Policy Gradient

Emile van Krieken
Reinforcement Learning Summer School

twitter.com/emilevankrieken

VRIJE UNIVERSITEIT AMSTERDAM

THIS LECTURE

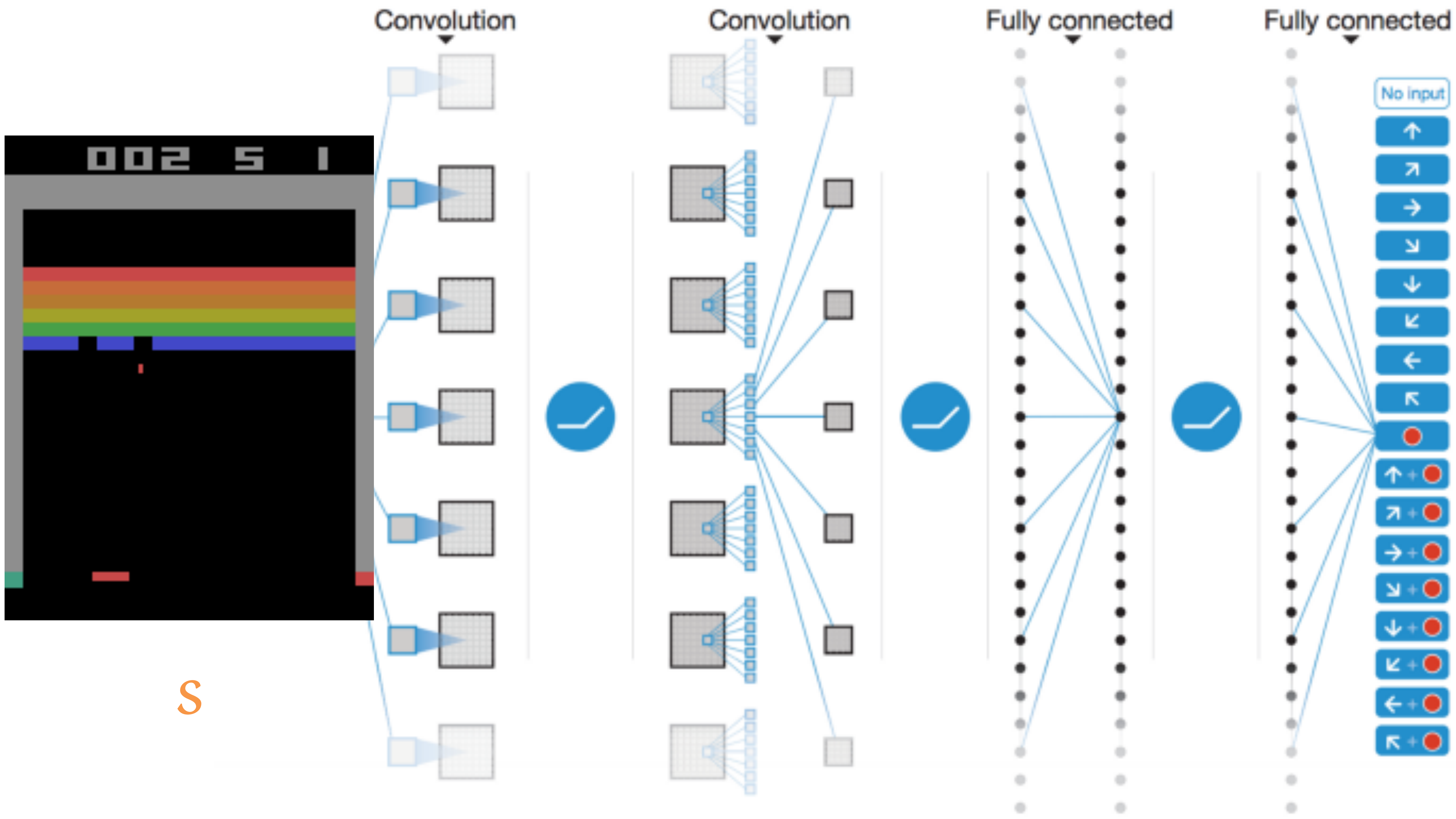
Deep Reinforcement Learning setting

- Neural network policies
- Model-free
- On-policy

Overview

- Deriving REINFORCE
- Actor-critic
- Advanced methods
  - TRPO, PPO
  - Soft Actor-Critic
  - World models

ATARI POLICY



s

Left

a

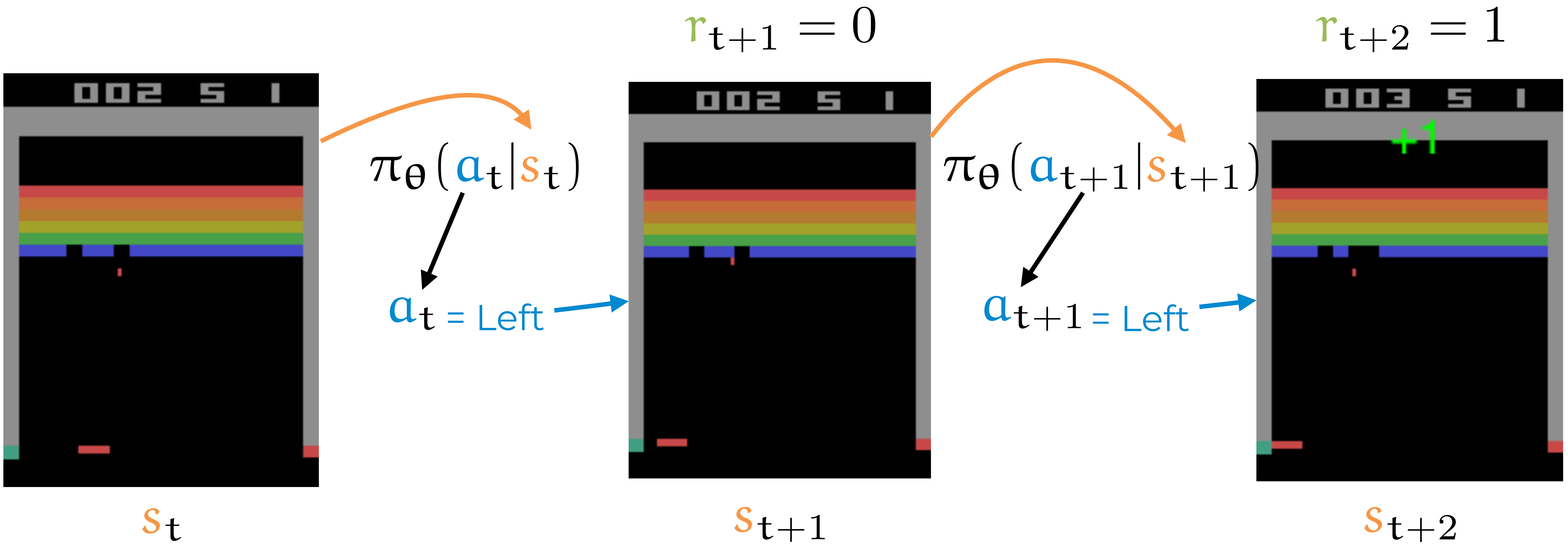
CNN policy network $\pi_\theta(a|s)$

Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. *Nature* 518, 529–533 (2015).

TRAJECTORIES

- Components of RL:
  - Actions a_t
  - States s_t
  - Rewards r_t
  - These are **random variables!**
- Trajectories $\boldsymbol{\tau} = s_0, a_0, r_1, s_1, a_1, r_2, s_2 \dots a_{T-1}, r_T, s_T$
- Initial state s_0
- Terminal state s_T

ATARI TRAJECTORY



Credit assignment:
What action causes reward?

https://becominghuman.ai/lets-build-an-atari-ai-part-0-intro-to-rl-9b2c5376e0ed

Trajectories $\tau = s_0, a_0, r_1, s_1, a_1, r_2, s_2 \ldots a_{T-1}, r_T, s_T$

Markov decision processes (MDPs):

$$p(\tau|\theta) = p(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) p(s_{t+1}, r_{t+1}|s_t, a_t)$$

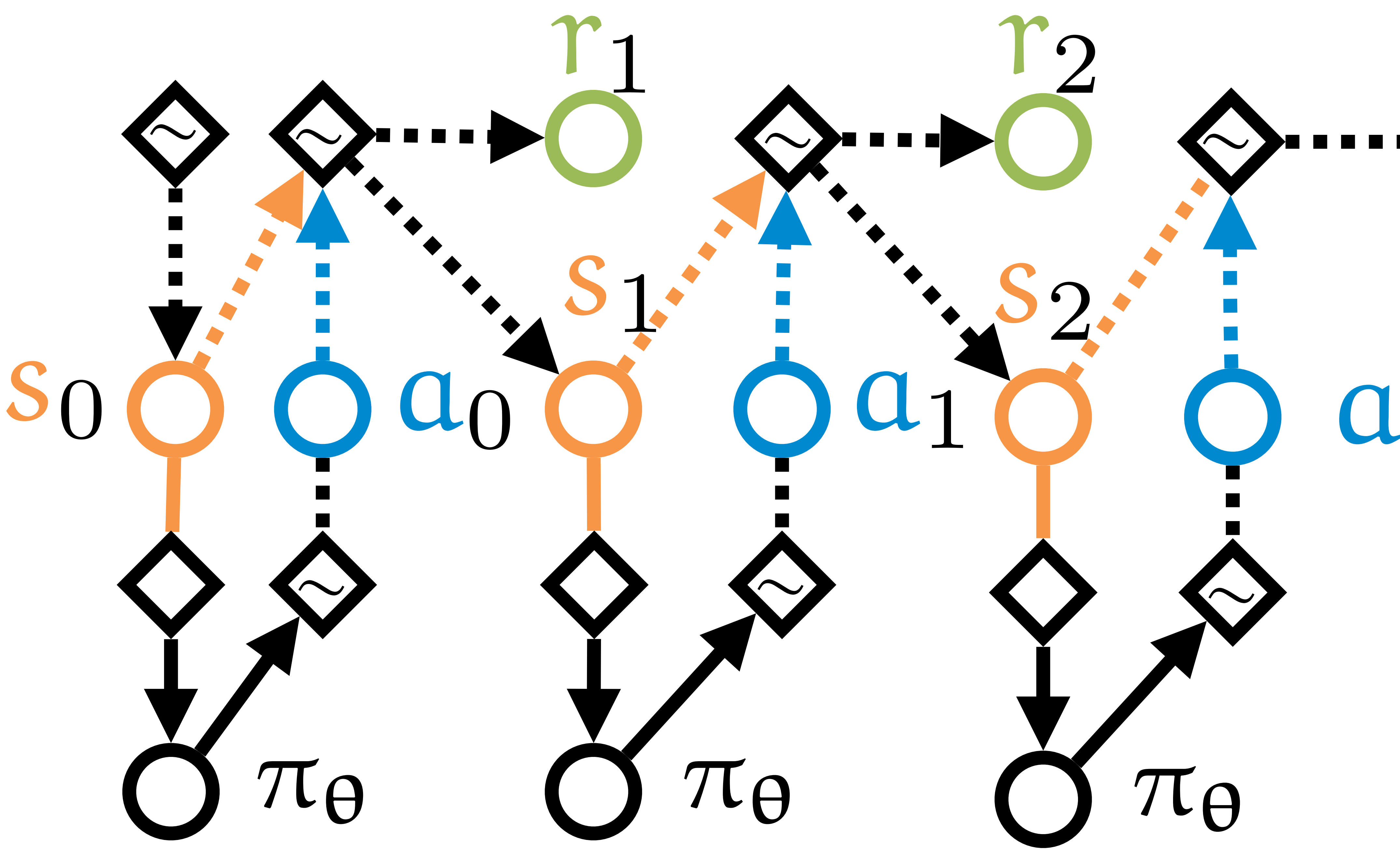
- Policy $\pi_\theta(a_t|s_t)$
- State transition distribution $p(s_{t+1}, r_{t+1}|s_t, a_t)$
- Initial state distribution $p(s_0)$


Environment
$p(s_0)$
$p(s_1, r_1|a_0, s_0)$
$p(s_2, r_2|a_1, s_1)$
$p(s_3, r_3|a_2, s$
r_1
r_2
r_3
s_0
s_1
s_2
s_3
$\pi_\theta(a_0|s_0)$
$\pi_\theta(a_1|s_1)$
$\pi_\theta(a_2|s_2)$
a_0
a_1
a_2
Agent
θ

OBSERVABILITY

- Full **observability** of state

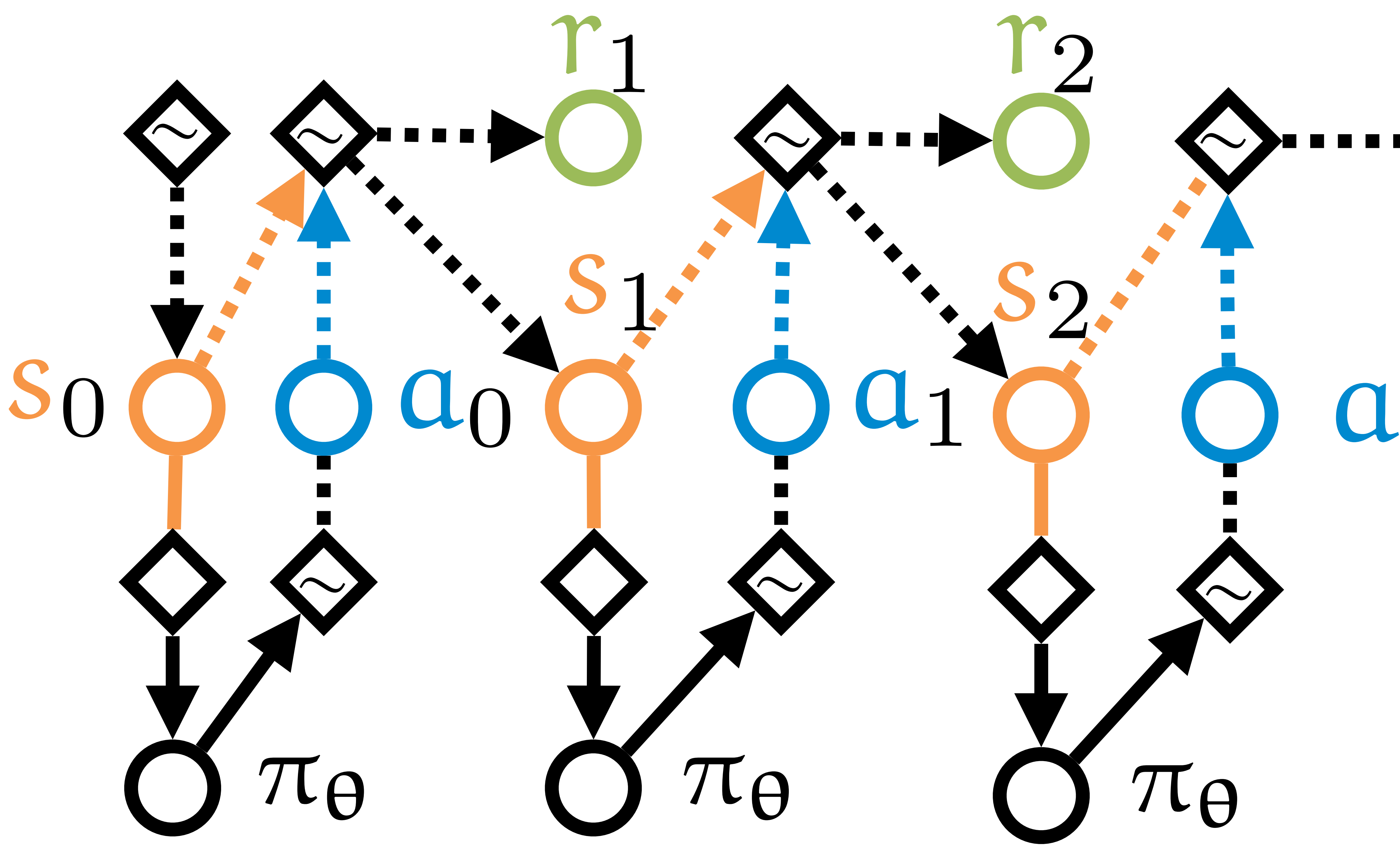
- Partial observability: POMDP
  - Out of scope!



s_t independent of history given s_{t-1}:

$$p(s_t | a_{t-1}, s_1, \ldots, s_{t-1}) = p(s_t | a_{t-1}, s_{t-1})$$

- Used to derive strong algorithms!
- Fundamental assumption behind RL
- No RL is completely "*model-free*"!



EXPECTED RETURN

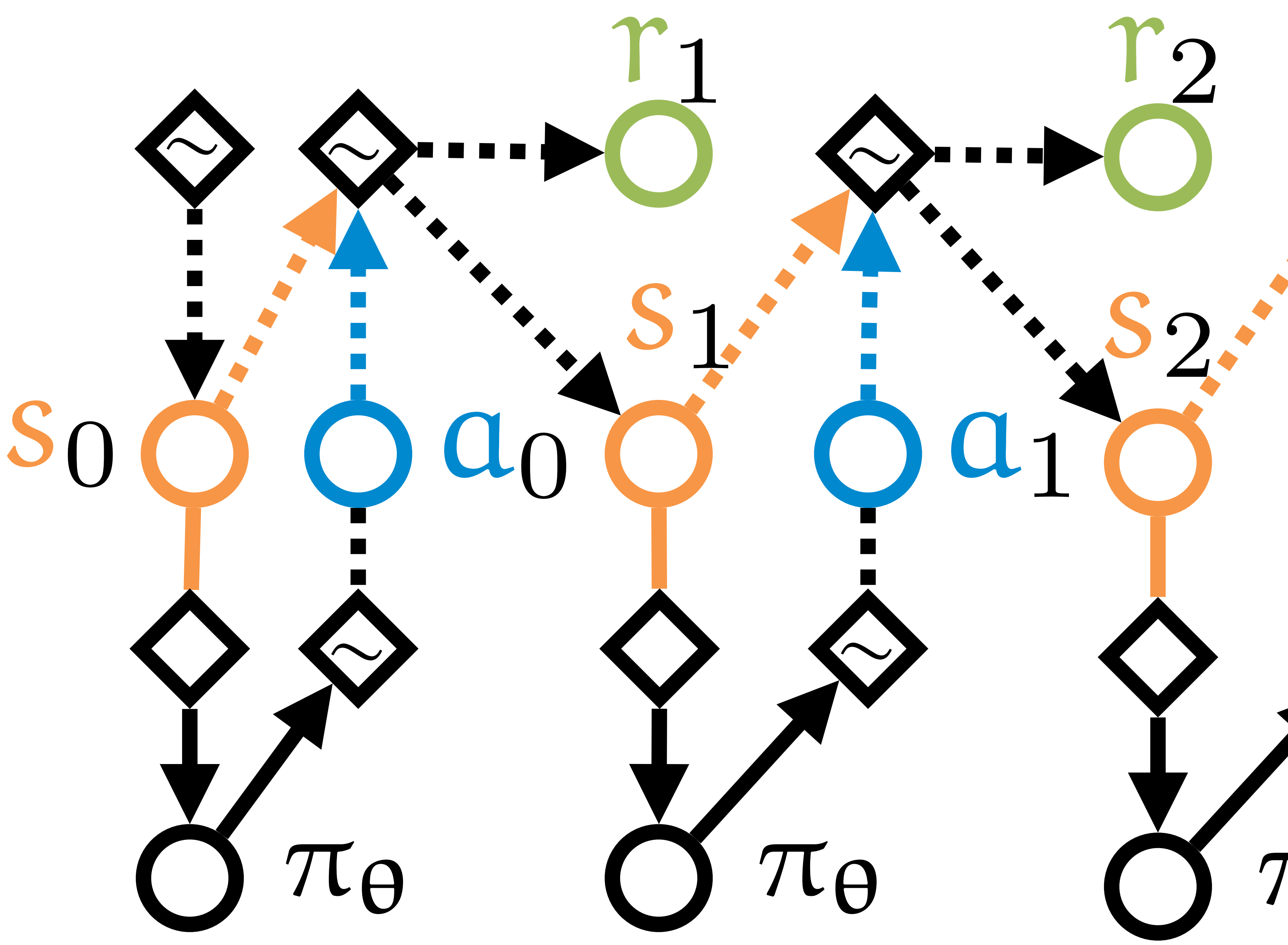
Total discounted reward $0 \leqslant \gamma \leqslant 1$

$$R_\gamma = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots + \gamma^{T-1} r_T$$

- Rewards and states are stochastic!
- **Goal:** Maximize expected return

$$J(\theta) = \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[R_\gamma]$$

$$\theta^* = \arg\max_\theta J(\theta)$$



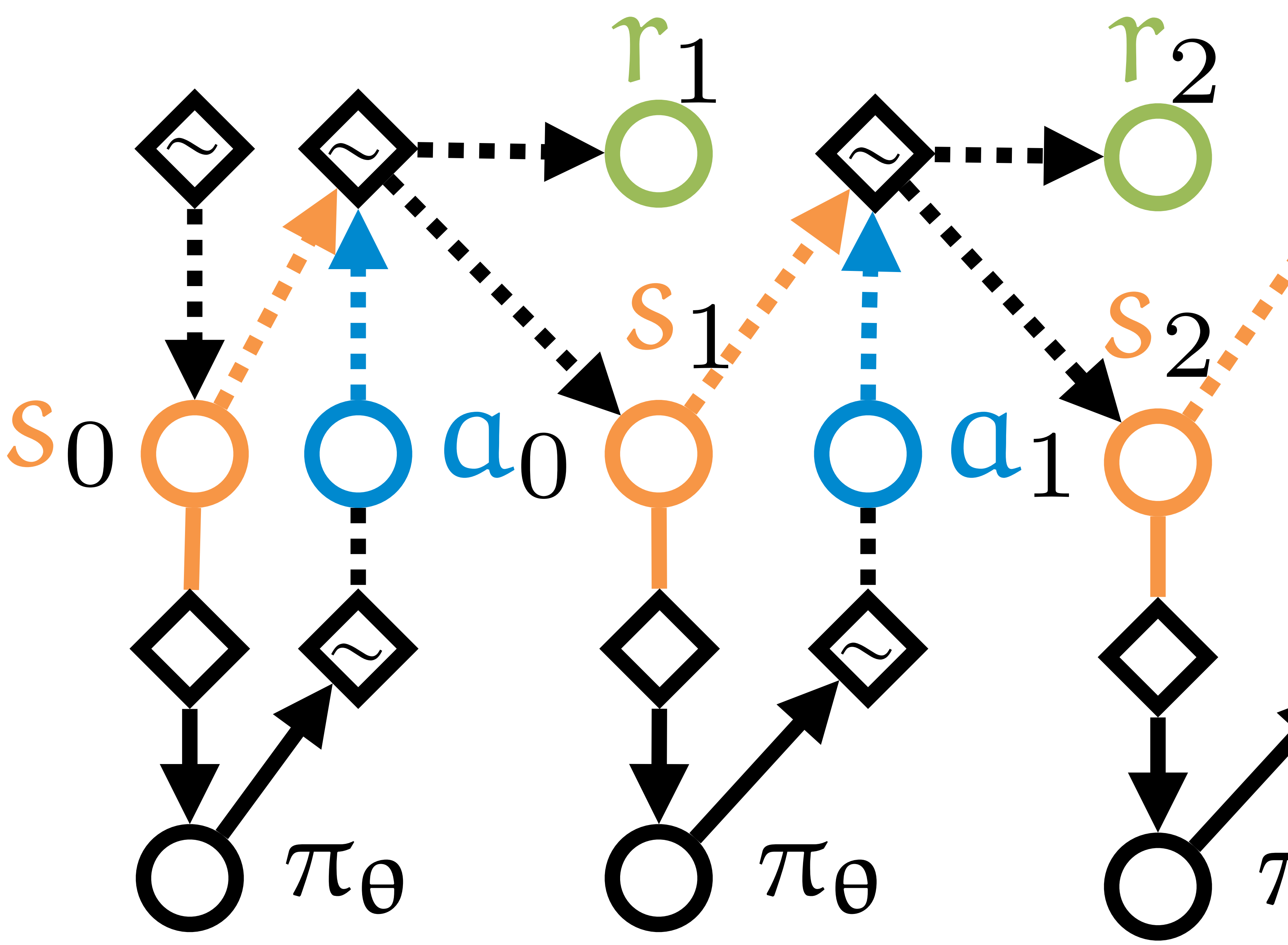
EXPECTED RETURN

$J(\theta) = \mathbb{E}_{p(\tau|\theta)}[R_\gamma]$

Expectation over **trajectories** by following policy π_θ

Requires summing (or integrating) over *all* trajectories!

→ Monte Carlo (sampling) estimation



MAXIMIZING EXPECTED RETURN

$J(\theta) = \mathbb{E}_{p(\tau|\theta)}[R_\gamma]$

How to find $\theta^* = \arg\max_\theta J(\theta)$?

- **Policy gradient methods**: Use $\nabla_\theta J(\theta)$ in gradient *ascent*
  $\theta_{i+1} = \theta_i + \alpha \nabla_{\theta_i} J(\theta_i)$

How to compute $\nabla_\theta J(\theta)$?
Gradient estimation!

SIMPLE REINFORCE

Simple **REINFORCE:**

1. $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\boldsymbol{\theta})$
2. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha R_\gamma \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)$

Sample trajectory

Gradient ascent

Let's derive algorithm!

JOINT DISTRIBUTION OF MDP

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{p(\tau|\theta)}[R_\gamma]$$

Expectation is over *all* trajectories τ :(

$$\nabla_\theta J(\theta) = \sum_\tau R_\gamma \nabla_\theta p(\tau|\theta)$$

To sample, we need an expression like

$$\sum_\tau p(\tau|\theta) f(\tau)$$

Solution: The **score function!**

THE SCORE FUNCTION

$$\nabla_\theta J(\theta) = \sum_\tau R_\gamma \nabla_\theta p(\tau|\theta)$$

$$= \sum_\tau R_\gamma \nabla_\theta p(\tau|\theta) \frac{p(\tau|\theta)}{p(\tau|\theta)}$$

Multiply by 1

$$= \sum_\tau R_\gamma p(\tau|\theta) \frac{\nabla_\theta p(\tau|\theta)}{p(\tau|\theta)}$$

This is an expression like $\sum_\tau p(\tau|\theta) f(\tau)$!

THE SCORE FUNCTION

$$\nabla_\theta J(\theta) = \sum_\tau R_\gamma \nabla_\theta p(\boldsymbol{\tau}|\theta)$$

$$= \sum_\tau R_\gamma \nabla_\theta p(\boldsymbol{\tau}|\theta) \frac{p(\boldsymbol{\tau}|\theta)}{p(\boldsymbol{\tau}|\theta)}$$

$$= \sum_\tau R_\gamma p(\boldsymbol{\tau}|\theta) \frac{\nabla_\theta p(\boldsymbol{\tau}|\theta)}{p(\boldsymbol{\tau}|\theta)}$$

$$= \sum_\tau p(\boldsymbol{\tau}|\theta) R_\gamma \nabla_\theta \log p(\boldsymbol{\tau}|\theta)$$

?

$$= \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[R_\gamma \nabla_\theta \log p(\boldsymbol{\tau}|\theta)]$$

Score function:

$$\nabla_\theta \log p(\boldsymbol{\tau}|\theta)$$

$$= \frac{\partial \log p(\boldsymbol{\tau}|\theta)}{\partial p(\boldsymbol{\tau}|\theta)} \frac{\partial p(\boldsymbol{\tau}|\theta)}{\partial \theta}$$

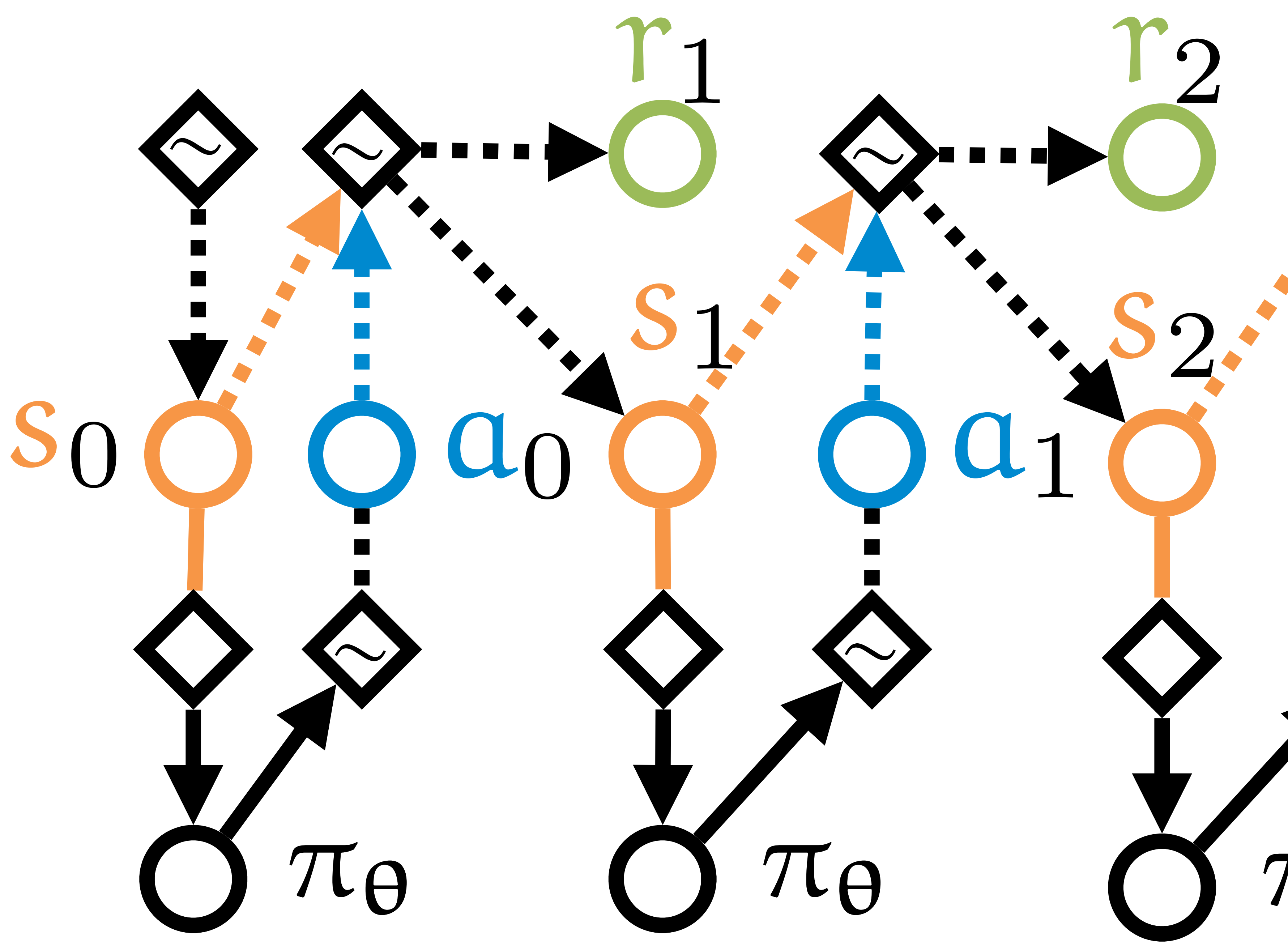
$$= \frac{1}{p(\boldsymbol{\tau}|\theta)} \nabla_\theta p(\boldsymbol{\tau}|\theta)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{p(\tau|\theta)}[R_\gamma \nabla_\theta \log p(\tau|\theta)]$$

How to compute $\nabla_\theta \log p(\tau|\theta)$?

MDP distribution over trajectories:

$$p(\tau|\theta) = p(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) p(s_{t+1}, r_{t+1}|s_t, a_t)$$



WORKING OUT MDP

$$p(\boldsymbol{\tau}|\theta) = p(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) p(s_{t+1}, r_{t+1}|s_t, a_t)$$

$$\log p(\boldsymbol{\tau}|\theta) = \log p(s_0) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}, r_{t+1}|s_t, a_t)$$

$$\nabla_\theta \log p(\boldsymbol{\tau}|\theta) = \nabla_\theta \log p(s_0) + \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

$$+ \nabla_\theta \log p(s_{t+1}, r_{t+1}|s_t, a_t)$$

$$\nabla_\theta \log p(\boldsymbol{\tau}|\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Gradient of environment wrt policy parameters θ is 0!

BASIC REINFORCE

$$\nabla_\theta \log p(\boldsymbol{\tau}|\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[R_\gamma \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

$$\approx R_\gamma \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t), \quad \boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\theta)$$

Monte Carlo (sample) estimate

$$\nabla_\theta J(\theta) = \mathbb{E}_{p(\tau|\theta)}[R_\gamma \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

Reinforce actions with high *total* return

- Reinforce a_{T-1} when r_1 is high?
- Only reinforce actions with good *consequences*!

VU

Gradient of reward at $t' + 1$:

$$\nabla_\theta \mathbb{E}_{p(\tau|\theta)}[\gamma^{t'} r_{t'+1}] = \mathbb{E}_{p(\tau|\theta)}[\gamma^{t'} r_{t'+1} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

$$= \mathbb{E}_{p(\tau|\theta)}[\gamma^{t'} r_{t'+1} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

Only influenced by actions until t'

BETTER REINFORCE

Sum over timesteps:

$$\nabla_\theta \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[R_\gamma] = \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[\sum_{t'=0}^{T-1} \gamma^{t'} r_{t'+1} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

Equivalent: Update actions based on following rewards

- Discounted reward to go

$$G_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'+1}$$

Gradient estimate:

$$\nabla_\theta \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[R_\gamma] = \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[\sum_{t=0}^{T-1} \gamma^{t} G_t \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

$$G_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'+1}$$

REINFORCE:

1. $\tau \sim p(\tau|\theta)$
2. $\theta \leftarrow \theta + \alpha \sum_{t=0}^{T-1} \gamma^t G_t \nabla_\theta \log \pi_\theta(a_t|s_t)$

Sample trajectory

Gradient ascent

ACTOR-CRITIC

VARIANCE

Variance:

$$\mathbb{V}[\mathbf{g}] = \mathbb{E}_{p_\theta}\left[\sum_{i=1}^{D}(\mathbf{g}_i - \mathbb{E}_{p_\theta}[\mathbf{g}_i])^2\right]$$

High variance

→ More samples needed

→ Unstable training

REINFORCE

- Simplest method to approximate policy gradient
- General and unbiased :)
- *Very* high **variance**! :(
  - Not **sample efficient**

REINFORCE:

$$\nabla_\theta \mathbb{E}_{p(\tau|\theta)}[R_\gamma] = \mathbb{E}_{p(\tau|\theta)}[\sum_{t=0}^{T-1} \gamma^t G_t \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

Reduce variance with **baseline** b_t:

$$\nabla_\theta \mathbb{E}_{p(\tau|\theta)}[R_\gamma] = \mathbb{E}_{p(\tau|\theta)}[\sum_{t=0}^{T-1} \gamma^t (G_t - b_t) \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

baseline

BASELINES

$$\mathbb{E}_{\pi_\theta(a_t|s_t)}[b_t \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

$$= \sum_{a_t} b_t \cancel{\pi_\theta(a_t|s_t)} \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\cancel{\pi_\theta(a_t|s_t)}}$$

BASELINES

$$\mathbb{E}_{\pi_\theta(a_t|s_t)}[b_t \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

$$= \sum_{a_t} b_t \cancel{\pi_\theta(a_t|s_t)} \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\cancel{\pi_\theta(a_t|s_t)}}$$

$$= \sum_{a_t} b_t \nabla_\theta \pi_\theta(a_t|s_t) = b_t \nabla_\theta \sum_{a_t} \pi_\theta(a_t|s_t)$$

$$= b_t \nabla_\theta 1 = 0$$

WHAT BASELINE?

REINFORCE with **baseline**:

$$\nabla_\theta \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[R_\gamma] = \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[\sum_{t=0}^{T-1} \gamma^t (G_t - b_t) \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

Value function:

$$V^\pi(s_t) = \mathbb{E}_{p(\boldsymbol{\tau}|\pi, s_t)}[G_t]$$

Value function baseline:

$$\nabla_\theta \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[R_\gamma] = \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[\sum_{t=0}^{T-1} \gamma^t (G_t - V^\pi(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

Act, receive reward.

How to reinforce?

$V^{\pi}(s_t) = 0.63$

REINFORCE:

I won!

Random reward

REINFORCE + baseline:

I won. That result is 37% better than expected!

Increase of random reward wrt expected reward

TRAINING VALUE FUNCTION

Like Deep Q-Learning, train neural network V_ϕ with regression.

1. Use rollouts:

$$\phi \leftarrow \phi - \alpha \sum_{t=0}^{T-1} (V_\phi(s_t) - G_t)^2$$

target: reward-to-go

2. Use bootstrapping (lower variance, biased):

$$\phi \leftarrow \phi - \alpha \sum_{t=0}^{T-1} (V_\phi(s_t) - \perp(r_{t+1} + \gamma V_\phi(s_{t+1})))^2$$

target: bootstrapped expected reward-to-go

REINFORCE with baseline:

1. $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\boldsymbol{\theta})$
2. $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha_c \sum_{t=0}^{T-1} \left(V_{\boldsymbol{\phi}}(s_t) - \bot(r_{t+1} + \gamma V_{\boldsymbol{\phi}}(s_t))\right)^2$
3. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_a \sum_{t=0}^{T-1} \gamma^t (G_t - V_{\boldsymbol{\phi}}(s_t)) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$

Discounted reward to-go

$$G = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'+1}$$

Q-function (state-action value function):

$$Q^{\pi}(s, a) = \mathbb{E}_{p(\boldsymbol{\tau}|\pi, s, a)}[G]$$

Q-FUNCTIONS IN POLICY GRADIENTS

$$Q^{\pi}(s, a) = \mathbb{E}_{p(\boldsymbol{\tau}|\pi, s, a)}[G]$$

Policy gradient:

$$\nabla_{\theta} \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[R_{\gamma}] = \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[\sum_{t=0}^{T-1} \gamma^{t} G_{t} \nabla_{\theta} \log \pi_{\theta}(a_{t}|s_{t})]$$

$$= \mathbb{E}_{p(\boldsymbol{\tau}|\theta)}[\sum_{t=0}^{T-1} \gamma^{t} Q^{\pi}(s_{t}, a_{t}) \nabla_{\theta} \log \pi_{\theta}(a_{t}|s_{t})]$$

critic actor

Much lower variance!

NOTATION UPDATE

- Declutter notation:
  - Current timestep t:
    $r_t = r, a_t = a, s_t = s, G_t = G$
  - Next timestep $t + 1$:
    $r_{t+1} = r', a_{t+1} = a', s_{t+1} = s', G_{t+1} = G'$

BOOTSTRAP ERROR

Minimize **bootstrapped** error using regression:

$$\arg\min_{\theta} \Big(\underbrace{Q_\theta(s_t, a_t)}_{\text{prediction}} - \underbrace{\mathbb{E}_{p(s_{t+1}, r_{t+1} | s_t, a_t)} [r_{t+1} + \gamma \max_{a_{t+1}} Q_\theta(s_{t+1}, a_{t+1})]}_{\text{target}} \Big)^2$$

REINFORCE VS ACTOR-CRITIC

Act, receive reward.

How to reinforce?

REINFORCE:

I won!

Random reward

Actor-critic:

I think I'll win with 0.63 probability!

Expected reward

BASELINES

Actor-critic:

$$\nabla_\theta \mathbb{E}_{p(\tau|\theta)}[R_\gamma] = \mathbb{E}_{p(\tau|\theta)}[\sum_{t=0}^{T-1} \gamma^t Q^\pi(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

Reduce variance even more with value function **baseline**:

$$\nabla_\theta \mathbb{E}_{p(\tau|\theta)}[R_\gamma] = \mathbb{E}_{p(\tau|\theta)}[\sum_{t=0}^{T-1} \gamma^t (Q^\pi(s_t, a_t) - V^\pi(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

advantage   actor

Advantage actor-critic

ADVANTAGE VS Q-FUNCTION

Act, receive reward.

How to reinforce?

Actor-critic:

I think I'll win with 63% probability!

Expected reward

Advantage actor-critic:

I think I'll be 3% *more likely* to win.

Expected increase in reward

Advantage actor-critic:

$$\nabla_\theta \mathbb{E}_{p(\tau|\theta)}[R_\gamma] = \mathbb{E}_{p(\tau|\theta)}[\sum_{t=0}^{T-1} \gamma^t (Q^\pi(s_t, a_t) - V^\pi(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

advantage A^π

Estimate V^π with V_ϕ (biased)

Estimate $Q^\pi(s_t, a_t)$ with $r_{t+1} + \gamma V_\phi(s_{t+1})$

$$A_\phi(s_t, r_{t+1}, s_{t+1}) = r_{t+1} + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

SUMMARY

Advantage actor-critic:

- Estimate advantage for current policy
- Use estimate to get improved policy

Like policy iteration, but with gentle steps

Online Actor-Critic:

1. $a \sim \pi_\theta(a|s)$ — Select actions according to policy
2. $s', r' \sim p(s', r'|s, a)$
3. $\phi \leftarrow \phi - \alpha_c (V_\phi(s) - \perp(r' + \gamma V_\phi(s')))^2$ — Update critic
4. $\theta \leftarrow \theta + \alpha_a A_\phi(s, r', s') \nabla_\theta \log \pi_\theta(a|s)$ — Update actor
5. $s \leftarrow s'$

A2C

Uses only a single sample

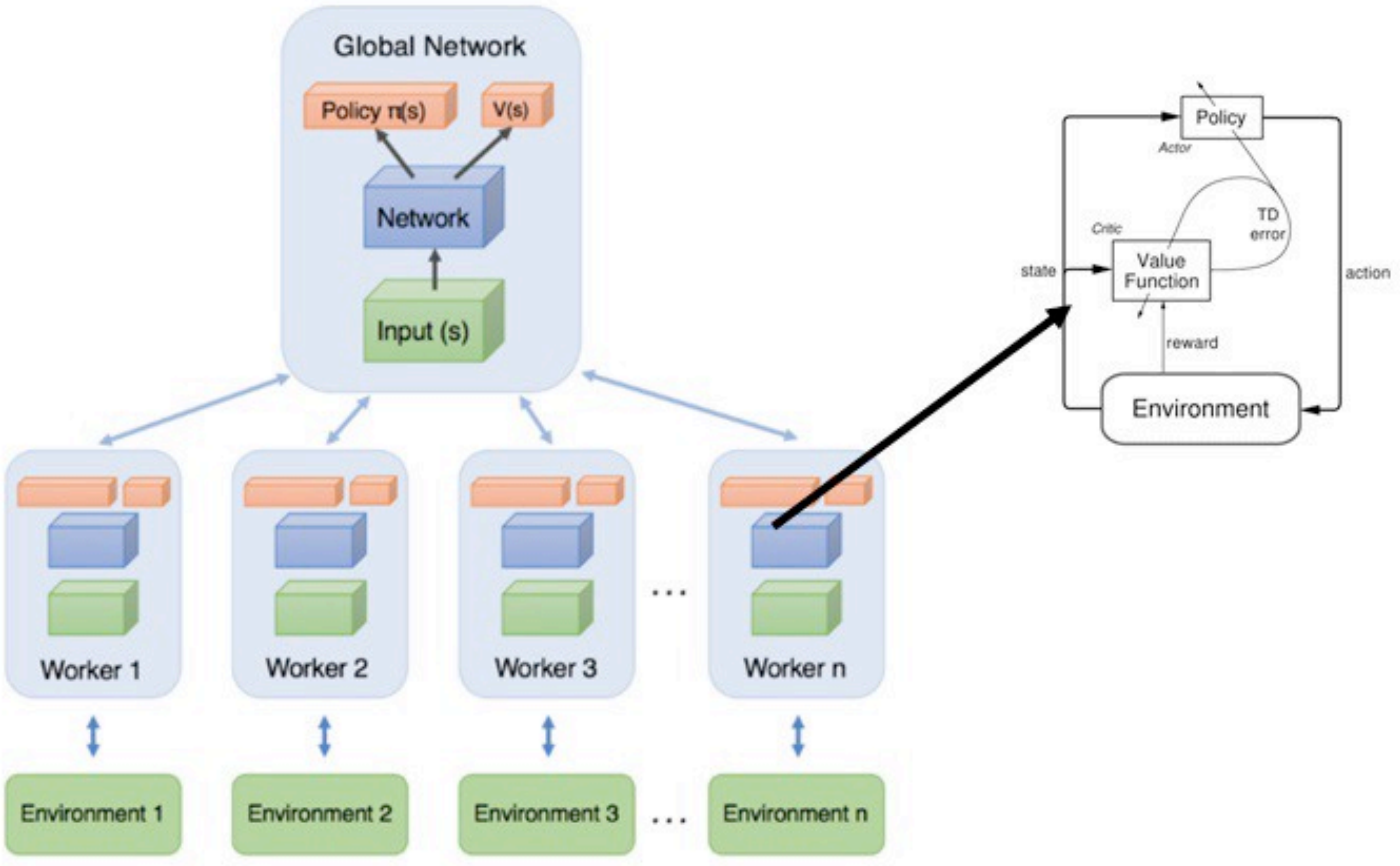
And batching over time would give correlated minibatches

A2C: Multiple online agents

Collect experiences at each step for minibatch.

Efficient method!

A3C


Global Network
Policy π(s)
V(s)
Network
Input (s)
Worker 1
Worker 2
Worker 3
Worker n
Environment 1
Environment 2
Environment 3
Environment n
Policy
Actor
Critic
TD error
state
Value Function
action
reward
Environment

ADVANCED POLICY GRADIENT METHODS

TRUST REGIONS

- Take small steps in policy space
  - Policy is *close* to another if KL-divergence is low
  - Normal policy gradient: Difference in *parameter space*

$$\theta_{k+1} = \arg\max_{\theta} \bar{A}\left(\theta_k, \theta\right)$$

$$\text{s.t. } \bar{D}_{KL}\left(\theta \| \theta_k\right) \leqslant \epsilon$$

- How to ensure closeness?

TRPO

$$\theta_{k+1} = \arg\max_{\theta} \bar{A}\left(\theta_k, \theta\right)$$

$$\text{s.t. } \bar{D}_{KL}\left(\theta \| \theta_k\right) \leqslant \epsilon$$

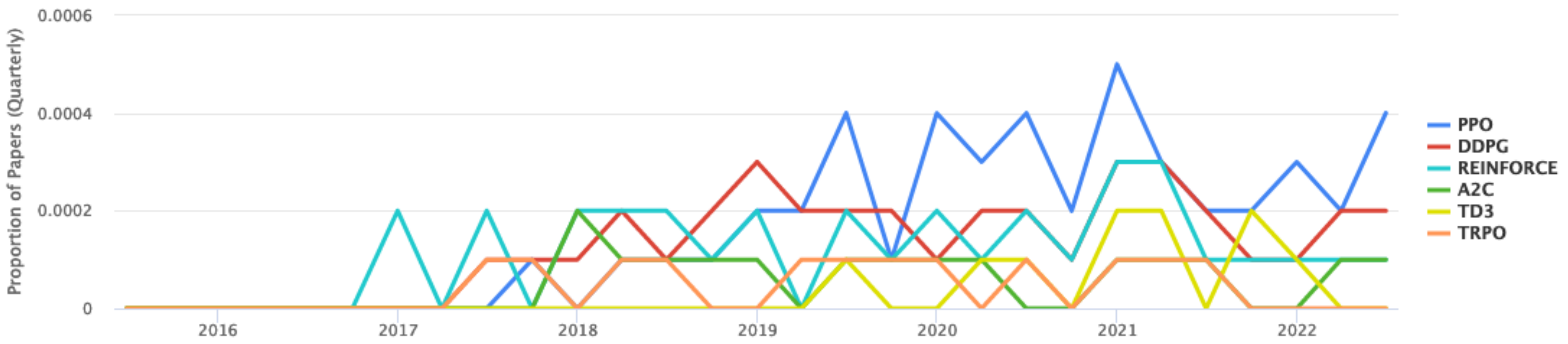
- How to ensure closeness?
- TRPO:
  - Uses *Natural Gradient*
  - Rescale AC-gradient by Fisher Information Matrix
  - Optimized using conjugate gradient
  - Complex to understand & implement well

PPO

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

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

- How to ensure closeness?
- PPO Clip:
  - Clipped 'importance weights' between old and new policy
  - Discourages large policy changes
  - Simple to implement & popular!



GRADIENT ESTIMATION

Policy gradient methods:

Estimate gradient of *expected* return

Gradient estimation:

Estimate gradient of *any* expectation

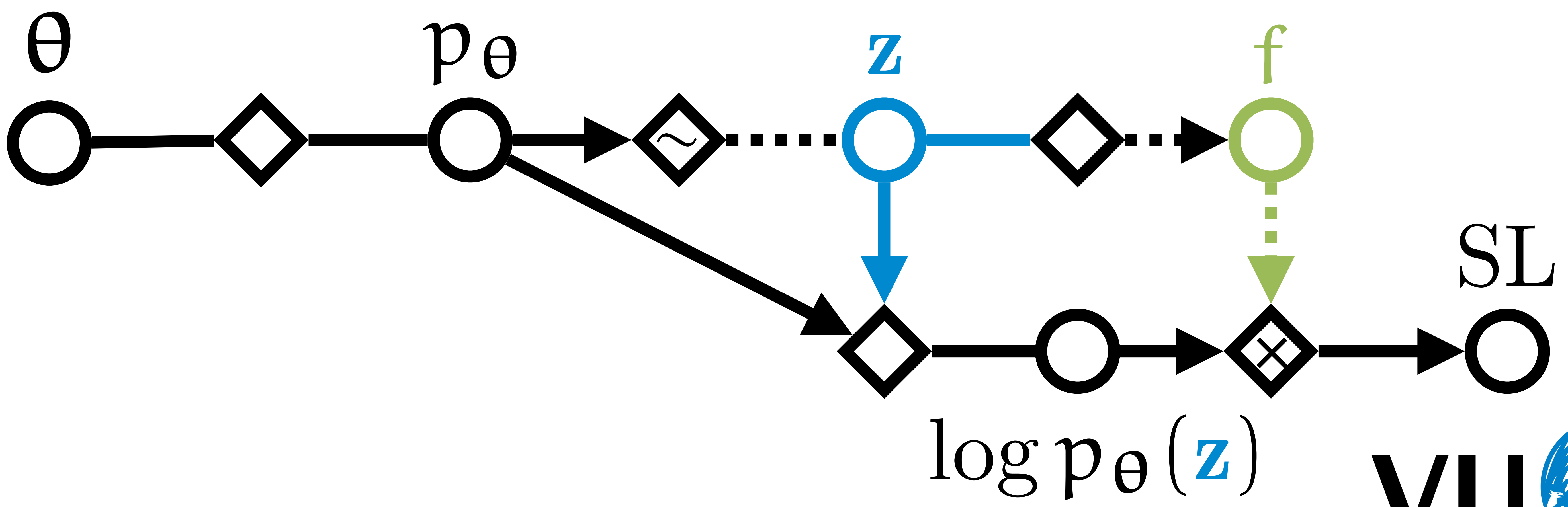
$$\arg\max_{\theta} \mathbb{E}_{p_\theta(\mathbf{z})}[f(\mathbf{z})]$$

SCORE FUNCTION

Recall **score function:**

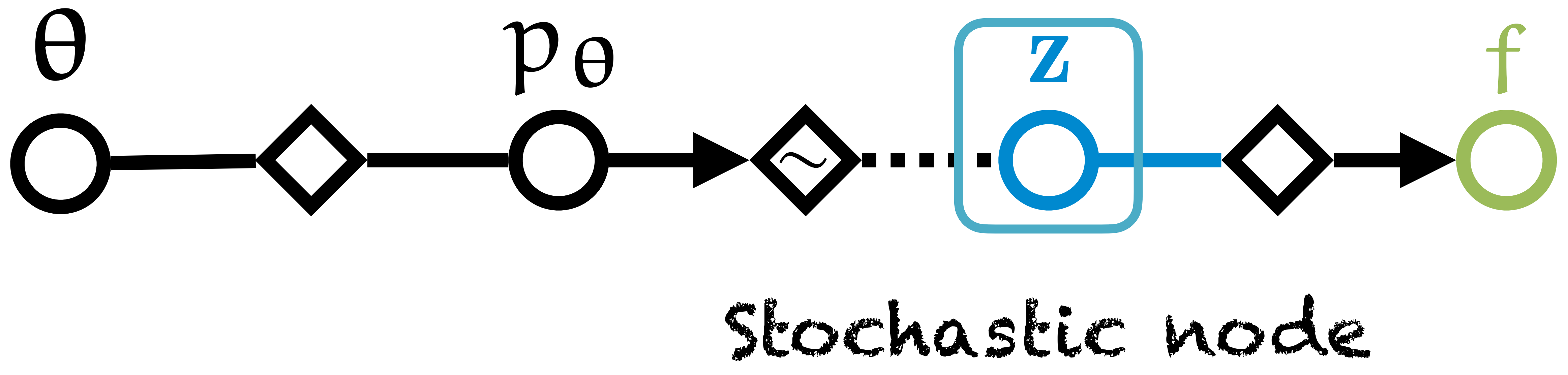
$$\nabla_\theta \mathbb{E}_{p_\theta(z)}[f(z)] = \mathbb{E}_{p_\theta(z)}[f(z)\frac{\nabla_\theta p_\theta(z)}{p_\theta(z)}]$$
$$= \mathbb{E}_{p_\theta(z)}[f(z)\nabla_\theta \log p_\theta(z)]$$

- *All* distributions $p_\theta(z)$
- *All* functions f
- But very high **variance**



Score function has high variance…

Can we do better?

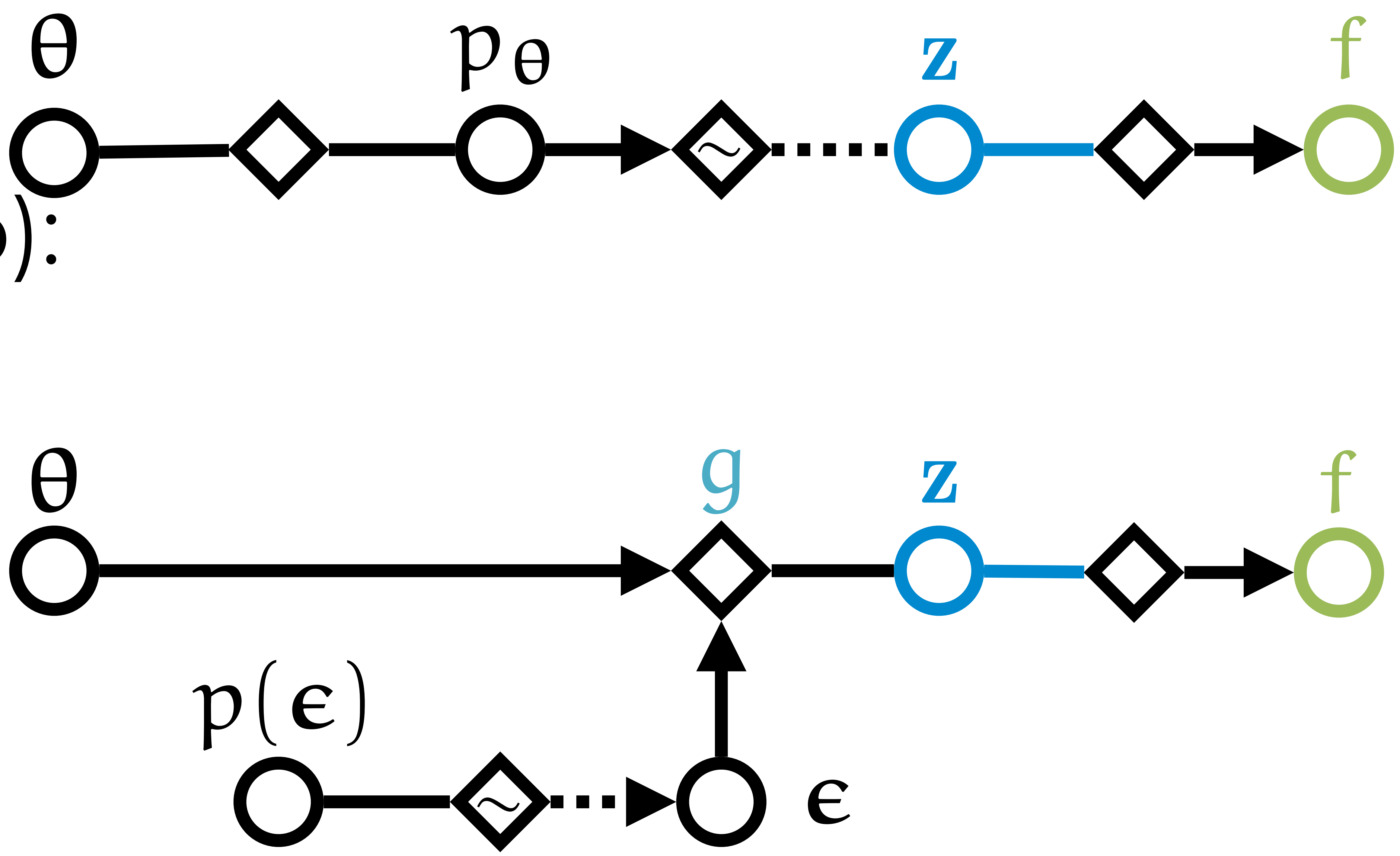


Reparameterization:

$$\mathbb{E}_{p_\theta(z)}[f(z)] = \mathbb{E}_{p(\epsilon)}[f(g(\theta, \epsilon))]$$

- Noise distribution $p(\epsilon)$

$$z = g(\theta, \epsilon) \sim p_\theta(z)$$



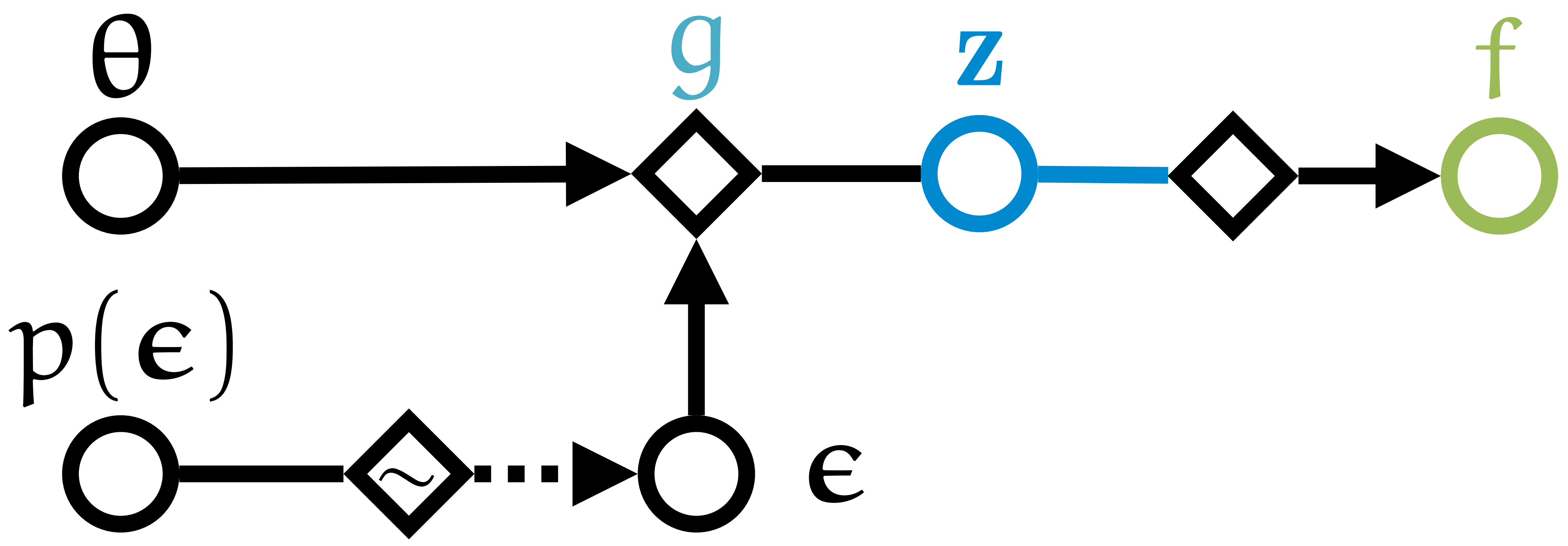
Pathwise derivative (=backprop):

$$g_{PD} = \frac{\partial f}{\partial z}\frac{\partial z}{\partial \theta}, \qquad \epsilon \sim p(\epsilon)$$

PATHWISE DERIVATIVE

Low variance :)

- Uses extra info: $\frac{\partial f}{\partial \mathbf{z}}$

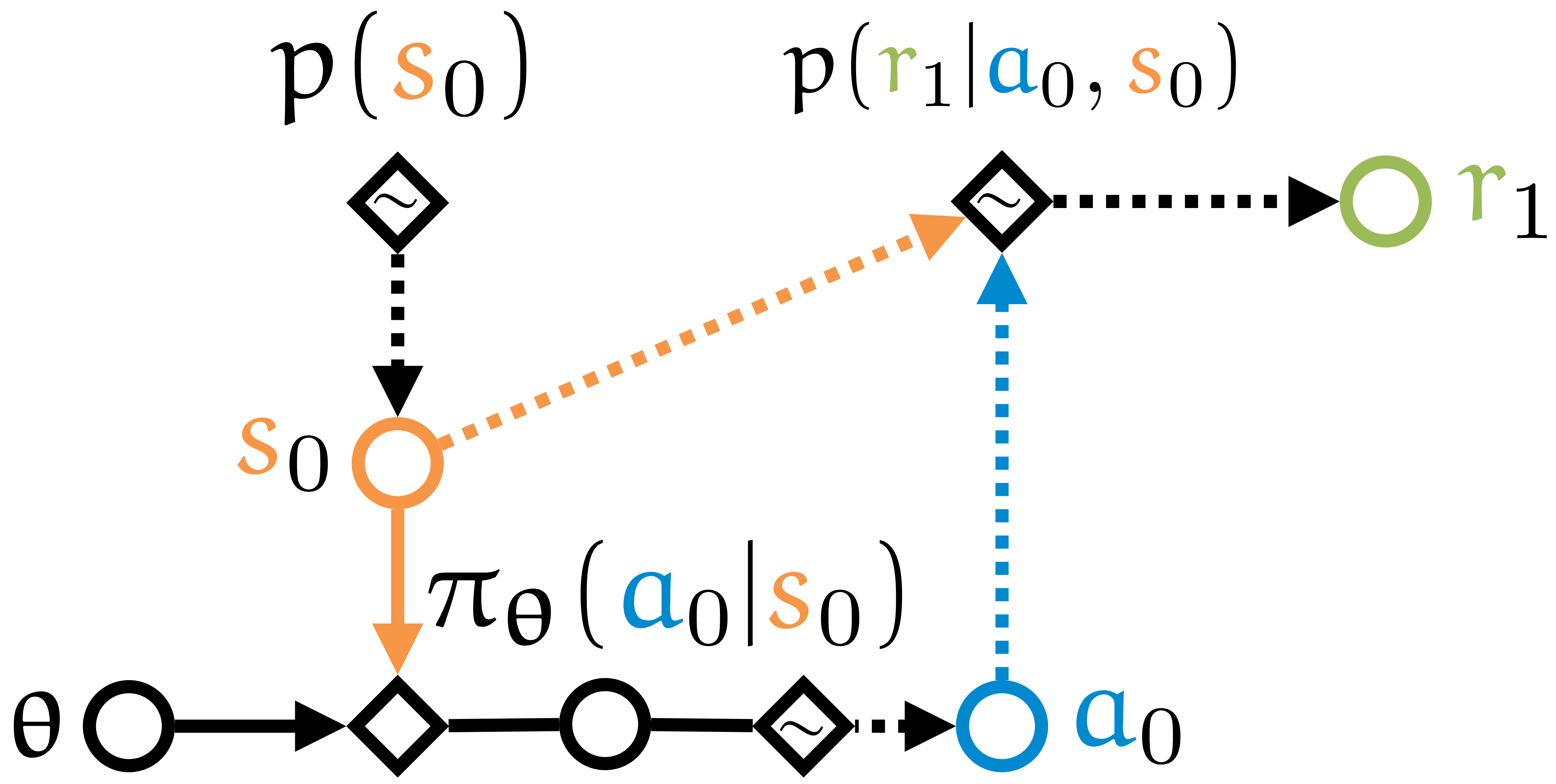


Requires:

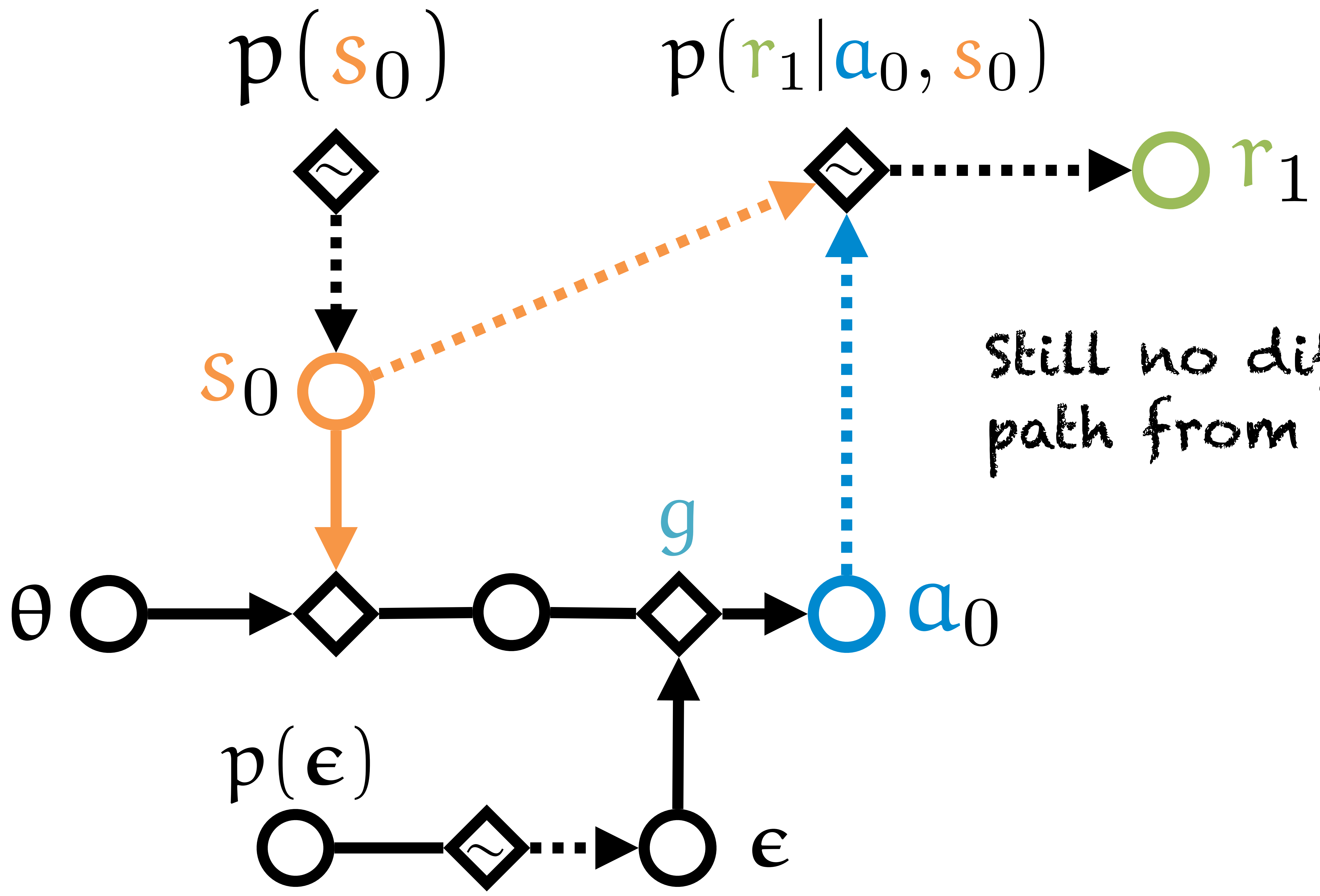
- Differentiable function f :(
- Appropriate *continuous* distribution $p_\theta(\mathbf{z})$:(

No reparameterization for *discrete* distributions

REPARAMETERIZATION IN RL?

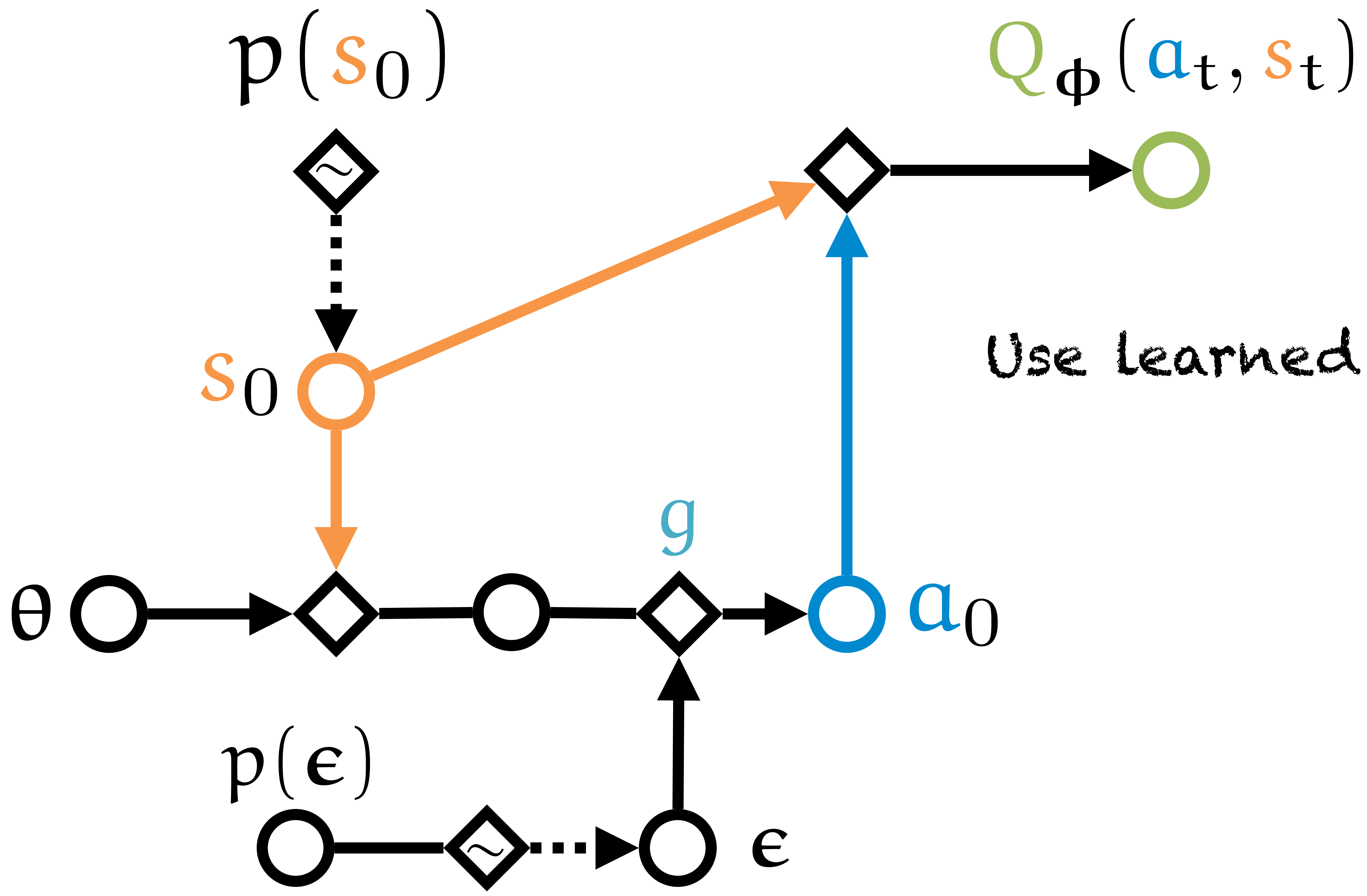


REPARAMETERIZATION IN RL?



Still no differentiable path from r_1 to θ!

REPARAMETERIZATION IN RL!

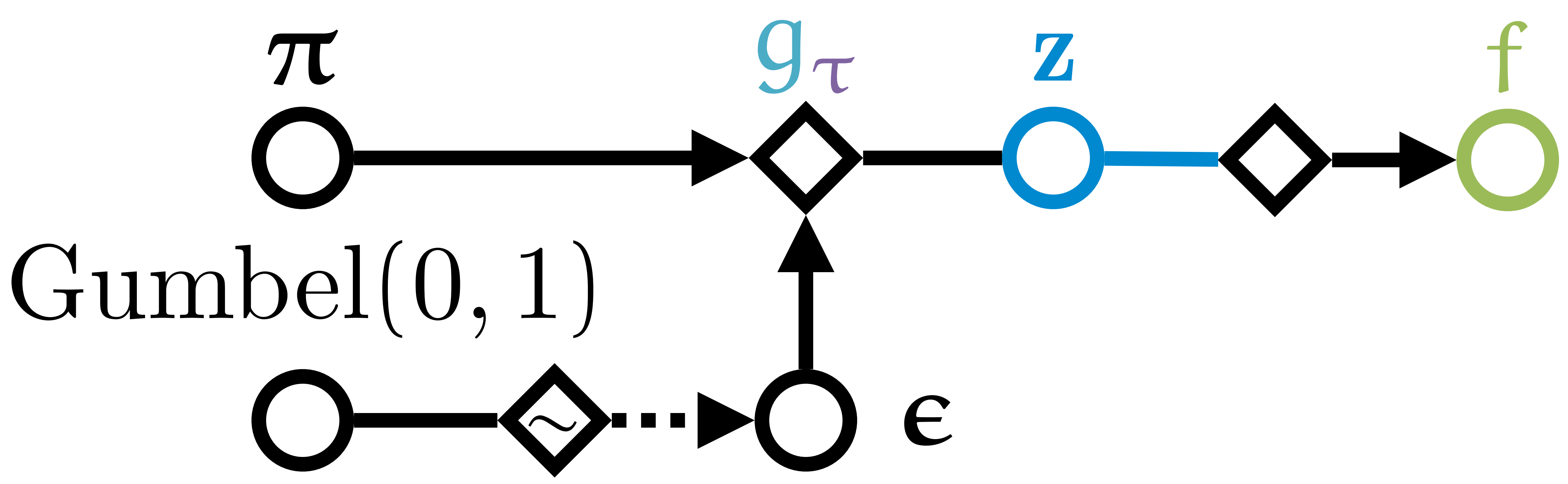


Use learned Q-function!

Probabilities $\pi_1, \ldots, \pi_K$, temperature $\tau > 0$

$\epsilon_1, \ldots, \epsilon_K \sim \text{Gumbel}(0, 1)$

$\mathbf{z} = g_\tau(\boldsymbol{\epsilon}, \boldsymbol{\pi}) = \text{Softmax}((\log \boldsymbol{\pi} + \boldsymbol{\epsilon})/\tau)$



Storchastic: Define computation graph with sampling steps.
Compute gradient estimators *automatically*!

- PyTorch library with easy API
- Many low-variance estimators implemented
- Focus on discrete distributions

STorchastic *https://github.com/HEmile/storchastic*

VU

SOFT ACTOR CRITIC

- **Off**-policy algorithm
- Uses reparameterization to maximize through critic
- Adds **entropy-regularization**
  - Encourage exploration
- Similar algorithms
  - Deep Deterministic Policy Gradient (DDPG)
  - Twin Delayed DDPG (TD3)

ENVIRONMENT

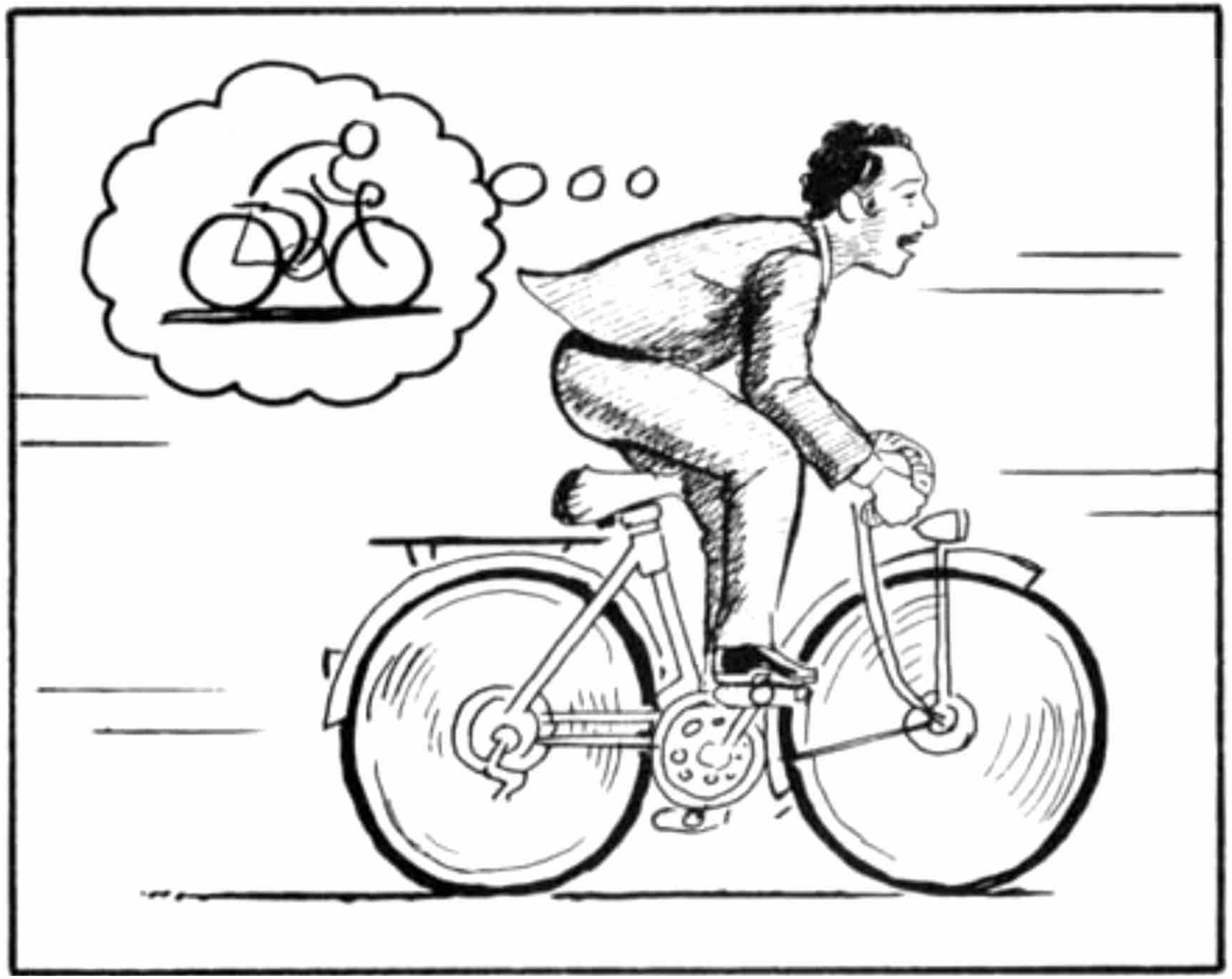
Actor-critic methods

- Model actor: Policy NN
- Model rewards: Value function NN

What about 3rd RL component: **environment**?

World Models

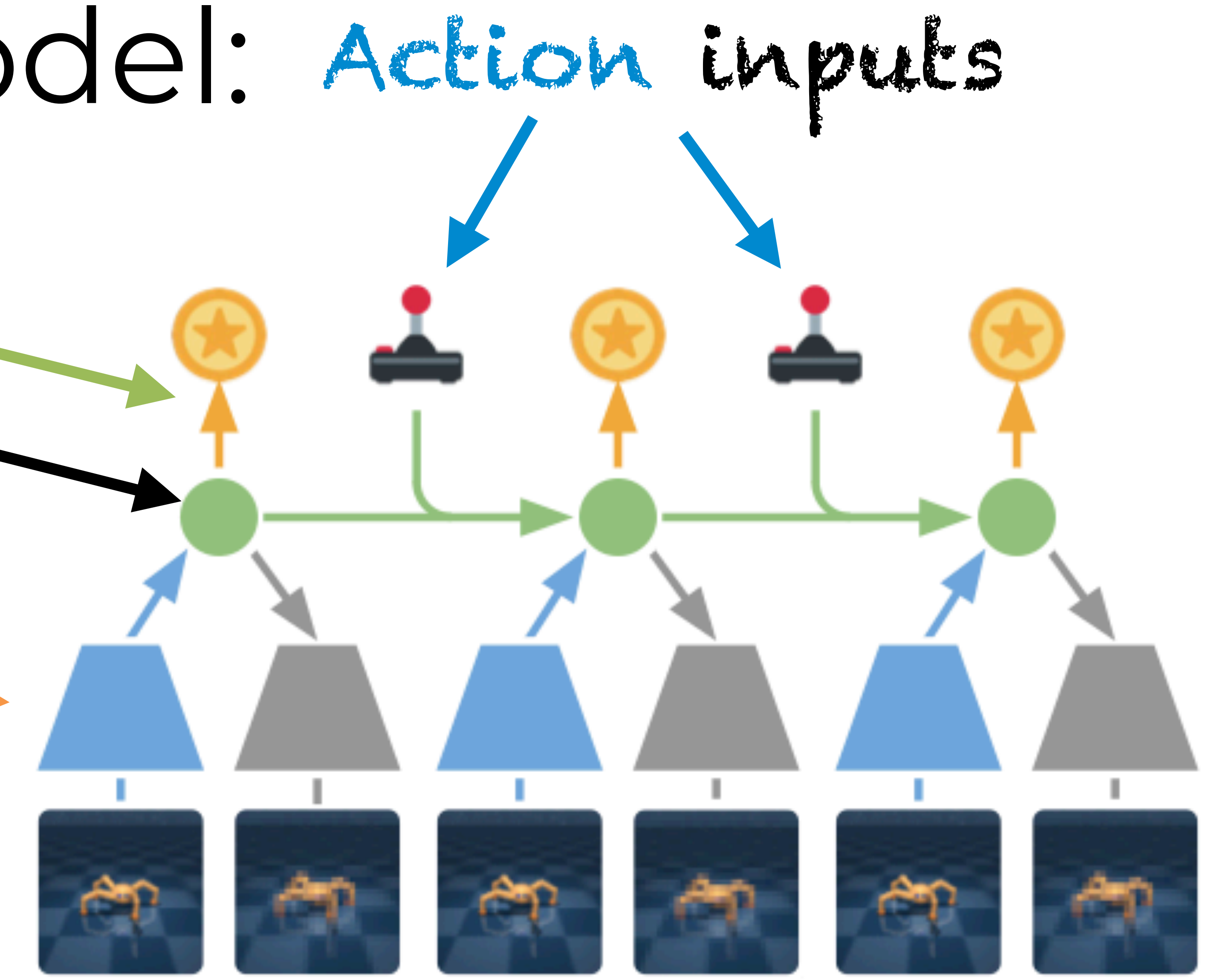
- Model **environment** using neural networks!



McCloud, Scott. Understanding Comics:The Invisible Art.Tundra Publishing, 1993.
Ha, David, and Jürgen Schmidhuber. "World models."

Components of our World Model:

- Reward model $p_\psi(r|z)$
- Transition model $p_\psi(z'|z, a)$
- State encoder $q_\psi(z|s)$



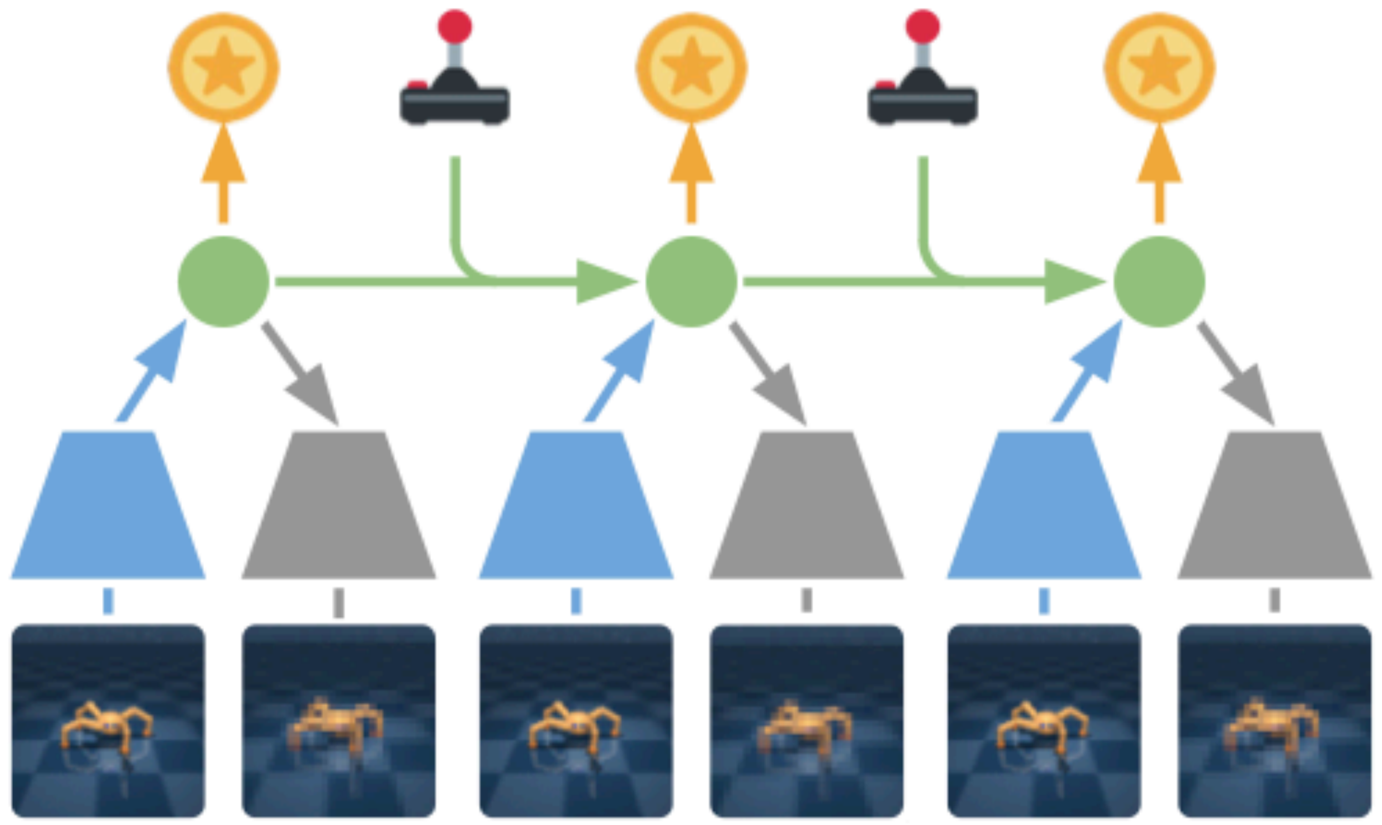
Hafner, Danijar, et al. "Dream to Control: Learning Behaviors by Latent Imagination." International Conference on Learning Representations. 2019.

TRANSITION DYNAMICS

World models can be used to **dream** trajectories

(more formally, **latent imagination**)

Use to train RL agents $\pi_\theta(a|z)$ *without interaction with environment*

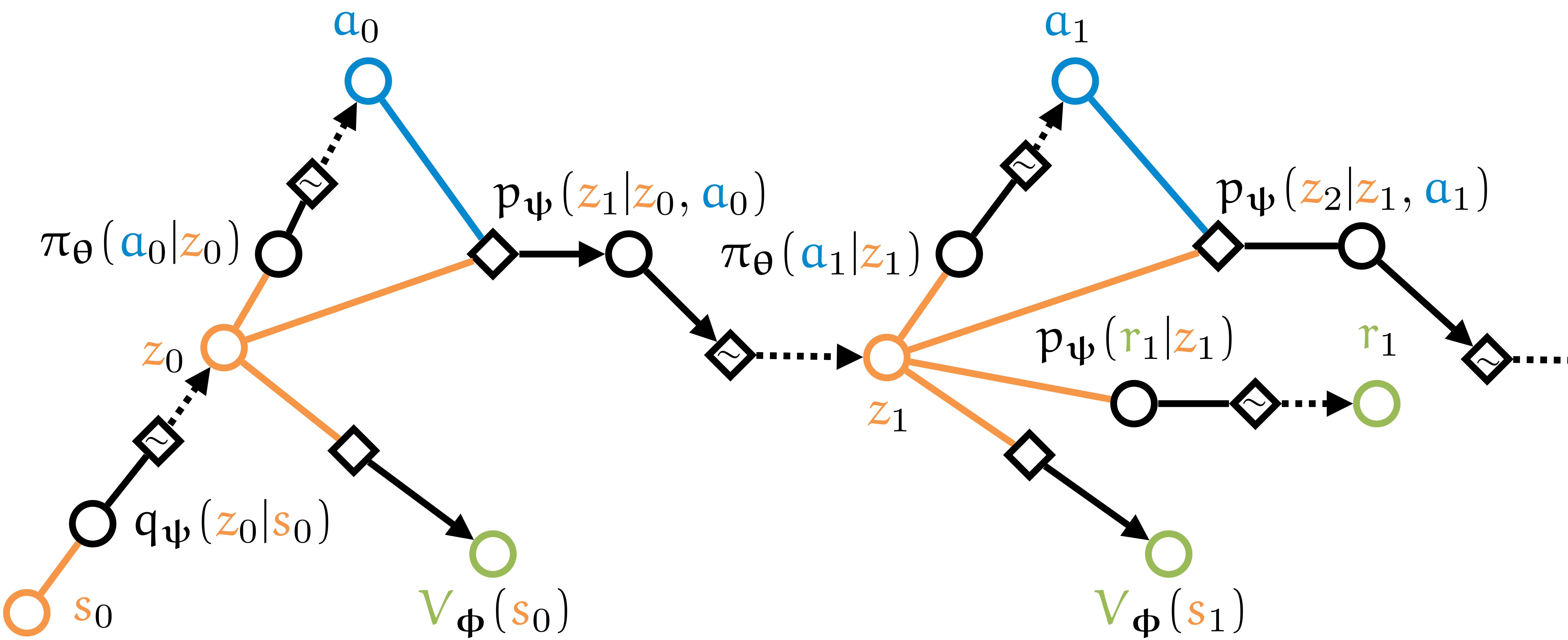


Hafner, Danijar, et al. "Dream to Control: Learning Behaviors by Latent Imagination." International Conference on Learning Representations. 2019.

DREAMING

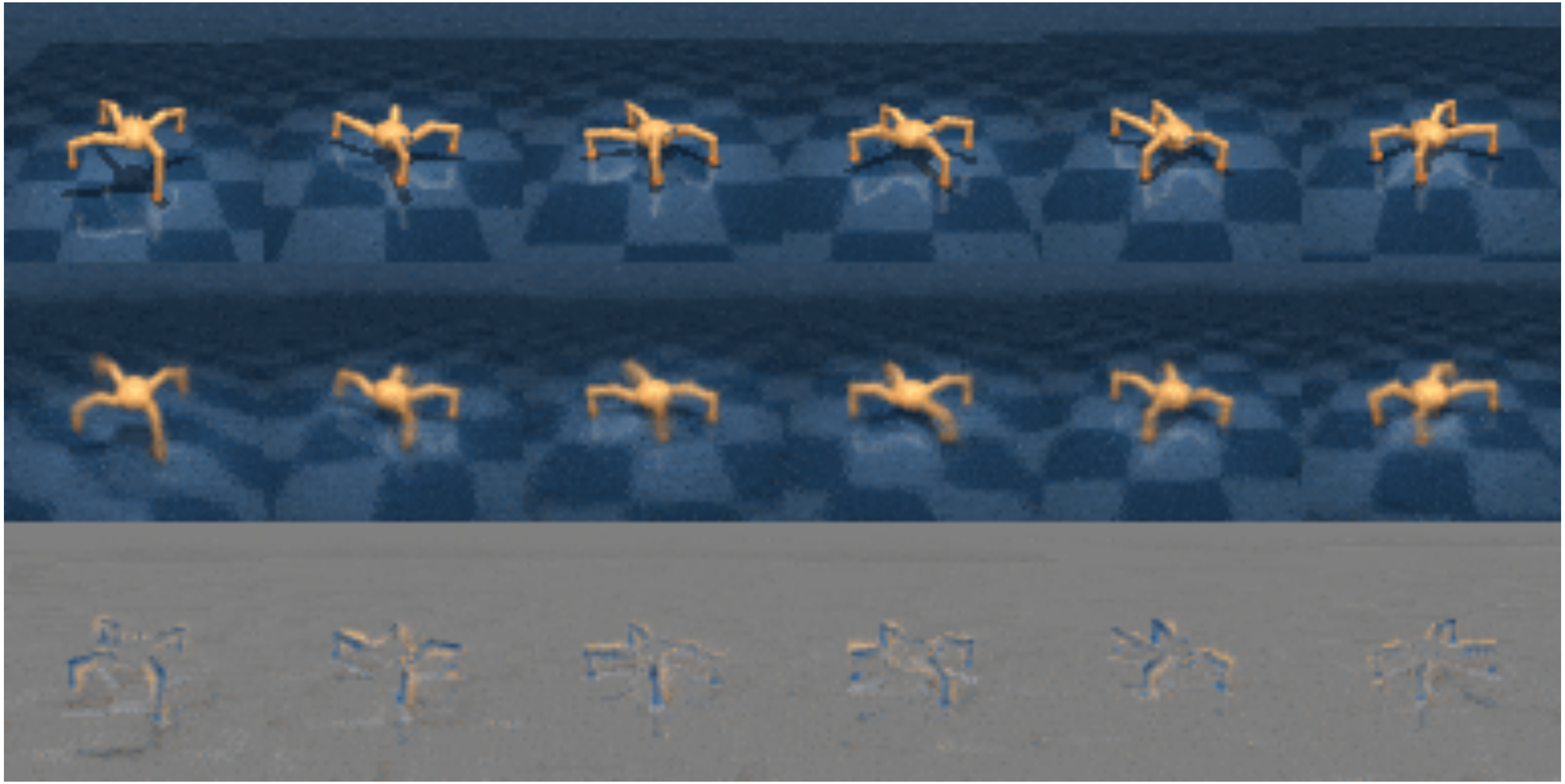
Ha, David, and Jürgen Schmidhuber. "World models."

TRAINING BY DREAMING



TRAINING BY BACKPROPAGATING THROUGH DREAM

DREAMER



Hafner, Danijar, et al. "Dream to Control: Learning Behaviors by Latent Imagination." International Conference on Learning Representations. 2019.

SUMMARY

- **REINFORCE**: Basic policy gradient algorithm
- **Actor-critic**: Add critic to reduce variance
- **TRPO/PPO**: Ensure small and controlled learning steps
- **SAC**: Use reparameterization and entropy regularization
- **World Models**: Train policy inside learned model

e.van.krieken@vu.nl

@EmilevanKrieken

THANK YOU!

e.van.krieken@vu.nl

https://github.com/HEmile/storchastic