



# 强化学习原理及应用 Reinforcement Learning (RL): Theories & Applications

*DCS6289 Spring 2022*

Biyang Hu (胡比洋)

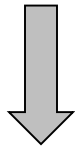
School of Computer Science and Engineering  
Sun Yat-Sen University

# Lecture 8: Model-based RL

26<sup>th</sup> April 2022

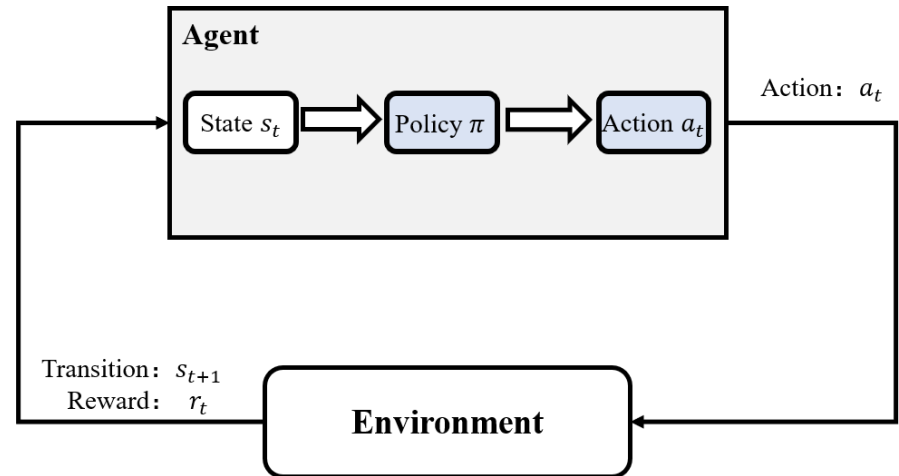
# Quick Recap

- States  $S \in \mathbb{R}^S$
- Actions  $A \in \mathbb{R}^a$
- Reward Function  $R: S \times A \rightarrow \mathbb{R}$
- Transition Function  $T: S \times A \rightarrow S$
- Discounted Factor  $\gamma \in (0,1)$
- Policy  $\pi: S \rightarrow A$



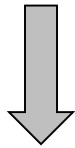
$$\max_{\pi} \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$$

Subject to  $a_t = \pi(s_t), s_{t+1} = T(s_t, a_t)$



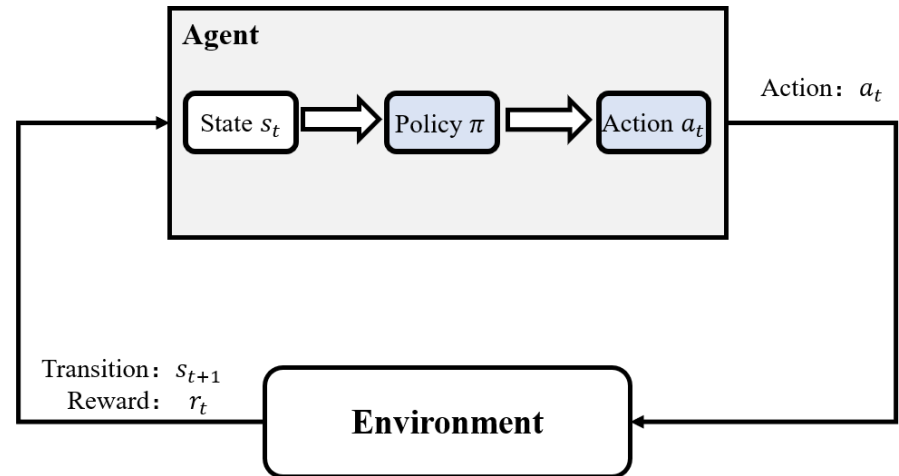
# Quick Recap

- States  $S \in \mathbb{R}^S$
- Actions  $A \in \mathbb{R}^a$
- Reward Function  $R: S \times A \rightarrow \mathbb{R}$
- Transition Function  $T: S \times A \rightarrow S$
- Discounted Factor  $\gamma \in (0,1)$
- Policy  $\pi: S \rightarrow A$



$$\max_{\pi} \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$$

Subject to  $a_t = \pi(s_t), s_{t+1} = T(s_t, a_t)$





# Model-free RL vs. Model-based RL

**Collect data**

$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^T$$

**Model-free:** learn policy directly from data

$$\mathcal{D} \rightarrow \pi \quad \text{e.g. } Q\text{-learning, policy gradient}$$

**Model-based:** learn model, then use it to learn or improve a policy

$$\mathcal{D} \rightarrow f \rightarrow \pi$$

# Model-free RL vs. Model-based RL

**Collect data**

$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^T$$

**Model-free:** learn policy directly from data

$$\mathcal{D} \rightarrow \pi \quad \text{e.g. } Q\text{-learning, policy gradient}$$

**Model-based:** learn model, then use it to learn or improve a policy

$$\mathcal{D} \rightarrow f \rightarrow \pi$$

## □ Why Model-based ?

|                        | Model-free | Model-based |
|------------------------|------------|-------------|
| Asymptotic Performance | +          | - / +       |
| Computation            | +          | -           |
| Sample Efficiency      | -          | +           |
| Exploration            | -          | +           |

# Model-free RL vs. Model-based RL

**Collect data**

$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^T$$

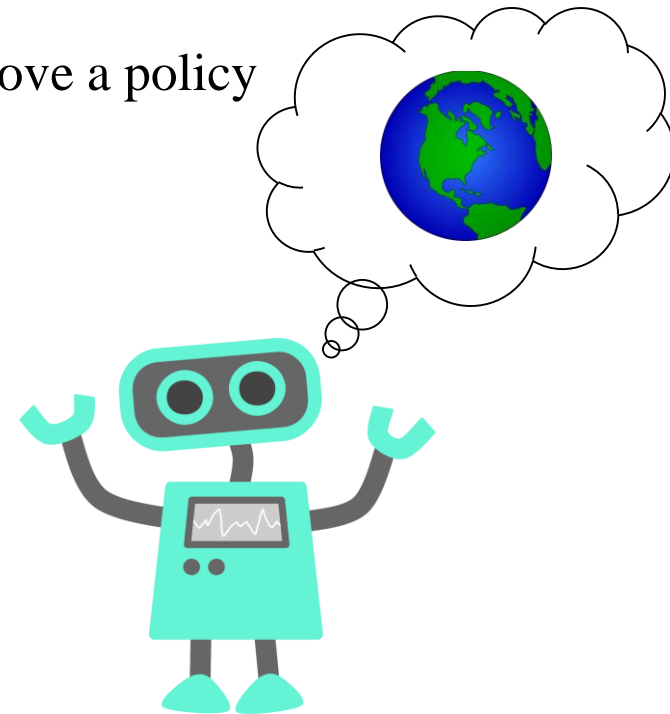
**Model-free:** learn policy directly from data

$$\mathcal{D} \rightarrow \pi \quad \text{e.g. } Q\text{-learning, policy gradient}$$

**Model-based:** learn model, then use it to learn or improve a policy

$$\mathcal{D} \rightarrow f \rightarrow \pi$$

□ **What is a Model ?**



# What is a Model ?

## Collect data

$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^T$$

**Model-free:** learn policy directly from data

$$\mathcal{D} \rightarrow \pi \quad \text{e.g. } Q\text{-learning, policy gradient}$$

**Model-based:** learn model, then use it to learn or improve a policy

$$\mathcal{D} \rightarrow f \rightarrow \pi$$

## □ What is a Model ?

**Definition:** a model is a representation that **explicitly** encodes knowledge about the structure of the environment and task. `state_next, reward, terminate, infos = env.step(action)`

**A transition/dynamics model:**

$$s_{t+1} = f_s(s_t, a_t)$$

**A rewards model:**

$$r_{t+1} = f_r(s_t, a_t)$$

**A inverse transition/dynamics model:**

$$a_t = f_s^{-1}(s_t, s_{t+1})$$



# What is a Model ?

## Collect data

$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^T$$

**Model-free:** learn policy directly from data

$$\mathcal{D} \rightarrow \pi \quad \text{e.g. } Q\text{-learning, policy gradient}$$

**Model-based:** learn model, then use it to learn or improve a policy

$$\mathcal{D} \rightarrow f \rightarrow \pi$$

## □ What is a Model ?

**Definition:** a model is a representation that **explicitly** encodes knowledge about the structure of the environment and task. `state_next, reward, terminate, infos = env.step(action)`

**A transition/dynamics model:**

$$s_{t+1} = f_s(s_t, a_t)$$

**A rewards model:**

$$r_{t+1} = f_r(s_t, a_t)$$

**A inverse transition/dynamics model:**

$$a_t = f_s^{-1}(s_t, s_{t+1})$$

Typically what is  
meant by the model  
in model-based RL

# What is a Model ?

**Collect data**

$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^T$$

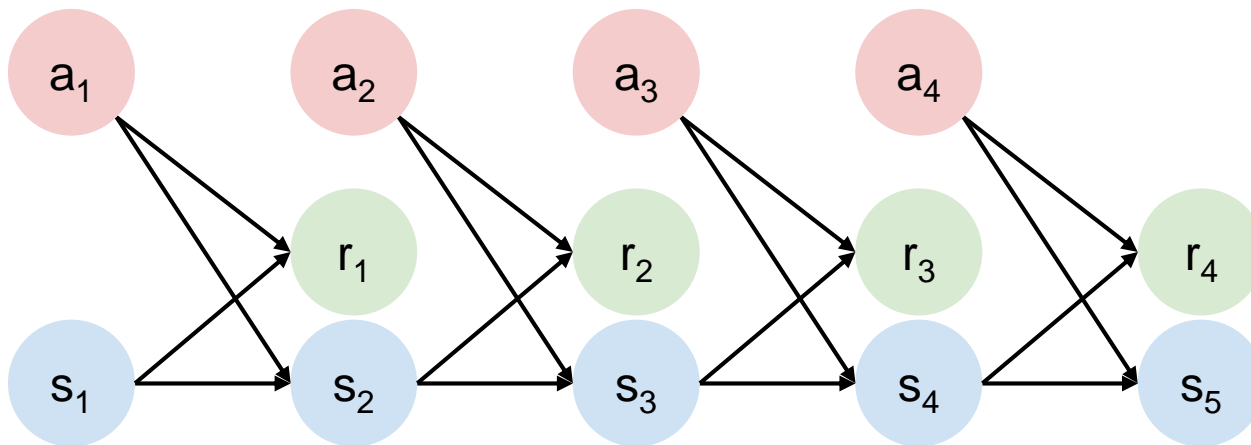
**Model-free:** learn policy directly from data

$$\mathcal{D} \rightarrow \pi \quad \text{e.g. } Q\text{-learning, policy gradient}$$

**Model-based:** learn model, then use it to learn or improve a policy

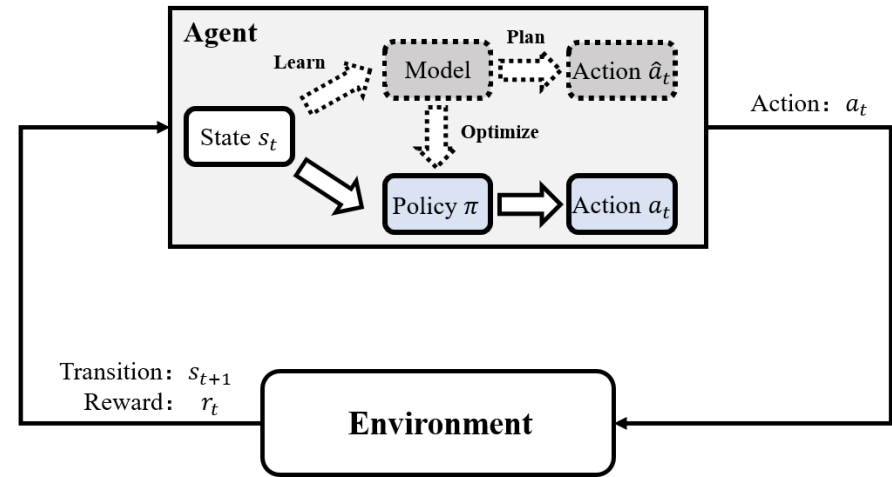
$$\mathcal{D} \rightarrow f \rightarrow \pi$$

□ **What is a Model ?**



# Model-based RL

## □ Model-based RL Process:



Run base policy  $\pi_0(a_t|s_t)$  to collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

**For \_ do**

Learn transition/dynamics model  $f(s, a)$

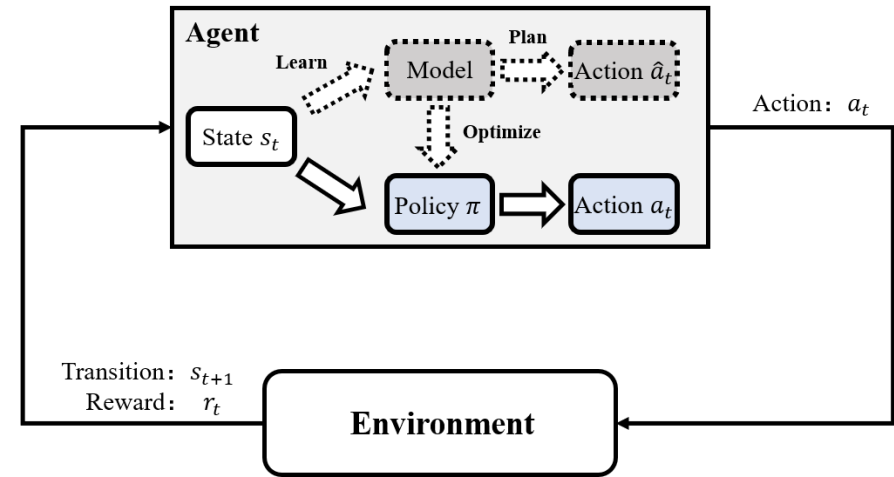
Use model  $f(s, a)$  to assist agent to update policy  $\pi(a_t|s_t)$

Execute  $\pi(a_t|s_t)$  and add the resulting data  $\{(s_t, a_t, s_{t+1})_j\}$  to  $\mathcal{D}$

**End For**

# Model-based RL

## □ Model-based RL Process:



Run base policy  $\pi_0(a_t|s_t)$  to collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

**For \_ do**

Learn transition/dynamics model  $f(s, a)$

**Model Building**

Use model  $f(s, a)$  to assist agent to update policy  $\pi(a_t|s_t)$

**Model Utilizing**

Execute  $\pi(a_t|s_t)$  and add the resulting data  $\{(s_t, a_t, s_{t+1})_j\}$  to  $\mathcal{D}$

**End For**

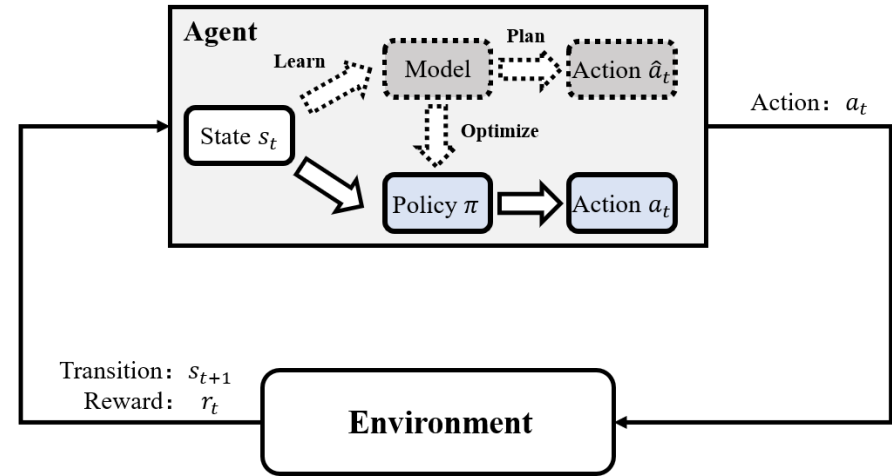
# Model-based RL

## □ Model Building

Collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$



**Deterministic Model or Probabilistic Model**



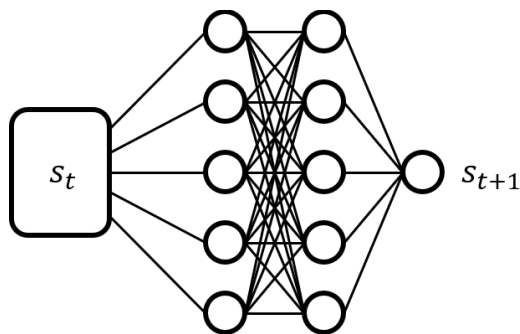
# Model-based RL

## □ Model Building

Collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$



**Deterministic Model** or Probabilistic Model



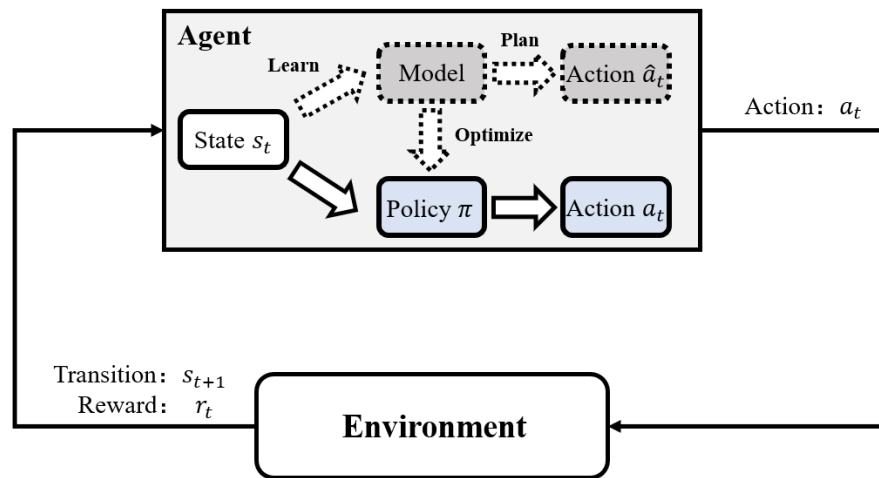
Learn transition model  $f(s, a)$  to minimize **Mean Squared Error (MSE)**:

$$\mathcal{L}_{MSE} = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} [f(s_t, a_t) - s_{t+1}]^2.$$

$$\hat{s}_{t+1} \sim f(s_t, a_t)$$

**Incremental Predict:**

$$\hat{s}_{t+1} \sim s_t + f(s_t, a_t)$$



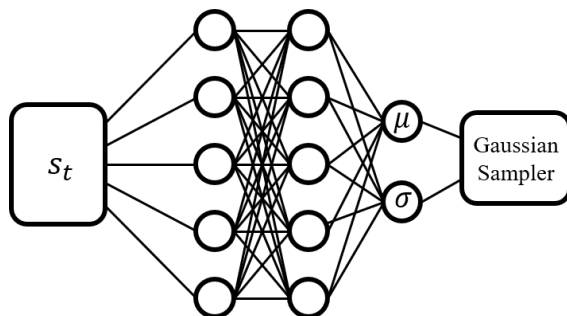
# Model-based RL

## Model Building

Collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$



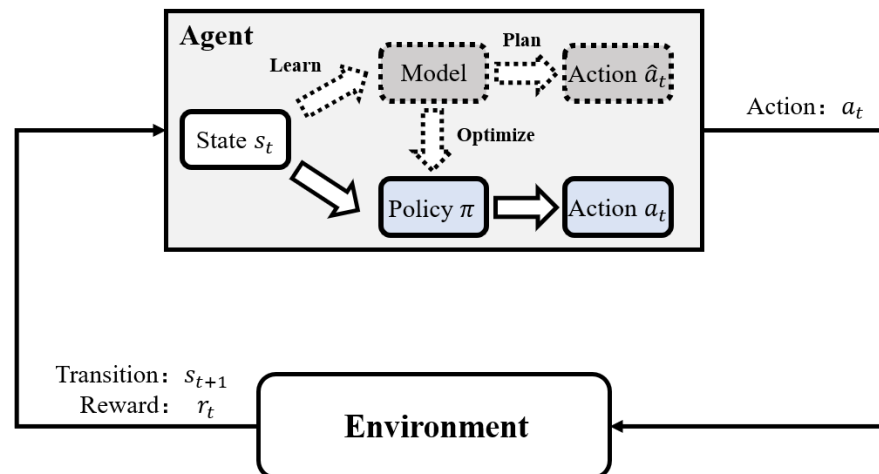
Deterministic Model or **Probabilistic Model**



Learn transition model  $f(s, a)$  to optimize **Negative Log Likelihood (NLL)**

$$\mathcal{L}_{NLL} = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \left( \frac{[\mu(s_t, a_t) - s_{t+1}]^2}{\sigma(s_t, a_t)} + \log \sigma(s_t, a_t) \right)$$

$$\hat{s}_{t+1} \sim \mathcal{N}(\mu(s_t, a_t), \sigma(s_t, a_t))$$



# Model-based RL

## □ Model Building:

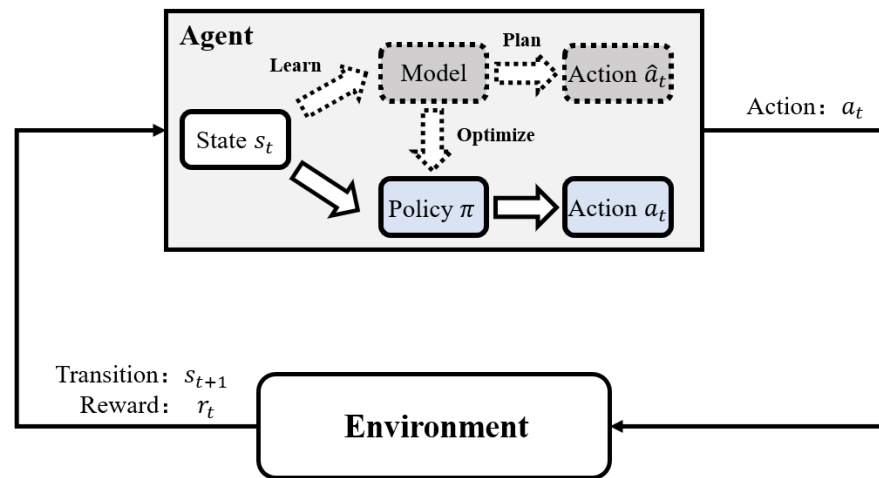
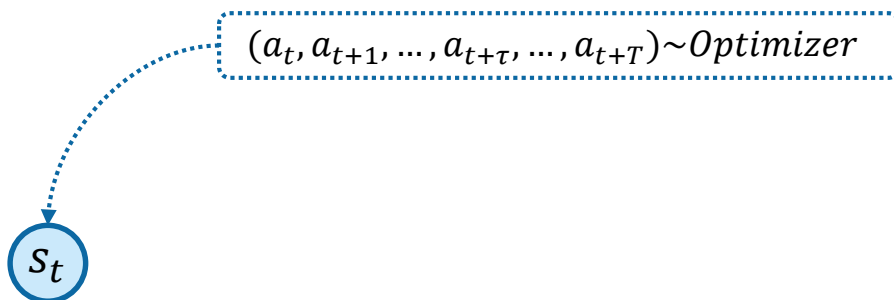
Collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

Learn transition model  $f(s, a)$  to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$

## □ Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function  $\pi$





# Model-based RL

## □ Model Building:

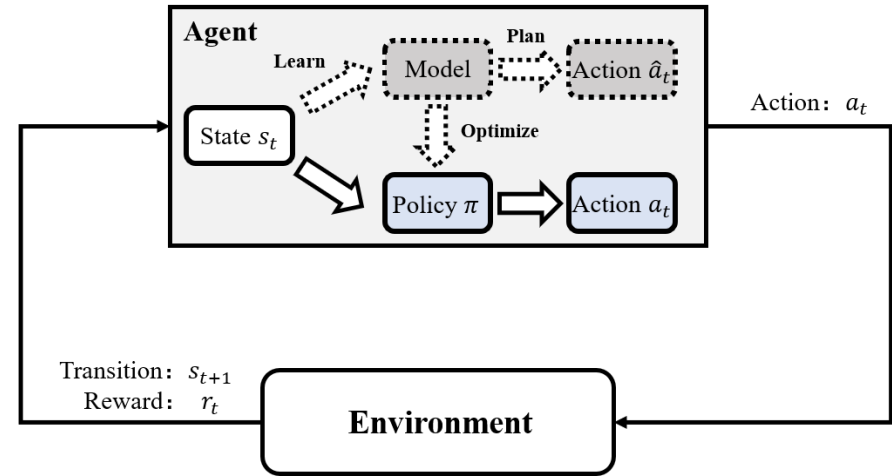
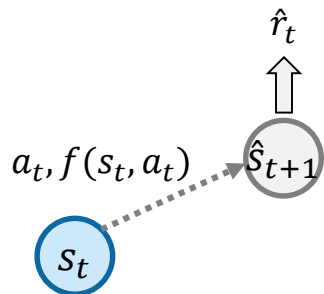
Collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

Learn transition model  $f(s, a)$  to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$

## □ Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function  $\pi$



# Model-based RL

## □ Model Building:

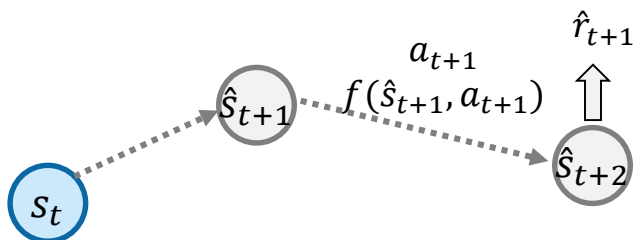
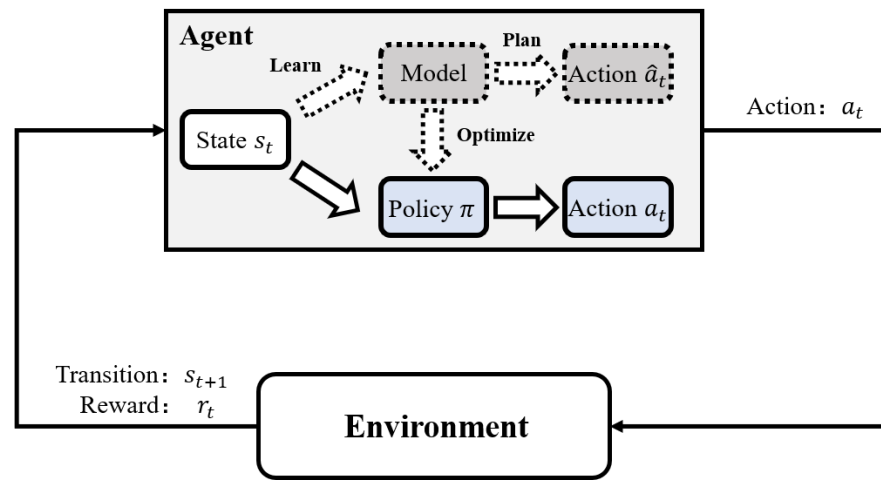
Collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

Learn transition model  $f(s, a)$  to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$

## □ Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function  $\pi$



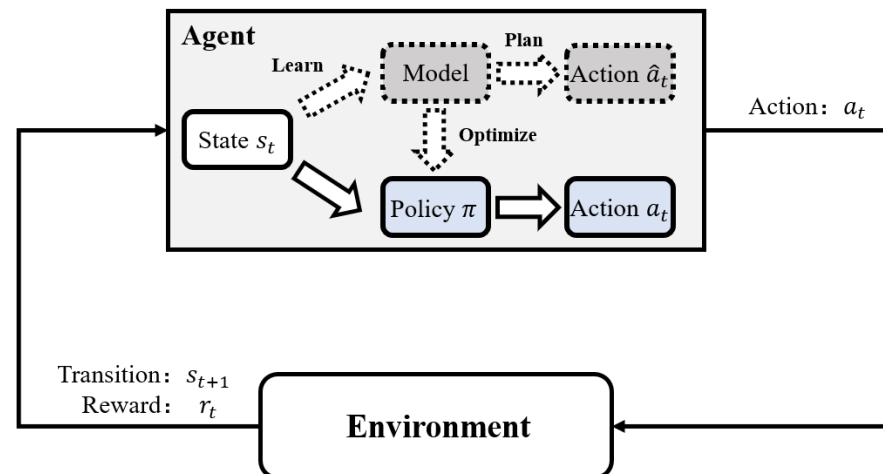
# Model-based RL

## □ Model Building:

Collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

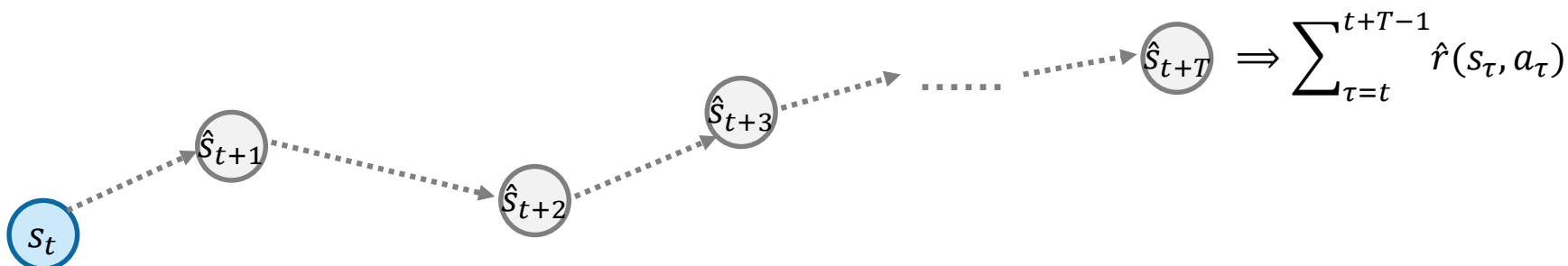
Learn transition model  $f(s, a)$  to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



## □ Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function  $\pi$



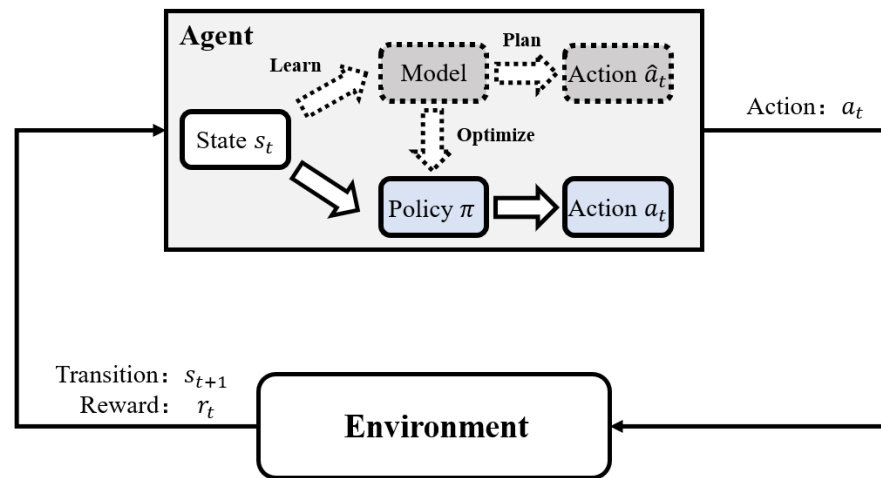
# Model-based RL

## Model Building:

Collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

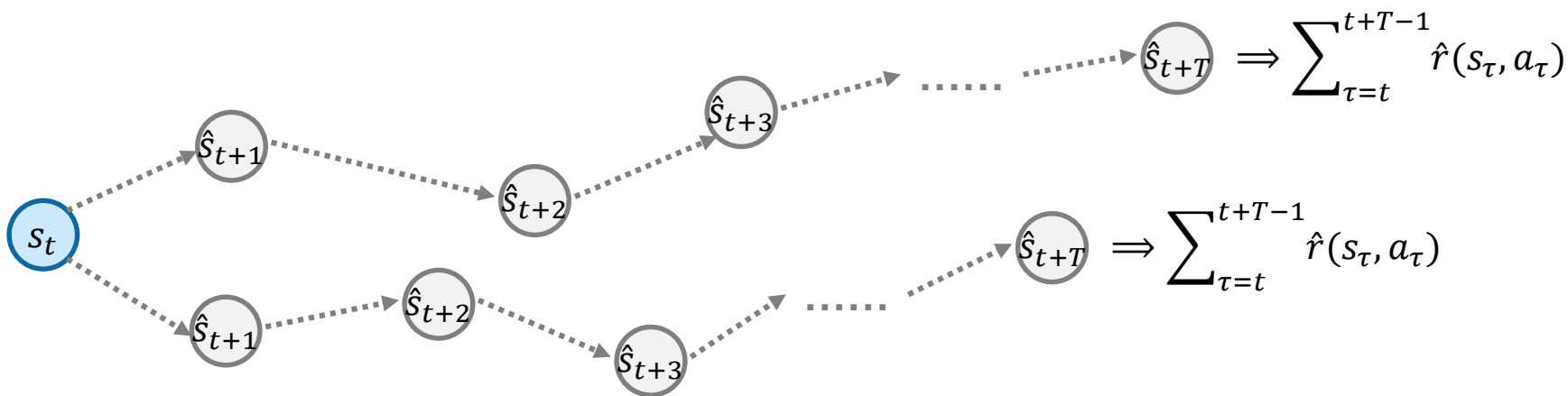
Learn transition model  $f(s, a)$  to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



## Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function  $\pi$



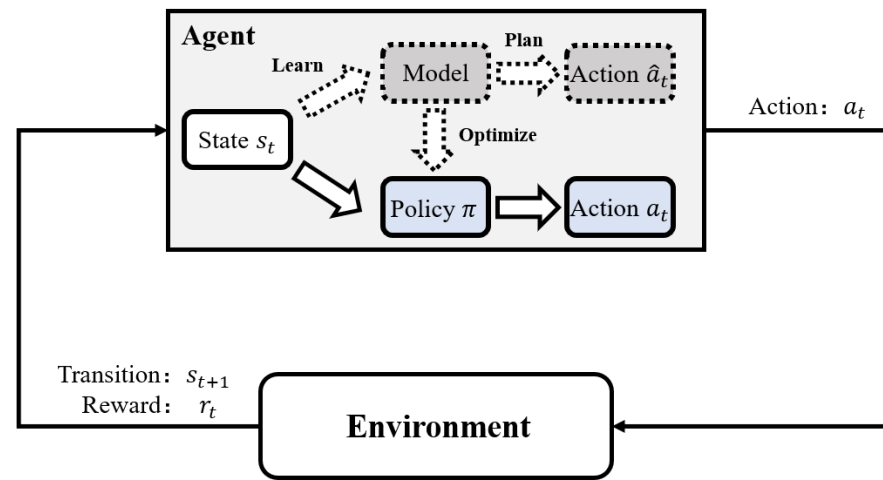
# Model-based RL

## Model Building:

Collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

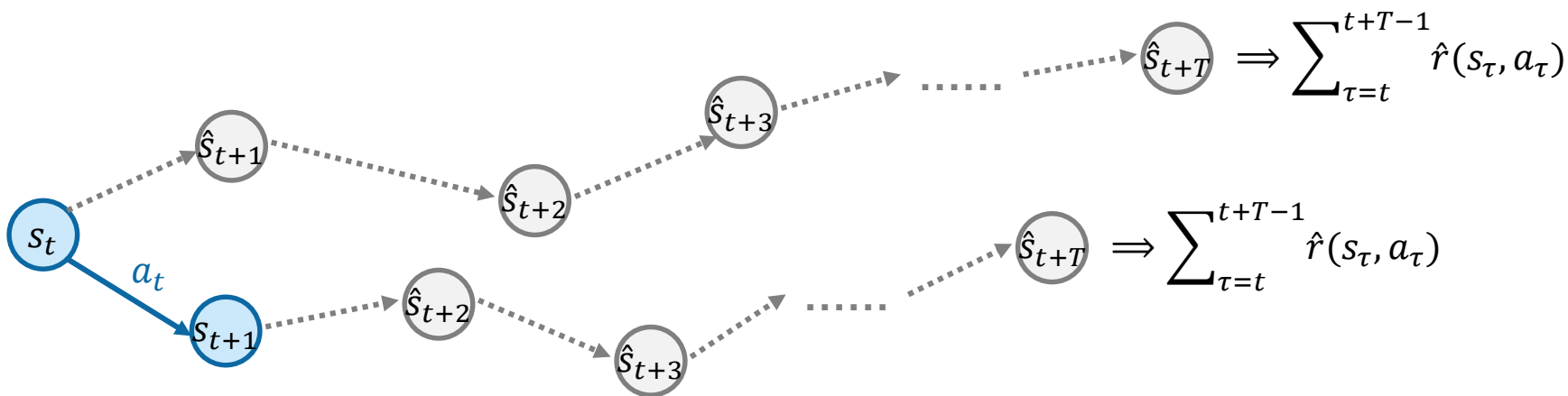
Learn transition model  $f(s, a)$  to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



## Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function  $\pi$



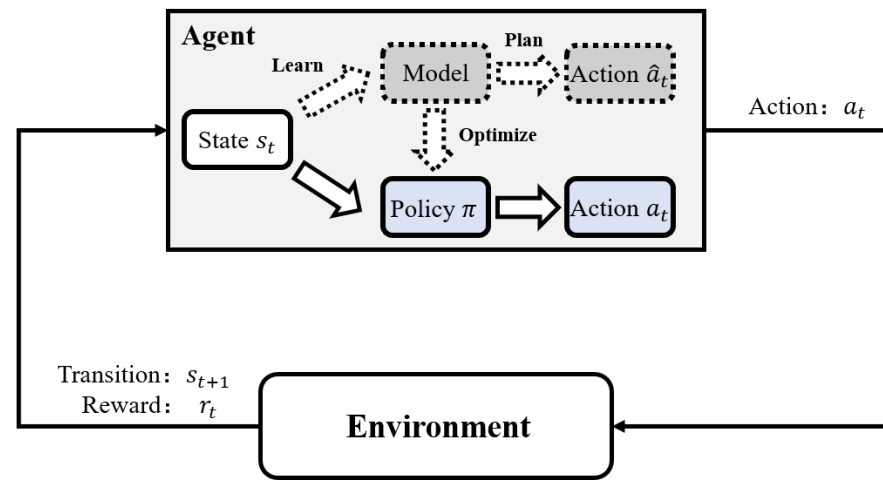
# Model-based RL

## Model Building:

Collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

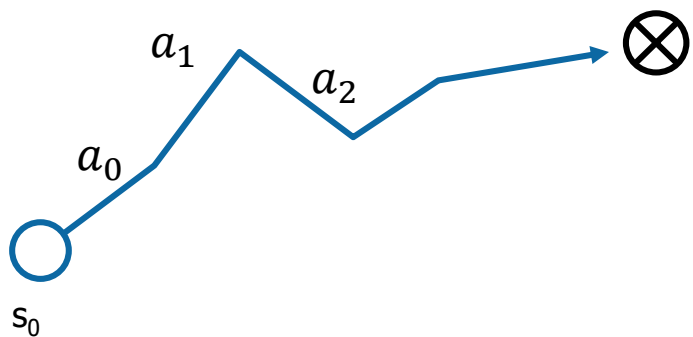
Learn transition model  $f(s, a)$  to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



## Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function  $\pi$



## Trajectory Optimization with Action Sequence:

$$s_t \rightarrow a_t \rightarrow s_{t+1} \rightarrow a_{t+1} \rightarrow \dots$$

$$J(a_0, \dots, a_H) = \sum_{t=0}^H \gamma^t r_t$$

# Model-based RL

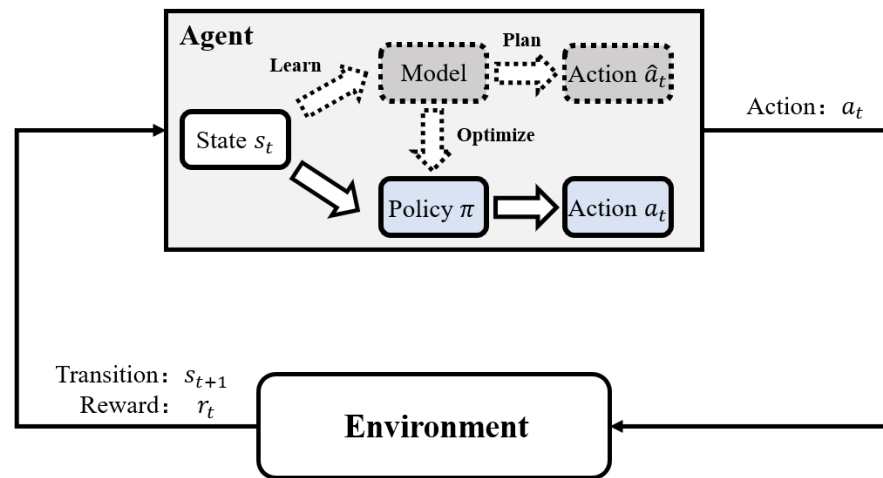
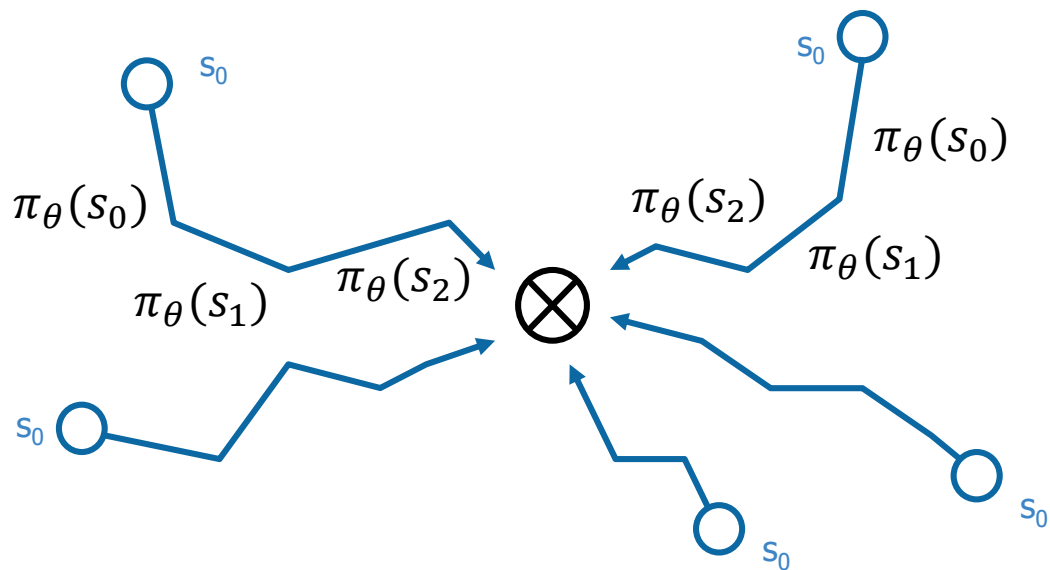
## Model Building:

Collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

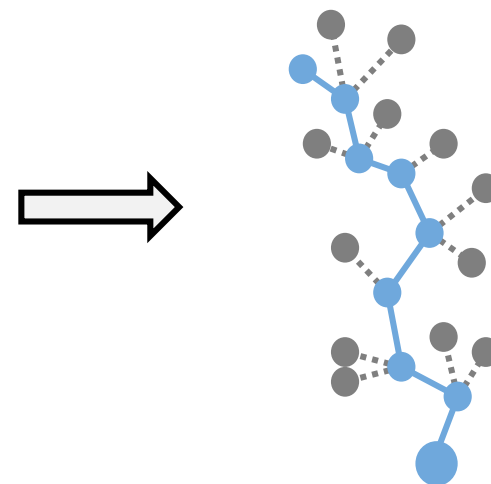
Learn transition model  $f(s, a)$  to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$

## Background Method (Optimize):



Mix **real** and **model-generated** experience and apply additional policy updates.



# Model-based RL

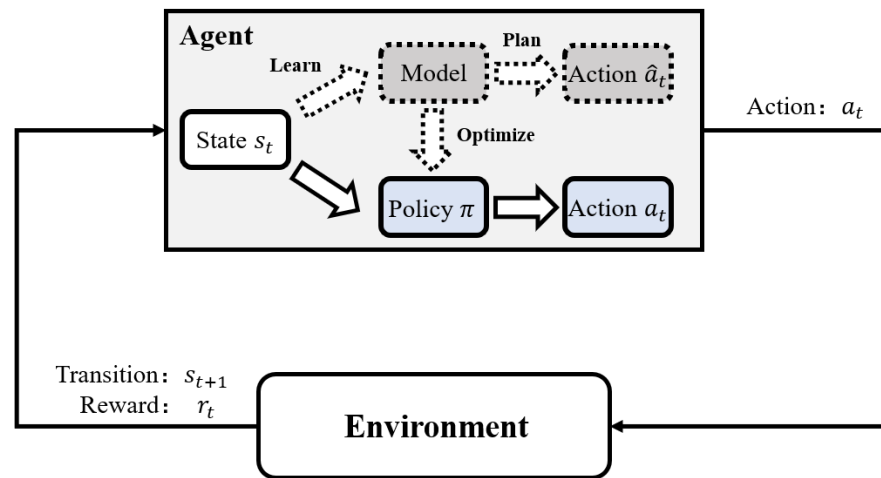
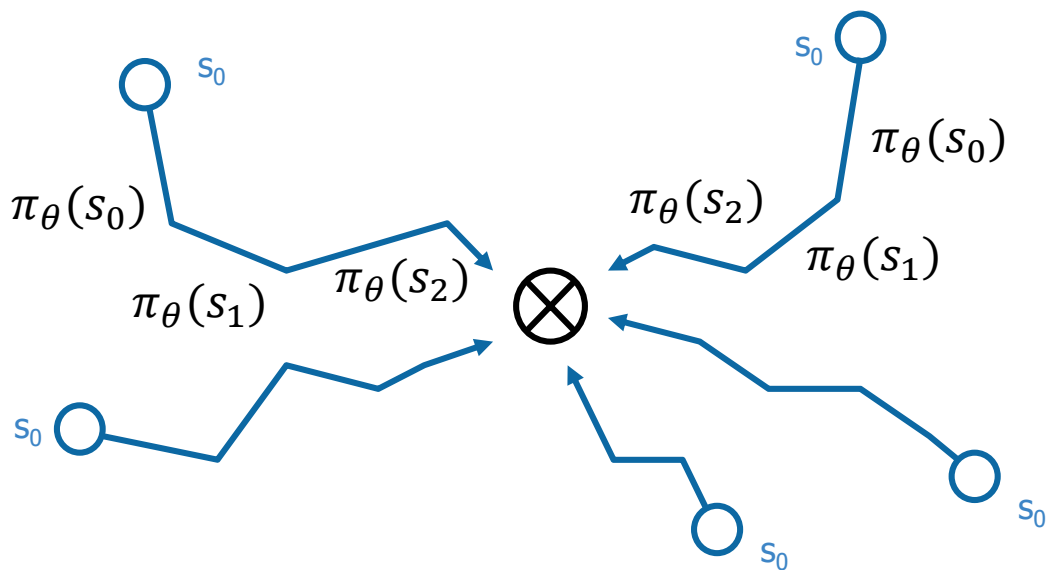
## Model Building:

Collect data  $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

Learn transition model  $f(s, a)$  to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$

## Background Method (Optimize):



## Policy Optimization with parameter $\theta$ :

$$s_t \rightarrow \pi_\theta(s_t) \rightarrow s_{t+1} \rightarrow \pi_\theta(s_{t+1}) \rightarrow \dots$$

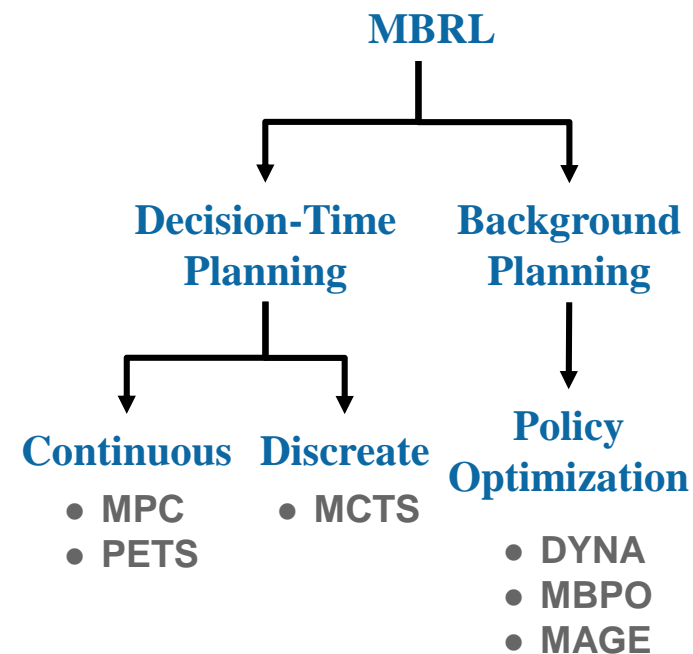
$$J(\theta) = \mathbb{E}_{s_0} \left[ \sum_{t=0}^H \gamma^t r_t \right], \quad a_t = \pi_\theta(s_t)$$



# Model-based RL

## □ Background vs. Decision-Time

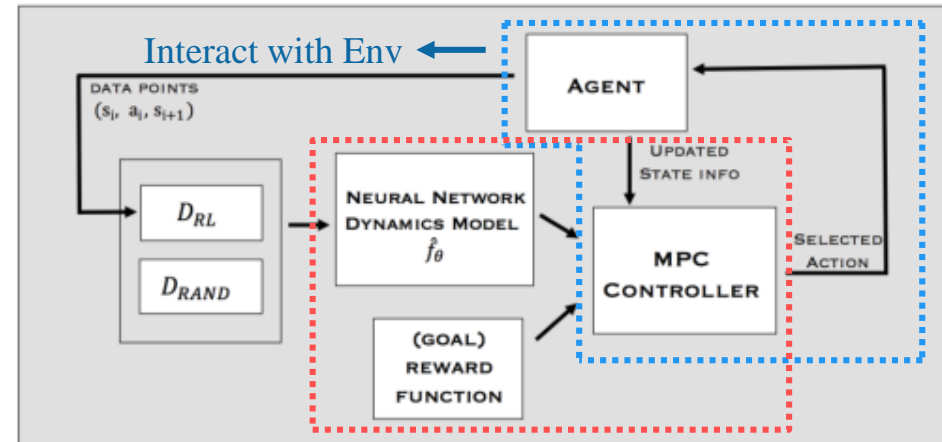
|                          | Background | Decision-Time |
|--------------------------|------------|---------------|
| Act without Learning     | -          | +             |
| Unfamiliar situations    | -          | +             |
| Computation              | +          | -             |
| Predictability Coherence | +          | -             |
| Discrete and Continuous  | +          | -             |



# Decision-Time

## □ MPC with Neural Network Model

- $\mathcal{D}_{RL} = \{(s_t, a_t, s_{t+1})_i\}$  to train neural network dynamic model.
- $\mathcal{D}_{RAND} = \{(a_1, a_2, \dots, a_T)\}$  to store the sampled actions sequence.
- (Goal) Reward Function



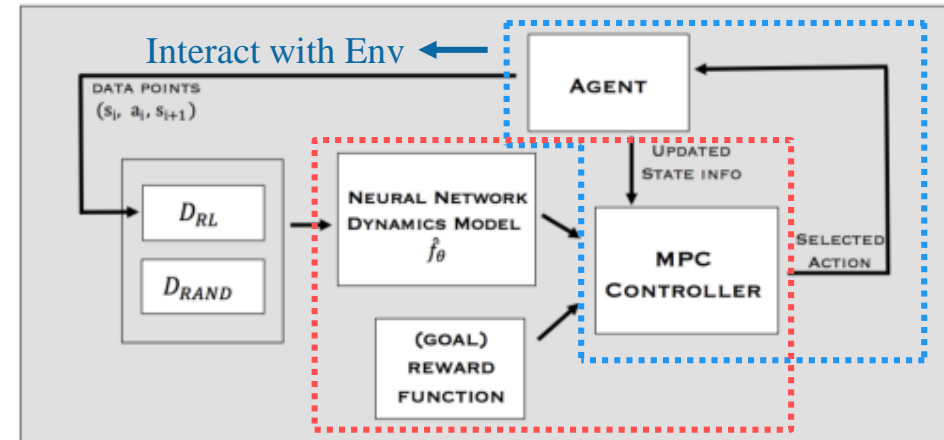
1. Planning based on Model
2. Agent action selection

# Decision-Time

## □ MPC with Neural Network Model

### Algorithm 1 Model-based Reinforcement Learning

- 1: gather dataset  $\mathcal{D}_{\text{RAND}}$  of random trajectories
- 2: initialize empty dataset  $\mathcal{D}_{\text{RL}}$ , and randomly initialize  $\hat{f}_{\theta}$
- 3: **for** iter=1 **to** max\_iter **do**
- 4:   train  $\hat{f}_{\theta}(s, a)$  by performing gradient descent on Eqn. 2 using  $\mathcal{D}_{\text{RAND}}$  and  $\mathcal{D}_{\text{RL}}$
- 5:   **for**  $t = 1$  **to**  $T$  **do**
- 6:     get agent's current state  $s_t$
- 7:     use  $\hat{f}_{\theta}$  to estimate optimal action sequence  $\mathbf{A}_t^{(H)}$  (Eqn. 4)
- 8:     execute first action  $a_t$  from selected action sequence  $\mathbf{A}_t^{(H)}$
- 9:     add  $(s_t, a_t)$  to  $\mathcal{D}_{\text{RL}}$
- 10:   **end for**
- 11: **end for**



1. Planning based on Model
2. Agent action selection

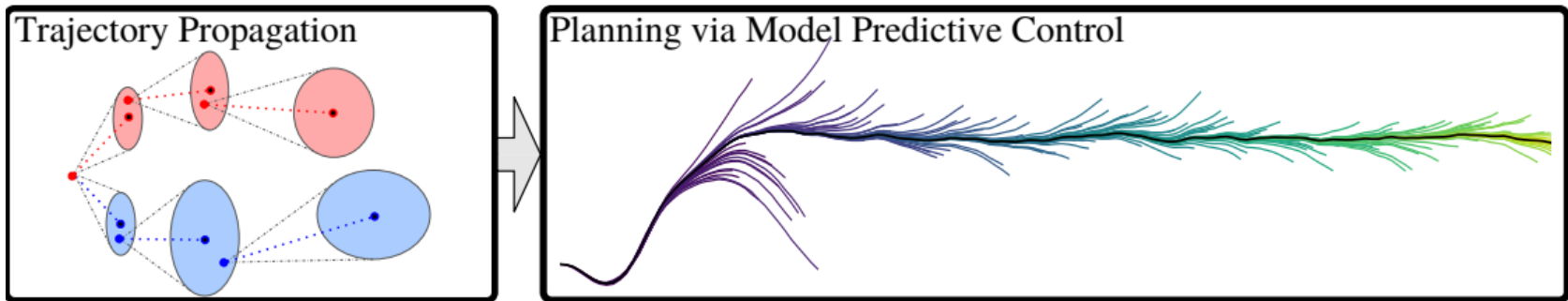
### Random Shooting (RS):

For  $t = 1$  to  $T$  do:

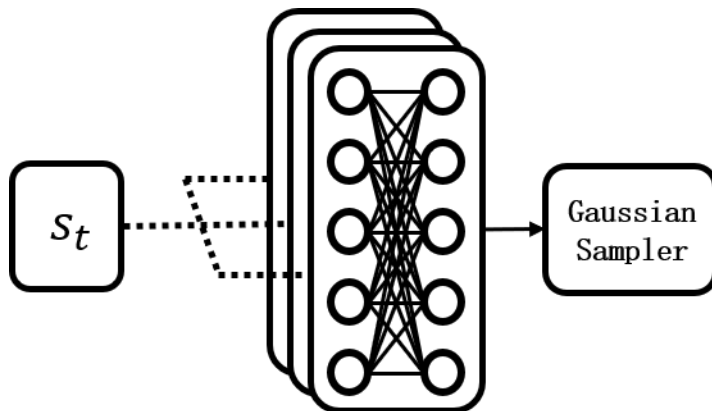
$$\text{random}(a_t) \in A \rightarrow \mathcal{D}_{\text{RAND}}$$

# Decision-Time

## Probabilistic Ensemble with Trajectory Sampling (PETS)



- Probabilistic Model Networks to capture **aleatoric uncertainty**.
- Model Ensembles to capture **epistemic uncertainty**.

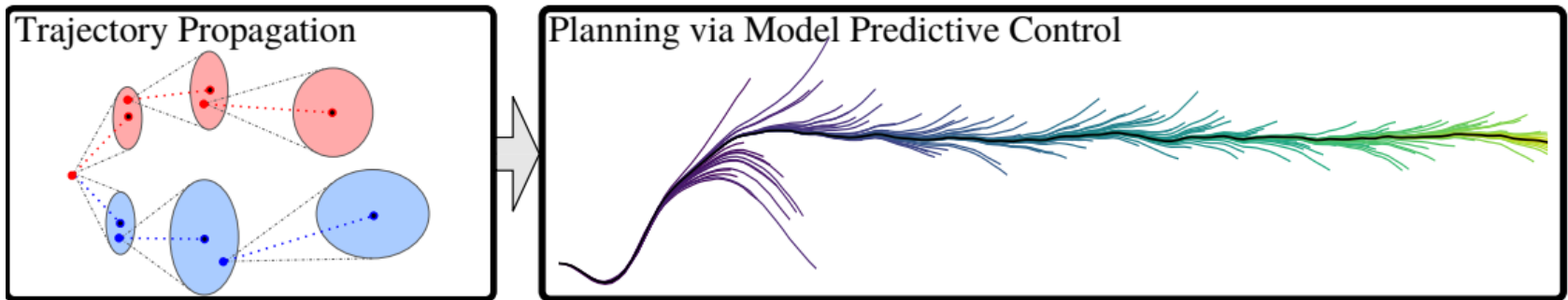


**Model Ensemble Output:**

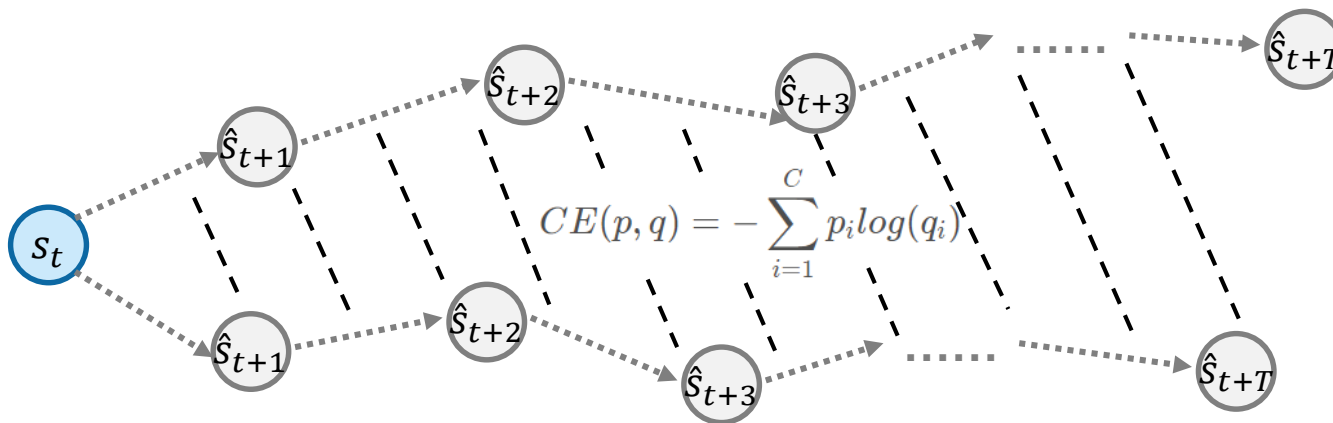
$$\hat{s}_{t+1} = \text{random} (f_1(s_t, a_t), f_2(s_t, a_t), \dots, f_M(s_t, a_t))$$

$$\hat{s}_{t+1} = \frac{1}{M} \sum_{j=1}^M f_j(s_t, a_t).$$

## □ Probabilistic Ensemble with Trajectory Sampling (PETS)

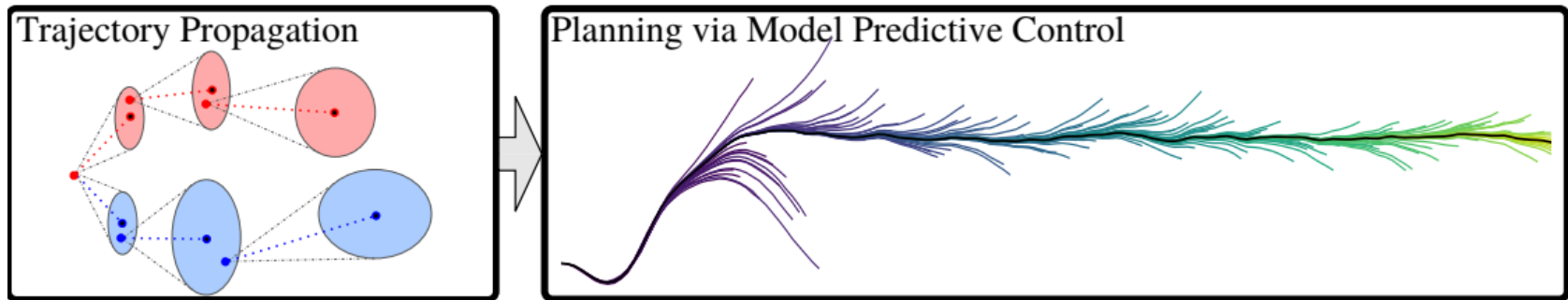


## □ Cross Entropy Method (CEM) instead of Random Shooting (RS)



# Decision-Time

## Probabilistic Ensemble with Trajectory Sampling (PETS)




---

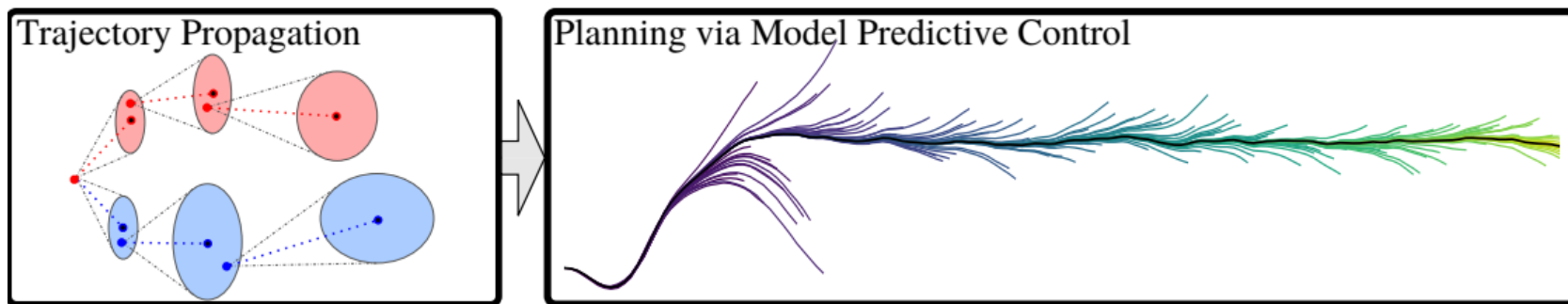
### Algorithm 1 Our model-based MPC algorithm ‘PETS’:

---

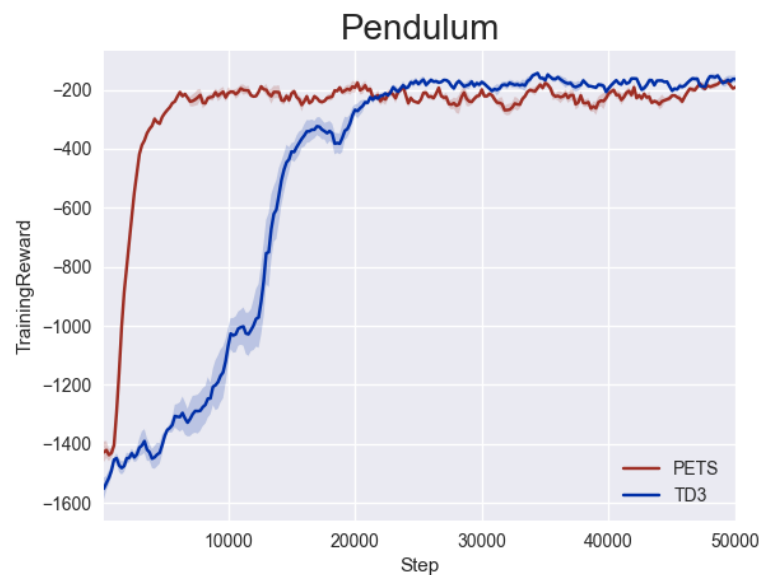
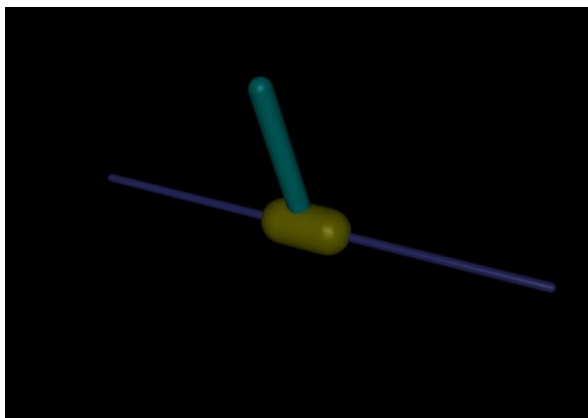
- 1: Initialize data  $\mathbb{D}$  with a random controller for one trial.
  - 2: **for** Trial  $k = 1$  to  $K$  **do**
  - 3:   Train a *PE* dynamics model  $\tilde{f}$  given  $\mathbb{D}$ .
  - 4:   **for** Time  $t = 0$  to TaskHorizon **do**
  - 5:     **for** Actions sampled  $\mathbf{a}_{t:t+T} \sim \text{CEM}(\cdot)$ , 1 to  $\tilde{N}\text{Samples}$  **do**
  - 6:       Propagate state particles  $\mathbf{s}_\tau^p$  using *TS* and  $\tilde{f}|\{\mathbb{D}, \mathbf{a}_{t:t+T}\}$ .
  - 7:       Evaluate actions as  $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^P r(\mathbf{s}_\tau^p, \mathbf{a}_\tau)$
  - 8:       Update  $\text{CEM}(\cdot)$  distribution.
  - 9:     Execute first action  $\mathbf{a}_t^*$  (only) from optimal actions  $\mathbf{a}_{t:t+T}^*$ .
  - 10:    Record outcome:  $\mathbb{D} \leftarrow \mathbb{D} \cup \{\mathbf{s}_t, \mathbf{a}_t^*, \mathbf{s}_{t+1}\}$ .
-

# Decision-Time

## Probabilistic Ensemble with Trajectory Sampling (PETS)



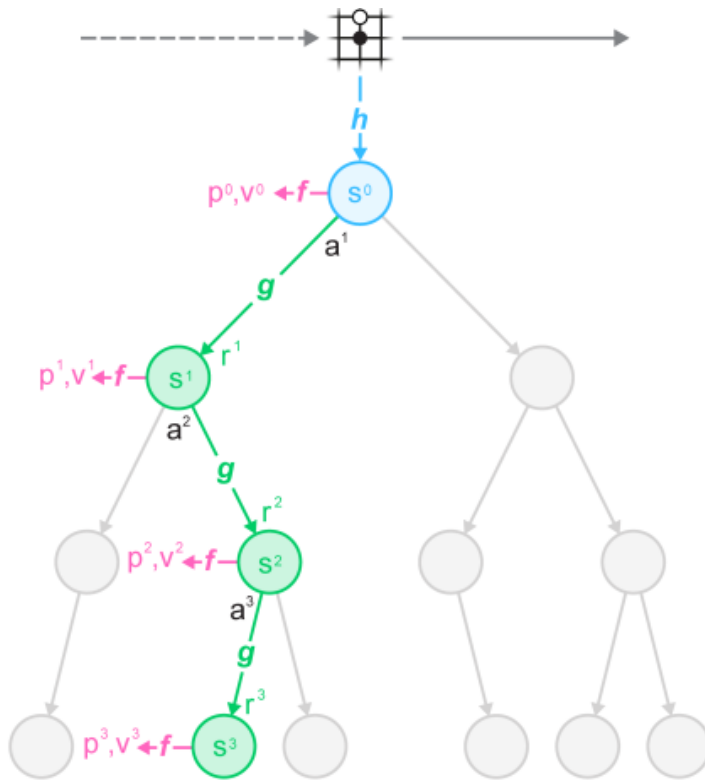
## Experiments



# Decision-Time

## □ MuZero: Mastering Atari, Go and Chess

### □ Monte-Carlo Tree Search (MCTS) with Dynamics Model



□ Representation Function:  $s^0 = h_{\theta}(o_1, \dots, o_n)$  to encode past observations.

□ Dynamic Function:  $r^k, s^k = g_{\theta}(s^{k-1}, a^k)$

□ Prediction Function:  $p^k, v^k = f_{\theta}(s^k)$

### MCTS Backup:

$$Q_{k+1}(s, a) = \frac{Q_k(s, a) \cdot N_k(s, a) + R}{N_k(s, a) + 1}$$

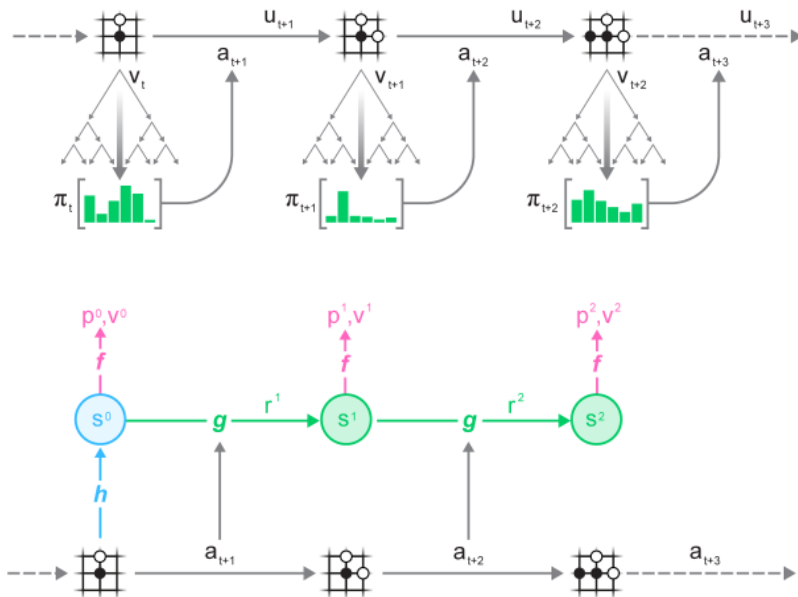
$$N_{k+1}(s, a) = N_k(s, a) + 1$$



# Decision-Time

## □ MuZero: Mastering Atari, Go and Chess

### □ Monte-Carlo Tree Search (MCTS) with Dynamics Model



□ Representation Function:  $s^0 = h_{\theta}(o_1, \dots, o_n)$  to encode past observations.

□ Dynamic Function:  $r^k, s^k = g_{\theta}(s^{k-1}, a^k)$

□ Prediction Function:  $p^k, v^k = f_{\theta}(s^k)$

### MCTS Backup:

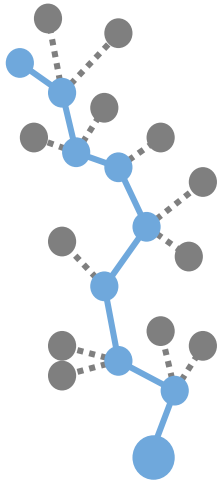
$$Q_{k+1}(s, a) = \frac{Q_k(s, a) \cdot N_k(s, a) + R}{N_k(s, a) + 1}$$

$$N_{k+1}(s, a) = N_k(s, a) + 1$$

# Background

## □ Dyna-Q

- Mix **real** and **model-generated** experiences and apply for additional policy updates.



### Tabular Dyna-Q

Initialize  $Q(s, a)$  and  $Model(s, a)$  for all  $s \in \mathcal{S}$  and  $a \in \mathcal{A}(s)$

Loop forever:

- $S \leftarrow$  current (nonterminal) state
- $A \leftarrow \epsilon$ -greedy( $S, Q$ )
- Take action  $A$ ; observe resultant reward,  $R$ , and state,  $S'$
- $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
- $Model(S, A) \leftarrow R, S'$  (assuming deterministic environment)
- Loop repeat  $n$  times:

Update Q values with  
real transitions

$S \leftarrow$  random previously observed state  
 $A \leftarrow$  random action previously taken in  $S$   
 $R, S' \leftarrow Model(S, A)$

Update Q values with  
model transitions

$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

## □ Model-Based Policy Optimization (MBPO)

- Dyna-Q updates the policy with both real and model-generated transitions.
- How to guarantee the policy improvement if we only use the model-generated transitions ?

**Definition:** Errors in the model can be exploited during policy optimization, resulting in large discrepancies between the predicted returns of the policy under the model and under the true dynamics.

$$\eta[\pi] \geq \hat{\eta}[\pi] - C.$$

$\eta[\pi]$  : the returns of the policy in true MDP;  $\hat{\eta}[\pi]$  : the returns under learned model.

## □ Model-Based Policy Optimization (MBPO)

- Dyna-Q updates the policy with both real and model-generated transitions.
- How to guarantee the policy improvement if we only use the model-generated transitions ?

**Definition:** Errors in the model can be exploited during policy optimization, resulting in large discrepancies between the predicted returns of the policy under the model and under the true dynamics.

$$\eta[\pi] \geq \hat{\eta}[\pi] - C.$$

The gap between true and model returns can be expressed in:

□ **Generalization Error:**  $\epsilon_m = \mathbb{E}_{s,a \sim D_{env}} [D_{TV}(p(s'|s, a) || p_{\theta}(s'|s, a))]$

□ **Policy Distribution:**  $\epsilon_{\pi} = D_{TV}(\pi_D(s) || \pi(s))$

# Background

## □ Model-Based Policy Optimization (MBPO)

- Dyna-Q updates the policy with both real and model-generated transitions.
- How to guarantee the policy improvement if we only use the model-generated transitions ?

The gap between:

$$\begin{aligned} |\eta_1 - \eta_2| &= \left| \sum_{s,a} (p_1(s,a) - p_2(s,a)) r(s,a) \right| \\ &= \left| \sum_{s,a} \left( \sum_t \gamma^t p_1^t(s,a) - p_2^t(s,a) \right) r(s,a) \right| \\ &= \left| \sum_t \sum_{s,a} \gamma^t (p_1^t(s,a) - p_2^t(s,a)) r(s,a) \right| \\ &\leq \sum_t \sum_{s,a} \gamma^t |p_1^t(s,a) - p_2^t(s,a)| r(s,a) \\ &\leq r_{\max} \sum_t \sum_{s,a} \gamma^t |p_1^t(s,a) - p_2^t(s,a)| \end{aligned}$$

# Background

## □ Model-Based Policy Optimization (MBPO)

- Dyna-Q updates the policy with both real and model-generated transitions.
- How to guarantee the policy improvement if we only use the model-generated transitions ?

The gap between:

$$|\eta_1 - \eta_2| \leq r_{\max} \sum_t \sum_{s,a} \gamma^t |p_1^t(s, a) - p_2^t(s, a)|$$

**Lemma:**

**Lemma B.1** (TVD of Joint Distributions). Suppose we have two distributions  $p_1(x, y) = p_1(x)p_1(y|x)$  and  $p_2(x, y) = p_2(x)p_2(y|x)$ . We can bound the total variation distance of the joint as:

$$D_{TV}(p_1(x, y) || p_2(x, y)) \leq D_{TV}(p_1(x) || p_2(x)) + E_{x \sim p_1}[D_{TV}(p_1(y|x) || p_2(y|x))]$$

# Background

## □ Model-Based Policy Optimization (MBPO)

- Dyna-Q updates the policy with both real and model-generated transitions.
- How to guarantee the policy improvement if we only use the model-generated transitions ?

The gap between:

$$|\eta_1 - \eta_2| \leq r_{\max} \sum_t \sum_{s,a} \gamma^t |p_1^t(s,a) - p_2^t(s,a)|$$

Lemma:

**Lemma B.2** (Markov chain TVD bound, time-varying). Suppose the expected KL-divergence between two transition distributions is bounded as  $\max_t E_{s \sim p_1^t(s)} D_{KL}(p_1(s'|s) || p_2(s'|s)) \leq \delta$ , and the initial state distributions are the same –  $p_1^{t=0}(s) = p_2^{t=0}(s)$ . Then the distance in the state marginal is bounded as:

$$D_{TV}(p_1^t(s) || p_2^t(s)) \leq t\delta$$

$$\delta = \epsilon_m + \epsilon_\pi$$

# Background

## □ Model-Based Policy Optimization (MBPO)

- Dyna-Q updates the policy with both real and model-generated transitions.
- How to guarantee the policy improvement if we only use the model-generated transitions ?

The gap between:

$$\begin{aligned}
 |\eta_1 - \eta_2| &\leq r_{\max} \sum_t \sum_{s,a} \gamma^t |p_1^t(s,a) - p_2^t(s,a)| \quad \begin{cases} \dashrightarrow D_{TV}(p_1^t(s)||p_2^t(s)) \leq t(\epsilon_m + \epsilon_\pi) \\ \dashrightarrow D_{TV}(\pi_1(a|s)||\pi_2(a|s)) \leq \epsilon_\pi \end{cases} \\
 &\leq 2r_{\max} \sum_t \gamma^t t(\epsilon_m + \epsilon_\pi) + \epsilon_\pi \\
 &\leq 2r_{\max} \left( \frac{\gamma(\epsilon_\pi + \epsilon_m)}{(1-\gamma)^2} + \frac{\epsilon_\pi}{1-\gamma} \right)
 \end{aligned}$$

The returns and model returns of the policy are bounded as:

$$\eta[\pi] \geq \hat{\eta}[\pi] - \underbrace{\left[ \frac{2\gamma r_{\max}(\epsilon_m + 2\epsilon_\pi)}{(1-\gamma)^2} + \frac{4r_{\max}\epsilon_\pi}{(1-\gamma)} \right]}_{C(\epsilon_m, \epsilon_\pi)}$$



## □ Model-Based Policy Optimization (MBPO)

- Dyna-Q updates the policy with both real and model-generated transitions.
- How to guarantee the policy improvement if we only use the model-generated transitions ?
- H-steps Q-target objective:

**Model-based Critic Loss Function:**

$$Loss_{critic} = \frac{1}{2} (y - Q(s_t, a_t))^2$$

**Dyna-Q one-step Q-target:**

$$y = \hat{r}(s_t, a_t) + \gamma Q(\hat{s}_{t+1}, \hat{a}_{t+1})$$

**Multi-steps Q-target:**

$$y = \sum_{k=t}^{H-1} \gamma^{k-t} \hat{r}_k + \gamma^H Q(\hat{s}_H, \hat{a}_H)$$

# Background

## □ Model-Based Policy Optimization (MBPO)

- Dyna-Q updates the policy with both real and model-generated transitions.
- How to guarantee the policy improvement if we only use the model-generated transitions ?
- H-steps Q-target objective
- MBPO Algorithm:

---

### Algorithm 2 Model-Based Policy Optimization with Deep Reinforcement Learning

---

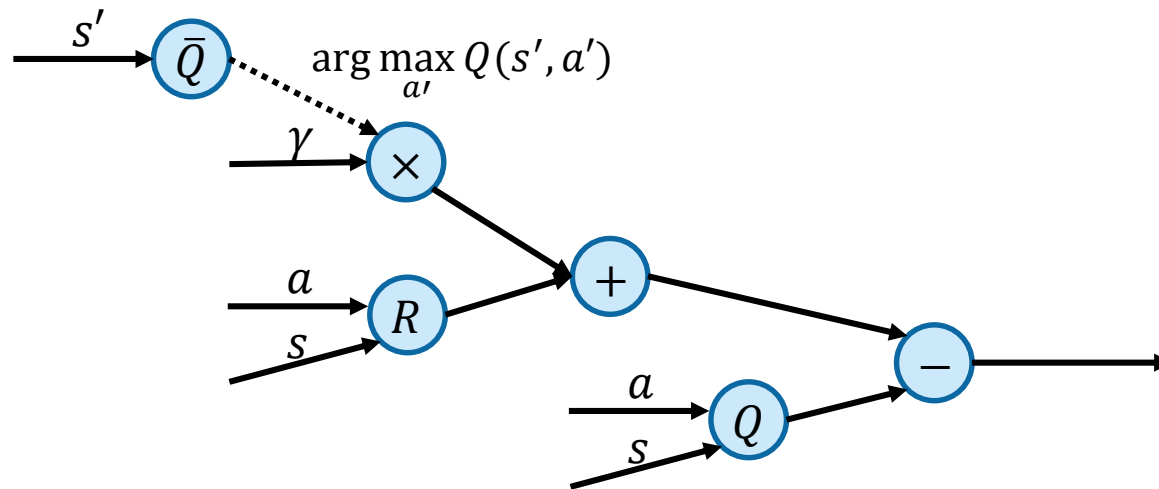
- 1: Initialize policy  $\pi_\phi$ , predictive model  $p_\theta$ , environment dataset  $\mathcal{D}_{\text{env}}$ , model dataset  $\mathcal{D}_{\text{model}}$
  - 2: **for**  $N$  epochs **do**
  - 3:   Train model  $p_\theta$  on  $\mathcal{D}_{\text{env}}$  via maximum likelihood
  - 4:   **for**  $E$  steps **do**
  - 5:     Take action in environment according to  $\pi_\phi$ ; add to  $\mathcal{D}_{\text{env}}$
  - 6:     **for**  $M$  model rollouts **do**
  - 7:       Sample  $s_t$  uniformly from  $\mathcal{D}_{\text{env}}$
  - 8:       Perform  $k$ -step model rollout starting from  $s_t$  using policy  $\pi_\phi$ ; add to  $\mathcal{D}_{\text{model}}$
  - 9:     **for**  $G$  gradient updates **do**
  - 10:       Update policy parameters on model data:  $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi, \mathcal{D}_{\text{model}})$
-

# Background

## □ Model-based Action-Gradient Estimator (MAGE)

- Typical critic network learned by temporal difference optimize for an accurate prediction of the **expected return** of the actor.
- MAGE trains the critic network to provide **action-gradients** for the use in the policy improvement step.

### Typical TD-Computation:

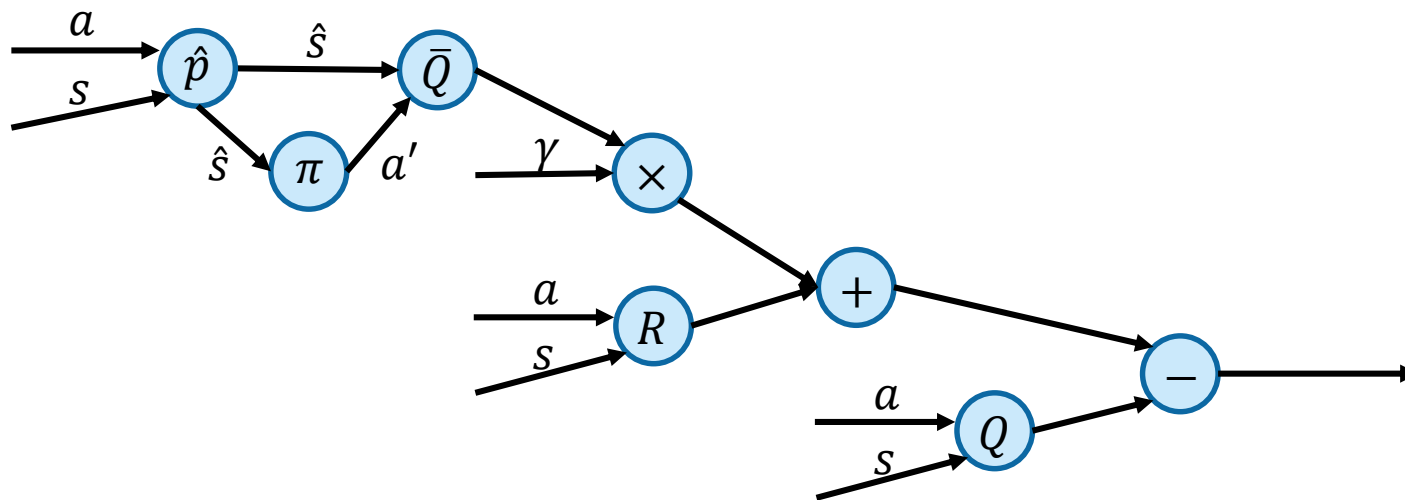


# Background

## □ Model-based Action-Gradient Estimator (MAGE)

- Typical critic network learned by temporal difference optimize for an accurate prediction of the **expected return** of the actor.
- MAGE trains the critic network to provide **action-gradients** for the use in the policy improvement step.

### MAGE TD-Computation:

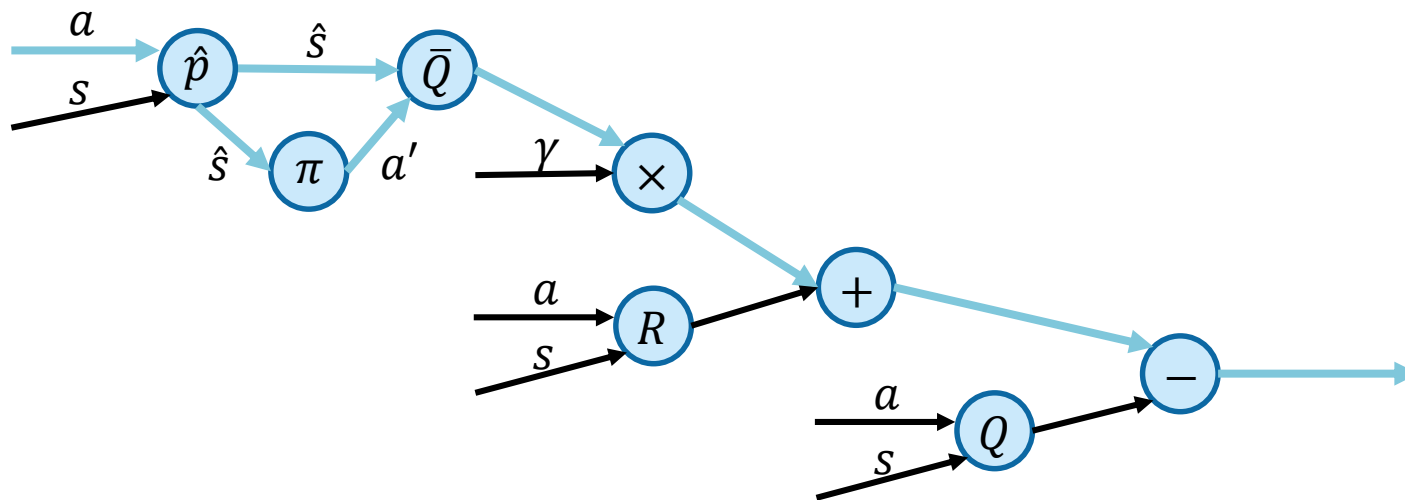


# Background

## □ Model-based Action-Gradient Estimator (MAGE)

- Typical critic network learned by temporal difference optimize for an accurate prediction of the **expected return** of the actor.
- MAGE trains the critic network to provide **action-gradients** for the use in the policy improvement step.

### MAGE TD-Computation:



# Background

## □ Model-based Action-Gradient Estimator (MAGE)

- Typical critic network learned by temporal difference optimize for an accurate prediction of the **expected return** of the actor.
- MAGE trains the critic network to provide **action-gradients** for the use in the policy improvement step.
- MAGE Algorithm:

---

### Algorithm 1 Model-based Action-Gradient-Estimator Policy Optimization (MAGE)

---

**Input:** Initial buffer  $\mathcal{B}$ , set of parameter vectors  $\{\omega, \phi, \theta\}$

**for** each iteration **do**

    Collect transition  $(s, a, s')$  acting according to exploratory version of  $\pi_\theta$

$\mathcal{B} \leftarrow \mathcal{B} \cup \{(s, a, s')\}$

**for** each model learning step **do**

$\omega \leftarrow \omega - \alpha_p \nabla_\omega \ell(s, a, s'; \omega), \quad (s, a, s') \sim \mathcal{B}$

**end for**

**for** each policy optimization step **do**

        Extract state  $s$  after sampling  $(s, a, s') \sim \mathcal{B}$

$\bar{\phi} \leftarrow \phi$

$\hat{\delta}(s, a, \hat{s}; \phi) \leftarrow r(s, \pi_\theta(s)) + \gamma Q_{\bar{\phi}}(\hat{s}, \pi_\theta(\hat{s})) - Q_\phi(s, \pi_\theta(s)), \quad \hat{s} \sim p_\omega(\cdot | s, \pi_\theta(s))$

$\phi \leftarrow \phi - \alpha_Q \nabla_\phi \left( \left\| \nabla_a \hat{\delta}(s, a, \hat{s}; \phi) \right\|_{a=\pi_\theta(s)} + \lambda \left| \hat{\delta}(s, a, \hat{s}; \phi) \right| \right)$

$\theta \leftarrow \theta + \alpha_\pi \nabla_\theta Q_\phi(s, \pi_\theta(s))$

**end for**

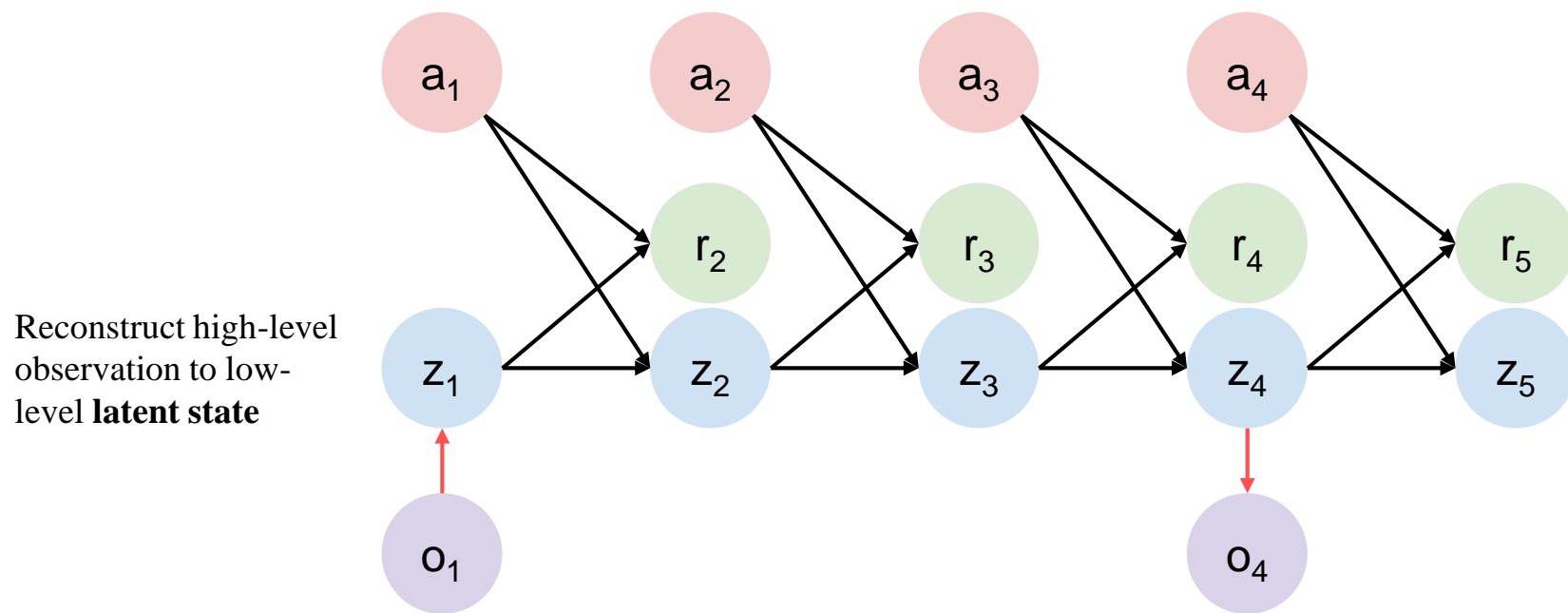
**end for**

---

# Model-based RL with Images

## □ Dreamer: Latent Imagination

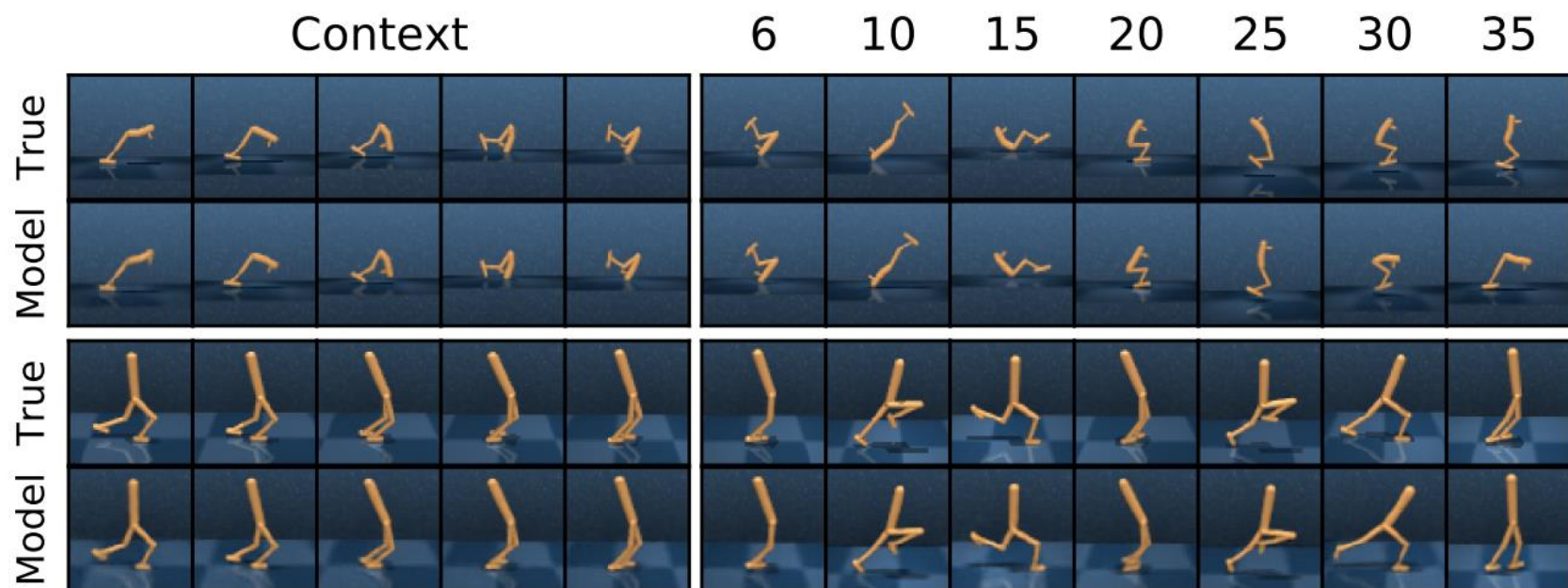
### □ Latent state-transition models



# Model-based RL with Images

## □ Dreamer: Latent Imagination

### □ Latent state-transition models

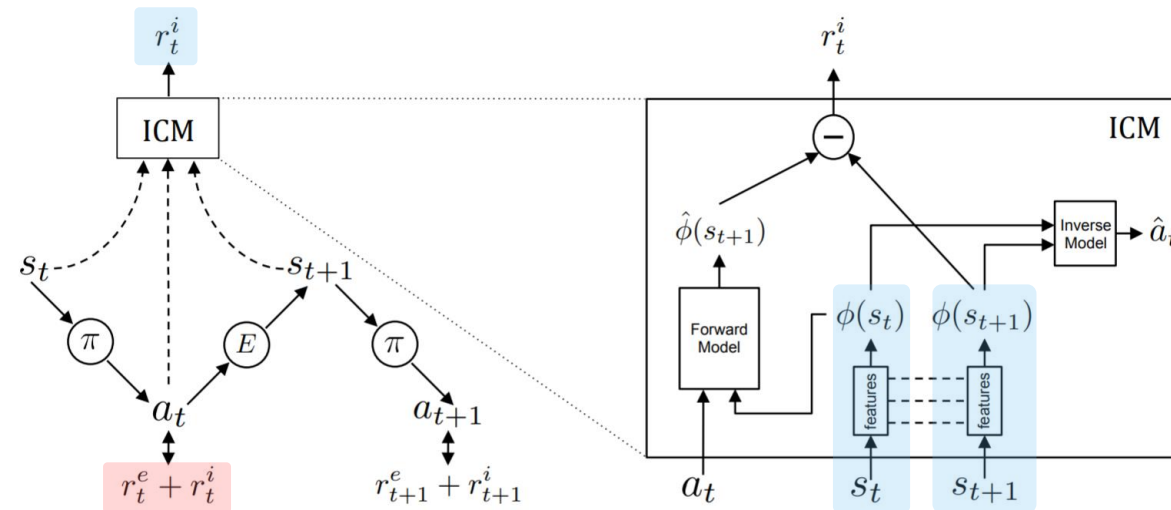




# Inverse Model

## □ Intrinsic Curiosity Module (ICM)

- Besides extrinsic reward, agent set **intrinsic reward** to express its familiarity with the true environment. (Curiosity Driven)
- Model Prediction:  $f(s_t, a_t) \rightarrow s_{t+1}$
- Intrinsic Reward:  $r_t^{in} \sim \|f(s_t, a_t) - s_{t+1}\|_2^2$



- **Forward Dynamics Model:**

$$f_{\psi_F}(\phi(s_t), a_t) \rightarrow \hat{\phi}(s_{t+1})$$

- **Inverse Model:**

$$g_{\psi_I}(\phi(s_t), \phi(s_{t+1})) \rightarrow \hat{a}_t$$

- **Intrinsic Reward:**

$$r_t^i = \|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2$$