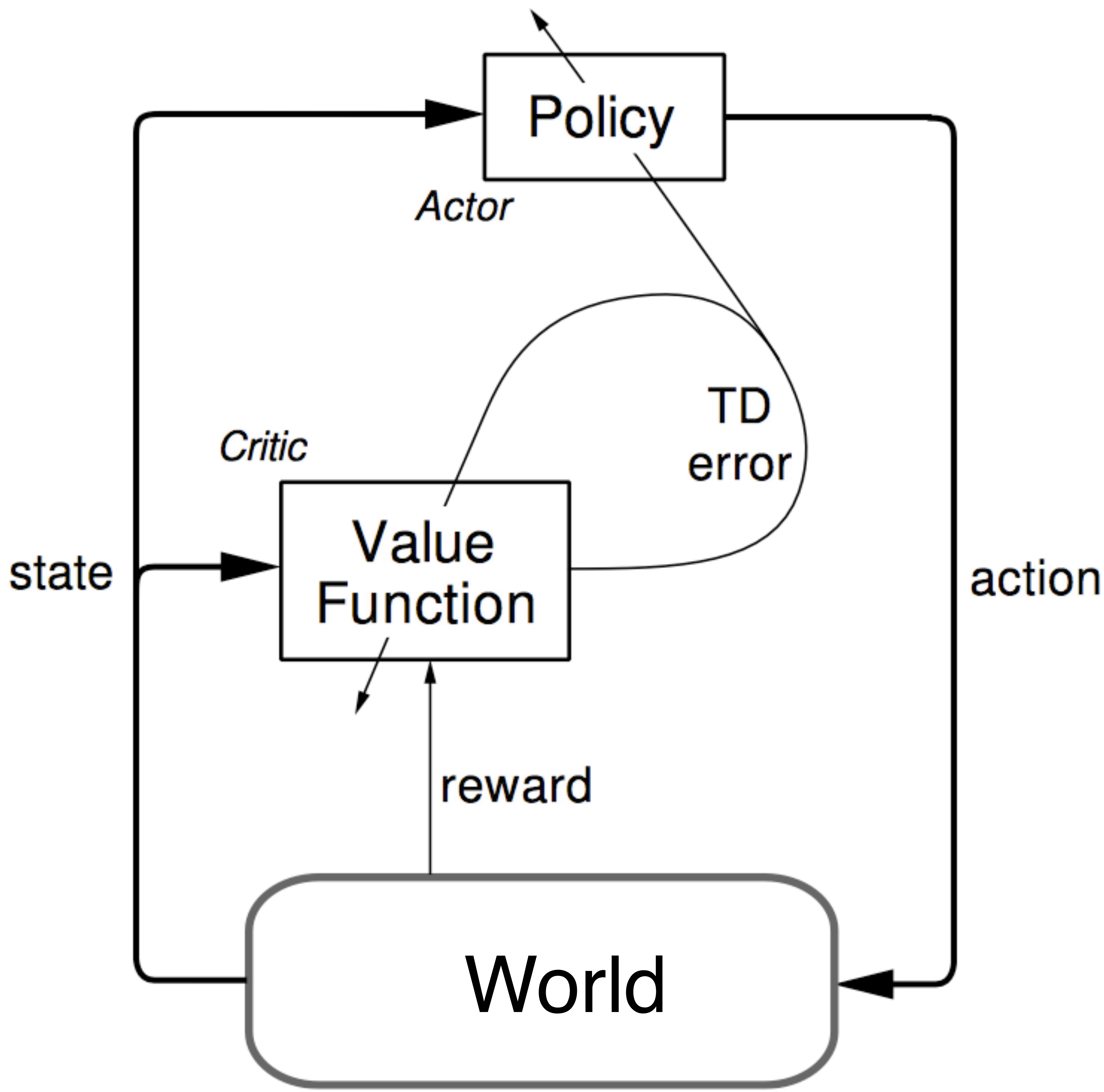
Actor-critic architecture



Actor-Critic methods

REINFORCE with baseline:

$$\boldsymbol{\theta}_{t+1} \triangleq \boldsymbol{\theta}_t + \alpha\overset{\gamma^t}{_\wedge}\Big(G_t - b(S_t)\Big)\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}$$

Actor-Critic method:

$$\begin{aligned}
\boldsymbol{\theta}_{t+1} &\triangleq \boldsymbol{\theta}_t + \alpha\overset{\gamma^t}{_\wedge}\Big(G_t^{(1)} - \hat{v}(S_t,\mathbf{w})\Big)\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \\
&= \boldsymbol{\theta}_t + \alpha\overset{\gamma^t}{_\wedge}\Big(R_{t+1}\overset{-\bar{R}_t}{_\wedge} + \gamma\hat{v}(S_{t+1},\mathbf{w}) - \hat{v}(S_t,\mathbf{w})\Big)\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}
\end{aligned}$$

A2C and A3C

- Pseudocode

for iteration$=1, 2, \ldots$ **do**

Agent acts for T timesteps (e.g., $T = 20$),

For each timestep t, compute

$$\hat{R}_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_t)$$
$$\hat{A}_t = \hat{R}_t - V(s_t)$$

$\hat{R}_t$ is target value function, in regression problem

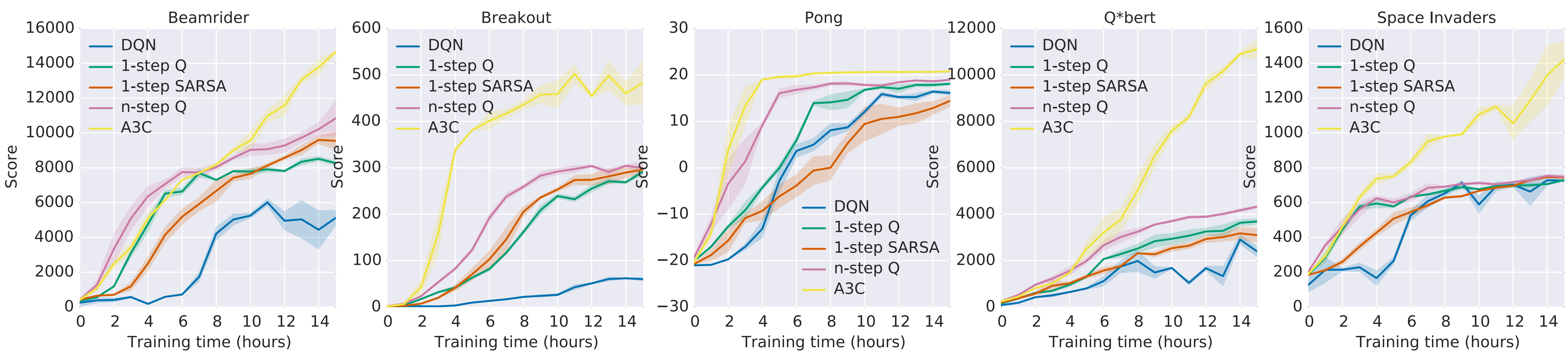
$\hat{A}_t$ is estimated advantage function

Compute loss gradient $g = \nabla_\theta \sum_{t=1}^{T} \left[-\log \pi_\theta(a_t \mid s_t) \hat{A}_t + c(V(s) - \hat{R}_t)^2 \right]$

g is plugged into a stochastic gradient descent variant, e.g., Adam.

end for

A3C results


Beamrider
Breakout
Pong
Q*bert
Space Invaders
Score
Training time (hours)
DQN
1-step Q
1-step SARSA
n-step Q
A3C

Revisiting the objective

- Let $\eta(\pi)$ denote the expected return of π
- We collect data with π_{old}. Want to optimize some objective to get a new policy π
- Define $L_{\pi_{\mathrm{old}}}(\pi)$ to be the "surrogate objective"[1]

$$L(\pi) = \mathbb{E}_{\pi_{\mathrm{old}}}\left[\frac{\pi(a \mid s)}{\pi_{\mathrm{old}}(a \mid s)} A^{\pi_{\mathrm{old}}}(s, a)\right]$$

$$\nabla_\theta L(\pi_\theta)\big|_{\theta_{\mathrm{old}}} = \nabla_\theta \eta(\pi_\theta)\big|_{\theta_{\mathrm{old}}} \quad \text{(policy gradient)}$$

- Local approximation to the performance of the policy; does not depend on parameterization of π

Trust Region Policy Optimization (TRPO)

- Constrained optimization problem

$$\max_{\pi} L(\pi), \text{ subject to } \overline{\mathrm{KL}}[\pi_{\mathrm{old}}, \pi] \leq \delta$$

$$\text{where } L(\pi) = \mathbb{E}_{\pi_{\mathrm{old}}}\left[\frac{\pi(a \mid s)}{\pi_{\mathrm{old}}(a \mid s)} A^{\pi_{\mathrm{old}}}(s, a)\right]$$

- Construct loss from empirical data

$$\hat{L}(\pi) = \sum_{n=1}^{N} \frac{\pi(a_n \mid s_n)}{\pi_{\mathrm{old}}(a_n \mid s_n)} \hat{A}_n$$

- Make quadratic approximation and solve with conjugate gradient algorithm

Proximal Policy Gradient (PPO)

- Use penalty instead of constraint

$$\underset{\theta}{\text{minimize}} \sum_{n=1}^{N} \frac{\pi_\theta(a_n \mid s_n)}{\pi_{\theta_{\mathrm{old}}}(a_n \mid s_n)} \hat{A}_n - \beta \overline{\mathrm{KL}}[\pi_{\theta_{\mathrm{old}}}, \pi_\theta]$$

- Pseudocode:

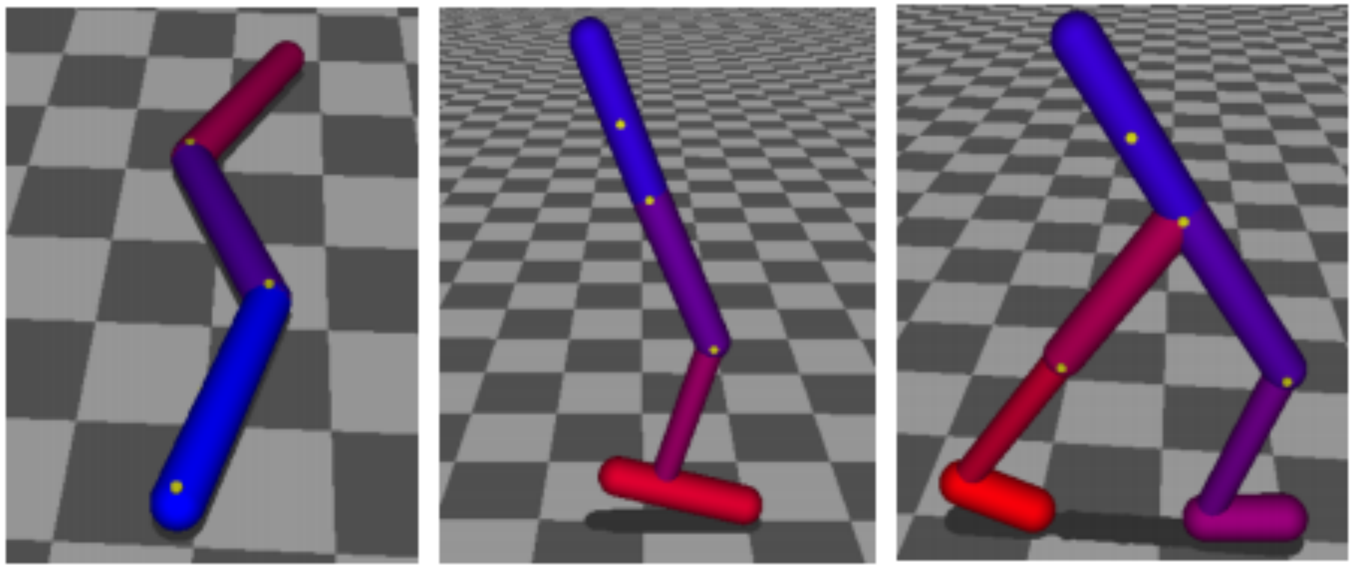
  for iteration$=1, 2, \ldots$ **do**
      Run policy for T timesteps or N trajectories
      Estimate advantage function at all timesteps
      Do SGD on above objective for some number of epochs
      If KL too high, increase β. If KL too low, decrease β.
  end for

- $\approx$ same performance as TRPO, but only first-order optimization

Results

Applied to

- Locomotion controllers in 2D



- Atari games with pixel input

Deep Deterministic Policy Gradient (DDPG)

- Incorporate replay buffer and target network ideas from DQN for increased stability
- Use lagged (Polyak-averaging) version of Q_ϕ and π_θ for fitting Q_ϕ (towards $Q^{\pi,\gamma}$) with TD(0)

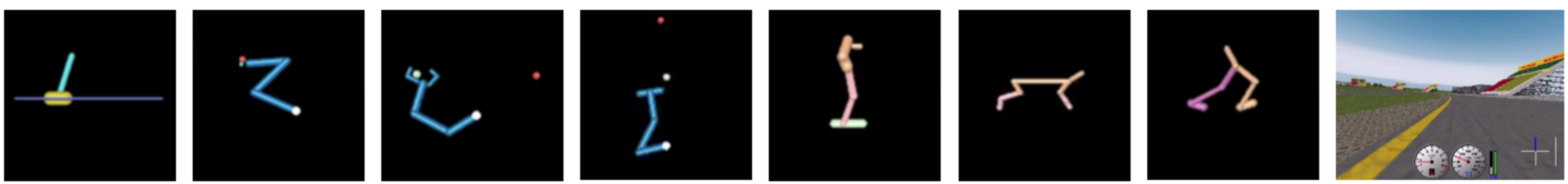
$$\hat{Q}_t = r_t + \gamma Q_{\phi'}(s_{t+1}, \pi(s_{t+1}; \theta'))$$

- Pseudocode:

for iteration$=1, 2, \ldots$ **do**
 Act for several timesteps, add data to replay buffer
 Sample minibatch
 Update π_θ using $g \propto \nabla_\theta \sum_{t=1}^{T} Q(s_t, \pi(s_t, z_t; \theta))$
 Update Q_ϕ using $g \propto \nabla_\phi \sum_{t=1}^{T} (Q_\phi(s_t, a_t) - \hat{Q}_t)^2$,
end for

DDPG results

Applied to 2D and 3D robotics tasks and driving with pixel input



The generality of policy-gradient

- Can be applied whenever we can compute the effect of parameter changes on the action probabilities,
- E.g., has been applied to spiking neuron models
- There are many possibilities other than linear-exponential and linear-gaussian, e.g., mixture of random, argmax, and fixed-width gaussian; learn the mixing weights, drift/diffusion models
- Can be applied whenever we can compute the effect of parameter changes on the action probabilities, $\nabla\pi(A_t|S_t,\boldsymbol{\theta})$