# Plan for today

- Finish DQN
- Briefly talk about Rainbow DQN
- DDPG
- BCQ
- Breakout room discussion

# DDPG, BCQ

Saurabh Gupta

# DDPG

$$\rho^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t P(s_t = s \mid \pi)$$

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma\, \mathbb{E}_{a_{t+1} \sim \pi}\left[Q^{\pi}(s_{t+1}, a_{t+1})\right]\right]$$

$$\begin{aligned}\nabla_{\theta^{\mu}} J &\approx \mathbb{E}_{s_t \sim \rho^{\beta}}\left[\nabla_{\theta^{\mu}} Q(s, a|\theta^{Q})|_{s=s_t, a=\mu(s_t|\theta^{\mu})}\right] \\ &= \mathbb{E}_{s_t \sim \rho^{\beta}}\left[\nabla_{a} Q(s, a|\theta^{Q})|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_{\mu}} \mu(s|\theta^{\mu})|_{s=s_t}\right]\end{aligned}$$

# DDPG

**Algorithm 1** DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ 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 $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    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**

# DDPG

- Replay buffer
- Target network
- Co-related noise

# DDPG

| environment | $R_{av,lowd}$ | $R_{best,lowd}$ | $R_{av,pix}$ | $R_{best,pix}$ | $R_{av,cntrl}$ | $R_{best,cntrl}$ |
|---|---|---|---|---|---|---|
| blockworld1 | 1.156 | 1.511 | 0.466 | 1.299 | -0.080 | 1.260 |
| blockworld3da | 0.340 | 0.705 | 0.889 | 2.225 | -0.139 | 0.658 |
| canada | 0.303 | 1.735 | 0.176 | 0.688 | 0.125 | 1.157 |
| canada2d | 0.400 | 0.978 | -0.285 | 0.119 | -0.045 | 0.701 |
| cart | 0.938 | 1.336 | 1.096 | 1.258 | 0.343 | 1.216 |
| cartpole | 0.844 | 1.115 | 0.482 | 1.138 | 0.244 | 0.755 |
| cartpoleBalance | 0.951 | 1.000 | 0.335 | 0.996 | -0.468 | 0.528 |
| cartpoleParallelDouble | 0.549 | 0.900 | 0.188 | 0.323 | 0.197 | 0.572 |
| cartpoleSerialDouble | 0.272 | 0.719 | 0.195 | 0.642 | 0.143 | 0.701 |
| cartpoleSerialTriple | 0.736 | 0.946 | 0.412 | 0.427 | 0.583 | 0.942 |
| cheetah | 0.903 | 1.206 | 0.457 | 0.792 | -0.008 | 0.425 |
| fixedReacher | 0.849 | 1.021 | 0.693 | 0.981 | 0.259 | 0.927 |
| fixedReacherDouble | 0.924 | 0.996 | 0.872 | 0.943 | 0.290 | 0.995 |
| fixedReacherSingle | 0.954 | 1.000 | 0.827 | 0.995 | 0.620 | 0.999 |
| gripper | 0.655 | 0.972 | 0.406 | 0.790 | 0.461 | 0.816 |
| gripperRandom | 0.618 | 0.937 | 0.082 | 0.791 | 0.557 | 0.808 |
| hardCheetah | 1.311 | 1.990 | 1.204 | 1.431 | -0.031 | 1.411 |
| hopper | 0.676 | 0.936 | 0.112 | 0.924 | 0.078 | 0.917 |
| hyq | 0.416 | 0.722 | 0.234 | 0.672 | 0.198 | 0.618 |
| movingGripper | 0.474 | 0.936 | 0.480 | 0.644 | 0.416 | 0.805 |
| pendulum | 0.946 | 1.021 | 0.663 | 1.055 | 0.099 | 0.951 |
| reacher | 0.720 | 0.987 | 0.194 | 0.878 | 0.231 | 0.953 |
| reacher3daFixedTarget | 0.585 | 0.943 | 0.453 | 0.922 | 0.204 | 0.631 |
| reacher3daRandomTarget | 0.467 | 0.739 | 0.374 | 0.735 | -0.046 | 0.158 |
| reacherSingle | 0.981 | 1.102 | 1.000 | 1.083 | 1.010 | 1.083 |
| walker2d | 0.705 | 1.573 | 0.944 | 1.476 | 0.393 | 1.397 |
| torcs | -393.385 | 1840.036 | -401.911 | 1876.284 | -911.034 | 1961.600 |

# DDPG

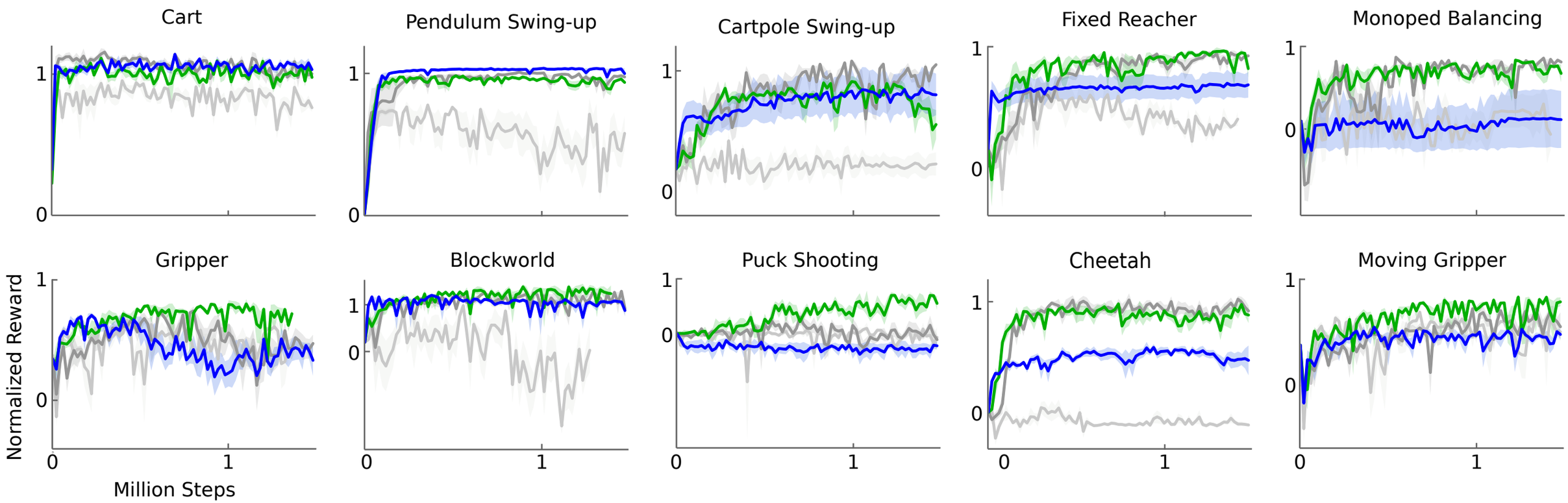



Figure 2: Performance curves for a selection of domains using variants of DPG: original DPG algorithm (minibatch NFQCA) with batch normalization (light grey), with target network (dark grey), with target networks and batch normalization (green), with target networks from pixel-only inputs (blue). Target networks are crucial.

# Over estimation of Q-values

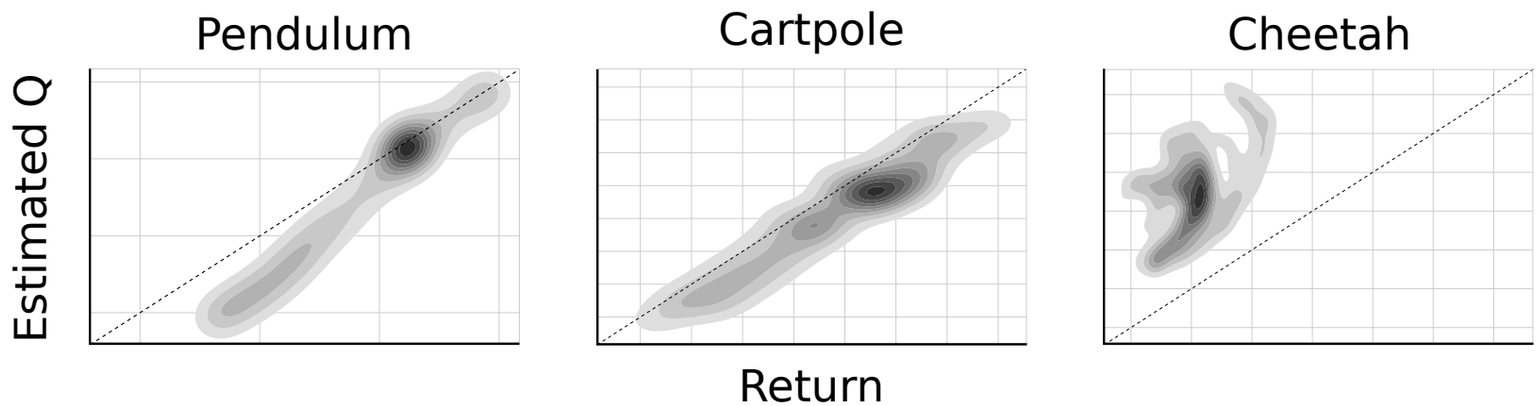


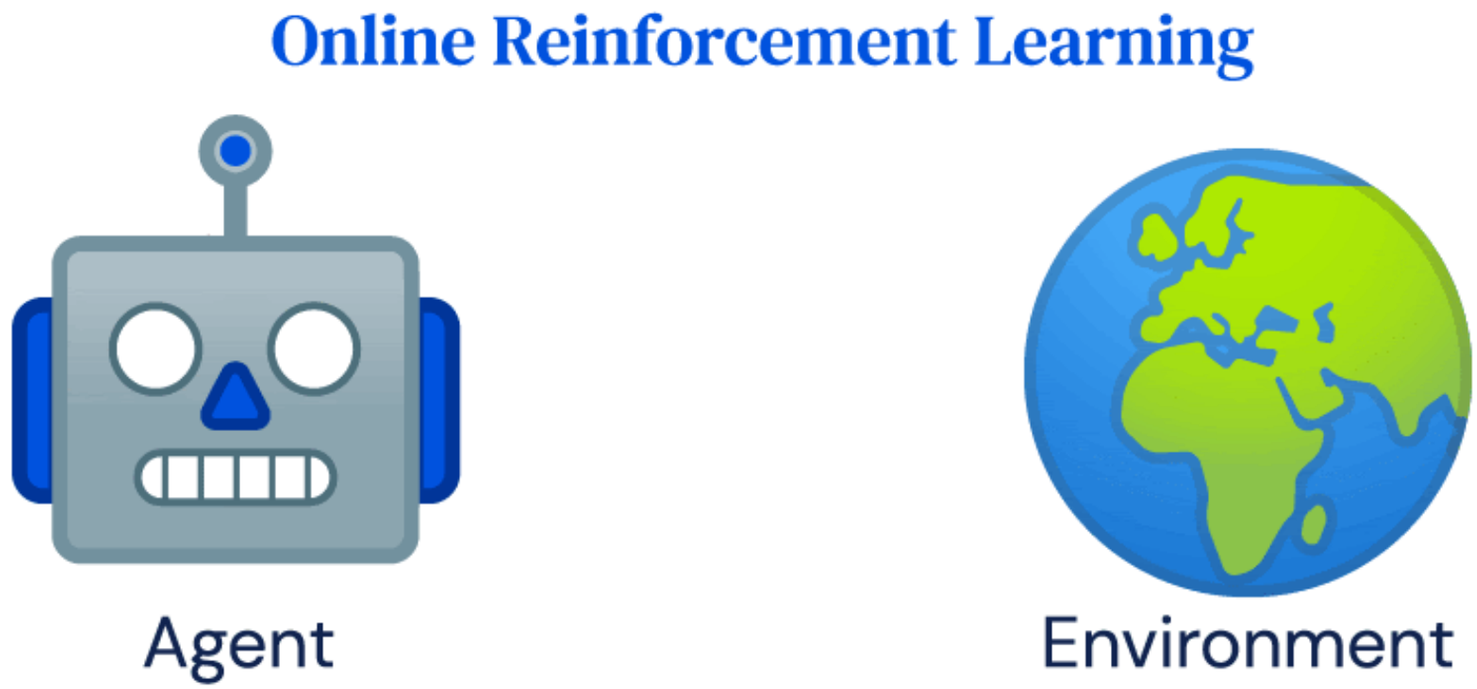

Figure 3: Density plot showing estimated Q values versus observed returns sampled from test episodes on 5 replicas. In simple domains such as pendulum and cartpole the Q values are quite accurate. In more complex tasks, the Q estimates are less accurate, but can still be used to learn competent policies. Dotted line indicates unity, units are arbitrary.

# Related Work

- TRPO
- PILCO
- iLQG

# Batch RL / Offline RL

Instead of actively interacting with the environment




# Batch RL / Offline RL

## Why batch RL?

- Re-use experience: gathering experience is the most expensive part of RL
- Gathering experience may be unsafe
- Learn from other's experience

# Problems with Off-line Learning

$$Q(s,a) \leftarrow r + \gamma \max_{a'} Q(s',a') \quad \forall (s,a,s',r) \in \mathscr{D}$$

- *Extrapolation error*
  - we do not know where our estimate of $Q(s',a')$ is good
  - even if we assume $Q(s',a')$ is an unbiased estimate, the **max** will cause it to become biased

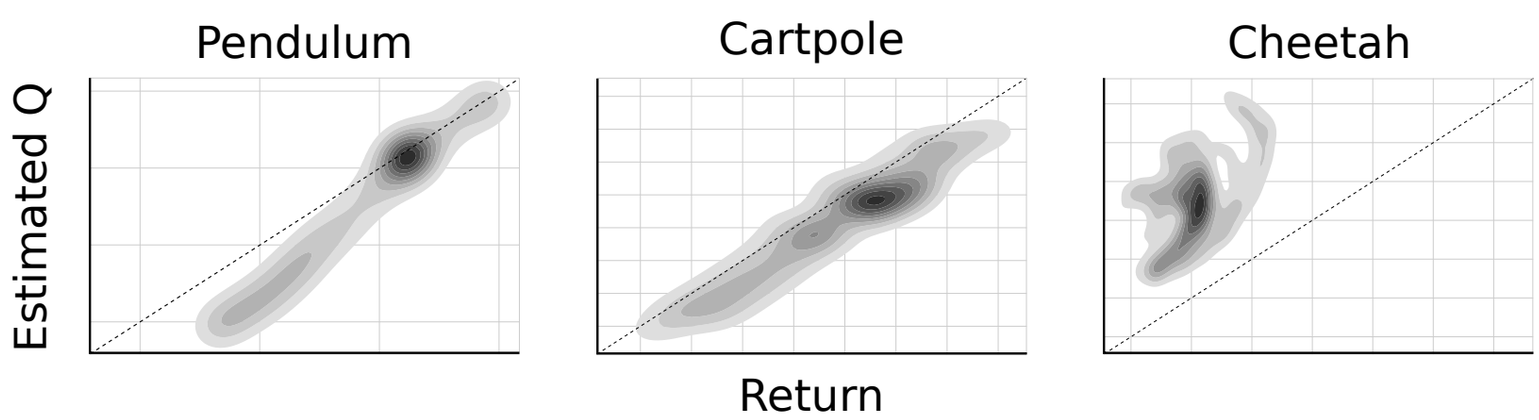
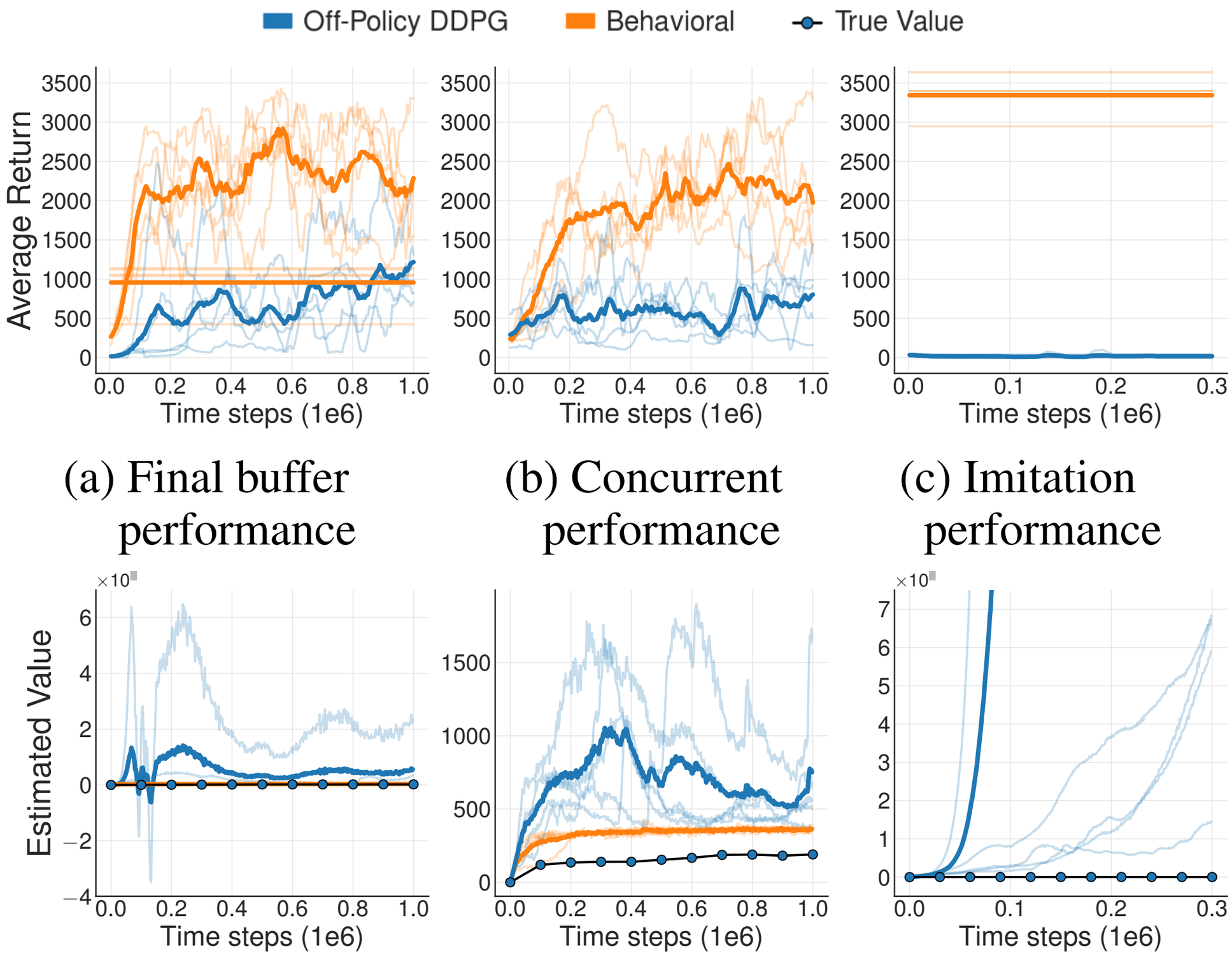



Figure 3: Density plot showing estimated Q values versus observed returns sampled from test episodes on 5 replicas. In simple domains such as pendulum and cartpole the Q values are quite accurate. In more complex tasks, the Q estimates are less accurate, but can still be used to learn competent policies. Dotted line indicates unity, units are arbitrary.

# Experiment 1




(a) Final buffer performance

(b) Concurrent performance

(c) Imitation performance

(d) Final buffer value estimate

(e) Concurrent value estimate

(f) Imitation value estimate

# But, existing methods work, don't they?

- DQN, DDPG aren't really off-policy, use $\epsilon$-greedy policies
- $\mathbf{max}$ introduces a bias, but unsubstantiated optimism can be tested in subsequent iterations.

# Batch-constrained Q-learning

- policy should induce a similar state-action distribution as dataset
  - minimize distance of selected action to data in batch
  - lead to states where familiar data is observed
  - maximize the value function
- train a pair of networks (use minimum of Q-value)

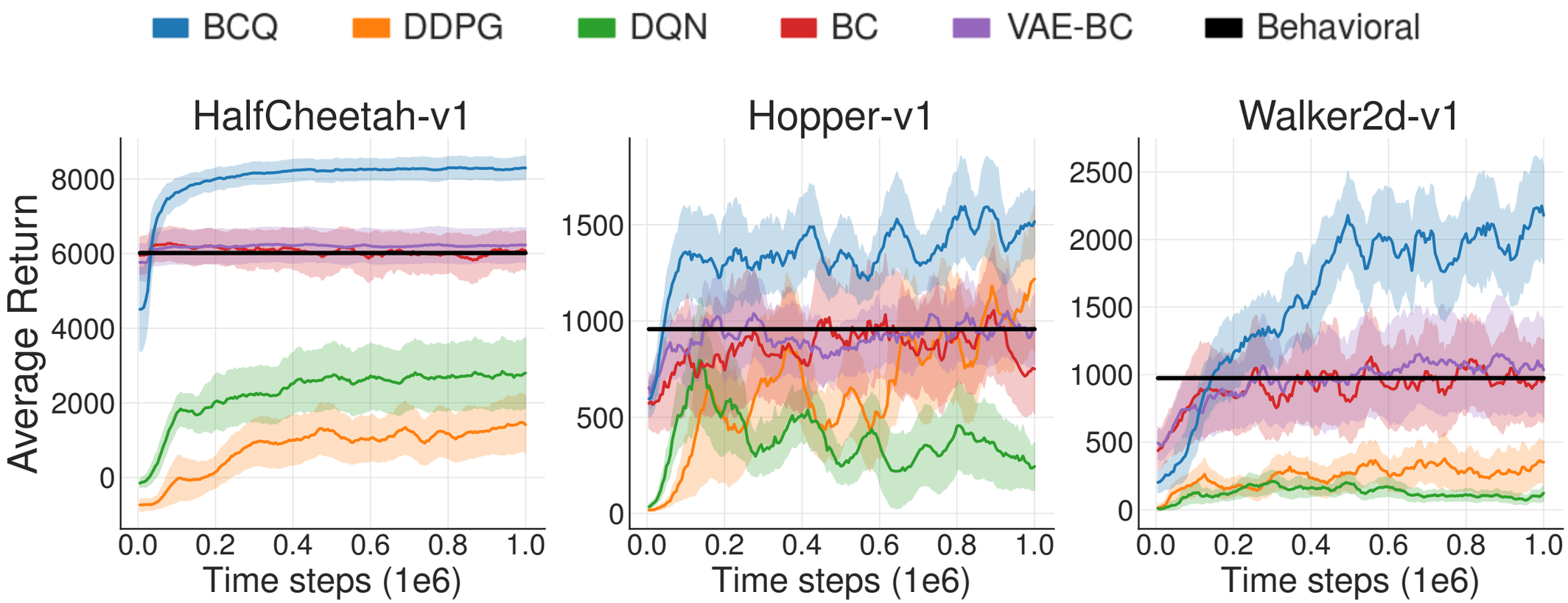
# Batch-constrained Q-learning

**Algorithm 1** BCQ

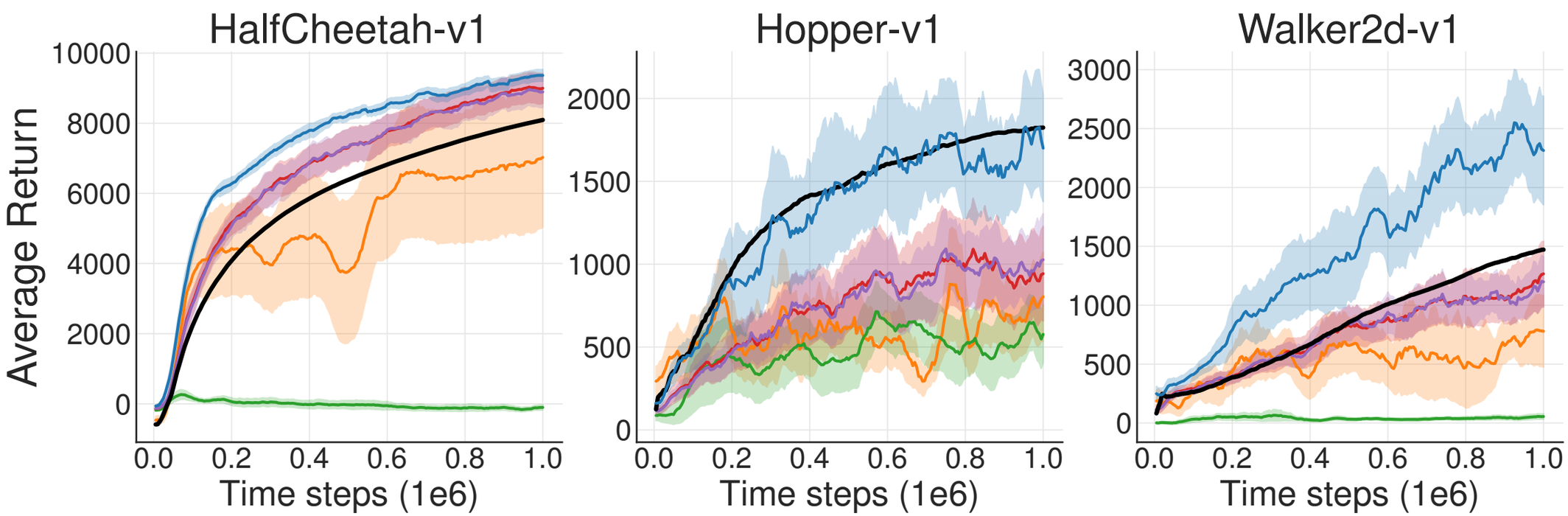
**Input:** Batch $\mathcal{B}$, horizon $T$, target network update rate $\tau$, mini-batch size $N$, max perturbation $\Phi$, number of sampled actions $n$, minimum weighting $\lambda$.
Initialize Q-networks $Q_{\theta_1}, Q_{\theta_2}$, perturbation network $\xi_\phi$, and VAE $G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$, with random parameters $\theta_1$, $\theta_2$, $\phi$, $\omega$, and target networks $Q_{\theta'_1}, Q_{\theta'_2}$, $\xi_{\phi'}$ with $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$.
**for** $t = 1$ **to** $T$ **do**
  Sample mini-batch of $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$
  $\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$
  $\omega \leftarrow \text{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$
  Sample $n$ actions: $\{a_i \sim G_\omega(s')\}_{i=1}^n$
  Perturb each action: $\{a_i = a_i + \xi_\phi(s', a_i, \Phi)\}_{i=1}^n$
  Set value target $y$ (Eqn. 13)
  $\theta \leftarrow \text{argmin}_\theta \sum (y - Q_\theta(s, a))^2$
  $\phi \leftarrow \text{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$
  Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$
  $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
**end for**

$$r + \gamma \max_{a_i} \left[ \lambda \min_{j=1,2} Q_{\theta'_j}(s', a_i) + (1 - \lambda) \max_{j=1,2} Q_{\theta'_j}(s', a_i) \right] \tag{13}$$

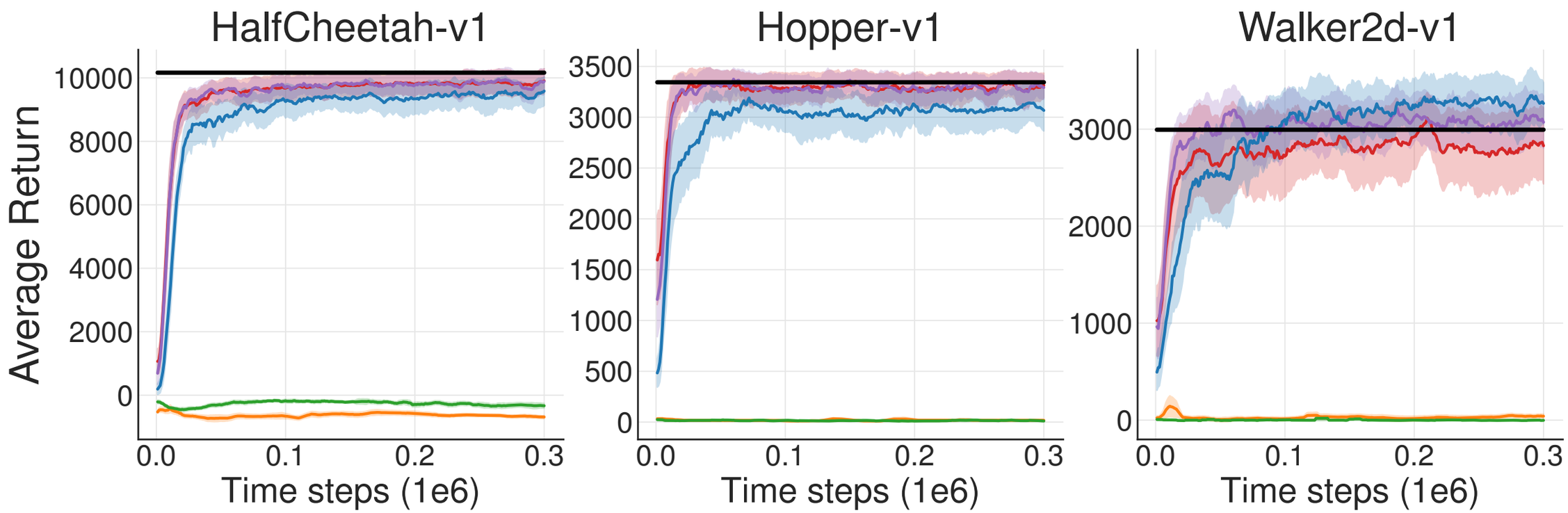
# Variational Auto Encoders



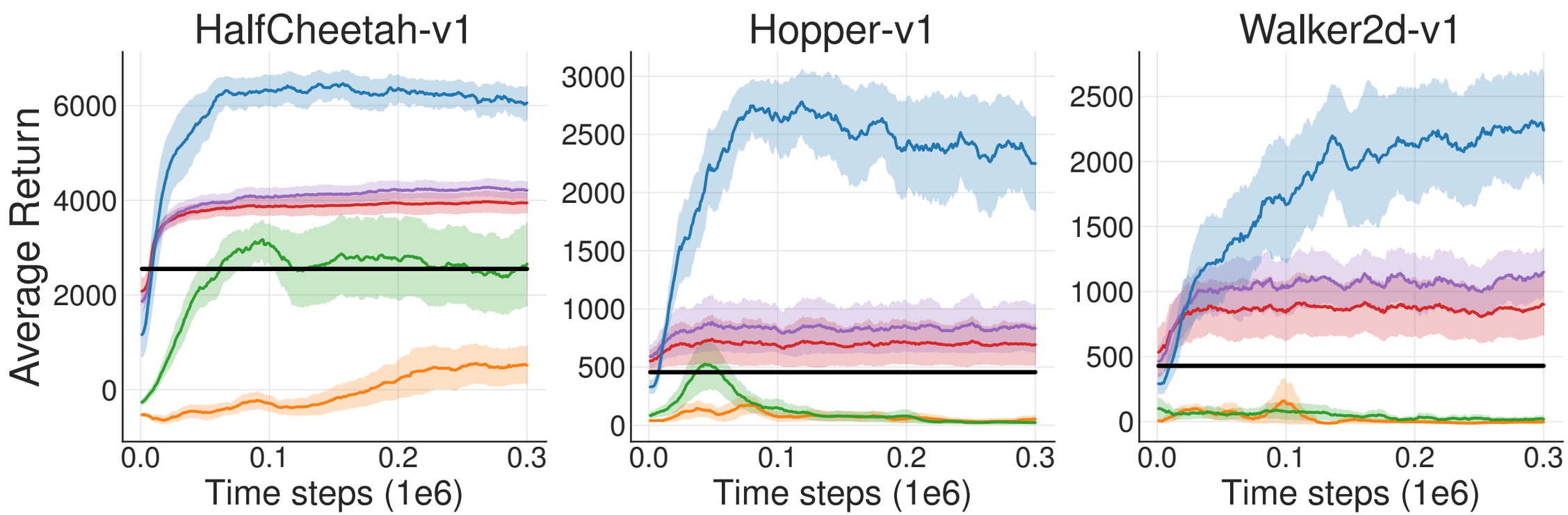

(a) Final buffer performance




(b) Concurrent performance

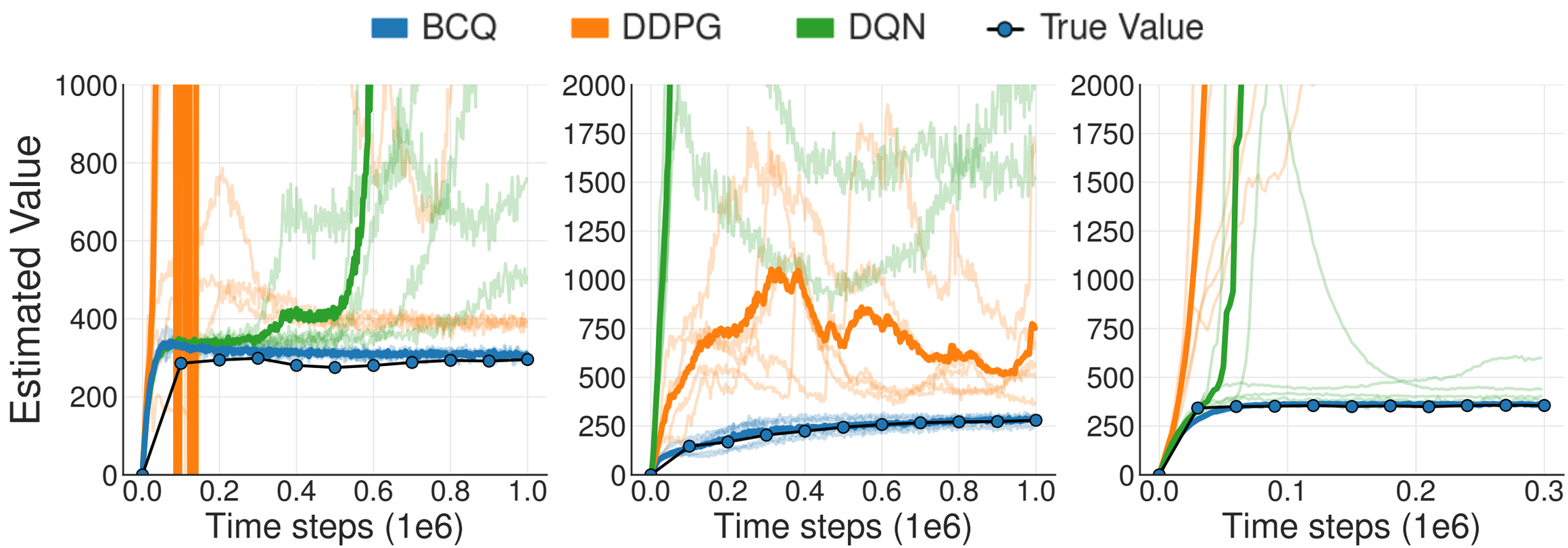



(c) Imitation performance




(d) Imperfect demonstrations performance

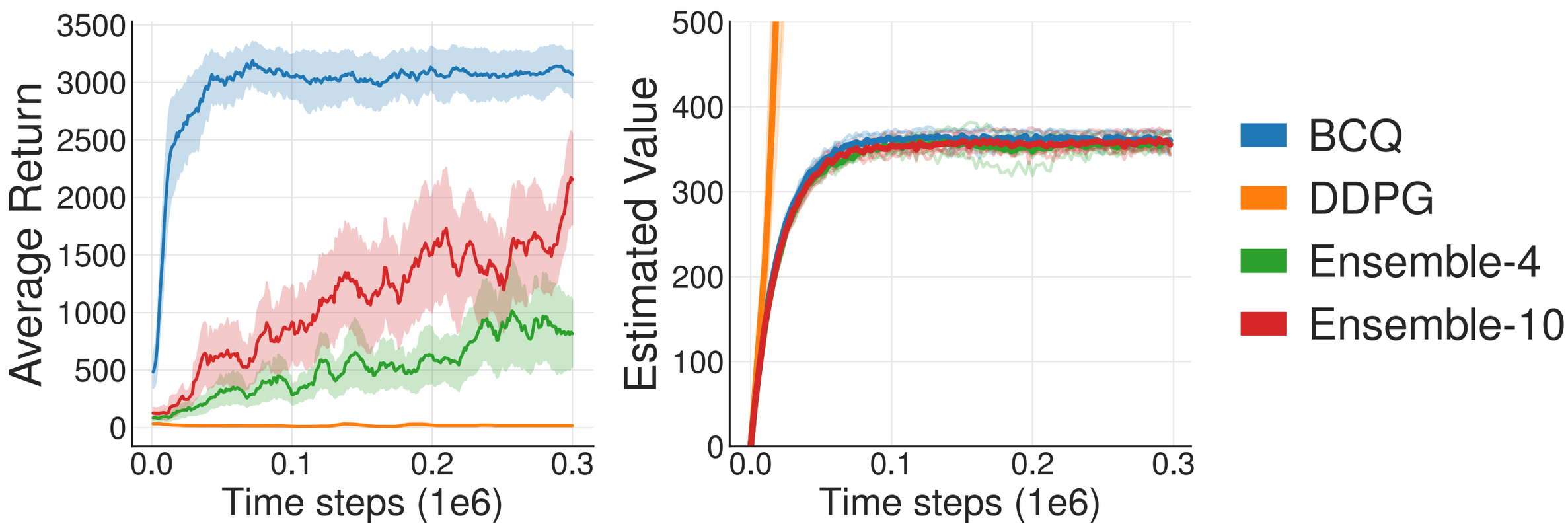
# Q-value Estimates



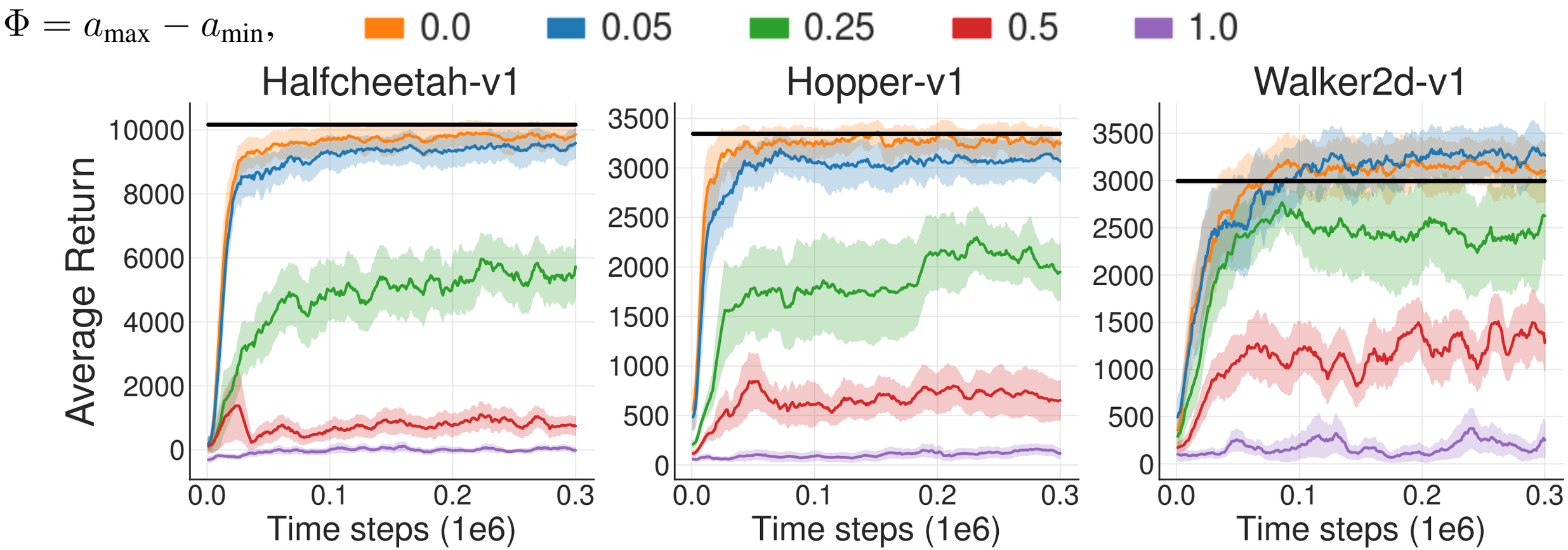

(a) Final Buffer (b) Concurrent (c) Imitation

# Related Work

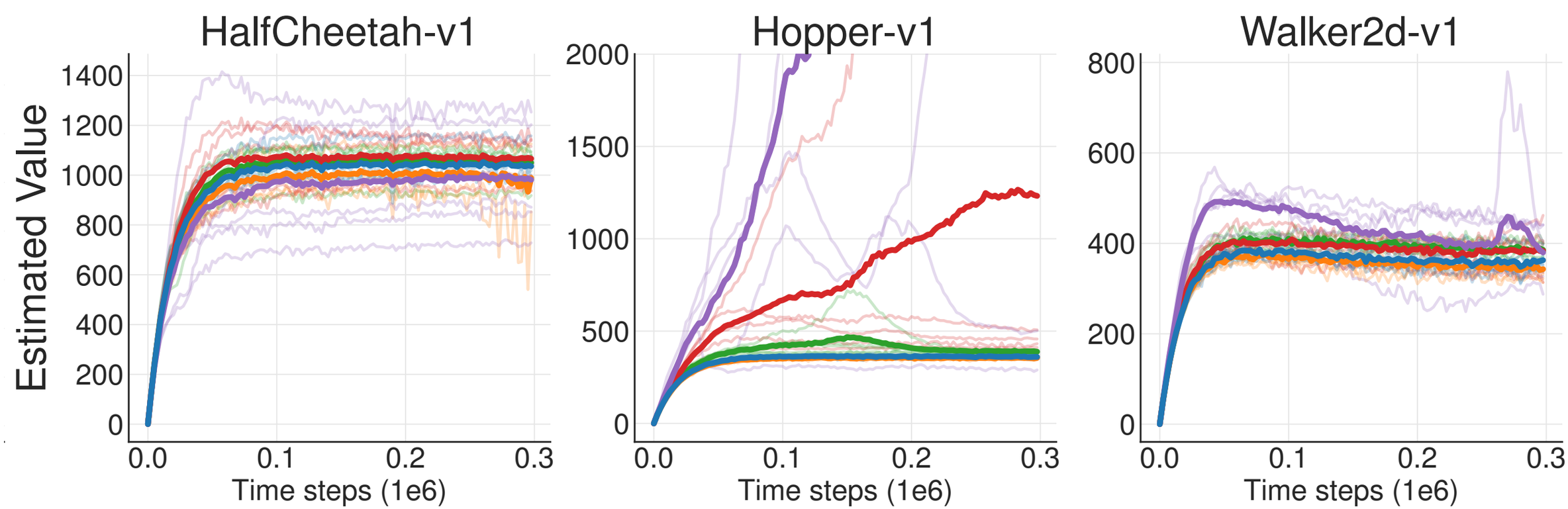
- Modeling uncertainty in neural networks




(a) Imitation performance (b) Imitation value estimates




(a) Imitation performance




(b) Imitation value estimates