CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING (DDPG)

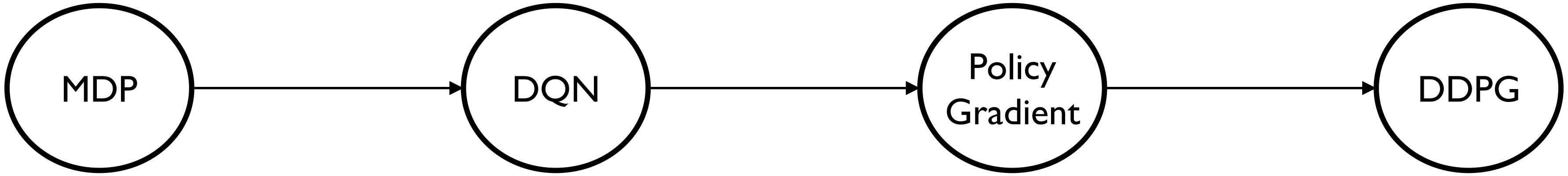
SEONVIN CHO

INFORMATION AND INTELLIGENCE SYSTEMS LAB.

ELECTRONIC ENGINEERING, HANYANG UNIVERSITY

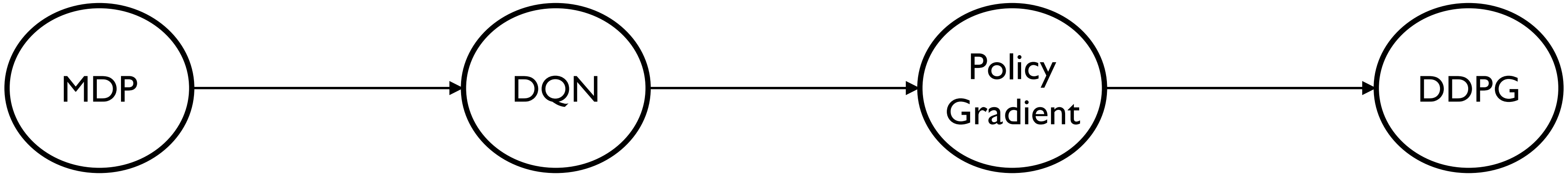
March 11, 2025

INTRO



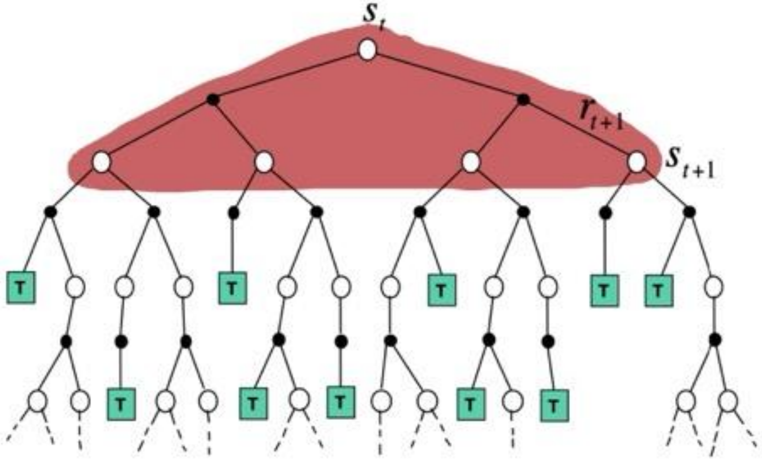
- **Dynamic programming**
  - instead of directly solving MDP is for the ease to solve difficult problems.
  - splits and solves problems only when we know dynamics / transition probabilities.
- **Monte Carlo**
  - deals with all samples → unbiased sample & high variance.
- **TD learning**
  - uses only comparison with next one → biased sample & low variance.

INTRO



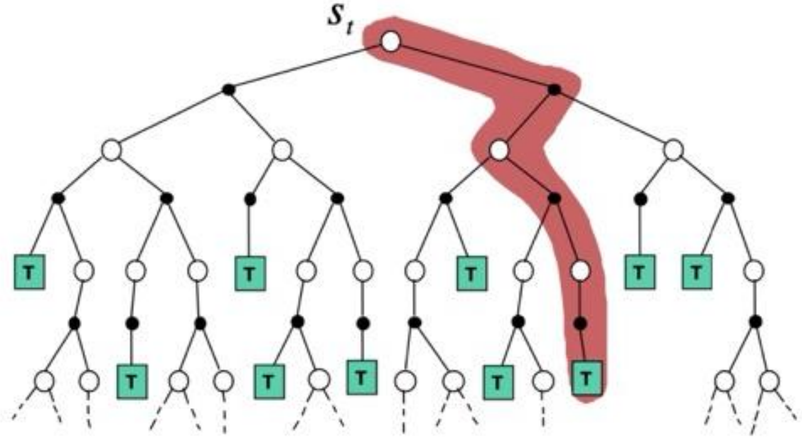
- **Dynamic Programming**

$$V(S_t) \leftarrow \mathbb{E}_\pi\left[R_{t+1} + \gamma V(S_{t+1})\right]$$



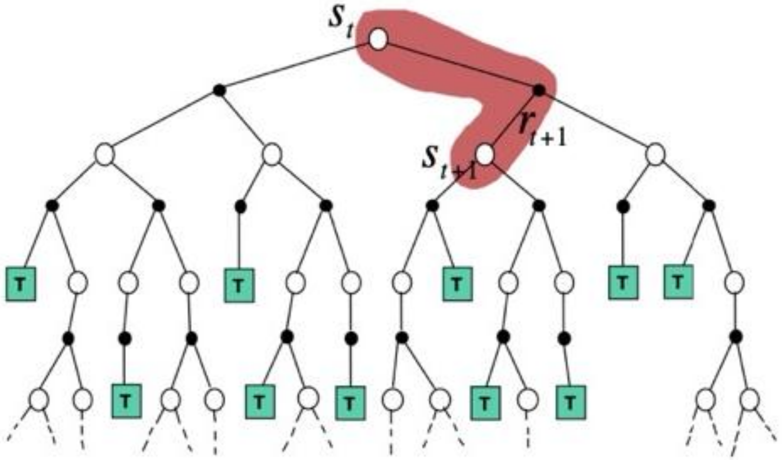
- **Monte Carlo**

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

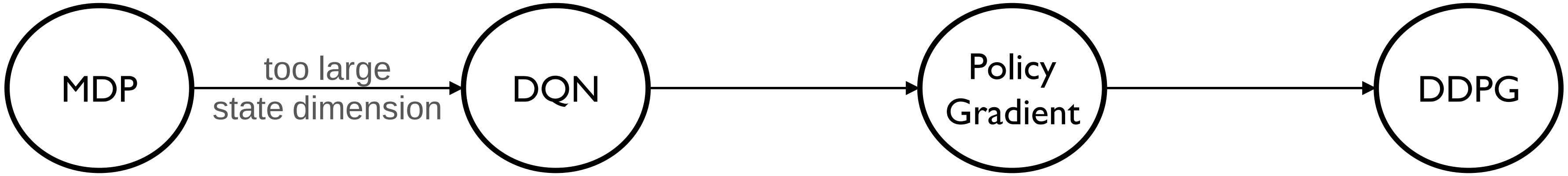


- **TD learning**

$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$

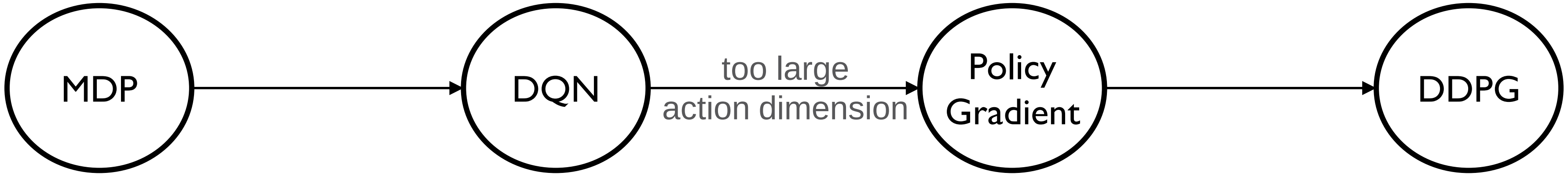


INTRO



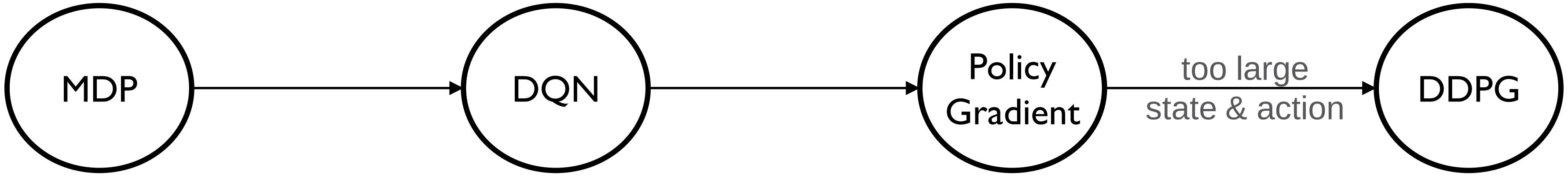
- **Deep Q-Network = Q-learning + Deep Neural Network**
  - Estimates the action-value function = Value-based method
  - Handles cases where the state dimension is large
    - while dealing with discrete actions in continuous state spaces.
  - Uses separate networks to mitigate value fluctuation.
  - Still solves the "argmax Q" problem when updating weights
    - Difficult to search for the optimal value if the function is nonlinear.

INTRO



- **Policy Gradient**
  - Estimates the parameterized policy = Policy-based method
  - Handles cases where the action dimension is large
    - while dealing with discrete states in continuous action spaces.
  - Uses gradients to directly optimize a parameterized policy
    - without considering whether the action space is continuous or discrete.
  - Computes returns and gradient samples after all episodes have ended.
    - unbiased and high variance

Intro



- **Deep Deterministic Policy Gradient**

  = DQN + Deterministic + Policy Gradient

  – Handles cases where both the state and action dimensions are large

  › while dealing with continuous state and action spaces.

  – Uses an Actor-Critic approach

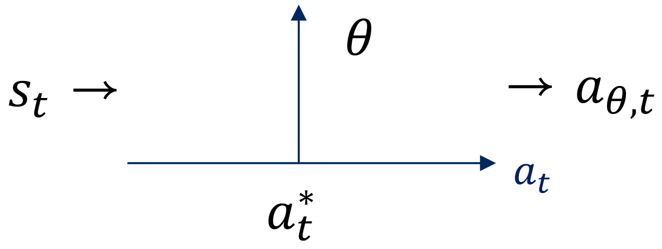
  › Employs a deterministic policy for continuous action spaces.

  – Off-policy learning

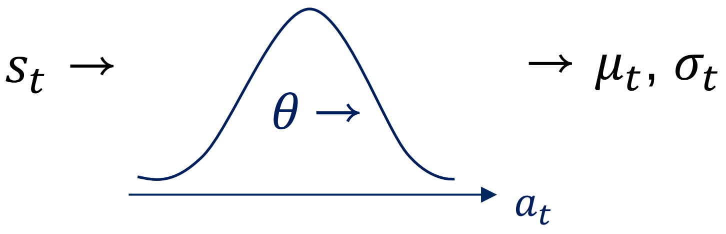
  › Utilize large replay buffers to leverage learning from uncorrelated transitions.

Deterministic Policy Gradient

- **Deterministic Policy**
  - parameterized action
  - $p(a_t|s_t) = \delta(a_t - a_t^*)$
    - delta function for only one optimal action

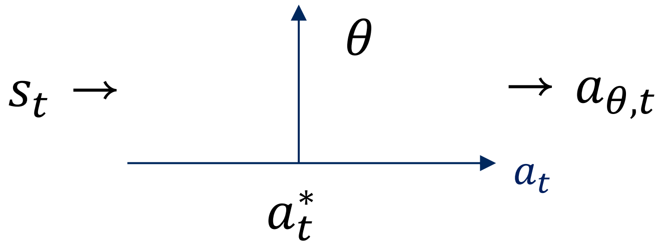


- **Stochastic Policy**
  - parameterized policy
  - $p_\theta(a_t|s_t)$
    - probability distribution for the action



DETERMINISTIC POLICY GRADIENT

- **Deterministic Policy**
  - parameterized action
  - $p(a_t|s_t) = \delta(a_t - a_t^*)$
    - delta function for only one optimal action



- **Policy Gradient Theorem**
  - $\nabla_\theta J(\theta) = \mathbb{E}_{s,a\sim\pi}[\nabla_\theta \log \pi_\theta(a|s) \cdot Q^\pi(s,a)]$

- **Deterministic Policy Gradient (DPG)**
  - $\nabla_\theta J(\theta) = \mathbb{E}_{s,a\sim\pi}[\nabla_\theta \mu_\theta(s) \cdot \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}]$

DETERMINISTIC POLICY GRADIENT (PROOF)

$$J_\theta = \mathbb{E}\{G_0\} = \int_{s_0:a_\infty} G_0\, P(s_0:a_\infty)\, ds_0:a_\infty = \int_{s_0}\int_{a_0:a_\infty} G_0\, P(a_0:a_\infty \mid s_0)\, da_0:a_\infty\, P(s_0)\, ds_0 = \int_{s_0} V(s_0)\, P(s_0)\, ds_0$$

$$V(s_0) = \int_{a_0:a_\infty} G_0\, P(a_0:a_\infty \mid s_0)\, da_0:a_\infty = \int_{a_0}\int_{s_1:a_\infty} G_0\, P(s_1:a_\infty \mid s_0, a_0)\, P(a_0 \mid s_0)\, ds_1:a_\infty\, da_0$$

$$= \int_{a_0} Q(s_0, a_0)\, \underbrace{P(a_0 \mid s_0)}_{=\delta(a_0 - a_{\theta,0})}\, da_0 = Q(s_0, a_{\theta,0})$$

$$Q(s_0, a_{\theta,0}) = \int_{s_1:a_{\theta,\infty}} (R_0 + \gamma G_1)\, P(s_1:a_\infty)\, ds_1:a_\infty = \int_{s_1, a_{\theta,1}}\int_{s_2:a_{\theta,\infty}} (R_0 + \gamma G_1)\, P(s_2:a_{\theta,\infty} \mid s_1, a_1)\, \underbrace{P(a_1 \mid s_1)}_{=\delta(a_1 - a_{\theta,1})}\, P(s_1 \mid s_0, a_{\theta,0})\, ds_2:a_{\theta,\infty}\, ds_1, a_{\theta,1}$$

$$= \int_{s_1, a_{\theta,1}} (R_0 + \gamma Q(s_1, a_1))\, P(a_{\theta,1} \mid s_1)\, P(s_1 \mid s_0, a_{\theta,0})\, ds_1, a_1$$

$a_{\theta,1}$

s_0, a_0의 함수로 정의

action 즉시 reward 받음 → next state $\quad s_0 \to a_0 \to r_0 \to s_1 \to \cdots$

$$= R_0 + \int_{s_1} \gamma Q(s_1, a_{\theta,1})\, P(s_1 \mid s_0, a_{\theta,0})\, ds_1$$

$s_0, a_{\theta,0}$

$$\nabla_\theta J_\theta = \int_{s_0} \nabla_\theta V(s_0)\, P(s_0)\, ds_0 = \int_{s_0} \nabla_\theta Q(s_0, a_{\theta,0})\, P(s_0)\, ds_0$$

$$\nabla_\theta Q(s_0, a_{\theta,0}) = \nabla_{a_{\theta,0}} R_0 \cdot a_{\theta,0}' + \int_{s_1} \gamma Q(s_1, a_{\theta,1})\, \nabla_\theta P(s_1 \mid s_0, a_{\theta,0})\, a_{\theta,0}'\, ds_1 + \int_{s_1} \gamma \nabla_\theta Q(s_1, a_{\theta,1})\, P(s_1 \mid s_0, a_{\theta,0})\, ds_1$$

$$= a_{\theta,0}' \cdot \nabla_{a_{\theta,0}} \left(R_0 + \int_{s_1} \gamma Q(s_1, a_{\theta,1})\, P(s_1 \mid s_0, a_{\theta,0})\, ds_1\right) + \int_{s_1} \gamma \nabla_\theta Q(s_1, a_{\theta,1})\, P(s_1 \mid s_0, a_{\theta,0})\, ds_1$$

$$= a_{\theta,0}' \cdot \nabla_{a_{\theta,0}} Q(s_0, a_{\theta,0}) + \int_{s_1} \gamma \nabla_\theta Q(s_1, a_{\theta,1})\, P(s_1 \mid s_0, a_{\theta,0})\, ds_1$$

$$\nabla_\theta J_\theta = \int_{s_0} a_{\theta,0}' \cdot \nabla_{a_{\theta,0}} Q(s_0, a_{\theta,0})\, P(s_0)\, ds_0 + \int_{s_0}\int_{s_1} \gamma a_{\theta,1}'\, \nabla_{a_{\theta,1}} Q(s_1, a_{\theta,1})\, P(s_1 \mid s_0, a_{\theta,0})\, ds_1\, P(s_0)\, ds_0 + \cdots$$

$$\cong \sum_{t=0}^{\infty} \int_{s_0:s_t} \boxed{\gamma^t a_{\theta,t}'\, \nabla_{a_{\theta,t}} Q_w(s_t, a_{\theta,t})}\, P(s_0) \prod_{k=1}^{t} P(s_k \mid s_{k-1}, a_{\theta,k-1})\, ds_0:s_t$$

$t\uparrow \to$ 특정값 $\approx 0 \to \gamma^t$ 제거

Off-policy ∴ correlation↓

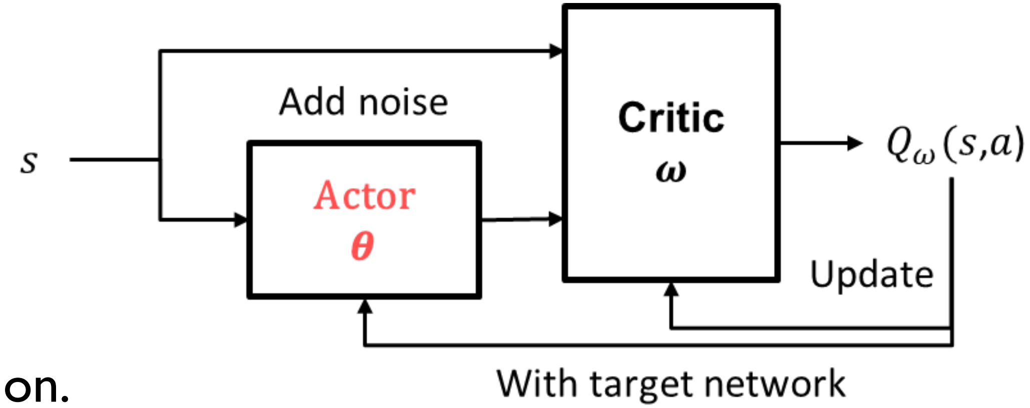
actor-critic

→ replay buffer → sample → update

$$= \sum_{t=0}^{\infty} \int_{s_0:s_t} \nabla_\theta Q_w(s_t, a_{\theta,t})\, P(s_0) \prod_{k=1}^{t} P(s_k \mid s_{k-1}, a_{\theta,k-1})\, ds_0:s_t$$

DDPG

- **DQN +Deterministic Policy Gradient**
  - parameterized action
  - $p(a_t|s_t) = \delta(a_t - a_t^*)$
    - Suitable for continuous action spaces.
    - Limited exploration
      - → Requires Gaussian noise for better exploration.



- Actor update based on the Q-value gradient.
- Critic update using TD error.

DDPG

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**
    noise for exploration
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    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
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    uniform random sample for replay buffer
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:
    uniform random sample for replay buffer

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

    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

DDPG

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**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    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
    Set $y_i = r_i + \gamma \boxed{Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})}$ target network
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - \boxed{Q(s_i, a_i|\theta^Q)})^2$ main network
    Update the actor policy using the sampled policy gradient:

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

    Update the target networks:

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

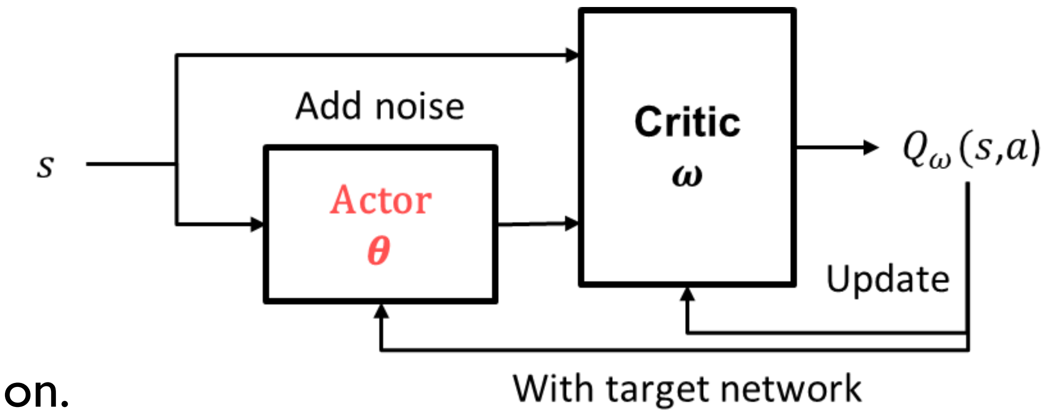
  end for
end for

separate networks similar to DQN!

DDPG

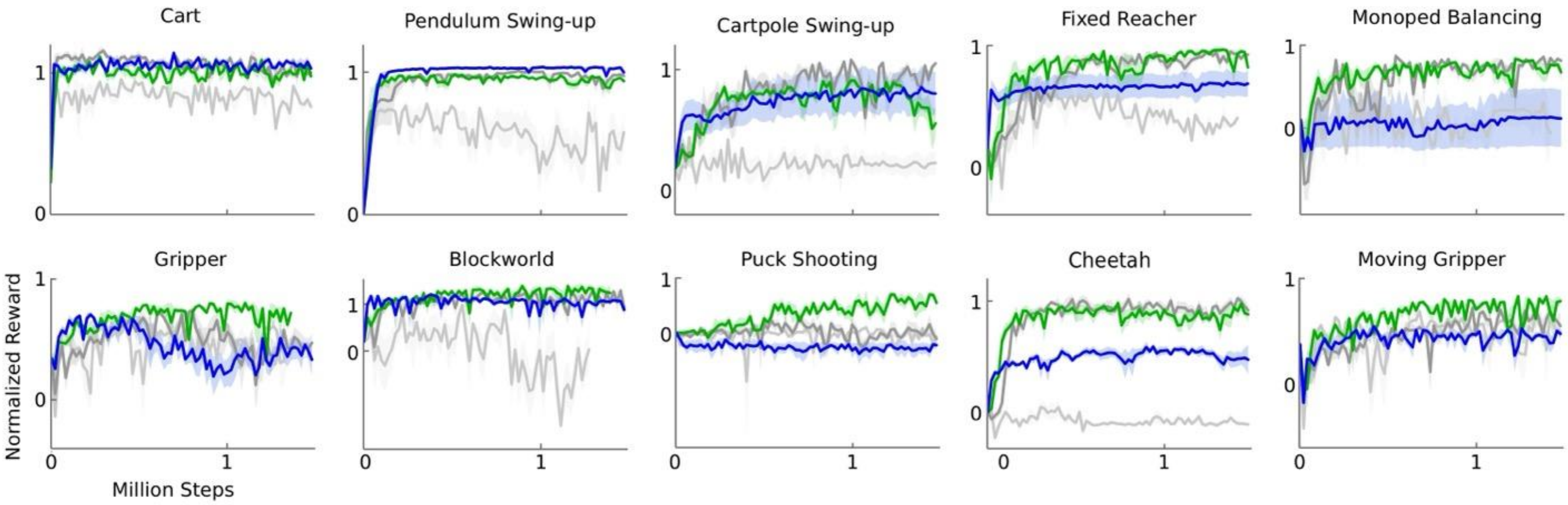
- **DQN +Deterministic Policy Gradient**
  - parameterized action
  - $p(a_t|s_t) = \delta(a_t - a_t^*)$
    - Suitable for continuous action spaces.
    - Limited exploration

      → Requires gaussian noise for better exploration.



- Actor update based on the Q-value gradient.
- Critic update using TD error.

RESULTS

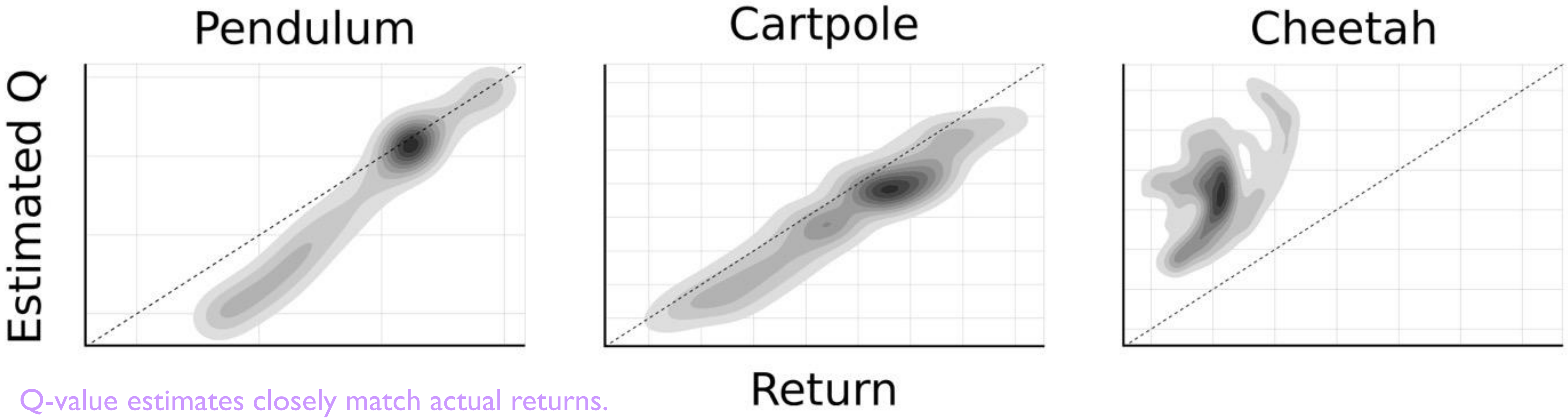


light grey = original DPG algorithm (minibatch NFQCA) with batch normalization
dark grey = original DPG algorithm with target network
green = original DPG algorithm with target networks and batch normalization
blue = original DPG algorithm with target networks from pixel-only inputs

Target networks are crucial!

RESULTS

Q-value estimates deviate more from actual returns.
Value estimation is more challenging in complex environments.



Q-value estimates closely match actual returns.
Accurate value estimation in simple environments.

Dashed line: Ideal case where Q-values match actual returns
Darker regions: More frequent Q-value and return pairs

SUMMARY

DDPG = DQN +Deterministic Policy + Policy Gradient Theorem

- DQN : off-policy & separate networks
  - Off-policy : target policy ≠ behavior policy
    - Uses a replay buffer to maximize sample efficiency.
  - Separate networks
    - Uses target network and main network for stationarity.
- Actor-Critic approach
  - Actor updates the policy network using the policy gradient.
  - Critic updates the Q-value function using TD targets with a target network.
- Gaussian Noise for exploration
- Soft target update (τ) to slowly update the target network