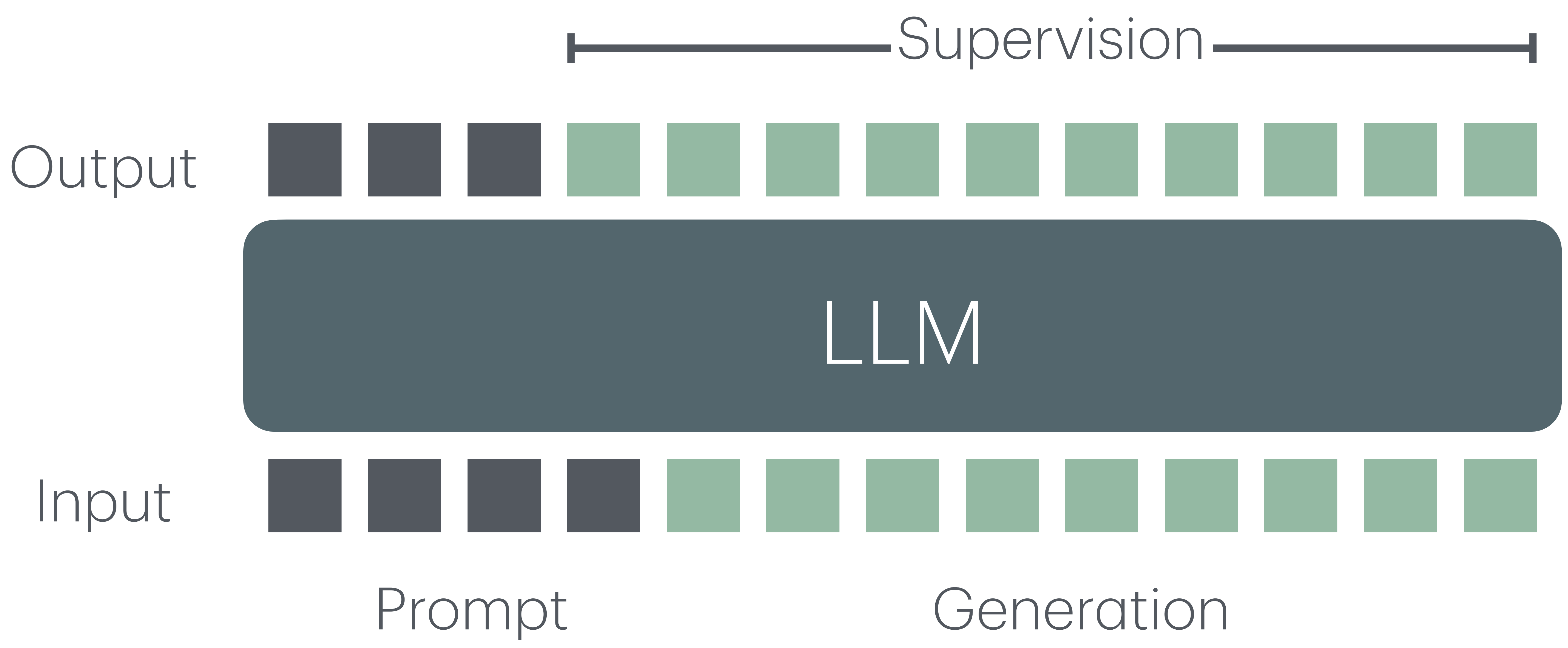
Bonus:
Reinforcement Learning and LLMs

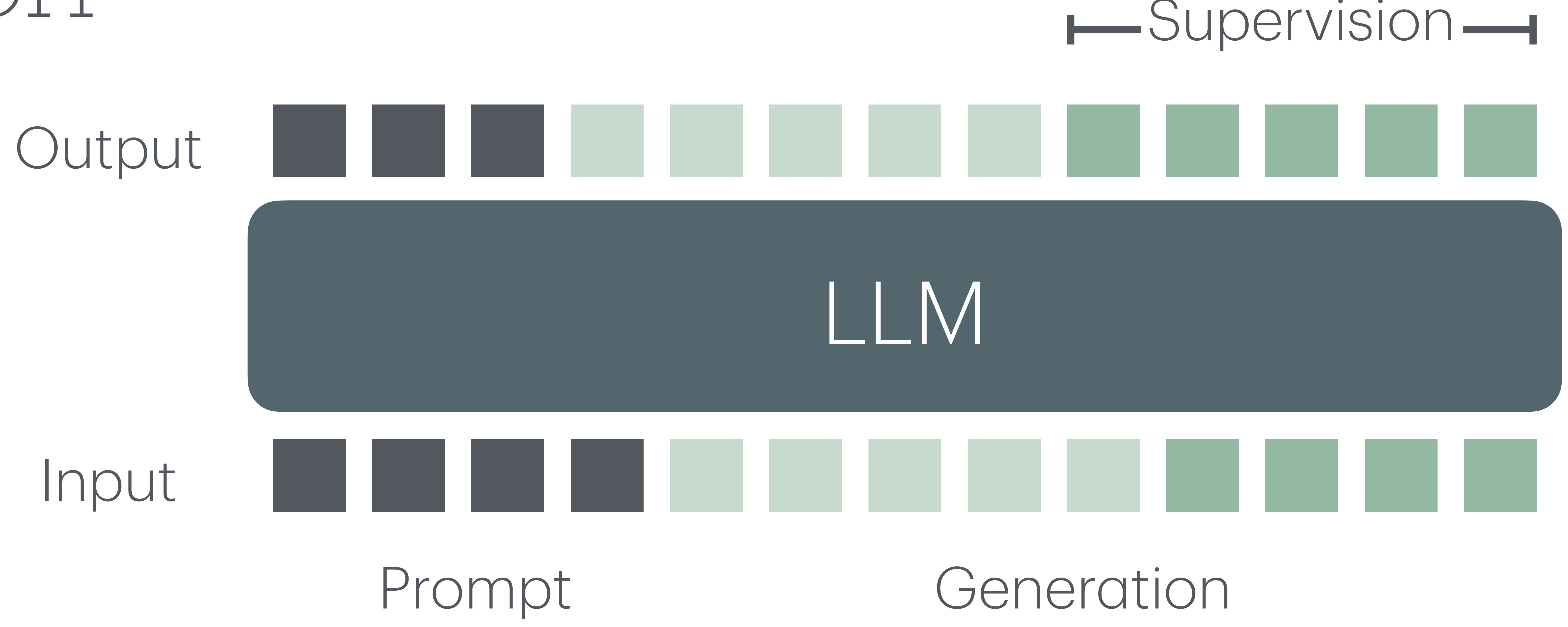
Philipp Krähenbühl, UT Austin

Teacher forcing

- Simple supervised learning
  - Input = Prompt + Target[0:-1]
  - Loss(output, Target[1:])



Outcome supervision



- What if we only supervise the final result?
- Generation
  - Loss(Generation)
- Teacher-forcing not possible
  - No supervised loss
- Solution: RL

Outcome supervision

Reinforcement Learning

- LLM $p_\theta(x_{t+1} \mid \mathbf{c}, x_1 \ldots x_t)$

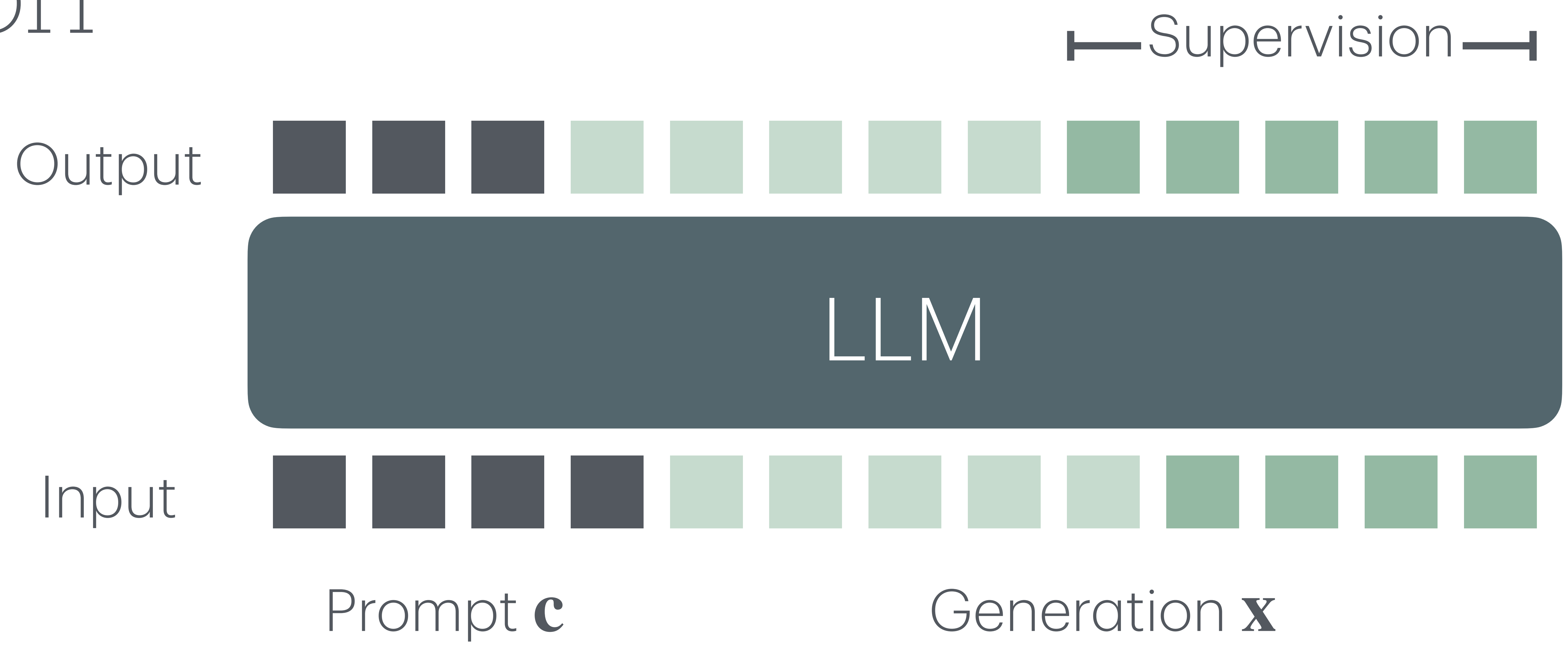
$$p_\theta(\mathbf{x} \mid \mathbf{c}) = \prod_{t=1}^{N} p_\theta(x_{t+1} \mid \mathbf{c}, x_1 \ldots x_t)$$

- Sampling / Generation

$$x_{t+1} \sim p_\theta(\,\cdot \mid \mathbf{c}, x_1 \ldots x_t)$$

- MDP

$$E_{\mathbf{x} \sim p_\theta(\cdot \mid \mathbf{c})}\Bigg[\underbrace{\sum_{t=1}^{N} r(x_t \mid \mathbf{c}, x_1 \ldots x_{t-1})}_{R(\mathbf{c},\mathbf{x})}\Bigg]$$



REINFORCE

maximize $E_{\mathbf{x}\sim p_\theta(\cdot|\mathbf{c})}\left[R(\mathbf{c},\mathbf{x})\right]$

- Using gradient ascent

  $$\nabla_\theta E_{\mathbf{x}\sim p_\theta(\cdot|\mathbf{c})}\left[R(\mathbf{c},\mathbf{x})\right] = E_{\mathbf{x}\sim p_\theta(\cdot|\mathbf{c})}\left[R(\mathbf{c},\mathbf{x})\,\nabla_\theta \log p_\theta(\mathbf{x}\,|\,\mathbf{c})\right]$$

- With a Monte-Carlo estimate

  $$\nabla_\theta E_{\mathbf{x}\sim p_\theta(\cdot|\mathbf{c})}\left[R(\mathbf{c},\mathbf{x})\right] \approx \frac{1}{K}\sum_{k=1}^{K} R(\mathbf{c},\mathbf{x}_k)\,\nabla_\theta \log p_\theta(\mathbf{x}_k\,|\,\mathbf{c})$$

  for $\mathbf{x}_k \sim p_\theta(\,\cdot\,|\,\mathbf{c})$

- REINFORCE K=1 works!!!

Initialize θ
for ever:
  Sample (or iterate over) $\mathbf{c}$
  $\mathbf{x} \sim p_\theta(\,\cdot\,|\,\mathbf{c})$
  $\theta \leftarrow \theta + \epsilon R(\mathbf{c},\mathbf{x})\,\nabla \log p_\theta(\mathbf{x}\,|\,\mathbf{c})$

Policy Gradient

$$E_{\mathbf{x}\sim p_\theta(\cdot|\mathbf{c})}\left[A(\mathbf{c},\mathbf{x})\,\nabla_\theta \log p_\theta(\mathbf{x}\,|\,\mathbf{c})\right]$$

- Even better

$$E_{\mathbf{x}\sim p_\theta(\cdot|\mathbf{c})}\left[\sum_{t=1}^{T} A(\mathbf{c},x_1\ldots x_t)\,\nabla_\theta \log p_\theta(x_t\,|\,\mathbf{c},x_1\ldots x_{t-1})\right]$$

Algorithm 1 Vanilla Policy Gradient Algorithm

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

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau\in\mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

  Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

  or via another gradient ascent algorithm like Adam.
  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.
end for

Credit: OpenAI

Proximal Policy Optimization

PPO

- Policy gradient
- Reuse rollouts (go slightly off-policy)
  - Basic Option: Importance weighting

    maximize $E_{\mathbf{x}\sim p_\psi(\cdot|\mathbf{c})}\left[\frac{p_\theta(\mathbf{x}\,|\,\mathbf{c})}{p_\psi(\mathbf{x}\,|\,\mathbf{c})}A(\mathbf{c},\mathbf{x})\right]$

  - Better Option: PPO-Clipping

    maximize $E_{\mathbf{x}\sim p_\psi(\cdot|\mathbf{c})}\left[\frac{1}{T}\sum_{t=1}^{T} CLIP\left(\frac{p_\theta(x_t\,|\,\mathbf{c},x_1\ldots x_{t-1})}{p_\psi(x_t\,|\,\mathbf{c},x_1\ldots x_{t-1})},A(\mathbf{c},\mathbf{x})\right)\right]$

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

for $k=0,1,2,\ldots$ **do**

Collect set of trajectories $\mathcal{D}_k=\{\tau_i\}$ by running policy $\pi_k=\pi(\theta_k)$ in the environment.

Compute rewards-to-go $\hat{R}_t$.

Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

Update the policy by maximizing the PPO-Clip objective:

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

typically via stochastic gradient ascent with Adam.

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.

end for

Credit: OpenAI

REINFORCE vs PPO

Initialize θ

for ever:

Sample (or iterate over) $\mathbf{c}$

$\mathbf{x} \sim p_\theta(\,\cdot\,|\,\mathbf{c})$

$\theta \leftarrow \theta + \epsilon R(\mathbf{c}, \mathbf{x}) \nabla \log p_\theta(\mathbf{x}\,|\,\mathbf{c})$

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

Compute rewards-to-go $\hat{R}_t$.

Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

Update the policy by maximizing the PPO-Clip objective:

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

typically via stochastic gradient ascent with Adam.

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.

end for

Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs

Arash Ahmadian
Cohere For AI

Chris Cremer
Cohere

Matthias Gallé
Cohere

Marzieh Fadaee
Cohere For AI

Julia Kreutzer
Cohere For AI

Olivier Pietquin
Cohere

Ahmet Üstün
Cohere For AI

Sara Hooker
Cohere For AI

{arash,olivier,ahmet,sarahooker}@cohere.com

Outcome supervision

Reinforcement Learning

- LLM $p_\theta(x_{t+1} \mid \mathbf{c}, x_1 \ldots x_t)$

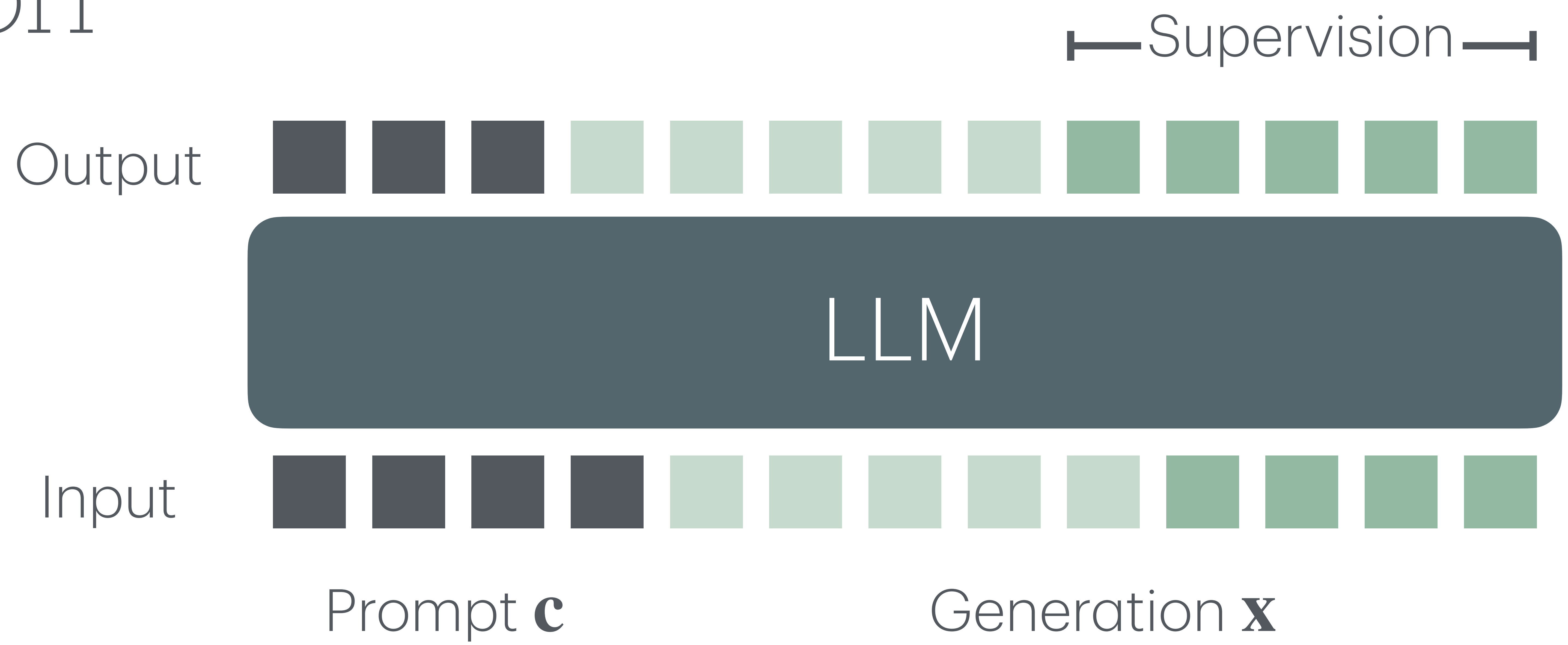
$$p_\theta(\mathbf{x} \mid \mathbf{c}) = \prod_{t=1}^{N} p_\theta(x_{t+1} \mid \mathbf{c}, x_1 \ldots x_t)$$

- Sampling / Generation

$$x_{t+1} \sim p_\theta(\cdot \mid \mathbf{c}, x_1 \ldots x_t)$$

- MDP

$$E_{\mathbf{x} \sim p_\theta(\cdot \mid \mathbf{c})} \left[\sum_{t=1}^{N} r(x_t \mid \mathbf{c}, x_1 \ldots x_{t-1}) \right]$$



Outcome supervision

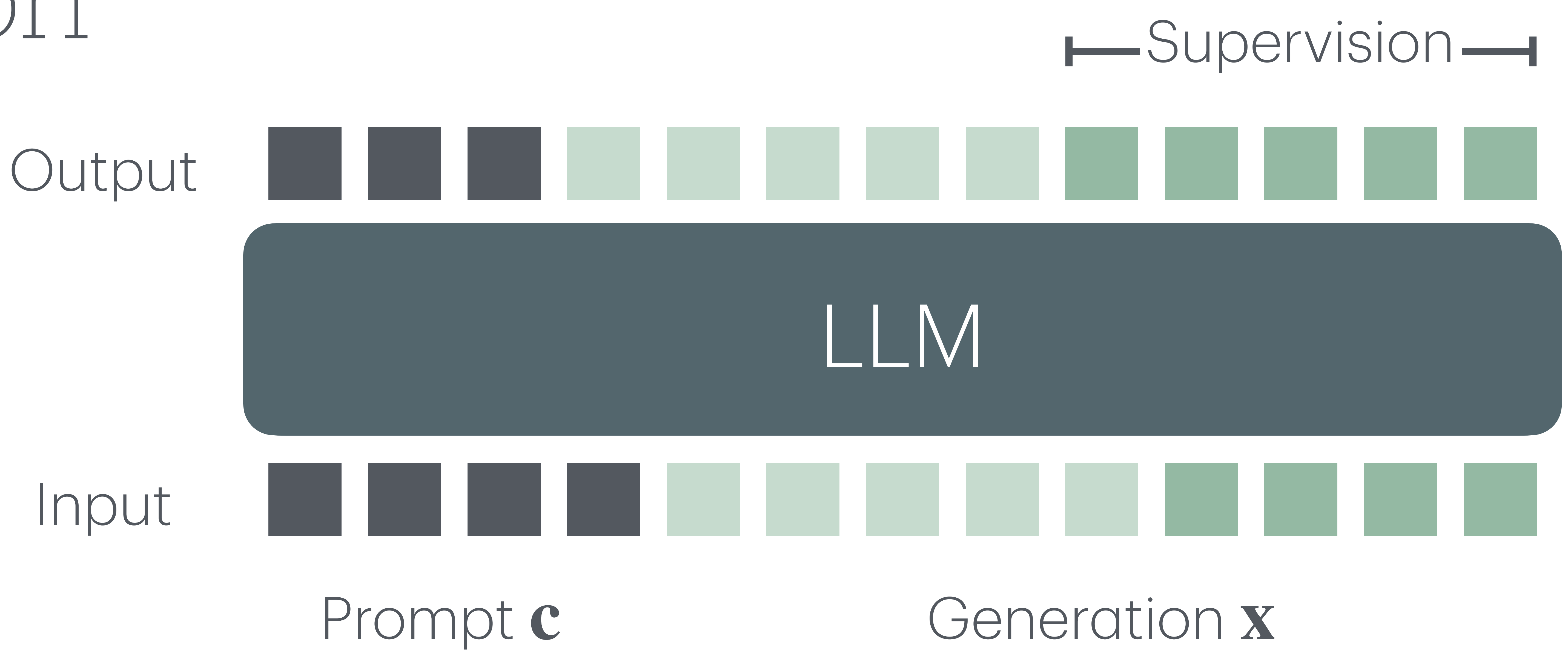
Reinforcement Learning

- LLM $p_\theta(\mathbf{x} \mid \mathbf{c})$
- Sampling / Generation

$$\mathbf{x} \sim p_\theta(\cdot \mid \mathbf{c})$$

- Contextual bandit

$$E_{\mathbf{x} \sim p_\theta(\cdot \mid \mathbf{c})}\left[R(\mathbf{c}, \mathbf{x})\right]$$



REINFORCE Leave One Out

- $\nabla_\theta E_{\mathbf{x}\sim p_\theta(\cdot|\mathbf{c})}\left[R(\mathbf{c},\mathbf{x})\right] \approx \frac{1}{K}\sum_{k=1}^{K}\left(R(\mathbf{c},\mathbf{x}_k) - b(\mathbf{c})\right)\nabla_\theta \log p_\theta(\mathbf{x}_k\,|\,\mathbf{c})$

  for $\mathbf{x}_k \sim p_\theta(\,\cdot\,|\,\mathbf{c})$

- $b(\mathbf{c}) = \frac{1}{K}\sum_{k=1}^{K} R(\mathbf{c},\mathbf{x}_k)$

Initialize θ

for ever:

  Sample (or iterate over) $\mathbf{c}$

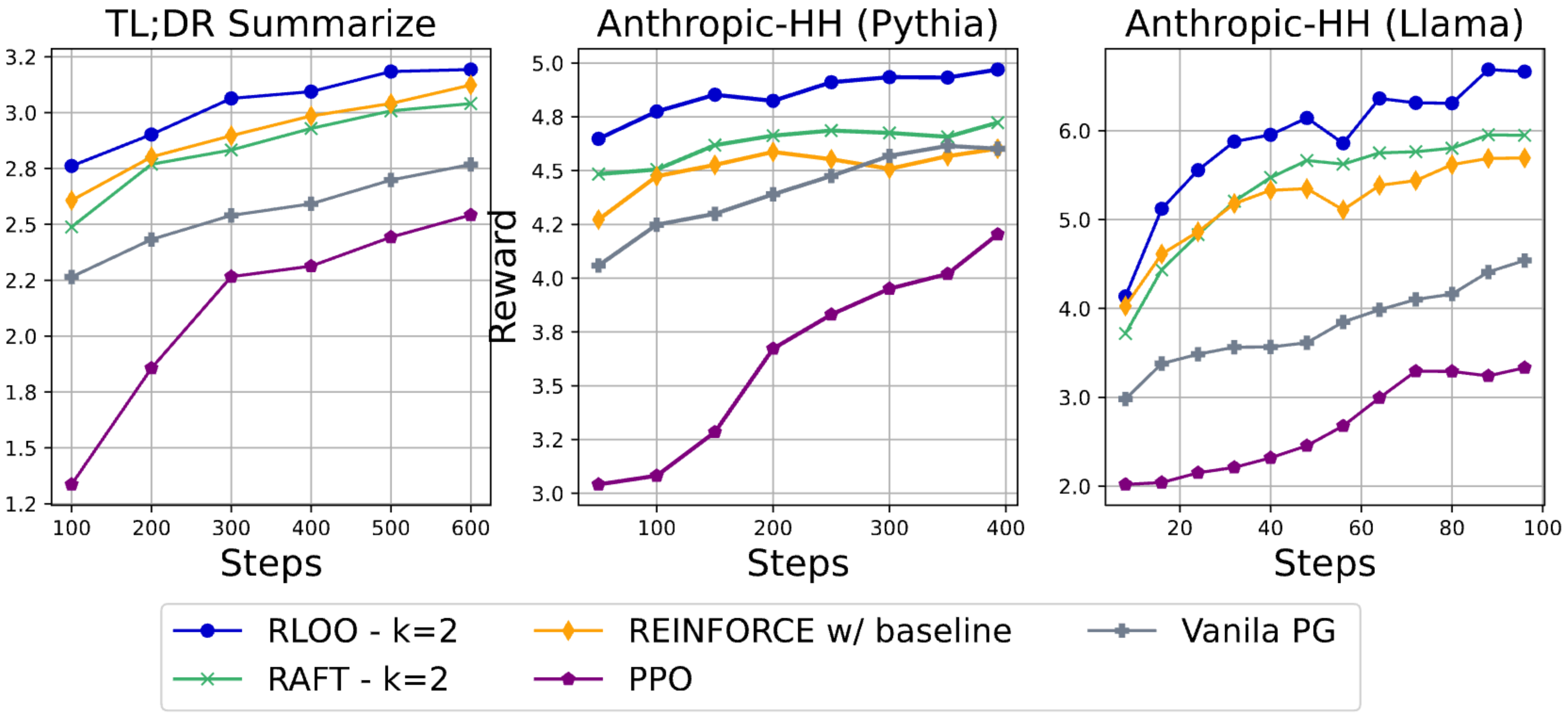
  $\mathbf{x}_k \sim p_\theta(\,\cdot\,|\,\mathbf{c})$ for k=1…K

  $b(\mathbf{c}) = \frac{1}{K}\sum_{k=1}^{K} R(\mathbf{c},\mathbf{x}_k)$

  $\theta \leftarrow \theta + \epsilon\frac{1}{K}\sum_{k=1}^{K}\left(R(\mathbf{c},\mathbf{x}_k) - b(\mathbf{x})\right)\nabla \log p_\theta(\mathbf{x}_k\,|\,\mathbf{c})$

Buy 4 REINFORCE Samples, Get a Baseline for Free!, Kool etal 2019

RLOO in LLMs



Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs, Ahmadian etal 2024

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

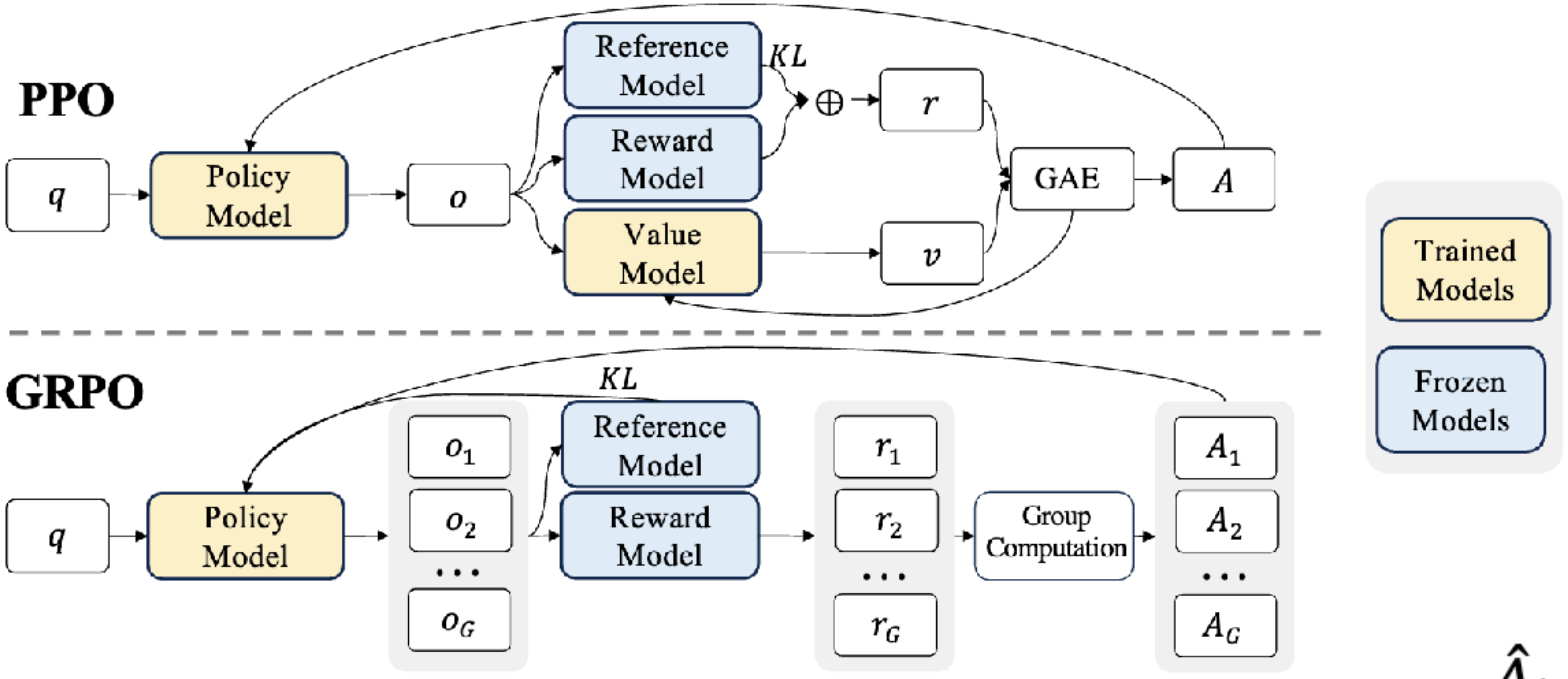
Zhihong Shao[1,2*†], Peiyi Wang[1,3*†], Qihao Zhu[1,3*†], Runxin Xu[1], Junxiao Song[1]
Xiao Bi[1], Haowei Zhang[1], Mingchuan Zhang[1], Y.K. Li[1], Y. Wu[1], Daya Guo[1*]

[1]DeepSeek-AI, [2]Tsinghua University, [3]Peking University

{zhihongshao,wangpeiyi,zhuqh,guoday}@deepseek.com
https://github.com/deepseek-ai/DeepSeek-Math

GRPO

$$\mathcal{J}_{PPO}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_{\theta_{old}}(O|q)] \frac{1}{|o|} \sum_{t=1}^{|o|} \min\left[\frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{old}}(o_t|q, o_{<t})} A_t, \text{clip}\left(\frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{old}}(o_t|q, o_{<t})}, 1-\varepsilon, 1+\varepsilon\right) A_t\right],$$



$$\hat{A}_{i,t} = \widetilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta || \pi_{ref}\right]\right\},$$

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, Shao etal 2024

Algorithm 1 Iterative Group Relative Policy Optimization

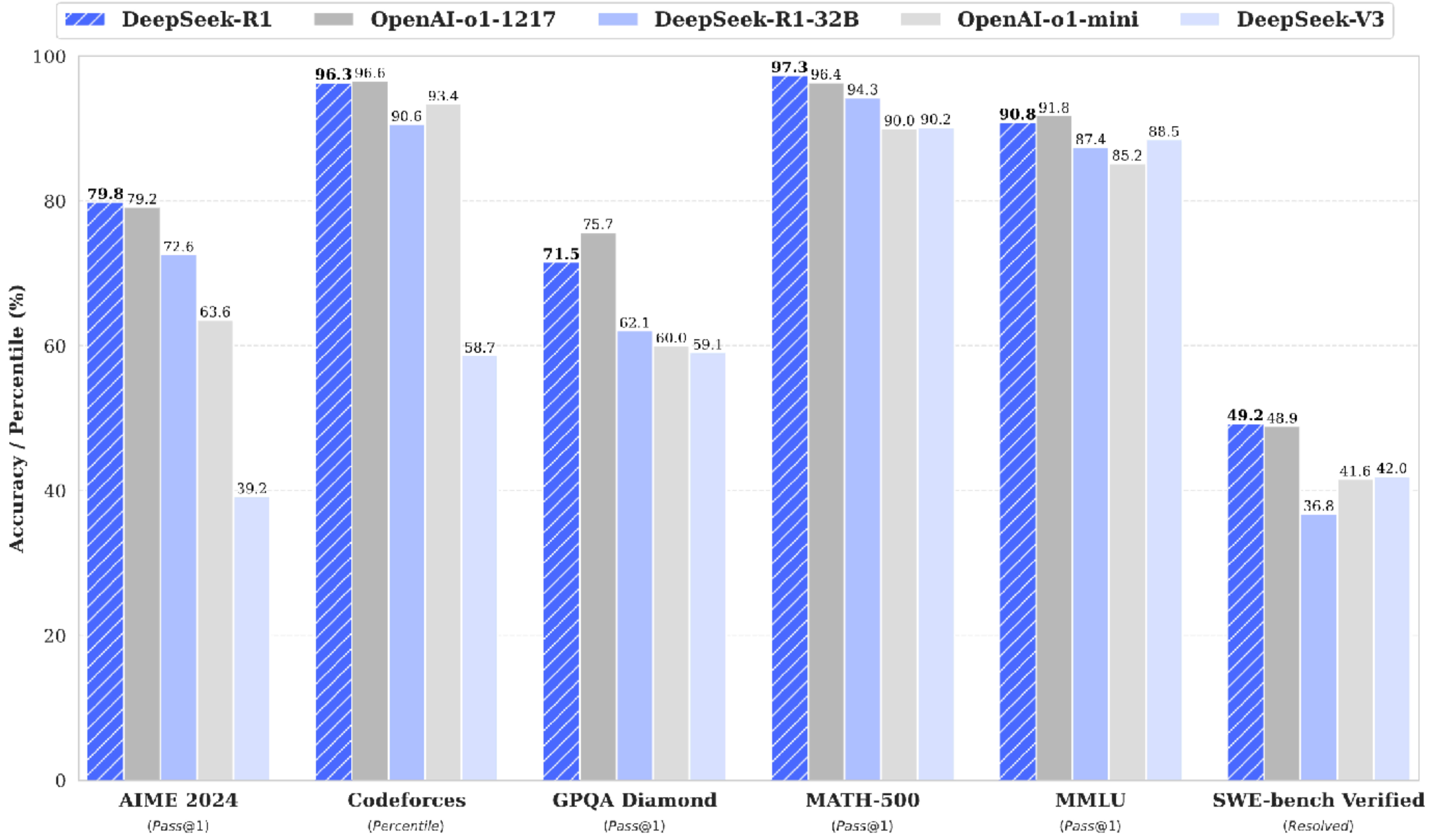
Input initial policy model $\pi_{\theta_{\text{init}}}$; reward models r_φ; task prompts $\mathcal{D}$; hyperparameters ε, β, μ

policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$
for iteration = 1, ..., I **do**
    reference model $\pi_{ref} \leftarrow \pi_\theta$
    for step = 1, ..., M **do**
        Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
        Update the old policy model $\pi_{\theta_{old}} \leftarrow \pi_\theta$
        Sample G outputs $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot \mid q)$ for each question $q \in \mathcal{D}_b$
        Compute rewards $\{r_i\}_{i=1}^{G}$ for each sampled output o_i by running r_φ
        Compute $\hat{A}_{i,t}$ for the t-th token of o_i through group relative advantage estimation.
        for GRPO iteration = 1, ..., μ **do**
            Update the policy model π_θ by maximizing the GRPO objective (Equation 21)
    Update r_φ through continuous training using a replay mechanism.

Output π_θ

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, Shao etal 2024

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning



DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, Deepseek team 2025

Step 1: Create a dataset of math puzzles and alike

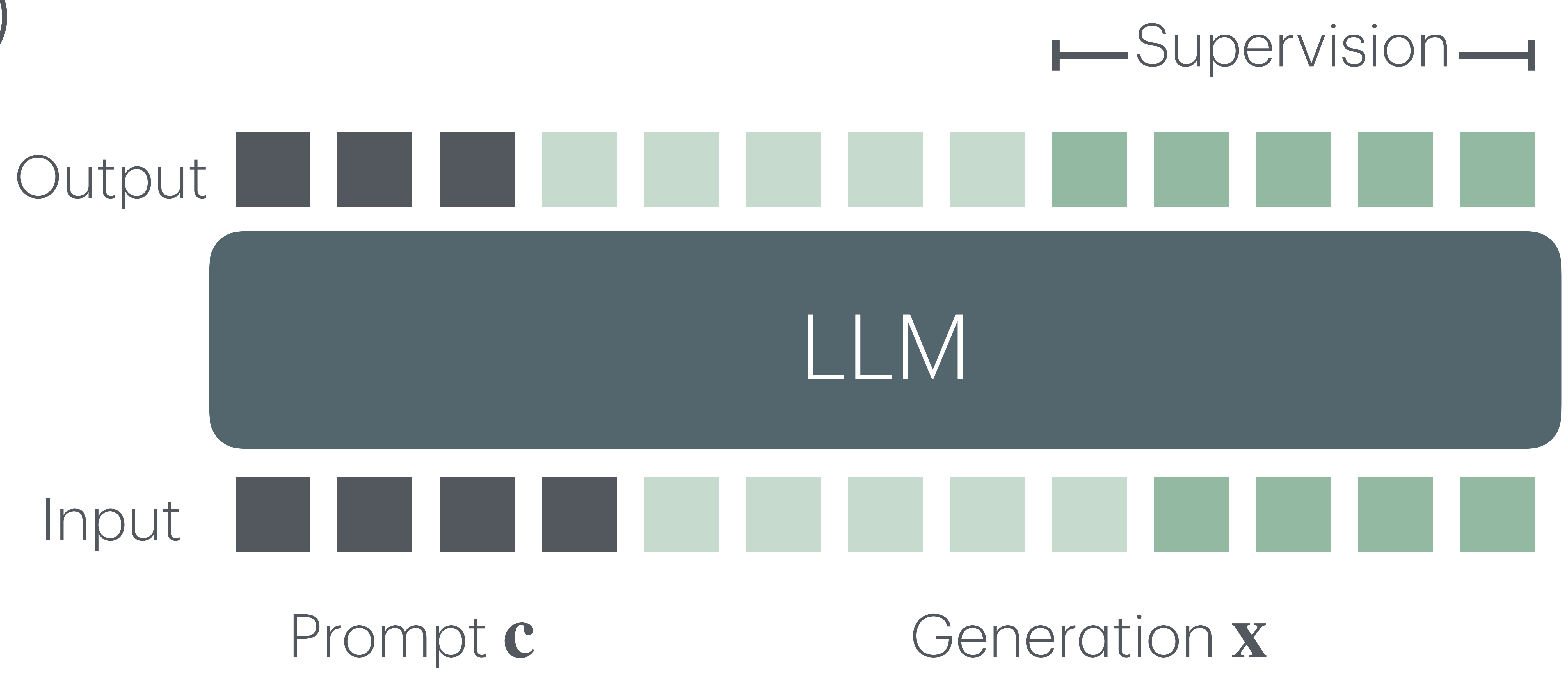
- Interesting prompts
- Easily verifiable answers
  - Math, Reasoning, Multiple-choice, …
- No details given in paper

Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a + x}} = x$ is equal to

$$\frac{-1 \pm \sqrt{1 + 4a}}{2}$$

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, Deepseek team 2025

Step 2: Run GRPO



- No teacher forcing
- Supervise just easily verifiable answers
- Learns reasoning
- R1-Zero
  - From just pre-trained model, no instruction tuning
  - It works

Step 3: Bootstrap Instruction tuned model

- Use instruction tuning data and R1-Zero data
- Train a chat-bot that can "reason"

	Benchmark (Metric)	Claude-3.5-Sonnet-1022	GPT-4o 0513	DeepSeek V3	OpenAI o1-mini	OpenAI o1-1217	DeepSeek R1
	Architecture	-	-	MoE	-	-	MoE
	# Activated Params	-	-	37B	-	-	37B
	# Total Params	-	-	671B	-	-	671B
English	MMLU (Pass@1)	88.3	87.2	88.5	85.2	**91.8**	90.8
	MMLU-Redux (EM)	88.9	88.0	89.1	86.7	-	**92.9**
	MMLU-Pro (EM)	78.0	72.6	75.9	80.3	-	**84.0**
	DROP (3-shot F1)	88.3	83.7	91.6	83.9	90.2	**92.2**
	IF-Eval (Prompt Strict)	**86.5**	84.3	86.1	84.8	-	83.3
	GPQA Diamond (Pass@1)	65.0	49.9	59.1	60.0	**75.7**	71.5
	SimpleQA (Correct)	28.4	38.2	24.9	7.0	**47.0**	30.1
	FRAMES (Acc.)	72.5	80.5	73.3	76.9	-	**82.5**
	AlpacaEval2.0 (LC-winrate)	52.0	51.1	70.0	57.8	-	**87.6**
	ArenaHard (GPT-4-1106)	85.2	80.4	85.5	92.0	-	**92.3**
Code	LiveCodeBench (Pass@1-COT)	38.9	32.9	36.2	53.8	63.4	**65.9**
	Codeforces (Percentile)	20.3	23.6	58.7	93.4	**96.6**	96.3
	Codeforces (Rating)	717	759	1134	1820	**2061**	2029
	SWE Verified (Resolved)	**50.8**	38.8	42.0	41.6	48.9	49.2
	Aider-Polyglot (Acc.)	45.3	16.0	49.6	32.9	**61.7**	53.3
Math	AIME 2024 (Pass@1)	16.0	9.3	39.2	63.6	79.2	**79.8**
	MATH-500 (Pass@1)	78.3	74.6	90.2	90.0	96.4	**97.3**
	CNMO 2024 (Pass@1)	13.1	10.8	43.2	67.6	-	**78.8**
Chinese	CLUEWSC (EM)	85.4	87.9	90.9	89.9	-	**92.8**
	C-Eval (EM)	76.7	76.0	86.5	68.9	-	**91.8**
	C-SimpleQA (Correct)	55.4	58.7	**68.0**	40.3	-	63.7

Algorithm 1 Iterative Group Relative Policy Optimization

Input initial policy model $\pi_{\theta_{\text{init}}}$; reward models r_φ; task prompts $\mathcal{D}$; hyperparameters ε, β, μ

policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$
for iteration = 1, ..., I **do**
    reference model $\pi_{ref} \leftarrow \pi_\theta$
    for step = 1, ..., M **do**
        Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
        Update the old policy model $\pi_{\theta_{old}} \leftarrow \pi_\theta$
        Sample G outputs $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot \mid q)$ for each question $q \in \mathcal{D}_b$
        Compute rewards $\{r_i\}_{i=1}^{G}$ for each sampled output o_i by running r_φ
        Compute $\hat{A}_{i,t}$ for the t-th token of o_i through group relative advantage estimation.
        for GRPO iteration = 1, ..., μ **do**
            Update the policy model π_θ by maximizing the GRPO objective (Equation 21)
    Update r_φ through continuous training using a replay mechanism.

Output π_θ

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, Shao etal 2024

Algorithm 1 Iterative Group Relative Policy Optimization

Input initial policy model $\pi_{\theta_{\text{init}}}$; reward models r_φ; task prompts $\mathcal{D}$; hyperparameters ε, β, μ

policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$
for iteration = 1, …, I **do**
    reference model $\pi_{ref} \leftarrow \pi_\theta$
    for step = 1, …, M **do**
        Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
        Update the old policy model $\pi_{\theta_{old}} \leftarrow \pi_\theta$
        Sample G outputs $\{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(\cdot \mid q)$ for each question $q \in \mathcal{D}_b$
        Compute rewards $\{r_i\}_{i=1}^G$ for each sampled output o_i by running r_φ
        Compute $\hat{A}_{i,t}$ for the t-th token of o_i through group relative advantage estimation.
        ~~**for** GRPO iteration = 1, …, μ **do**~~
            Update the policy model π_θ by maximizing the GRPO objective (Equation 21)
    Update r_φ through continuous training using a replay mechanism.

Output π_θ

$$\hat{A}_{i,t} = \widetilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

max length is set to 1024, and the training batch size is 1024. The policy model only has a single update following each exploration stage. We evaluate DeepSeekMath-RL 7B on benchmarks

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \cancel{\text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta || \pi_{ref}\right]\right\},$$

GRPO = RLOO with advantage normalization

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, Shao etal 2024

RLOO vs GRPO

RLOO

Initialize θ

for ever:

Sample (or iterate over) $\mathbf{c}$

$\mathbf{x}_k \sim p_\theta(\cdot \mid \mathbf{c})$ for k=1...K

$$b(\mathbf{c}) = \frac{1}{K}\sum_{k=1}^{K} R(\mathbf{c}, \mathbf{x}_k)$$

$$\theta \leftarrow \theta + \epsilon \frac{1}{K}\sum_{k=1}^{K} \left(R(\mathbf{c}, \mathbf{x}_k) - b(\mathbf{x})\right) \nabla \log p_\theta(\mathbf{x}_k \mid \mathbf{c})$$

GRPO (in practice)

Initialize θ

for ever:

Sample (or iterate over) $\mathbf{c}$

$\mathbf{x}_k \sim p_\theta(\cdot \mid \mathbf{c})$ for k=1...K

$$b(\mathbf{c}) = \frac{1}{K}\sum_{k=1}^{K} R(\mathbf{c}, \mathbf{x}_k)$$

$$\theta \leftarrow \theta + \epsilon \frac{1}{K}\sum_{k=1}^{K} \frac{R(\mathbf{c}, \mathbf{x}_k) - b(\mathbf{x})}{\mathrm{std}(\mathbf{x})} \nabla \log p_\theta(\mathbf{x}_k \mid \mathbf{c})$$

Interactive Digital Agents

- Train LLMs that interact with API's on the users behalf

Reinforcement Learning for Long-Horizon Interactive LLM Agents, Chen … Krähenbühl 2025

Interactive Digital Agents

LOOP

- Use simulator for API interactions
  - AppWorld
- Training data = 24 scenarios (simple request, initial simulator state, test cases)
- Trained using simple combination of Leave-One-Out estimator and PPO

Algorithm 1 Leave-One-Out Proximal Policy Optimization

Input: Policy p_θ, dataset of tasks and initial states $\mathcal{D}$
Output: Policy p_θ maximizing $\mathbb{E}_{\mathbf{s}_0,\mathbf{c}\sim\mathcal{D}}\left[L_\theta(\mathbf{s}_0,\mathbf{c})\right]$ (Eq. 7)

for iteration $= 1, 2, \ldots$ **do**
  $\mathbf{B} \leftarrow \{\}$ ▷ *Initialize rollout buffer*
  for $(\mathbf{s}_0, \mathbf{c}) \sim \mathcal{D}$ **do** ▷ *Rollout collection*
    Collect K rollouts $\mathbf{x}_1, \ldots, \mathbf{x}_K \overset{\text{i.i.d.}}{\sim} \rho_\theta(\cdot|\mathbf{s}_0, \mathbf{c})$
    Estimate advantages $A_1, \ldots, A_K$ using Eq. 3
    $\mathbf{B} \leftarrow \mathbf{B} \cup \{(\mathbf{x}_1, A_1), \ldots, (\mathbf{x}_K, A_K)\}$
  for epoch $= 1, \ldots, N_{\text{epoch}}$ **do** ▷ *Policy update*
    for mini-batch $\{(\mathbf{x}_i, A_i)\}_{i=1}^{M} \sim \mathbf{B}$ **do**
      Update policy using PPO gradient (Eq. 5)

$$A(\mathbf{c}, \mathbf{x}_k) = \frac{K}{K-1}\left(R(\mathbf{c}, \mathbf{x}_k) - \frac{1}{K}\sum_{i=1}^{K} R(\mathbf{c}, \mathbf{x}_i)\right). \quad (3)$$

$$L_\theta^{\text{MDP}}(\mathbf{c}) = \mathbb{E}_{\mathbf{x}\sim p_\psi(\cdot|\mathbf{c})}\left[\frac{1}{|\mathbf{x}|}\sum_{t=1}^{|\mathbf{x}|}\min\left(\frac{p_\theta(x_t|\mathbf{c},x_{1:t-1})}{p_\psi(x_t|\mathbf{c},x_{1:t-1})}A(\mathbf{c},\mathbf{x}),\, g_\epsilon(A(\mathbf{c},\mathbf{x}))\right)\right]. \quad (5)$$

Reinforcement Learning for Long-Horizon Interactive LLM Agents, Chen … Krähenbühl 2025

Type	Algorithm	Action	Strictly on-policy	Normalized reward	Test Normal (Test-N)		Test Challenge (Test-C)	
					TGC	SGC	TGC	SGC
NFT	GPT-4o	–	–	–	48.8	32.1	30.2	13
NFT	OpenAI o1	–	–	–	61.9	41.1	36.7	19.4
NFT	Llama 3 70B	–	–	–	24.4	17.9	7.0	4.3
NFT	Qwen 2.5 32B	–	–	–	39.2 ± 3.5	18.6 ± 2.0	21.0 ± 1.4	7.5 ± 1.2
SFT	SFT-GT	–	–	–	6.2 ± 0.7	1.8 ± 0.0	0.8 ± 0.2	0.1 ± 0.3
SFT	RFT	–	–	–	47.9 ± 3.7	26.4 ± 2.3	26.4 ± 1.8	11.4 ± 2.3
SFT	EI	–	–	–	58.3 ± 2.8	36.8 ± 6.0	32.8 ± 0.7	17.6 ± 1.3
DPO	DPO-MCTS	–	–	–	57.0 ± 1.5	31.8 ± 4.2	31.8 ± 1.3	13.7 ± 1.5
DPO	DMPO	–	–	–	59.0 ± 1.2	36.6 ± 4.7	36.3 ± 1.8	18.4 ± 2.3
RL	PPO (learned critic)	token			50.8 ± 3.7	28.9 ± 7.9	26.4 ± 0.5	10.5 ± 2.1
RL	RLOO	traj	✓		57.2 ± 2.6	35.7 ± 2.9	36.7 ± 1.6	17.4 ± 1.4
RL	GRPO	token	✓[3]	✓	58.0 ± 1.8	36.8 ± 3.9	39.5 ± 1.9	22.4 ± 0.8
RL	GRPO no kl	token	✓[3]	✓	59.0 ± 1.4	35.7 ± 2.9	42.7 ± 1.3	21.3 ± 1.7
RL	LOOP (bandit)	traj			53.3 ± 3.4	33.6 ± 3.2	27.7 ± 1.5	13.0 ± 0.9
RL	LOOP (turn)	turn			64.1 ± 2.2	43.5 ± 3.5	40.8 ± 1.5	26.5 ± 2.4
RL	**LOOP (token)**	token			$\mathbf{71.3 \pm 1.3}$	$\mathbf{53.6 \pm 2.2}$	$\mathbf{45.7 \pm 1.3}$	$\mathbf{26.6 \pm 1.5}$
RL	LOOP RwNorm (token)	token		✓	61.9 ± 4.0	44.1 ± 7.8	39.8 ± 1.3	20.4 ± 2.1

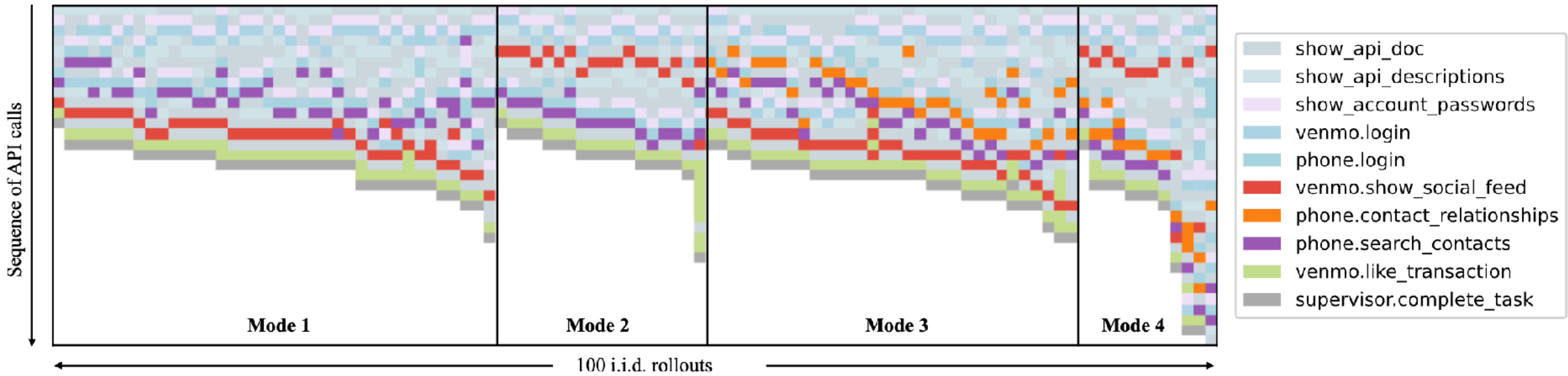
Reinforcement Learning for Long-Horizon Interactive LLM Agents, Chen ... Krähenbühl 2025

RL vs SFT

- Generations from LLM are very diverse even after RL training

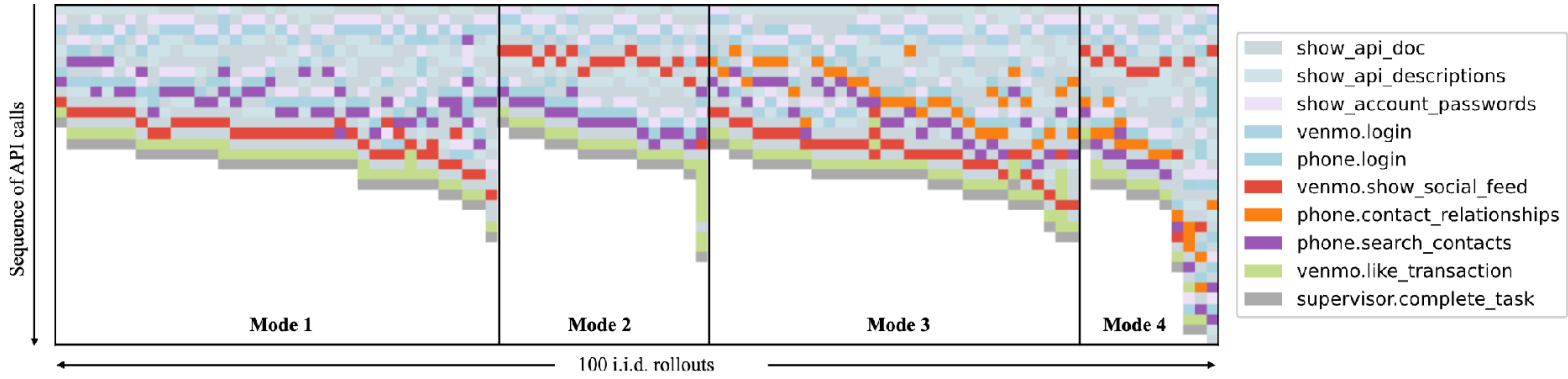
Reinforcement Learning for Long-Horizon Interactive LLM Agents, Chen … Krähenbühl 2025

RL vs SFT



Reinforcement Learning for Long-Horizon Interactive LLM Agents, Chen … Krähenbühl 2025

RL vs SFT



- Generations from LLM are very diverse even after RL training
  - 98 / 100 solutions are correct
  - Only 3 repeat overall structure of commands (no exact repetition)
- Early in training: Awesome exploration
- Late in training: No collapse / overfitting

Reinforcement Learning for Long-Horizon Interactive LLM Agents, Chen … Krähenbühl 2025

Reinforcement Learning and LLMs

- Easy to implement
    - RLOO, LOOP are just generation + reward computation + mean subtraction + training
    - Great open-source tools exist for all of this
- Much more flexible
- Less data-hungry
- It is here to stay

Initialize θ

for ever:

Sample (or iterate over) $\mathbf{c}$

$\mathbf{x}_k \sim p_\theta(\cdot \mid \mathbf{c})$ for k=1...K

$$b(\mathbf{c}) = \frac{1}{K}\sum_{k=1}^{K} R(\mathbf{c}, \mathbf{x}_k)$$

$$\theta \leftarrow \theta + \epsilon \frac{1}{K}\sum_{k=1}^{K} \left(R(\mathbf{c}, \mathbf{x}_k) - b(\mathbf{x})\right) \nabla \log p_\theta(\mathbf{x}_k \mid \mathbf{c})$$

References