

# Reinforcement Learning and Recent Advances in Agentic AI

Dam Quang Tuan

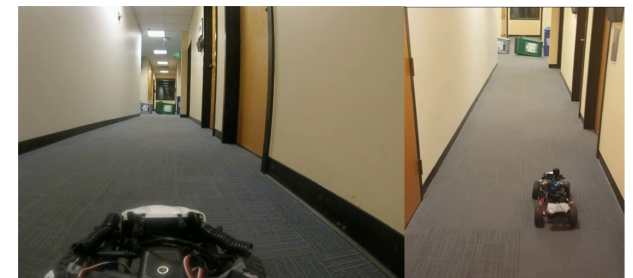
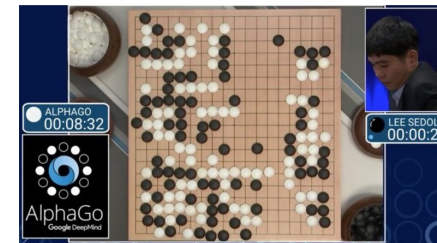
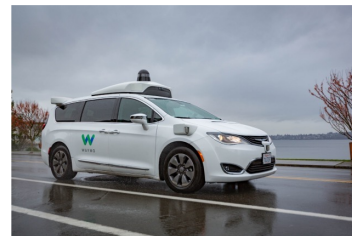
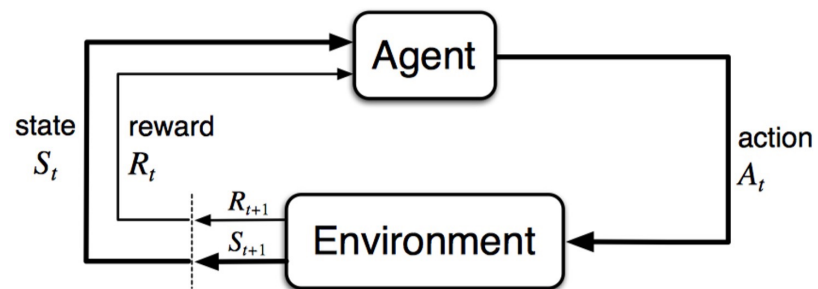


SOICT

# Learning a Policy – RL basics

## Reinforcement learning

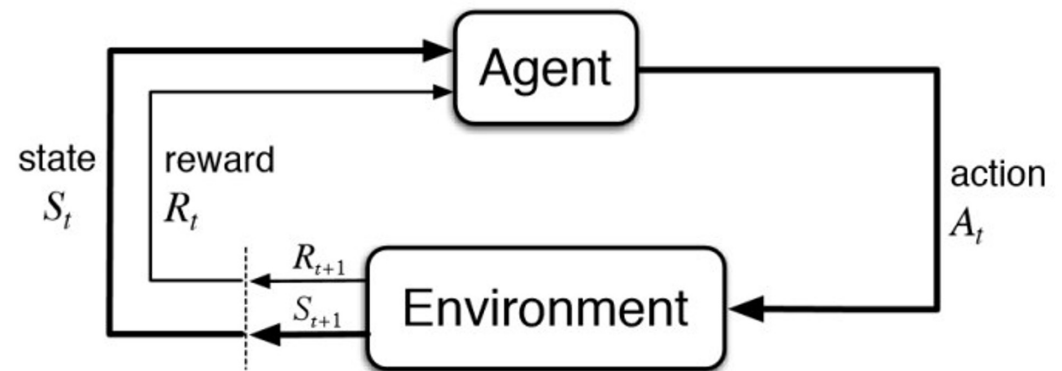
- Introduction to RL
- Markov Decision Processes (MDPs)
- Solving known MDPs using value and policy iteration
- Solving unknown MDPs using function approximation and Q-learning



# 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  $\gamma$ .
- Horizon  $H$ .

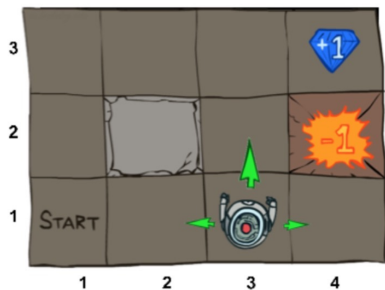


**Return:**

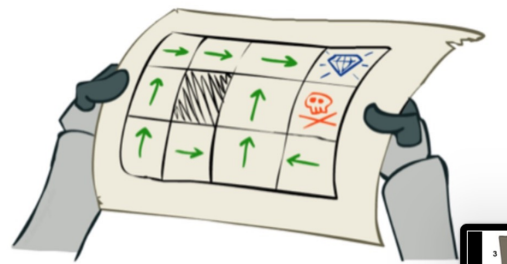
$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

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

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



$\pi$ :



# RL vs Supervised Learning

---

## Reinforcement Learning

- Sequential decision making
- Maximize cumulative reward
- Sparse rewards
- Environment maybe unknown



## Supervised Learning

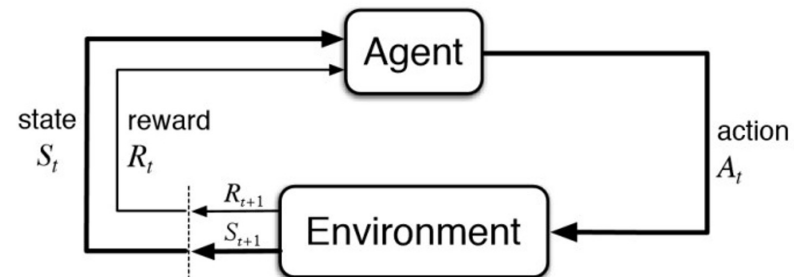
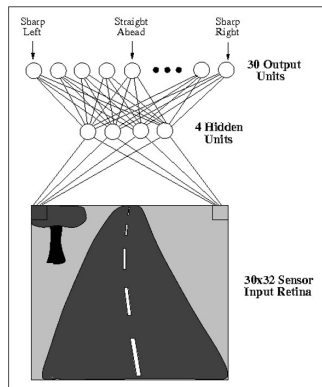
- One-step decision making
- Maximize immediate reward
- Dense supervision
- Environment always known





# Intersection Between RL and Supervised Learning

## Imitation learning



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

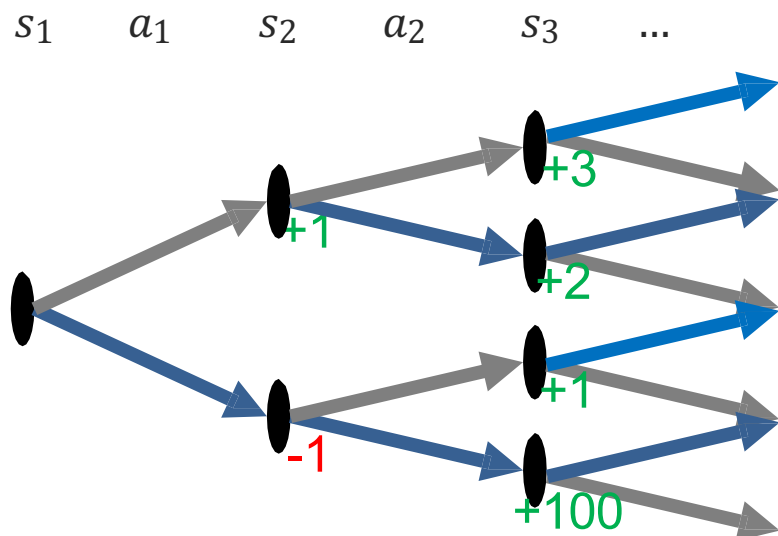
$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \dots$$

**Perform supervised learning by predicting expert action**

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

**But: distribution mismatch between training and testing**  
**Hard to recover from sub-optimal states**  
**Sometimes not safe/possible to collect expert trajectories**

## RL as Exploring a Tree



$\pi$  which action to take from each  $s$

$$V^\pi(s) = \mathbb{E}_\pi [G_t | S_t = s] \quad V^*(s) = \max_\pi V^\pi(s)$$

State-value function: how much total reward should I expect following  $\pi$  from  $s$ ?

$$V^\pi(s_1) = 99$$

$$V^\pi(s_1) = 99$$

$$Q^\pi(s, a) = \mathbb{E}_\pi [G_t | S_t = s, A_t = a] \quad Q^*(s, a) = \max_\pi Q^\pi(s, a)$$

Action-value function: how much total reward should I expect taking  $a$ , then following  $\pi$ , from  $s$ ?

$$Q^\pi(s_1, up) = 3$$

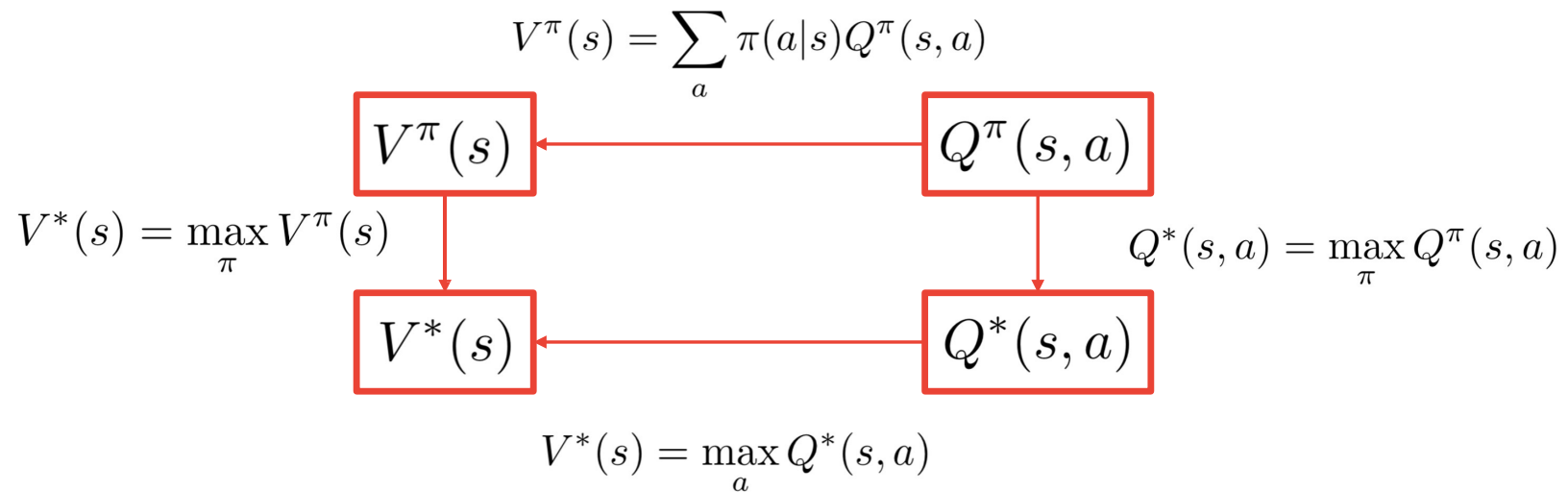
$$Q^*(s_1, up) = 4$$

# Relationships Between State and Action Values

---

State value functions

Action value functions



# Value-based Methods

---

## Value Based

- Learned Value Function
- Implicit policy (e.g.  $\epsilon$ -greedy)

### State value functions

$$\begin{array}{l} V^\pi(s) \\ V^*(s) \end{array}$$

### Action value functions

$$\begin{array}{l} Q^\pi(s, a) \\ Q^*(s, a) \end{array}$$

Optimal policy can be found by maximizing over  $Q^*(s, a)$

$$\pi^*(a|s) = \begin{cases} 1 - \epsilon, & \text{if } a = \arg \max_a Q^*(s, a) \\ \epsilon, & \text{else} \end{cases}$$

Optimal policy can also be found by maximizing over  $V^*(s')$   
with **one-step look ahead**

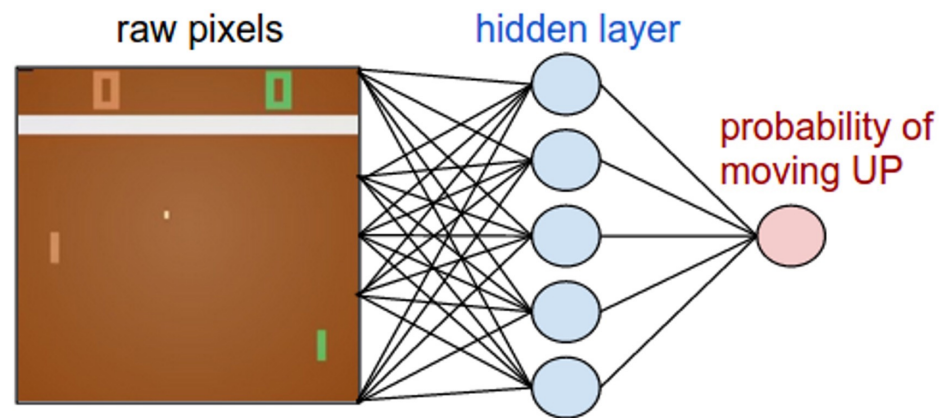
$$\pi^*(a|s) = \begin{cases} 1 - \epsilon, & \text{if } a = \arg \max_a \mathbb{E}_{s'} [r(s, a, s') + \gamma V^*(s')] \\ \epsilon, & \text{else} \end{cases}$$

# Policy-based Methods

- Policy Based

- No Value Function
- Learned Policy

$$\pi_{\theta}(s, a) = \mathbb{P}[a \mid s, \theta]$$



- Often  $\pi$  can be simpler than Q or V

- E.g., robotic grasp

$Q(s,a)$  and  $V(s)$  very high-dimensional  
But policy could be just 'open/close hand'

- V: doesn't prescribe actions

- Would need dynamics model (+ compute 1 Bellman back-up)

- Q: need to be able to efficiently solve  $\arg \max_a Q^*(s, a)$

- Challenge for continuous / high-dimensional action spaces

## Value-based vs Policy-based

---

$$\pi^*(a|s) = \begin{cases} 1 - \epsilon, & \text{if } a = \arg \max_a Q^*(s, a) \\ \epsilon, & \text{else} \end{cases}$$

### Value-based

- More sample efficient, respects MDP structure
- Easier to add human knowledge about states and actions
- More complex algorithm
- Can't handle continuous argmax, harder to understand, sometimes values are more complex than policies

$$\pi_\theta(s, a) = \mathbb{P}[a \mid s, \theta]$$

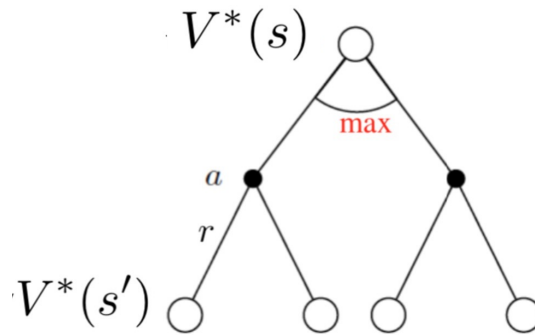
### Policy-based

- Less sample efficient, more akin to trial-and-error
- Harder to add human knowledge
- Simpler algorithm
- Directly learns policy, can be more interpretable

# Policy-based RL

---

## Recursive definition

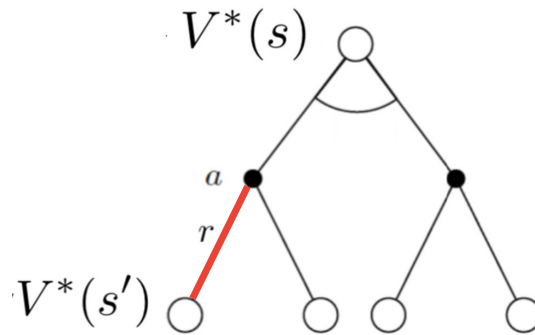


$$V^*(s) = \max_a Q^*(s, a)$$

# Bellman Optimality for State Value Functions

---

## Recursive definition



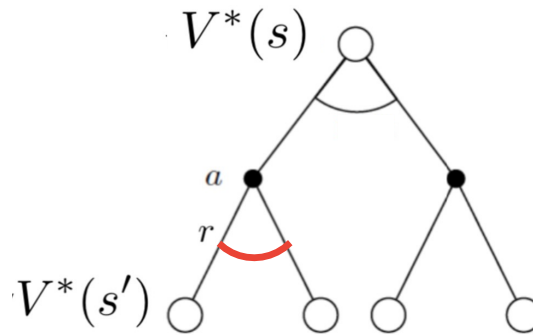
$$\begin{aligned} V^*(s) &= \max_a Q^*(s, a) \\ &= \max_a \mathbb{E}_{s'} [r(s, a, s') + \gamma V^*(s')] \end{aligned}$$



# Bellman Optimality for State Value Functions

---

## Recursive definition

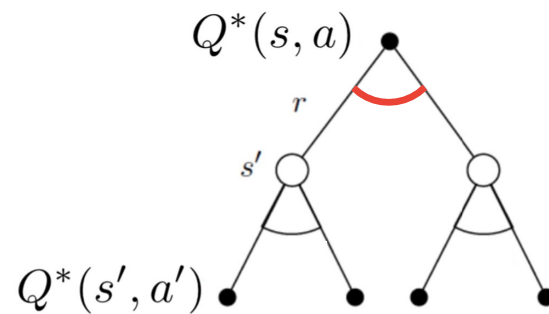


$$\begin{aligned} V^*(s) &= \max_a Q^*(s, a) \\ &= \max_a \mathbb{E}_{s'} [r(s, a, s') + \gamma V^*(s')] \\ &= \max_a \left[ \sum_{s'} p(s'|s, a) (r(s, a, s') + \gamma V^*(s')) \right] \end{aligned}$$

# Bellman Optimality for Action Value Functions

---

## Recursive definition



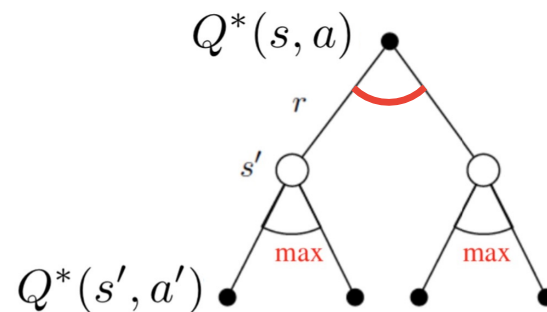
$$Q^*(s, a) = \mathbb{E}_{s'} [r(s, a, s') + \gamma V^*(s')]$$

---

# Bellman Optimality for Action Value Functions

---

## Recursive definition



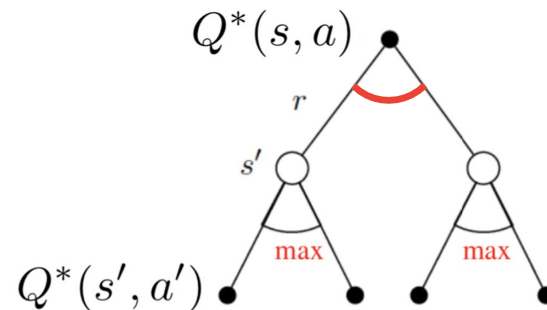
$$\begin{aligned} Q^*(s, a) &= \mathbb{E}_{s'} [r(s, a, s') + \gamma V^*(s')] \\ &= \mathbb{E}_{s'} \left[ r(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right] \end{aligned}$$

---

# Bellman Optimality for Action Value Functions

---

## Recursive definition



$$\begin{aligned} Q^*(s, a) &= \mathbb{E}_{s'} [r(s, a, s') + \gamma V^*(s')] \\ &= \mathbb{E}_{s'} \left[ r(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right] \\ &= \sum_{s'} p(s'|s, a) \left( r(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right) \end{aligned}$$

# Solving the Bellman Optimality Equations

---

## Recursive definition

$$V^*(s) = \max_a \left[ \sum_{s'} p(s'|s, a) (r(s, a, s') + \gamma V^*(s')) \right]$$

Solve by iterative methods

$$V_{[k+1]}^*(s) = \max_a \left[ \sum_{s'} p(s'|s, a) (r(s, a, s') + \gamma V_{[k]}^*(s')) \right]$$

# Value Iteration

---

Algorithm:

Start with  $V_0^*(s) = 0$  for all  $s$ .

For  $k = 1, \dots, H$ :

For all states  $s$  in  $S$ :

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma V_{k-1}^*(s'))$$

# Value Iteration

---

Algorithm:

Start with  $V_0^*(s) = 0$  for all  $s$ .

For  $k = 1, \dots, H$ :

For all states  $s$  in  $S$ :

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma V_{k-1}^*(s'))$$

$$\pi_k^*(s) \leftarrow \arg \max_a \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma V_{k-1}^*(s'))$$

Find the best action according to one-step look ahead

This is called a **value update** or **Bellman update/back-up**

**Repeat until policy converges. Guaranteed to converge to optimal policy.**

## Q-Value Iteration

---

$Q^*(s, a)$  = expected utility starting in  $s$ , taking action  $a$ , and (thereafter) acting optimally

Bellman Equation:

$$Q^*(s, a) = \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma \max_{a'} Q^*(s', a'))$$

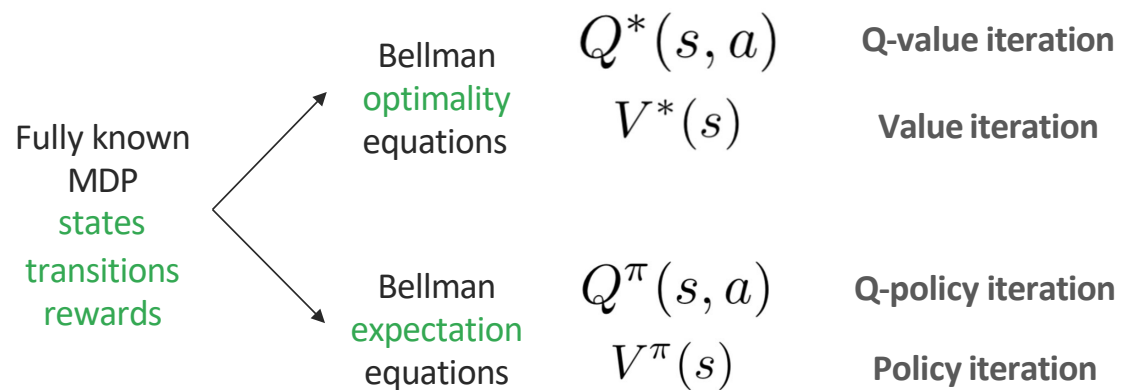
Q-Value Iteration:

$$Q_{k+1}^*(s, a) \leftarrow \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma \max_{a'} Q_k^*(s', a'))$$



## Summary: Exact Methods

---



Repeat until policy converges. Guaranteed to converge to optimal policy.

### Limitations:

Iterate over and store for all states and actions: requires small, discrete state and action space  
Update equations require fully observable MDP and known transitions

## Unknown MDPs?

---

$Q^*(s, a)$  = expected utility starting in  $s$ , taking action  $a$ , and (thereafter) acting optimally

Bellman Equation:

$$Q^*(s, a) = \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma \max_{a'} Q^*(s', a'))$$

Q-Value Iteration:

$$Q_{k+1}^*(s, a) \leftarrow \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma \max_{a'} Q_k^*(s', a'))$$

**This is problematic when do not know the transitions**

# Tabular Q-learning

---

- Q-value iteration:  $Q_{k+1}(s, a) \leftarrow \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma \max_{a'} Q_k(s', a'))$
- Rewrite as expectation:  $Q_{k+1} \leftarrow \mathbb{E}_{s' \sim P(s'|s, a)} \left[ R(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$

# Tabular Q-learning

---

- Q-value iteration:  $Q_{k+1}(s, a) \leftarrow \sum_{s'} P(s'|s, a)(R(s, a, s') + \gamma \max_{a'} Q_k(s', a'))$
- Rewrite as expectation:  $Q_{k+1} \leftarrow \mathbb{E}_{s' \sim P(s'|s, a)} \left[ R(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$
- (Tabular) Q-Learning: replace expectation by samples
  - For an state-action pair (s,a), receive:  $s' \sim P(s'|s, a)$  **simulation and exploration**
  - Consider your old estimate:  $Q_k(s, a)$
  - Consider your new sample estimate:

$$\text{target}(s') = r(s, a, s') + \gamma \max_{a'} Q_k(s', a')$$

$$\text{error}(s') = \left( r(s, a, s') + \gamma \max_{a'} Q_k(s', a') - Q_k(s, a) \right)$$

## Tabular Q-learning

---

learning  
rate



$$\begin{aligned} Q_{k+1}(s, a) &= Q_k(s, a) + \alpha \text{ error}(s') \\ &= Q_k(s, a) + \alpha \left( r(s, a, s') + \gamma \max_{a'} Q_k(s', a') - Q_k(s, a) \right) \end{aligned}$$

**Key idea: implicitly estimate the transitions via simulation**

# Tabular Q-learning

---

## Bellman optimality

Algorithm:

Start with  $Q_0(s, a)$  for all  $s, a$ .

Get initial state  $s$

For  $k = 1, 2, \dots$  till convergence

    Sample action  $a$ , get next state  $s'$

    If  $s'$  is terminal: \_\_\_\_\_

        target =  $r(s, a, s')$

        Sample new initial state  $s'$

    else:

        target =  $r(s, a, s') + \gamma \max_{a'} Q_k(s', a')$

$Q_{k+1}(s, a) = Q_k(s, a) + \alpha \left( r(s, a, s') + \gamma \max_{a'} Q_k(s', a') - Q_k(s, a) \right)$

$s \leftarrow s'$

$$Q^*(s, a) = \mathbb{E}_{s'} \left[ r(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right]$$

# Tabular Q-learning

---

Algorithm:

Start with  $Q_0(s, a)$  for all  $s, a$ .

Get initial state  $s$

For  $k = 1, 2, \dots$  till convergence

Sample action  $a$ , get next state  $s'$

If  $s'$  is terminal:

$$\text{target} = r(s, a, s')$$

Sample new initial state  $s'$

else:

$$\text{target} = r(s, a, s') + \gamma \max_{a'} Q_k(s', a')$$

$$Q_{k+1}(s, a) = Q_k(s, a) + \alpha \left( r(s, a, s') + \gamma \max_{a'} Q_k(s', a') - Q_k(s, a) \right)$$

$$s \leftarrow s'$$

- Choose random actions?
- Choose action that maximizes  $Q_k(s, a)$  (i.e. greedily)?
- $\epsilon$ -Greedy: choose random action with prob.  $\epsilon$ , otherwise choose action greedily

## Exploration and Exploitation

---

Poor estimates of  $Q(s,a)$  at the start:

Bad initial estimates in the first few cases can drive policy into sub-optimal region, and never explore further.

$$\pi(s) = \begin{cases} \max_a \hat{Q}(s, a) & \text{with probability } 1 - \epsilon \\ \text{random action} & \text{otherwise} \end{cases}$$

Gradually decrease epsilon as policy is learned.



# Tabular Q-learning

---

Algorithm:

Start with  $Q_0(s, a)$  for all  $s, a$ .

Get initial state  $s$

For  $k = 1, 2, \dots$  till convergence

Sample action  $a$ , get next state  $s'$

If  $s'$  is terminal:

$$\text{target} = r(s, a, s')$$

Sample new initial state  $s'$

else:

$$\text{target} = r(s, a, s') + \gamma \max_{a'} Q_k(s', a')$$

$$Q_{k+1}(s, a) = Q_k(s, a) + \alpha \left( r(s, a, s') + \gamma \max_{a'} Q_k(s', a') - Q_k(s, a) \right)$$

$$s \leftarrow s'$$

Tabular: keep a  $|S| \times |A|$  table of  $Q(s, a)$

Still requires small and discrete state and action space

How can we generalize to unseen states?

- $\epsilon$ -Greedy: choose random action with prob.  $\epsilon$ , otherwise choose action greedily

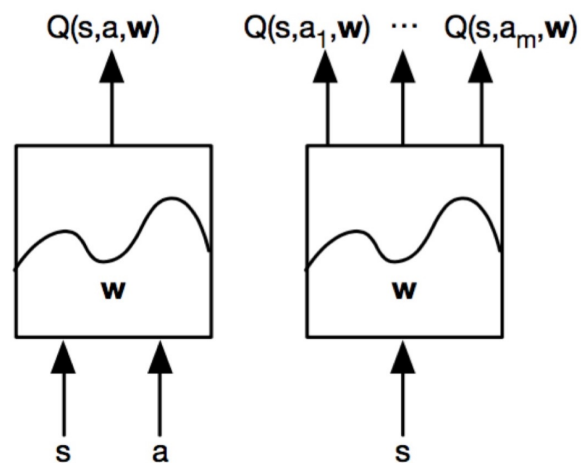
# Deep Q-learning

---

Q-learning with function approximation to **extract informative features** from **high-dimensional** input states.

Represent value function by Q network with weights  $\mathbf{w}$

$$Q(s, a, \mathbf{w}) \approx Q^*(s, a)$$



+ high-dimensional, continuous states  
+ generalization to new states

# Deep Q-learning

---

- 📖 Optimal Q-values should obey Bellman equation

$$Q^*(s, a) = \mathbb{E}_{s'} \left[ r + \gamma \max_{a'} Q(s', a')^* \mid s, a \right]$$

- 📖 Treat right-hand  $r + \gamma \max_{a'} Q(s', a', \mathbf{w})$  as as a target

- 📖 Minimize MSE loss by stochastic gradient descent

$$l = \left( r + \gamma \max_a Q(s', a', \mathbf{w}) - Q(s, a, \mathbf{w}) \right)^2$$

# Deep Q-learning Challenges

---

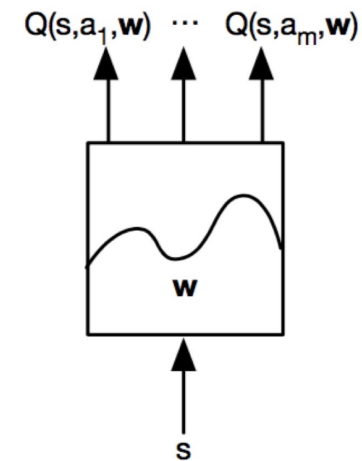
- Minimize MSE loss by stochastic gradient descent

$$l = \left( r + \gamma \max_a Q(s', a', \mathbf{w}) - Q(s, a, \mathbf{w}) \right)^2$$

- Converges to  $Q^*$  using table lookup representation

- But **diverges** using neural networks due to:

- Correlations between samples
- Non-stationary targets



## Deep Q-learning: Experience Replay

---

📁 To remove correlations, build data-set from agent's own experience

$s_1, a_1, r_2, s_2$
$s_2, a_2, r_3, s_3$
$s_3, a_3, r_4, s_4$
$\dots$
$s_t, a_t, r_{t+1}, s_{t+1}$

→  $s, a, r, s'$

exploration, epsilon greedy is important!

📁 Sample random mini-batch of transitions (s,a,r,s') from **D**

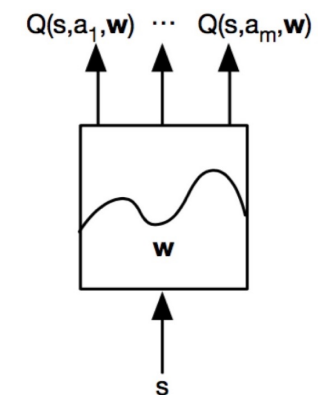
## Deep Q-learning: Fixed Q-targets

- Sample random mini-batch of transitions  $(s, a, r, s')$  from  $D$
- Compute Q-learning targets w.r.t. old fixed parameters  $\mathbf{w^-}$
- Optimize MSE between Q-network and Q-learning targets

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

- Use stochastic gradient descent
- Update  $\mathbf{w^-}$  with updated  $\mathbf{w}$  every  $\sim 1000$  iterations

$s_1, a_1, r_2, s_2$
$s_2, a_2, r_3, s_3$
$s_3, a_3, r_4, s_4$
...
$s_t, a_t, r_{t+1}, s_{t+1}$



## Policy Gradients

---

Formally, let's define a class of parameterized policies  $\Pi = \{\pi_\theta, \theta \in \mathbb{R}^m\}$

For each policy, define its value:

$$J(\theta) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t | \pi_\theta \right]$$

# Policy Gradients

---

Writing in terms of trajectories  $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$

Probability of a trajectory

$$\begin{aligned} p(\tau; \theta) &= \pi_{\theta}(a_0|s_0)p(s_1|s_0, a_0) \\ &\times \pi_{\theta}(a_1|s_1)p(s_2|s_1, a_1) \\ &\times \pi_{\theta}(a_2|s_2)p(s_3|s_2, a_2) \\ &\times \dots \\ &= \prod_{t \geq 0} p(s_{t+1}|s_t, a_t)\pi_{\theta}(a_t|s_t) \end{aligned}$$

Reward of a trajectory

$$r(\tau) = \sum_{t \geq 0} \gamma^t r_t$$

$$J(\theta) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t | \pi_{\theta} \right] = \mathbb{E}_{\tau \sim p(\tau; \theta)} [r(\tau)]$$



# Policy Gradients

---

Formally, let's define a class of parameterized policies  $\Pi = \{\pi_\theta, \theta \in \mathbb{R}^m\}$

For each policy, define its value:

$$J(\theta) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t | \pi_\theta \right] = \mathbb{E}_{\tau \sim p(\tau; \theta)} [r(\tau)]$$

We want to find the optimal policy  $\theta^* = \arg \max_{\theta} J(\theta)$

How can we do this?

Gradient ascent on policy parameters

## REINFORCE Algorithm

---

Expected reward:  $J(\theta) = \mathbb{E}_{\tau \sim p(\tau; \theta)} [r(\tau)]$

$$= \int_{\tau} r(\tau) p(\tau; \theta) d\tau$$

# REINFORCE Algorithm

---

Expected reward:  $J(\theta) = \mathbb{E}_{\tau \sim p(\tau; \theta)} [r(\tau)]$

$$= \int_{\tau} r(\tau) p(\tau; \theta) d\tau$$

$$p(\tau; \theta) = \prod_{t \geq 0} p(s_{t+1} | s_t, a_t) \pi_{\theta}(a_t | s_t)$$

Now let's differentiate this:  $\nabla_{\theta} J(\theta) = \int_{\tau} r(\tau) \nabla_{\theta} p(\tau; \theta) d\tau$

**Intractable**

# REINFORCE Algorithm

---

Expected reward:  $J(\theta) = \mathbb{E}_{\tau \sim p(\tau; \theta)} [r(\tau)]$

$$= \int_{\tau} r(\tau) p(\tau; \theta) d\tau$$

$$p(\tau; \theta) = \prod_{t \geq 0} p(s_{t+1} | s_t, a_t) \pi_{\theta}(a_t | s_t)$$

Now let's differentiate this:  $\nabla_{\theta} J(\theta) = \int_{\tau} r(\tau) \nabla_{\theta} p(\tau; \theta) d\tau$  **Intractable**

However, we can use a nice trick:  $\nabla_{\theta} p(\tau; \theta) = p(\tau; \theta) \frac{\nabla_{\theta} p(\tau; \theta)}{p(\tau; \theta)} = p(\tau; \theta) \nabla_{\theta} \log p(\tau; \theta)$

# REINFORCE Algorithm

---

Expected reward:  $J(\theta) = \mathbb{E}_{\tau \sim p(\tau; \theta)} [r(\tau)]$

$$= \int_{\tau} r(\tau) p(\tau; \theta) d\tau$$

$$p(\tau; \theta) = \prod_{t \geq 0} p(s_{t+1} | s_t, a_t) \pi_{\theta}(a_t | s_t)$$

Now let's differentiate this:  $\nabla_{\theta} J(\theta) = \int_{\tau} r(\tau) \nabla_{\theta} p(\tau; \theta) d\tau$  **Intractable**

However, we can use a nice trick:  $\nabla_{\theta} p(\tau; \theta) = p(\tau; \theta) \frac{\nabla_{\theta} p(\tau; \theta)}{p(\tau; \theta)} = p(\tau; \theta) \nabla_{\theta} \log p(\tau; \theta)$   
If we inject this back:

$$\begin{aligned} \nabla_{\theta} J(\theta) &= \int_{\tau} (r(\tau) \nabla_{\theta} \log p(\tau; \theta)) p(\tau; \theta) d\tau \\ &= \mathbb{E}_{\tau \sim p(\tau; \theta)} [r(\tau) \nabla_{\theta} \log p(\tau; \theta)] \end{aligned}$$

# REINFORCE Algorithm

---

Can we compute these without knowing the transition probabilities?

We have:

$$p(\tau; \theta) = \prod_{t \geq 0} p(s_{t+1} | s_t, a_t) \pi_{\theta}(a_t | s_t)$$

---

# REINFORCE Algorithm

---

Can we compute these without knowing the transition probabilities?

We have: 
$$p(\tau; \theta) = \prod_{t \geq 0} p(s_{t+1} | s_t, a_t) \pi_{\theta}(a_t | s_t)$$

Thus: 
$$\log p(\tau; \theta) = \sum_{t \geq 0} (\log p(s_{t+1} | s_t, a_t) + \log \pi_{\theta}(a_t | s_t))$$

# REINFORCE Algorithm

---

Can we compute these without knowing the transition probabilities?

We have: 
$$p(\tau; \theta) = \prod_{t \geq 0} p(s_{t+1} | s_t, a_t) \pi_{\theta}(a_t | s_t)$$

Thus: 
$$\log p(\tau; \theta) = \sum_{t \geq 0} (\log p(s_{t+1} | s_t, a_t) + \log \pi_{\theta}(a_t | s_t))$$

And when differentiating: 
$$\nabla_{\theta} \log p(\tau; \theta) = \sum_{t \geq 0} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

Doesn't depend on  
transition probabilities



# REINFORCE Algorithm

---

Can we compute these without knowing the transition probabilities?

We have: 
$$p(\tau; \theta) = \prod_{t \geq 0} p(s_{t+1} | s_t, a_t) \pi_{\theta}(a_t | s_t)$$

Thus: 
$$\log p(\tau; \theta) = \sum_{t \geq 0} (\log p(s_{t+1} | s_t, a_t) + \log \pi_{\theta}(a_t | s_t))$$

And when differentiating: 
$$\nabla_{\theta} \log p(\tau; \theta) = \sum_{t \geq 0} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$
 Doesn't depend on transition probabilities

Therefore when sampling a trajectory, we can estimate gradients:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p(\tau; \theta)} [r(\tau) \nabla_{\theta} \log p(\tau; \theta)] \approx \sum_{t \geq 0} r(\tau) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

# Policy Gradients

Gradient estimator:

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

Interpretation:

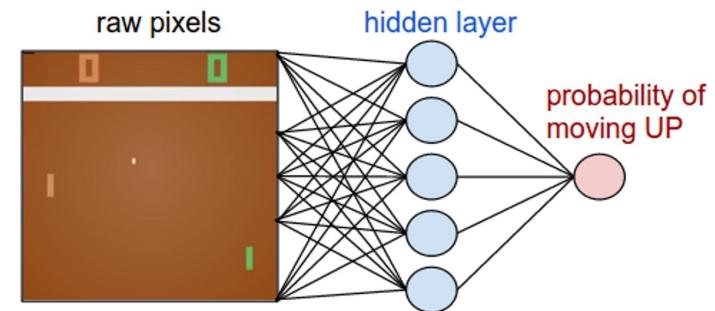
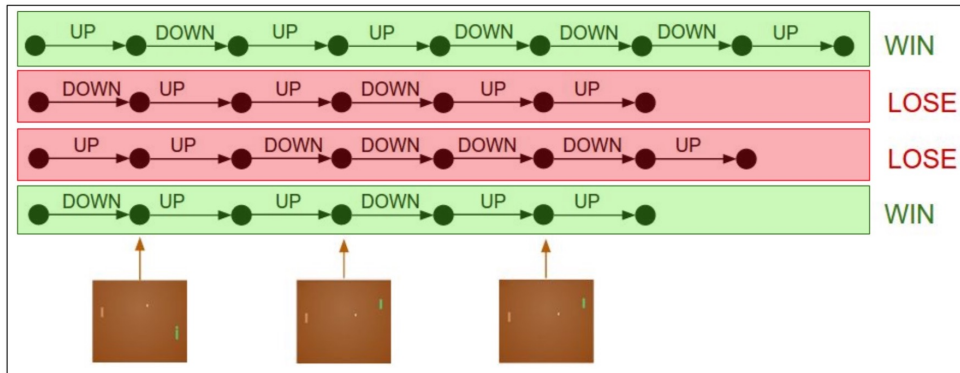
- If  $r(\text{trajectory})$  is high, push up the probabilities of the actions seen
- If  $r(\text{trajectory})$  is low, push down the probabilities of the actions seen

Pretend every action we took here was the correct label.

maximize:  $\log p(y_i | x_i)$

Pretend every action we took here was the wrong label.

maximize:  $(-1) * \log p(y_i | x_i)$



$$\sum_i A_i * \log p(y_i | x_i)$$

# Policy Gradients

Gradient estimator:

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

Interpretation:

- If **r(trajectory)** is high, push up the probabilities of the actions seen
- If **r(trajectory)** is low, push down the probabilities of the actions seen

## REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic)

Input: a differentiable policy parameterization  $\pi(a|s, \theta), \forall a \in \mathcal{A}, s \in \mathcal{S}, \theta \in \mathbb{R}^n$

Initialize policy weights  $\theta$

Repeat forever:

Generate an episode  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$  following  $\pi(\cdot | \cdot, \theta)$

For each step of the episode  $t = 0, \dots, T-1$ :

$G_t \leftarrow$  return from step  $t$

$\theta \leftarrow \theta + \alpha \gamma^t G_t \nabla_{\theta} \log \pi(A_t | S_t, \theta)$

**epsilon greedy**

# Policy Gradients

---

Gradient estimator:

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

Interpretation:

- If **r(trajectory)** is high, push up the probabilities of the actions seen
- If **r(trajectory)** is low, push down the probabilities of the actions seen

Might seem simplistic to say that if a trajectory is good then all its actions were good. **But in expectation, it averages out!**

However, this also suffers from high variance because credit assignment is really hard - can we help this estimator?

## Variance Reduction with a Baseline

---

**Problem:** The raw reward of a trajectory isn't necessarily meaningful. E.g. if all rewards are positive, you keep pushing up probabilities of all actions.

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

**Idea:** Introduce a baseline function dependent on the state, which gives us an estimator:

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

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

Recall: **Q** and **V** - action and state value functions!

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

Using this, we get the estimator:

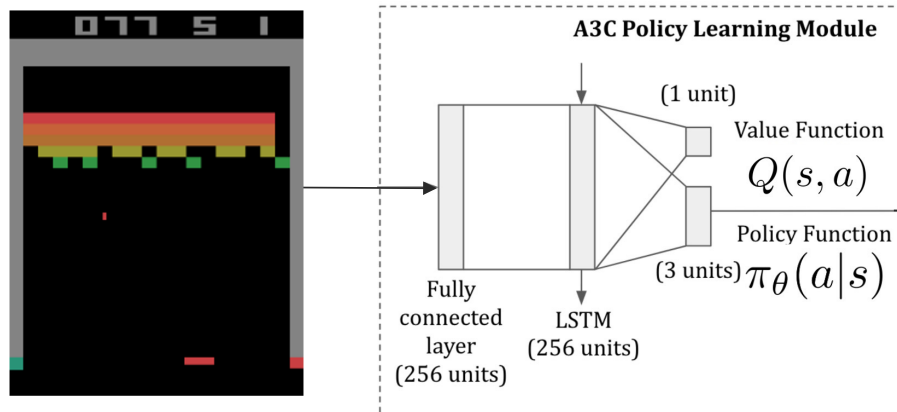
$$\nabla_{\theta} J(\theta) \approx \sum_{t \geq 0} (Q^{\pi_{\theta}}(s_t, a_t) - V^{\pi_{\theta}}(s_t)) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

# Actor-Critic Methods

**Problem:** we don't know Q and V - can we learn them?

**Yes,** using Q-learning! We can combine Policy Gradients and Q-learning by training both an **actor** (the policy) and a **critic** (the Q function)

Exploration + experience replay  
Decorrelate samples  
Fixed targets



**Critic: evaluates how good the action is**

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

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

**Actor: decides what actions to take**

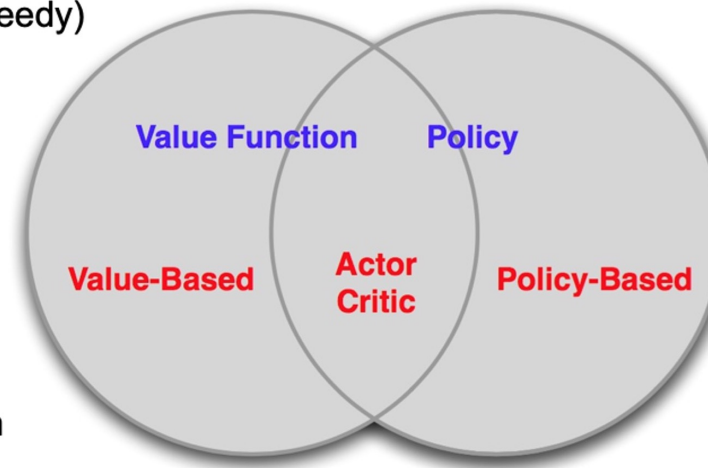
$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} (Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)) \nabla_\theta \log \pi_\theta(a_t | s_t)$$

Variance reduction with a baseline

# Summary: RL Methods

---

- **Value Based**
    - Learned Value Function
    - Implicit policy (e.g.  $\epsilon$ -greedy)
  - **Policy Based**
    - No Value Function
    - Learned Policy
  - **Actor-Critic**
    - Learned Value Function
    - Learned Policy
- Value iteration**  
**Policy iteration**  
**(Deep) Q-learning**
- Policy gradients**
- Actor (policy)**  
**Critic (Q-values)**





---

# Reinforcement Learning in Large Language Models

## Actor-Critic, TRPO, PPO, and GRPO

# Introduction

---

- DeepSeek models leverage advanced RL algorithms to improve reasoning ability
- Key algorithms:
  - Actor-Critic
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO)
  - Group Relative Policy Optimization (GRPO) - DeepSeek's innovation
- Applications in:
  - DeepSeekMath (mathematical reasoning)
  - DeepSeek-R1 (general reasoning capabilities)

# Scaling Laws in AI

- **Scaling Law:** Model performance improves predictably with:

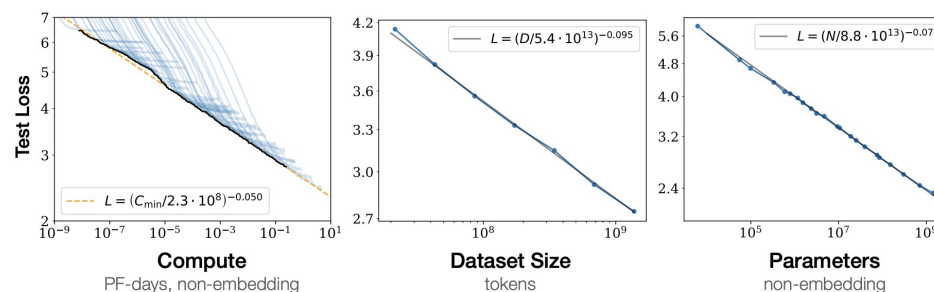
- More parameters ( $N$ )
- More training data ( $D$ )
- More compute ( $C$ )

- Key equation (Kaplan et al. 2020, OpenAI):

$$L(N, D) = \left( \frac{N_c}{N} \right)^{\alpha_N} + \left( \frac{D_c}{D} \right)^{\alpha_D} \quad (1)$$

- Observed across:

- Language models
- Vision models
- Multimodal systems

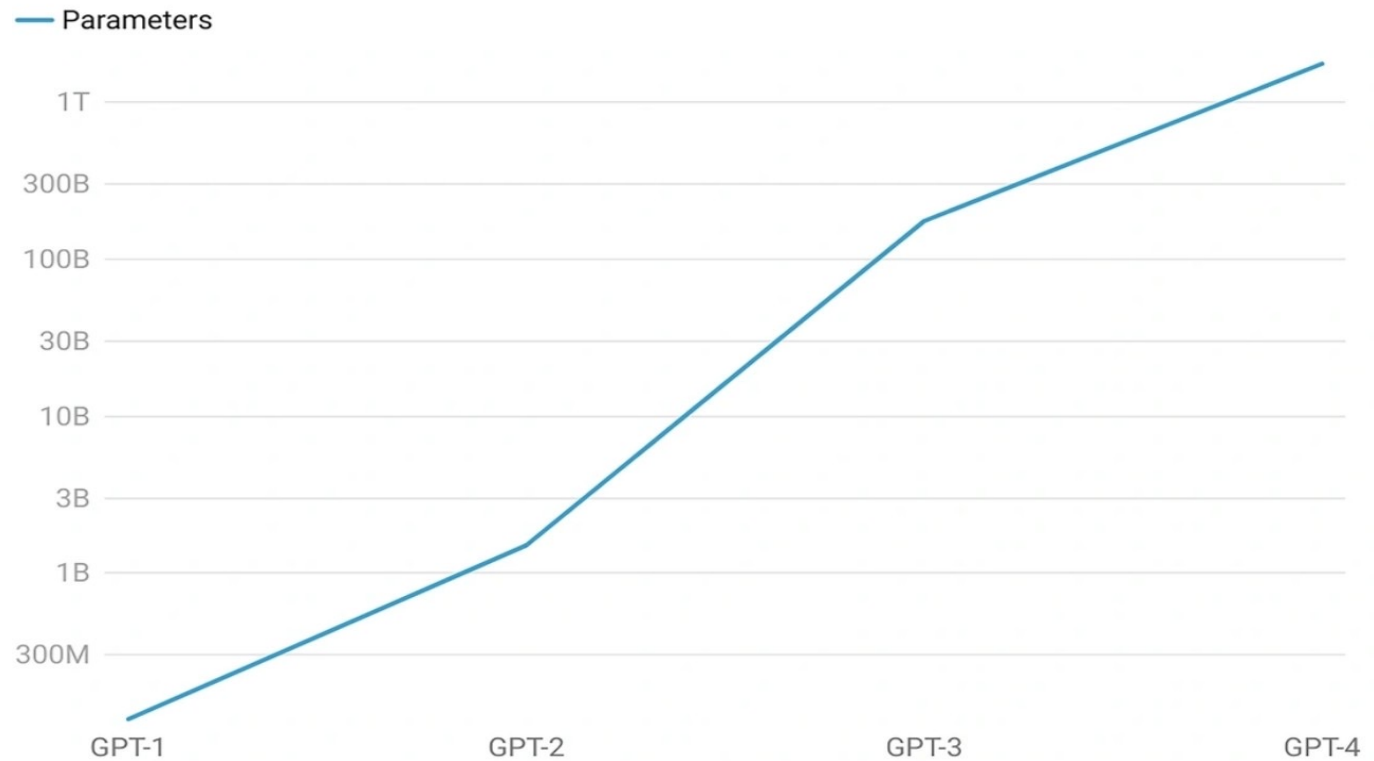


# The Scaling Timeline

Figure: Scaling law curve

## ChatGPT Parameters

The number of parameters in successive models of ChatGPT has increased massively



# Test-Time Compute

---

## Definition

The computational resources allocated **during inference** to solve a task

Reason: Further scaling in the training phase becomes difficult due to the scarcity of data and computational resources.

### System 1:

- GPT-4, Deepseek-V3
- Fast, intuitive
- Limited reasoning steps

### System 2:

- GPT-o1, Deepseek-R1
- Slow, deep thinking
- Multi-step reasoning

## Key Insight

Performance can scale with **inference compute** independently of model size

Reference: Test-Time Compute: from System-1 Thinking to System-2 Thinking, Yixin Ji, Juntao Li, Hai Ye, Kaixin Wu, Kai Yao, Jia Xu, Linjian Mo, Min Zhang, arXiv Mar, 2025

# The Performance of Deepseek R1

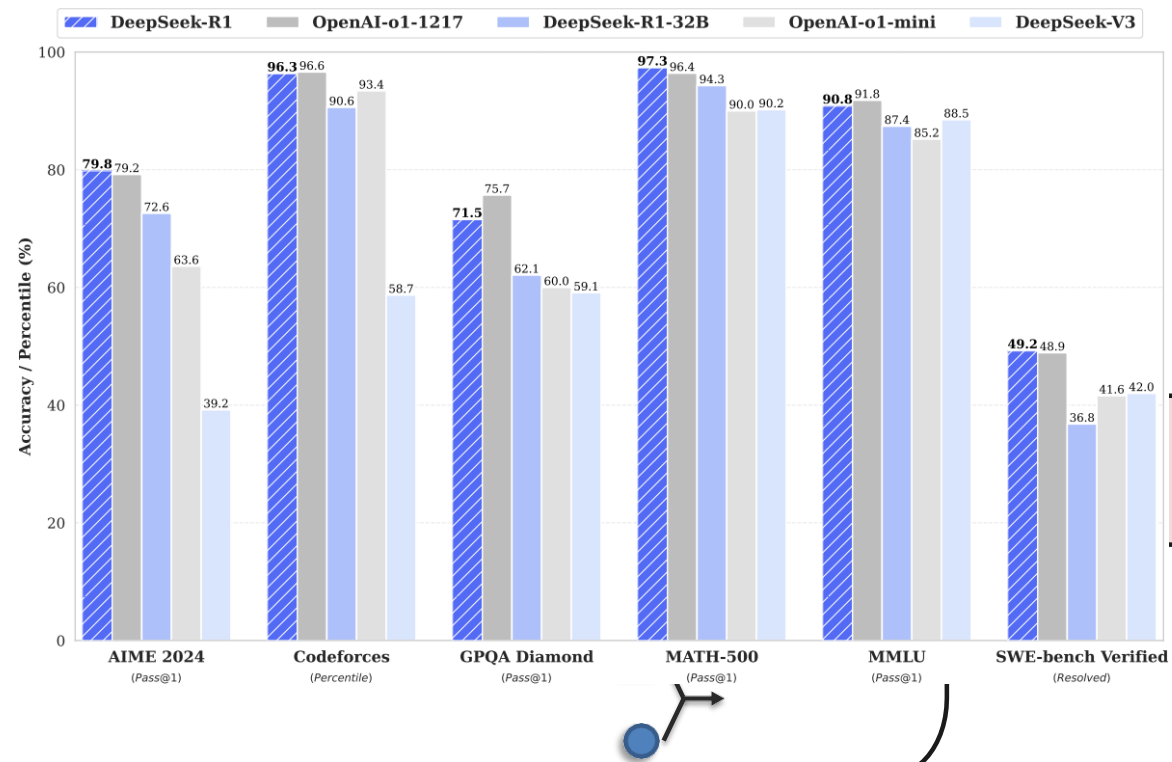
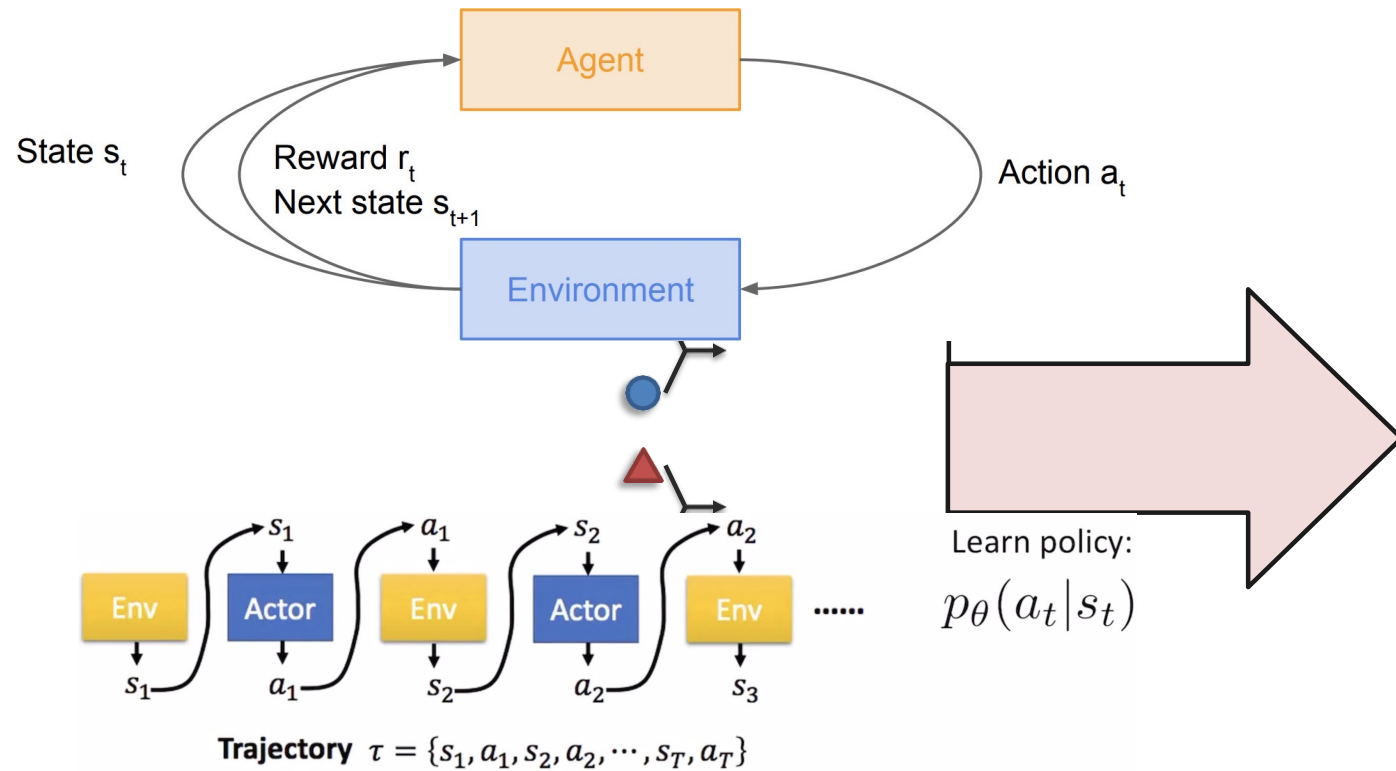


Figure: Benchmark performance of DeepSeek-R1.

# Reinforcement Learning

---



## Notations

---

- $s$ : the state that inputs to the agent (In the LLM context, the state is the input prompt + already generated tokens.)
- $a$ : the action that the agent outputs. (In the LLM context, the action is the next token to be predicted.)
- $\pi_{\theta}(s, a)$ : a policy function.  $\theta$  is the parameters of the agent (usually a neural network in the policy-based RL). In the LLM context, the  $\theta$  is the parameters of an LLM.  $\pi_{\theta}(s, a)$  is the LLM's output probability distribution of the next token ( $a$ ) with input tokens ( $s$ ).
- $R$  or  $r$ : the reward under state  $s$  and action  $a$ . In the context of LLM, the reward could be given by human feedback or another LLM.
- $\gamma$  is a hyperparameter between  $[0, 1]$  that allows the model to focus on the reward of the current step



## “Expected Returns” Objective

Now we formalize the “Expected Returns” objective  $J(\theta)$

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1} \right]$$

Value Function  $V^\pi(s)$  and Action Value function  $Q^\pi(s, a)$  defined as:

$$V^\pi(s) = \mathbb{E}_{k=0}^{\infty} \left[ \gamma^k \cdot R_{k+1} \middle| S_t = s \right] \quad \text{for all } t = 0, 1, 2, \dots$$

$$Q^\pi(s, a) = \mathbb{E}_{k=0}^{\infty} \left[ \gamma^k \cdot R_{k+1} \middle| S_t = s, A_t = a \right] \quad \text{for all } t = 0, 1, 2, \dots$$

Advantage Function  $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$

$Q$  describes the expected return under the  $\pi_\theta$ ,  $s$  and  $a$ ,  $V$  describes the expected return under  $\pi_\theta$  and  $s$ , and  $J$  describes the expected return under  $\pi_\theta$

## Recap: Policy Gradient

### Key Idea

Directly optimize the policy  $\pi_\theta(a|s)$  using gradient ascent:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a)]$$

- Objective: Maximize expected return  $J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_t \gamma^t r_t]$
- Uses Monte-Carlo estimation of returns
- $Q(s, a)$  decide which action to be reinforced,  $\nabla_\theta \log \pi_\theta(a|s)$  decide how to reinforce this action (by updating policy network parameters)

### REINFORCE Algorithm

- 1 Sample trajectory  $\tau = (s_0, a_0, r_0, \dots, s_T)$
- 2 Compute returns  $G_t = \sum_{k=t}^T \gamma^{k-t} r_k$
- 3 Update policy:  $\theta \leftarrow \theta + \alpha \sum_t G_t \nabla_\theta \log \pi_\theta(a_t|s_t)$

## Recap: Policy Gradient

---

### REINFORCE Algorithm:

- Sample trajectory  $\tau^i$  from  $\pi_\theta(a_t|s_t)$

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^T \gamma^t R_{t+1} \nabla \log \pi_\theta(a_t|s_t) \right]$$

$$= \sum_{t=1}^T \nabla \log \pi_\theta(a_t|s_t) \sum_{k=t}^T \gamma^{k-t} R_{k+1}$$

- Issue: We need to sample whole trajectory to get this term (Monte Carlo)  
Make policy gradient learn slowly.

## Recap: Actor-Critic Framework

- Two components:
  - Actor**: Policy network  $\pi_{\theta}(a|s)$ , also called actor model, decide which action to take for next step.
  - Critic**: Value network  $V_{\phi}(s)$ , measure how good the present state is.
- Advantage function:  $A(s_t, a_t) = Q(s_t, a_t) - V(s_t) = r_t + V(s_{t+1}) - V(s_t)$
- Policy gradient update:  
$$\nabla_{\theta} J(\theta) = E[\nabla_{\theta} \log \pi_{\theta}(a|s) A(s, a)]$$
- $A(s,a)$  reflects how good the action we've taken compared to other candidates.
- Using  $A(s,a)$  instead of  $Q(s,a)$  can make the training more stable.

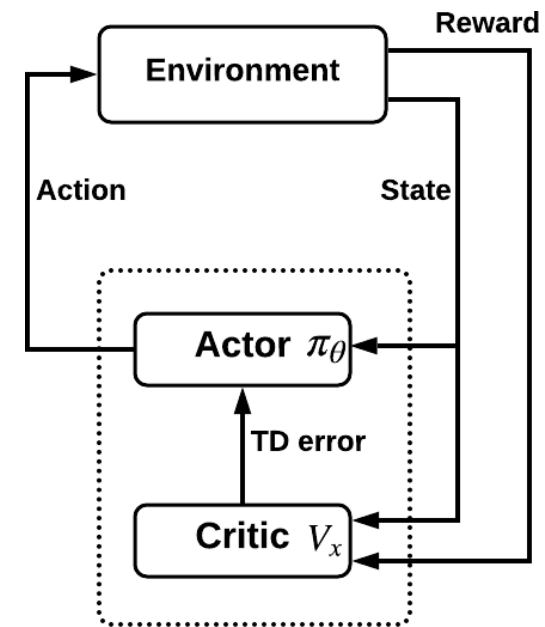
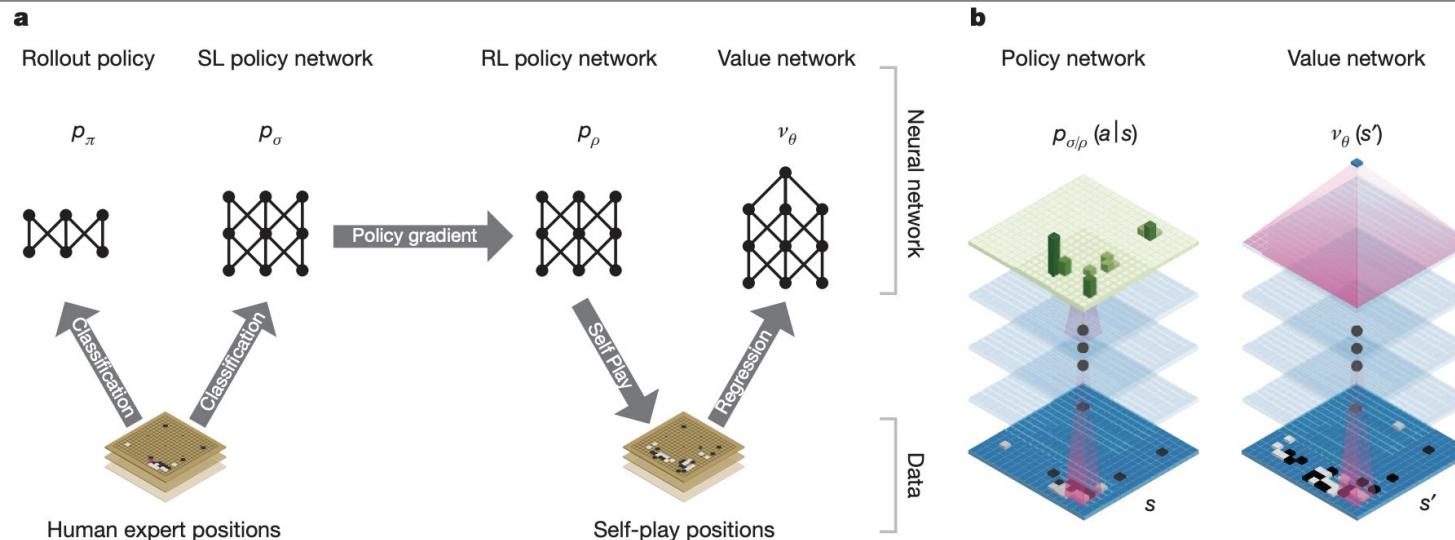


Figure: Actor-Critic architecture

# The Actor-Critic Networks in AlphaGo



**Figure 1 | Neural network training pipeline and architecture.** **a**, A fast rollout policy  $p_\pi$  and supervised learning (SL) policy network  $p_\sigma$  are trained to predict human expert moves in a data set of positions. A reinforcement learning (RL) policy network  $p_\rho$  is initialized to the SL policy network, and is then improved by policy gradient learning to maximize the outcome (that is, winning more games) against previous versions of the policy network. A new data set is generated by playing games of self-play with the RL policy network. Finally, a value network  $v_\theta$  is trained by regression to predict the expected outcome (that is, whether

the current player wins) in positions from the self-play data set.

**b**, Schematic representation of the neural network architecture used in AlphaGo. The policy network takes a representation of the board position  $s$  as its input, passes it through many convolutional layers with parameters  $\sigma$  (SL policy network) or  $\rho$  (RL policy network), and outputs a probability distribution  $p_\sigma(a|s)$  or  $p_\rho(a|s)$  over legal moves  $a$ , represented by a probability map over the board. The value network similarly uses many convolutional layers with parameters  $\theta$ , but outputs a scalar value  $v_\theta(s')$  that predicts the expected outcome in position  $s'$ .

Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489 (2016).

# Trust Region Policy Optimization (Schulman, 2015)

---

The problem of policy gradient:

- Large policy change destroys training
- Improper learning rate causes vanishing or exploding gradient
- Poor sample efficiency. PG needs over 10 million or more training time steps for toy experiments.



Line search  
(like gradient ascent)



Trust region

## TRPO

---

Key Equations Let  $\pi_\theta$  denote a policy with parameters  $\theta$ . The theoretical TRPO update is:

$$\theta_{k+1} = \arg \max_{\theta} \mathcal{L}(\theta_k, \theta)$$

$$\text{s.t. } \bar{D}_{KL}(\theta || \theta_k) \leq \delta$$

where  $\mathcal{L}(\theta_k, \theta)$  is the surrogate advantage, a measure of how policy  $\pi_\theta$  performs relative to the old policy  $\pi_{\theta_k}$  using data from the old policy:

$$\mathcal{L}(\theta_k, \theta) = \underbrace{\mathbb{E}_{s, a \sim \pi_{\theta_k}} \left[ \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a) \right]}_{\text{Surrogate Advantage}}$$

and  $\bar{D}_{KL}(\theta || \theta_k)$  is an average KL-divergence between policies across states visited by the old policy:

$$\bar{D}_{KL}(\theta || \theta_k) = \mathbb{E}_{s \sim \pi_{\theta_k}} [D_{KL}(\pi_\theta(\cdot|s) || \pi_{\theta_k}(\cdot|s))]$$

## TRPO and PPO

---

### Trust Region Policy Optimization (TRPO), ICML2015

Constrained optimization to ensure stable updates:  $\max_{\theta} \mathbb{E}[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)} A(s, a)]$

- Subject to KL-divergence constraint:  $\mathbb{E}[KL(\pi_{\theta_{old}} || \pi_{\theta})] \leq \delta$

### Proximal Policy Optimization (PPO) (OpenAI, 2017)

Simplified version with clipped objective:

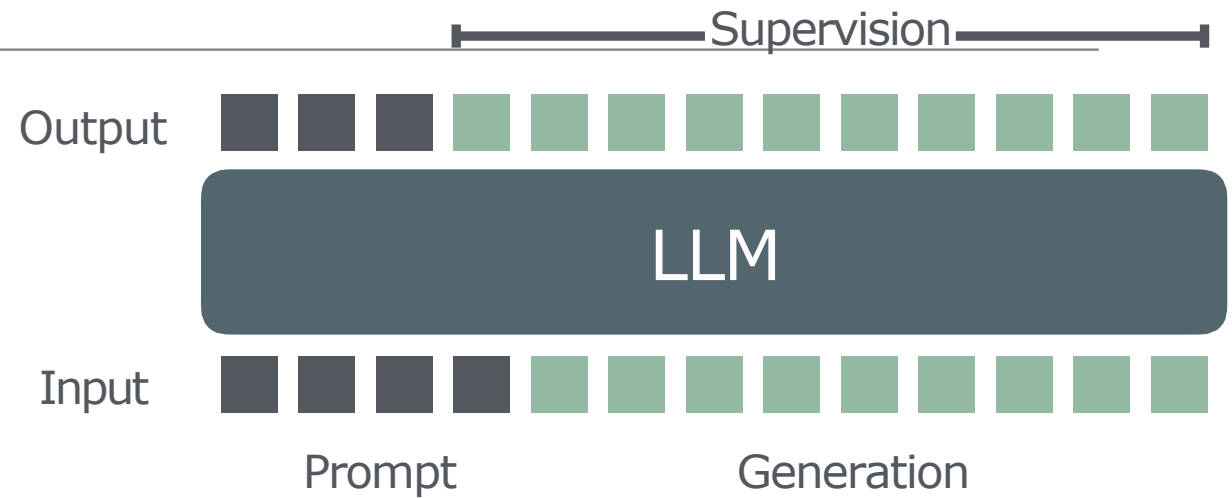
$$L^{CLIP}(\theta) = \mathbb{E}[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t)]$$

- Where  $r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$
- Using a simple clip function to take the place of KL divergence
- Used in ChatGPT's RLHF Algorithm

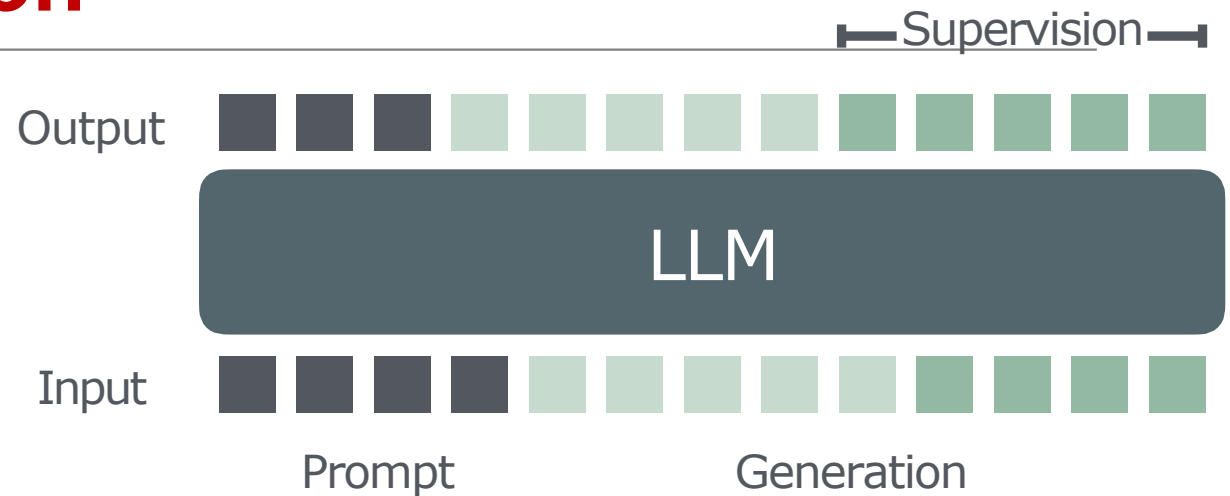


## Teacher forcing

- Simple supervised learning
  - $\text{Input} = \text{Prompt} + \text{Target}[0:-1]$
  - $\text{Loss}(\text{output}, \text{Target}[1:])$



# Outcome supervision



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

# Outcome supervision

## Reinforcement Learning

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

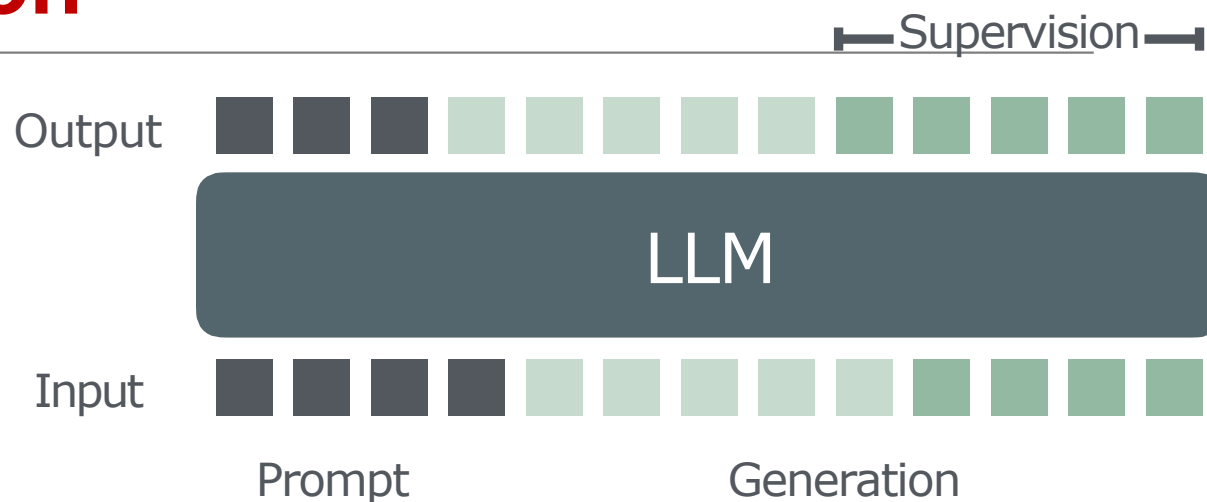
$$p_{\theta}(\mathbf{x} | \mathbf{c}) = \prod_{t=1}^N p_{\theta}(x_{t+1} | \mathbf{c}, x_1 \dots x_t)$$

- Sampling / Generation

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

- MDP

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



# Interactive Digital Agents

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



# Application of PPO in LLM

Nisan Stiennon, Learning to summarize from human feedback, NIPS 2020, OpenAI.

## 1 Collect human feedback

A Reddit post is sampled from the Reddit TL;DR dataset.



Various policies are used to sample a set of summaries.



Two summaries are selected for evaluation.



A human judges which is a better summary of the post.



"j is better than k"

## 2 Train reward model

One post with two summaries judged by a human are fed to the reward model.



The reward model calculates a reward  $r$  for each summary.



$r_j$

$r_k$

The loss is calculated based on the rewards and human label, and is used to update the reward model.

$$\text{loss} = \log(\sigma(r_j - r_k))$$

"j is better than k"

## 3 Train policy with PPO

A new post is sampled from the dataset.



The policy  $\pi$  generates a summary for the post.



The reward model calculates a reward for the summary.



The reward is used to update the policy via PPO.

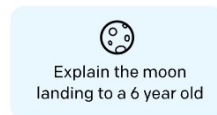
$r$

# Application of PPO in LLM InstructGPT OpenAI2022

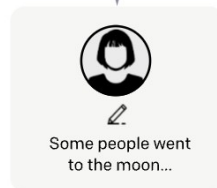
## Step 1

**Collect demonstration data, and train a supervised policy.**

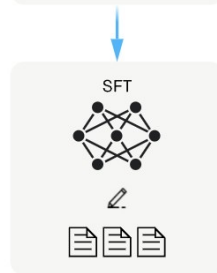
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



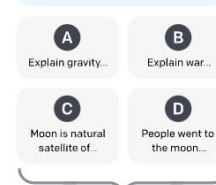
This data is used to fine-tune GPT-3 with supervised learning.



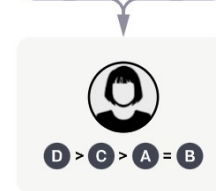
## Step 2

**Collect comparison data, and train a reward model.**

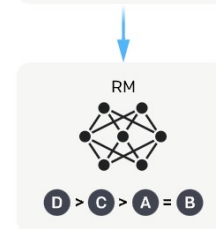
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



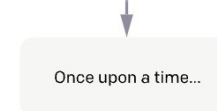
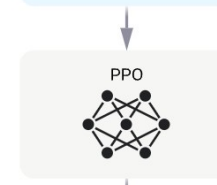
## Step 3

**Optimize a policy against the reward model using reinforcement learning.**

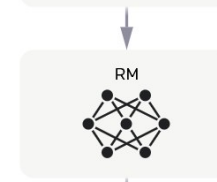
A new prompt is sampled from the dataset.



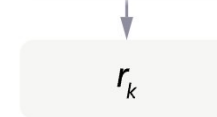
The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



## Reinforcement Learning in GPT-o1

Scaling of Search and Learning: A Roadmap to Reproduce o1 from Reinforcement Learning Perspective, Zhiyuan Zeng, et al, 2024

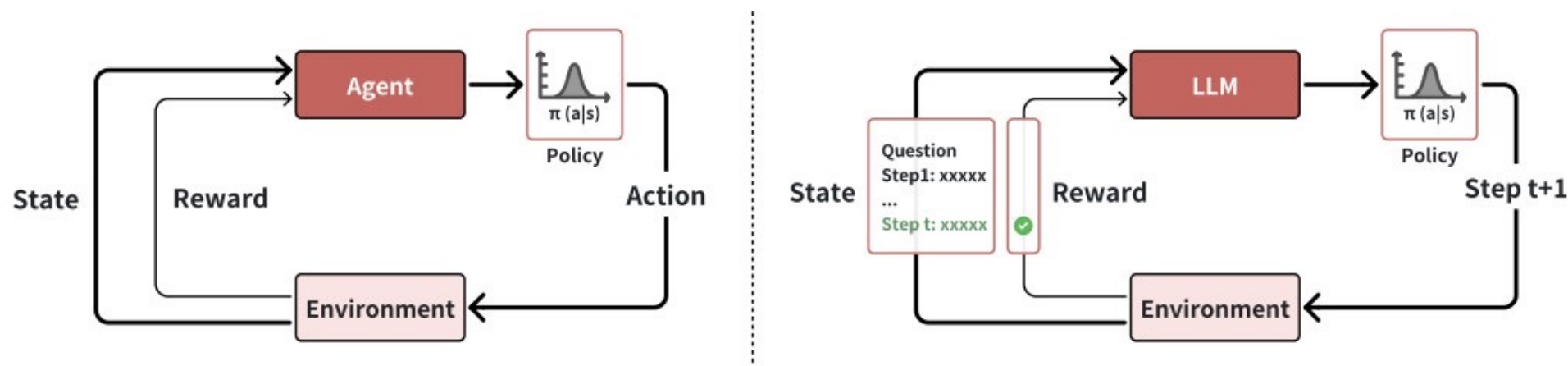


Figure 3: The visualization of the interaction between agent and environment in reinforcement learning for LLMs. Left: traditional reinforcement learning. Right: reinforcement learning for LLMs. The figure only visualizes the step-level action for simplicity. In fact, the action of LLM can be either token-, step-, or solution-level.

## Group Relative Policy Optimization (GRPO)

---

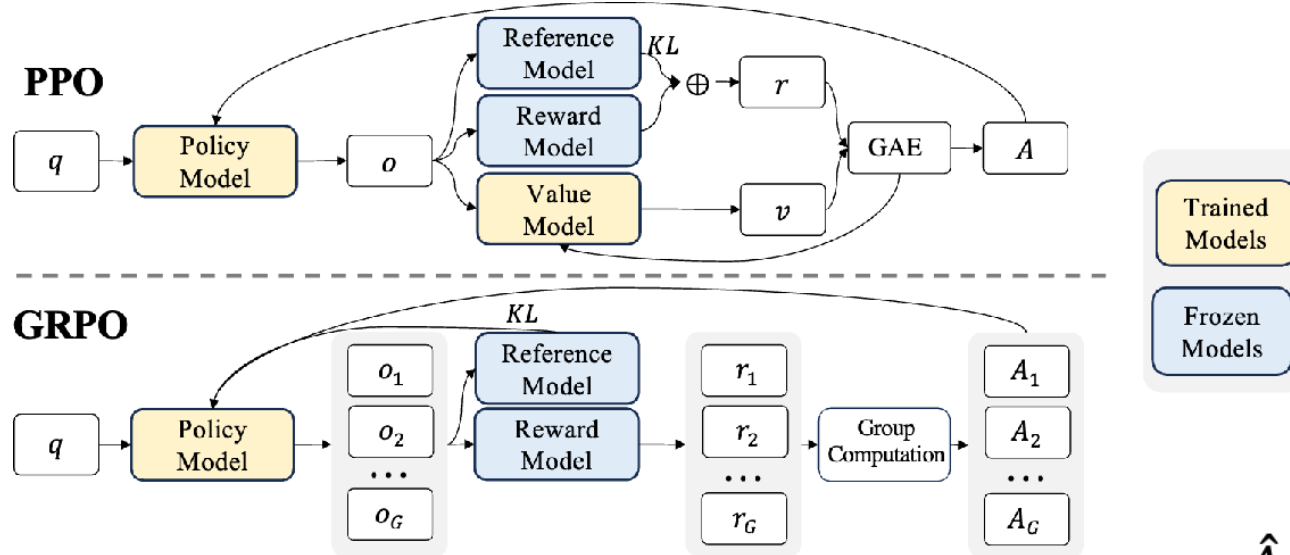
- Proposed in DeepSeekMath (arkiv 2402.03300v3)
- Key improvements over PPO:
  - Eliminates critic network - uses group statistics as baseline
  - More memory efficient (no separate value network)
  - Better for mathematical reasoning tasks
- Objective function:
$$J_{GRPO}(\theta) = \mathbb{E}\left[\frac{1}{G} \sum_{i=1}^G \min\left(\frac{\pi_{\theta}(o_i|q)}{\pi_{\theta_{old}}(o_i|q)} \hat{A}_i, \text{clip}\left(\frac{\pi_{\theta}(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1 - \epsilon, 1 + \epsilon\right) \hat{A}_i\right) - \beta D_{KL}(\pi_{\theta} || \pi_{ref})\right]$$
- $\hat{A}_i$  computed from group rewards

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_1, r_2, \dots, r_G\})}{\text{std}(\{r_1, r_2, \dots, r_G\})}.$$



# GRPO

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



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

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

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

# PPO vs GRPO

---

Feature	OpenAI o1 (PPO)	DeepSeek R1 (GRPO)
Models Trained	2 (policy + critic)	1 (policy only)
Training Method	Compares responses one by one	Ranks multiple responses at once
Computational Cost	High (training two models)	Low (training only one model)
Training Speed	Slower	Faster
Self-Verification	Weak	Strong (better ranking method)



## GRPO in DeepSeekMath

- Applied to 7B parameter model
- Training details:
  - 144K math questions (GSM8K & MATH)
  - 64 samples per question
  - Batch size 1024
  - KL coefficient 0.04
- Results:
  - GSM8K: 82.9%  $\rightarrow$  88.2%
  - MATH: 46.8%  $\rightarrow$  51.7%
  - Out-of-domain improvements too

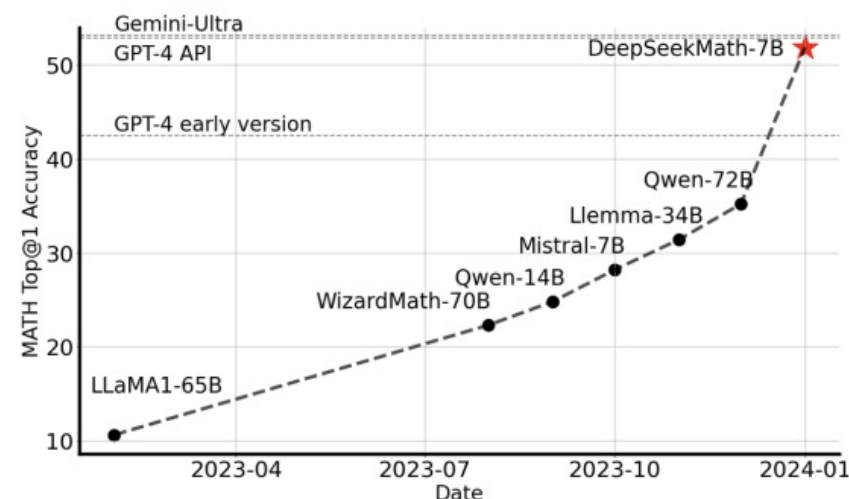
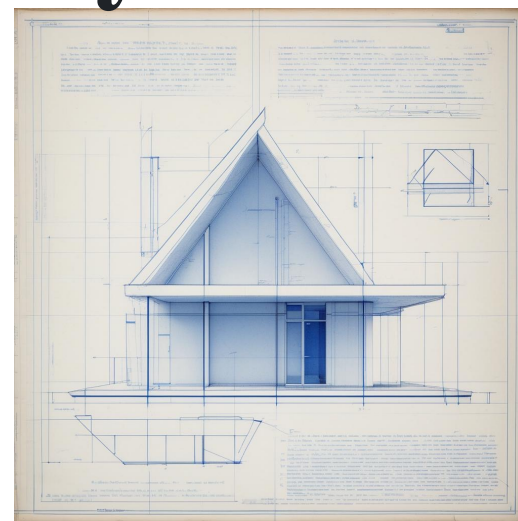


Figure: Performance improvement with GRPO on the MATH dataset

# RL-based Reasoning MLLM:

What has the community done?

What could the community do next?



*"The senses are the organs by which man perceives the world, and the soul acts through them as through tools."*

# **What has the community done?**





# First Success: Multimodality

---



**Vision (perception) :**

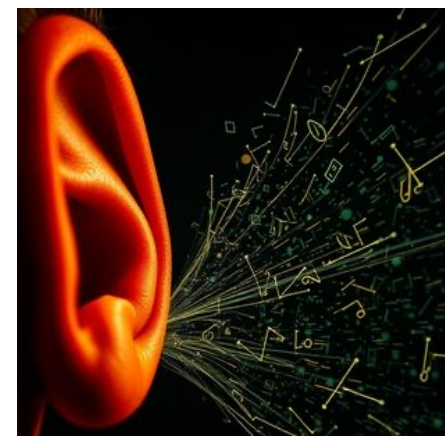
**Image**

<b>R1-V</b>	<b>OThink-MR1</b>
<b>VLM-R1</b>	<b>Think or Not Think</b>
<b>R1-Vision</b>	<b>OpenVLThinker</b>
<b>MMR1</b>	<b>Reason-RFT</b>
<b>Visual-RFT</b>	<b>Q-Insight</b>
<b>MM-Eureka</b>	<b>R1-Zero-VSI</b>
<b>Seg-Zero</b>	<b>Ocean-R1</b>
<b>Vision-R1</b>	.....
<b>VisualThinker-R1-Zero</b>	
<b>R1-VL</b>	



**Vision (Temporal) : Video**

**Temporal-R1**  
**SEED-Bench-R1**  
**Video-R1**  
**TimeZero**  
**Open R1 Video**  
**Open-LLaVA-Video-R1**  
.....

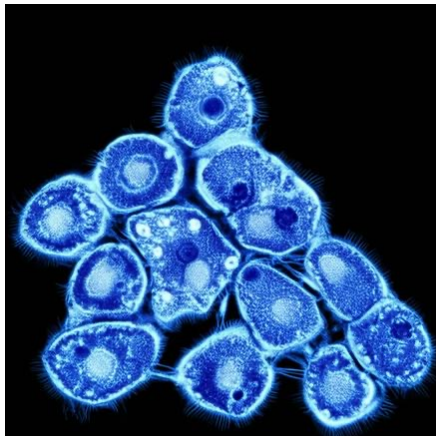


**Audio**

**Audio-Reasoner**  
**R1-AQA**  
.....

# First Success: Multimodality

---



**Medical Vision**

**MedVLM-R1**

**Med-R1**

.....



**Omni**

**R1-Omni**

.....



**Graphical User Interface**

**UI-R1**

.....



**Metaverse**

**MetaSpatial**

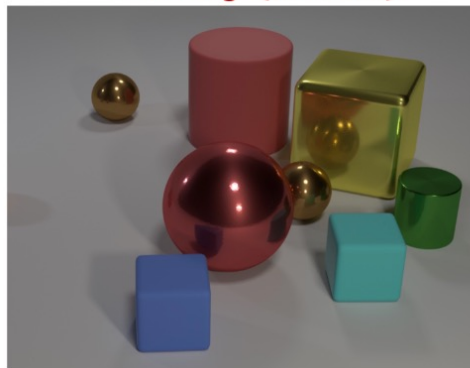
.....



# Second Success: Diverse Task——Take Vision as an Example

## R1-V

Training (CLEVR-A)

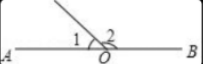


OOD Testing (Super-CLEVR)



#1

image



problem

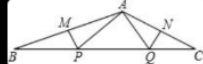
In the given diagram, if angle 1 has a measure of 35.0 degrees, what is the measure of angle 2?

solution

<answer> 145° </answer>

#5

image



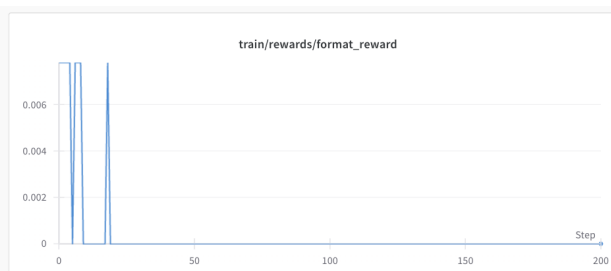
problem

In the given diagram, if angle B measures 20° and angle C measures 30°, and MP and NQ bisect AB and AC perpendicularly, what is the measur...

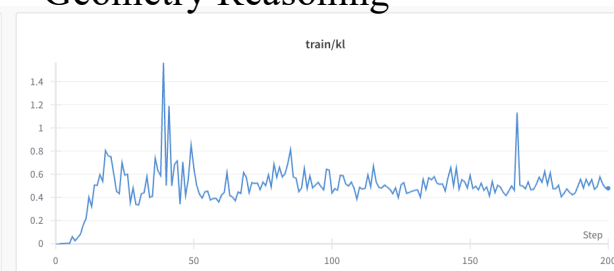
solution

<answer> 80° </answer>

Vision Counting



Geometry Reasoning





# Second Success: Diverse Task——Take Vision as an Example

## VLM-R1

Version	Base VLM	Checkpoint	Task Type
VLM-R1-Qwen2.5VL-3B-OVD-0321	Qwen2.5VL-3B	<a href="#">omlab/VLM-R1-Qwen2.5VL-3B-OVD-0321</a>	Open-Vocabulary Detection
VLM-R1-Qwen2.5VL-3B-Math-0305	Qwen2.5VL-3B	<a href="#">omlab/VLM-R1-Qwen2.5VL-3B-Math-0305</a>	Multi-Modal Math
VLM-R1-Qwen2.5VL-3B-REC-500steps	Qwen2.5VL-3B	<a href="#">omlab/Qwen2.5VL-3B-VLM-R1-REC-500steps</a>	REC/Reasoning-Grounding

Training on RefCOCO+/+g

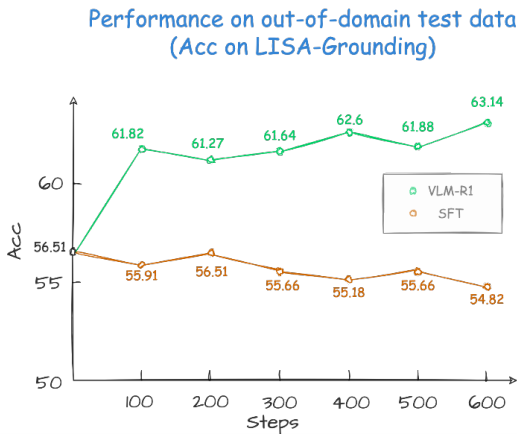
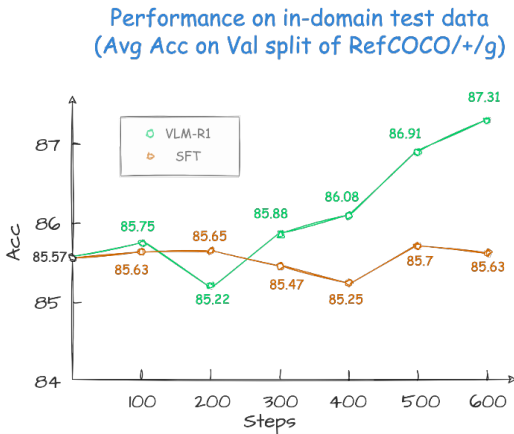


the lady with the blue shirt

Testing on out-of-domain data LISA-Grounding

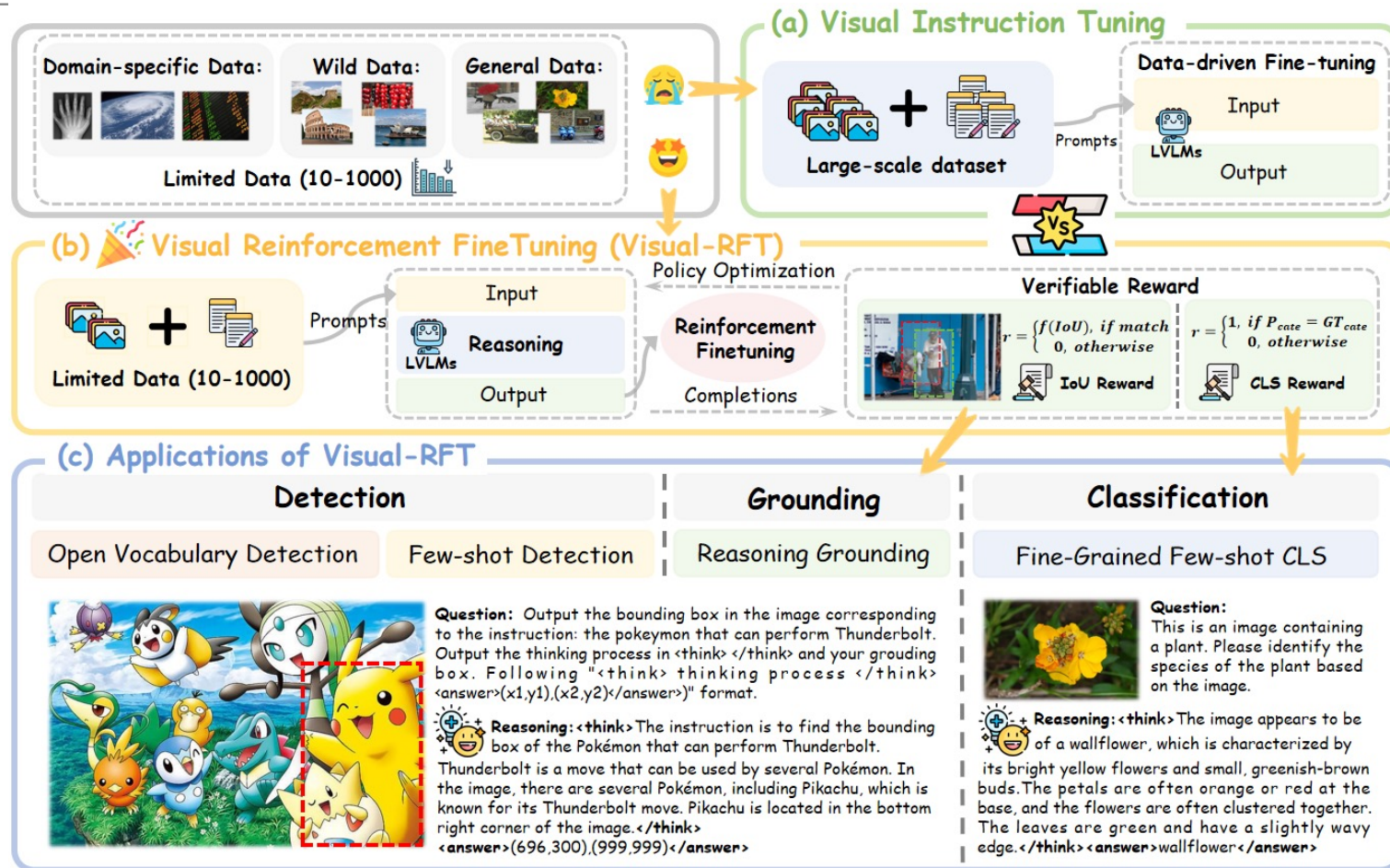


the soccer goalkeeper



# Second Success: Diverse Task——Take Vision as an Example

## Visual-RFT



# Second Success: Diverse Task——Take Vision as an Example

## Seg-Zero



### Vision & Language Prompt

User: Who is most likely to be the *player* in this picture?

### Directly Segment

LISA format

Assistant: Sure, it is < SEG >. In this image, [Captioning Part]

### Reasoning-Chain Guided

Assistant: The player is most likely the one *wearing the baseball uniform*, including the helmet, belt, and baseball glove. The person kneeling down is likely the coach or another adult, as *they are not wearing a uniform*. So, the answer is [SEG-REF]  
In this image, [Captioning Part]

Seg-Zero



### Vision & Language Prompt



User: Who is most likely to be the *speaker* in this picture?

### With CoT

Thinking: The speaker is the person *standing at the podium*, who is the central figure in the image. The podium is a raised platform, typically used for speeches or announcements. ..., and *the podium is the most relevant object to identify the speaker*.



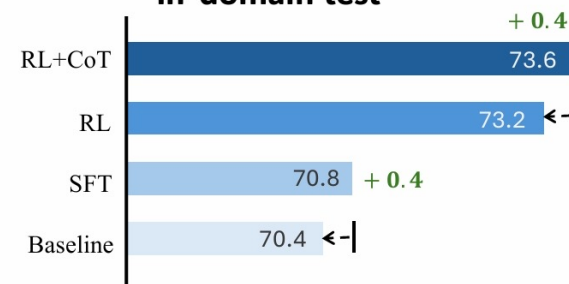
Answer: [SEG-REF]

### Without CoT

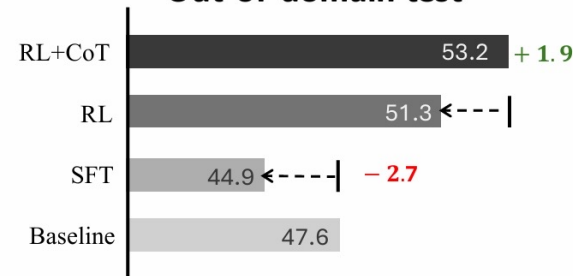


Answer: <SEG>

### In-domain test



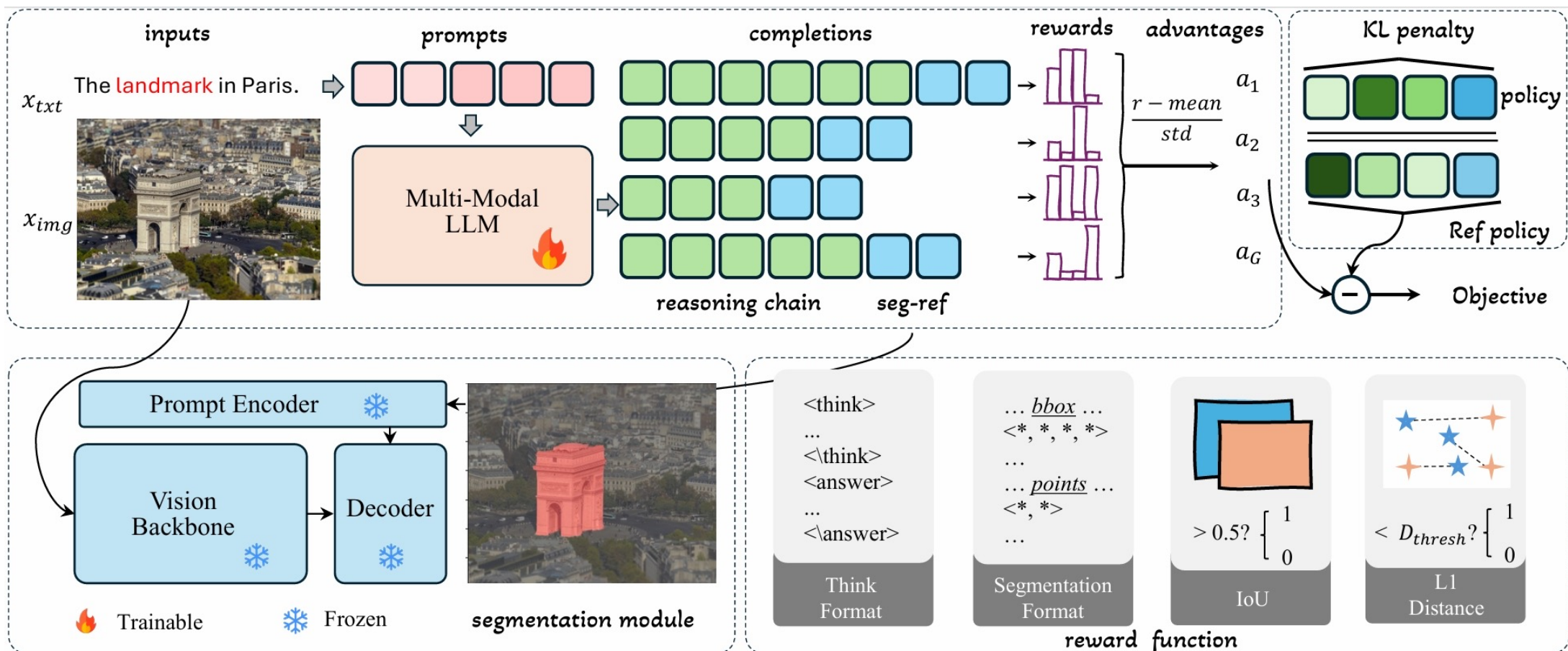
### Out-of-domain test





# Second Success: Diverse Task——Take Vision as an Example

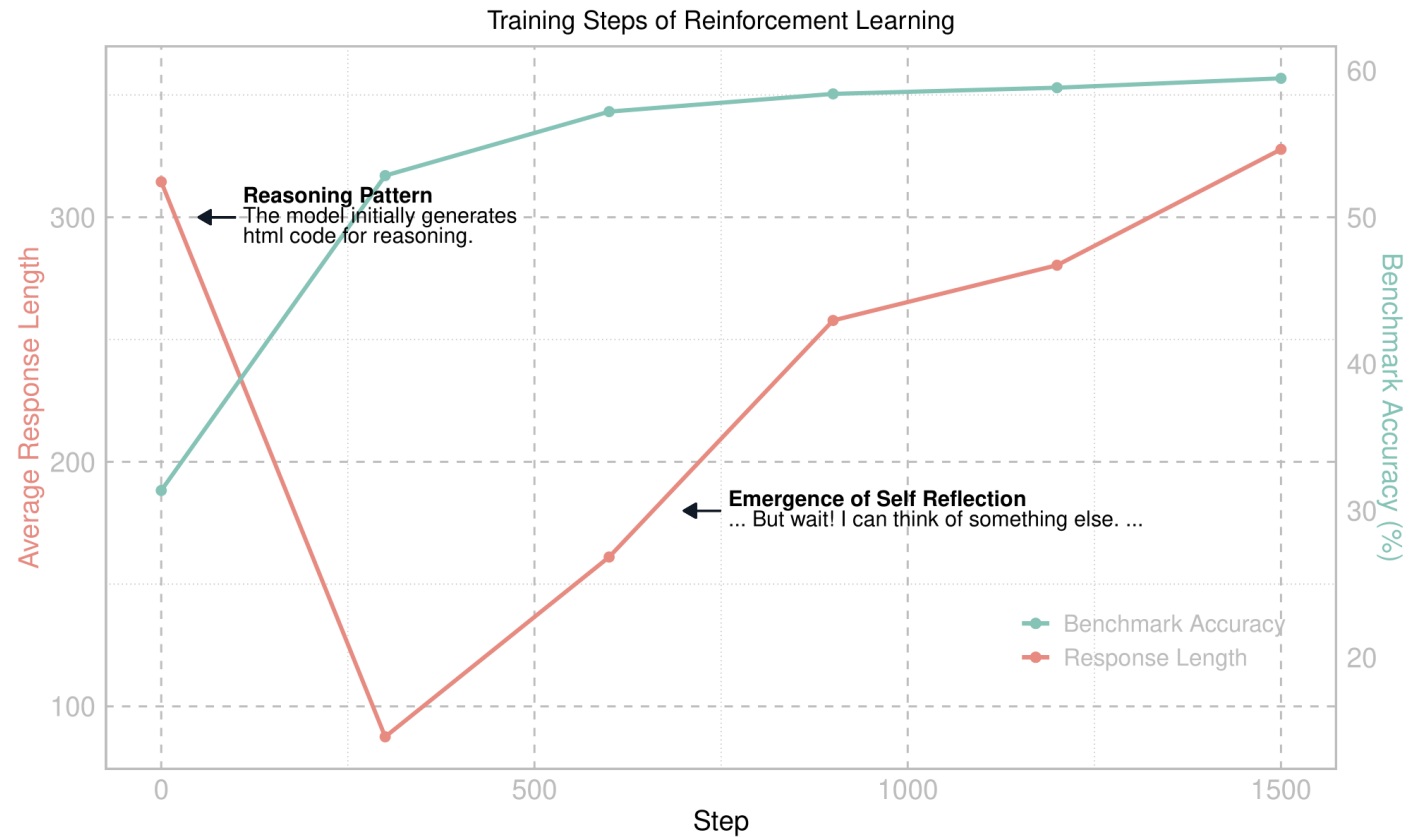
## Seg-Zero



# Second Success: Diverse Task——Take Vision as an Example

## VisualThinker-R1-Zero

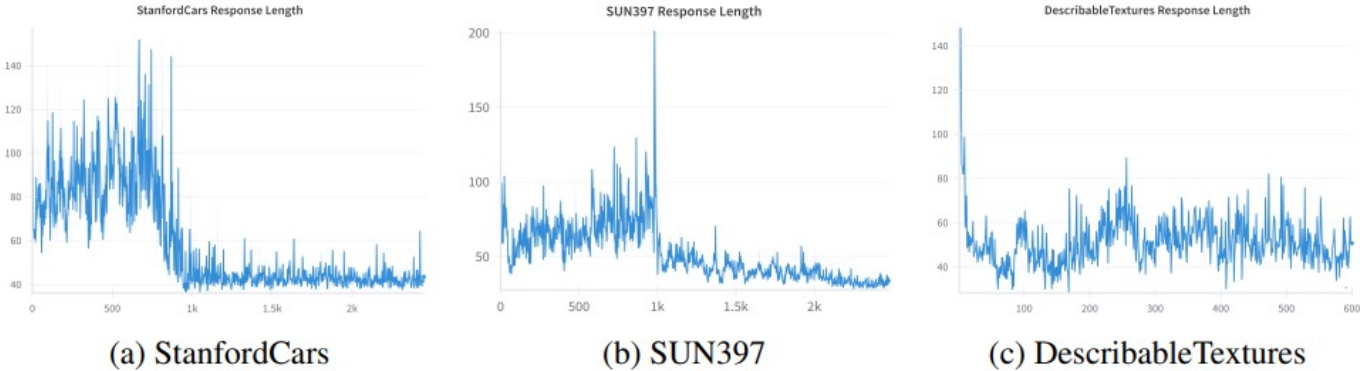
CVBench: Cambrian Vision-Centric Benchmark——2D and 3D understanding



# Second Success: Diverse Task——Take Vision as an Example

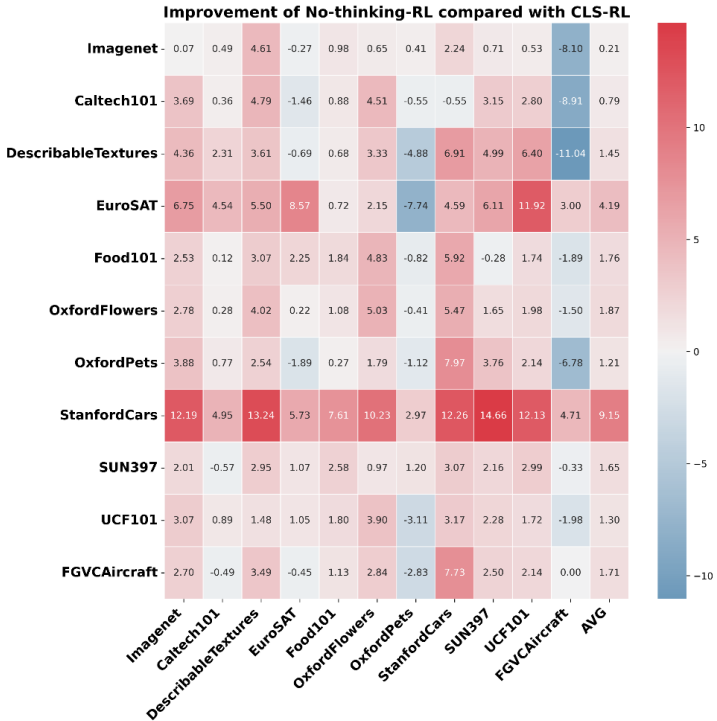
## Think or Not Think

### Image Classification



	ImageNet	Caltech101	DTD	EuroSAT	Food101	Flowers 102	OxfordPets	S.C.	SUN397	UCF101	F.A.	Average
Qwen2VL	70.8	88.56	54.79	45.68	77.54	64.43	73.89	35.77	63.83	66.22	42.75	62.21
SFT	41.60	93.91	71.336	<b>75.16</b>	75.75	<b>96.87</b>	85.80	71.13	41.66	63.81	60.15	70.65
CLS-RL	92.24	98.09	69.92	49.46	88.94	86.56	<b>87.24</b>	80.24	84.57	82.1	74.41	81.25
No-Thinking-RL	<b>92.31</b>	<b>98.46</b>	<b>73.52</b>	58.02	<b>90.78</b>	91.6	86.13	<b>92.5</b>	<b>86.72</b>	<b>83.82</b>	<b>74.41</b>	<b>84.39</b>

Model	Total Accuracy	Count Acc	Relation Acc	Depth Acc	Distance Acc
No-Thinking-RL	0.7676	0.6967	0.8446	0.8067	0.7383
VisualThinker-R1-Zero	0.7036	0.6612	0.8338	0.6850	0.6367





# Second Success: Diverse Task——Take Vision as an Example

## Visual-Spatial Reasoning



Question: Measuring from the closest point of each object, which of these objects (stove, tv, table, sofa) is the closest to the stool?  
Options: A. stove B. tv C. table D. sofa

Ground Truth: D

Vanilla-mode: C

Think-mode:

To determine which object is the closest to the stool, let's analyze the positions of the objects in the video:

1. The stove is located on the left side of the kitchen.
2. The TV is located on the right side of the kitchen.
3. The table is located in the center of the kitchen.
4. The sofa is not visible in the video.

Given this information, the closest object to the stool is the table. So, the correct answer is: C. table.



### Object Count

Question: How many trash can(s)in the room?

Answer: 2

### Object Size

Question: What is the length of the longest dimension (length,width, or height) of the coffee table, measured in centimeters?

Answer: 113

### Room Size

Question: What is the size of this room (in square meters)?

Answer: 47.9

### Relative Direction

Question: If I am standing by the shelf and facing the shower is the bicycle to the left or the right of the shower?

Answer: left

### Absolute Distance

Question: What is the distance between the shower and the kitchen counter (in meters)?

Answer: 6.1

### Relative Distance

Question: Which of these objects (sink, pillow, bed, guitar)is the closest to the bicycle?

Answer: sink

Methods	Eval. Mode	Avg	Obj. Count	Abs. Dist.	Obj. Size	Room Size	Rel. Dist.	Rel. Dir.	Route Plan	Appr. Order
Open-source										
Qwen2-VL-2B	V	23.3	21.4	3.4	32.3	31.1	26.7	27.7	24.7	18.9
+ SFT	V	29.6	29.6	23.5	47.4	33.5	26.9	28.3	28.8	18.6
+ DPO	V	23.9	21.7	3.7	34.8	32.4	27.1	28.5	24.2	18.6
+ vsGRPO-T	V	26.1	24.7	10.7	37.4	36.2	27.3	29.5	25.7	17.9
+ vsGRPO-O	V	28.0	26.2	16.4	44.8	38.2	27.0	29.3	24.2	18.2
+ vsGRPO-T	T	29.6	35.0	28.2	34.7	25.2	28.0	38.5	28.5	18.7
+ vsGRPO-O	O	31.2	34.6	22.5	44.8	33.7	29.4	41.8	26.8	15.8
+ vsGRPO-V	V	35.4	53.6	29.0	52.7	43.4	28.1	30.9	26.8	18.9
Qwen2-VL-7B	V	32.2	39.4	25.0	25.8	43.2	32.6	30.9	27.8	32.6
+ SFT	V	38.1	44.7	27.6	46.1	50.4	34.0	35.7	33.0	33.4
+ DPO	V	32.6	39.1	25.2	26.5	44.2	32.6	30.9	29.3	33.3
+ vsGRPO-V	V	40.7	59.9	29.6	50.8	48.3	35.4	35.6	34.0	31.5
IVL2-2B	V	27.4	21.8	24.9	22.0	35.0	33.8	44.2	30.5	7.1
LNV-7B	V	35.6	48.5	14.0	47.8	24.2	43.5	42.4	34.0	30.6
IVL2-40B	V	36.0	34.9	26.9	46.5	31.8	42.1	32.2	34.0	39.6
LNV-72B	V	40.9	48.9	22.8	57.4	35.3	42.4	36.7	35.0	48.6
Close-source										
GPT-4o	V	34.0	46.2	5.3	43.8	38.2	37.0	41.3	31.5	28.5
Gemini-1.5 Pro	V	48.8	49.6	28.8	58.6	49.4	46.0	48.1	42.0	68.0

# Second Success: Diverse Task——Take Vision as an Example

## MMR1, MM\_Eureka

Visual Math Reasoning

Model	size	MathVista	MathVision	LogicVista	MathVerse_V	MathVerse
Close-sourced						
<a href="#">GPT-4o 1120</a>	-	60.0	31.2	52.8	40.6	-
<a href="#">Gemini-2.0-flash</a>	-	70.4	43.6	52.3	47.8	-
<a href="#">Claude3.7-Sonnet</a>	-	66.8	41.9	58.2	46.7	-
R1-related						
<a href="#">LLaVA-CoT</a>	11B	52.5	19.9	39.6	22.6	-
<a href="#">Open-R1-Multimodal</a>	7B	60.6	-	-	-	-
<a href="#">Mulberry</a>	7B	63.1	-	-	-	-
<a href="#">LMM-R1</a>	3B	63.2	26.4	-	-	41.6
<a href="#">R1-Onevision</a>	7B	-	26.2	-	-	44.1
<a href="#">MM-Eureka</a>	8B	67.1	22.2	-	-	40.4
<a href="#">MM-Eureka</a>	38B	64.2	26.6	-	-	48.9
Open-sourced						
<a href="#">Ovis2-8b</a>	8B	71.8	25.9	39.4	42.3	-
<a href="#">MiniCPM-o-2.6</a>	8B	71.9	21.7	36.0	35.0	-
<a href="#">VITA-1.5</a>	7B	66.2	19.5	38.9	-	23.4
<a href="#">Qwen2.5-VL</a> (official)	7B	68.2	25.4	47.9	41.1	-
<a href="#">Qwen2.5-VL</a> (reproduced)	7B	67.5	25.6	46.8	42.5	46.9
Ours						
MMR1-math-v0	7B	71.0	30.2	50.8	45.1	49.8

Model	Base Model	MathVista	MathVerse	MathVision	OlympidBench	K12
Qwen2.5-VL-7B-Instruct	-	68.2	47.9	25.4	15.3	36.0
Qwen2.5-VL-32B-Instruct	-	74.7	49.4	40.0	33.3	44.6
InternVL2.5-VL-8B-Instruct	-	64.4	39.5	19.7	8.0	24.8
InternVL2.5-VL-38B-Instruct	-	71.9	49.4	31.8	29.3	37.2
MM-EUREKA-InternVL-8B	InternVL2.5-7B-Instruct	67.1	40.4	22.2	8.6	27.0
MM-EUREKA-Qwen-7B (flush)	Qwen2.5VL-7B-Instruct	72.7 (+4.5)	48.3 (+0.4)	25.5 (+0.1)	26.6 (+11.3)	49.0 (+13.0)
MM-EUREKA-Qwen-7B (clear)	Qwen2.5VL-7B-Instruct	73.0 (+4.8)	50.3 (+2.4)	26.9 (+1.5)	25.3 (+10.0)	48.6 (+12.6)

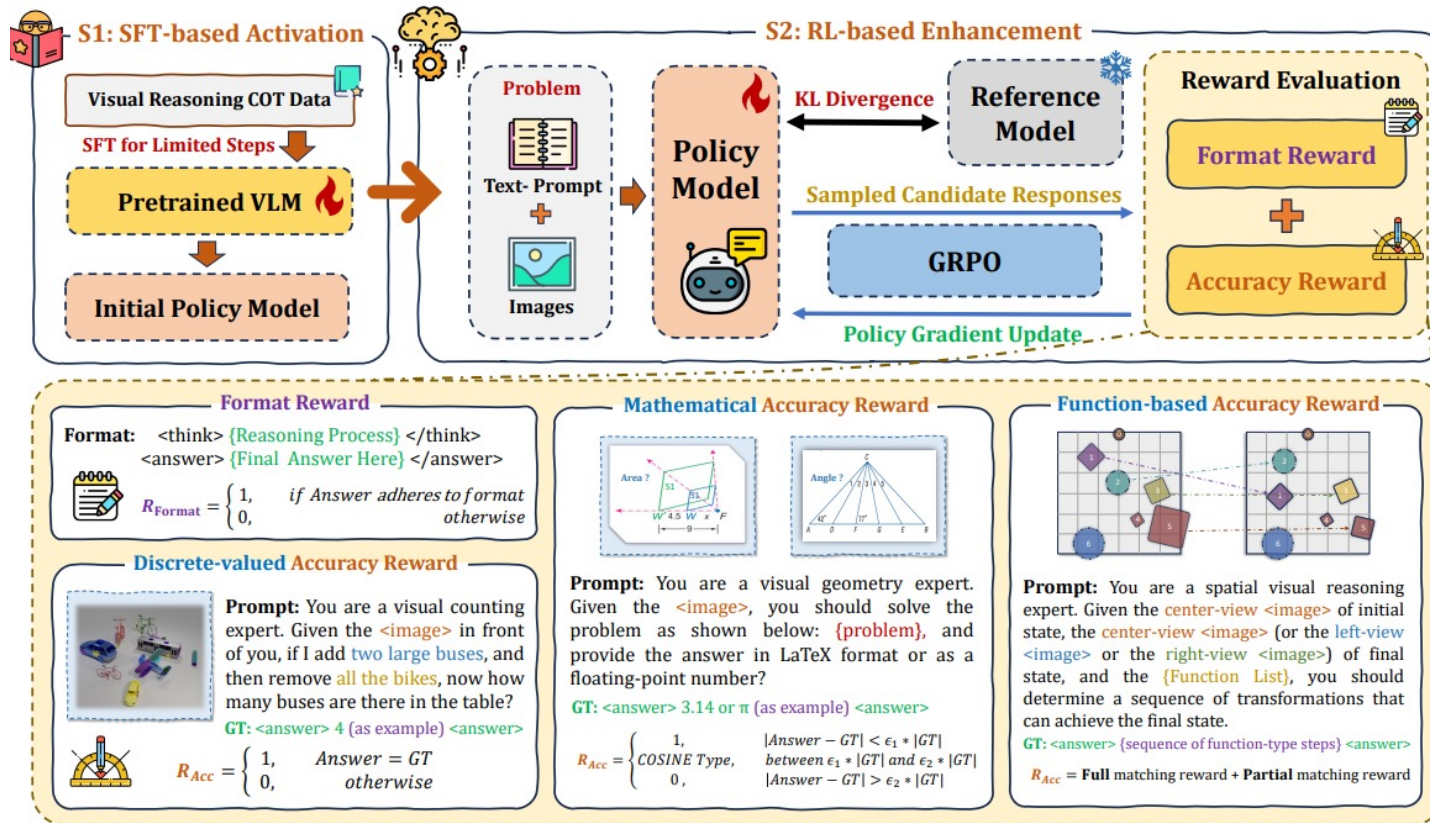




# Second Success: Diverse Task——Take Vision as an Example

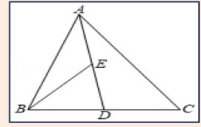
## Reason-RFT

Visual Counting, Structure Perception, Spatial Transformation



# Third Success: Better Algorithms——Take Vision as an Example

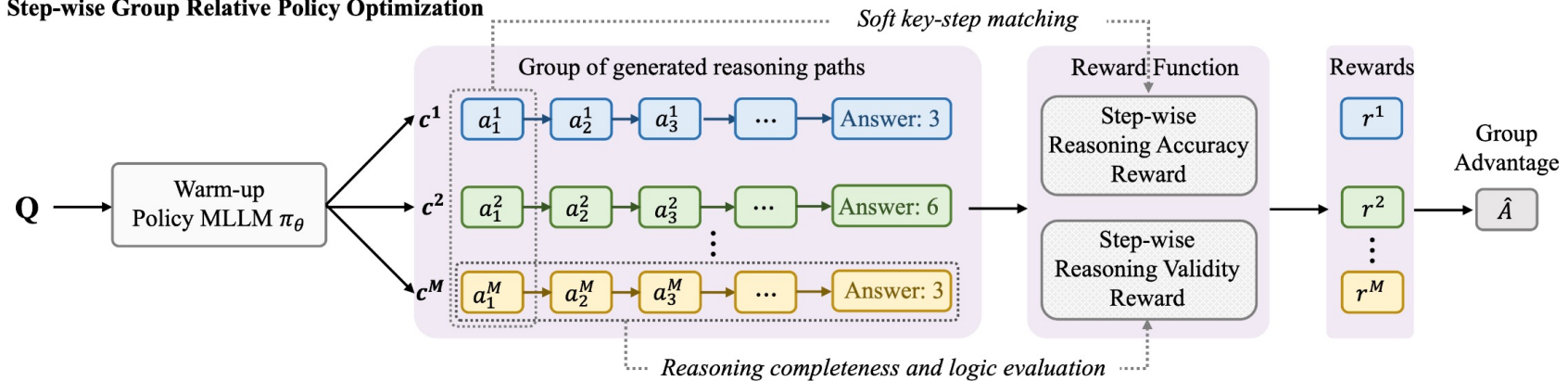
## R1-VL



**Question:** In the given diagram, triangle ABC has AD as its median and point E is the midpoint of AD. If the area of triangle ABC is 12, what is the area of triangle ABE?

**Answer:** *Step 1:* Since **AD is a median**, it divides triangle ABC into two equal areas: ABD and ACD. *Step 2:* Segment **AE is half of AD**, splitting triangle ABD into two triangles of **equal area**: ABE and BED. *Step 3:* The area of triangle ABD is half of triangle ABC, which is  **$\frac{12}{2} = 6$** . *Step 4:* Since **E is the midpoint** of AD, triangle ABE is half of triangle ABD. Therefore, the area of triangle ABE is  **$\frac{6}{2} = 3$** . *The final answer is 3.*

### Step-wise Group Relative Policy Optimization



#### (a) Step-wise Reasoning Accuracy Reward

**Pre-extracted key steps with Augmentations:**

1. AD is a median; median is AD
2. equal area; ...
3. AE is half of AD;  $AE = 1/2 AD$
4.  $\frac{12}{2} = 6$ ;  $12/2 = 6$ , ...
5. E is the midpoint; ...
6.  $\frac{6}{2} = 3$ ;  $6/2 = 3$ , ...

**Soft key-step matching :**

#Description: The image shows ...; #Rationale: The question asks for the area...; #Step1: ... we find **AD is a median** of ...; #Step2: ... AE splits triangle ABD ...; #Step3: ... The area of triangle ABD is  $12/2 = 6$ , ..., and the area of triangle ABE is  $\frac{6}{2} = 3$ . #The final answer is: 3.

**Step-wise Matching score: 3/6**

#### (b) Step-wise Reasoning Validity Reward

#Description → #Rationale → #Step1 → ... → #Step N → #Answer. ✓

i. Reasoning completeness

#Description → #Rationale → #Answer. ✗ Missing reasoning steps

#Description → #Step1 → ... → #Step N → #Answer. ✗ Missing rationale

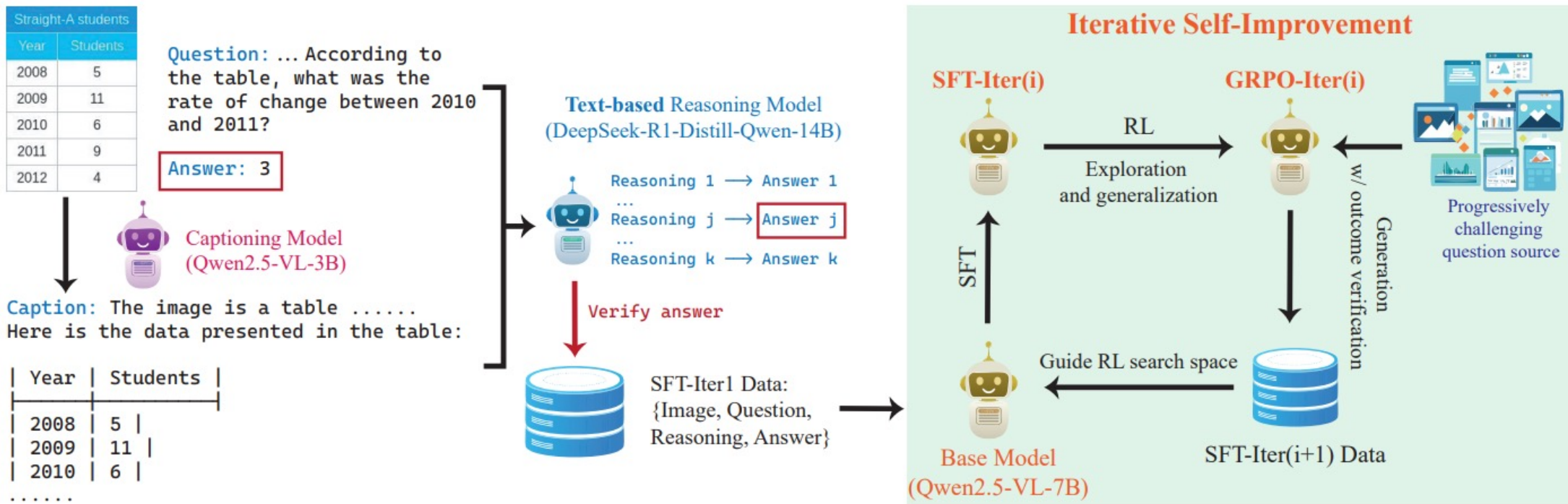
ii. Reasoning logic

#Description → #Rationale → #Answer → #Step1 ... → #Step N. ✗

#Description → #Step3 → #Rationale → ... → #Step 1 → #Answer ✗

# Third Success: Better Algorithms——Take Vision as an Example

## OpenVLThinker: Iterative Self-Improvement





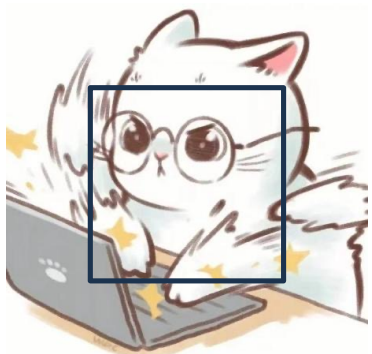
# **What could the community do next?**



# To Do 1: Focus Further than Textual Modality

Take Vision Images as an Example

---



**Center Crop**



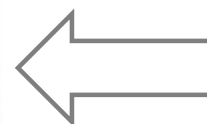
**Flip**



**Invert**



**rotation**



**Initial**



**Random Crop**



**Affine**



**Diffusion (Add Noise)**

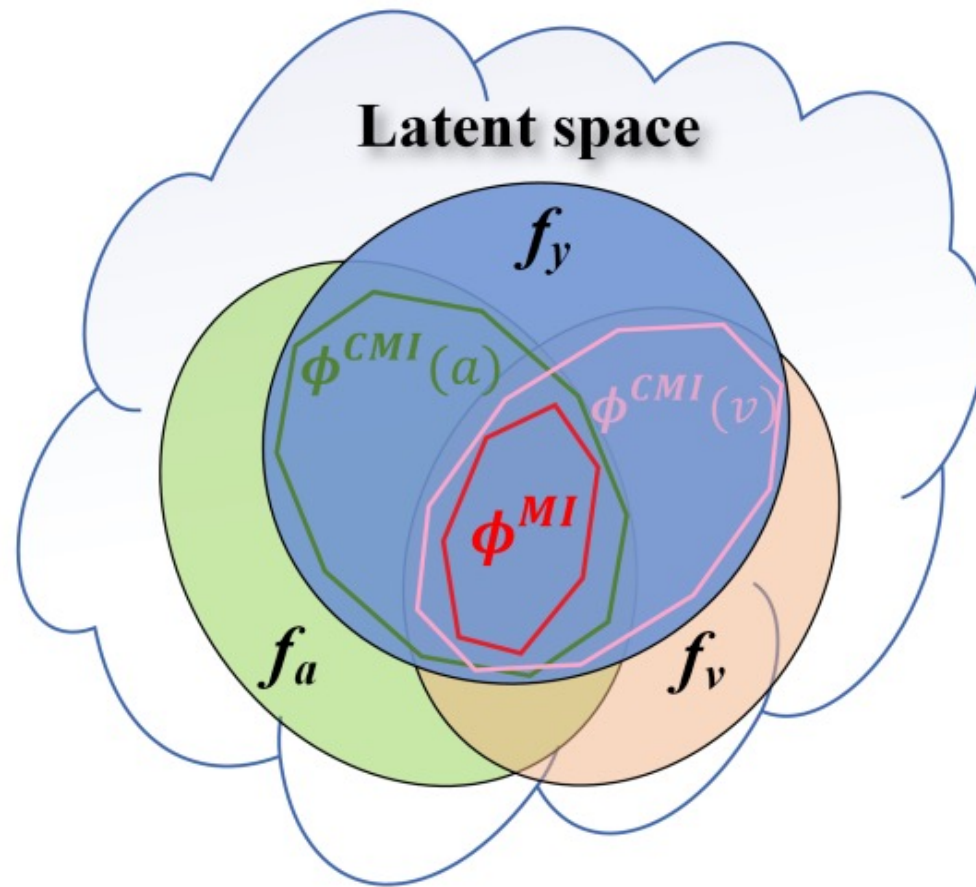


**Mixed**



## To Do 2: Give Attention to Multi-modal Asymmetric

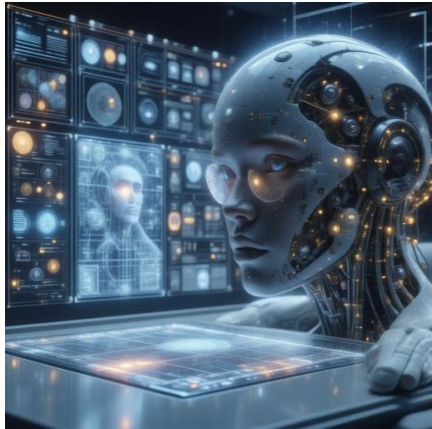
---





# To Do 3: Call for Multimodal Reasoning Agents

---



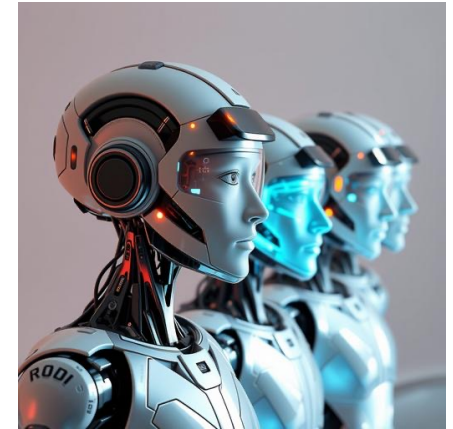
**Deep Reasoning**



**Trustworthy Action**




**Environmental Aware**



**Multi-Agent System**

## References

---

1. Awesome RL-based Reasoning MLLMs, Sun Haoyuan, Tsinghua University, <https://github.com/Sun-Haoyuan23>
  2. Multi Modal Machine Learning, 11-777 • Fall 2023 • Carnegie Mellon University
  3. Reinforcement Learning and LLMs, Philipp Krähenbühl, UT Austin
  4. Reinforcement Learning in DeepSeek Models Actor-Critic, TRPO, PPO, and GRPO, Yanjie Li Hong Kong Polytechnic University
- 

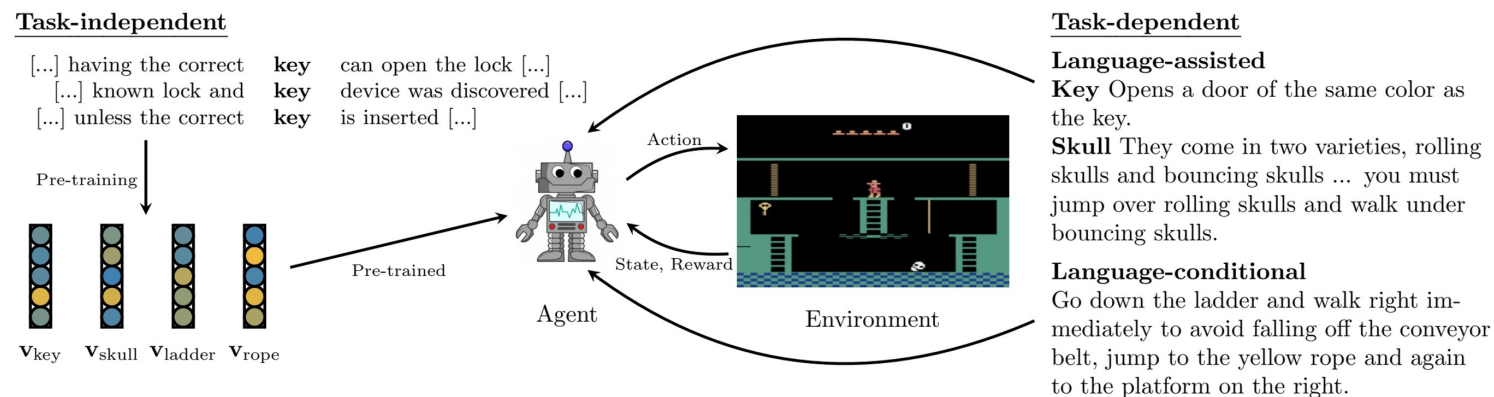


---

# Thanks



# Back to Reasoning: Interactive Reasoning



# Language-conditional RL: Instruction Following

---

Language specifies the task



Go to the green torch

**Train**

Go to the short red torch  
Go to the blue keycard  
Go to the largest yellow object  
Go to the green object



**Test**

Go to the tall green torch  
Go to the red keycard  
Go to the smallest blue object

**Fusion**

**Alignment**

Ground language

Recognize objects

Navigate to objects

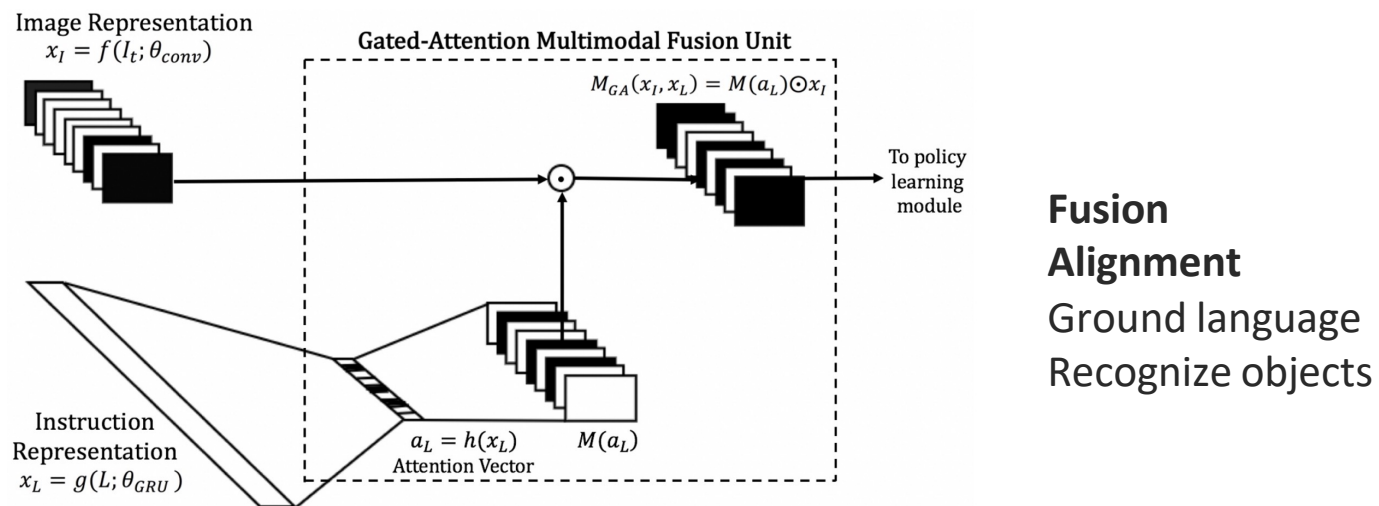
Generalize to unseen objects

[Misra et al., Mapping Instructions and Visual Observations to Actions with Reinforcement Learning. EMNLP 2017]

[Chaplot et al., Gated-Attention Architectures for Task-Oriented Language Grounding. AAAI 2018]

# Language-conditional RL: Instruction Following

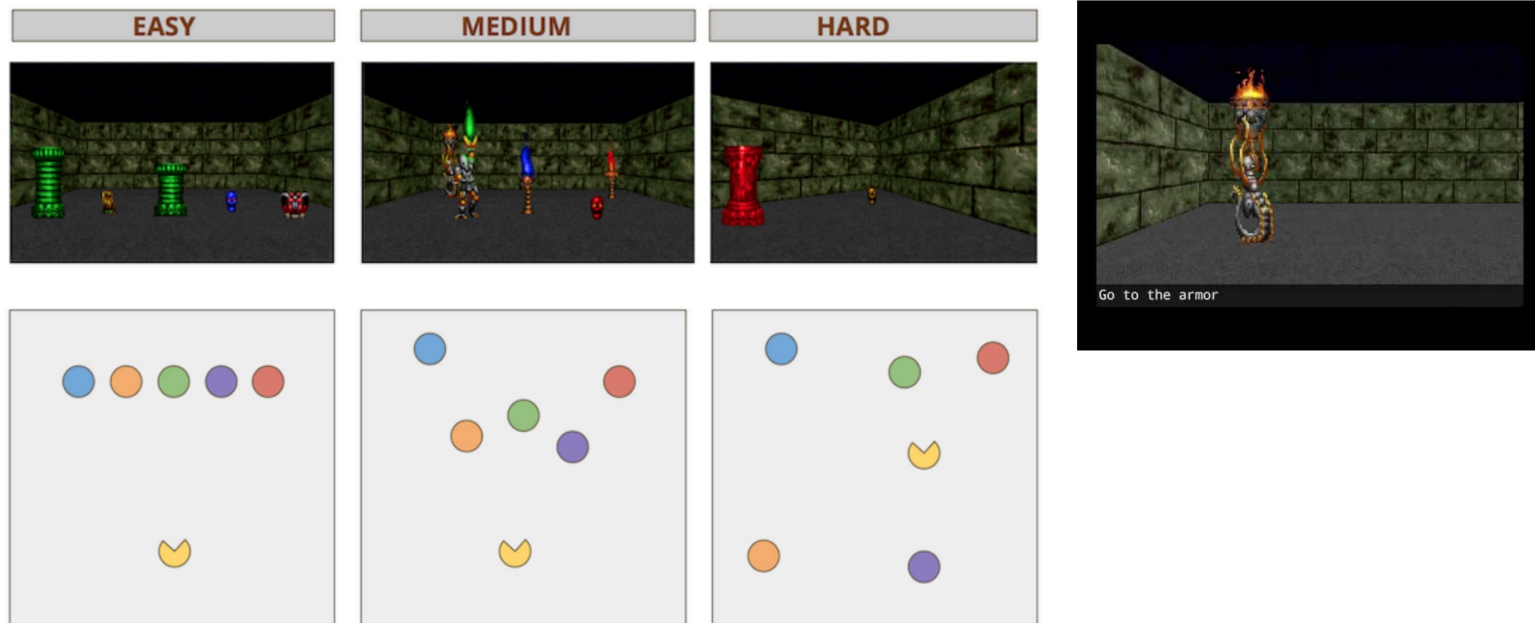
- Gated attention via element-wise product



[Chaplot et al., Gated-Attention Architectures for Task-Oriented Language Grounding. AAAI 2018]

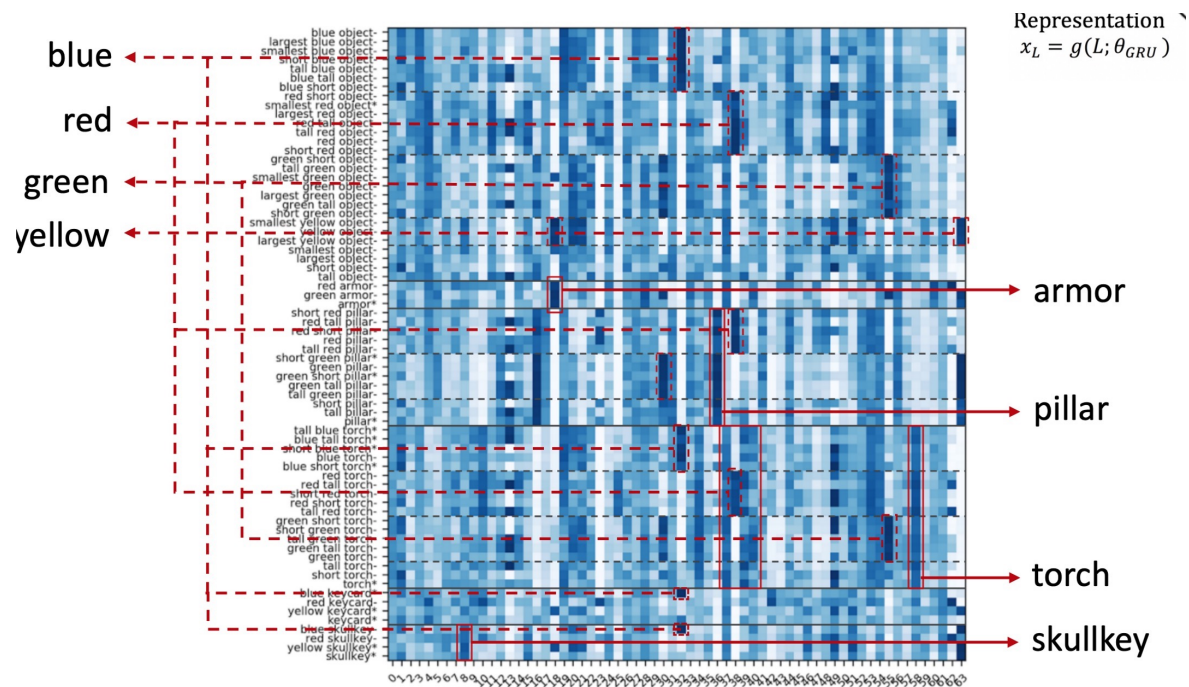
# Language-conditional RL: Instruction Following

---



[Chaplot et al., Gated-Attention Architectures for Task-Oriented Language Grounding. AAAI 2018]

# Language-conditional RL: Instruction Following

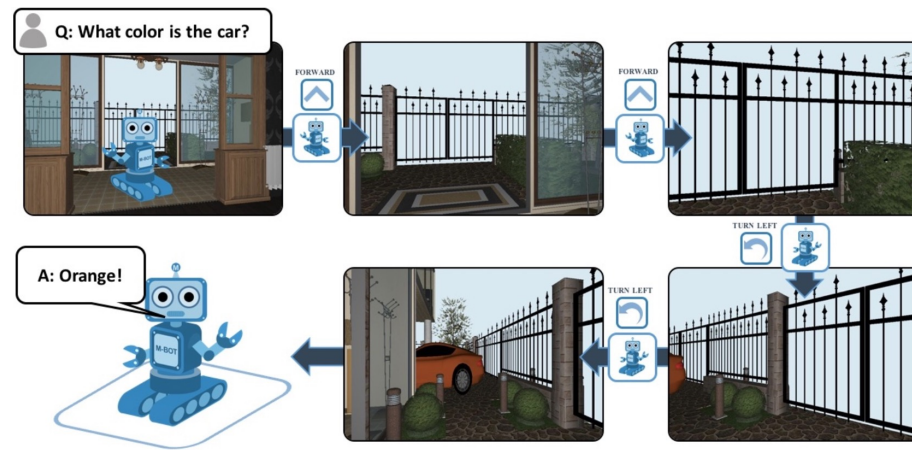


Grounding is  
important for  
generalization  
blue armor, red pillar  
-> blue pillar

[Chaplot et al., Gated-Attention Architectures for Task-Oriented Language Grounding. AAAI 2018]

# Language-conditional RL: Embodied QA

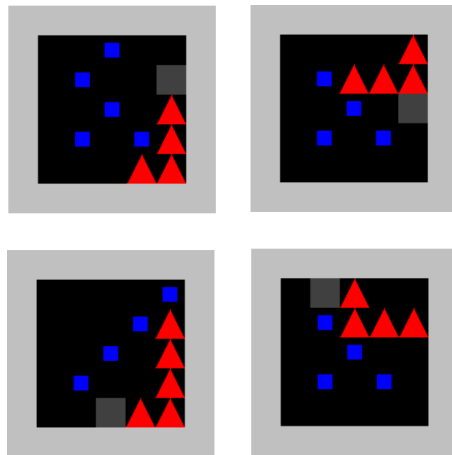
## Navigation + QA



[Das et al., Embodied Question Answering. CVPR 2018]

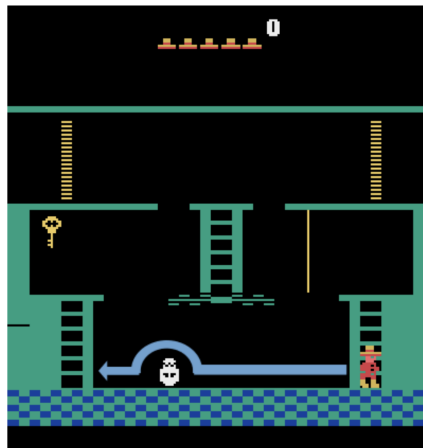
# Language-assisted RL: Language to Rewards

Language specifies the rewards rather than actions



*“build an L-like shape  
from red blocks”*

Goal specification  
(Bahdanau et al. 2019)



*“Jump over the skull  
while going to the  
left”*

Reward shaping  
(Goyal et al. 2019)

JetBlue		Delta	
longest stop	2h	longest stop	2h
price	\$100	price	\$10

*“I prefer JetBlue,  
even if it’s  
expensive”*

Preferences  
(Lin et al. 2022)

<https://arxiv.org/abs/1806.01946>,  
<https://arxiv.org/abs/1902.07742>,  
<https://www.ijcai.org/proceedings/2019/331>,  
<https://arxiv.org/abs/2204.02515>

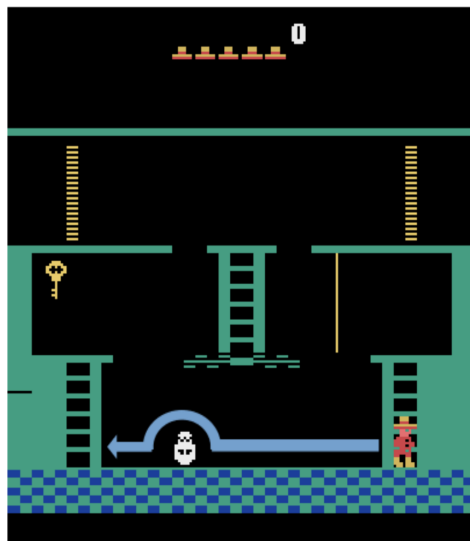
[Goyal et al., Using Natural Language for Reward Shaping in Reinforcement Learning. IJCAI 2019]



# Language-assisted RL: Language to Rewards

---

Language specifies the rewards rather than actions



Montezuma's  
revenge

Sparse, long-term reward problem

General solution: reward shaping via auxiliary rewards

Natural language for reward shaping

← *"Jump over the skull while going to the left"*

from Amazon Mturk :-(  
asked annotators to play the  
game and describe entities

Intermediate rewards to speed up learning

# Language-assisted RL: Domain knowledge

---

Language as domain knowledge – instruction manuals



*The natural resources available where a population settles affects its ability to produce food and goods. Build your city on a plains or grassland square with a river running through it if possible.*

Figure 1: An excerpt from the user manual of the game Civilization II.

# Language-assisted RL: Domain knowledge

## Language as domain knowledge – instruction manuals



*The natural resources available where a population settles affects its ability to produce food and goods. Build your city on a plains or grassland square with a river running through it if possible.*

### Map tile attributes:

- Terrain type (e.g., grassland, mountain, etc)
- Tile resources (e.g. wheat, coal, wildlife, etc)

### City attributes:

- City population
- Amount of food produced

### Unit attributes:

- Unit type (e.g., worker, explorer, archer, etc)
- Is unit in a city ?

1. Choose **relevant** sentences
2. Label words into **action-description**, **state-description**, or **background**

# Language-assisted RL: Domain knowledge

## Language as domain knowledge – instruction manuals



- Phalanxes are twice as effective at defending cities as warriors. ✓
- Build the city on plains or grassland with a river running through it. ✓
- You can rename the city if you like, but we'll refer to it as washington.
- There are many different strategies dictating the order in which advances are researched

Relevant sentences

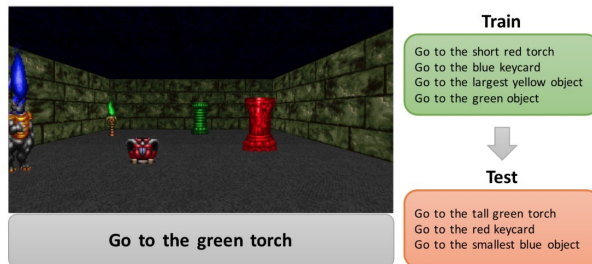
- After the road is built, use the settlers to start improving the terrain.  
S S S A A A A A
- When the settlers becomes active, chose build road.  
S S S A A A
- Use settlers or engineers to improve a terrain square within the city radius  
A S X A A S A X S S S S

A: action-description

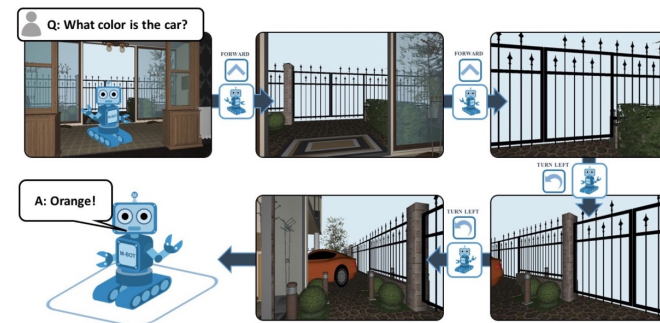
S: state-description

# Summary: Interactive Reasoning

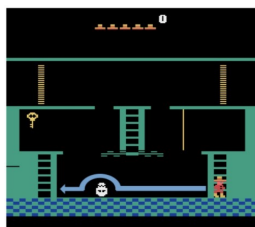
## Instruction following



## Embodied QA



## Reward shaping



*"Jump over the skull while going to the left"*

## Domain knowledge



*The natural resources available where a population settles affects its ability to produce food and goods. Build your city on a plains or grassland square with a river running through it if possible.*

Figure 1: An excerpt from the user manual of the game Civilization II.





# Interactive Reasoning Challenges

## Learning from open-ended manuals



A L I E N  
20th Century Fox  
Games of the Century  
(picture of the ALIEN movie poster)  
"In space no one can hear you scream"  
Game Instructions  
Fox Video Games

A L I E N

TO SET UP: Set up your video computer system and left joystick controller as instructed in your manufacturer owner's manual. Move the Color/B-W lever to the correct setting. Turn the power OFF and insert the Alien game cartridge.

(Screen shot of the ALIEN maze setup: Alien, Alien Egg, Human, Pulsar and Play Level-demo mode only)

TO BEGIN: Turn the power ON. Use the Game Select lever and Difficulty Switches to choose a play level. Press the Game Reset lever and get ready to run for your life.

THE OBJECTIVE: Your job is to run through the hallways of your space ship and crush all the Alien Eggs which have been placed there. You must also avoid or destroy the adult Aliens and snatch up as many prizes as possible.

THE CONTROLS: Tilt the joystick forward, backward, left and right to maneuver through the hallways. To smash Eggs, simply run over them. You may travel off one side of the maze and back into the other using the "Hyperwarp Passage." Each Human is equipped with a Flame Thrower that is activated by the joystick button (see below).

SCREEN DISPLAY: The Play Level and Humans allowed per Play Level are displayed in the bottom left corner of the screen when Alien is not in play. During the game, the current score and Humans remaining are shown there.

LEVELS OF PLAY/DIFFICULTY SWITCHES/BONUS ROUNDS: Each game of Alien lasts until you run out of Humans. If you can clear all of the Eggs out of a playing screen, you get the chance to earn extra points in a "Bonus Round" and then are returned to a new and more difficult playing screen. All points and Humans remaining are carried over to the new screens.

Bonus Rounds: The object of the Bonus Round is to travel STRAIGHT UP to the top of the screen and grab the prize shown there. You have only eight seconds to do so. You do not lose a human if you fail, but you earn the point value of the prize if you succeed.

Left Difficulty Switch A: Aliens travel in random order about the screen.

Left Difficulty Switch B: Aliens travel in fixed patterns about the screen.

Right Difficult Switch B: Capturing a Pulsar has standard effect on the Aliens.

Right Difficulty Switch A: Capturing a Pulsar has no effect on the Aliens.

(Screen shot of ALIEN maze: Flame Thrower, Prize, Hyperwarp Passages, Humans Remaining and Current Score)

LEVEL 1 - NORMAL GAME PLAY: You begin with three Humans and receive a bonus Human after successfully clearing the second screen. Prizes appear in chart order.

LEVEL 2 - ADVANCED GAME PLAY: You begin with two Humans and receive no bonus Humans. Prizes appear in chart order.

LEVEL 3 - FOR EXPERTS ONLY: You begin with three Humans and receive no bonus Human after clearing the first screen. All Prizes in Level 3 are Saturns.

LEVEL 4 - EASY PRACTICE GAME: You begin with six Humans and receive 1 bonus Human after clearing the first scene. All Prizes in Level 4 are also Saturns.

OBJECTS/SCORING: Each time an Alien catches you, one Human is lost. You score points for smashing Eggs and frying Aliens with the aid of your Flame Thrower or Pulsar. In addition, you can gain points for picking up Prizes. Be sure to record your high scores on the back of this booklet!

(Screen shot of the bonus round with the human at the bottom of the screen, the prize at the top of the screen and the horizontal moving Aliens in the centre portion -- similar to the road portion of Frogger.)

FLAME THROWER - 1 PER HUMAN: A spurt of flame from this contraption cause Aliens to turn away from you or become immobilized for a short period of time. Use the Throwers carefully. Each has only four seconds of flame and the Thrower will not operate in the extreme left or right areas of the screen. You can also use the Flame Thrower to run over a Pulsar without picking it up, allowing you to save the Pulsar to use at a later time.

PULSARS - 3 PER MAZE: Capturing a Pulsar causes the Aliens to weaken and turn blue. Then, for a short period of time, you can destroy them by running over and touching them. The instant the Aliens return to their original color, however, they once again become deadly.

PRIZES - 2 PER MAZE: Prizes appear in all levels of play and in the Bonus Rounds.

POINT CHART:

OBJECT	POINTS	PRIZES	POINTS	POINTS
Eggs		10	Rocket	500
Pulsar	100		Saturn	1,000
1st Alien		500	Star Ship	2,000
2nd Alien		1,000	1st Surprise	2,000-3,000
3rd Alien		2,000	2nd Surprise	3,000
Completed Screen		1	3rd Surprise	5,000

HINTS FROM DALLAS NORTH...

A good playing strategy is to crush all of the Eggs in one area at a time, keeping within easy reach of a Pulsar. The best way to destroy Aliens is to sit near a Pulsar until the Aliens are almost upon you. Then grab that Pulsar and go get 'em!

Use the Hyperwarp Passage to ditch Aliens. Many times they won't follow you in.

If you're having trouble with the Bonus Rounds, try going between the Alien pairs rather than around them.

SUPER SMASHERS (a place to enter your high scores)

Name	Level	Score
------	-------	-------

# Interactive Reasoning Challenges

Open  
challenges

## Learning from text-based games



[Zhong et al., SILG: The Multi-environment Symbolic Interactive Language Grounding Benchmark. NeurIPS 2021]

# Interactive Reasoning Challenges

Open  
challenges

Learning from lots of offline data



[Fan et al., MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge. arXiv 2022]



# Interactive Reasoning Challenges

Open  
challenges

Hard to specify reward, but only final goal



[Habitat Rearrangement Challenge 2022]

# Summary: RL Methods

---

Epsilon greedy + exploration

Experience replay

Decorrelate samples

Fixed targets

› Value Based

Value iteration  
Policy iteration  
(Deep) Q-learning

- Learned Value Function
- Implicit policy (e.g.  $\epsilon$ -greedy)

› Policy Based

Policy gradients

- No Value Function
- Learned Policy

Variance reduction with a baseline

› Actor-Critic

Actor (policy)  
Critic (Q-values)

- Learned Value Function
- Learned Policy

