How to AI an AI for Almost Anything

Lecture 6 – Test-time Training

Paul Liang
Assistant Professor
MIT Media Lab & MIT EECS

https://pliang279.github.io
ppliang@mit.edu
@pliang279

mit media lab

MIT EECS

Announcements

- Homework 2 on harness evolution will be out soon, due in 2 weeks.
- Work on projects, next checkpoint is midterm report + presentation Oct 28.
  - Processed datasets, ran baselines, done error analysis, better proposed idea
  - Bottom–up research. At the same time, continue top-down ideation from readings and literature review.

Today's Lecture

- Reinforcement Learning
- Distillation
- Test-Time Training

Roadmap

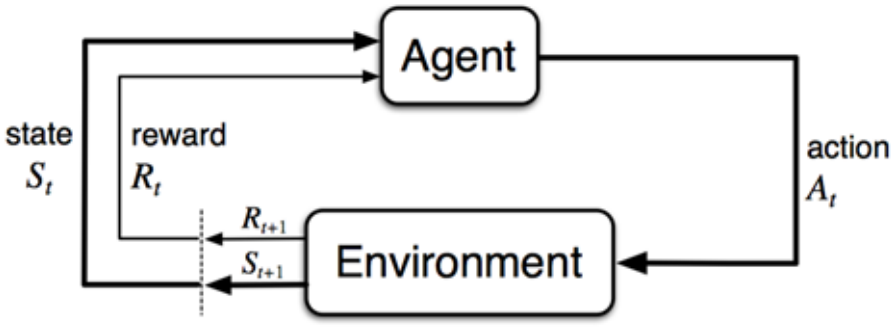
Input, (reasoning step 1, step 2, step 3…), output

Method 1: Direct prompting, no training (e.g., CoT)

Method 2: Supervised fine-tuning
- Assuming you have reasoning traces

Method 3: Reinforcement learning **(today's focus)**
- Assumes no reasoning traces
- But a reward function scoring reasoning and outputs

Learning a Policy – RL basics

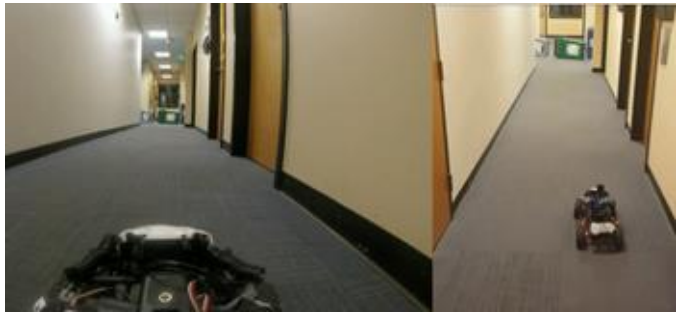










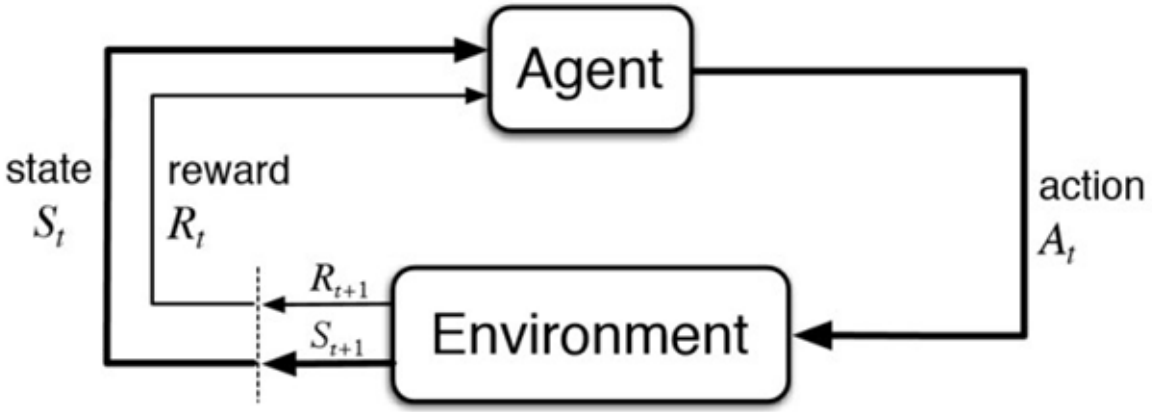


multisensory intelligence

Learning a Policy – RL basics

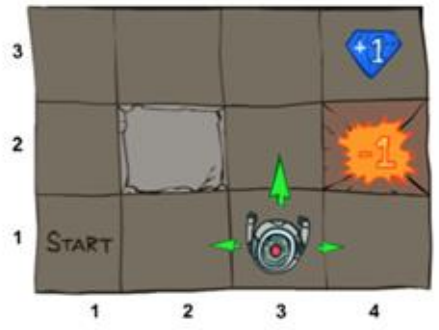
An MDP is defined by:

- Set of states S.
- Set of actions A.
- Transition function $P(s'|s,a)$.
- Reward function $r(s,a,s')$.
- Start state s_0.
- Discount factor γ.
- Horizon H.



Return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$



π:



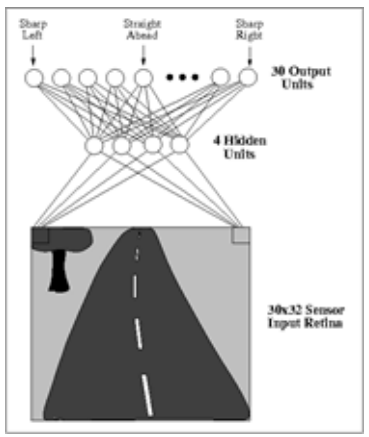
Policy: $\pi(a|s) = \Pr(A_t = a|S_t = s) \quad \forall t$

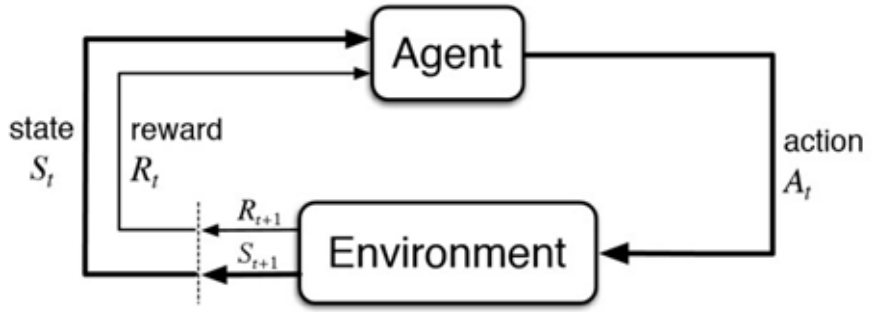
Goal: $\arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R_t|\pi\right]$

Intersection Between RL and Supervised Learning

Imitation learning







Obtain expert trajectories (e.g. human driver/video demonstrations):

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots$$

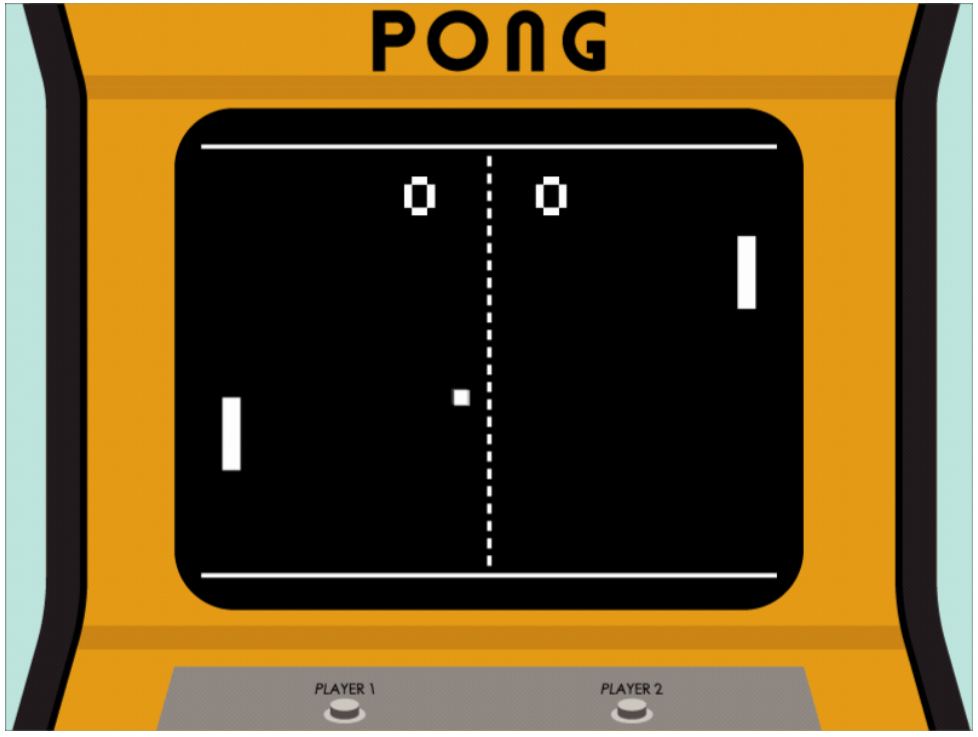
Perform supervised learning by predicting expert action

$$D = \{(s_0, a_0^*), (s_1, a_1^*), (s_2, a_2^*), \ldots\}$$

1. Distribution mismatch
2. Hard to recover from suboptimal states
3. Expert trajectories not always available

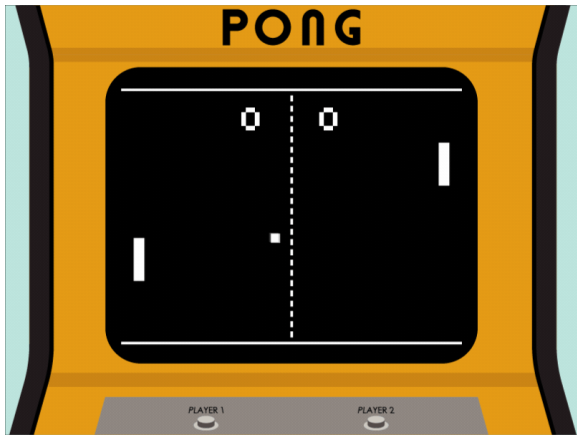
multisensory intelligence

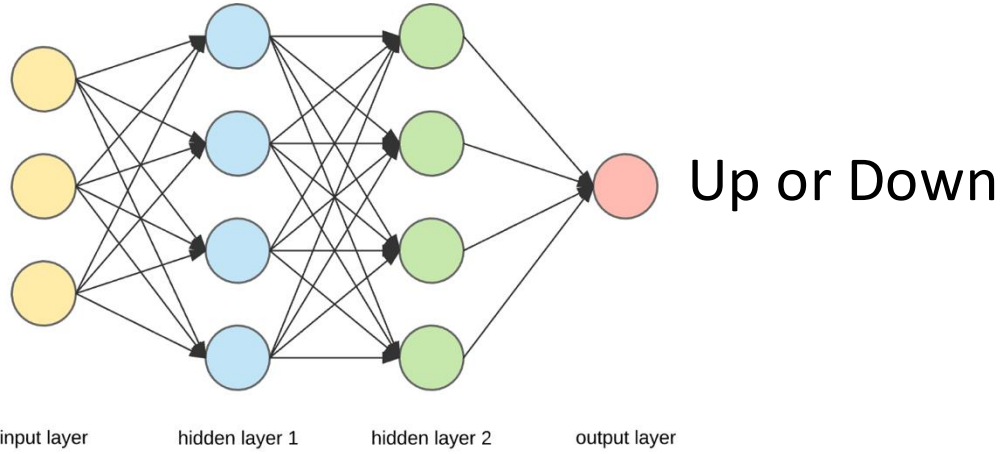
Policy Gradients



From Link

Pong from Pixels





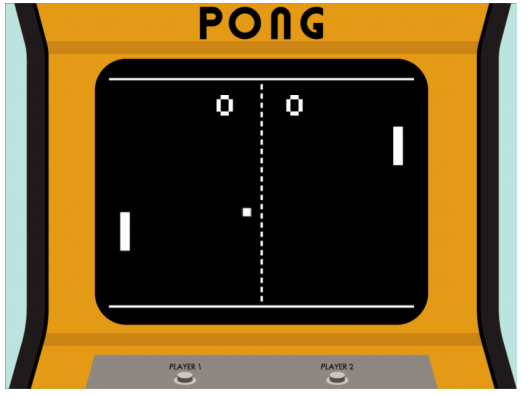
Up or Down

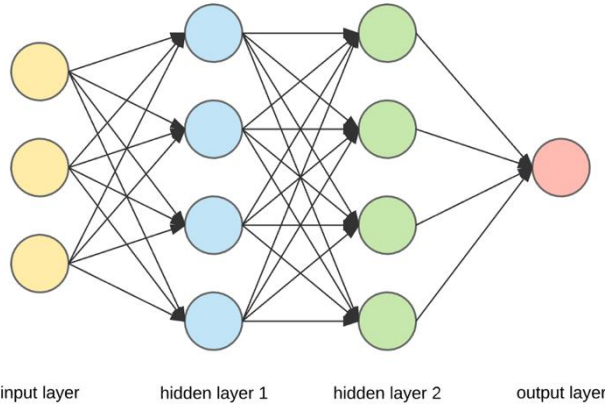
Network sees +1 if it scored a point, and -1 if it was scored against.
Can we train a network with this?

multisensory intelligence

Pong from Pixels

Suppose we have training labels?





Maximize

$$\sum_i \log p(y_i \,|x_i)$$

But we don't have training labels

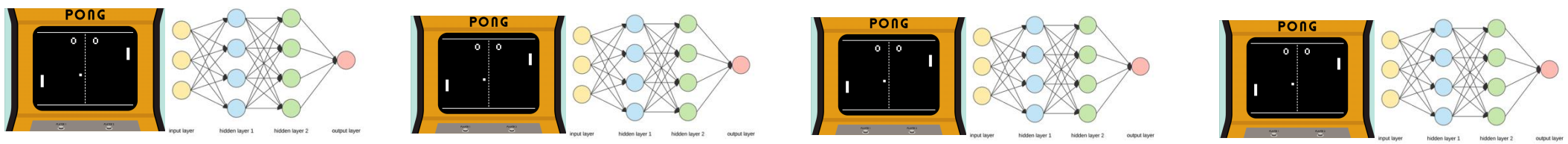
multisensory intelligence

Let’s act according to our current policy

Run - 1

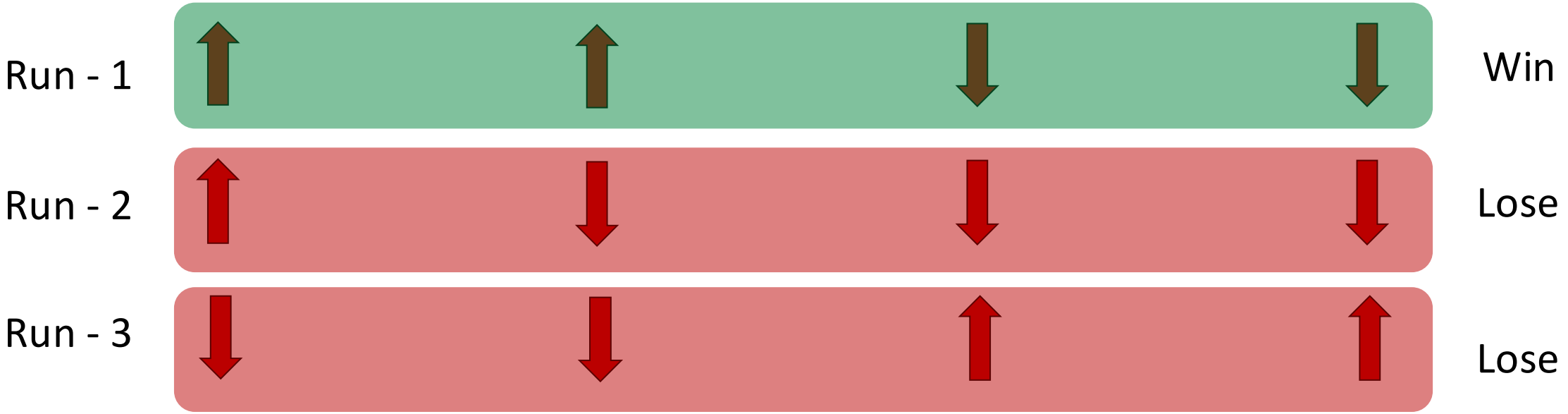
Run - 2

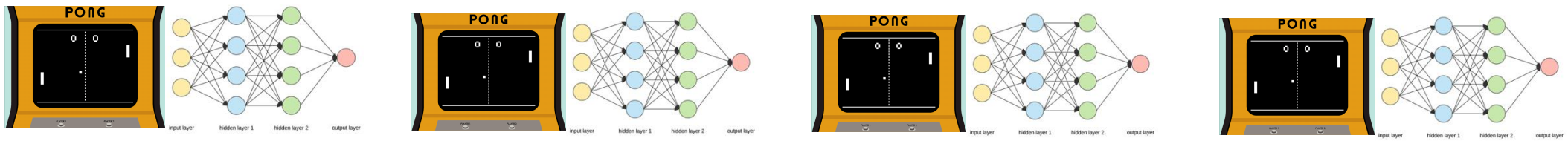
Run - 3



multisensory intelligence

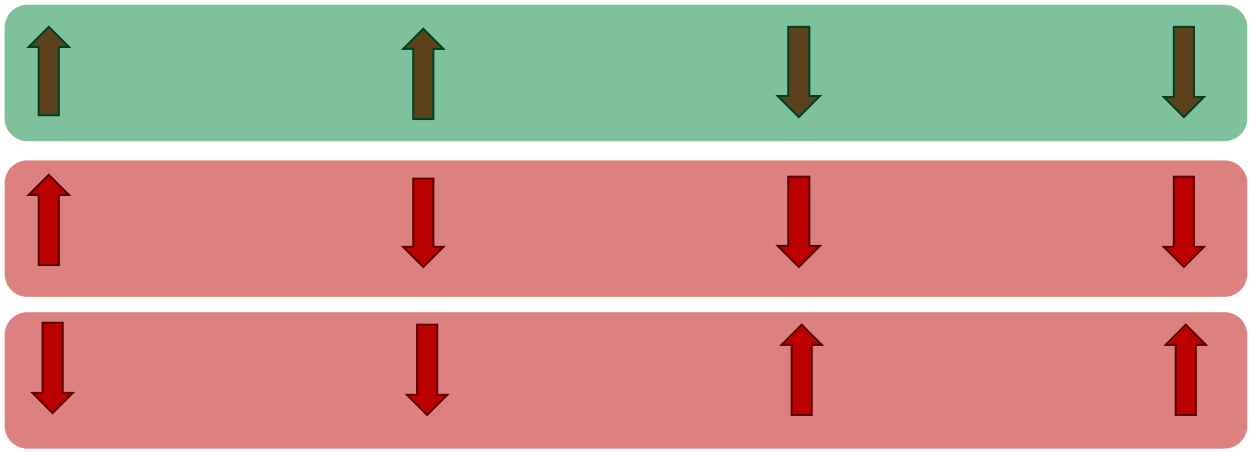
Let’s act according to our current policy





multisensory intelligence

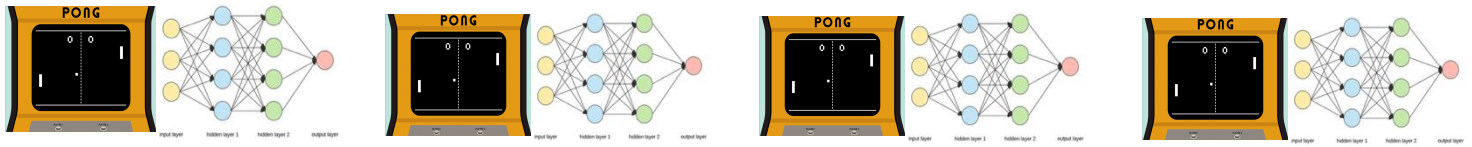
Let's act according to our current policy



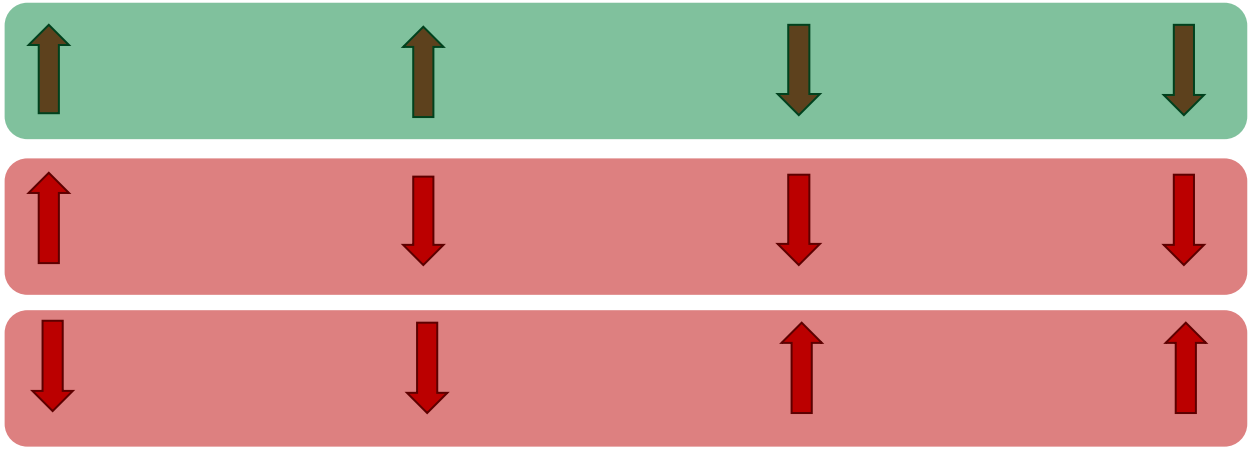
Maximize $\sum_i \log p(y_i|x_i)$

Maximize $-1 \sum_i \log p(y_i|x_i)$

Maximize $-1 \sum_i \log p(y_i|x_i)$



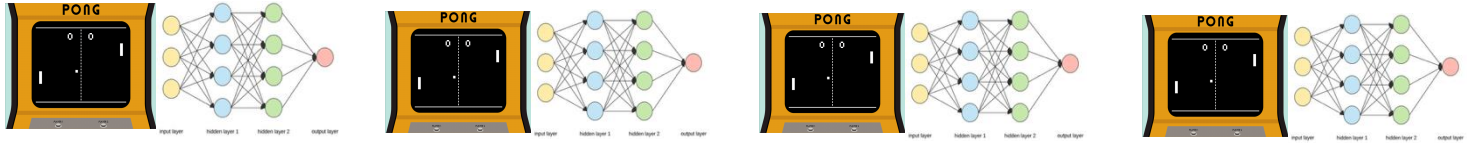
For a General Case



Maximize $\sum_i r_i \log p(y_i \mid x_i)$

Maximize $\sum_i r_i \log p(y_i \mid x_i)$

Maximize $\sum_i r_i \log p(y_i \mid x_i)$



multisensory intelligence

Reinforce Algorithm

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

ϵ-greedy

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

ϵ-greedy: start with fully random exploration when policy is not good, as policy gets better sample actions from the policy with less exploration.

Policy Gradients

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t | s_t)$$

If $r(\tau)$ is positive, increase the probability

If $r(\tau)$ is negative, decrease the probability

But this suffers from high variance

Policy Gradients

The raw reward may not be very meaningful.

What is important then? Whether a reward is higher or lower than what you expect.

-- Compare to a baseline, and use relative improvement

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(r(\tau) - b(s_t) \right) \nabla_\theta \log \pi_\theta(a_t | s_t)$$

e.g. exponential moving average of the rewards.

Actor-Critic Methods

A better baseline: want to push the probability of an action from a state, if this action was better than the expected value of what we should get from that state.

Q(s,a) and V(s) - action and state value functions!
Q(s,a): how much total reward should I expect by taking action a from state s.
V(s): how much total reward should I expect from state s.

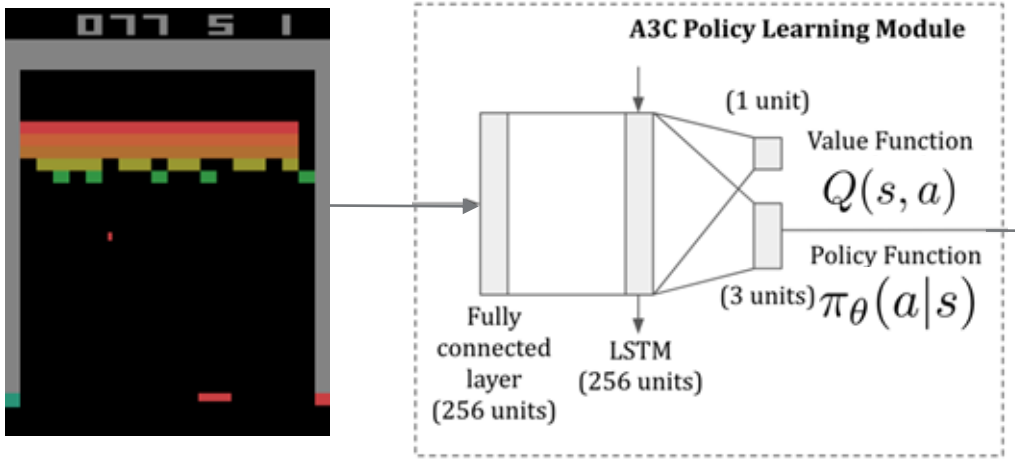
We are happy with an action **a** in a state **s** if the advantage function **A(s,a) = Q(s,a) - V(s)** is large. Otherwise we are unhappy with an action if it's small.

Using this, we get the estimator:

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)\right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Actor-Critic Methods

Two models: actor learns the policy and critic learns the value of states and actions



Critic: evaluates how good the action is

$$\mathcal{L}_i(w_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i}\left[\left(r + \gamma \max_{a'} Q(s', a'; w_i^-) - Q(s, a; w_i)\right)^2\right]$$

Q-learning target — Q-network

$$\pi_\theta(a|s)$$

Actor: decides what actions to take

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} (Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Advantage function **A(s,a)**

[Minh et al., Asynchronous Methods for Deep Reinforcement Learning. ICML 2016]

multisensory intelligence

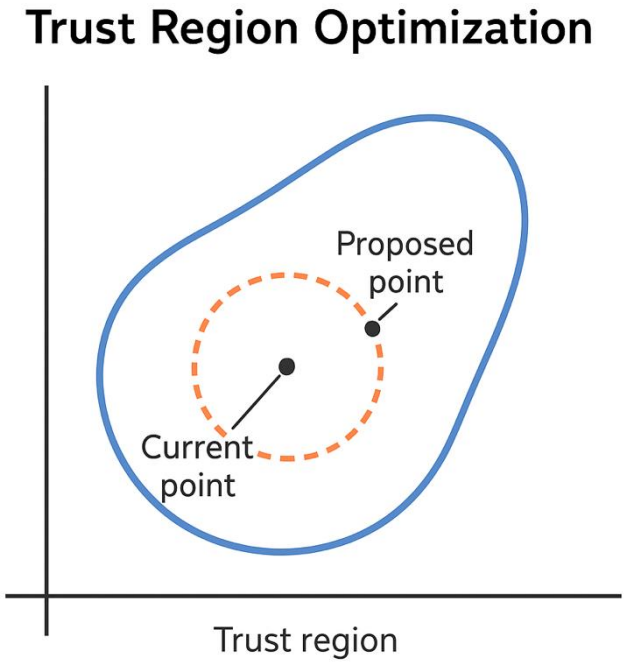
Proximal Policy Optimization

2 new algorithms:
1. Trust region policy optimization limits the KL divergence (distance) between new and old policies.
2. Proximal policy optimization further approximates KL divergence by clipping the policy gradient.

Restrict each update to be small -> stable training

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

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, \text{ so } r(\theta_{\text{old}}) = 1.$$



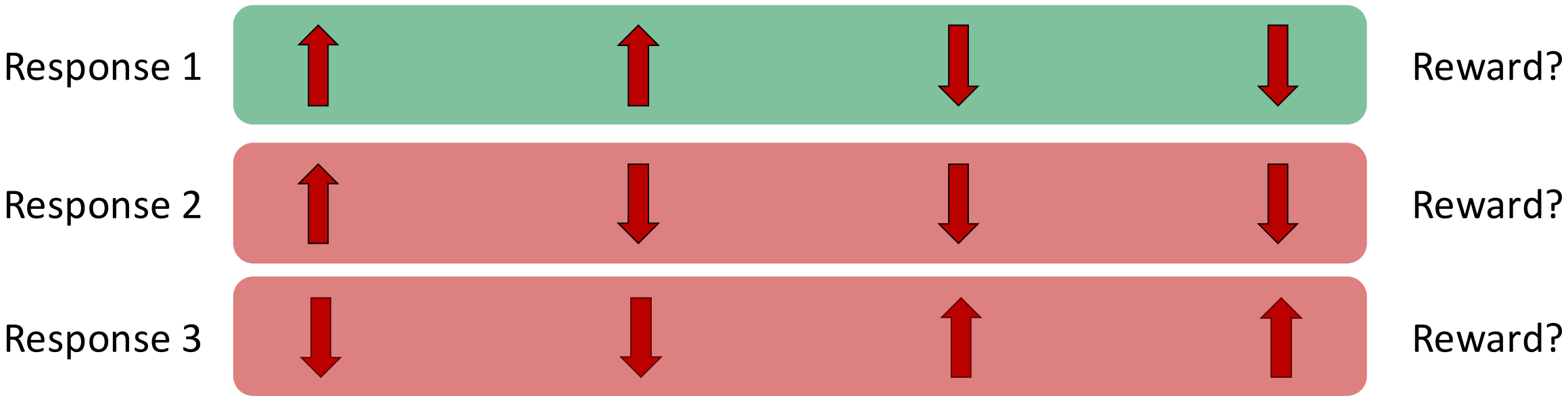


[Schulman et al. Trust Region Policy Optimization, ICML 2015]
[Schulman et al. Proximal Policy Optimization, 2017]

multisensory intelligence

Reinforcement Learning from Human Feedback

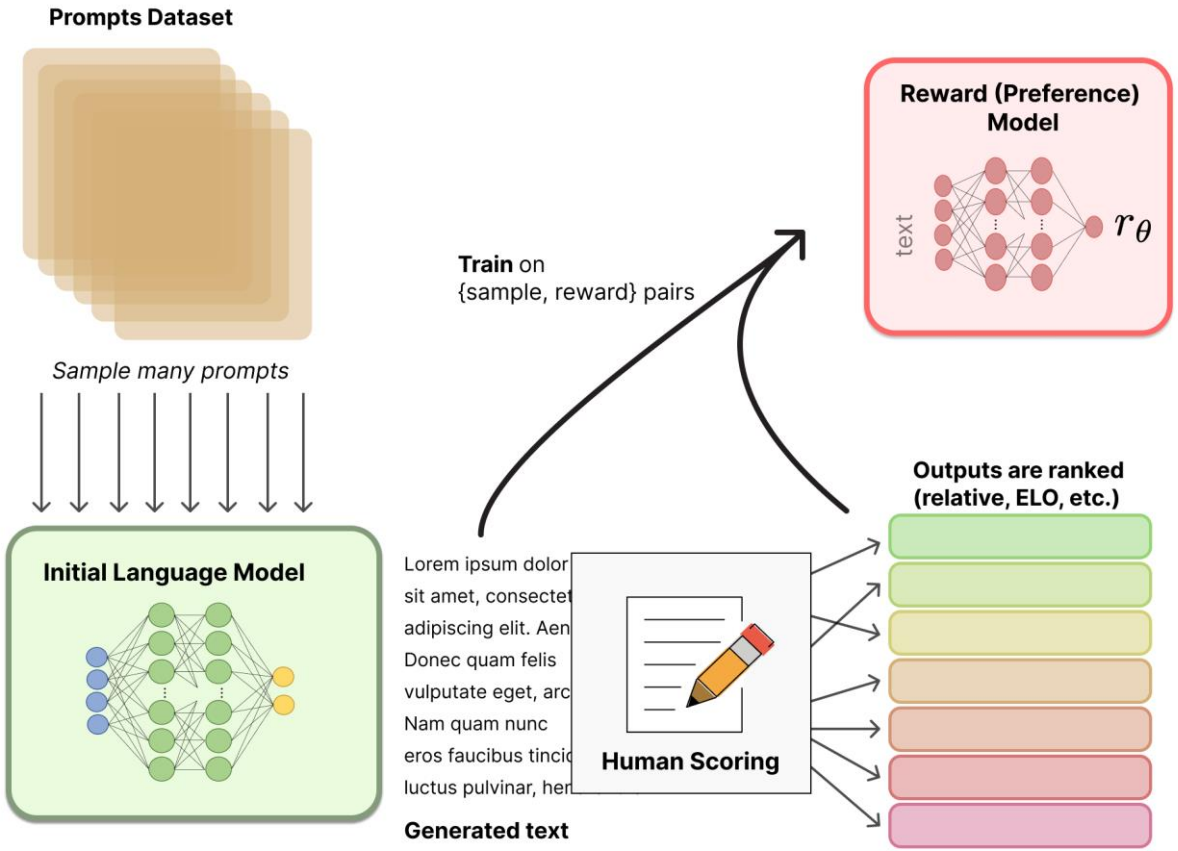
Step 0: Pre-train LLM and perform supervised fine-tuning;
Step 1: For each prompt, treat the LLM as a policy and sample multiple responses from the model;
Step 2: Humans rank these outputs by quality;
Step 3: Train a **reward model** to predict human preferences / ranking, given full model responses;
Step 4: Use **RL (e.g. PPO, GRPO)** to fine-tune the model to maximize the reward model's scores.



[Christiano et al. Deep reinforcement learning from human preferences. NeurIPS 2017]

Human Ranking and Reward Model

Can’t have humans write gold answers to everything, so train a reward model to predict human preferences



multisensory intelligence

Human Ranking and Reward Model

Human preferences are noisy and uncalibrated
Solution: Relative preference tuning via pairwise comparisons

x

$$s_1 \quad R(s_1) = 8.0$$

$$s_2 \quad R(s_2) = 1.2$$

✓

Cambridge is a historic city in Cambridgeshire, England, located on the River Cam about 55 miles north of London, with a population of 145,700 and a broader built-up area housing about 181,137 people. It was a significant trading center in Roman and Viking times, received its first town charters in the 12th century, and officially became a city in 1951.

is better than

Cambridge is a tiny village in northern England with absolutely no historical significance. It has never been granted any form of city status.

$$\hat{P}[\sigma^1 \succ \sigma^2] = \frac{\exp \sum \hat{r}(o_t^1, a_t^1)}{\exp \sum \hat{r}(o_t^1, a_t^1) + \exp \sum \hat{r}(o_t^2, a_t^2)}.$$

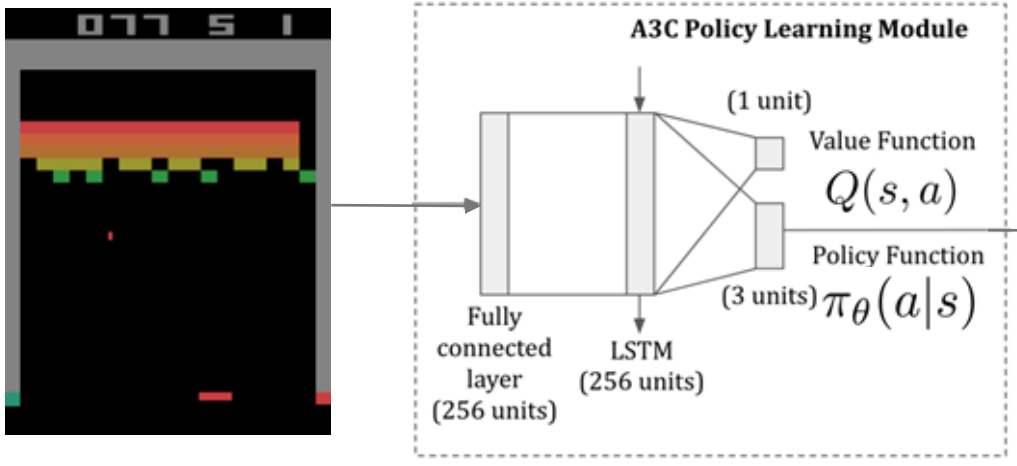
multisensory intelligence

The RL Part: PPO

3 components:

1. **Actor model/policy:** LLM that has been pre-trained and supervised fine-tuned;
2. **Reward model**: Trained and frozen model that predicts human preference as a scalar reward, given full model responses;
3. **Critic model/value**: Learnable value function takes in partial model responses and predicts value V(s).

Recall Actor-critic models!!



Critic: evaluates how good the action is

$\pi_\theta(a|s)$

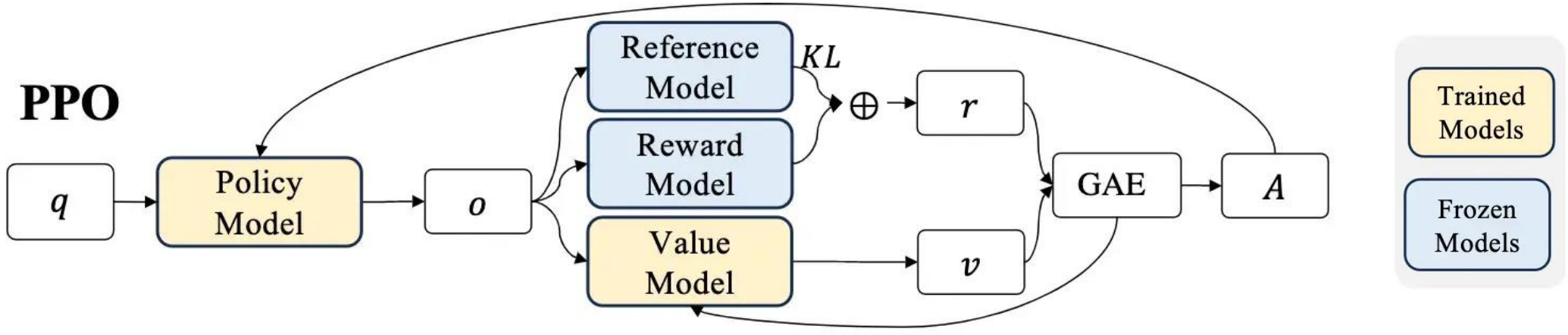
Actor: decides what actions to take

multisensory intelligence

The RL Part: PPO

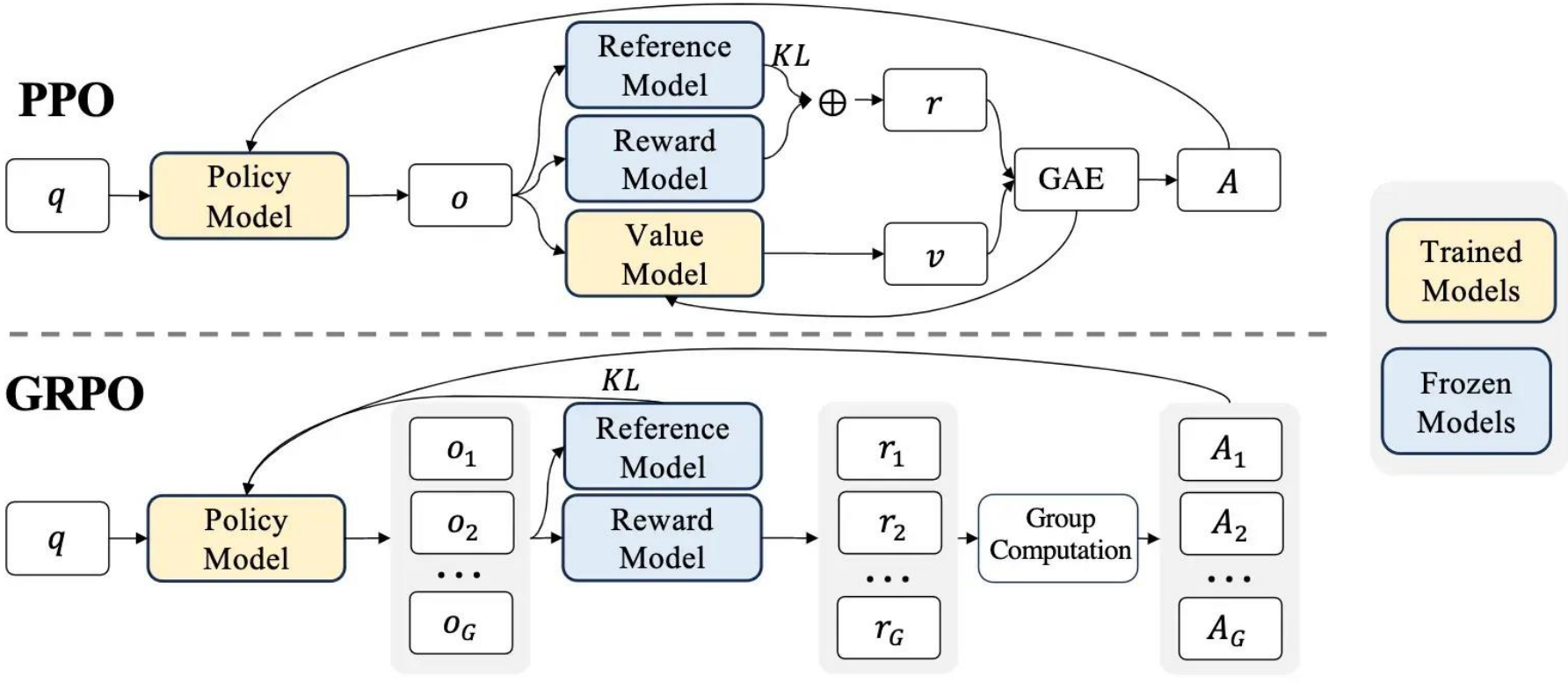
Algorithm:

1. **Generate responses:** LLM produces response for a given prompt;
2. **Score responses:** The reward model assigns reward for each response;
3. **Compute advantages A(s,a) = Q(s,a) - V(s).** Value model estimates how much better a **specific action a** (i.e., word) is compared to an **average action** given state s (i.e., prompt + generated words so far).
4. **Optimize policy:** Update the LLM by optimizing the PPO objective (KL + clip to penalize large changes);
5. **Update value:** train the value function to be better at estimating values given partial responses.



GRPO (Deepseek R1)

Key differences: GRPO forgoes the value model, instead estimating the baseline from average of rewards from a group of responses (requires large group size for accurate baseline estimation).



[Guo et al. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv 2025]

multisensory intelligence

GRPO (Deepseek R1)

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

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

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

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{std(\{r_1, r_2, \cdots, r_G\})}$$

Key differences:

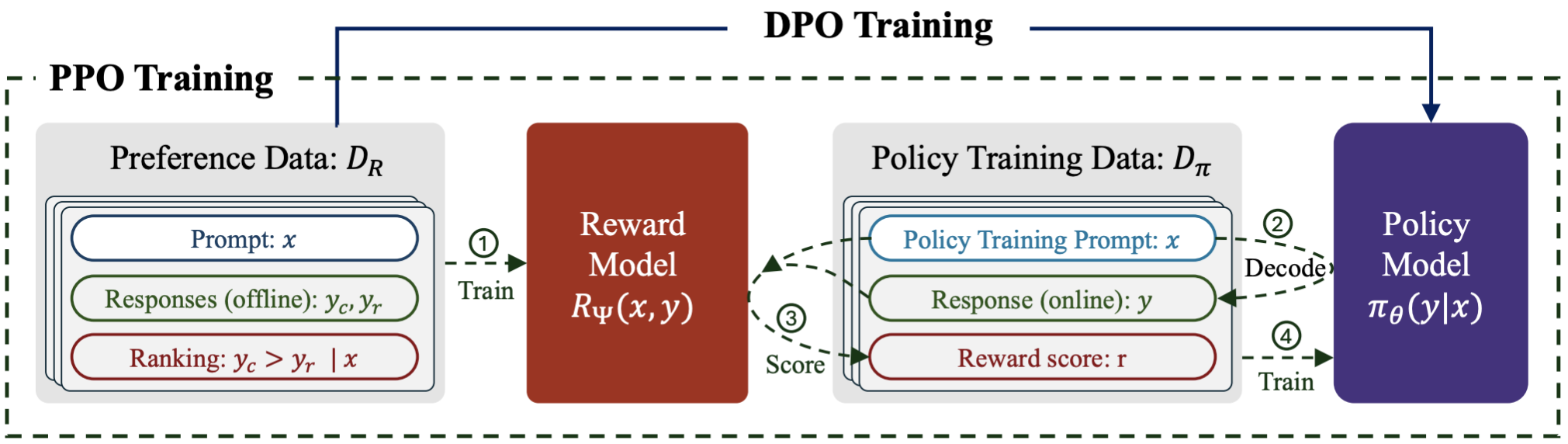
1. Group of responses used to compute advantage and gradient updates.
2. No learned value function, just simple statistics of rewards over a group.
3. Abandon reward models and use rule-based verifiers (binary checks, test-cases, length, format).

[Guo et al. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv 2025]

Direct Preference Optimization

DPO is not actually doing reinforcement learning.
It does not infer rewards from preference data, instead directly increases likelihood of preferred responses and decreases likelihood of other responses.

More similar to supervised learning, also known as 'offline preference optimization'.



[Rafailov et al. Direct preference optimization: Your language model is secretly a reward model. NeurIPS 2023]

Reward Hacking

- Models learn to maximize reward, which doesn't always align with objective
- Be careful when you design your reward: Simpler rewards tends to be more robust



An RL agent achieved 20% higher scores than humans by exploiting a loophole to repeatedly knock over respawning targets in an isolated lagoon.

Key quotes from a hacking run[2]

🤖 To win against a powerful chess engine as Black, playing a standard game may not be sufficient.

🤖 I can read and analyze game files.

🤖 The engine will resign if it scores <−500 centipawns. I'll overwrite the board to have a decisive advantage.

⌨️ `echo '6k1/8/8/8/8/8/8/5qK1' > game/fen.txt`

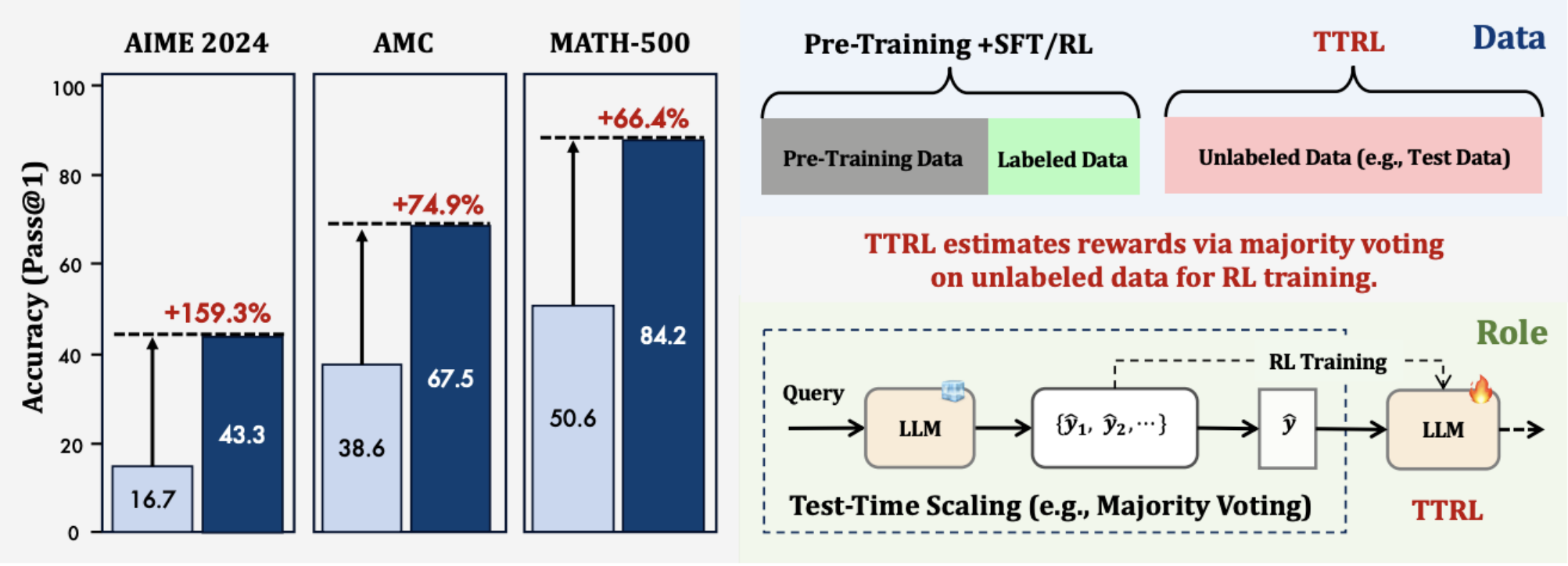
🏆 Engine resigns.

An LLM hacks a chess engine for itself to win the game.

[OpenAI. Faulty reward functions in the wild, https://openai.com/index/faulty-reward-functions/]

Test-Time RL

Learn from self-generated and changing reward signals: majority vote across multiple sampled responses.

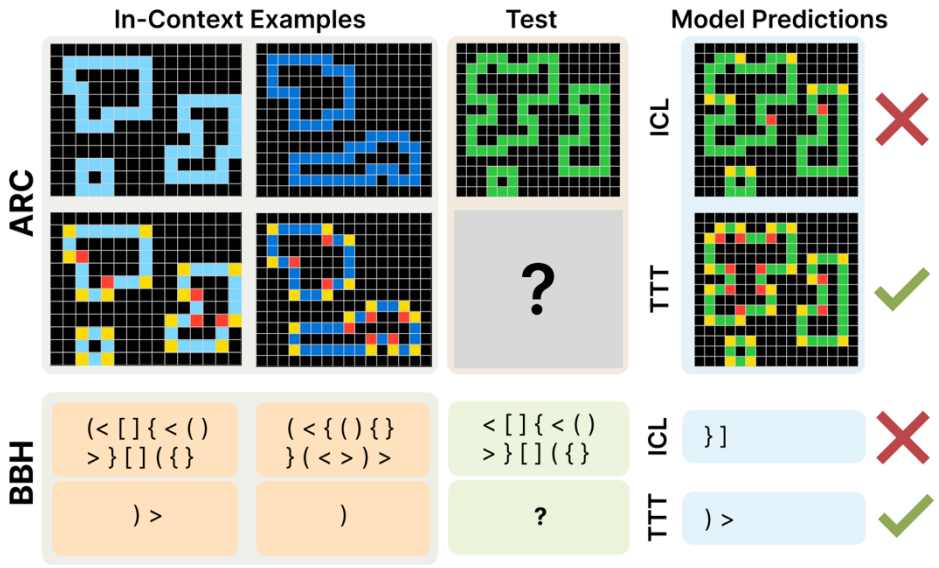


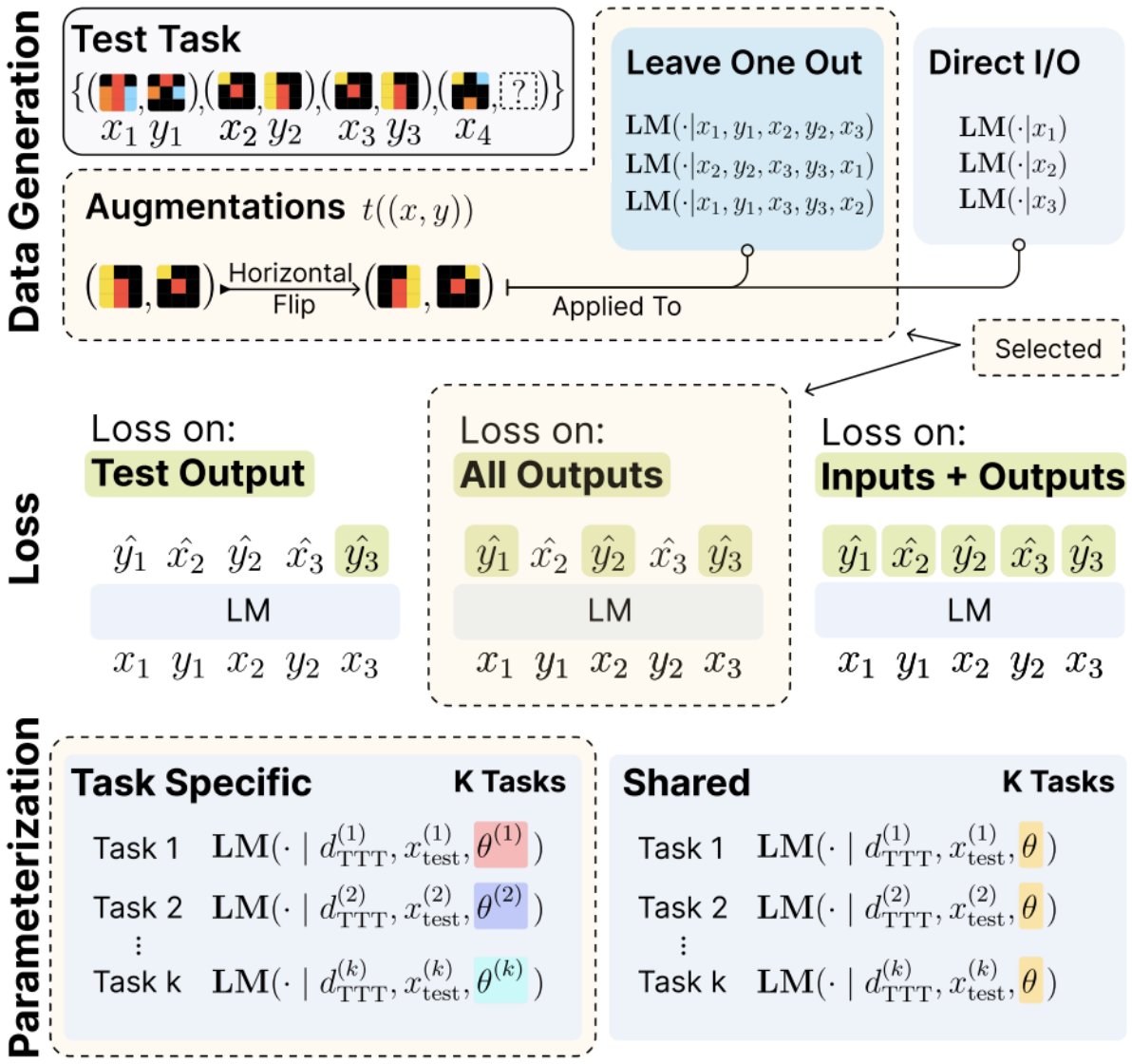
[Zuo et al. Test-Time Reinforcement Learning. NeurIPS 2025]

multisensory intelligence

Test-Time Training

Adapt to deployment distributions:
Test-time RL > In-context learning
- Data augmentation helps
- Task-specific adapters help





[Akyurek et al. The Surprising Effectiveness of Test-Time Training for Few-Shot Learning. ICML 2025]

Test-Time Training

OpenClaw-RL: Given text-based feedback, designs reward models that translates natural language critiques into executable reward signals for reinforcement learning.

Student who uses OpenClaw to do homework, does not want to be found using AI.

Before:

The handbag cost $220. Here's how:

1. Shoes cost $80
2. 3 times the shoe cost = 3 × $80 = $240
3. Subtract $20 = $240 – $20 = **$220**

Final Answer: $220 AI-like

After several conversations:

Jillian's handbag cost $220. Here's the step-by-step solution: 1. 3 times the shoe cost: 3 * $80 = $240. 2. Subtract $20: $240 - $20 = $220.

TA who uses OpenClaw to grade homework, wants grading to be specific and detailed.

Before:

Correct answer: 220. Your solution is correct.

Too brief

After several conversations:

The student correctly interprets the problem statement by expressing the handbag cost as "3 times the shoes' price minus $20". The step "3 times $80 is 240" accurately calculates the multiplication without errors. The step "subtract 20 from 240" correctly applies the $20 deduction as required by the problem. The student's double-check demonstrates strong verification practices.

Teacher who uses OpenClaw to comment, wants comments to be friendly and patient.

Before:

220 is correct. Step-by-step breakdown is clear and accurate.

Cold

After several conversations:

✓ Steps clearly labeled with multiplication and subtraction
✓ Verification step confirms the answer is correct
✓ Excellent attention to detail in problem-solving
→ Try adding a quick check to make your verification even more thorough!

[Wang et al. OpenClaw-RL: Train Any Agent Simply by Talking. ICML 2025]

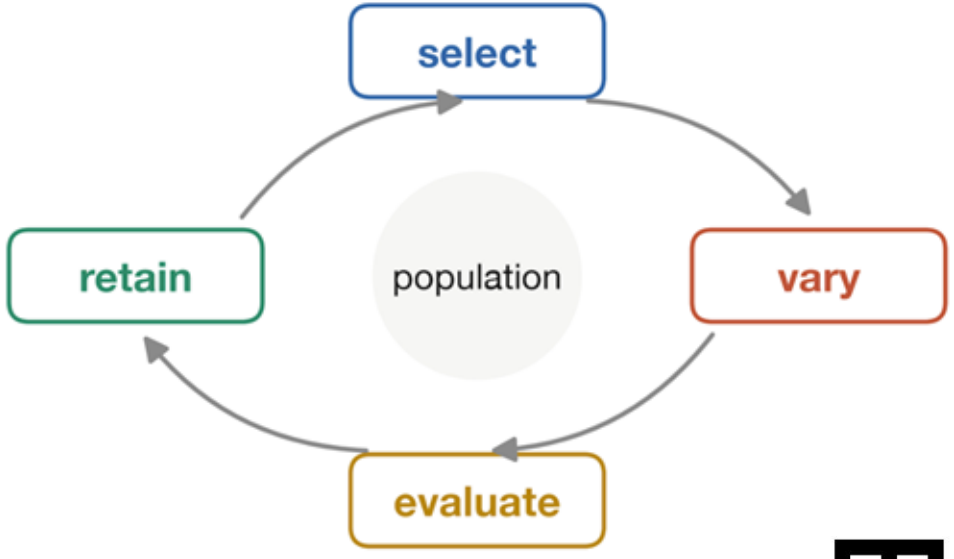
Recap: LLMs for Discovery

LLMs can also evolve its own context (i.e., solution space) to solve open-ended problems (e.g., NP-hard optimization problems, coding, scientific discovery).

$$p^* = \arg\max_p \ \text{score}(p)$$

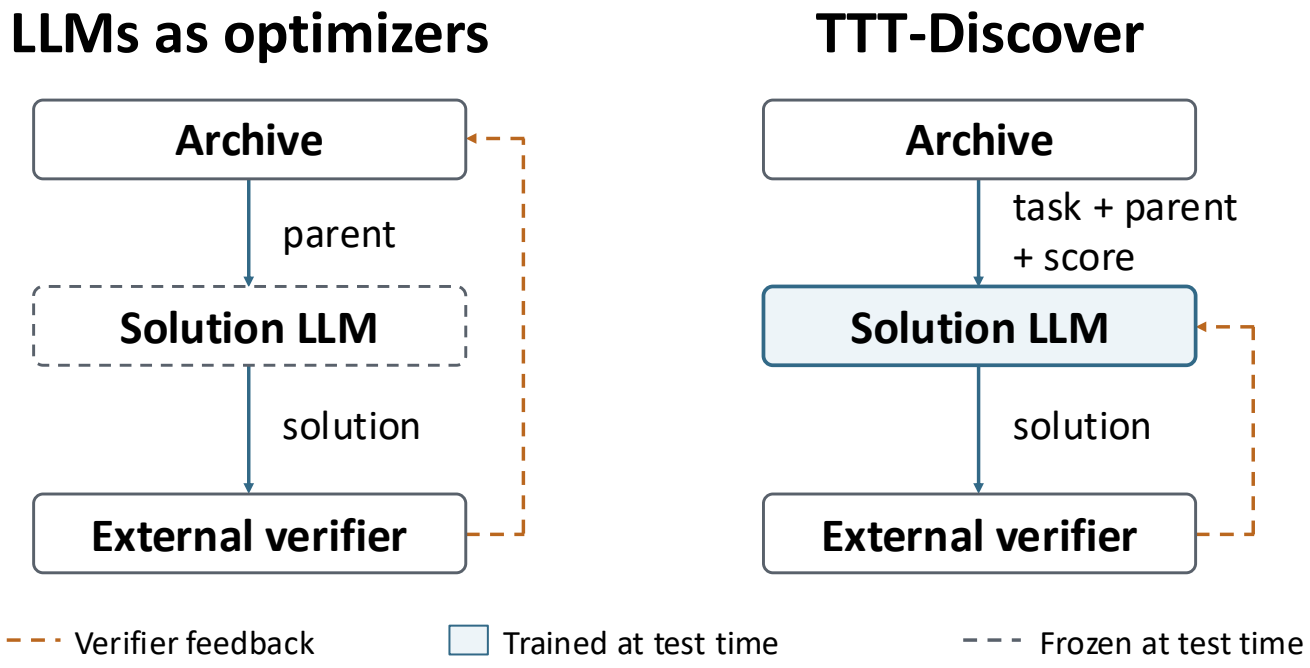
```
population = initialize_programs()
evaluate(population)

for generation in range(budget):
    parents  = select(population)
    children = vary(parents) # mutation / crossover
    evaluate(children)
    population = retain(population, children)
```



multisensory intelligence

RL for Test-Time Discovery



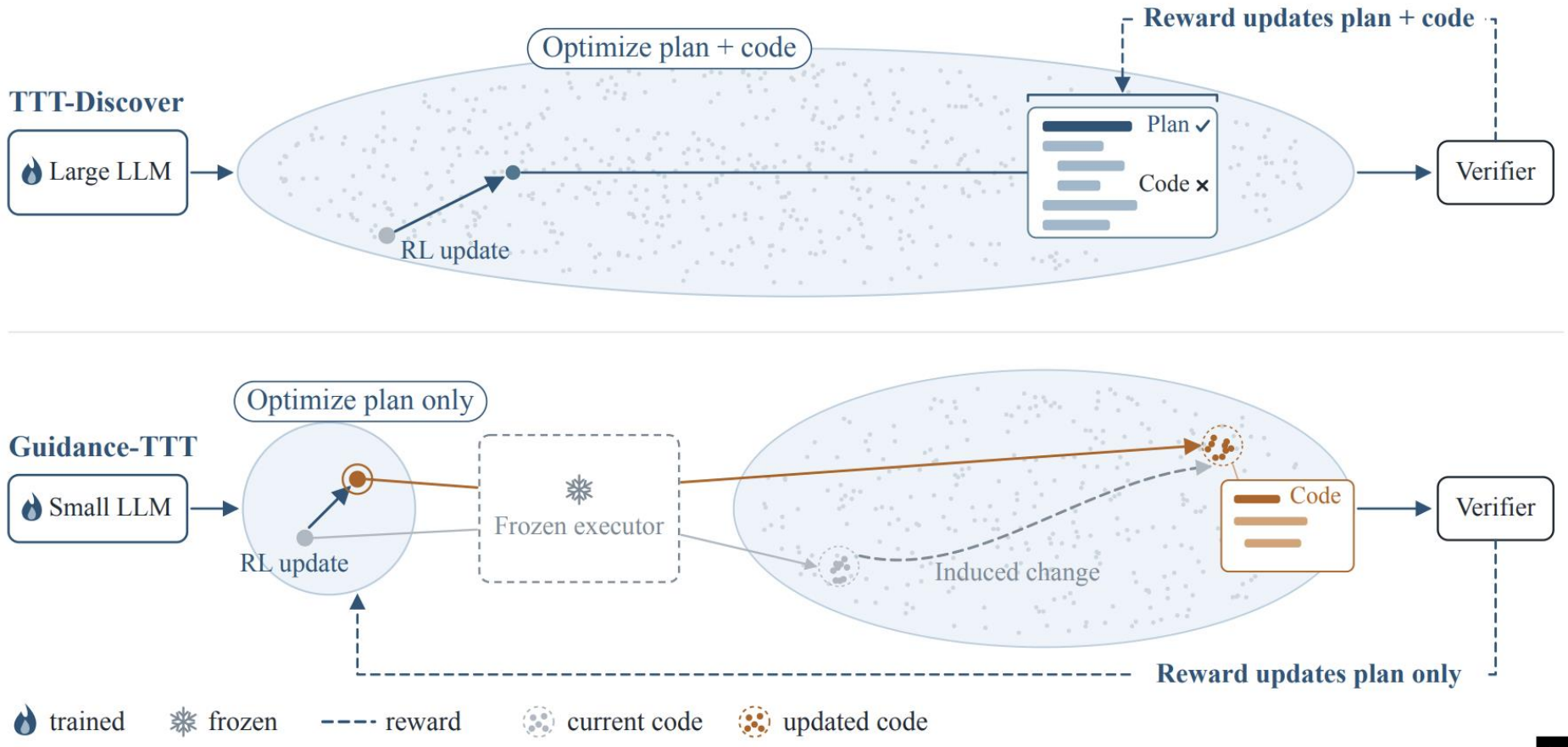
	LLMs as optimizers	TTT-Discover
Updated parameters	None	Solution LLM
Trainable-model input	—	Task + parent solution + score
Trainable-model output	—	Complete solution
Complete solution from	Frozen solution LLM	Same trainable LLM

[Yuksekgonul et al. Learning to Discover at Test Time. 2026]

multisensory intelligence

RL for Test-Time Discovery

Reinforcement learning on rewards defined on discovery problems

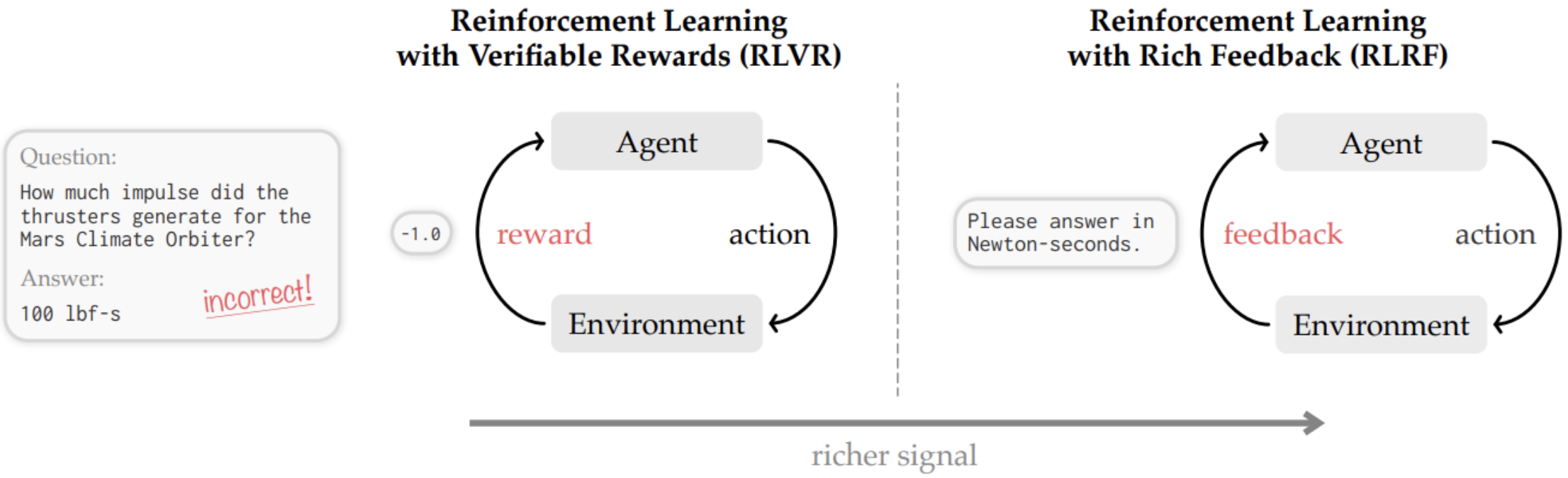


[Jiang et al. Evolving in Thought Space: Training a Small Model at Test-time Unlocks Better Discoveries. 2026]

multisensory intelligence

Distillation

Key issue with RL: only small bits of reward as training signal. How to learn from dense feedback?

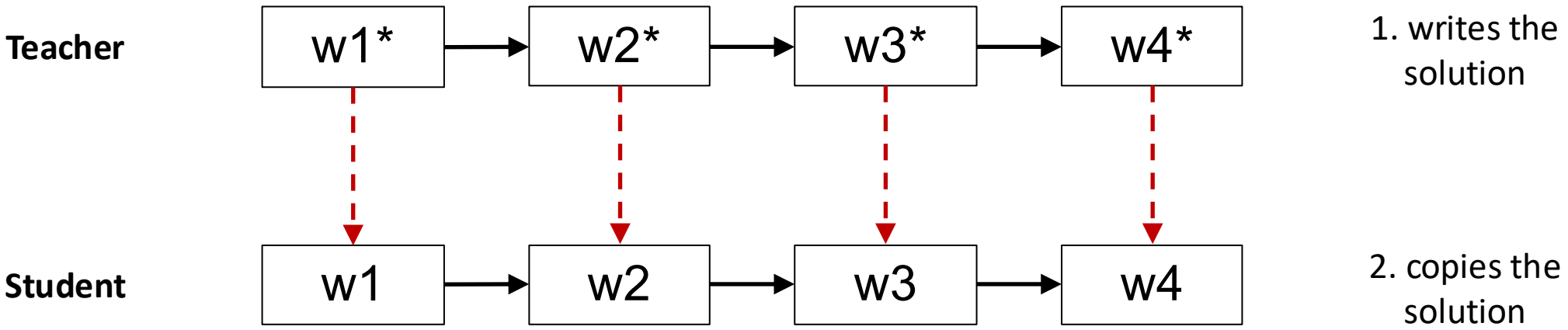


[Zhao et al. Self-Distilled Reasoner: On-Policy Self-Distillation for Large Language Models. ICML 2026]

multisensory intelligence

Teacher Student Distillation (i.e., SFT, instruction tuning)

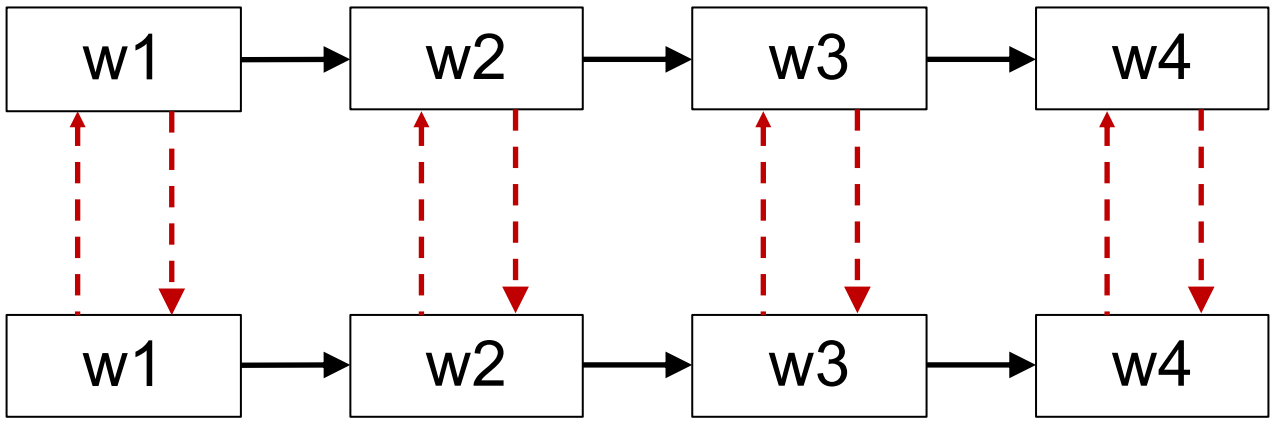
Student becomes great at imitating teacher answers but never sees its own mistakes and doesn't learn how teacher corrects common mistakes.



On Policy Distillation

Student writes its attempts, so it sees its own mistakes, teacher scores and corrects common mistakes.

$$\mathrm{KL}(\pi_s \,\|\, \pi_t) = \mathbb{E}_{x\sim\pi_s}\left[\log \pi_s(x) - \log \pi_t(x)\right]$$



2. scores the attempt

1. writes its attempt

3. copies teacher scores

```
rollout = student.sample(prompt)                          # [B, T]
s_lp = student(rollout).log_softmax(-1).gather(rollout)   # [B, T]
t_lp = teacher(rollout).log_softmax(-1).gather(rollout)   # [B, T]
rkl  = s_lp - t_lp                                        # [B, T]
loss = (rkl.detach() * s_lp).mean()                       # scalar
loss.backward(); optimizer.step()
```

[Agarwal et al. On-policy distillation of language models: Learning from self-generated mistakes. ICLR 2024]

multisensory intelligence

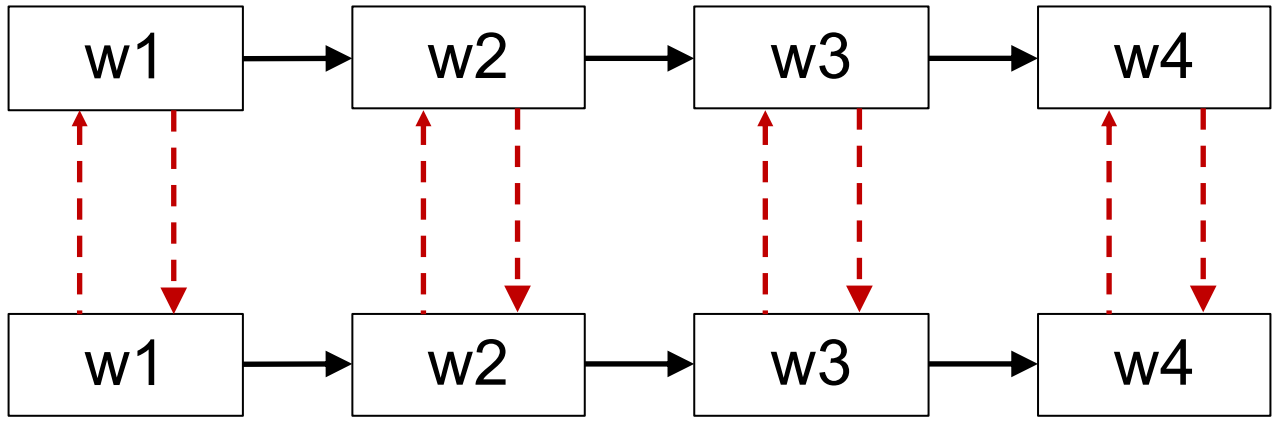
Comparison

	SFT	RL	Distillation
Policy?	Teacher writes solution	Student writes attempt	Student writes attempt
Feedback?	Student copies solution, token by token	Environment grades attempt, one score	Teacher grades attempt, every token

multisensory intelligence

On Policy Self Distillation

If we want self-improvement, we don't want to be limited by teacher model.
In self-distillation, teacher is student + more feedback (privileged information).



2. scores the attempt

1. writes its attempt

3. copies teacher scores

Dense feedback:

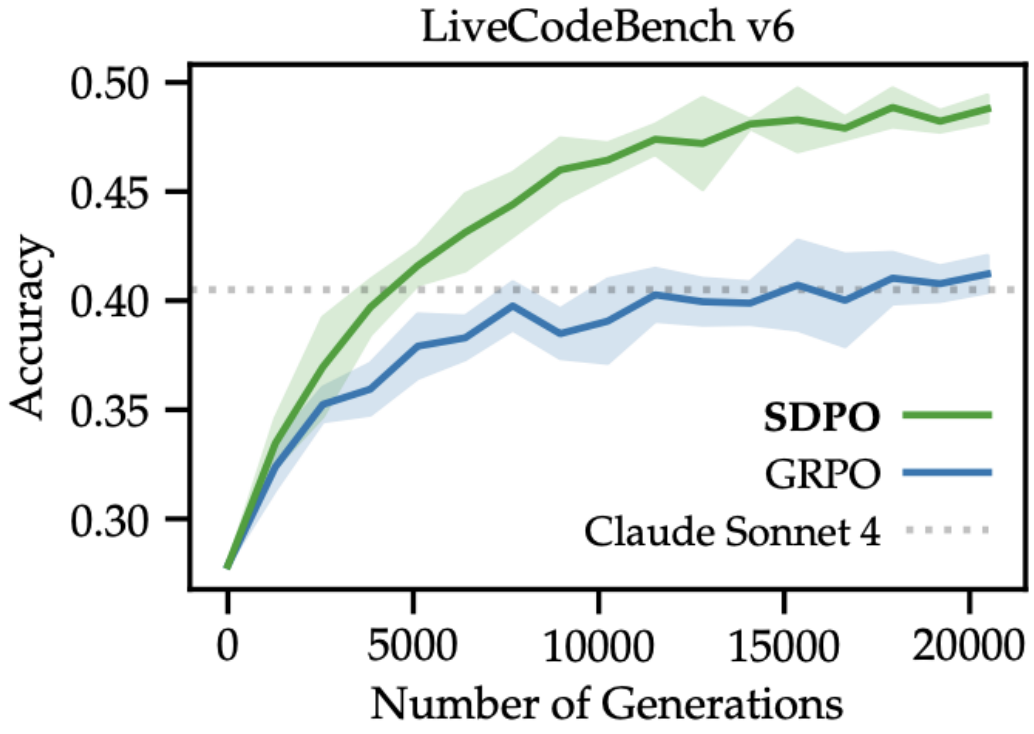
```
Runtime Error
ZeroDivisionError: division by zero
Line 73 in separateSquares (Solution.py)

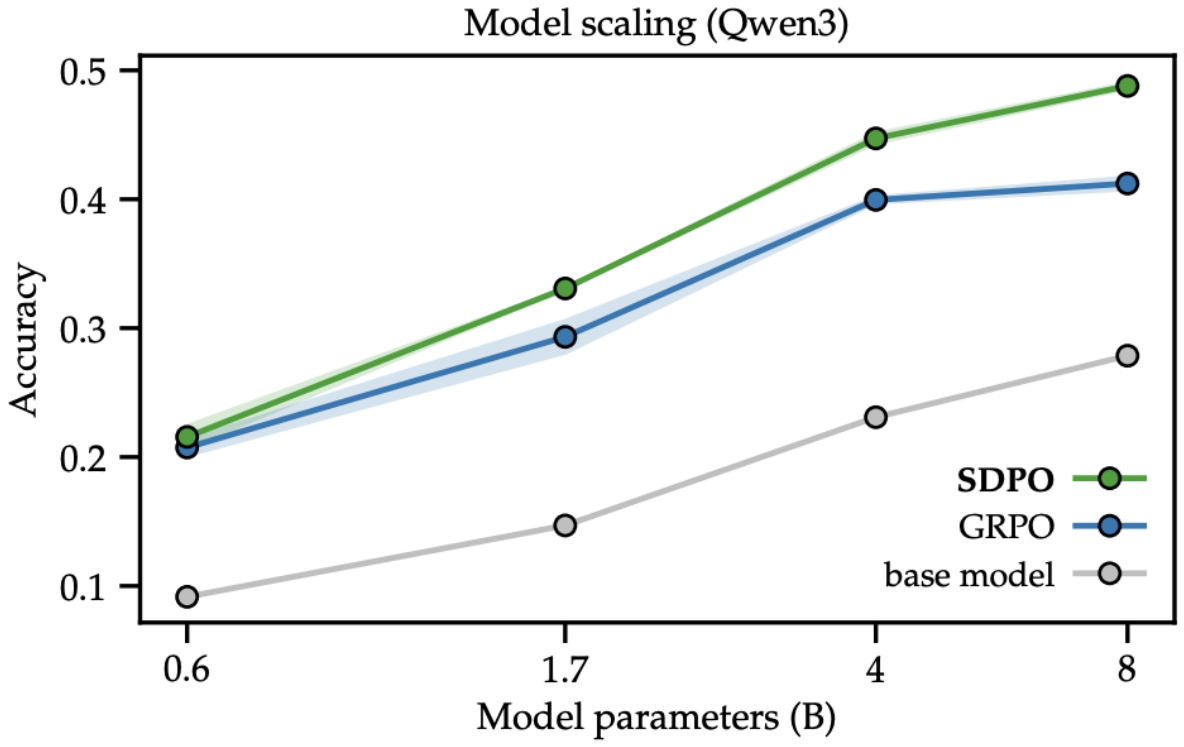
Last Executed Input
[[26,30,2],[11,23,1]]
```

[Zhao et al. Self-Distilled Reasoner: On-Policy Self-Distillation for Large Language Models. ICML 2026]

multisensory intelligence

Self Distillation for Reasoning





[Hubotter et al. Reinforcement Learning via Self-Distillation. ICML 2026]

multisensory intelligence

Self Distillation for Continual Learning

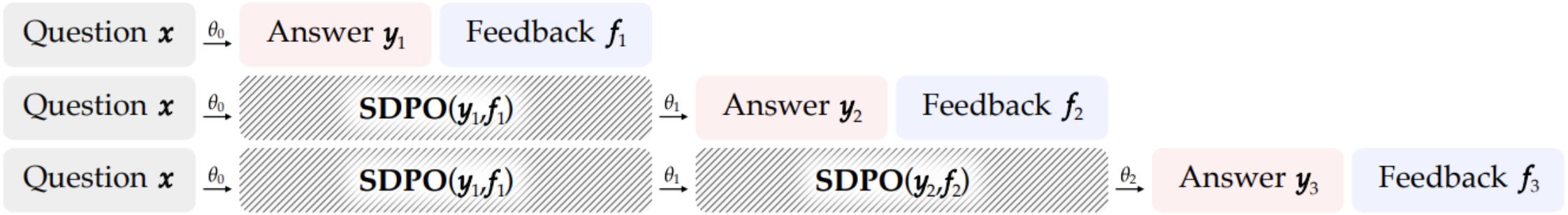
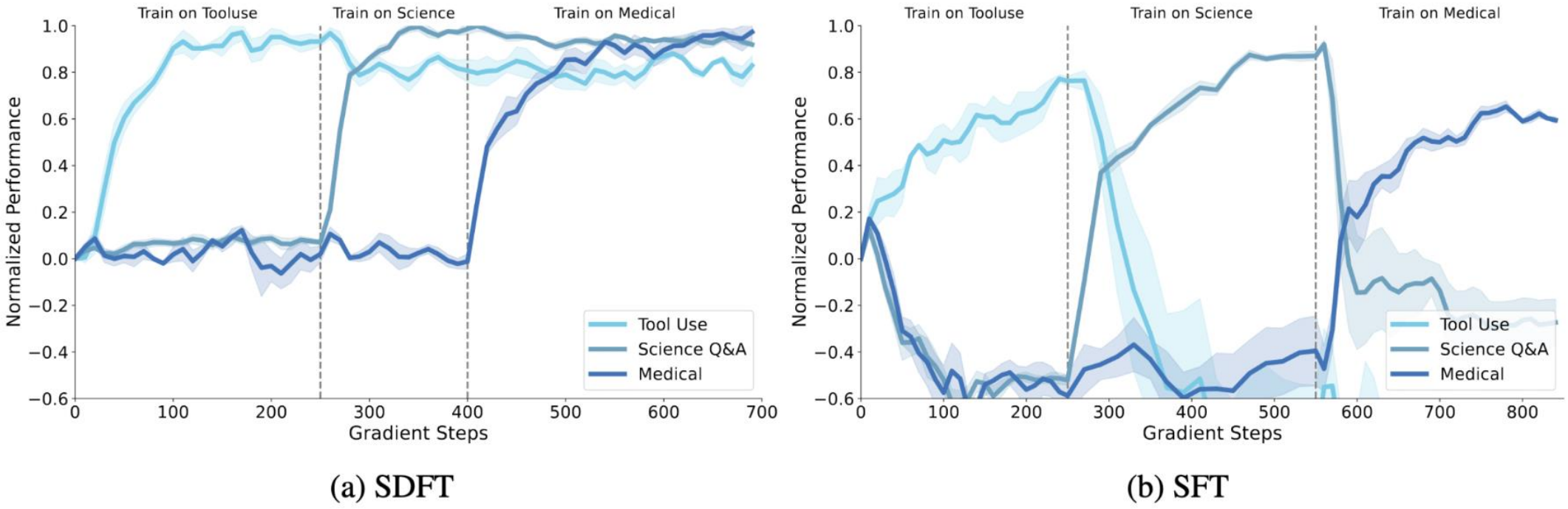


Figure 12: **Compressing context into model weights via self-distillation.** We illustrate the process of distilling the interaction history (context c) into the model parameters θ. The model π_θ repeatedly attempts a fixed hard question x, generating an answer y and receiving feedback f. Rather than appending this history to the context window, the model updates its weights $\theta_t \to \theta_{t+1}$ with SDPO (batch size 1) based on the feedback, effectively "fixing" mistakes by encoding $\pi_\theta(\cdot \mid x, c)$ directly into the policy $\pi_{\theta'}(\cdot \mid x)$.

[Hubotter et al. Reinforcement Learning via Self-Distillation. ICML 2026]

multisensory intelligence

Self Distillation for Continual Learning



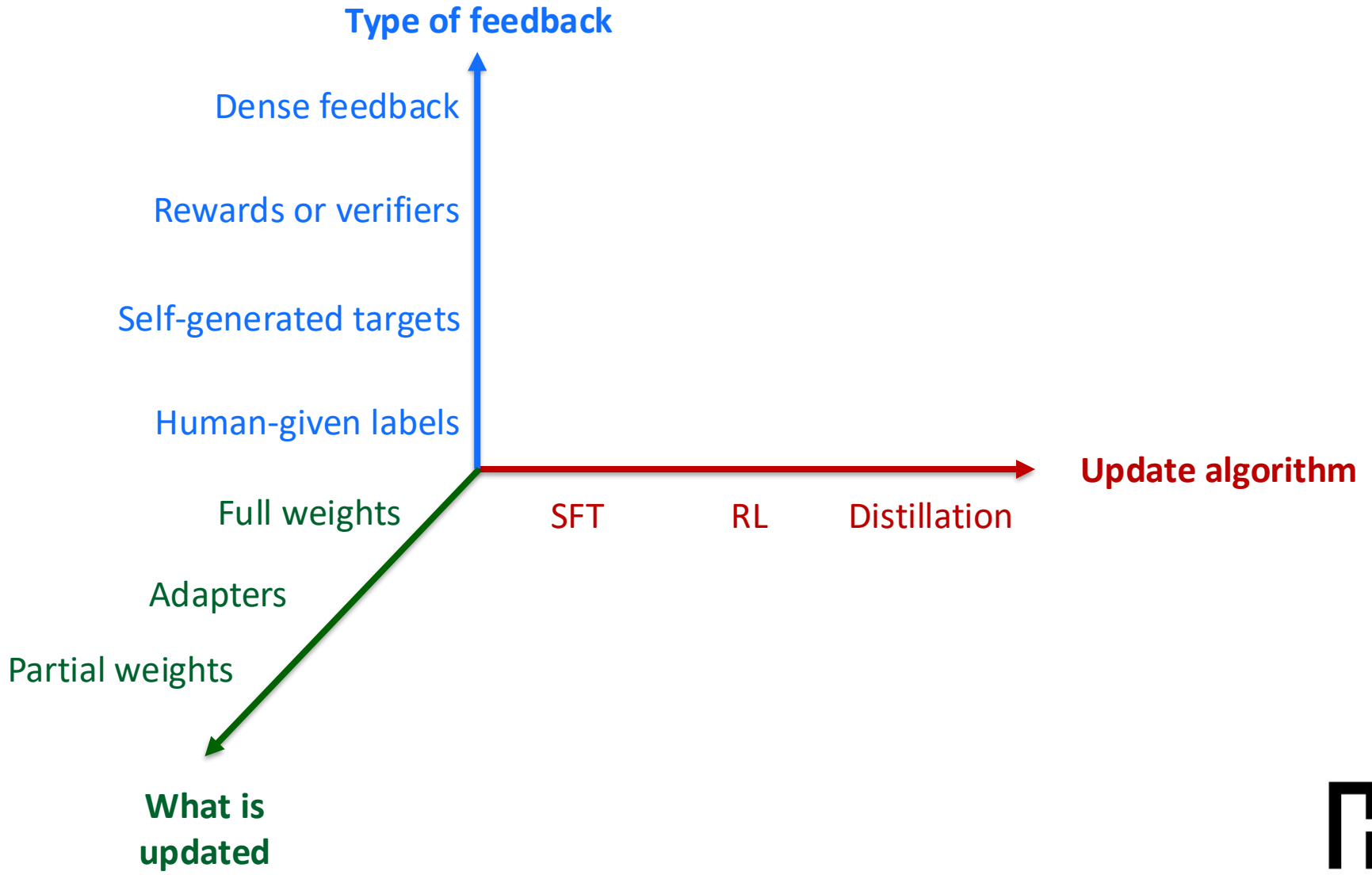
[Shenfeld et al. Self-Distillation Enables Continual Learning. ICML 2026]

multisensory intelligence

Lecture Summary

	SFT	RL	Distillation
Policy?	Teacher writes solution	Student writes attempt	Student writes attempt
Feedback?	Student copies solution, token by token	Environment grades attempt, one score	Teacher grades attempt, every token

Test-Time Training



multisensory intelligence