by just taking gradient steps:

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$$

A crash course on policy gradient

We can then aim to maximise performance by just taking gradient steps:

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$$

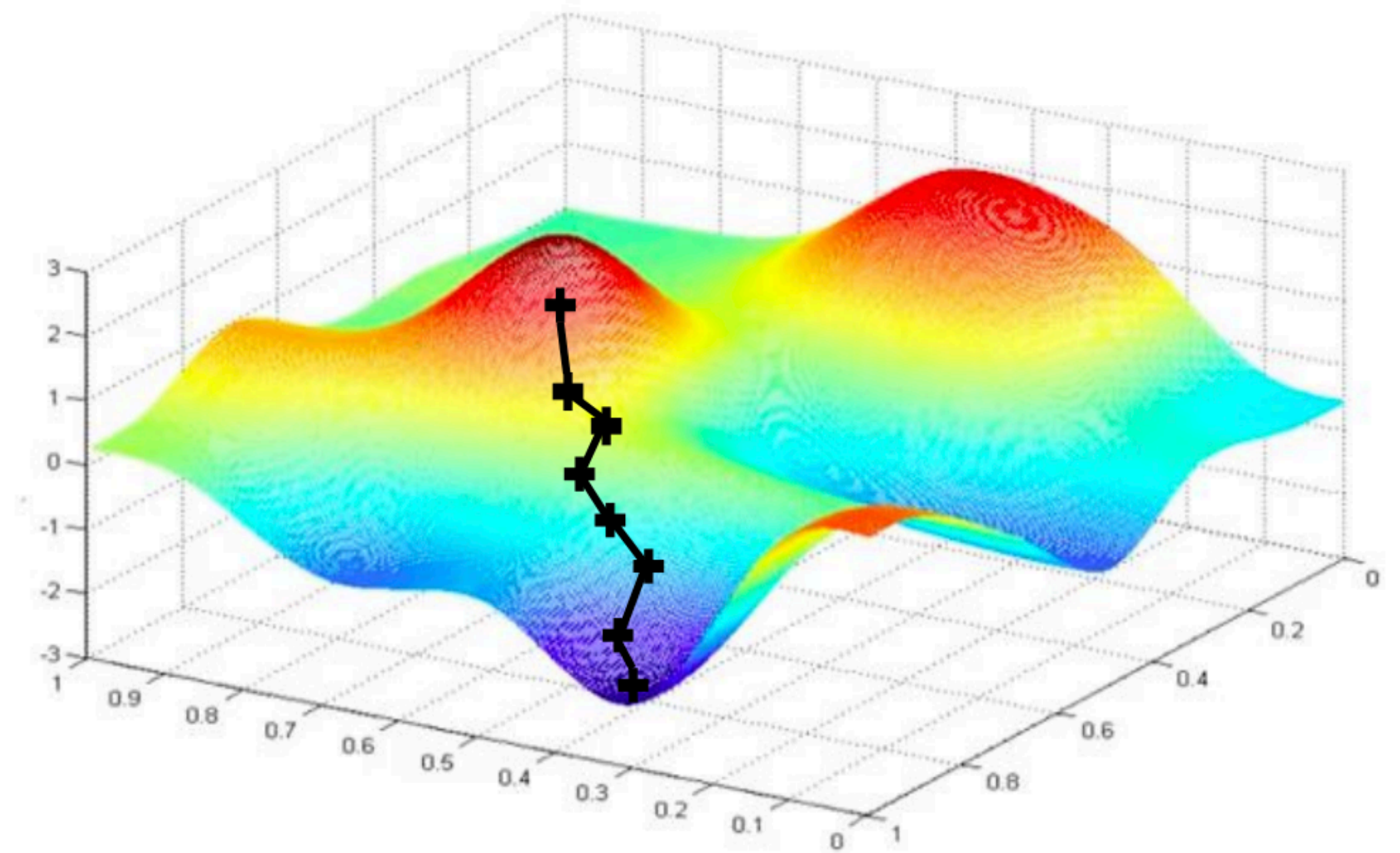
More basic details on gradient steps in a bit!

Basically just: if we take these steps, we will lower the loss associated with $J(\theta)$

Stochastic Gradient Descent

Algorithm

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Pick single data point i
4. Compute gradient, $\frac{\partial J_i(\theta)}{\partial \theta}$
5. Update weights, $\theta \leftarrow \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$
6. Return weights



A crash course on policy gradient

We can then aim to maximise performance by just taking gradient steps:

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$$

$J(\theta)$ here is our objective function. We will set it as the ‘true’ value of the initial state under our policy:

$$J(\theta) = v_{\pi_{\theta}}(s_0)$$

A crash course on policy gradient

We can then aim to maximise performance by just taking gradient steps:

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$$

$J(\theta)$ here is our objective function. We will set it as the ‘true’ value of the initial state under our policy:

$$J(\theta) = v_{\pi_\theta}(s_0)$$

v_{π_θ} is the value function for our parameterised policy, and s_0 is the initial state.

We could do simple gradient descent, but in practice we can plug into Adam, etc.

A crash course on policy gradient

With some math (see *Sutton & Barto*, Chap. 13), we can show:

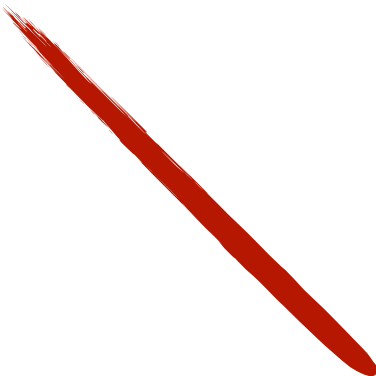
$$\nabla J(\theta) \propto \mathbb{E}_{\pi} \left[G_t \nabla_{\theta} \ln(\pi(A_t | S_t, \theta)) \right]$$



Expectation
under our
policy



Expected
return
(reward)



probability
over next
action

A crash course on policy gradient

With some math (see *Sutton & Barto*, Chap. 13), we can show:

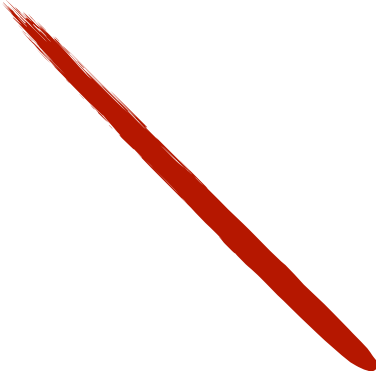
$$\nabla J(\theta) \propto \mathbb{E}_{\pi} \left[G_t \nabla_{\theta} \ln(\pi(A_t | S_t, \theta)) \right]$$



Expectation
under our
policy



Expected
return
(reward)



probability
over next
action

This is exactly the REINFORCE update!

REINFORCE: The full algorithm

Basically:

```
while True:
```

```
    Generate a full episode (“rollout”) starting from  $s_0$  using  $\pi_\theta$ 
```

```
    Compute reward over the rollouts
```

```
    Compute the REINFORCE loss and apply gradient
```

How is this instantiated for LMs?

$$\mathbb{E}_{\pi} \left[G_t \nabla_{\theta} \ln(\pi(A_t \mid S_t, \theta)) \right]$$

$$G_t = R(S_t, A_t)$$

We have some reward function

$$S_t = x$$

Our state is just the text seen so far

$$A_t = y_t$$

Our action is producing a token.
Generating a full answer = episode.

$$\mathbb{E}_{\pi_{\theta}} [R(x, y) \nabla_{\theta} \ln \pi_{\theta}(y \mid x)]$$

How is this instantiated for LMs?

In practice, we draw prompts (x) from a dataset (D), and sample completions (y) from our model, and do this in batches:

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} R(x, y_i) \nabla_{\theta} \ln \pi_{\theta}(y_i | x) \right]$$

How is this instantiated for LMs?

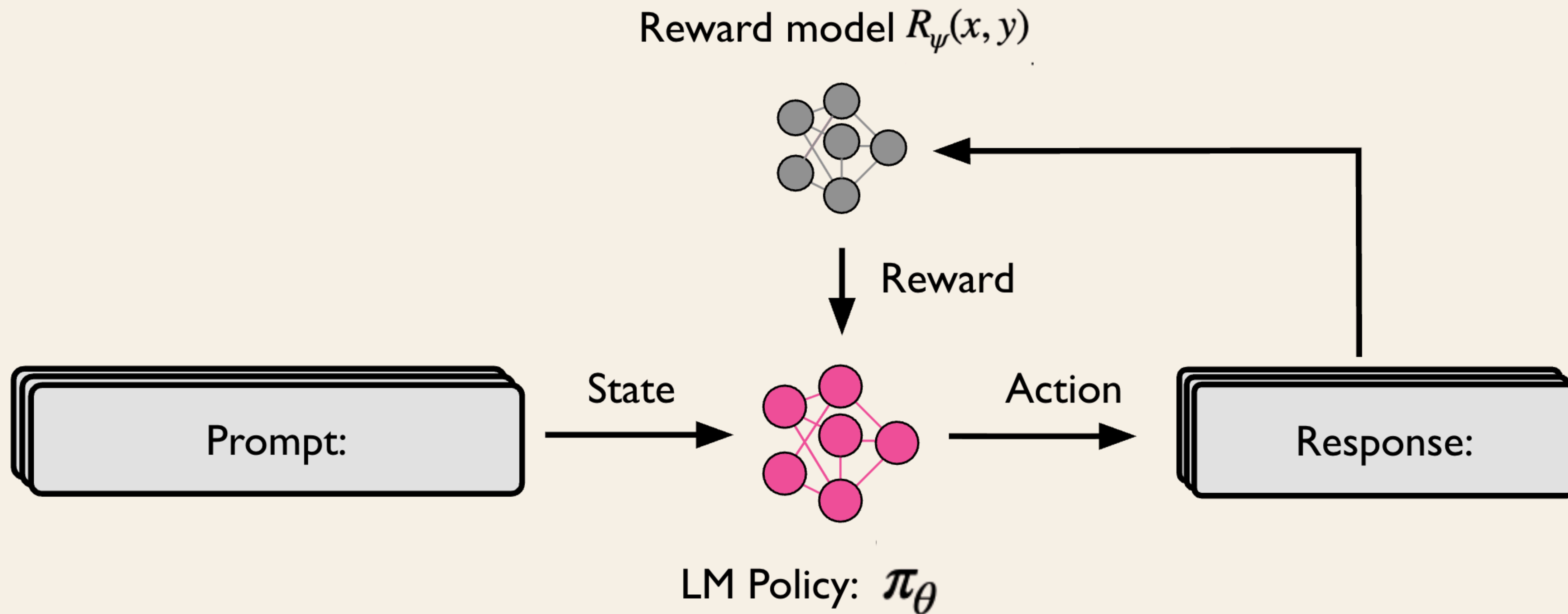
In practice, we draw prompts (x) from a dataset (D), and sample completions (y) from our model, and do this in batches:

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} R(x, y_i) \nabla_{\theta} \ln \pi_{\theta}(y_i | x) \right]$$

The diagram illustrates the instantiation of the equation for Large Models (LMs) using red arrows pointing from descriptive labels to specific components of the formula:

- over prompts**: Points to the expectation operator $\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)}$.
- over responses**: Points to the inner sum over y_i .
- over batch**: Points to the outer sum over B .
- reward**: Points to the term $R(x, y_i)$.
- logprobs of response**: Points to the term $\ln \pi_{\theta}(y_i | x)$.

How is this instantiated for LMs?



Going beyond REINFORCE

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} R(x, y_i) \nabla_{\theta} \ln \pi_{\theta}(y_i | x) \right]$$

Issues:

- High variance
- Reward scale can cause issues... (consider neg. vs pos. reward)
- We can only use samples directly from our current loop

Baselining & Advantage

Reduce variance: baselining

Core idea: try to normalise our reward as much as possible.

$$(R(y, x) - b) \nabla_{\theta} \ln \pi_{\theta}(y | x)$$

Reduce variance: baselining

Core idea: try to normalise our reward as much as possible.

$$(R(y, x) - b) \nabla_{\theta} \ln \pi_{\theta}(y | x)$$

Most common choice for baseline: **value estimate**

$$(R(y, x) - \hat{V}_{\theta'}(x, y_{<t})) \nabla_{\theta} \ln \pi_{\theta}(y | x)$$

Reduce variance: baselining

Core idea: try to normalise our reward as much as possible.

$$(R(y, x) - b) \nabla_{\theta} \ln \pi_{\theta}(y | x)$$

Most common choice for baseline: **value estimate**

$$(R(y, x) - \hat{V}_{\theta'}(x, y_{<t})) \nabla_{\theta} \ln \pi_{\theta}(y | x)$$

We call this form of normalising **the advantage function**:

$$A_{\theta'}(x, y) = R(x, y_t) - \hat{V}_{\theta'}(x, y_{<t})$$

Reduce variance: baselining

Core idea: try to normalise our reward as much as possible.

$$(R(y, x) - b) \nabla_{\theta} \ln \pi_{\theta}(y | x)$$

Most common choice for baseline: **value estimate**

$$(R(y, x) - \hat{V}_{\theta'}(x, y_{<t})) \nabla_{\theta} \ln \pi_{\theta}(y | x)$$

We call this form of normalising the advantage function:

$$A_{\theta'}(x, y) = R(x, y_t) - \hat{V}_{\theta'}(x, y_{<t})$$

lowest poss.
variance if we
knew V exactly!

Other forms of estimating advantage

$$A_{\theta'}(x, y) = R(x, y_t) - \hat{V}_{\theta'}(x, y_{<t})$$

PPO: Use Generalised Advantage Estimation (GAE)

HIGH-DIMENSIONAL CONTINUOUS CONTROL USING
GENERALIZED ADVANTAGE ESTIMATION

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan and Pieter Abbeel
Department of Electrical Engineering and Computer Science
University of California, Berkeley
`{joschu,pcmoritz,levine,jordan,pabbeel}@eecs.berkeley.edu`

Other forms of estimating advantage

$$A_{\theta'}(x, y) = R(x, y_t) - \hat{V}_{\theta'}(x, y_{<t})$$

PPO: Use Generalised Advantage Estimation (GAE)

HIGH-DIMENSIONAL CONTINUOUS CONTROL USING
GENERALIZED ADVANTAGE ESTIMATION

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan and Pieter Abbeel
Department of Electrical Engineering and Computer Science
University of California, Berkeley
{joschu, pcmoritz, levine, jordan, pabbeel}@eecs.berkeley.edu

$$\delta_t = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$$

TD residual - “how much better than expected will we do taking a particular action”
we use a learnt value function

TD Residual?

$$\delta_t = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$$

TD residual - “how much better than expected will we do taking a particular action”

This should be familiar...

TD Residual?

The Bellman Equations

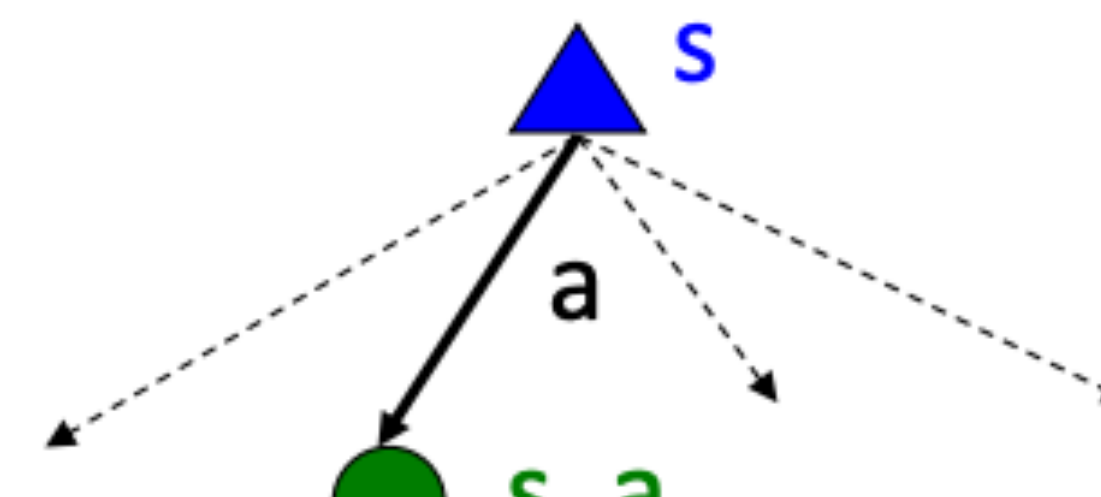
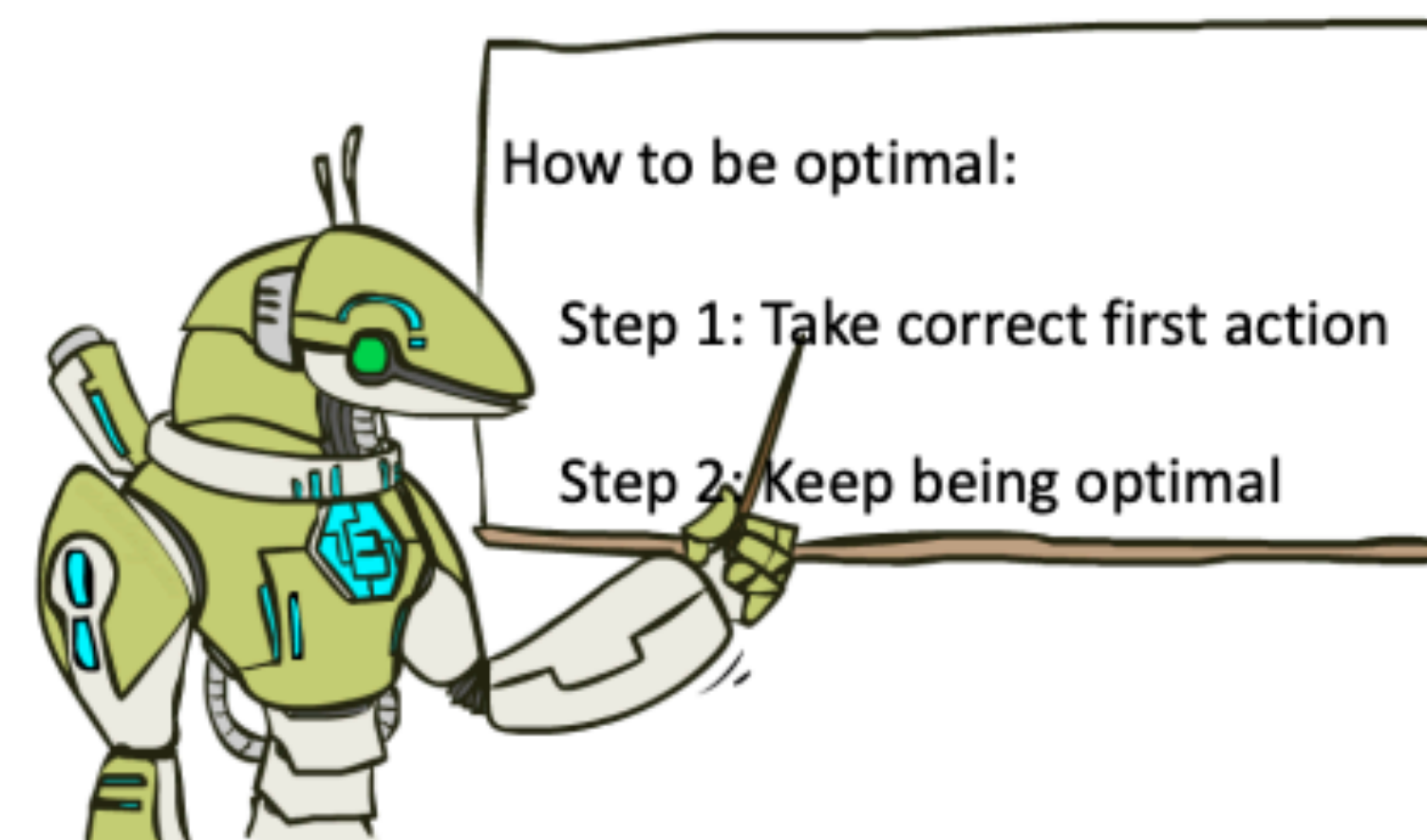
- Definition of “optimal utility” via expectimax recurrence gives a simple one-step lookahead relationship amongst optimal utility values

$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

$$V^*(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$



Other forms of estimating advantage

$$A_{\theta'}(x, y) = R(x, y_t) - \hat{V}_{\theta'}(x, y_{<t})$$

PPO: Use Generalised Advantage Estimation (GAE)

HIGH-DIMENSIONAL CONTINUOUS CONTROL USING
GENERALIZED ADVANTAGE ESTIMATION

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan and Pieter Abbeel
Department of Electrical Engineering and Computer Science
University of California, Berkeley
{joschu, pcmoritz, levine, jordan, pabbeel}@eecs.berkeley.edu

$$\delta_t = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$$

TD residual - “how much better than expected did we do?”
we use a learnt value function

Other forms of estimating advantage

$$A_{\theta'}(x, y) = R(x, y_t) - \hat{V}_{\theta'}(x, y_{<t})$$

PPO: Use Generalised Advantage Estimation (GAE)

HIGH-DIMENSIONAL CONTINUOUS CONTROL USING
GENERALIZED ADVANTAGE ESTIMATION

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan and Pieter Abbeel
Department of Electrical Engineering and Computer Science
University of California, Berkeley
{joschu,pcmoritz,levine,jordan,pabbeel}@eecs.berkeley.edu

$$\delta_t = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$$

TD residual - “how much better than expected did we do?”
we use a learnt value function

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}^V$$

note that γ is usually set to 1, λ to 0.95 for RL w/ LMs!

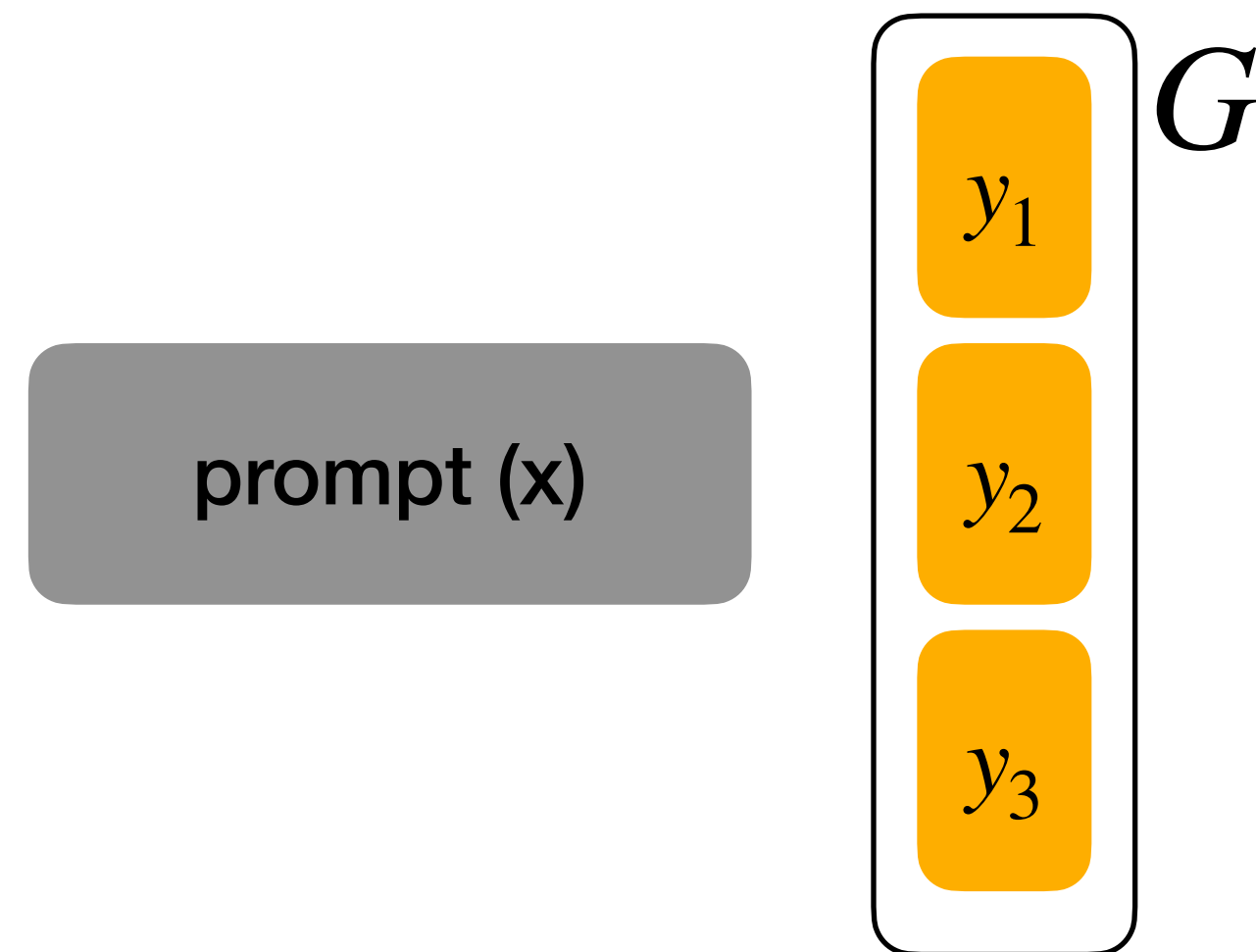
We also typically normalise the advantages within the batch (whitening)

How to estimate advantage?

GRPO (Shao et al 2024): Use Group Estimates

How to estimate advantage?

GRPO (Shao et al 2024): Use Group Estimates



$$A_j = \frac{R(x, y_j) - \text{mean}(R_G)}{\text{std}(R_G)}$$

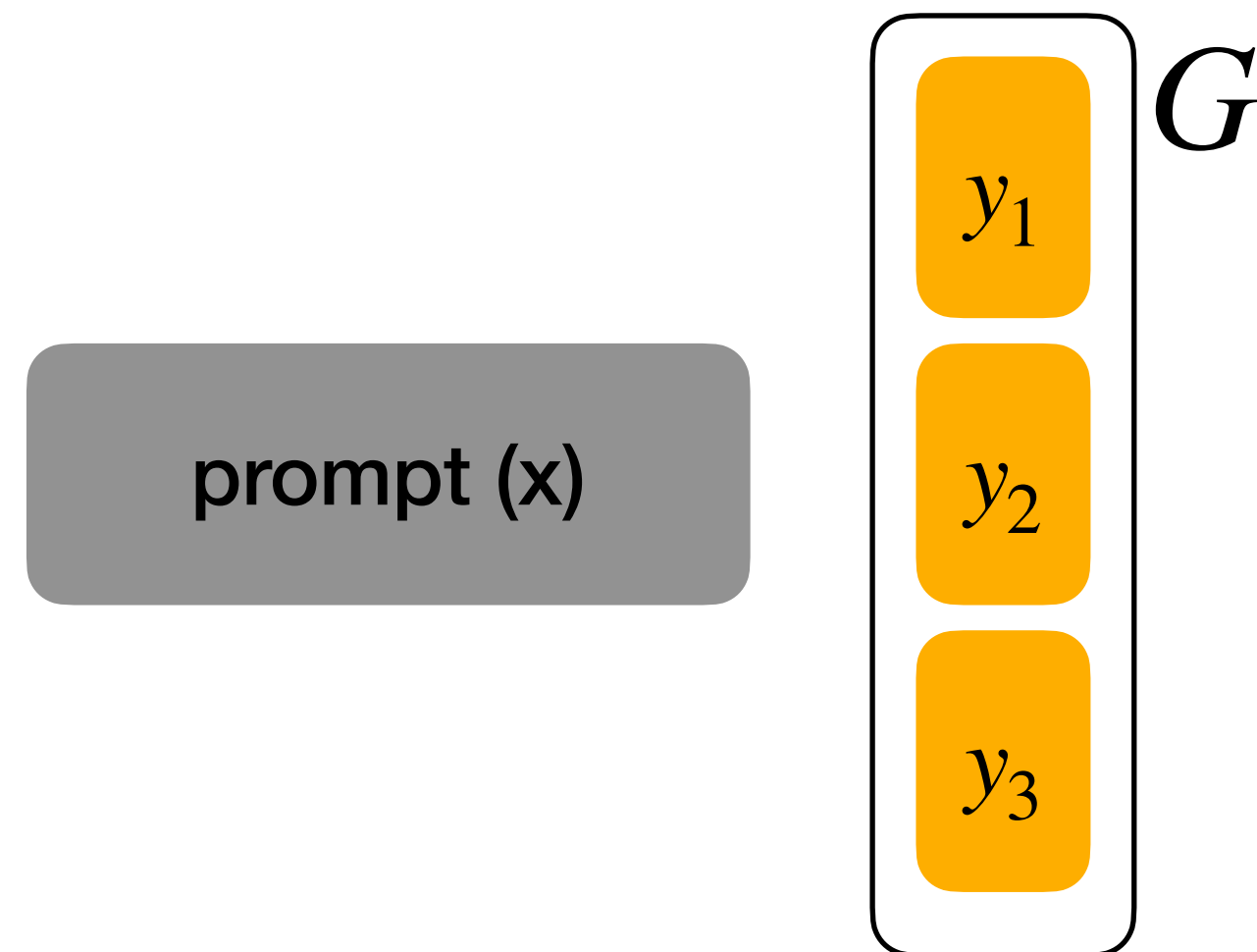
No value estimation needed! Just take multiple samples per prompt and use average improvement over the group as advantage.

How to estimate advantage?

RLOO (Ahmadian et al 2024): Use leave-one-out estimate

How to estimate advantage?

RLOO (Ahmadian et al 2024): Use leave-one-out estimate



$$A_j = R(y_j, x) - \frac{1}{G-1} \sum_{i \neq j} R(y_{(i)}, x)$$

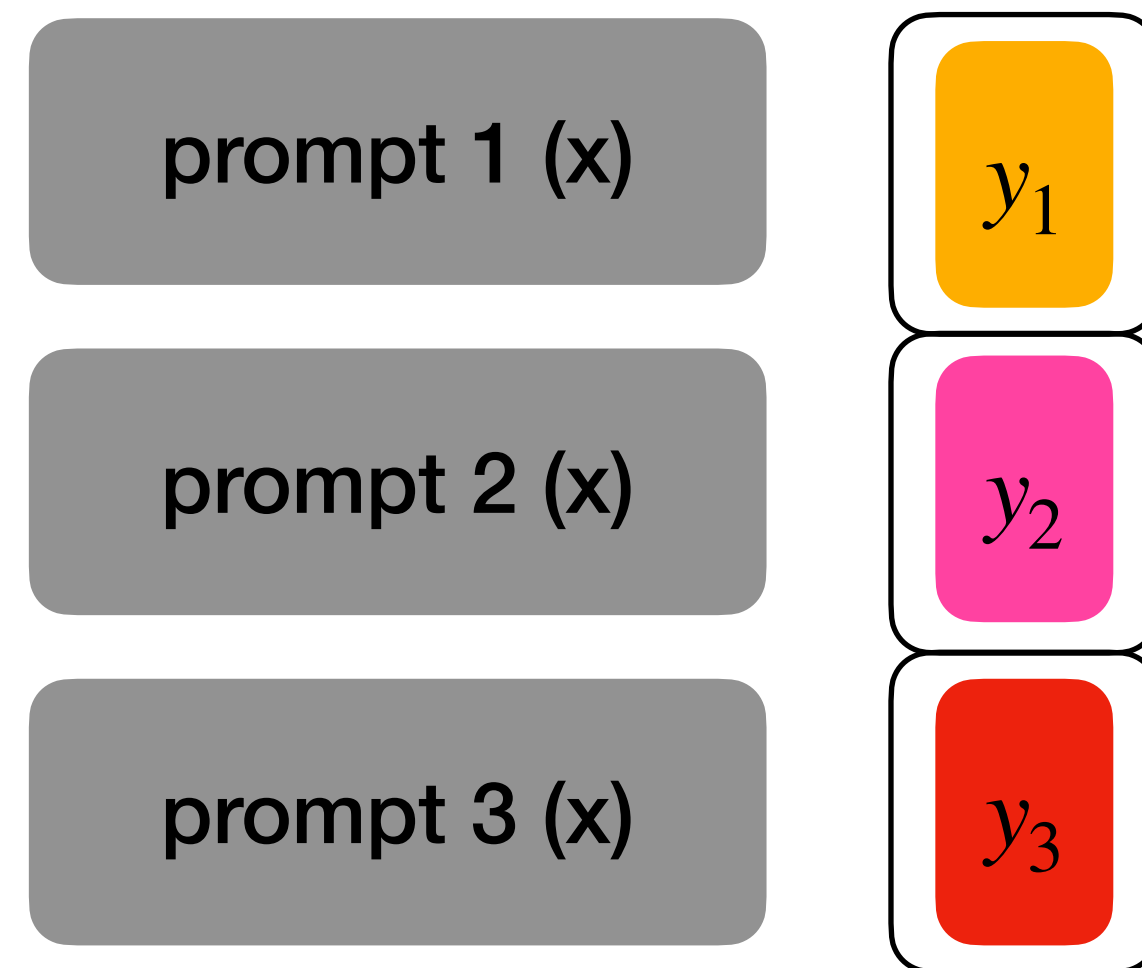
Use average of other samples from the same prompt.

How to estimate advantage?

REINFORCE++ (Hu et al. 2025): Use batch estimate (no groups!)

How to estimate advantage?

REINFORCE++ (Hu et al. 2025): Use batch estimate (no groups!)



$$A_j = \frac{R(x, y_j) - \text{mean}(R)}{\text{std}(R)}$$

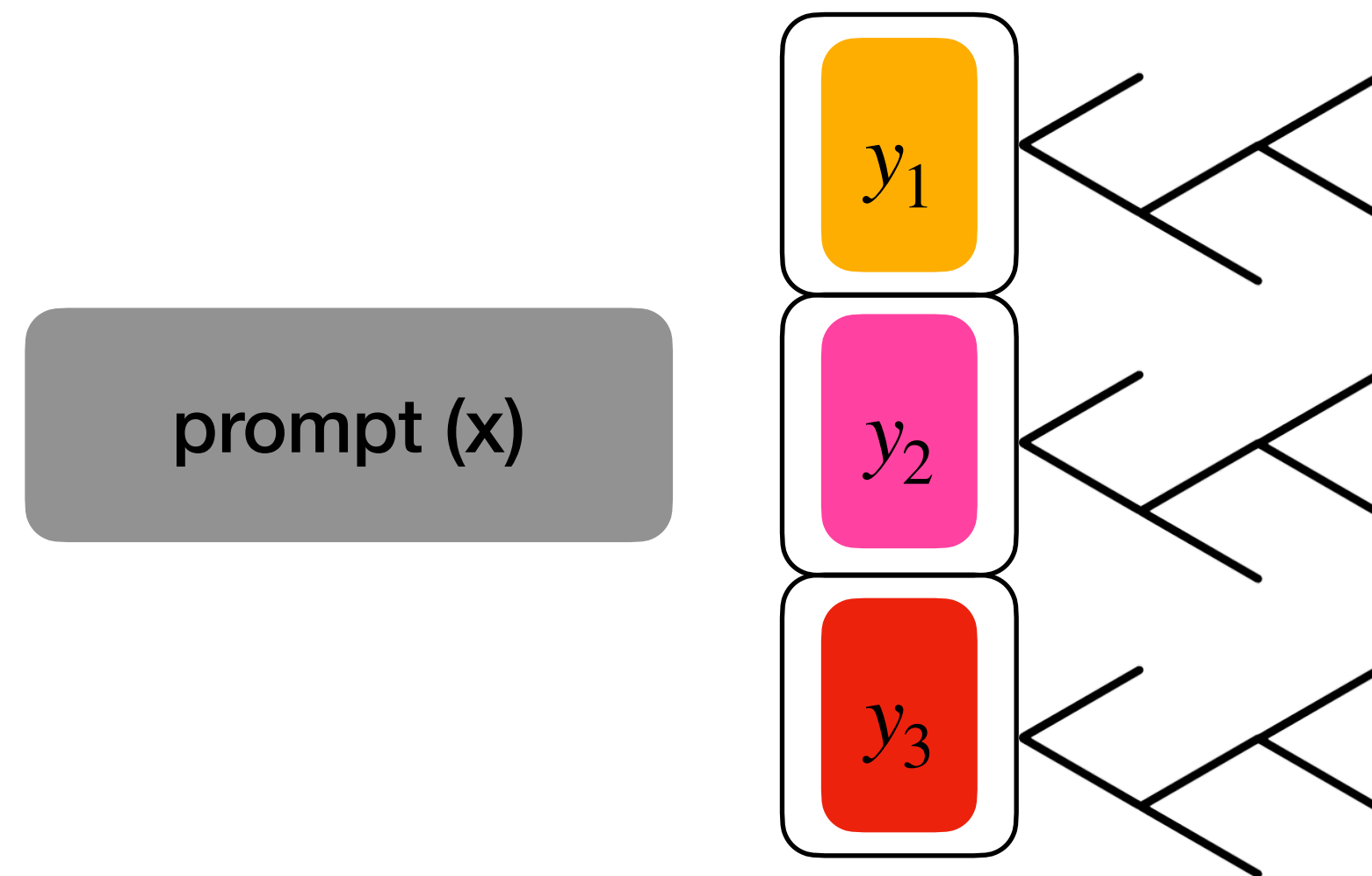
Just normalise across everything in batch. Commonly done anyway (“whitening”).

How to estimate advantage?

VinePPO (Kazemnejad et al 2024): Use Monte Carlo Estimates

How to estimate advantage?

VinePPO (Kazemnejad et al 2024): Use Monte Carlo Estimates

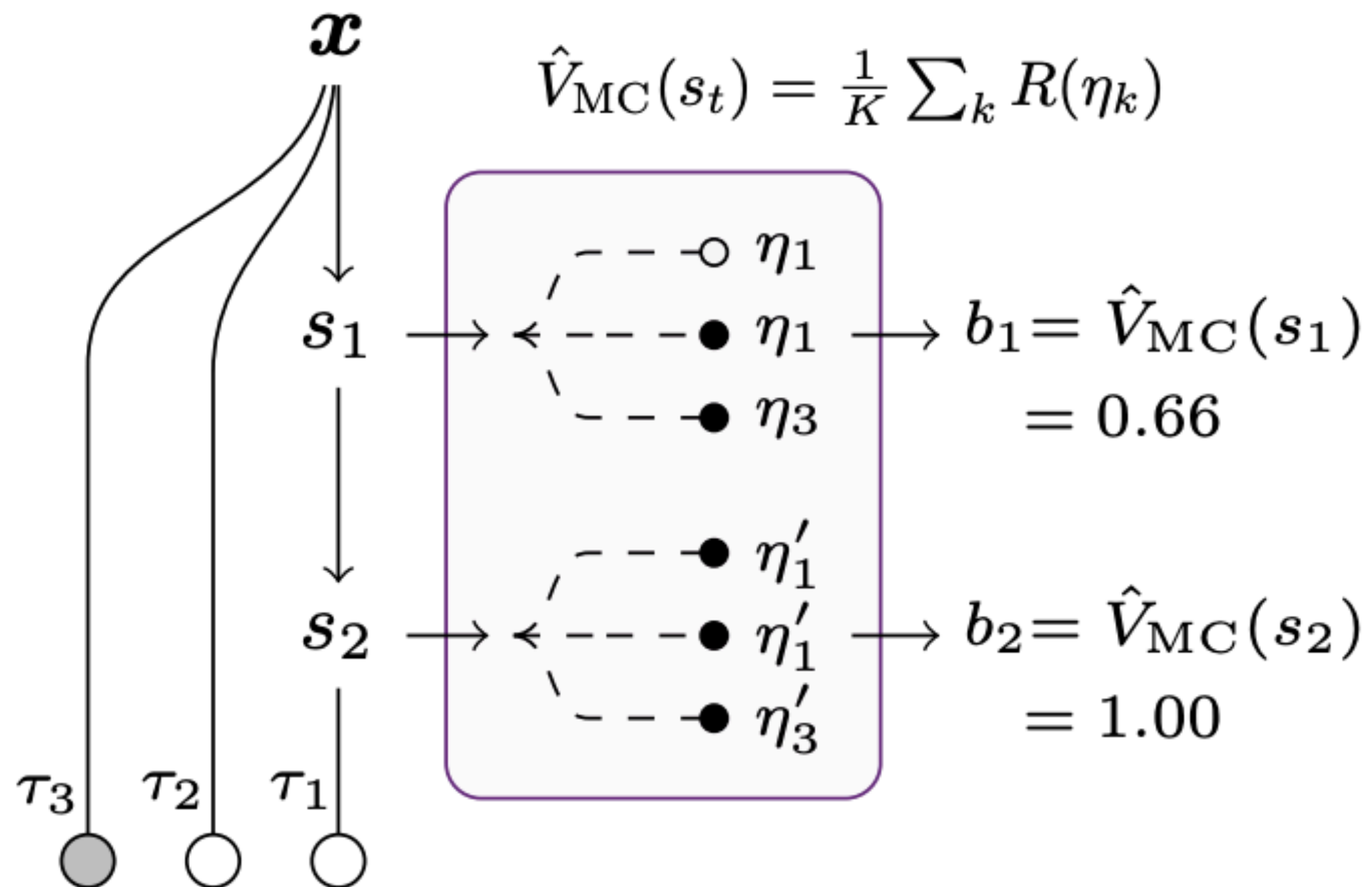


No value estimation
needed! Just take
multiple samples *per*
~token and use those!

How to estimate advantage?

VinePPO (Kazemnejad et al 2024): Use Monte Carlo Estimates

VinePPO

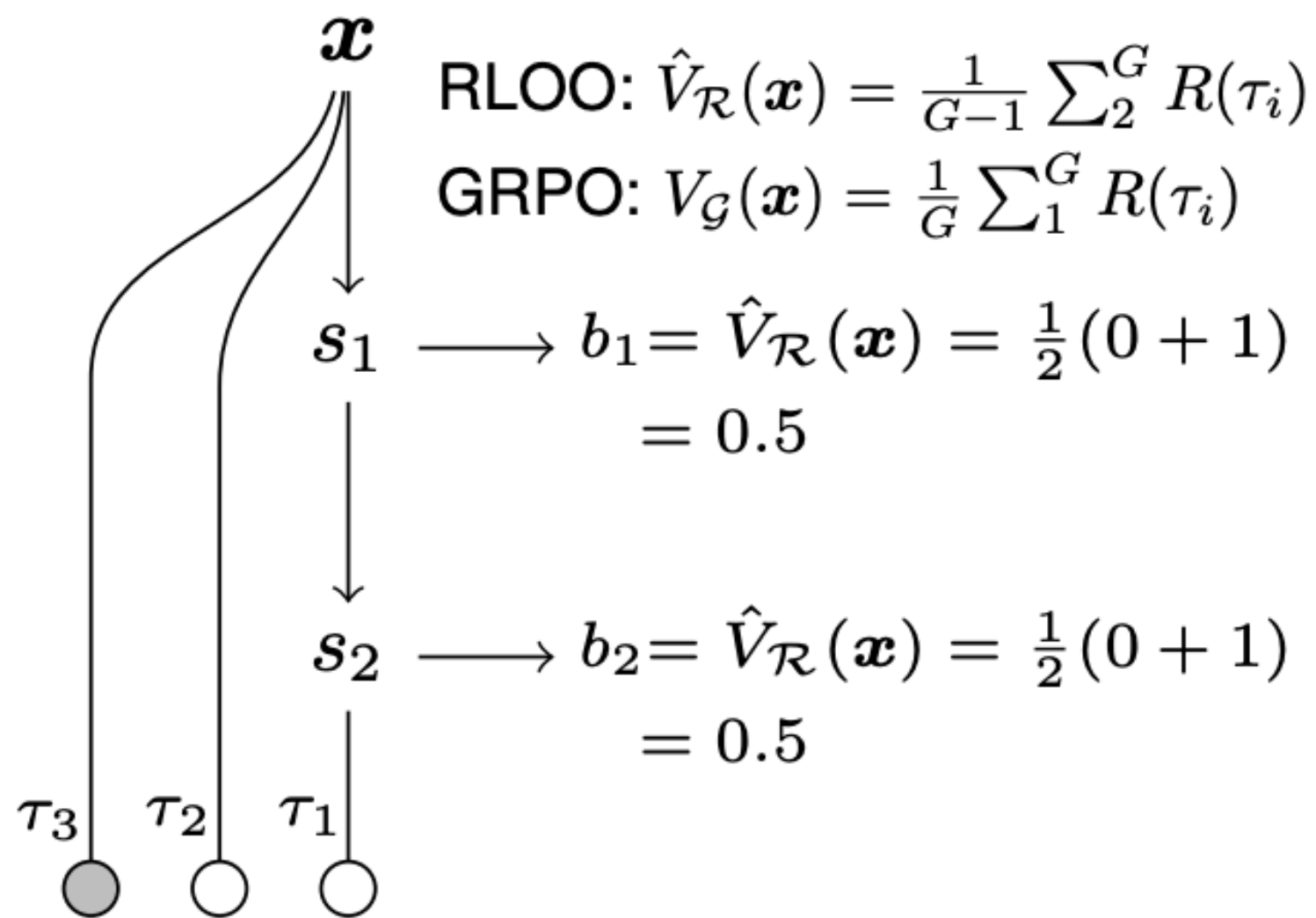


No value estimation needed! Just take multiple samples *per token* and use those!

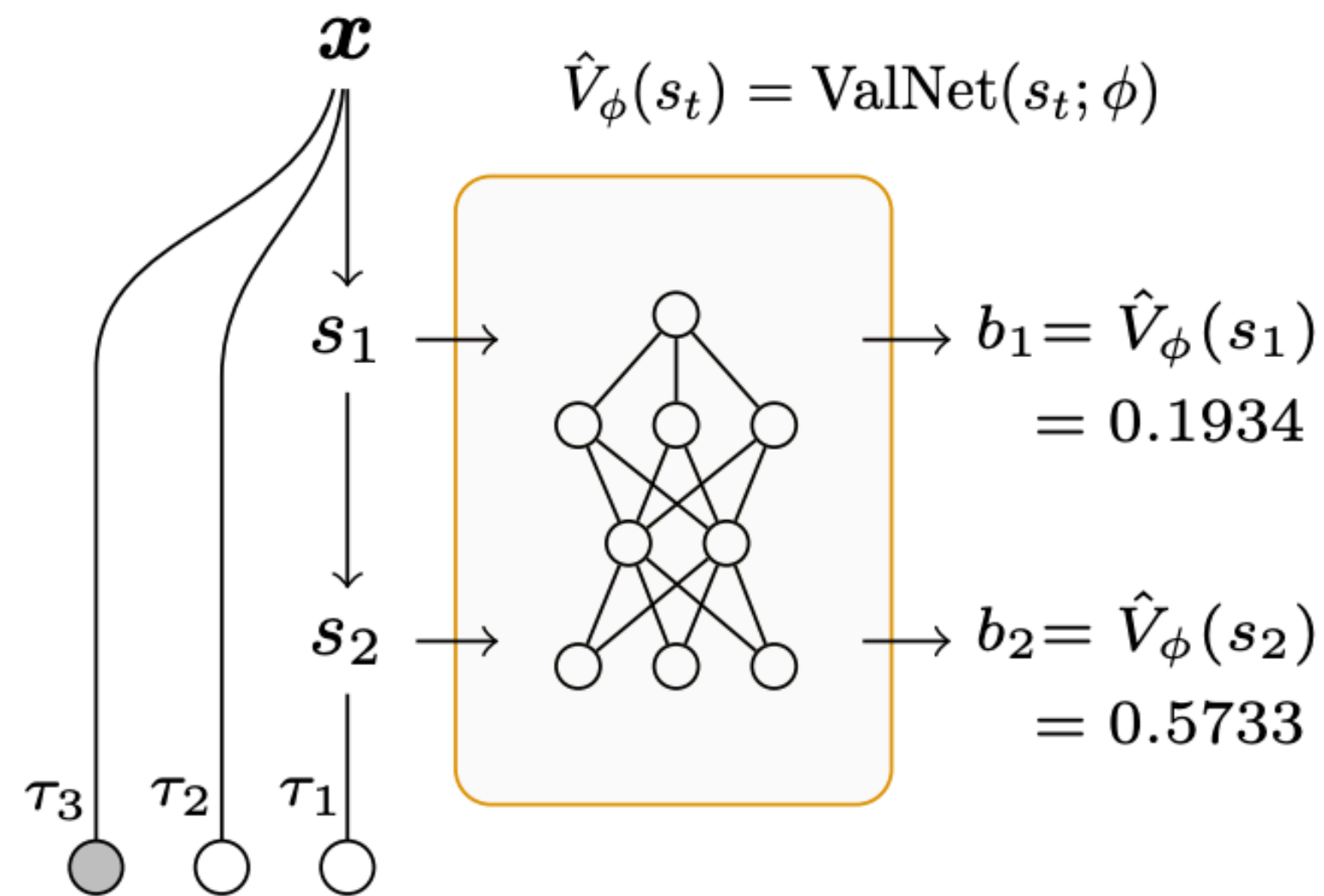
A summary of advantage estimates

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)}[(A(y, x) \nabla_{\theta} \log \pi_{\theta}(y | x)]$$

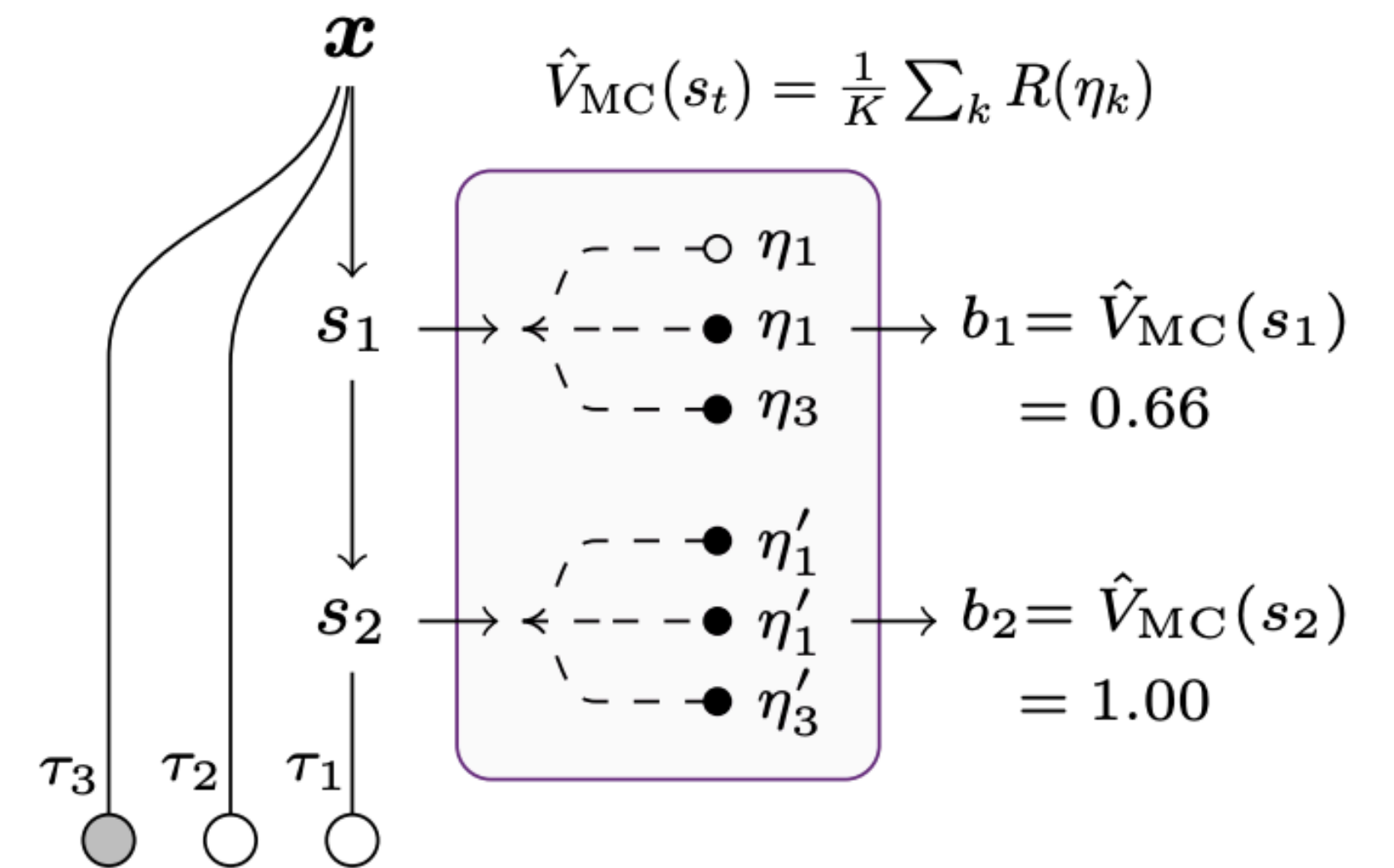
RLOO / GRPO



PPO



VinePPO



Going beyond REINFORCE

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} R(x, y_i) \nabla_{\theta} \ln \pi_{\theta}(y_i | x) \right]$$

Issues:

- High variance
- Reward scale can cause issues... (consider neg. vs pos. reward)
- We can only use samples directly from our current loop

Going beyond REINFORCE

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} R(x, y_i) \nabla_{\theta} \ln \pi_{\theta}(y_i | x) \right]$$

Issues:

- High variance 
- Reward scale can cause issues... (consider neg. vs pos. reward) 
- We can only use samples directly from our current loop

Going beyond REINFORCE

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} R(x, y_i) \nabla_{\theta} \ln \pi_{\theta}(y_i | x) \right]$$

Issues:

- High variance 
- Reward scale can cause issues... (consider neg. vs pos. reward) 
- We can only use samples directly from our current loop

**Importance Sampling & Off-
Policy data**

Revisiting REINFORCE

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} R(x, y_i) \nabla_{\theta} \ln \pi_{\theta}(y_i | x) \right]$$

Cannot reuse old data! -> can we get more efficient?

What if we re-used our old data anyway?

Well, our problem is we are doing our loss assuming **sampling from our current policy...**

Train on “old” samples w/ IS

We can reuse old samples with importance sampling!

$$\mathbb{E}_{\tau \sim p_{\theta}}[f(\tau)] = \mathbb{E}_{\tau \sim p_{\beta}} \left[\frac{p_{\theta}(\tau)}{p_{\beta}(\tau)} f(\tau) \right]$$

Train on “old” samples w/ IS

We can reuse old samples with importance sampling!

$$\mathbb{E}_{\tau \sim p_{\theta}}[f(\tau)] = \mathbb{E}_{\tau \sim p_{\beta}} \left[\frac{p_{\theta}(\tau)}{p_{\beta}(\tau)} f(\tau) \right]$$

some function on
trajectories over
policy

Train on “old” samples w/ IS

We can reuse old samples with importance sampling!

$$\mathbb{E}_{\tau \sim p_{\theta}}[f(\tau)] = \mathbb{E}_{\tau \sim p_{\beta}} \left[\frac{p_{\theta}(\tau)}{p_{\beta}(\tau)} f(\tau) \right]$$

some function on
trajectories over
policy

Same function, but
now we drawing
trajectories from diff
policy, and reweight
accordingly!

Applying IS to REINFORCE

We can apply this directly to reinforce! Our distribution is just the older policy.

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \dots \ln \pi_{\theta}(y_i | x)$$

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta_{\text{old}}}(\cdot | x)} \dots \frac{\ln \pi_{\theta}(y_i | x)}{\ln \pi_{\theta_{\text{old}}}(y_i | x)}$$

Applying IS to REINFORCE

We can apply this directly to reinforce! Our distribution is just the older policy.

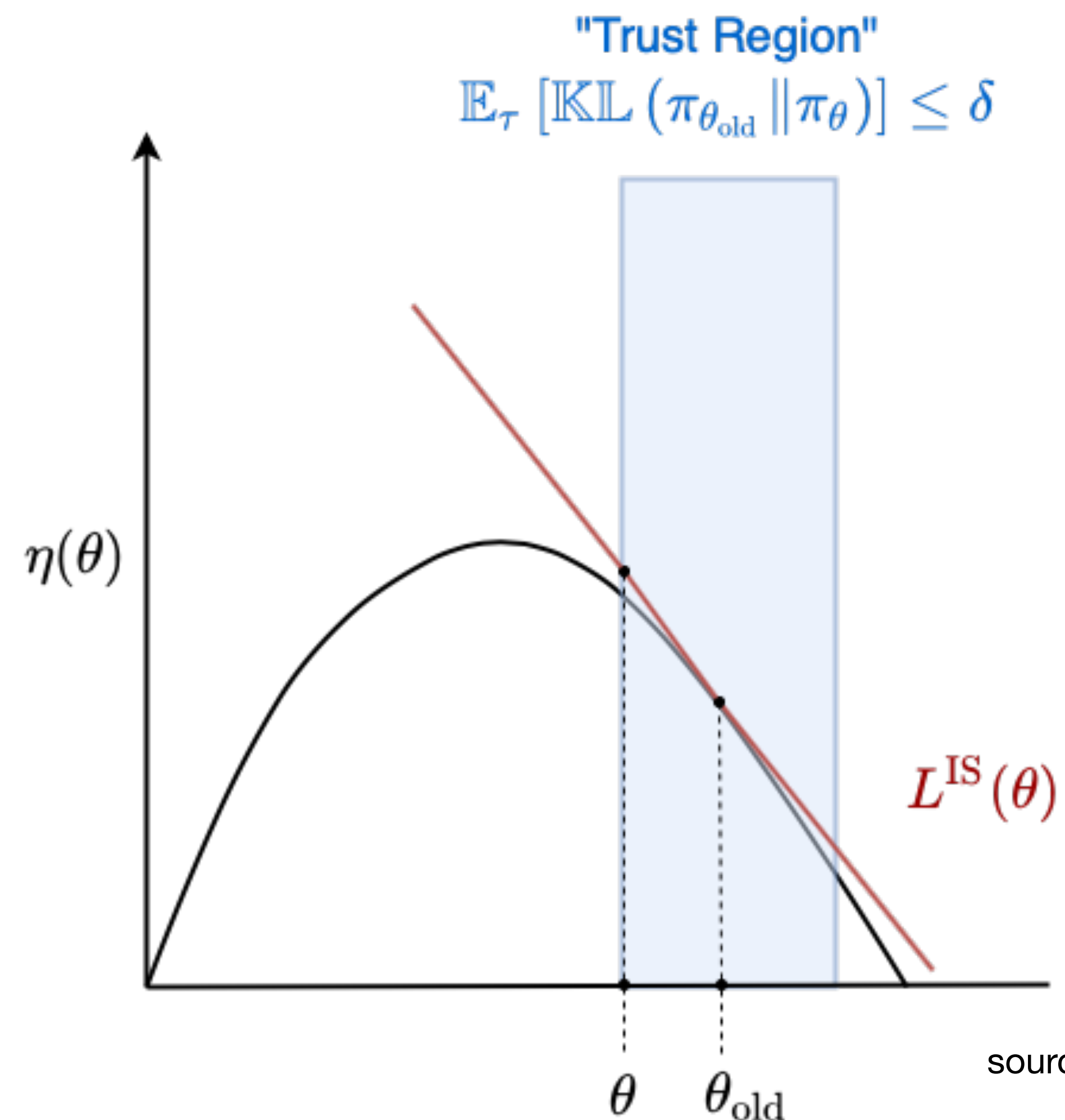
$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \dots \ln \pi_{\theta}(y_i | x)$$

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta_{\text{old}}}(\cdot | x)} \dots \frac{\ln \pi_{\theta}(y_i | x)}{\ln \pi_{\theta_{\text{old}}}(y_i | x)}$$

problem: if the ratio gets too crazy, our estimate ‘blows’ up.

Applying IS to REINFORCE

Main fix: just stay within some “safe” region!



TRPO: Keep within some KL constraint

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta_{\text{old}}}(\cdot | x)} \cdots \frac{\ln \pi_{\theta}(y_i | x)}{\ln \pi_{\theta_{\text{old}}}(y_i | x)}$$

$$\text{s.t.} \quad \hat{\mathbb{E}}_t \left[\text{KL} \left[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_{\theta}(\cdot | s_t) \right] \right] \leq \delta$$

Don't let the policy stray too far!

Requires some more complex techniques to solve...

PPO: just clip the ratio!

Use the same importance sampling technique:

$$r_i = \frac{\ln \pi_{\theta}(y_i | x)}{\ln \pi_{\theta_{\text{old}}}(y_i | x)}$$

But now just clip the ratio to avoid moving too far:

$$L = \hat{\mathbb{E}} \left[\min \left(r_i(\theta) \hat{A}_i, \text{clip}(r_i(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_i \right) \right]$$

Why min on top of clip? Because if the estimate would make advantage higher, its okay - we just want to be pessimistic about true advantage.

Standard PPO = REINFORCE+clip+advantage

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} \left[\min \left(r_i(\theta) \hat{A}_i, \text{clip}(r_i(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_i \right) \right] \right]$$

$$r_i = \frac{\ln \pi_{\theta}(y_i | x)}{\ln \pi_{\theta_{\text{old}}}(y_i | x)}$$

$$A_i(x, y) = R(y, x) - \hat{V}_{\theta'}(x)$$

Can we improve the IS further?

DAPO (Yu et al 2025): Uncouple the clipping parameter, and raise the higher one.

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot|x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} \left[\min \left(r_i(\theta) \hat{A}_i, \text{clip} \left(r_i(\theta), 1 - \epsilon_{\text{lower}}, 1 + \epsilon_{\text{higher}} \right) \hat{A}_i \right) \right] \right]$$

Can we improve the IS further?

DAPO (Yu et al 2025): Uncouple the clipping parameter, and raise the higher one.

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot|x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} \left[\min \left(r_i(\theta) \hat{A}_i, \text{clip} \left(r_i(\theta), 1 - \epsilon_{\text{lower}}, 1 + \epsilon_{\text{higher}} \right) \hat{A}_i \right) \right] \right]$$

Basically: allow larger updates when raising token probabilities, but don't further decrease the lower bound.

Minimax paper goes further and sets ϵ_{lower} to infinity to avoid lower bound clipping!

Can we improve the IS further?

CISPO (Minimax 2025): directly clip ratio, instead of overall advantage.

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot|x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} \left[\min \left(r_i(\theta), \text{clip} \left(r_i(\theta), 1 - \epsilon_{lower}, 1 + \epsilon_{higher} \right) \right) \hat{A}_i \right] \right]$$

Can we improve the IS further?

CISPO (Minimax 2025): directly clip ratio, instead of overall advantage.

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot|x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} \left[\min \left(r_i(\theta), \text{clip} \left(r_i(\theta), 1 - \epsilon_{lower}, 1 + \epsilon_{higher} \right) \right) \hat{A}_i \right] \right]$$

Basically: directly clip the ratio, to allow large advantages to still have some effect - even when lots of clipping might be happening.

Can we improve the IS further?

GSPO (Zheng et al 2025): use sequence-level importance sampling instead of token-level.

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} \left[\min \left(\textcolor{red}{s}_i(\theta) \hat{A}_i, \text{clip}(\textcolor{red}{s}_i(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_i \right) \right] \right]$$

$$s_i(\theta) = \left(\frac{\pi_{\theta}(y_i | x)}{\pi_{\theta_{\text{old}}}(y_i | x)} \right)^{\frac{1}{|y_i|}} = \exp \left(\frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \ln \frac{\pi_{\theta}(y_i | x)}{\pi_{\theta_{\text{old}}}(y_i | x)} \right)$$

Can we improve the IS further?

GSPO (Zheng et al 2025): use sequence-level importance sampling instead of token-level.

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} \left[\min \left(\textcolor{red}{s}_i(\theta) \hat{A}_i, \text{clip}(\textcolor{red}{s}_i(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_i \right) \right] \right]$$

$$s_i(\theta) = \left(\frac{\pi_{\theta}(y_i | x)}{\pi_{\theta_{\text{old}}}(y_i | x)} \right)^{\frac{1}{|y_i|}} = \exp \left(\frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \ln \frac{\pi_{\theta}(y_i | x)}{\pi_{\theta_{\text{old}}}(y_i | x)} \right)$$

Basically: compute the importance ratio of the whole sequence, and then apply the same value to each token. Reduces noise of token-to-token.

Going beyond REINFORCE

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} A(x, y_i) \nabla_{\theta} \ln \pi_{\theta}(y_i | x) \right]$$

Issues:

- High variance 
- Reward scale can cause issues... (consider neg. vs pos. reward) 
- We can only use samples directly from our current loop

Going beyond REINFORCE

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot|x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} \left[\min \left(r_i(\theta) \hat{A}_i, \text{clip}(r_i(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_i \right) \right] \right]$$

Issues:

- High variance 
- Reward scale can cause issues... (consider neg. vs pos. reward) 
- We can only use samples directly from our current loop 

Going beyond REINFORCE

$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot | x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i}^{y_T} \left[\min \left(r_i(\theta) \hat{A}_i, \text{clip}(r_i(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_i \right) \right] \right]$$

$$r_i = \frac{\ln \pi_{\theta}(y_i | x)}{\ln \pi_{\theta_{\text{old}}}(y_i | x)}$$

$$A_i(x, y) = R(y, x) - \hat{V}_{\theta'}(x)$$

One last detail: KL penalty

In practice, we sometime also apply a KL penalty to the reward:

$$\max_{\pi_{\theta}} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta}(y|x)} [r_{\phi}(x, y)] - \beta \mathbb{D}_{\text{KL}} [\pi_{\theta}(y | x) || \pi_{\text{ref}}(y | x)]$$

However, these days often the KL penalty is completely removed.
So, I'm going to skip over this.

Backing up to the full algorithm

Algorithm 1 PPO-Clip

- 1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
- 4: Compute rewards-to-go $\hat{R}_t$.
- 5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k} .
- 6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

- 7: Fit value function by regression on mean-squared error:

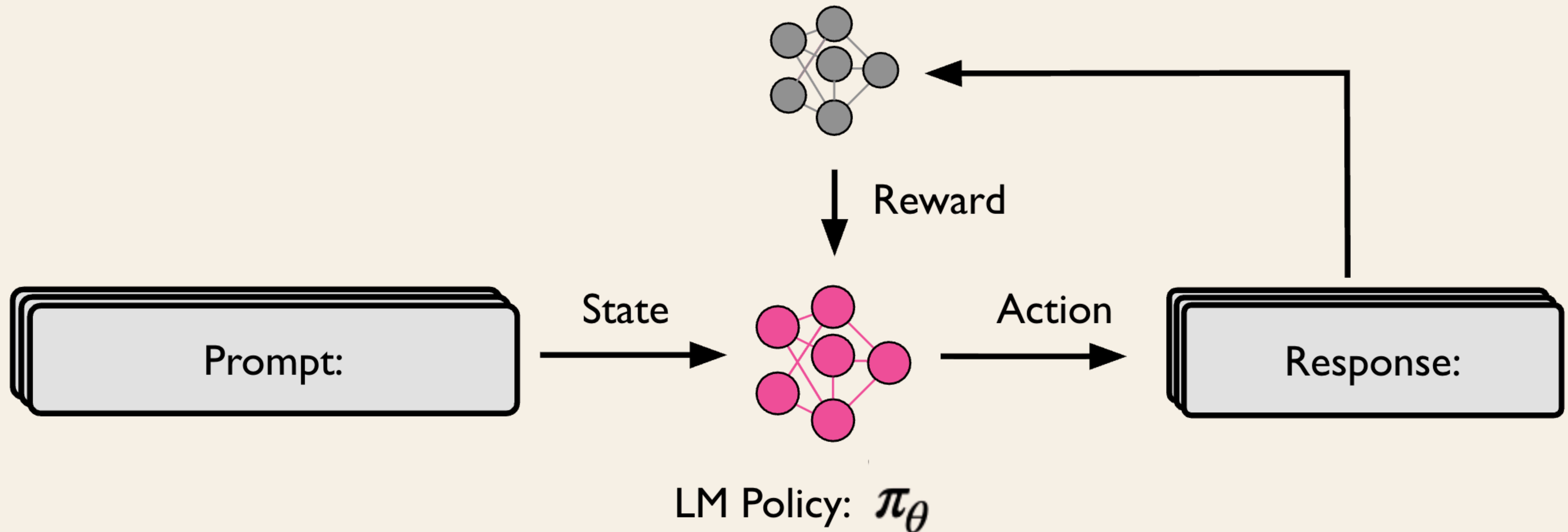
$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

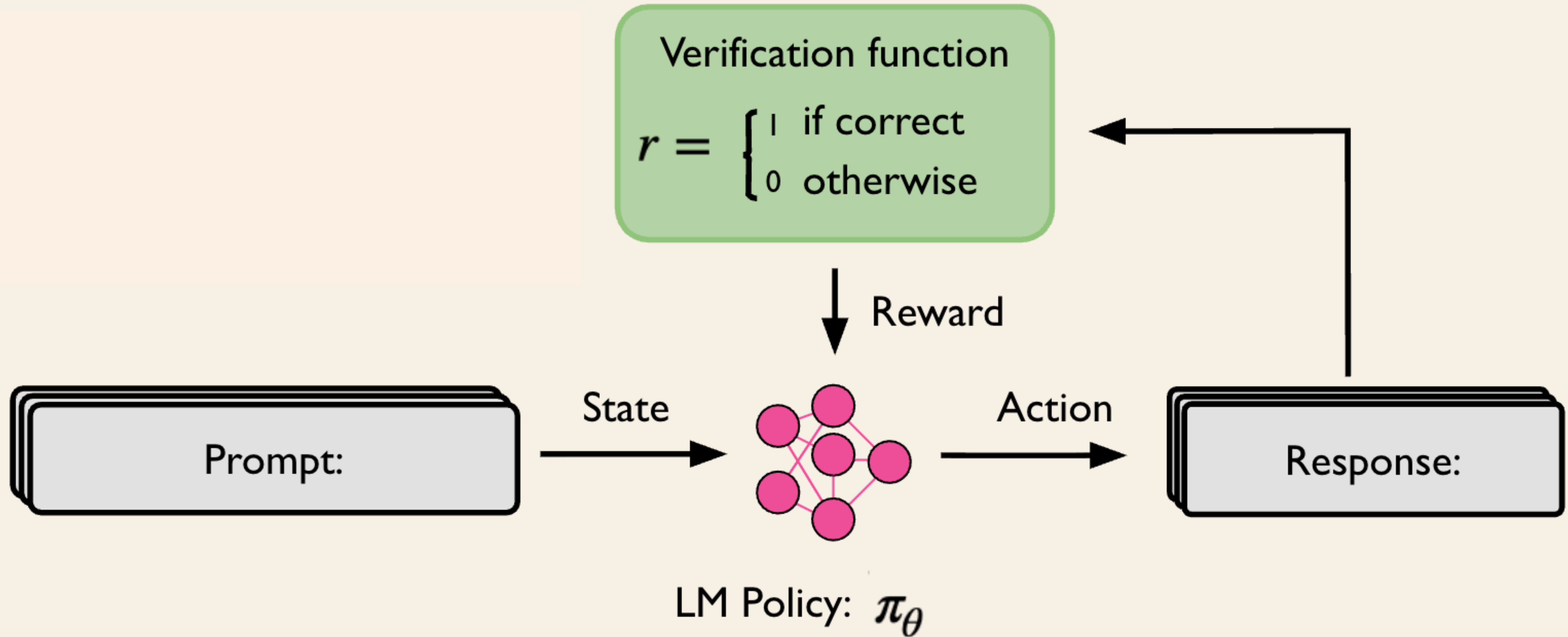
- 8: **end for**

Backing up to the full algorithm

Reward model $R_{\psi}(x, y)$



Backing up to the full algorithm



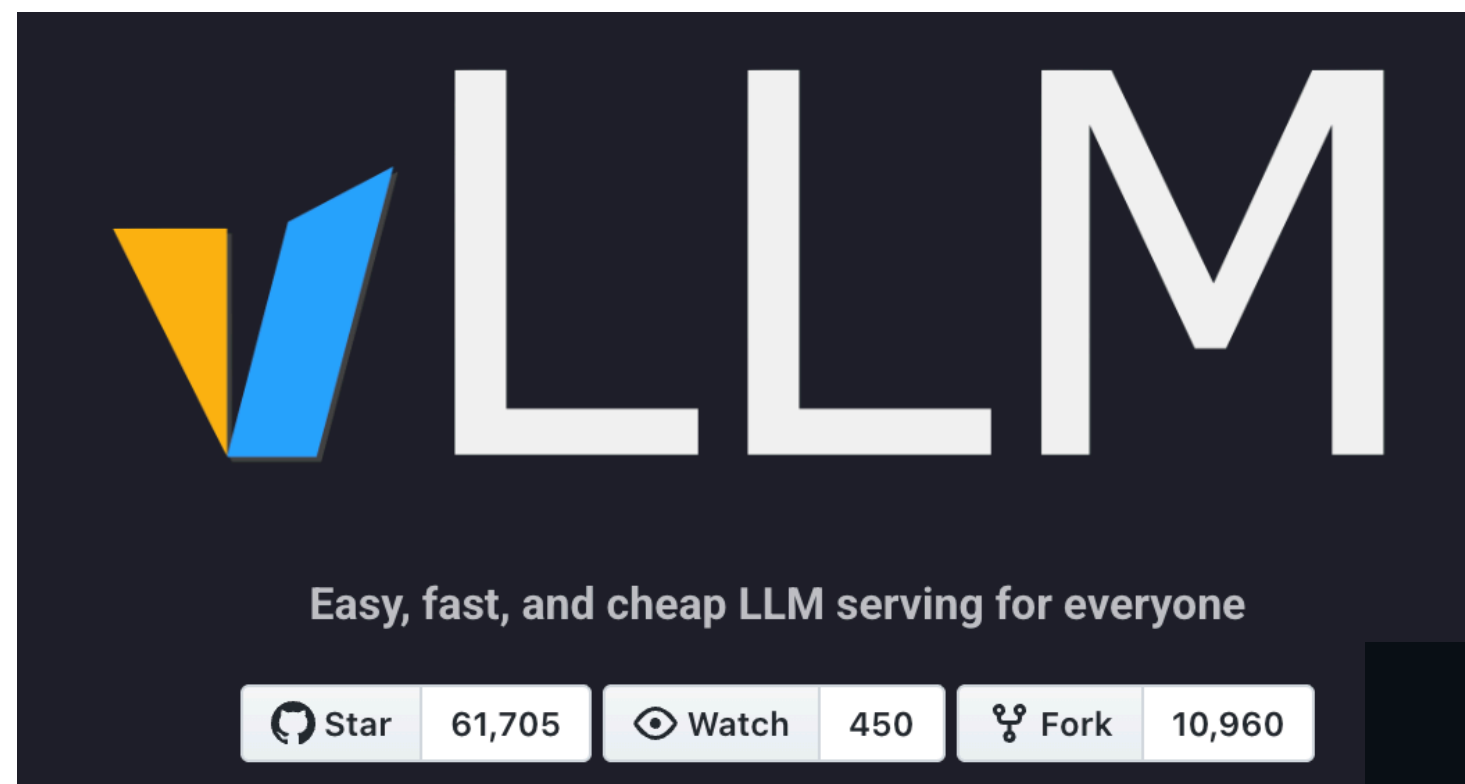
Adapting for LMs

Adapting for LMs

1. LM inference is slow. How can we speed things up?
2. How do we deal with long sequences?
3. How do we deal with zero-advantage groups (GRPO)?

Speeding up inference.

Lots of work done on speeding up inference by great teams focussed on inference-only settings. Much of this can be applied in the RL setting!



NVIDIA TensorRT

Speeding up inference: logprob mismatch.

One big issue: logprob computation can mismatch between inference and training! (Yao et al, 2025)

$$\mathbb{E}_{a \sim \pi_{\text{sampler}}(\theta_{\text{old}})} \left[\nabla_{\theta} \min \left(\frac{\pi_{\text{learner}}(a, \theta)}{\pi_{\text{learner}}(a, \theta_{\text{old}})} \hat{A}, \text{clip} \left(\frac{\pi_{\text{learner}}(a, \theta)}{\pi_{\text{learner}}(a, \theta_{\text{old}})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A} \right) \right].$$

We are sampling via vllm, but computing our logprobs in our training code.

Speeding up inference: logprob mismatch.

One big issue: logprob computation can mismatch between inference and training!

Is this problem familiar?

Train on “old” samples w/ IS

We can reuse old samples with importance sampling!

$$\mathbb{E}_{\tau \sim p_{\theta}}[f(\tau)] = \mathbb{E}_{\tau \sim p_{\beta}} \left[\frac{p_{\theta}(\tau)}{p_{\beta}(\tau)} f(\tau) \right]$$

some function on
trajectories over
policy

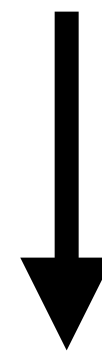
Same function, but
now we drawing
trajectories from diff
policy, and reweight
accordingly!

Speeding up inference: logprob mismatch.

One big issue: logprob computation can mismatch between inference and training!

We apply the same solution as we did for training on old data: importance sampling!

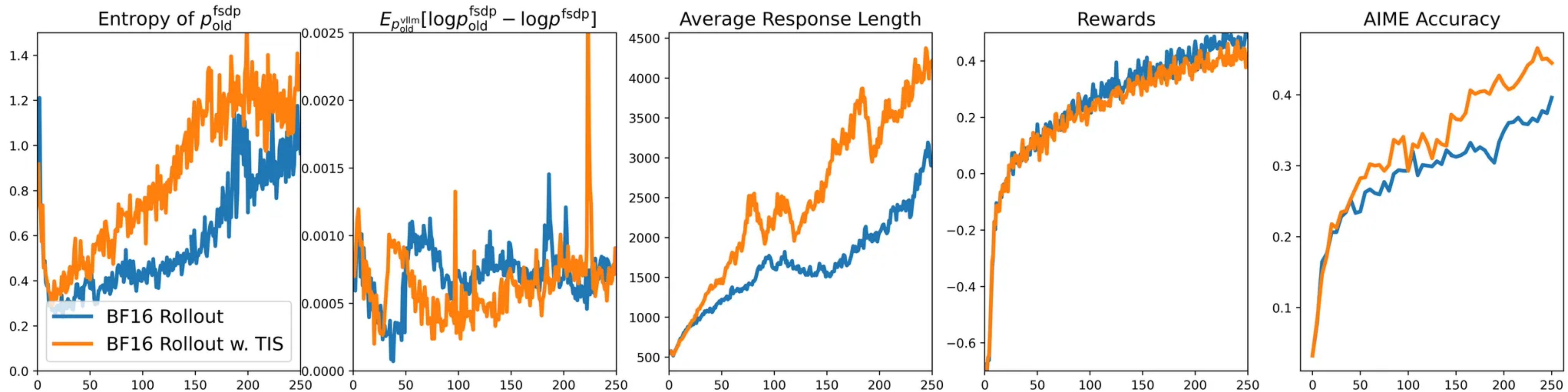
$$\mathbb{E}_{a \sim \pi_{\text{sampler}}(\theta_{\text{old}})} \left[\nabla_{\theta} \min \left(\frac{\pi_{\text{learner}}(a, \theta)}{\pi_{\text{learner}}(a, \theta_{\text{old}})} \hat{A}, \text{clip} \left(\frac{\pi_{\text{learner}}(a, \theta)}{\pi_{\text{learner}}(a, \theta_{\text{old}})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A} \right) \right].$$



$$\mathbb{E}_{a \sim \pi_{\text{sampler}}(\theta_{\text{old}})} \left[\underbrace{\min \left(\frac{\pi_{\text{learner}}(a, \theta_{\text{old}})}{\pi_{\text{sampler}}(a, \theta_{\text{old}})}, C \right)}_{\text{truncated importance ratio}} \cdot \nabla_{\theta} \min \left(\frac{\pi_{\text{learner}}(a, \theta)}{\pi_{\text{learner}}(a, \theta_{\text{old}})} \hat{A}, \text{clip} \left(\frac{\pi_{\text{learner}}(a, \theta)}{\pi_{\text{learner}}(a, \theta_{\text{old}})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A} \right) \right]$$

Speeding up inference: logprob mismatch.

This can be the difference between collapse and success.
Empirically, I find it more important the more off-policy you go.



source: Yao et al. 2025

Speeding up inference: logprob mismatch.

More recently: maybe just using FP32 precision for LM head works?

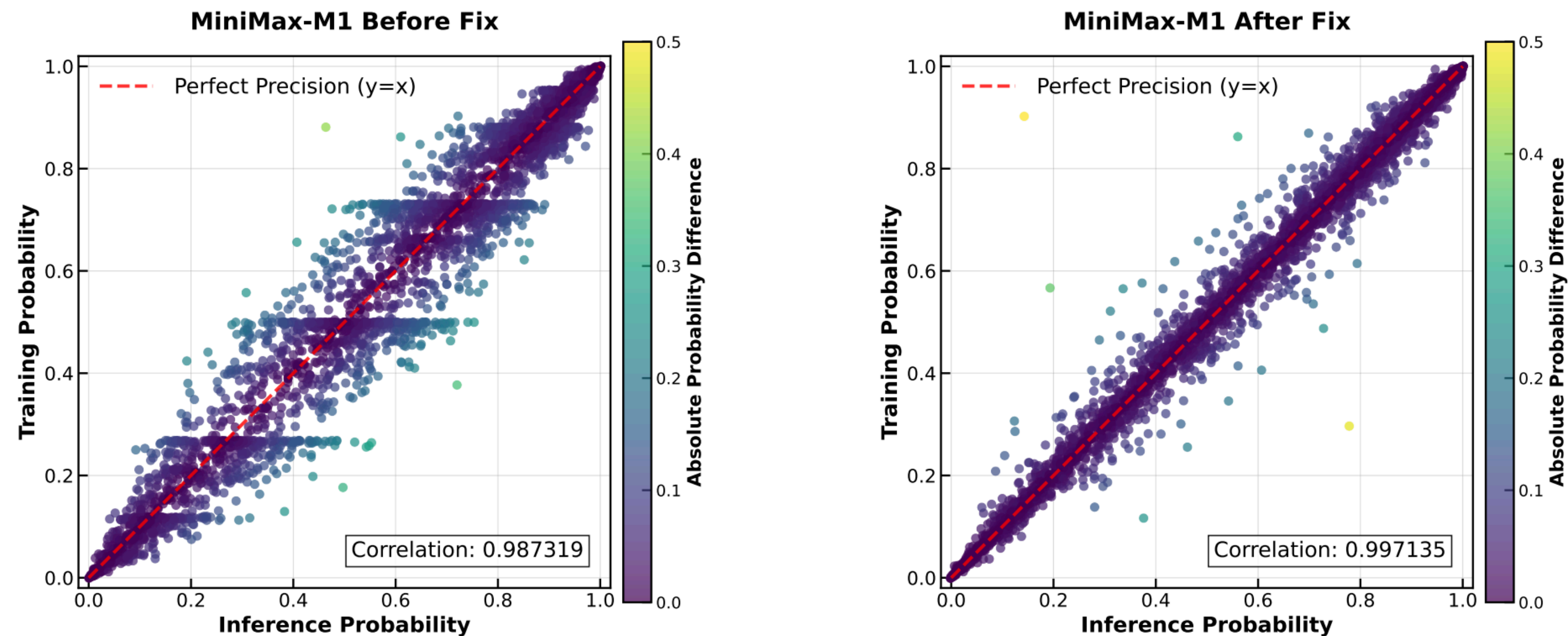
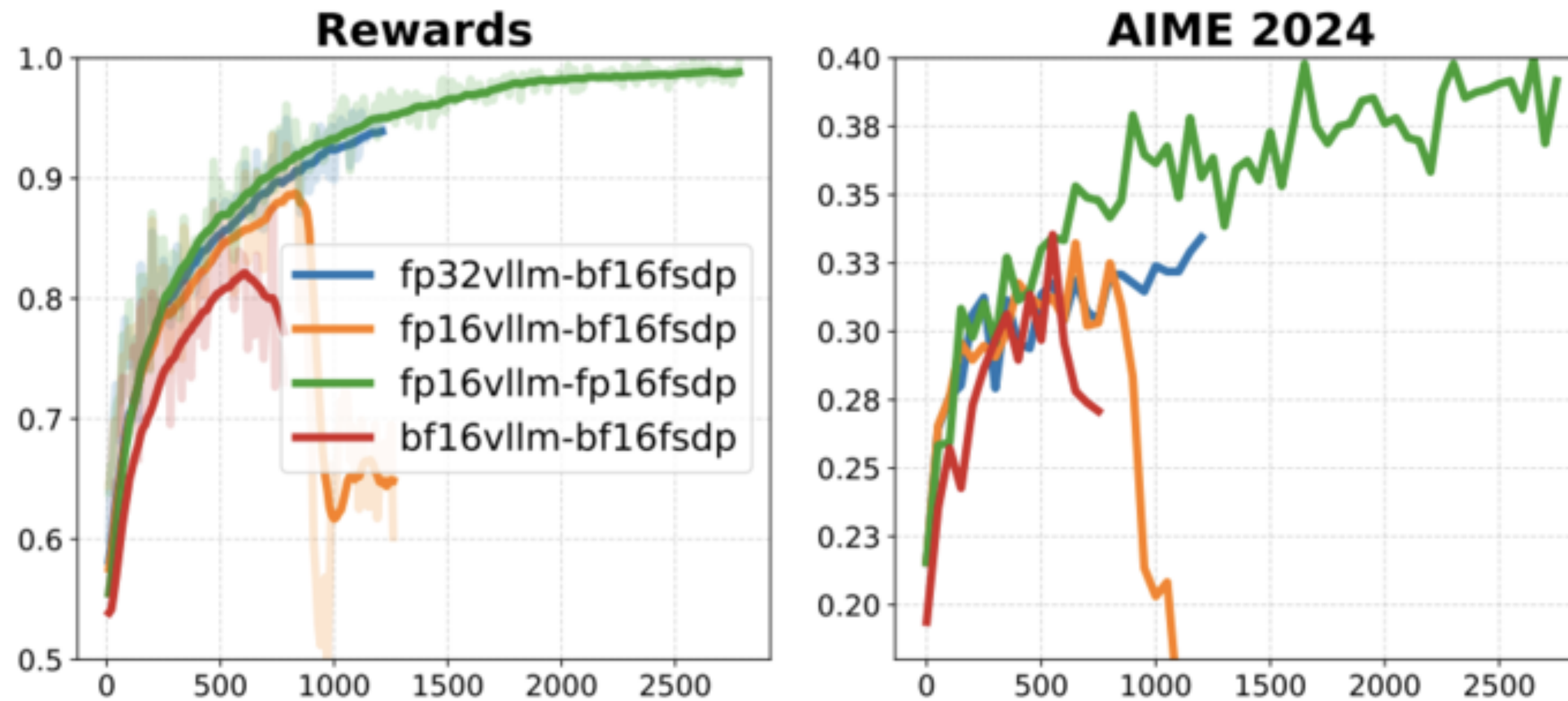


Figure 3 | Probability of tokens in training-mode code vs. probability of tokens in inference-mode code. Each point in the figures represents an individual token. The Pearson correlation coefficient is indicated in the figures. Theoretically, the two probabilities should be identical, and all the tokens should be exactly on the diagonal line. **Left:** Correlation of the M1 model before our fix; **Right:** Correlation of the M1 model after applying our fix of using FP32 precision for the LM output head.

Speeding up inference: logprob mismatch.

More recently: maybe just using FP16 precision fixes all these issues???



only stable when:

1. FP32 vllm,
BF16 training.

2. FP16 vllm,
FP16 training

Speeding up inference: logprob mismatch.

Even more recently: maybe theres just bugs in the inference code????



@danielhanchen, glad you liked the post! You're spot on to suspect lower-level implementation issues. That's exactly what we found in the original blog.

The `disable_cascade_attn` finding (Sec 4.2.4) was the symptom, but the root cause was that silent FlashAttention-2 kernel bug we detailed, which was particularly problematic on A100s:

On A100 GPUs (and L20s), under certain batch/seq lengths, the kernel triggers its `split_kv` path. This path had a bug that incorrectly transposed the LSE layout.

This corrupted tensor caused a "complete precision collapse" in Cascade Attention, which is what we measured as that massive mismatch KL.

Good luck with the FP16/BF16 tests, I'd love to know what you find!

yingru.notion.site/When-Speed-Kil...

source: <https://x.com/RichardYRLi/status/1984858850143715759>

4.2.4 Disable Cascade Attention in VLLM

According to this [GitHub issue](#), we set `disable_cascade_attn=True` when initializing the vLLM engine and found that it significantly helps reduce the training-inference mismatch in experiments conducted on A100 GPUs. We performed two on-policy GRPO experiments using Qwen3-14B-Base as the base model on A100 GPUs, with `disable_cascade_attn` set to `True` and `False`, respectively. The results are shown in Figure 21. It can be observed that after disabling cascade attention, the `vllm-kl` metric decreased from the range of $5e-2$ to $1e-1$ to around $1e-3$, indicating a substantial reduction in training-inference mismatch. In addition, the rewards on the training set also increased appropriately.

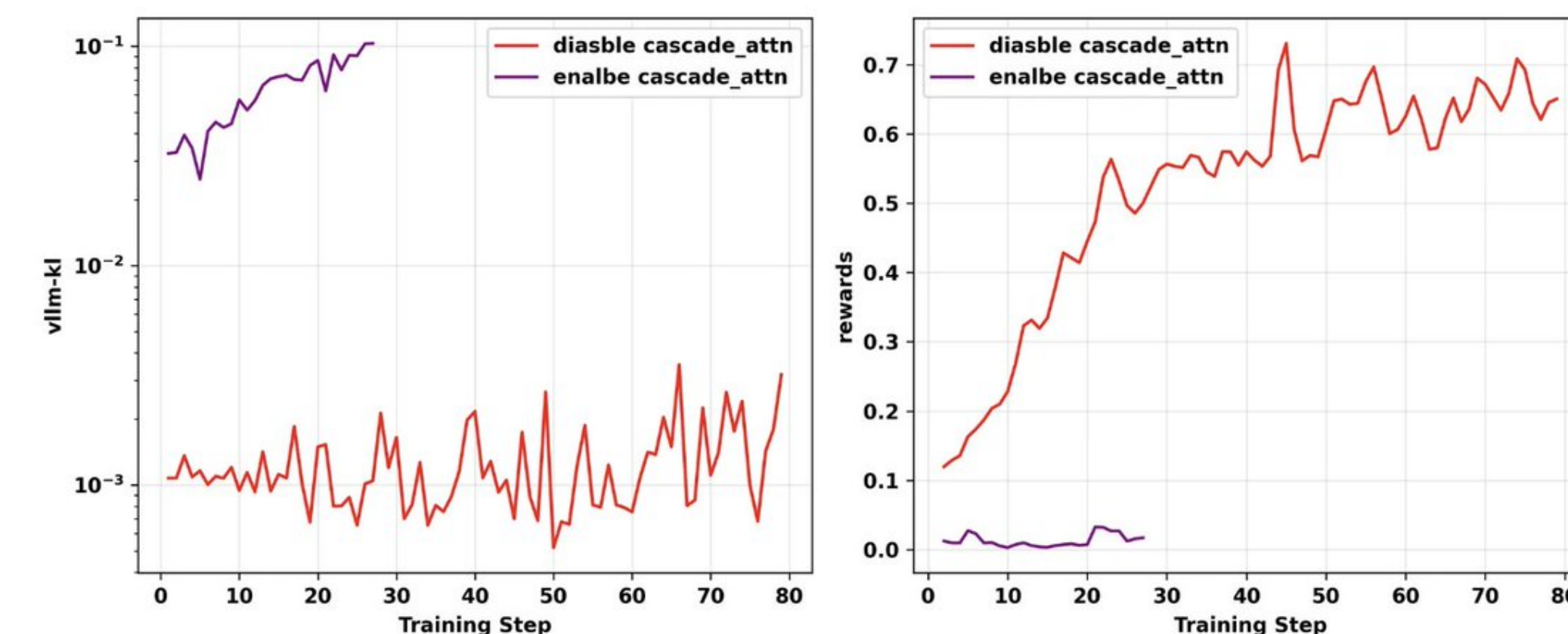


Figure 21: Results of the ablation study on using cascade attention. We conducted two on-policy experiments on A100 GPUs: one with `cascade attention enabled` and another with the same configuration but with `cascade attention disabled`. Left: The dynamics of `vllm-kl` metric. Right: the dynamics of training rewards

source: Liu et al 2025

Speeding up inference: logprob mismatch.

Even more recently: maybe theres just bugs in the inference code????



Speeding up inference: async RLHF

Let's review the standard algorithm for PPO/REINFORCE:

```
while True:
```

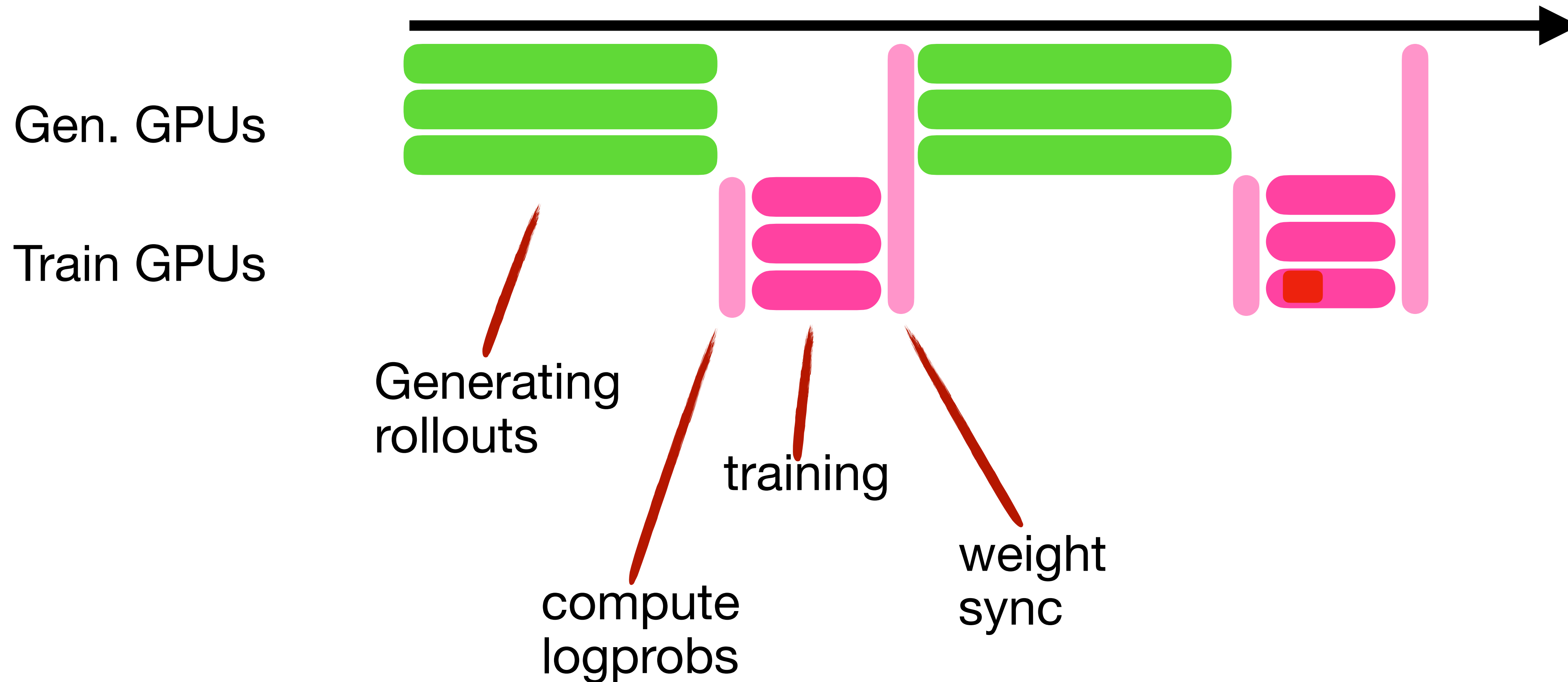
```
    Generate a full episode ("rollout") starting from  $s_0$  using  $\pi_\theta$ 
```

```
    Compute reward over the rollouts
```

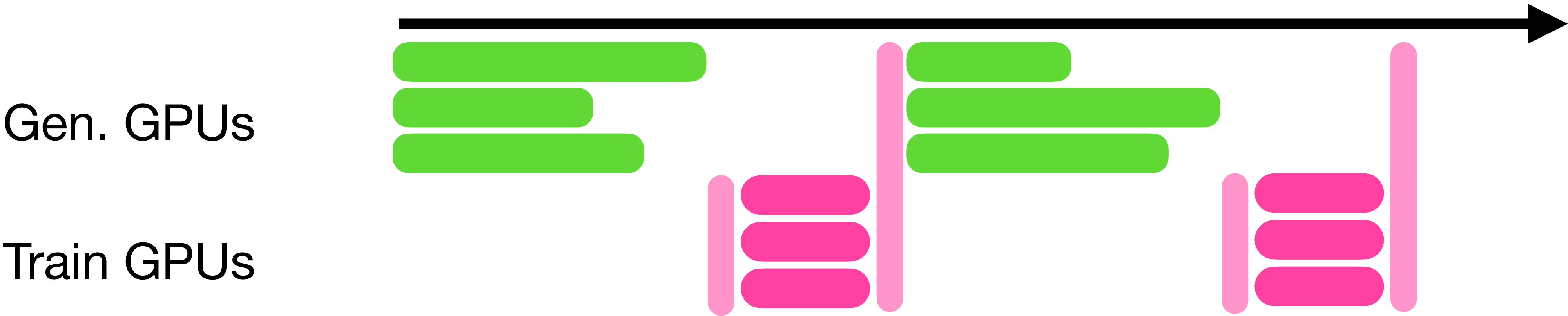
```
    Compute the loss and apply gradient (potentially mult. steps)
```


Speeding up inference: async RLHF

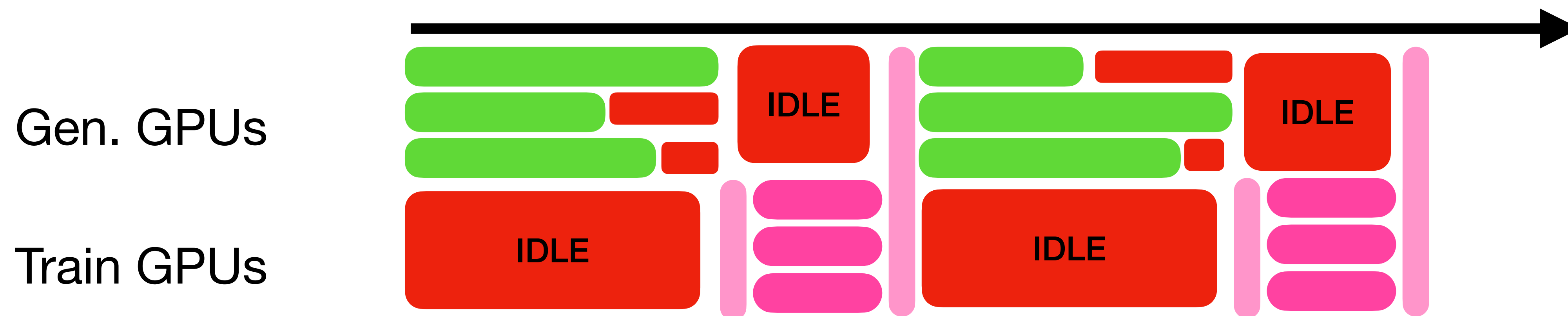
We can visualise this flow like:



Speeding up inference: async RLHF

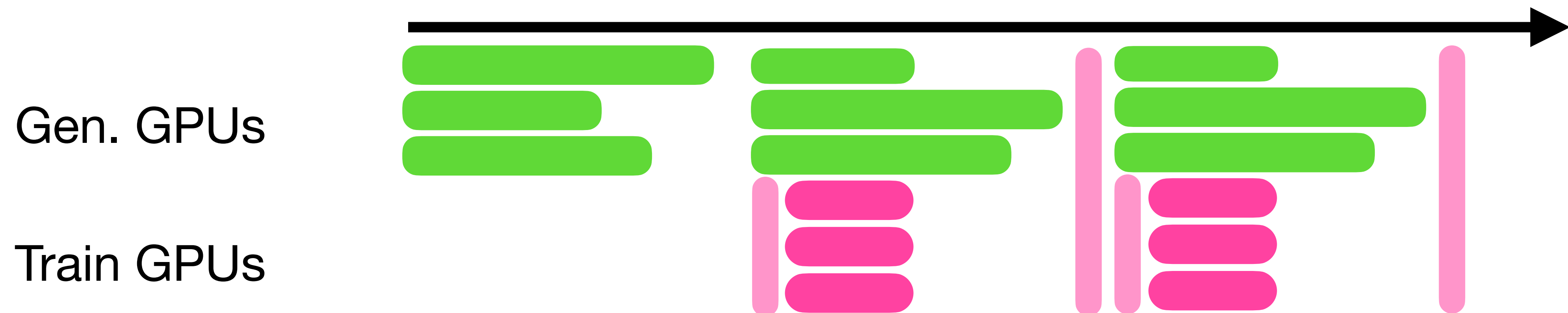


Speeding up inference: async RLHF



Speeding up inference: async RLHF

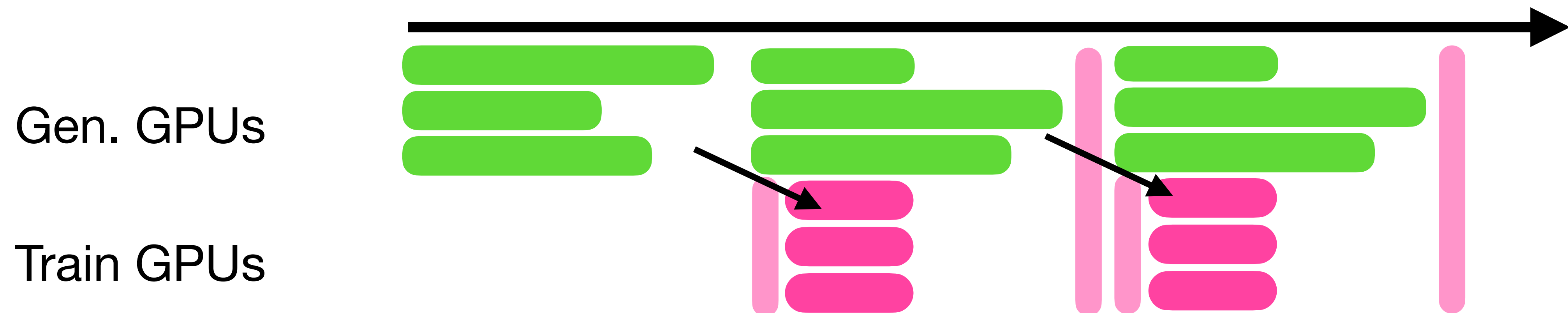
Fix 1: Overlap training and inference! This also makes our overall algorithm faster if we have enough compute!



Drawback: our inference is now 1 step off our training (off-policy). In practice, this doesn't hurt until > 8 steps. Applying importance sampling also helps!

Speeding up inference: async RLHF

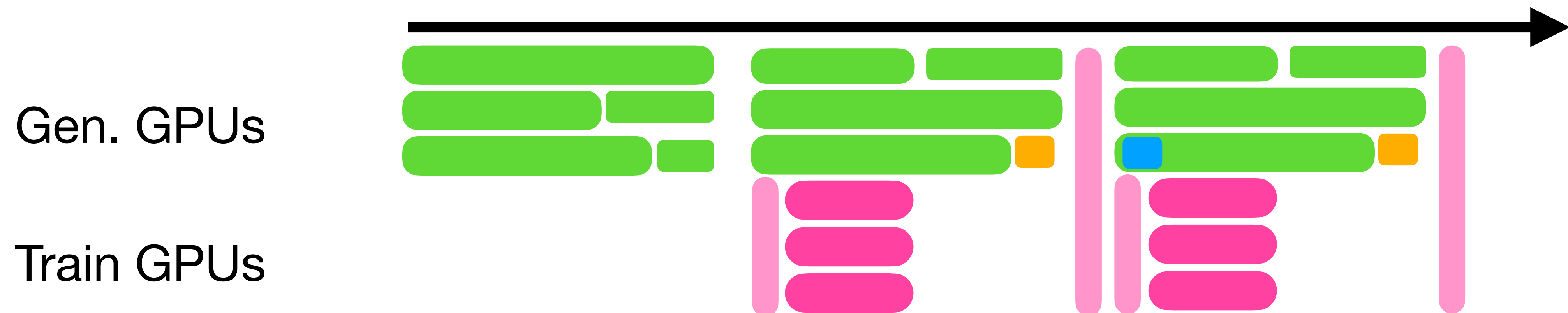
Fix 1: Overlap training and inference! This also makes our overall algorithm faster if we have enough compute!



Drawback: our inference is now 1 step off our training (off-policy). In practice, this doesn't hurt until > 8 steps. Applying importance sampling also helps!

Speeding up inference: async RLHF

Fix 2: Overlap training and inference, and continuously generate samples when inference compute frees up.

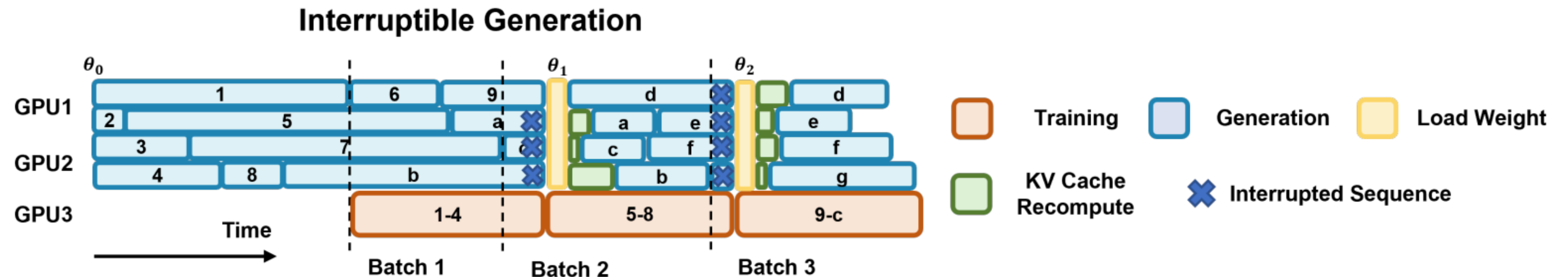


What do we do with interrupted generations? Just recompute KV cache and then continue.

Speeding up inference: async RLHF

Fix 2: Overlap training and inference, and continuously generate samples when inference compute frees up.

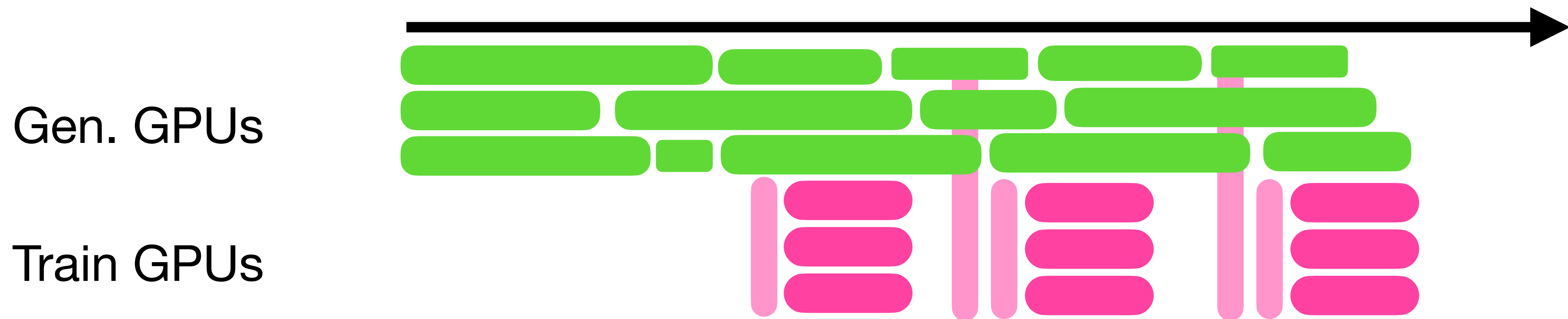
This is exactly what AReal does!



source: Fu et al 2025

Speeding up inference: async RLHF

Fix 3: Overlap training and inference, and continuously generate samples when inference compute frees up, and don't recompute KV cache!



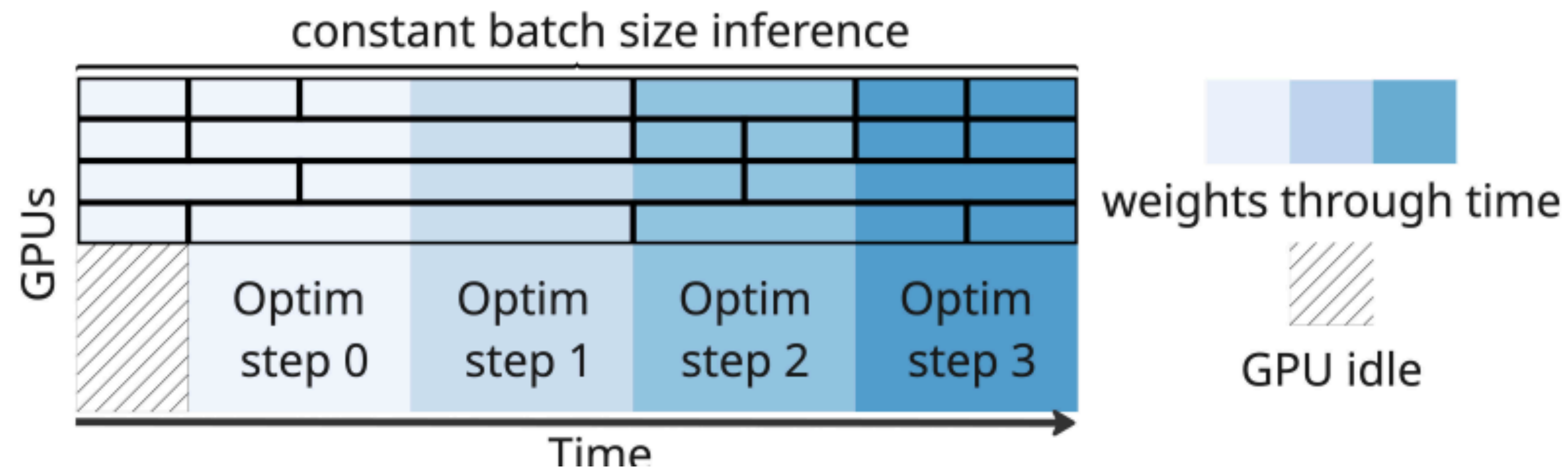
Run a training step whenever we have a batch available.

Scalable wrt inference compute.

Speeding up inference: async RLHF

Fix 3: Overlap training and inference, and continuously generate samples when inference compute frees up, and don't recompute KV cache!

This technique was proposed by pipelineRL.

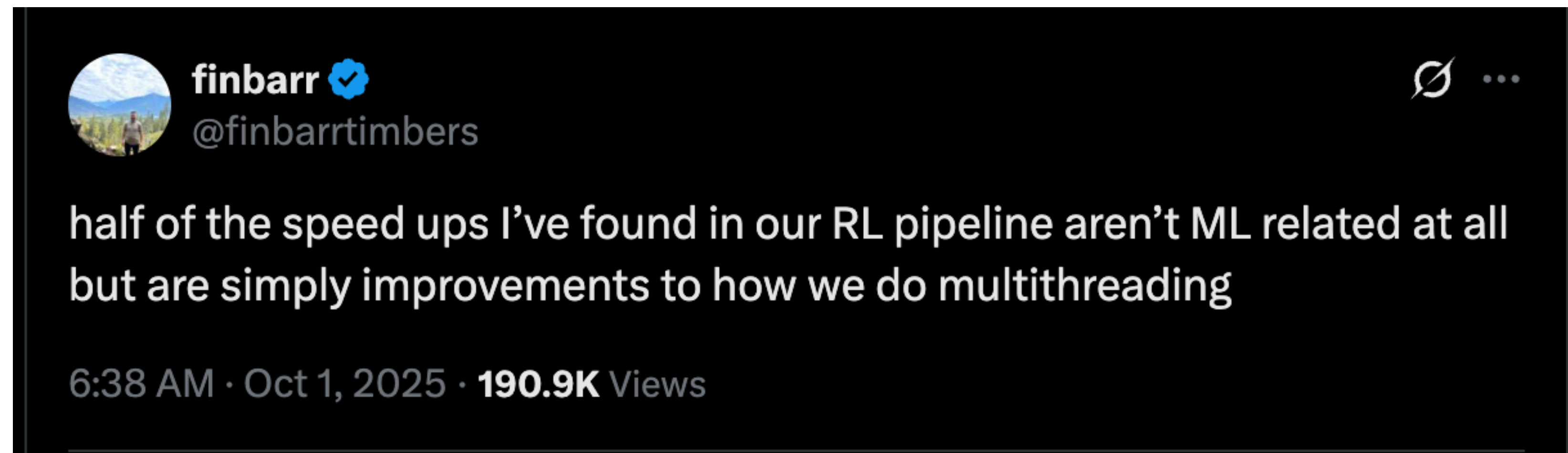


source: Piché et al 2025

Speeding up inference: async RLHF

Fix 3: Overlap training and inference, and continuously generate samples when inference compute frees up, and don't recompute KV cache!

This technique was proposed by pipelineRL. Requires being careful about multithreading!



source: <https://x.com/finbarrtimbers/status/1973382120451060161>

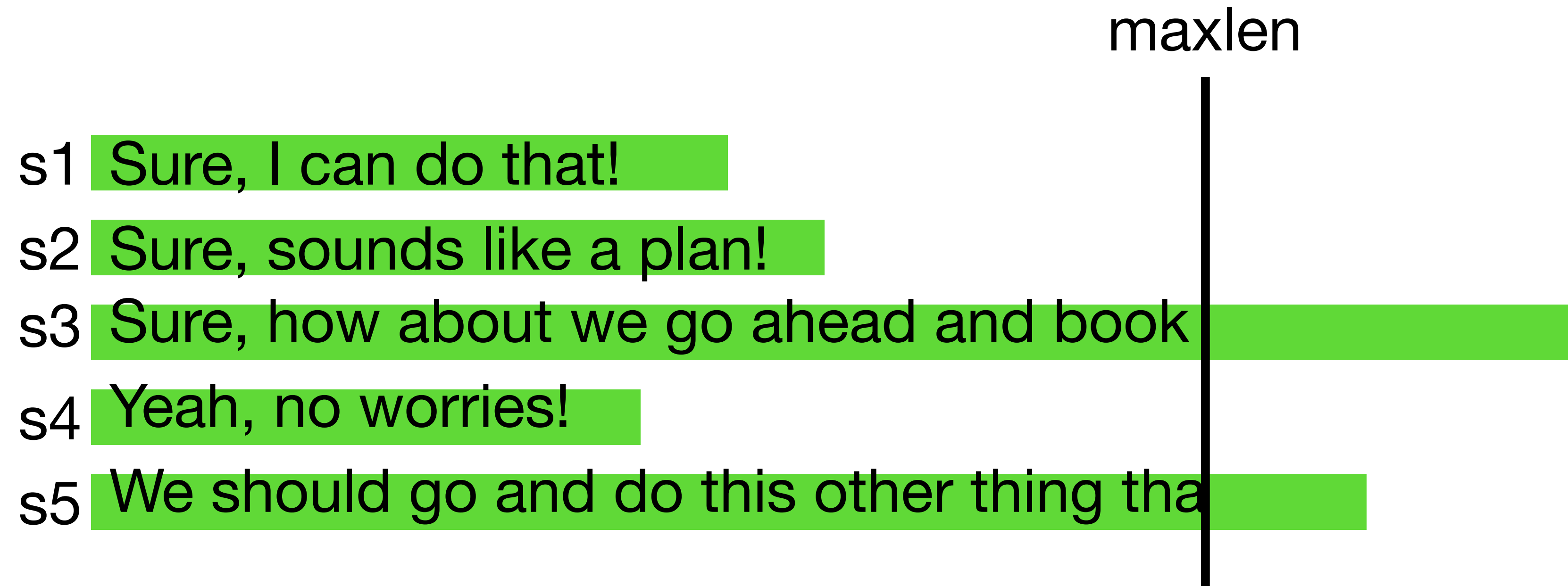
Adapting for LMs

1. LM inference is slow. How can we speed things up? 
2. How do we deal with long sequences?
3. How do we deal with zero-advantage groups (GRPO)?

Dealing with LM sequences

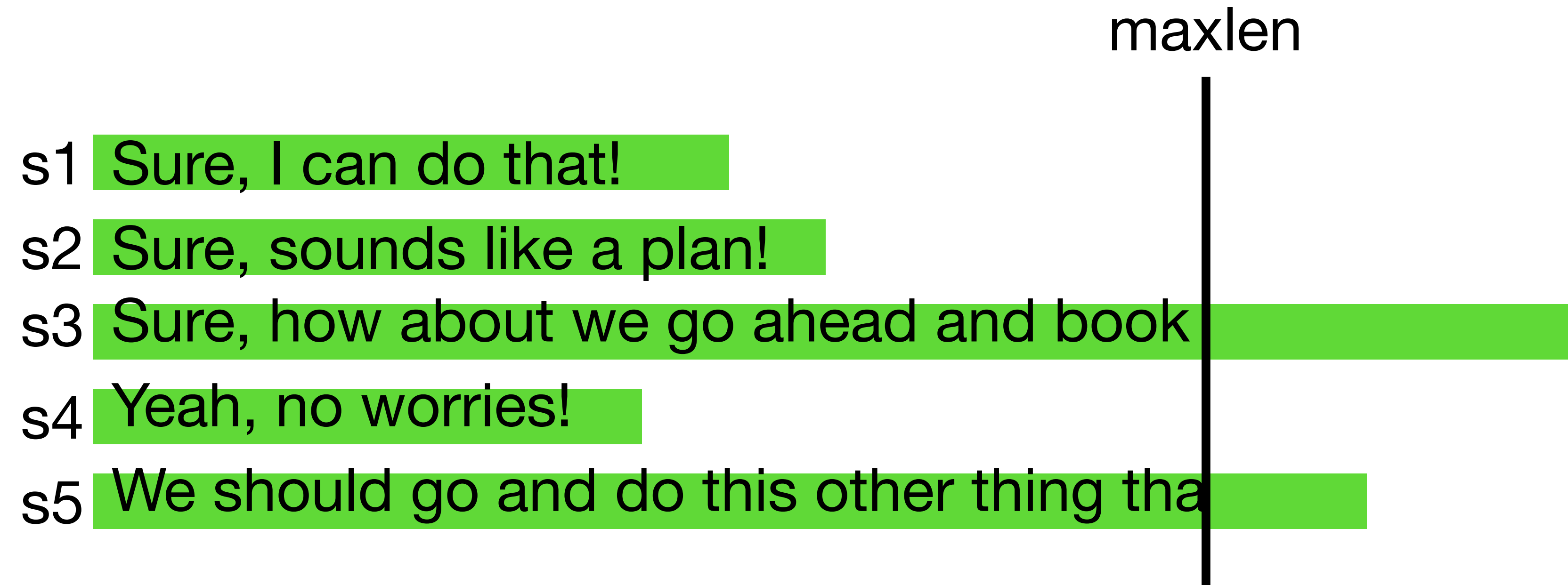
In practice, we have some **maximum length** we allow models to decode to.

This means some sequences will be **unfinished**.



Dealing with LM sequences

Vanilla: Treat and score as usual. Trains model to use only up to maxlen.

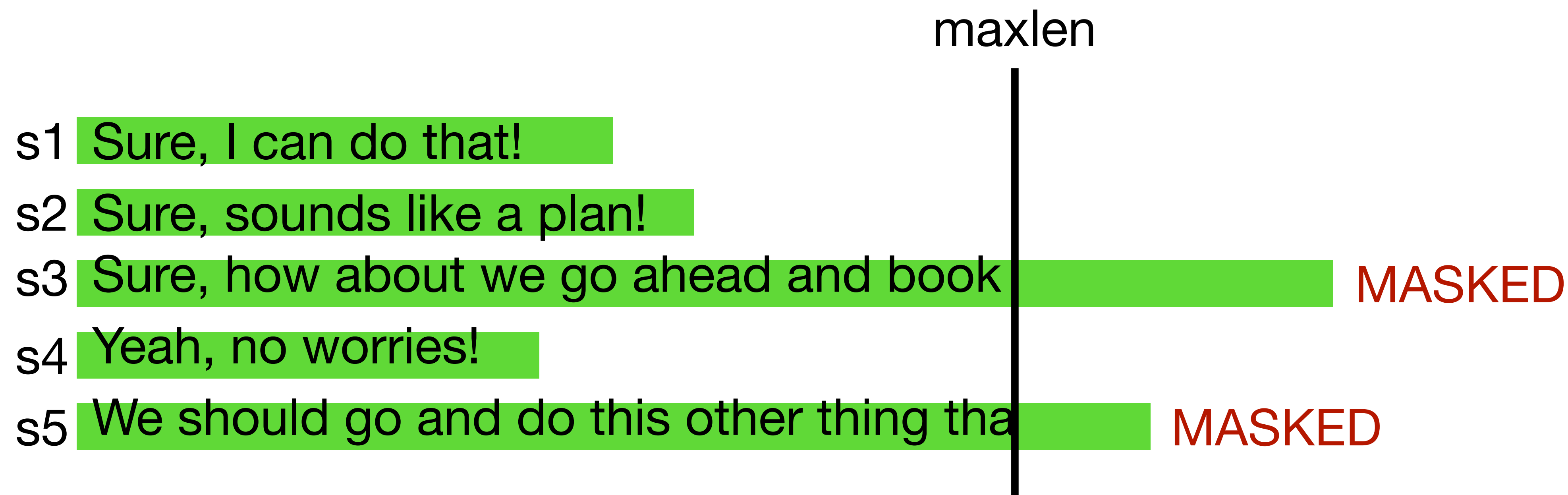


Dealing with LM sequences

Vanilla: Treat and score as usual. Trains model to use only up to maxlen.

Overlong filtering (DAPO): Filter and remove samples that don't finish.

Prevents penalising model for strategies that might be correct, but didn't finish [debated].

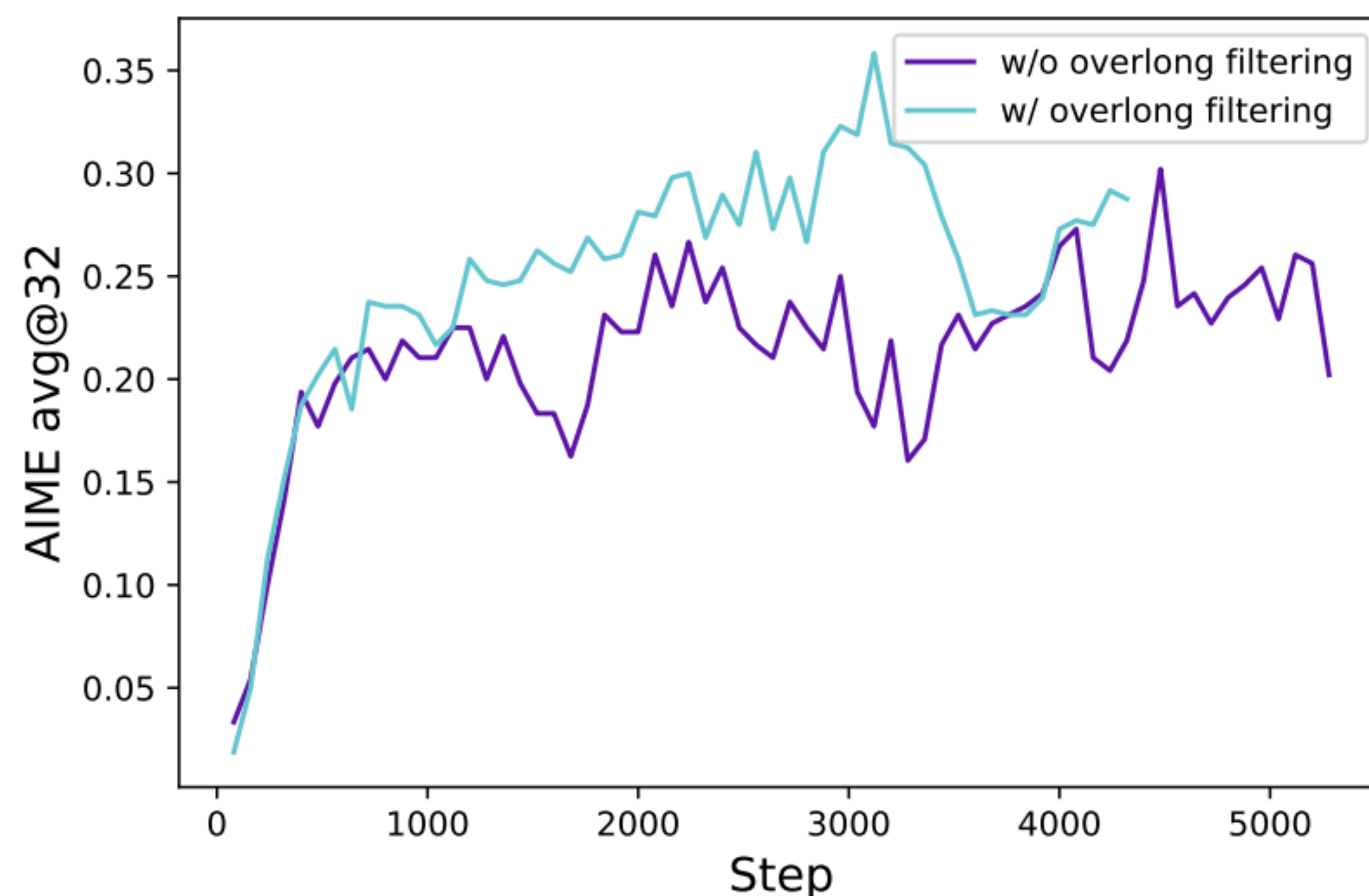


Dealing with LM sequences

Vanilla: Treat and score as usual. Trains model to use only up to maxlen.

Overlong filtering (DAPO): Filter and remove samples that don't finish.

Prevents penalising model for strategies that might be correct, but didn't finish [debated].



(a) Performance on AIME.

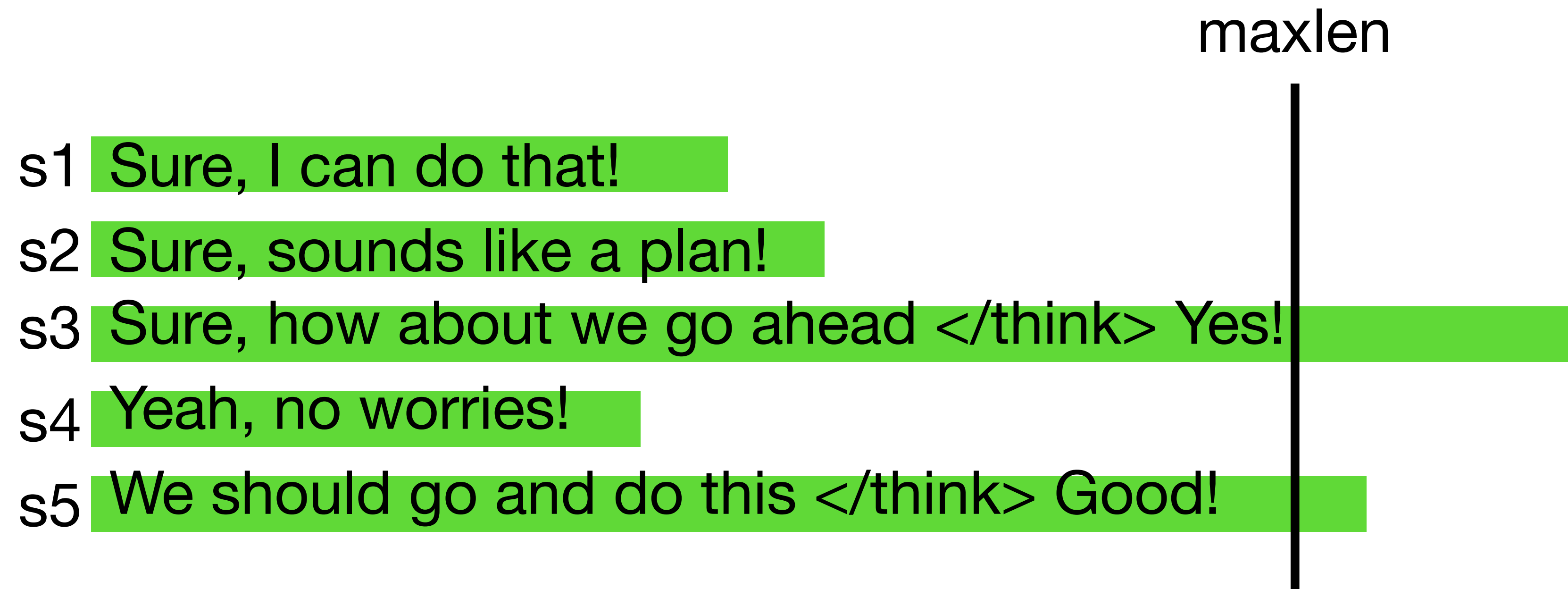
Dealing with LM sequences

Vanilla: Treat and score as usual. Trains model to use only up to maxlen.

Overlong filtering (DAPO): Filter and remove samples that don't finish.

Prevents penalising model for strategies that might be correct, but didn't finish [debated].

Interruptions (ScaleRL): Apply *budget forcing* and make the model stop (append “okay, now to stop thinking and give an answer</answer>” in 10-12k token range).



Dealing with LM sequences

Length penalties can also be used to control LM generation length, but these aren't super widely applied in the open?

$$r(y, y_{gold}, n_{gold}) = \mathbb{I}(y = y_{gold}) - \alpha \cdot |n_{gold} - n_y|,$$

gold len actual len



Basically subtract (or add) how far from “gold” length we are!

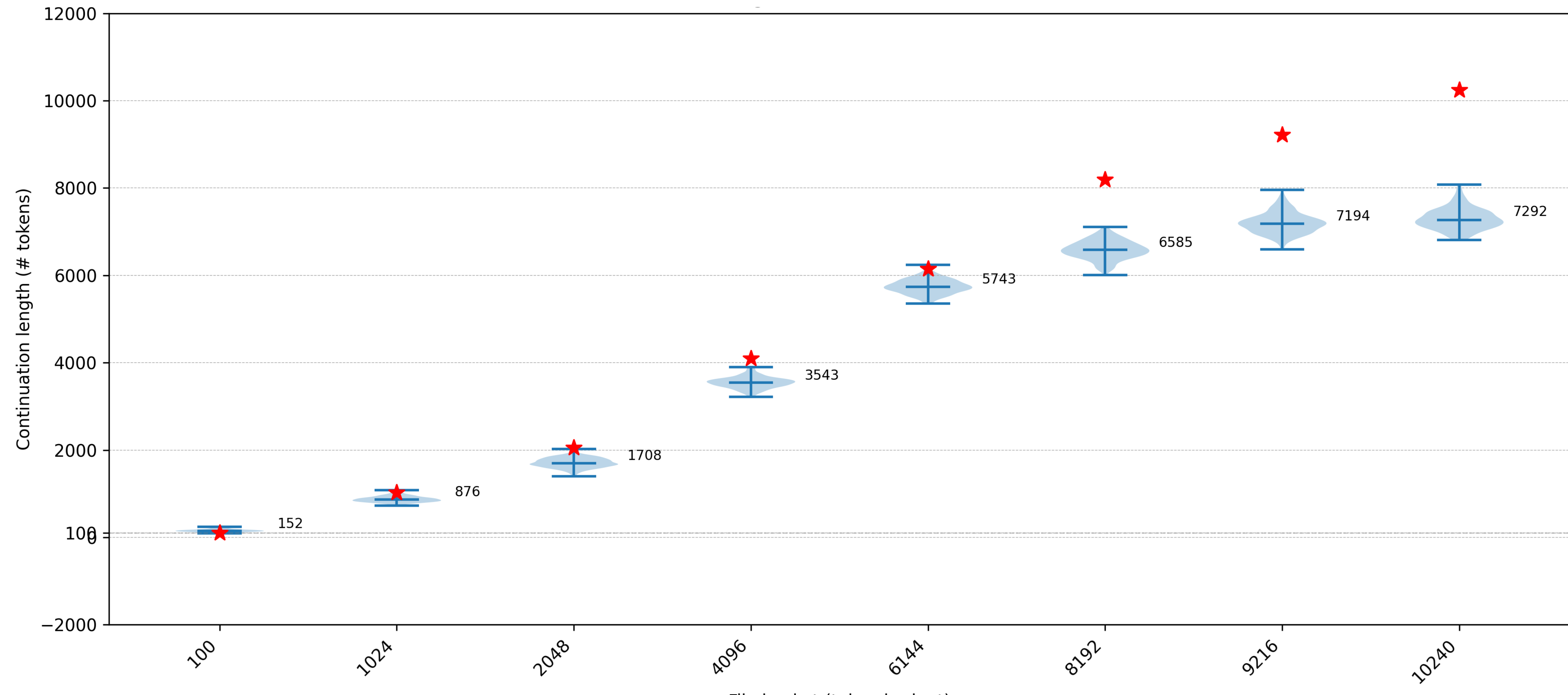
Published as a conference paper at COLM 2025

**L1: Controlling How Long A Reasoning Model Thinks
With Reinforcement Learning**

Pranjal Aggarwal & Sean Welleck
Carnegie Mellon University
{pranjal, swelleck}@cs.cmu.edu

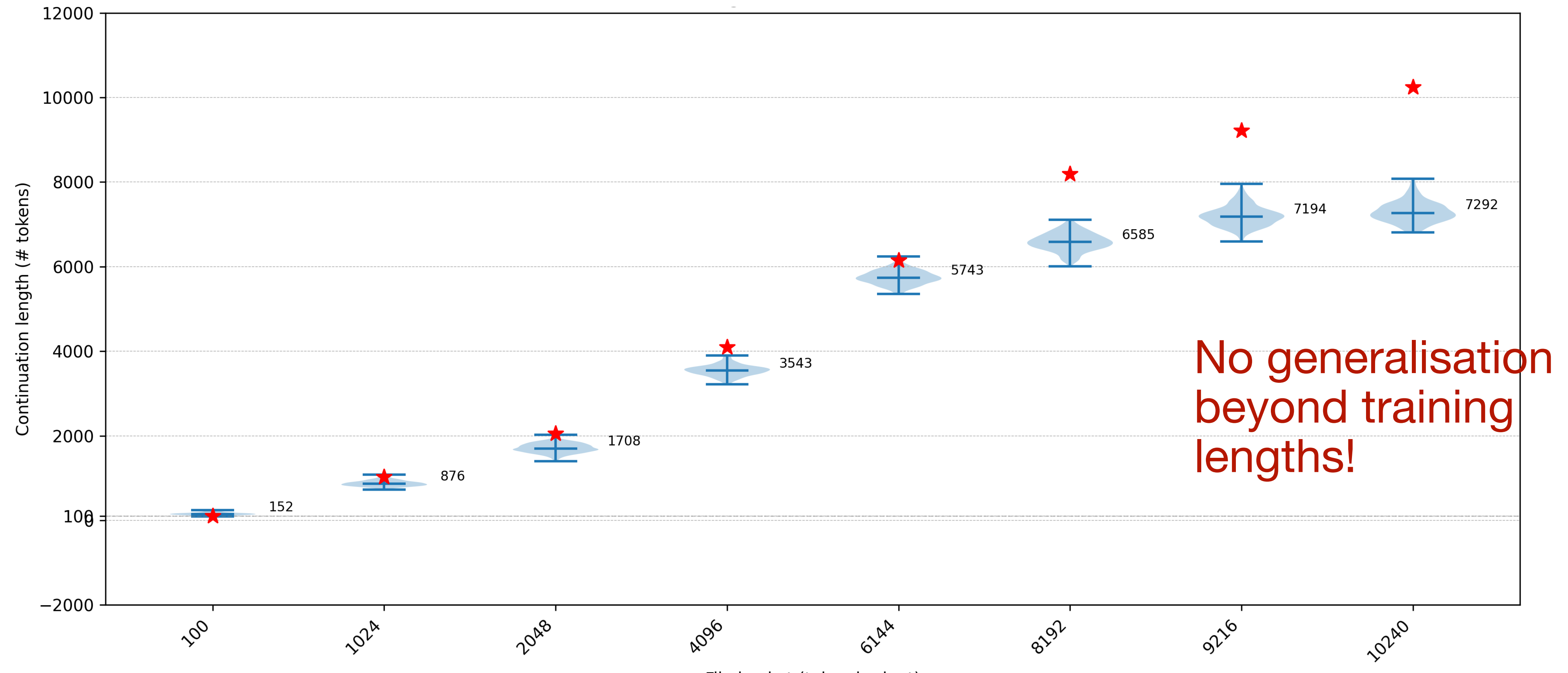
Dealing with LM sequences

Length penalties can also be used to control LM generation length, but these aren't super widely applied in the open?



Dealing with LM sequences

Length penalties can also be used to control LM generation length, but these aren't super widely applied in the open?



Adapting for LMs

1. LM inference is slow. How can we speed things up? 
2. How do we deal with long sequences? 
3. How do we deal with zero-advantage groups (GRPO)?

Dealing with never getting it right.

With GRPO, we might get all advantages in a group being 0 if they get the same score!

	G	R	A
prompt (x)	y_1	1	0
	y_2	1	0
	y_3	1	0

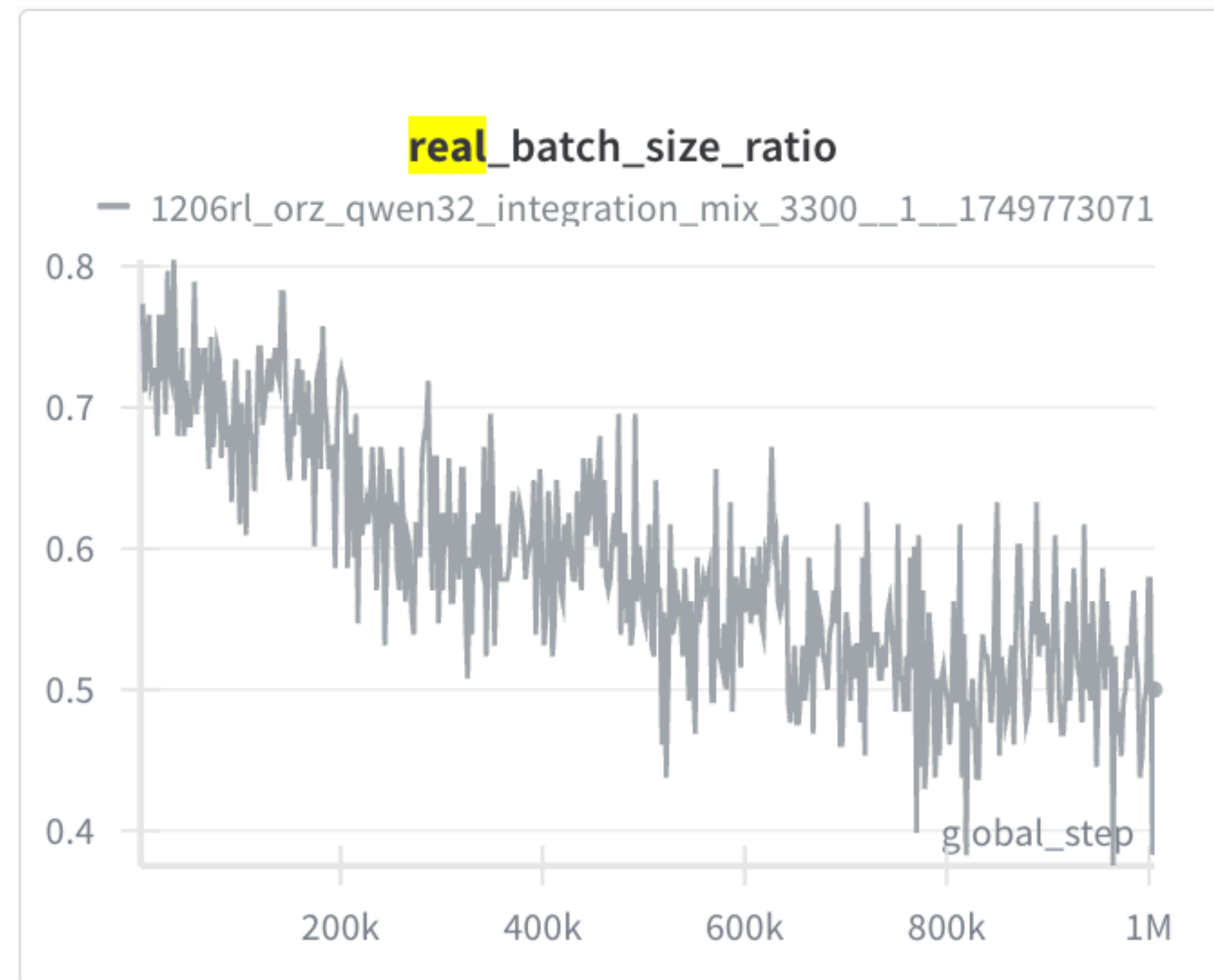
$$\mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot|x)} \left[\frac{1}{B} \sum \frac{1}{T} \sum_{y_i} \left[\min \left(r_i(\theta) \hat{A}_i, \text{clip}(r_i(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_i \right) \right] \right]$$

$$A_j = \frac{R(x, y_j) - \text{mean}(R_G)}{\text{std}(R_G)}$$

Solution: just throw these samples out!

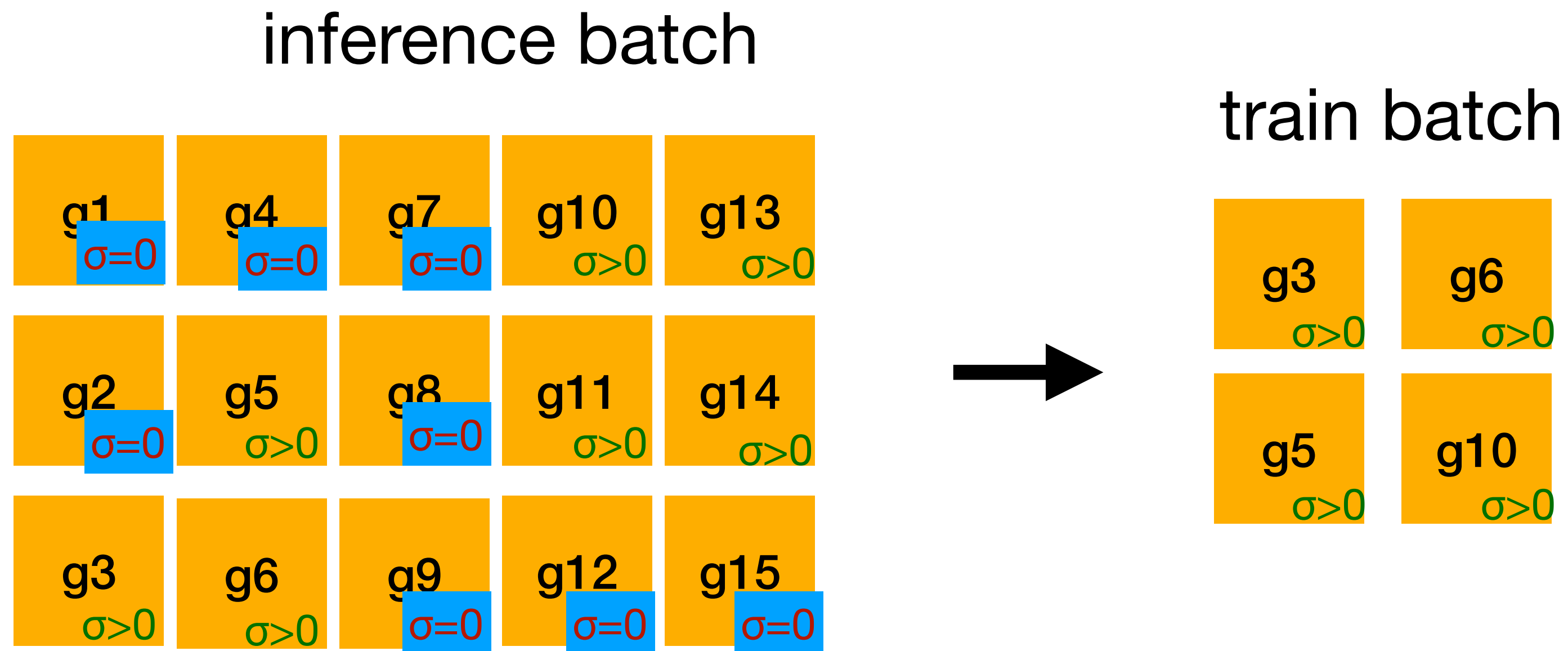
Throwing samples out has issues...

If we throw these samples out, we get a reduced batch size over time...



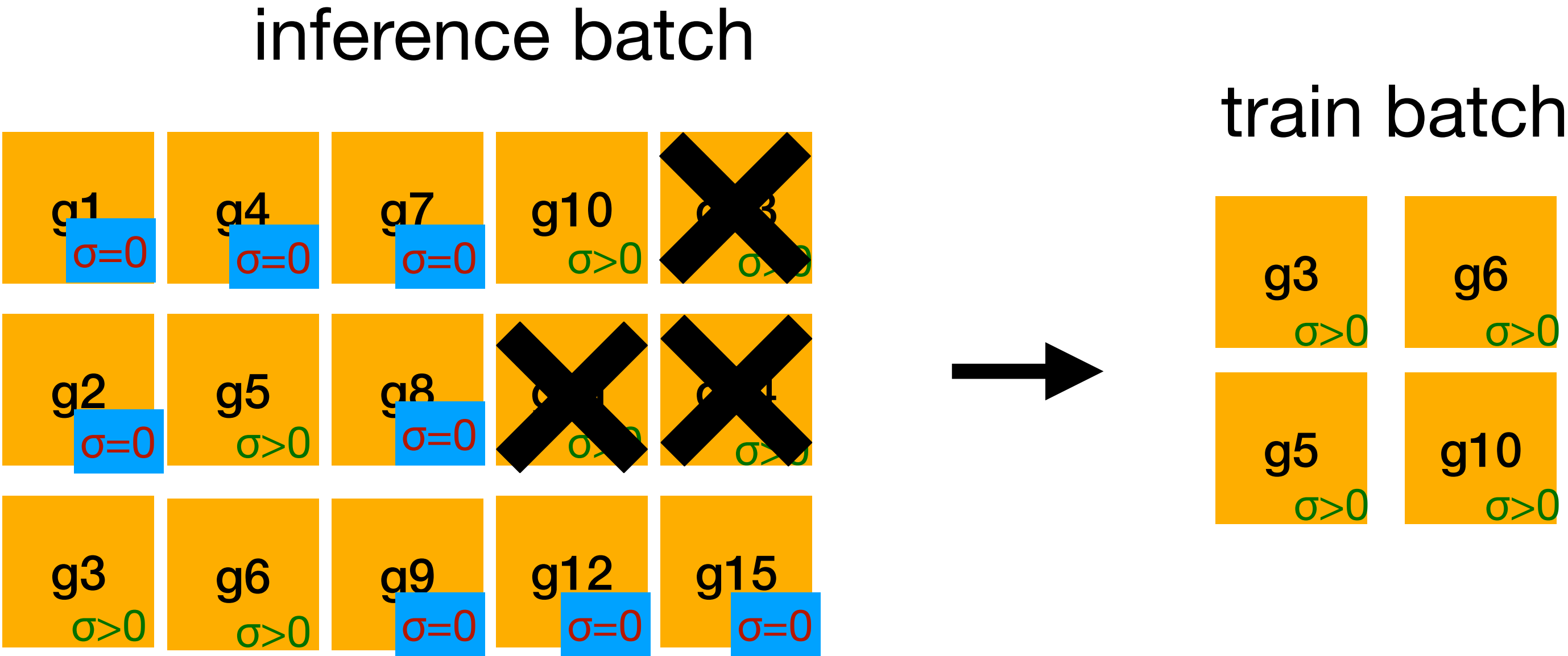
Pulling new samples.

Fix 1: Generate more samples than we need, and downsample to form a batch (“dynamic sampling”, DAPO)



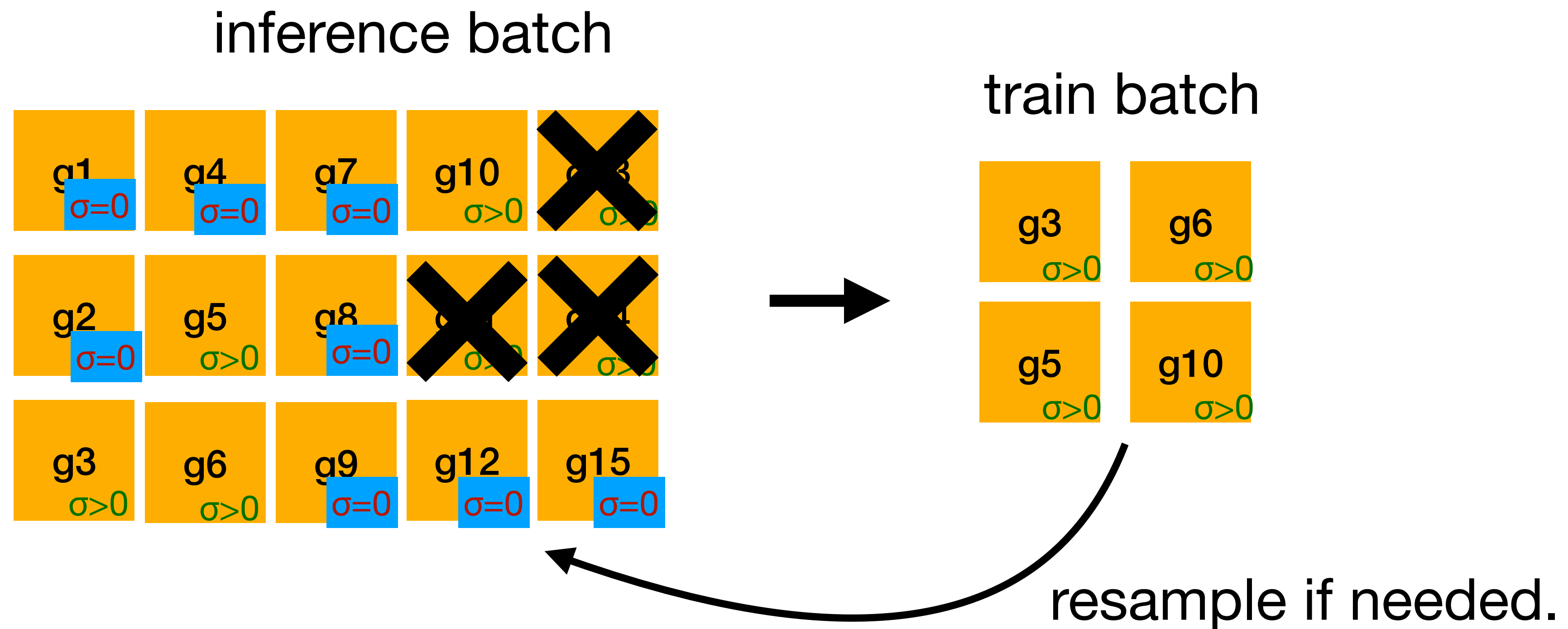
Pulling new samples.

Fix 1: Generate more samples than we need, and downsample to form a batch (“dynamic sampling”, DAPO)



Pulling new samples.

Fix 1: Generate more samples than we need, and downsample to form a batch (“dynamic sampling”, DAPO)



Pulling new samples.

Fix 1: Generate more samples than we need, and downsample to form a batch (“dynamic sampling”, DAPO)

Here, `over_sampling_batch_size` needs to be greater than `rollout_batch_size`. For example:

```
--rollout-batch-size 32 \  
--n-samples-per-prompt 8 \  
--over-sampling-batch-size 64 \  

```

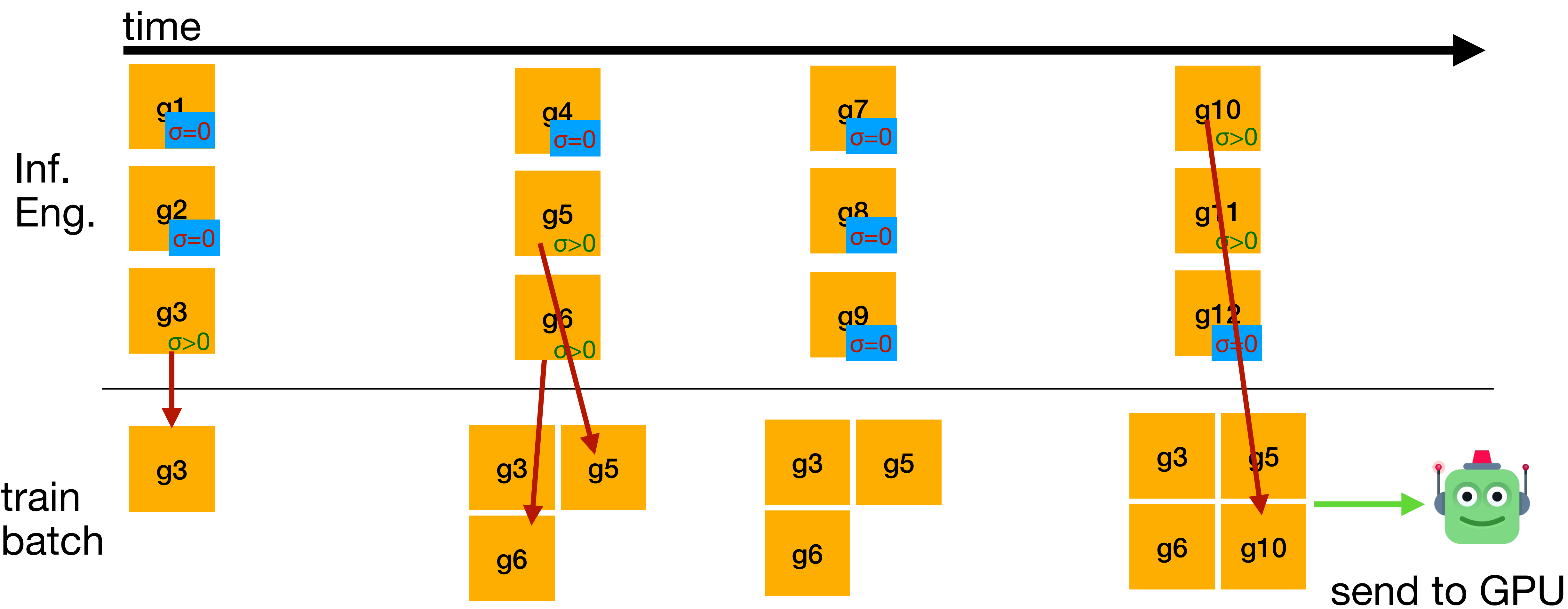
source: <https://thudm.github.io/slme/examples/glm4-9B.html>

Pulling new samples.

Fix 2: Actively pull samples from inference engine on demand until we fill our batch (“active refill”, Olmo 3 paper)

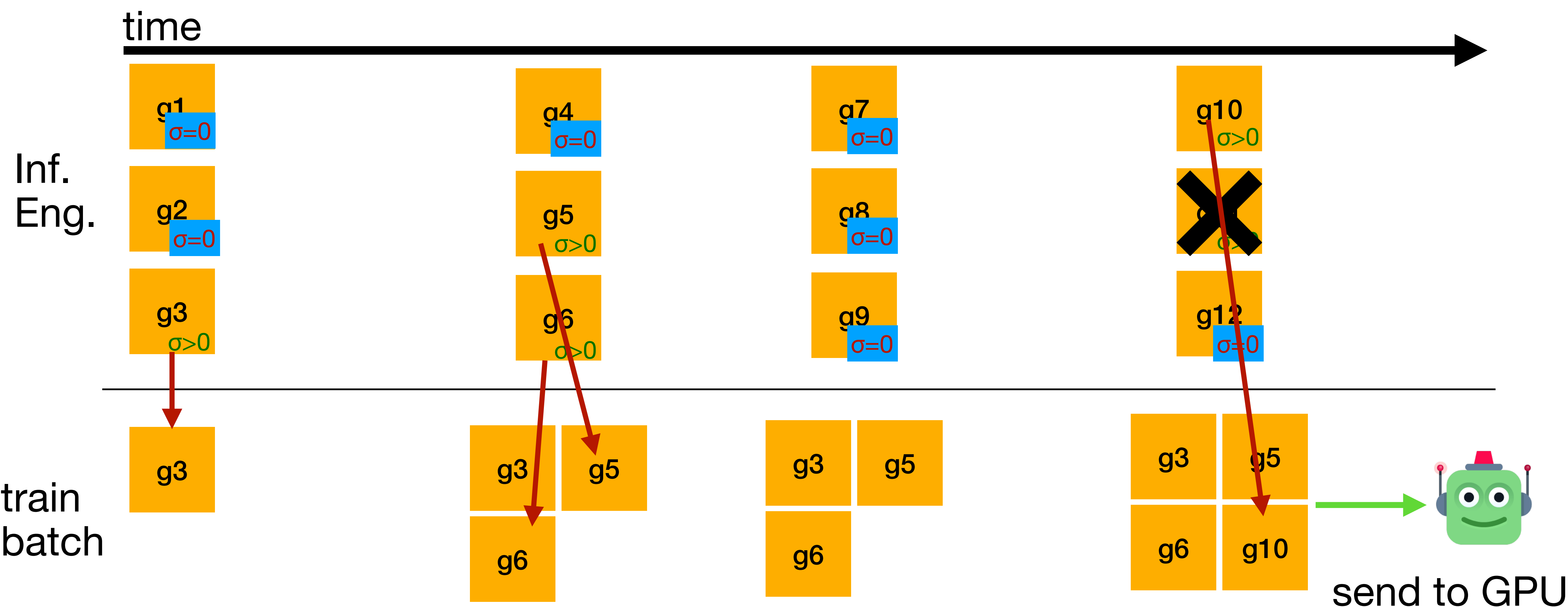
Pulling new samples.

Fix 2: Actively pull samples from inference engine on demand until we fill our batch (“active refill”, Olmo 3 paper)



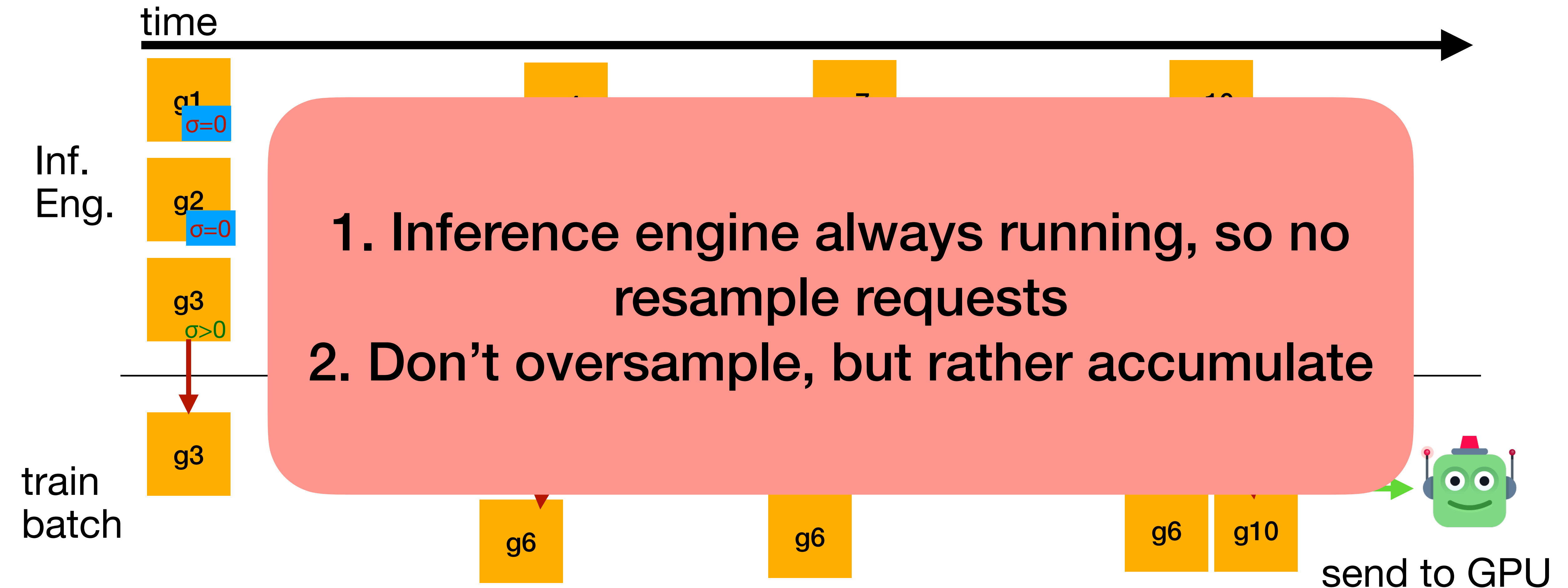
Pulling new samples.

Fix 2: Actively pull samples from inference engine on demand until we fill our batch (“active refill”, Olmo 3 paper)



Pulling new samples.

Fix 2: Actively pull samples from inference engine on demand until we fill our batch (“active refill”, Olmo 3 paper)



Adapting for LMs

1. LM inference is slow. How can we speed things up? ✓
2. How do we deal with long sequences? ✓
3. How do we deal with zero-advantage groups (GRPO)? ✓

Putting it all together.

You now know everything!

Many modern papers are just a combination of the things I have just covered.

DAPO

- use clip-higher in importance clipping
- filter out too long generations
- oversample prompts to avoid small batch
- token-level loss



DAPO: An Open-Source LLM Reinforcement Learning System at Scale

¹ByteDance Seed ²Institute for AI Industry Research (AIR), Tsinghua University

³The University of Hong Kong

⁴SIA-Lab of Tsinghua AIR and ByteDance Seed

Full author list in Contributions



You now know everything!

Many modern papers are just a combination of the things I have just covered.

Dr GRPO

- use sum-and-constant loss
- remove std dev from GRPO advantage

Understanding R1-Zero-Like Training: A Critical Perspective

Zichen Liu^{*+1,2}, Changyu Chen^{*1,3}, Wenjun Li^{*3}, Penghui Qi^{*1,2},
Tianyu Pang¹, Chao Du¹, Wee Sun Lee², Min Lin¹

¹Sea AI Lab

²National University of Singapore

³Singapore Management University



You now know everything!

Many modern papers are just a combination of the things I have just covered.

Minimax Paper

- Clip importance ratio only (CISPO)
- Clip-higher
- FP32 precision for logprob mismatch

 MINIMAX

MiniMax-M1: Scaling Test-Time Compute Efficiently with Lightning Attention

MiniMax¹



How much does all this matter? ScaleRL knows!

Khatri et al., 2025

Read this, and you pretty much know most of what you need to know these days.

The Art of Scaling Reinforcement Learning Compute for LLMs

Devvrit Khatri^{2,*†}, Lovish Madaan^{1,3,*},
Rishabh Tiwari^{4†}, Rachit Bansal^{5†}, Sai Surya Duvvuri^{2†}, Manzil Zaheer^{1†},
Inderjit S. Dhillon², David Brandfonbrener¹, Rishabh Agarwal^{6,†}

¹Meta, ²UT Austin, ³UCL, ⁴UC Berkeley, ⁵Harvard University, ⁶Periodic Labs

*Equal contribution, [†]Work done at Meta

Practical example: DR Tulu



DR Tulu

Context: Pelosi, the first and only female House Speaker, has served her San Francisco district for decades and capped a long career with this decision. She described it as the end of a decades-long career and highlighted her role leading the House.

Searched: 10 links | Browsed: 4 links | Tool calls: 6 | Failed: 1

You: exit

Goodbye!

✂ Stopping Search vLLM server...

^[A

✂ Stopping MCP server...

root@saturn-cs-aus-246:/weka/oe-adapt-default/hamishi/dr-tulu/agent# uv run --extra vllm python scripts/launch_chat.py --model rl-research/DR-Tulu-8B

=== Interactive Chat Launcher ===

⚠ MCP server is not running on port 8000

Launch MCP server? (y/n): y

✂ Launching MCP server on port 8000...

📁 MCP server output will be logged to /tmp/mcp_server_8000.log

⌚ Waiting for MCP server to start...

⚠ MCP server process started but port check failed. Continuing anyway...

⚠ vLLM server does not appear to be accessible at http://localhost:30004/v1

✂ Auto-launching Search Agent vLLM server: rl-research/DR-Tulu-8B on port 30004 (GPU 0)...

✂ Launching vLLM server for model rl-research/DR-Tulu-8B on port 30004...

📁 vLLM output for rl-research/DR-Tulu-8B will be logged to /tmp/vllm_server_30004.log

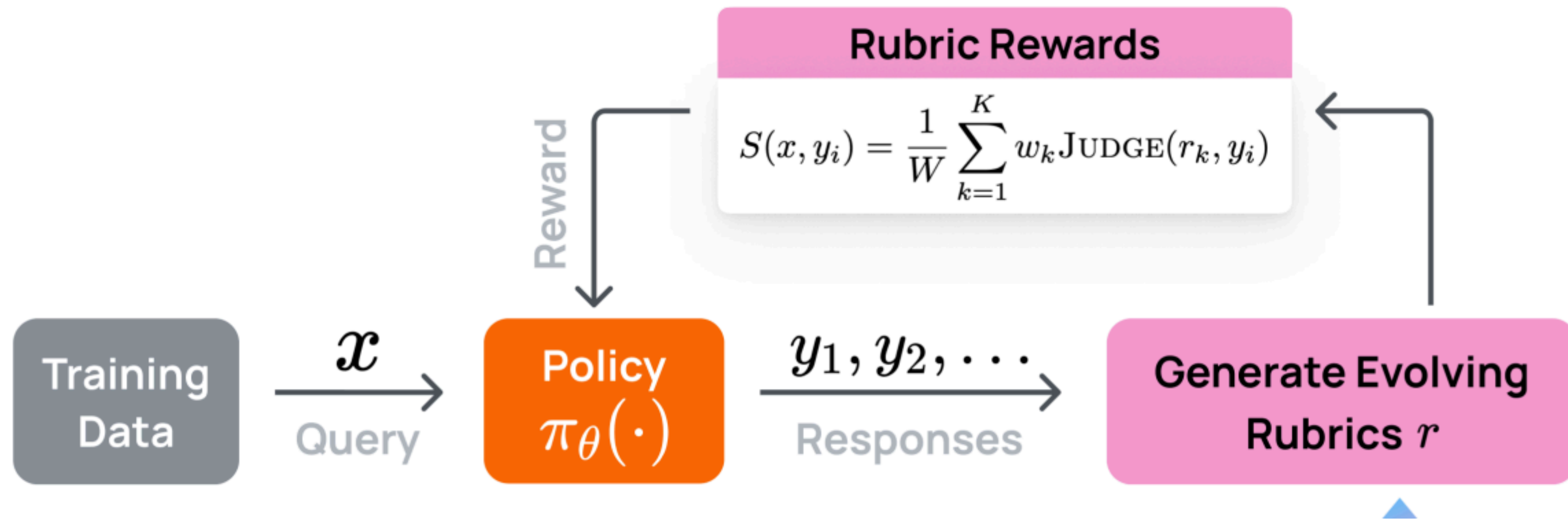
⌚ Waiting for vLLM server to become ready (this may take a few minutes)...

⌚ Still waiting for vLLM server (30s)...

✓ vLLM server started (PID: 15462)

✂ Starting interactive chat...

Practical example: DR Tulu



Practical example: DR Tulu

User query

Describe how genetically engineering T cells could be used as an anti-inflammatory therapy for IBD by changing cytokine signaling between T cells and macrophages.

📄 ↓ Generate rubrics with search

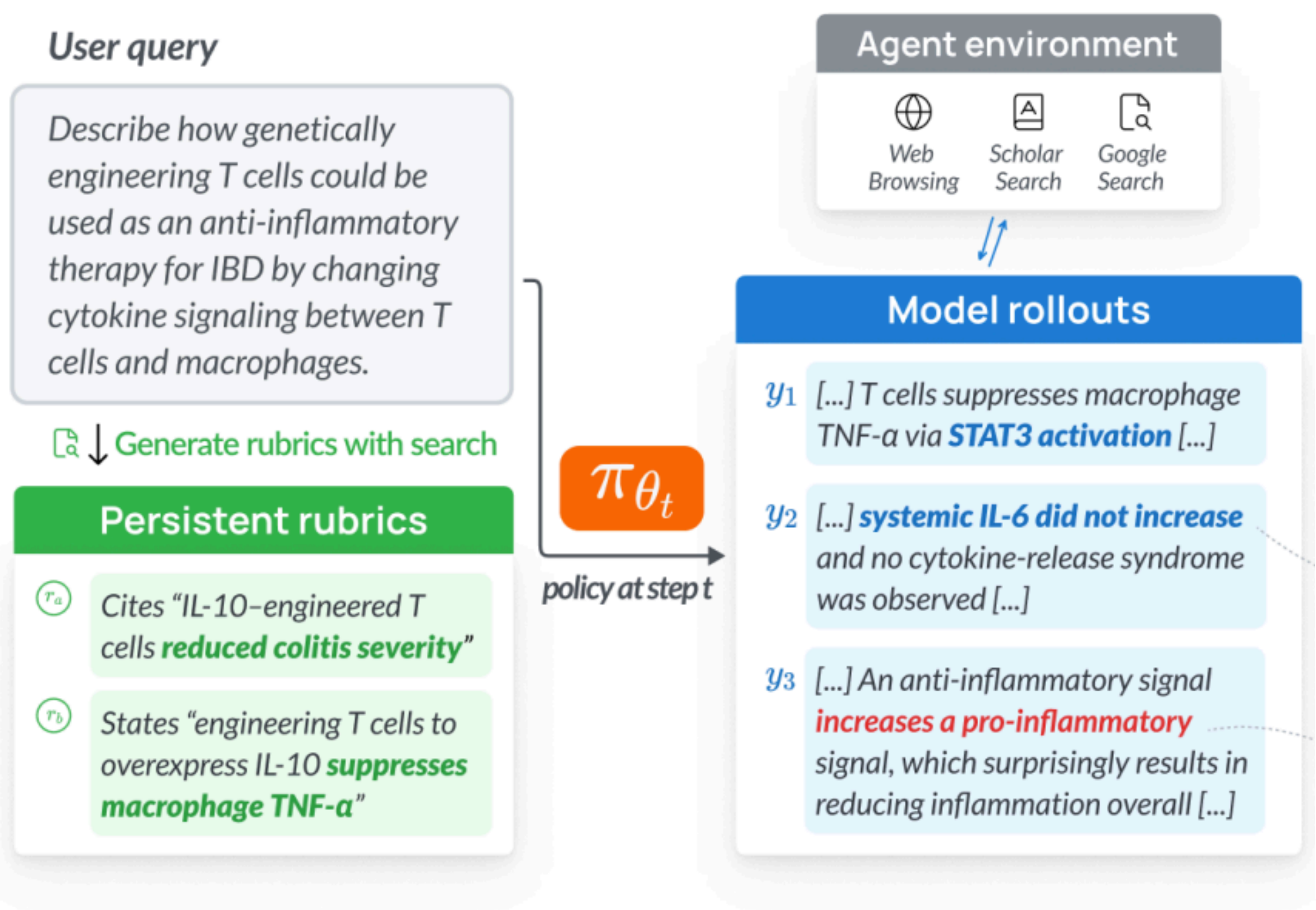
Persistent rubrics

- r_a Cites “IL-10-engineered T cells **reduced colitis severity**”
- r_b States “engineering T cells to overexpress IL-10 **suppresses macrophage TNF- α** ”

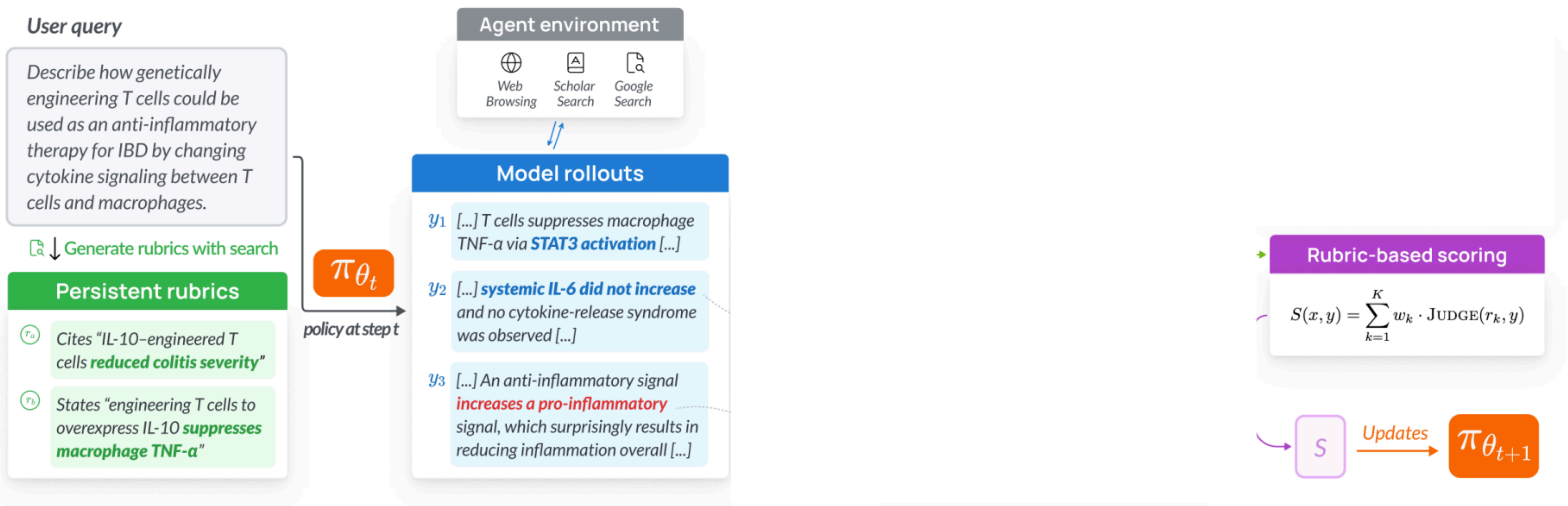
π_{θ_t}

policy at step t

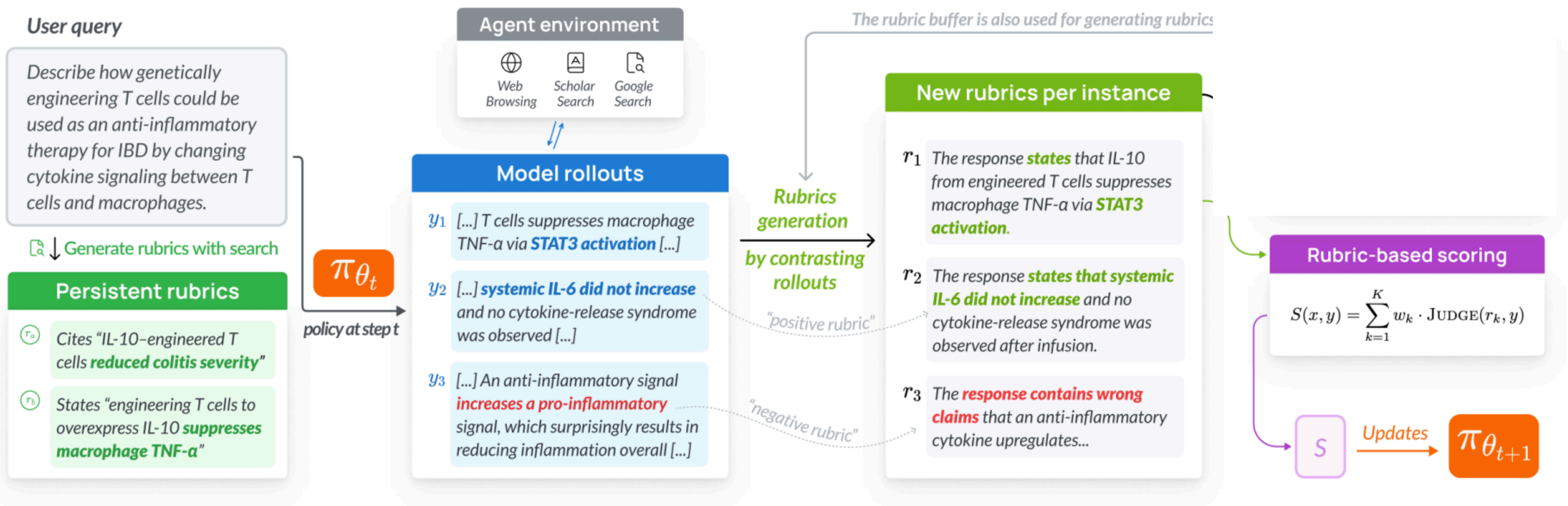
Practical example: DR Tulu



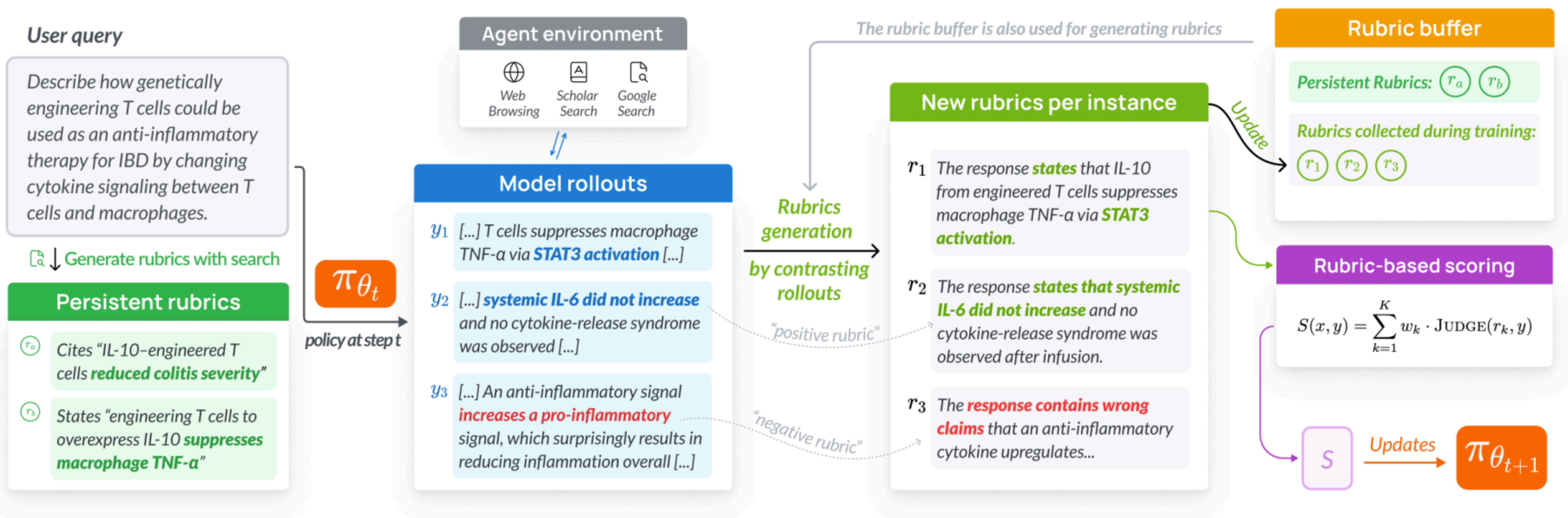
Practical example: DR Tulu



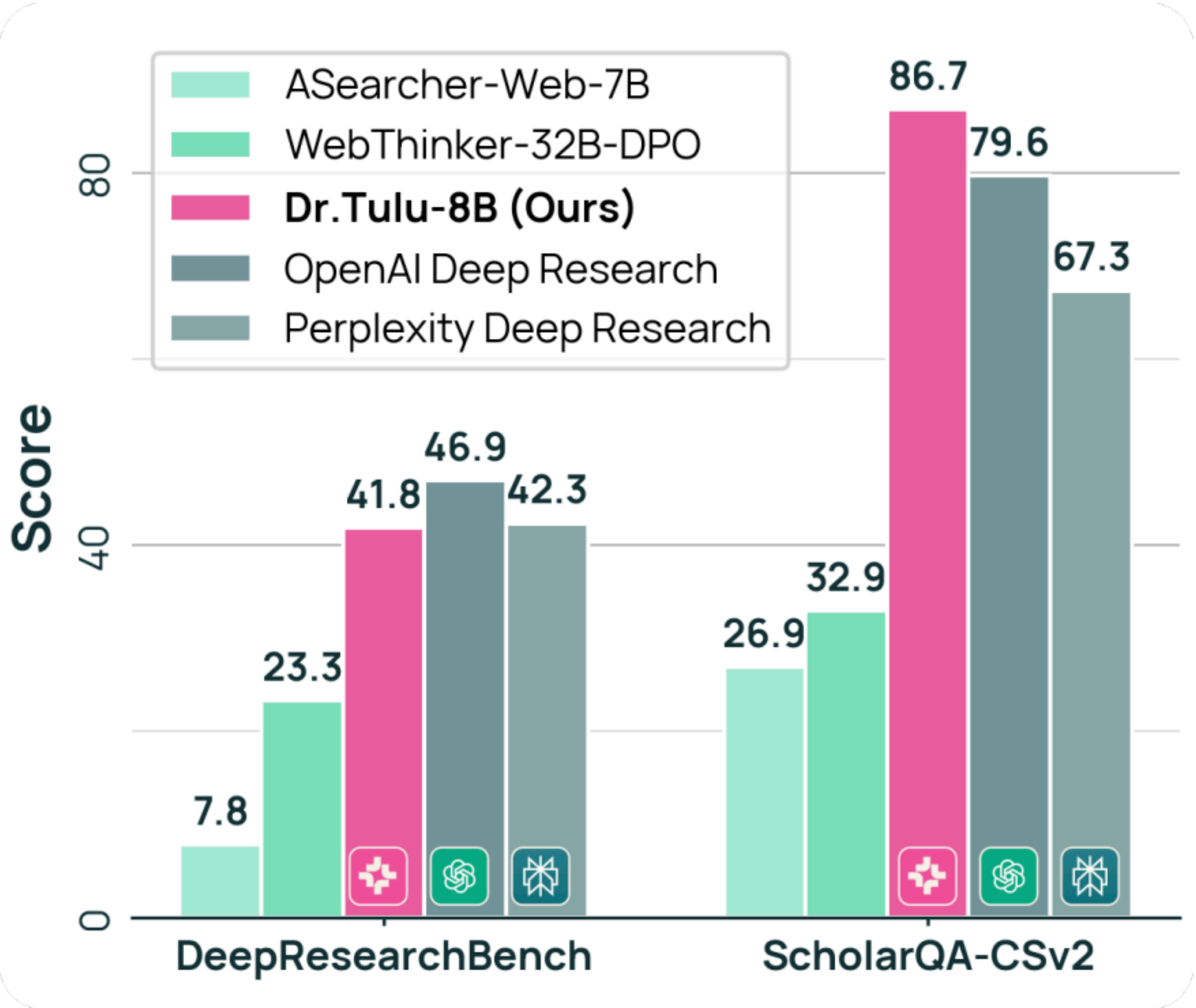
Practical example: DR Tulu



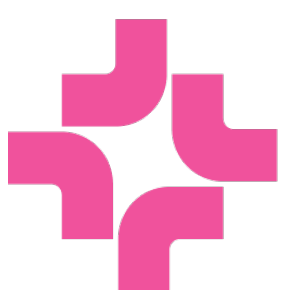
Practical example: DR Tulu



Practical example: DR Tulu



Practical example:

 Olmo 3

Practical example:

 Olmo 3

Announcing Olmo 3, a leading fully open LM suite built for reasoning, chat, & tool use, and an open model flow—not just the final weights, but the entire training journey.

Best fully open 32B reasoning model & best 32B base model. 

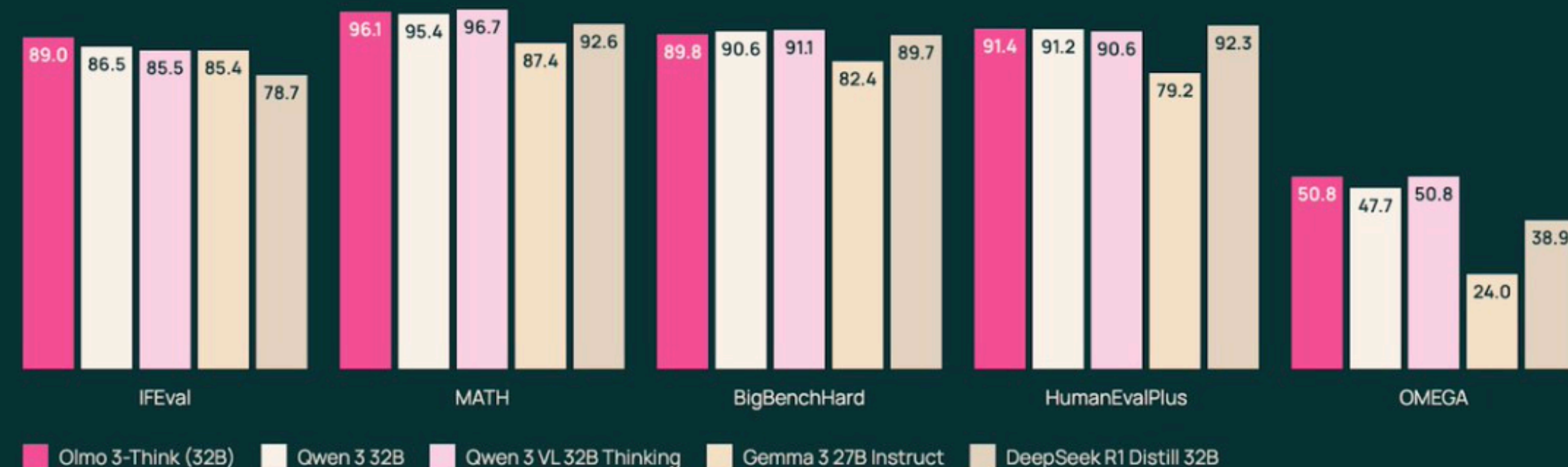
Olmo 3-Base (32B) vs open peers

Benchmarks where Olmo 3-Base (32B) has the highest score among these large open models (higher is better). See more in our technical report.



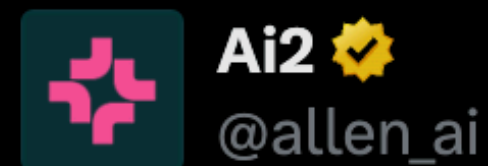
Olmo 3-Think (32B) vs open peers

Selected math, reasoning, coding, and QA benchmarks comparing Olmo 3-Think (32B) to strong open models of similar scale (higher is better). See more in our technical report.



Practical example:

Olmo 3



Announcing Olmo 3, a leading fully open LM suite built for reasoning, chat, & tool use, and an open model flow—not just the final weights, but the entire training journey.

Best fully open 32B reasoning model & best 32B base model. 🧵

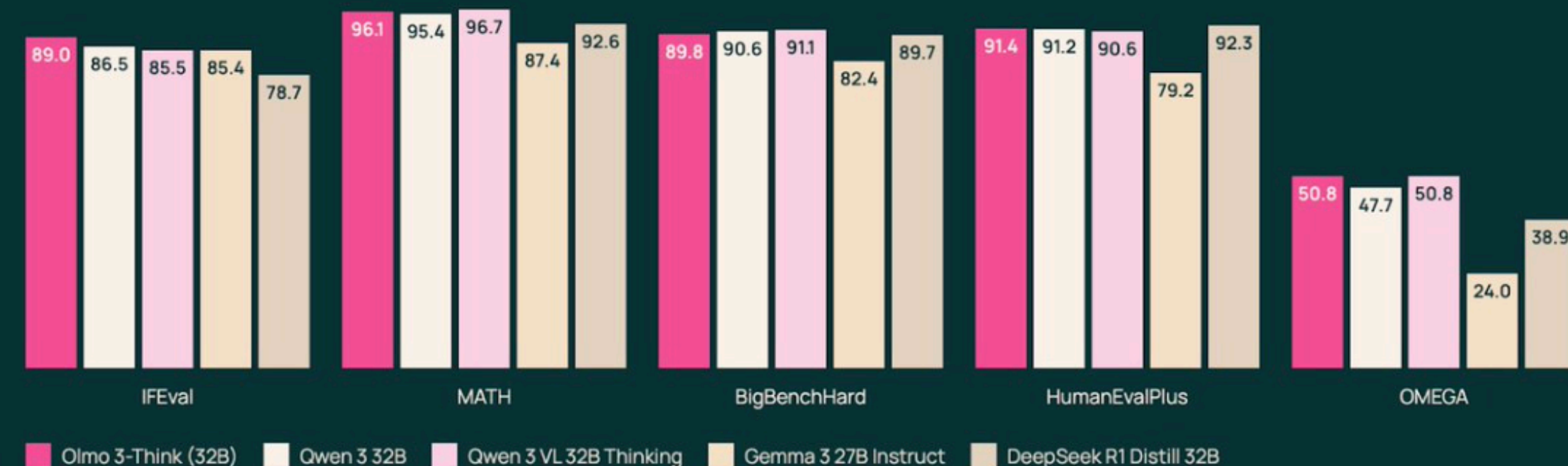
Olmo 3-Base (32B) vs open peers

Benchmarks where Olmo 3-Base (32B) has the highest score among these large open models (higher is better). See more in our technical report.

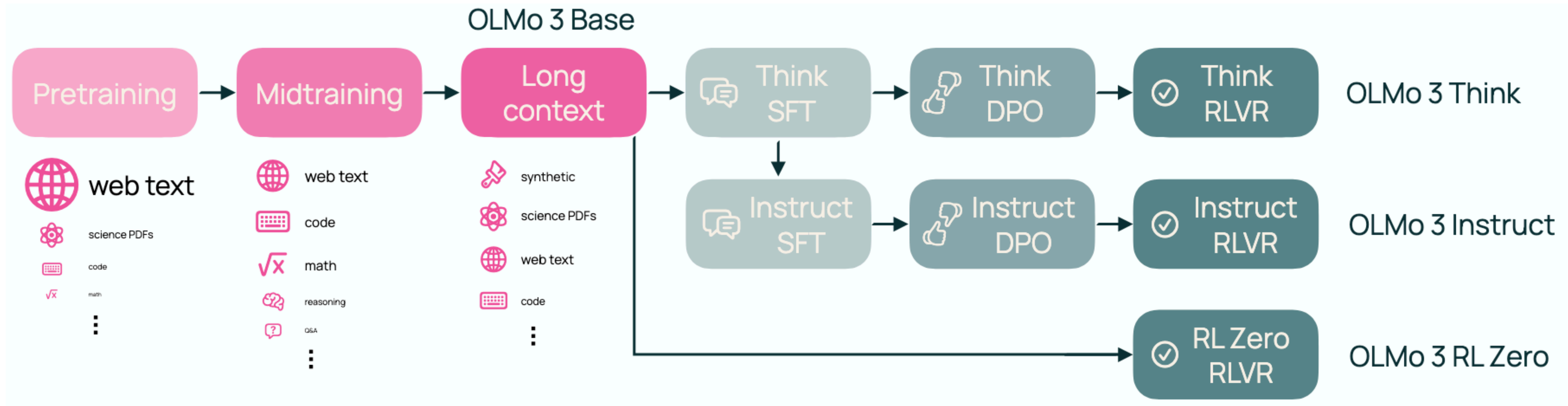


Olmo 3-Think (32B) vs open peers

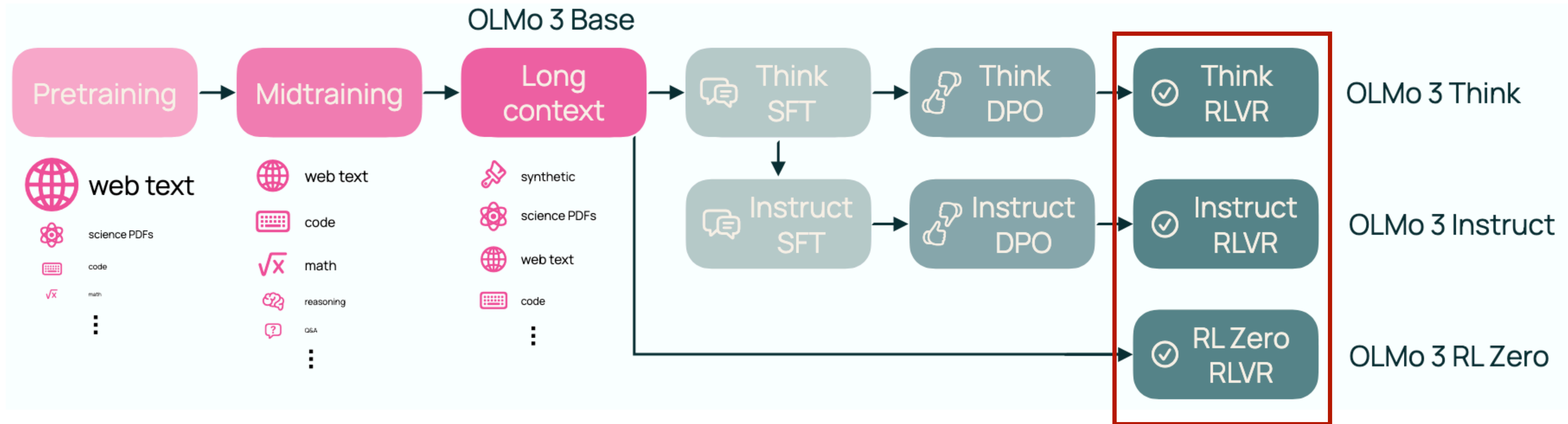
Selected math, reasoning, coding, and QA benchmarks comparing Olmo 3-Think (32B) to strong open models of similar scale (higher is better). See more in our technical report.



Practical example: Olmo 3

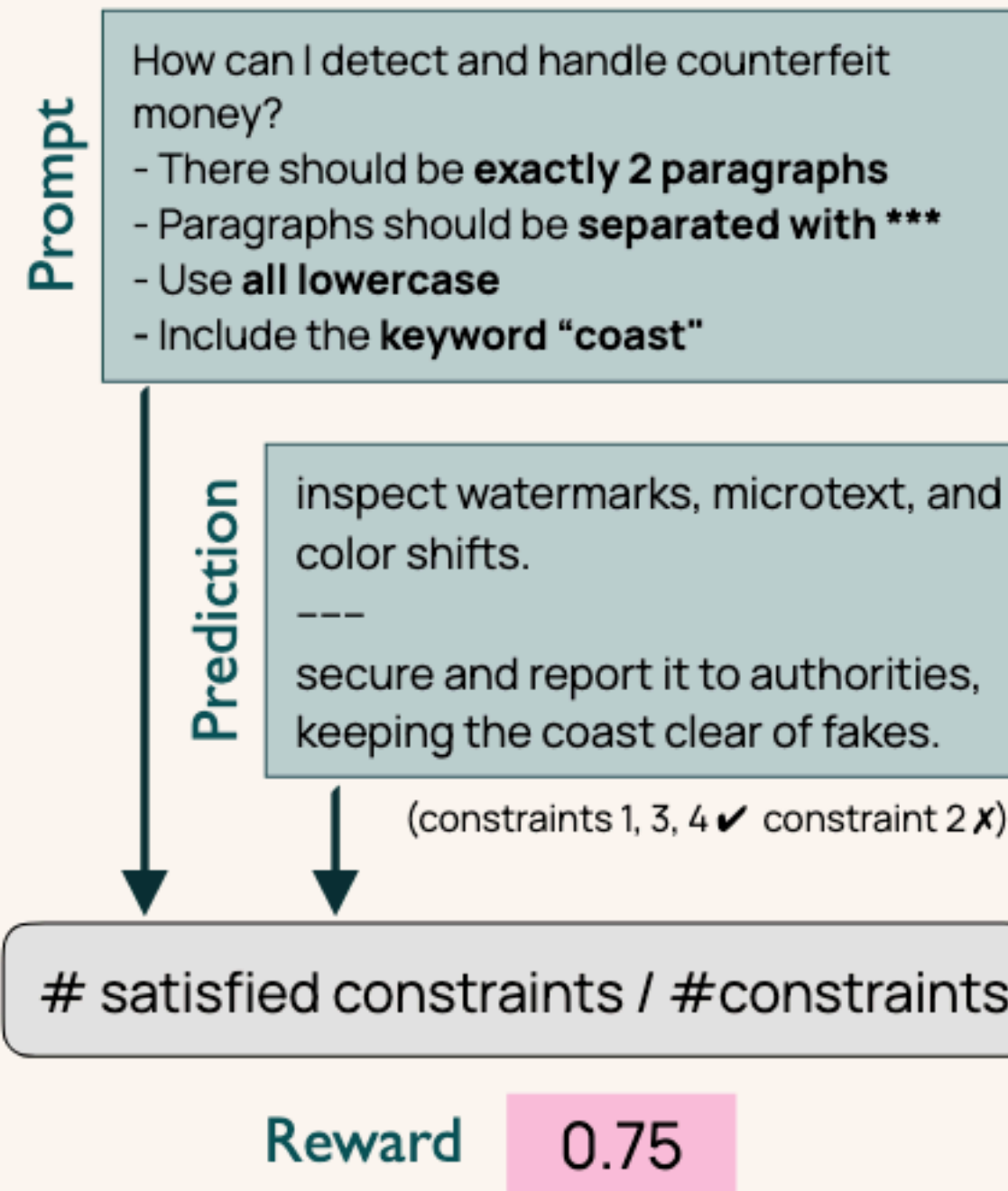


Practical example: Olmo 3

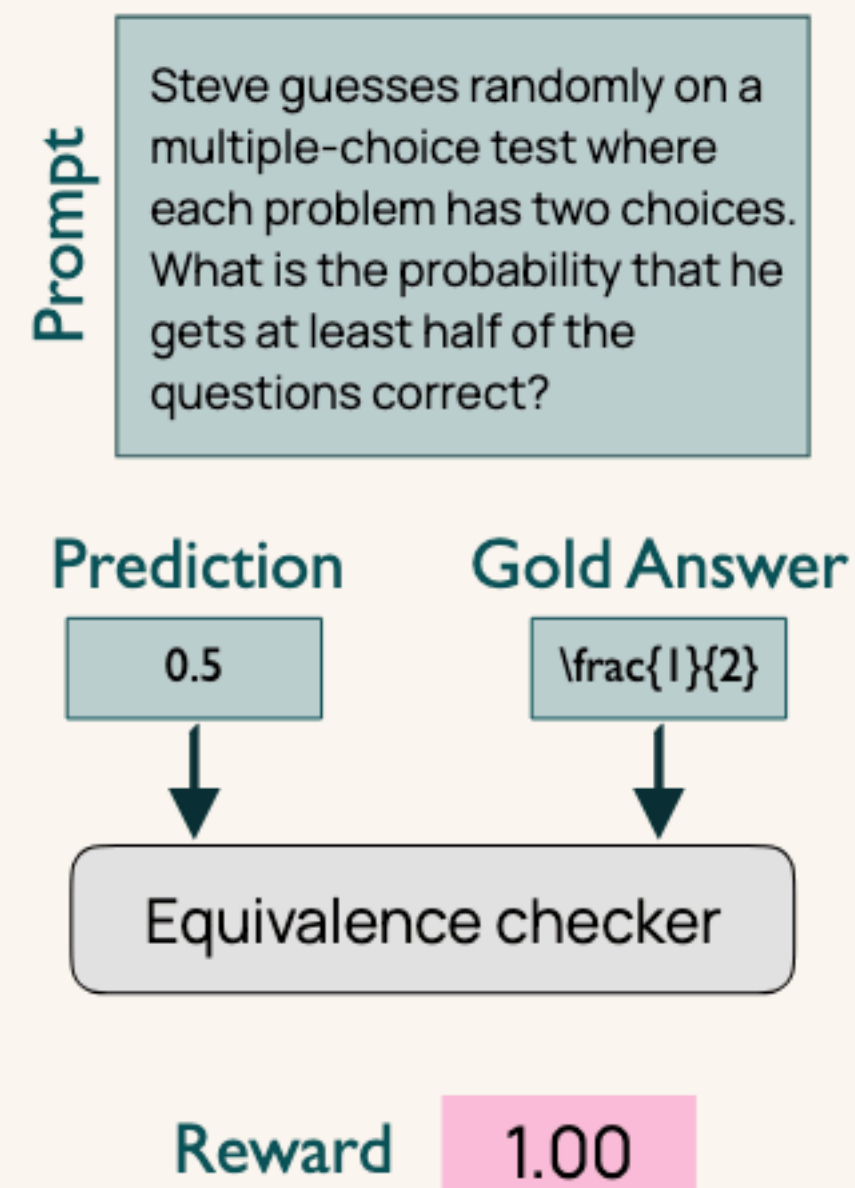


Practical example: Olmo 3

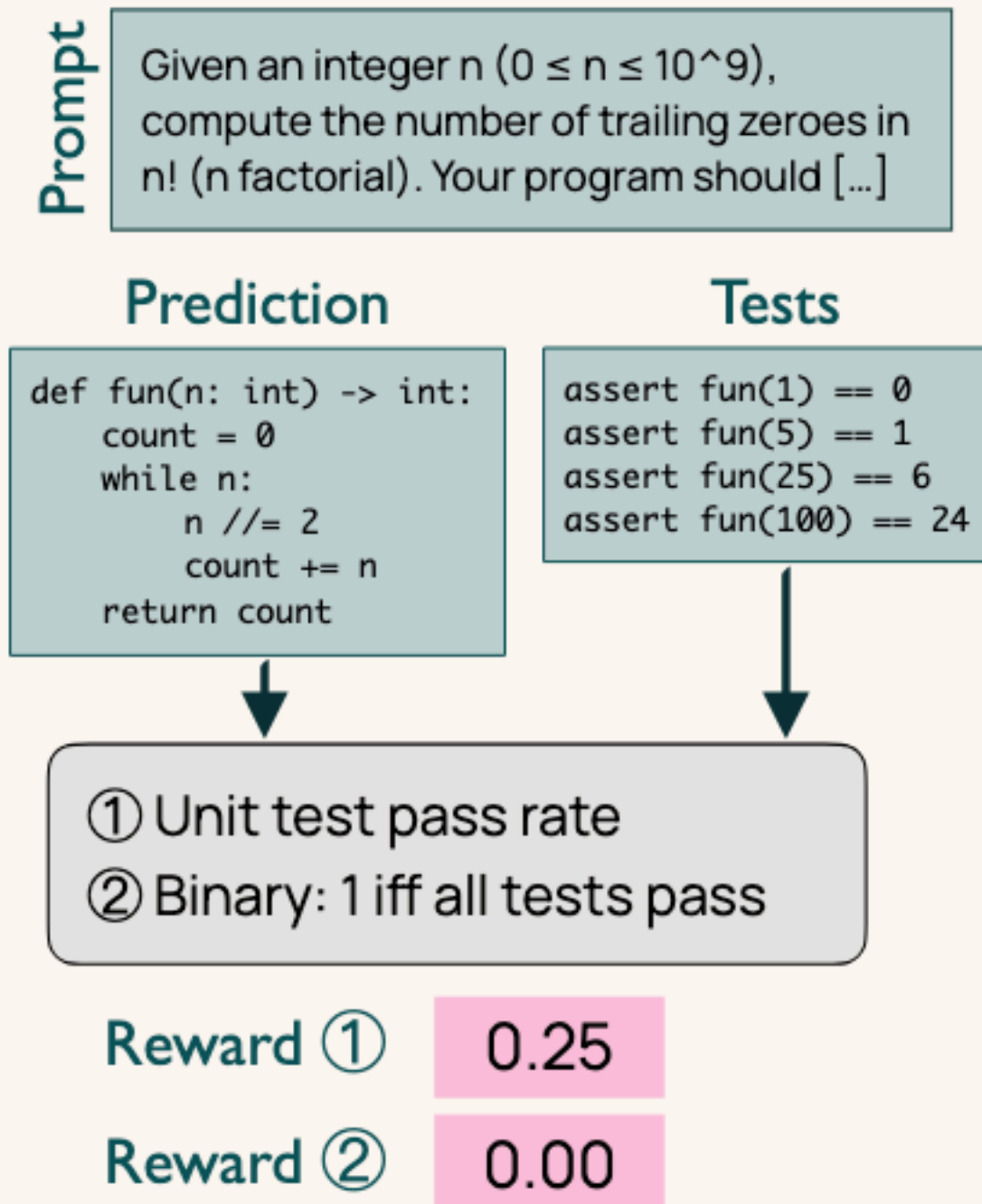
Instruction Following



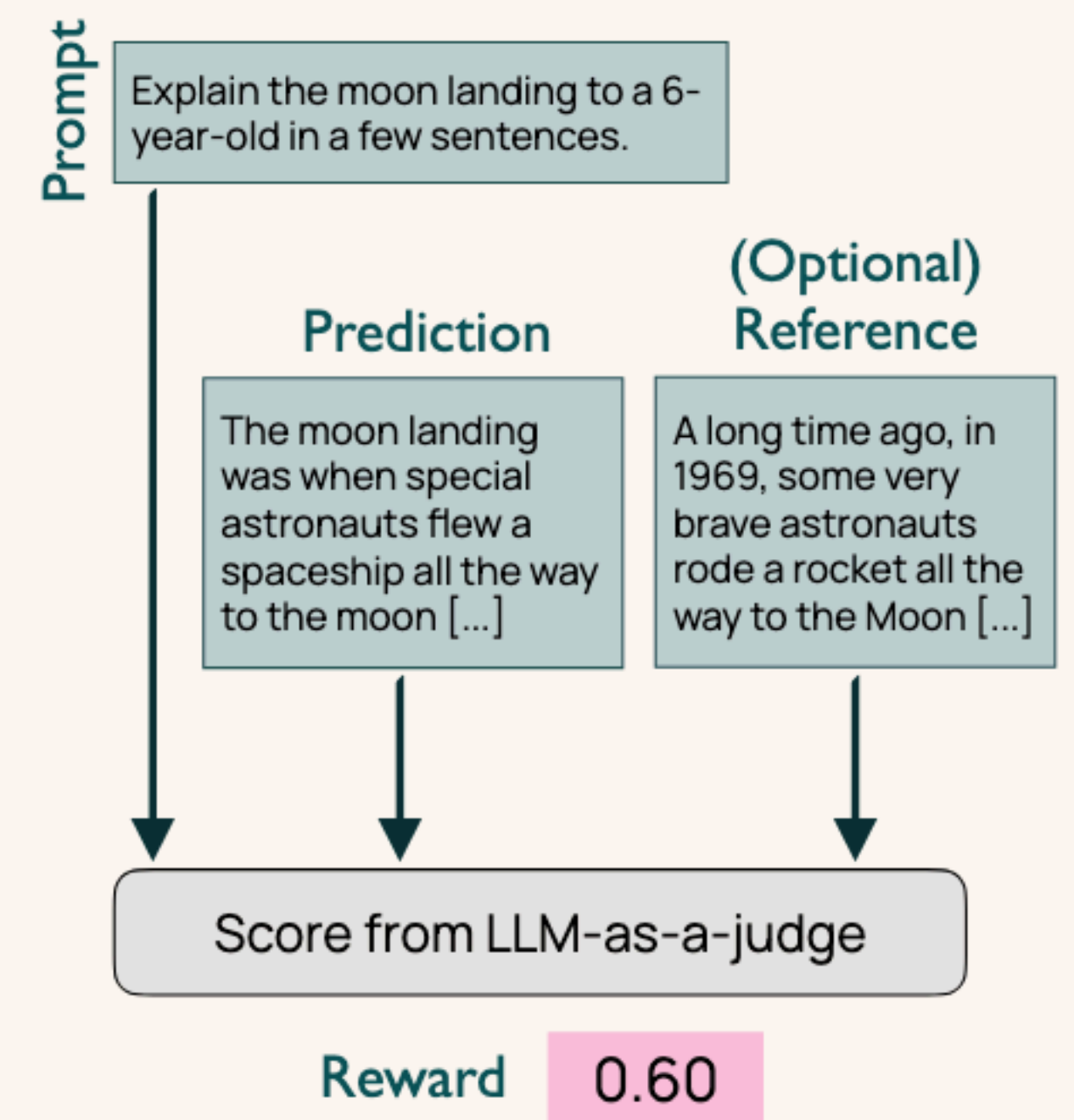
Math



Coding



General Chat



Verifiable Tasks

Non-Verifiable Tasks

Practical example: Olmo 3

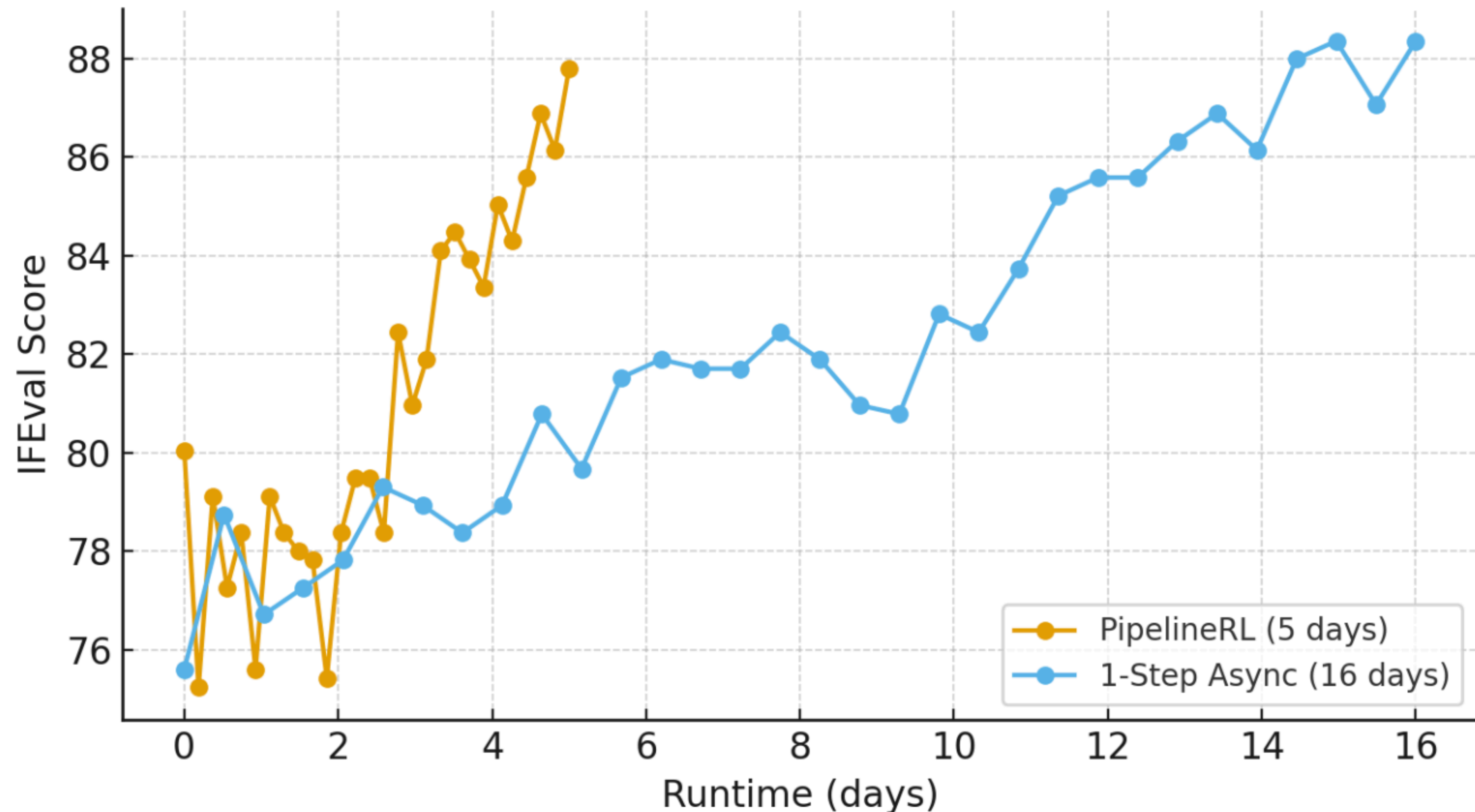
We make the following improvements over vanilla GRPO:²²

- **Zero Gradient Signal Filtering:** We remove groups of instances whose rewards are all identical (i.e., a batch with zero standard deviation in their advantage) to avoid training on samples that provide zero gradient, similar to DAPO (Yu et al., 2025).
- **Active Sampling:** We maintain a consistent batch size in spite of zero gradient filtering with a novel, more efficient version of dynamic sampling (Yu et al., 2025), see OLMORL Infra for details.
- **Token-level loss:** We use a token-level loss to normalize the loss by the total number of tokens across the batch (Yu et al., 2025), rather than per-sample to avoid a length bias.
- **No KL Loss** We remove the KL loss as a common practice (GLM-4.5 Team et al., 2025; Yu et al., 2025; Liu et al., 2025b) as it allows less-restricted policy updates, and removing it does not lead to over-optimization or destabilized training.
- **Clip Higher:** We set the upper-bound clipping term in the loss to a slightly higher value than the lower bound to enable larger updates on tokens, as proposed by Yu et al. (2025).
- **Truncated Importance Sampling:** To adjust for differences between log probabilities from the inference and training engines, we multiply the loss by the truncated importance sampling ratio, following Yao et al. (2025).
- **No standard deviation normalization:** When calculating advantage, we do not normalize by the standard deviation of the group, following Liu et al. (2025b). This removes a difficulty bias, where questions with low standard deviation in their rewards (e.g. too hard or too easy) have their advantages significantly increased by the normalization term.

OlmoRL Formulation Our final objective function includes a token-level loss, truncated importance sampling, clip-higher, and no standard deviation in the advantage calculation:

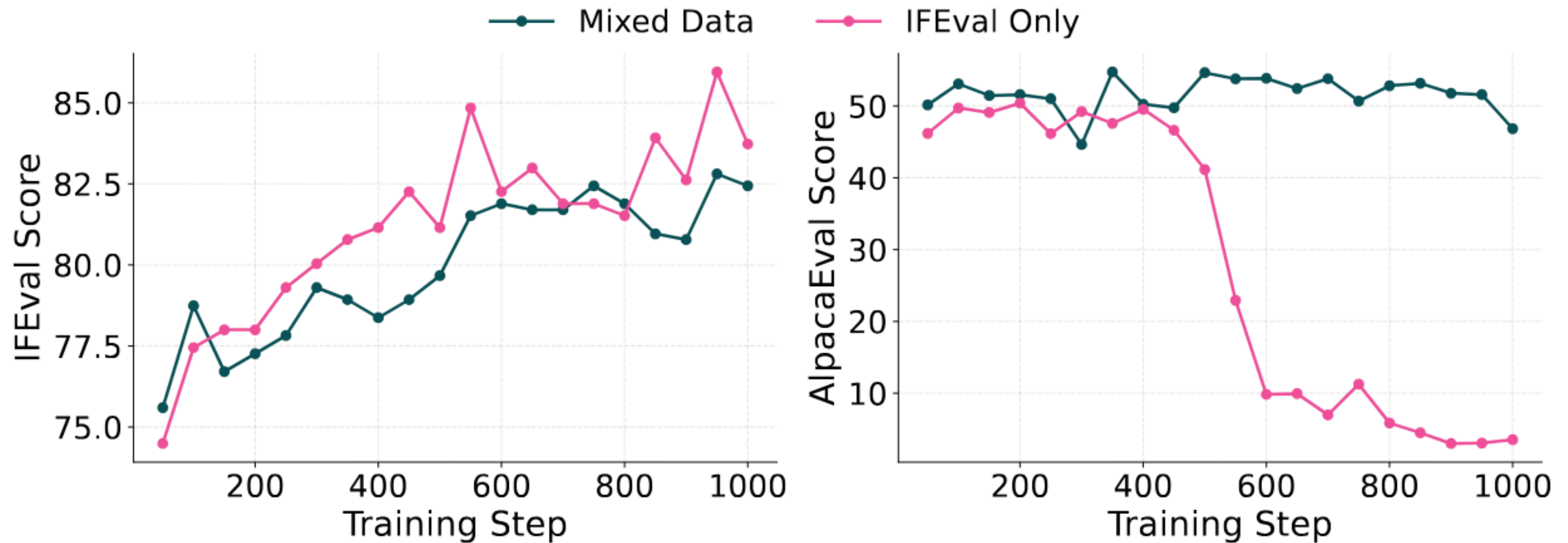
Practical example: Olmo 3

Also used pipelineRL to significantly speed up training

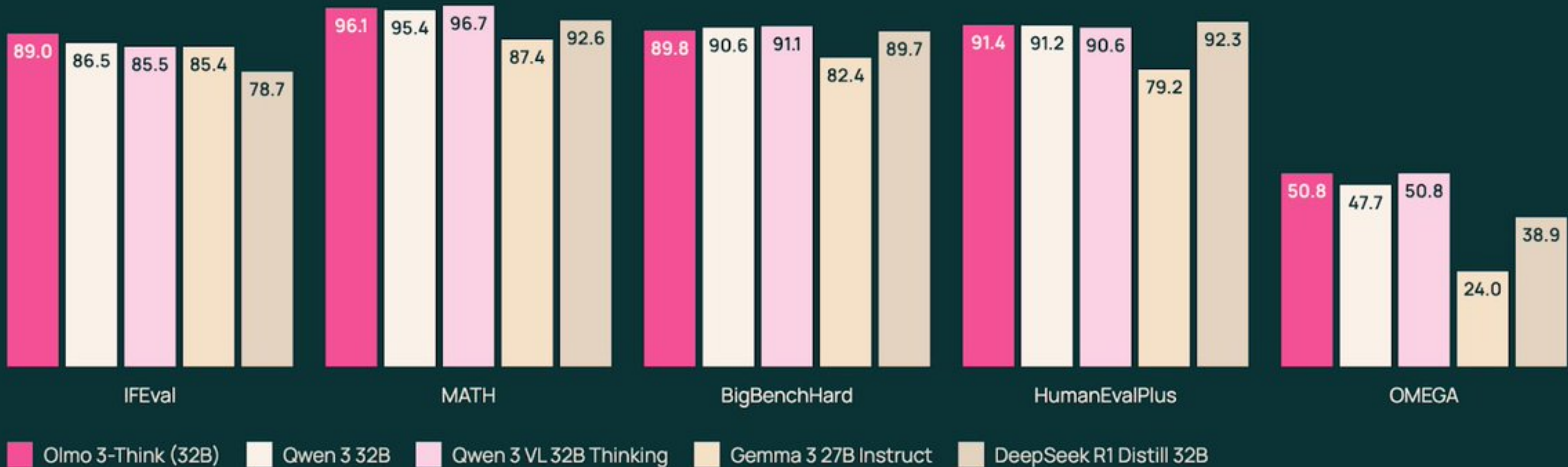


Practical example: Olmo 3

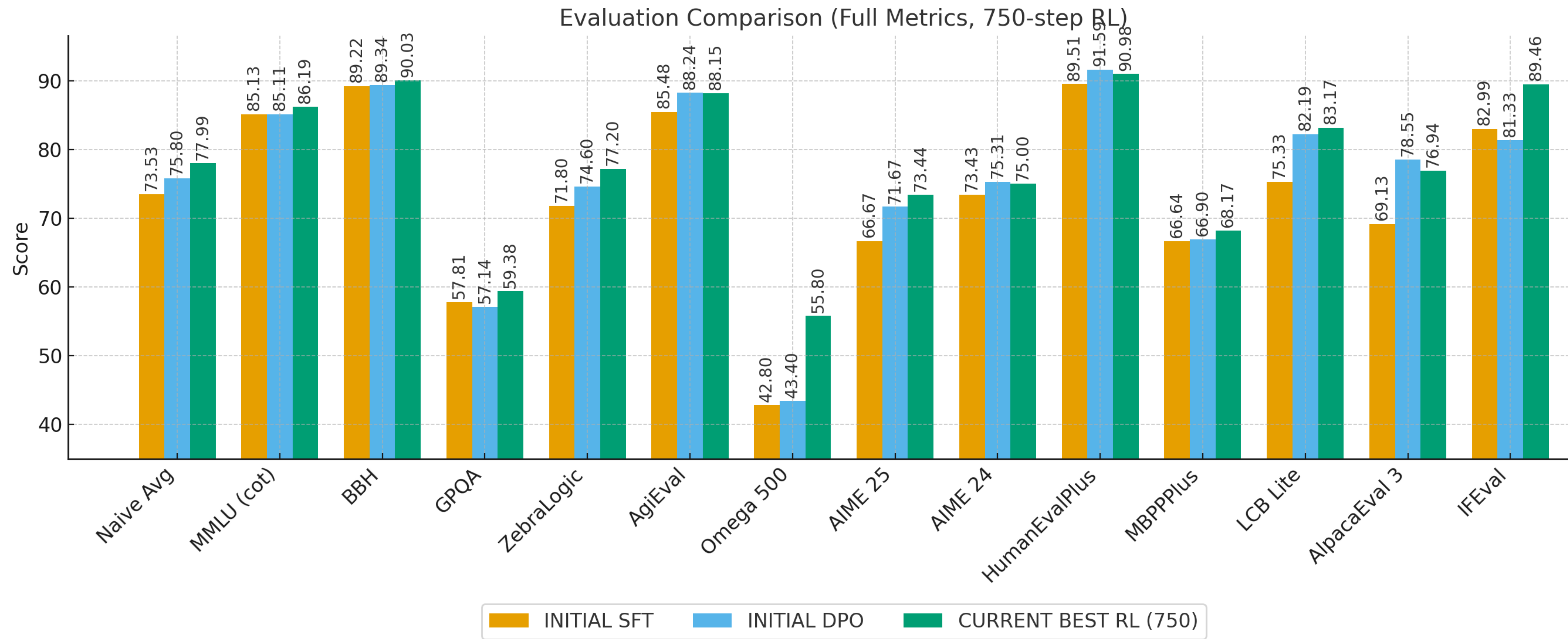
Mixing data prevents overfitting and collapse



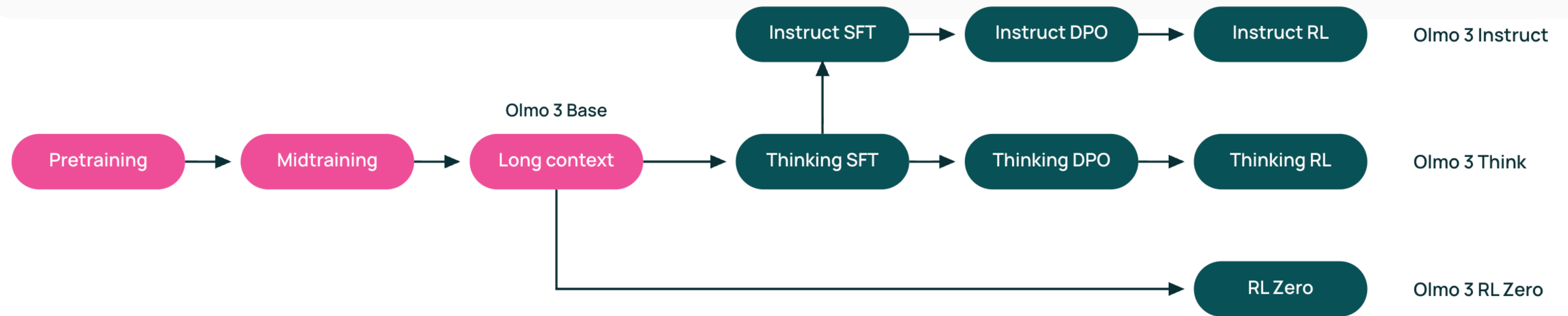
Practical example: Olmo 3



Practical example: Olmo 3



Practical example: Olmo 3



Explore the Model Flow

Click on any stage to learn more about it and download artifacts.

<https://allenai.org/blog/olmo3>

Thanks for listening!

Go forth and train SOTA!

Further reading

- Always start with Sutton & Barto (chapter 13)
- Then OpenAI's spinning up RL course is great (<https://spinningup.openai.com/en/latest/index.html>)
- Costa's 37 implementation details of PPO is amazing (<https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/>)
- ScaleRL (Khatri et al., 2025) is a great overview of recent approaches.



Thanks for listening!

Thanks to the many contributors for this work and slides:

Finbarr Timbers, Nathan Lambert, Teng Xiao, Michael Noukhovitch, Saumya Malik, Scott Geng, Faeze Brahman, Valentina Pyatkin, Saurabh Shah, Costa Huang, Hannaneh Hajishirzi, H2Lab as a whole.



Thanks for listening!

Thanks to the many contributors for this work and slides:

All my code for RL training can be found on github @ open-instruct

Happy to answer further questions etc:

✉ hamishiv@cs.washington.edu

✕ @hamishivi (and most other things)



Thanks for listening!

Citations for the work discussed:

1. Levine S. Supervised Learning of Behaviors (Lecture 2). Published online August 2021.
2. Liu J, Li Y, Fu Y, Wang J, Liu Q, Shen Y. When Speed Kills Stability: Demystifying RL Collapse from the Inference-Training Mismatch. Published online September 2025. <https://yingru.notion.site/When-Speed-Kills-Stability-Demystifying-RL-Collapse-from-the-Inference-Training-Mismatch-271211a558b7808d8b12d403fd15edda>
3. Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge. *Nature*. 2017;550(7676):354-359. doi:10.1038/nature24270
4. Berseth G. Is Exploration or Optimization the Problem for Deep Reinforcement Learning? *arXiv [csLG]*. Published online 2025. <http://arxiv.org/abs/2508.01329>
5. Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. Second. The MIT Press; 2018. <http://incompleteideas.net/book/the-book-2nd.html>
6. Schulman J, Moritz P, Levine S, Jordan M, Abbeel P. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. ; 2016.
7. Shao Z, Wang P, Zhu Q, et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv [csCL]*. Published online 2024. <http://arxiv.org/abs/2402.03300>
8. Ahmadian A, Cremer C, Gallé M, et al. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs. *arXiv [csLG]*. Published online 2024. <http://arxiv.org/abs/2402.14740>
9. Hu J, Liu JK, Xu H, Shen W. REINFORCE++: An Efficient RLHF Algorithm with Robustness to Both Prompt and Reward Models. *arXiv [csCL]*. Published online 2025. <http://arxiv.org/abs/2501.03262>
10. Kazemnejad A, Aghajohari M, Portelance E, et al. VinePPO: Unlocking RL Potential For LLM Reasoning Through Refined Credit Assignment. *arXiv [csLG]*. Published online 2024. <http://arxiv.org/abs/2410.01679>
11. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal Policy Optimization Algorithms. *arXiv [csLG]*. Published online 2017. <http://arxiv.org/abs/1707.06347>
12. Yu Q, Zhang Z, Zhu R, et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv [csLG]*. Published online 2025. <http://arxiv.org/abs/2503.14476>
13. MiniMax, :, Chen A, et al. MiniMax-M1: Scaling Test-Time Compute Efficiently with Lightning Attention. *arXiv [csCL]*. Published online 2025. <http://arxiv.org/abs/2506.13585>
14. Zheng C, Liu S, Li M, et al. Group Sequence Policy Optimization. *arXiv [csLG]*. Published online 2025. <http://arxiv.org/abs/2507.18071>
15. Yao F, Liu L, Zhang D, Dong C, Shang J, Gao J. Your Efficient RL Framework Secretly Brings You Off-Policy RL Training. *Feng Yao's Notion*. Published online August 2025. <https://fengyao.notion.site/off-policy-rl>
16. Qi P, Liu Z, Zhou X, et al. Defeating the Training-Inference Mismatch via FP16. *arXiv [csLG]*. Published online 2025. <http://arxiv.org/abs/2510.26788>
17. Fu W, Gao J, Shen X, et al. AReal: A Large-Scale Asynchronous Reinforcement Learning System for Language Reasoning. *arXiv [csLG]*. Published online 2025. <http://arxiv.org/abs/2505.24298>
18. Piché A, Kamaloo E, Pardinas R, Chen X, Bahdanau D. PipelineRL: Faster On-policy Reinforcement Learning for Long Sequence Generation. *arXiv [csLG]*. Published online 2025. <http://arxiv.org/abs/2509.19128>
19. Khatri D, Madaan L, Tiwari R, et al. The Art of Scaling Reinforcement Learning Compute for LLMs. *arXiv [csLG]*. Published online 2025. <http://arxiv.org/abs/2510.13786>
20. P. Aggarwal and S. Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025.