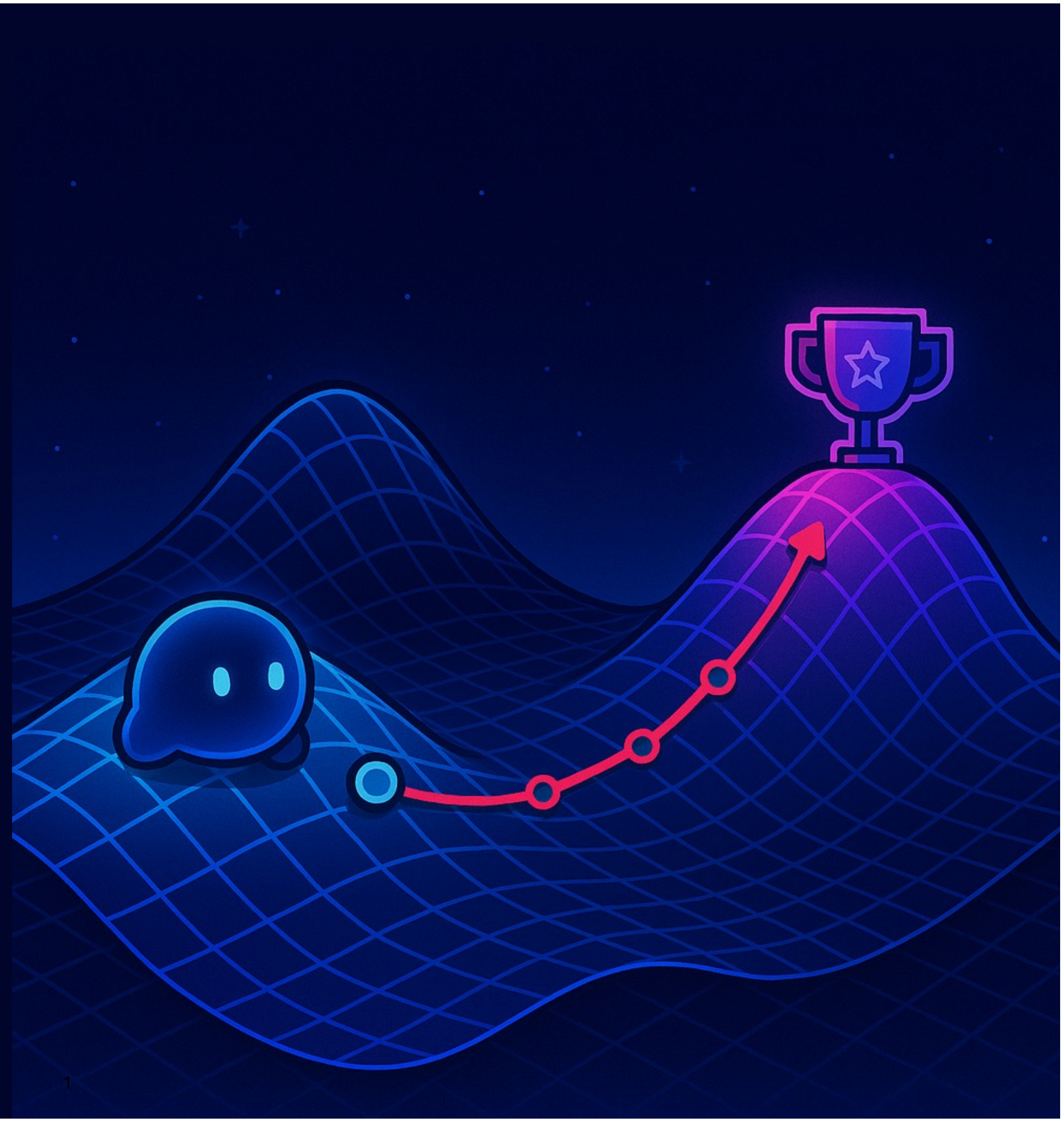
Deep Deterministic Policy Gradients (DDPG)

Lain Mustafaoglu

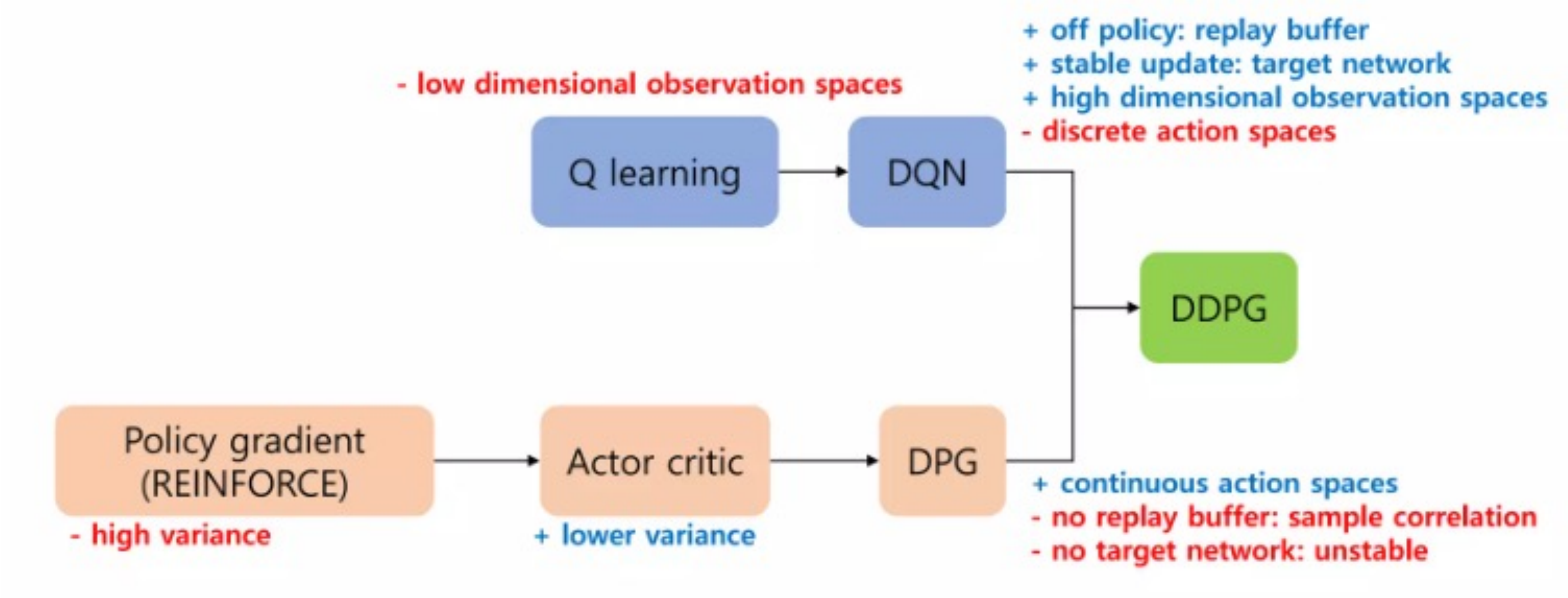
CS395T: Foundations of Machine Learning for Systems Researchers



Overview and Challenges

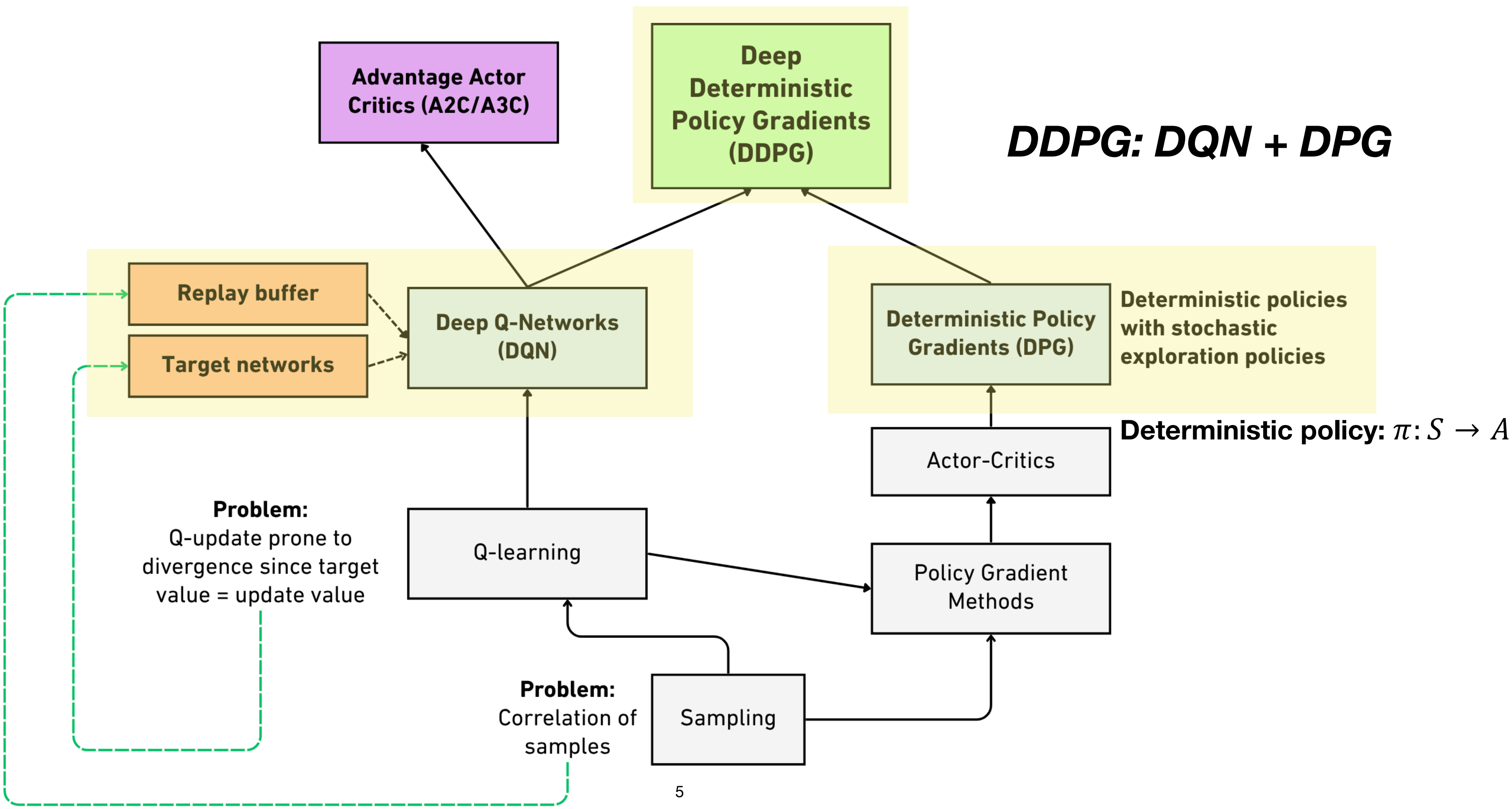
- **Previously:** discrete action spaces
  - One method we've seen so far: **deep Q-networks (DQNs)**
    - Q-table implemented with a neural network (Q-network)
    - Select action by finding $\max\limits_{a \in A} Q(s, a)$
- **Today's lecture:** moving to continuous actions
  - Two challenges with this:
    - Finding $\max_{a \in A} Q(s, a)$ to select actions
    - Computing $\max\limits_{a \in A} Q(s_{t+1}, a)$ in the update rule

DDPG: DQN + DPG



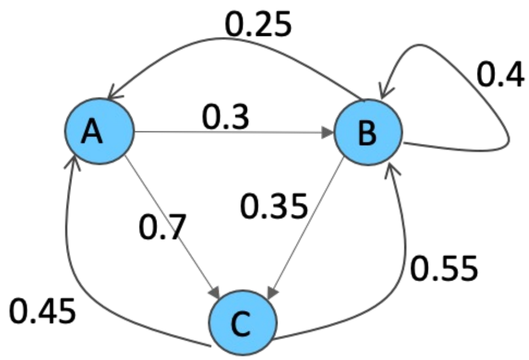
- low dimensional observation spaces
Q learning
DQN
+ off policy: replay buffer
+ stable update: target network
+ high dimensional observation spaces
- discrete action spaces
DDPG
Policy gradient
(REINFORCE)
- high variance
Actor critic
+ lower variance
DPG
+ continuous action spaces
- no replay buffer: sample correlation
- no target network: unstable

Road to DDPG



Advantage Actor Critics (A2C/A3C)
Deep Deterministic Policy Gradients (DDPG)
DDPG: DQN + DPG
Replay buffer
Target networks
Deep Q-Networks (DQN)
Deterministic Policy Gradients (DPG)
Deterministic policies with stochastic exploration policies
Deterministic policy: $\pi: S \rightarrow A$
Actor-Critics
Problem:
Q-update prone to divergence since target value = update value
Q-learning
Policy Gradient Methods
Problem:
Correlation of samples
Sampling

Stationary Distribution of Markov Process



Transition matrix T = $\begin{array}{cc} \begin{array}{ccc} \text{A} & \text{B} & \text{C} \end{array} & \\ \begin{pmatrix} 0.0 & 0.3 & 0.7 \\ 0.25 & 0.4 & 0.35 \\ 0.45 & 0.55 & 0.0 \end{pmatrix} & \begin{array}{c} \text{A} \\ \text{B} \\ \text{C} \end{array} \end{array}$

Discrete-time Markov process

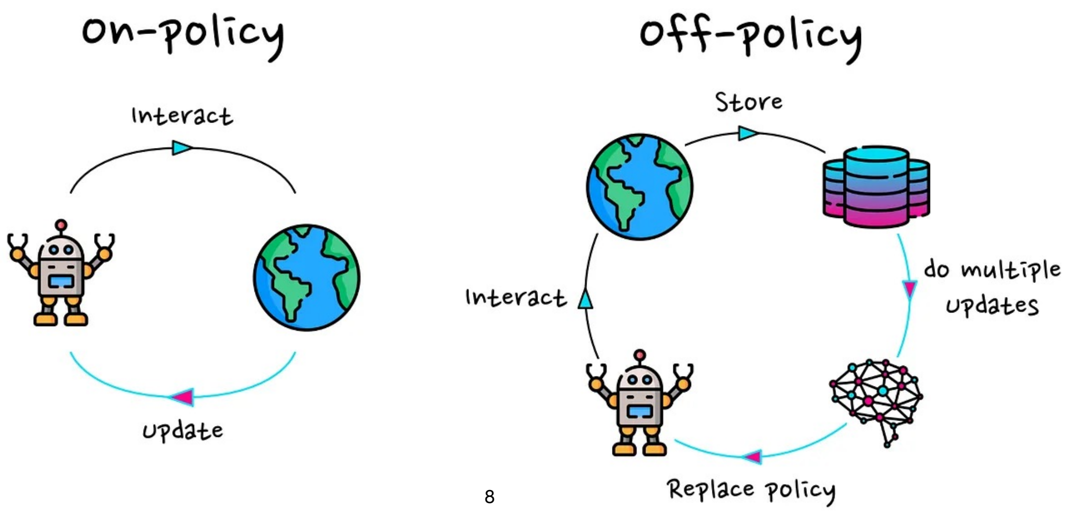
- **Stationary distribution on states ρ:** what fraction of time is spent on the average in each state?
- **Eigenvector computation:** Tx = λx for $\lambda = 1$
  - For our problem, x = $[0.253, 0.423, 0.324]^T$
- For MDPs, ρ is a function of a policy π: ρ^π
- Previous lectures: assumed unique starting state s_0
  - This lecture: agent can start in any state with probability ρ^π

On-Policy vs. Off-Policy Methods

On-Policy vs. Off-Policy

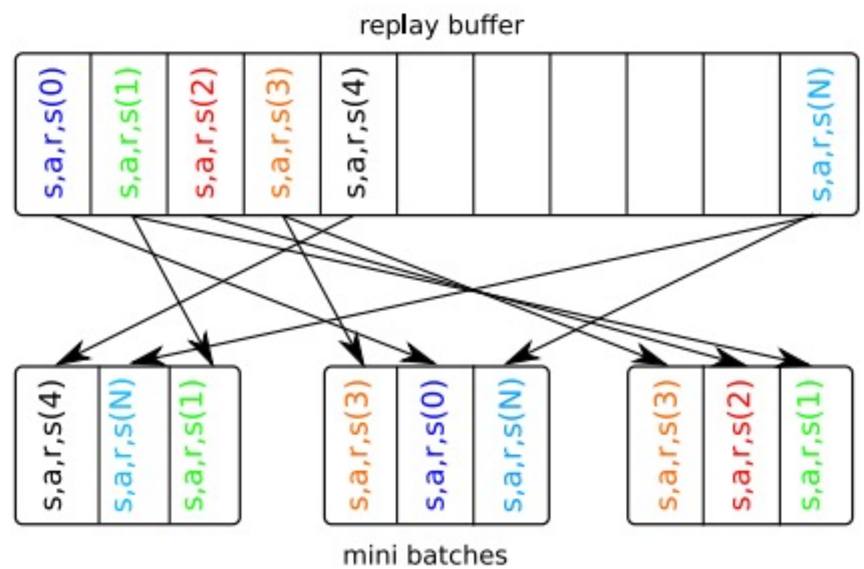
Are we learning from data collected by the *current* policy, or data from *any* policy?

- Off-policy updates learn from any transition
  - More sample efficient
  - Implemented with **experience replay buffer**



Replay Buffer (I)

- Addresses sample correlation by using samples from any past policy
- **Implementations:** First in, first out (FIFO), prioritized, etc.
- Samples processed in mini batches



Replay Buffer (II): Importance Sampling

- How do we optimize a policy using samples collected by another policy?

- **One method:** importance sampling (recall Monte Carlo lecture)

$$\mathbb{E}_{s\sim\rho^{\pi},\, a\sim\pi(\cdot|s)}\big[f(s,a)\big] = \mathbb{E}_{s\sim\rho^{\beta},\, a\sim\beta(\cdot|s)}\left[\underbrace{\frac{\rho^{\pi}(s)}{\rho^{\beta}(s)} \cdot \frac{\pi(a \mid s)}{\beta(a \mid s)}}_{\text{Importance sampling ratios}} f(s,a)\right]$$

Target Networks

- Bootstrapping with TD targets from off-policy buffer: target uses own network's predictions at next state
  - **Problem:** moving target makes updates unstable!

- **Solution:** Target networks $Q_{\phi'}$ and $\mu_{\theta'}$
  - Keep lagging copy of actor and critic networks and update periodically
  - Eliminates "high-frequency noise" in updates and promotes stability

Deterministic Policy Gradient Theorem

Policy gradient (stochastic policies):

$\pi: S \to P(A)$

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

- π_θ: stochastic policy

Deterministic off-policy policy gradient:

$\pi: S \to A$

$$\nabla_\theta J_\beta(\mu_\theta) = \mathbb{E}_{s\sim\rho^\beta}\left[\nabla_\theta \mu_\theta(s) \left.\nabla_a Q^\mu(s,a)\right|_{a=\mu_\theta(s)}\right]$$

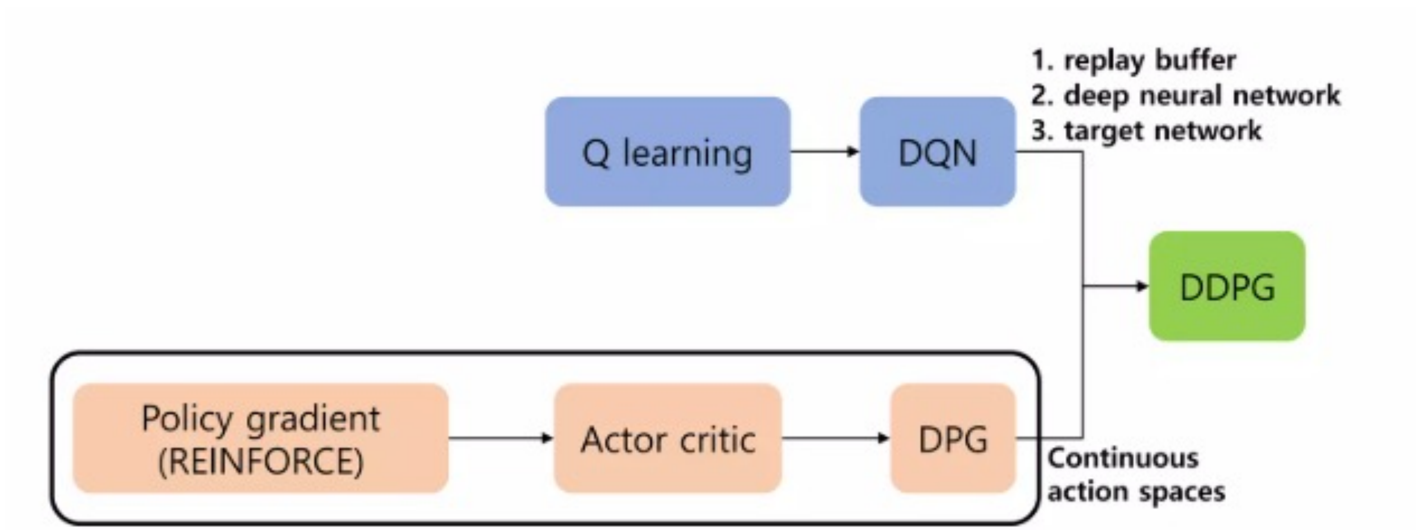
Transitions generated by behavior policy β

- β: stochastic behavior policy
- μ_θ: deterministic target policy
- ρ^π and ρ^β: state visitation distributions under π and β

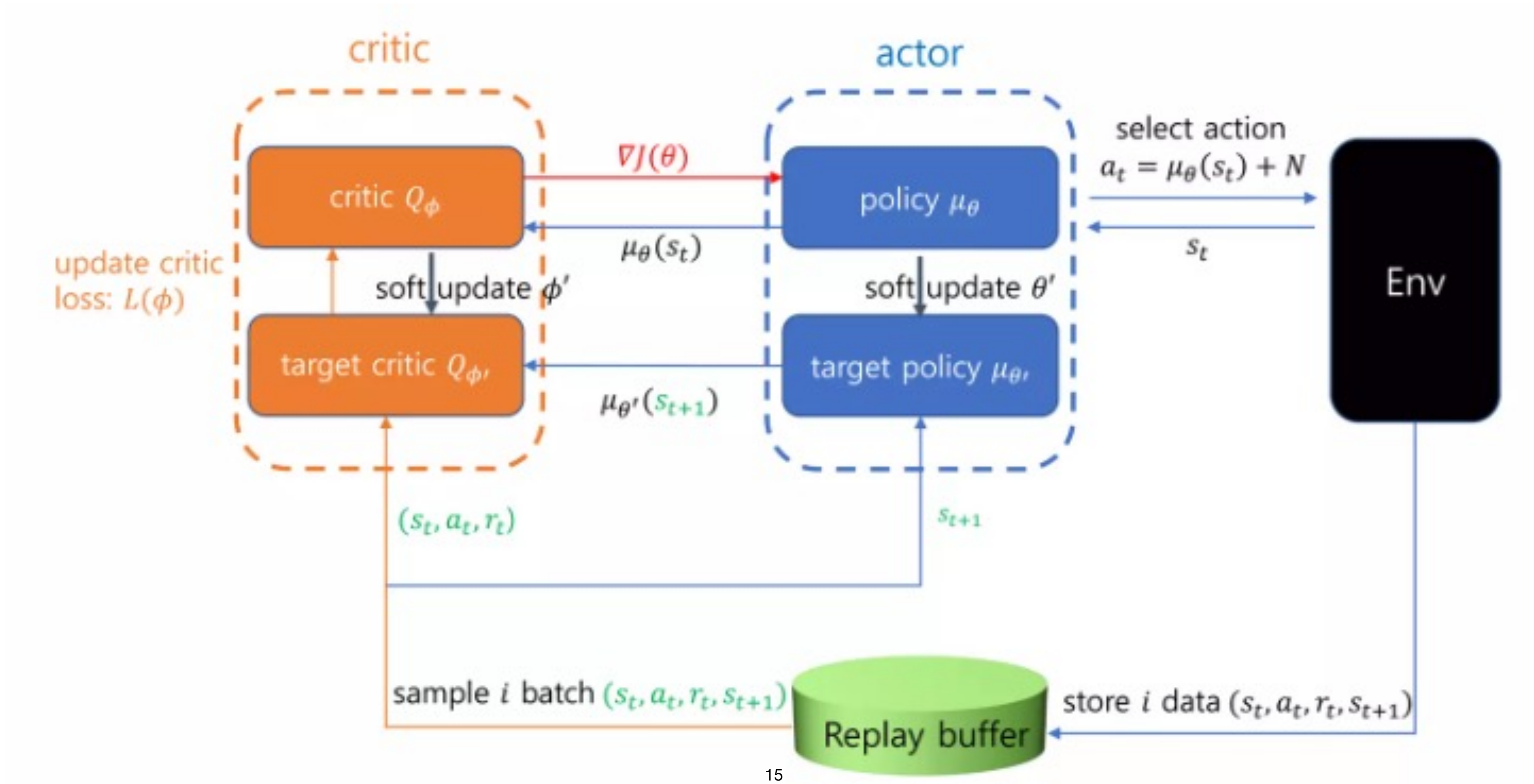
Deep Deterministic Policy Gradient (DDPG)

Unifies **DQN-style off-policy learning** and **deterministic policy gradients** (Silver et al. 2014) for continuous control

- **Key idea:** learn deterministic policy while choosing actions with stochastic behavior policy
- **Exploration policy with noise:** $a_t = \mu_{theta}(s_t) + \mathcal{N}$



DDPG



critic
actor
critic Q_ϕ
$\nabla J(\theta)$
policy μ_θ
select action
$a_t = \mu_\theta(s_t) + N$
$\mu_\theta(s_t)$
s_t
update critic
loss: $L(\phi)$
soft update ϕ'
soft update θ'
Env
target critic $Q_{\phi'}$
target policy $\mu_{\theta'}$
$\mu_{\theta'}(s_{t+1})$
(s_t, a_t, r_t)
s_{t+1}
sample i batch (s_t, a_t, r_t, s_{t+1})
Replay buffer
store i data (s_t, a_t, r_t, s_{t+1})

Algorithm

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**

Add noise for exploration
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise

Replay buffer
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R

Critic update
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$

Actor update
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

Target network update
    Update the target networks:

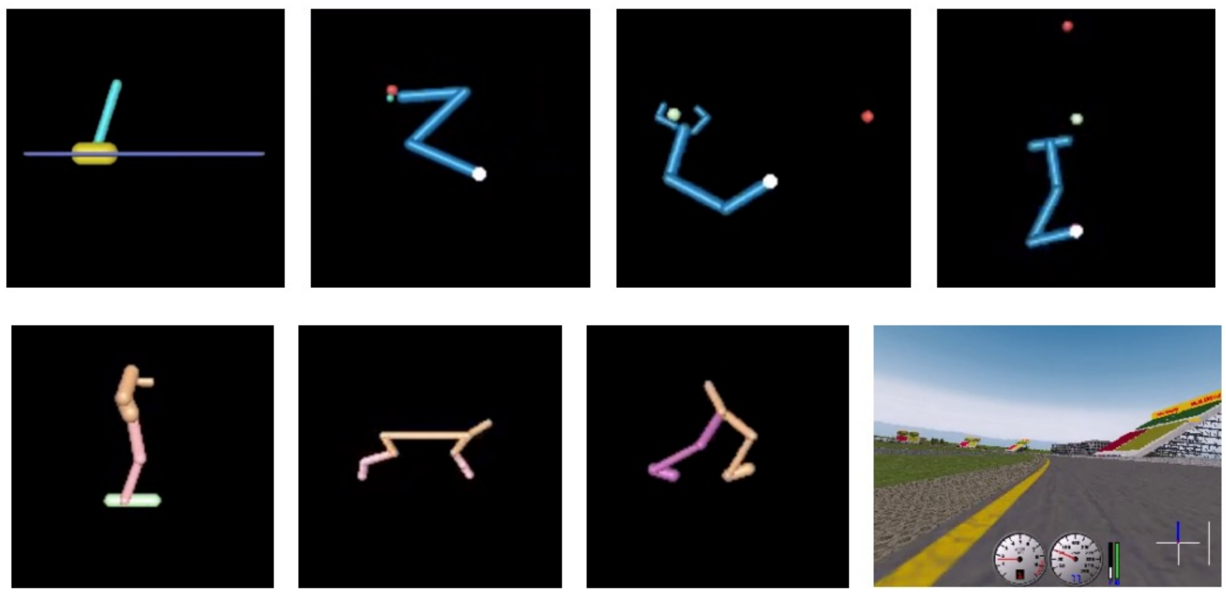
$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

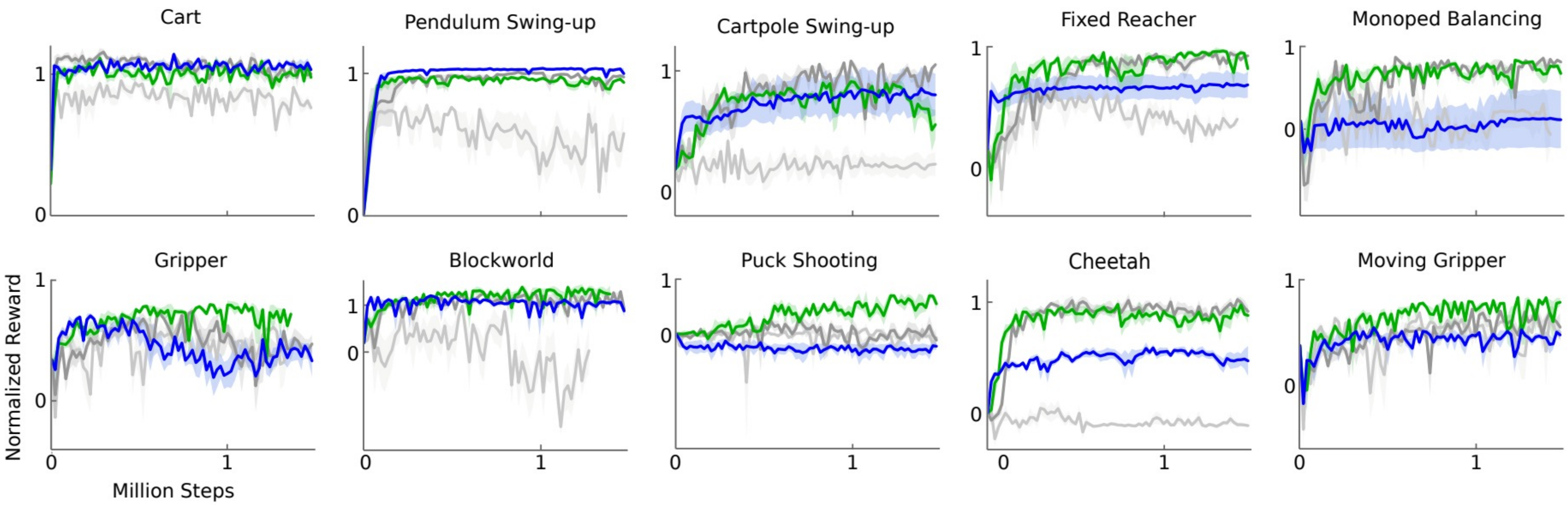
  end for
end for

Experiments

MuJoCo and Torcs for continuous control experiments



Results



Light Grey: Original DPG
Dark Grey: Target Network
Green: Target Network + Batch Norm
Blue: Target Network from pixel-only inputs

- Sample efficiency of a factor of 20 vs. DQN!

Key Takeaways

Method	Policy	Exploration	Off-policy
DQN	Greedy	Epsilon-greedy	Yes (replay buffer, FIFO)
DPG	Deterministic	Stochastic behavior policy	No
DDPG	Deterministic	Stochastic behavior policy	Yes (replay buffer, FIFO)
A2C/A3C	Stochastic	Epsilon-greedy with multiple learners + entropy	No

Any questions?