Continuous control with deep reinforcement learning

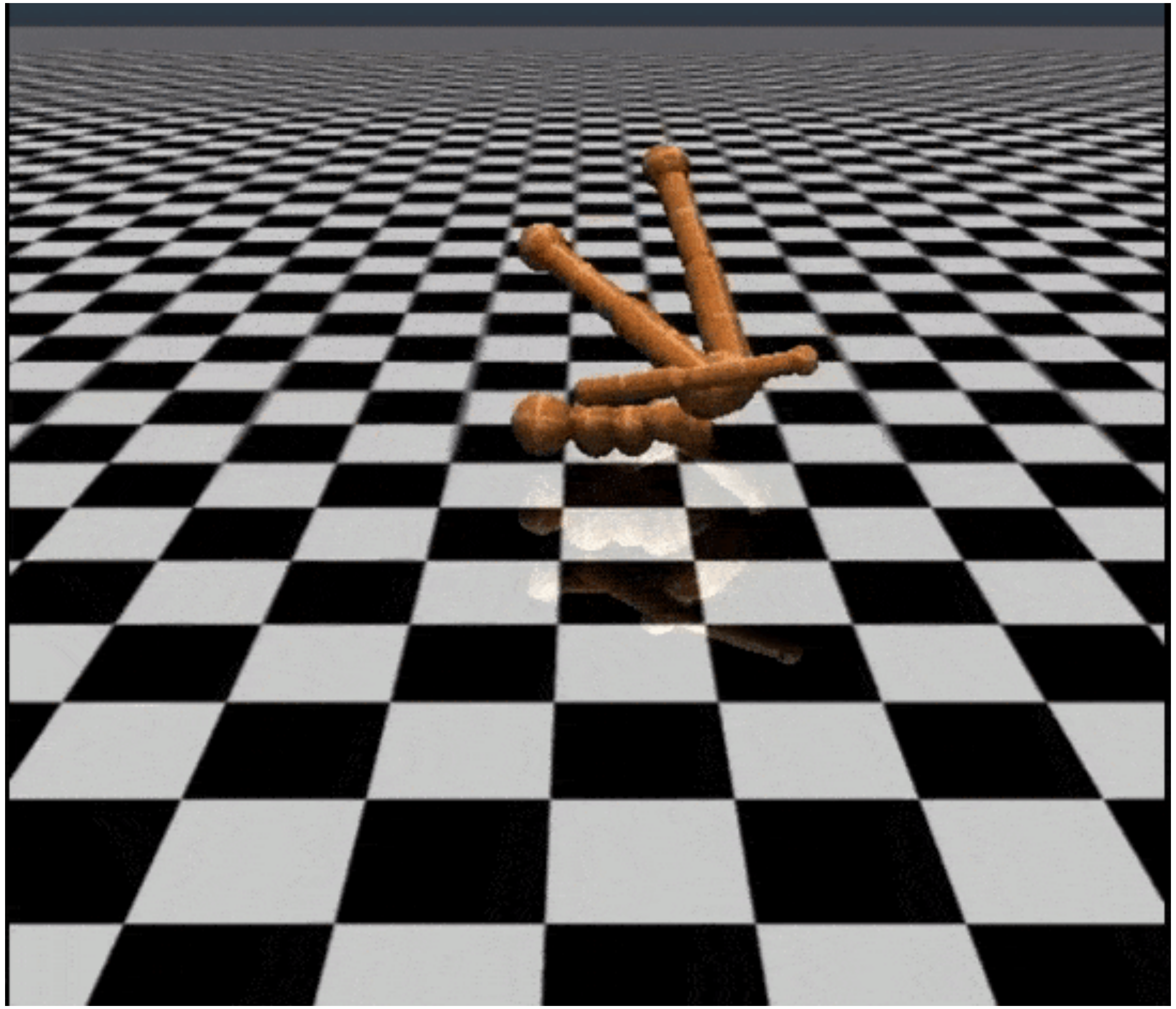
Based on a paper by Lillicrap, Timothy P., et al. 2016

Sreekar Chigurupati . 6th April 2022

Context

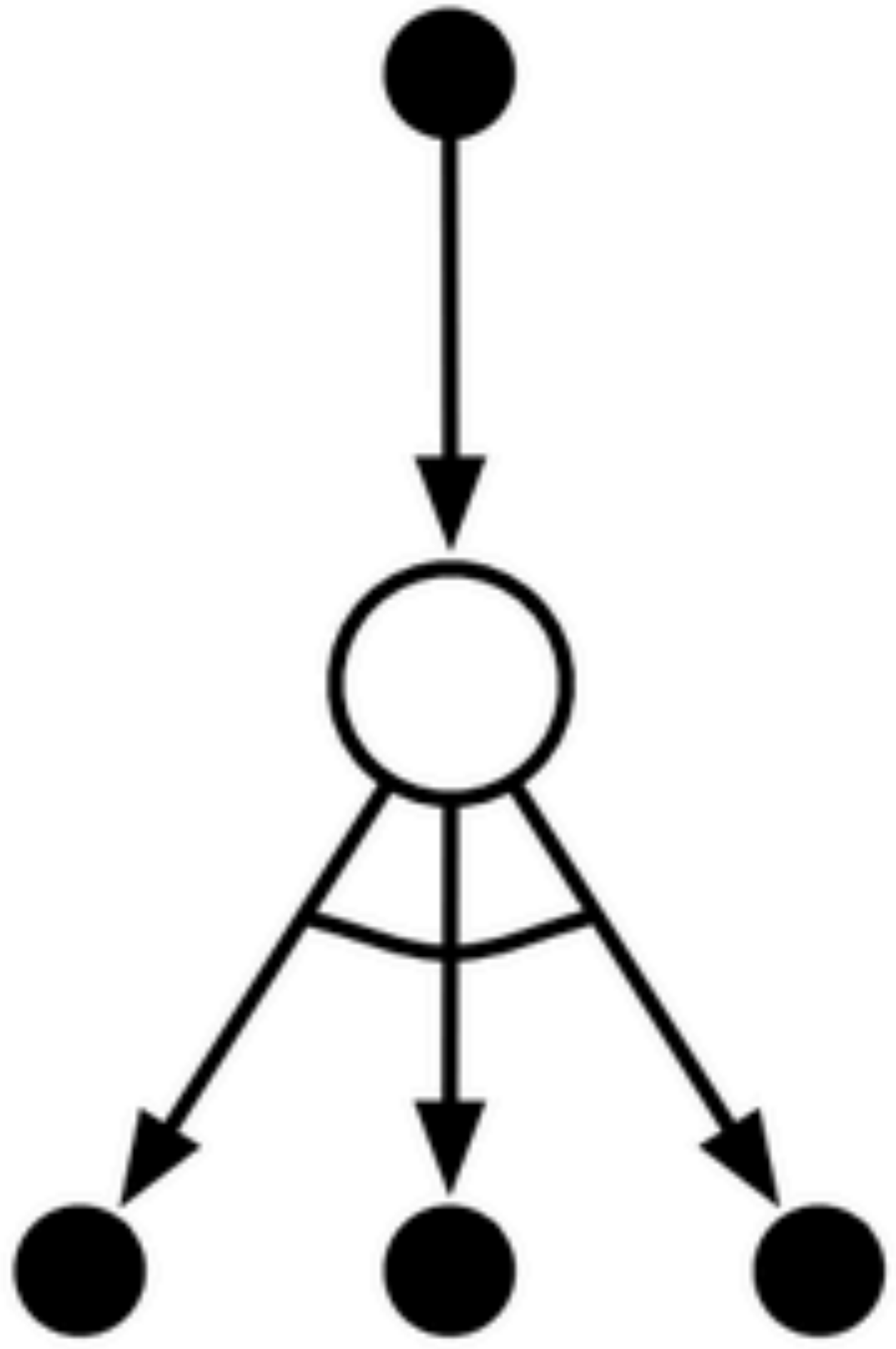
Continuous control

- Deep Q-Learning already successful
- DQN used to solve Atari suite
- Novelty: DQN applied to continuous space



Q-Learning

- Off-policy temporal difference control algorithm
- $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$
- Q directly approximates q*
- Optimizes over action space

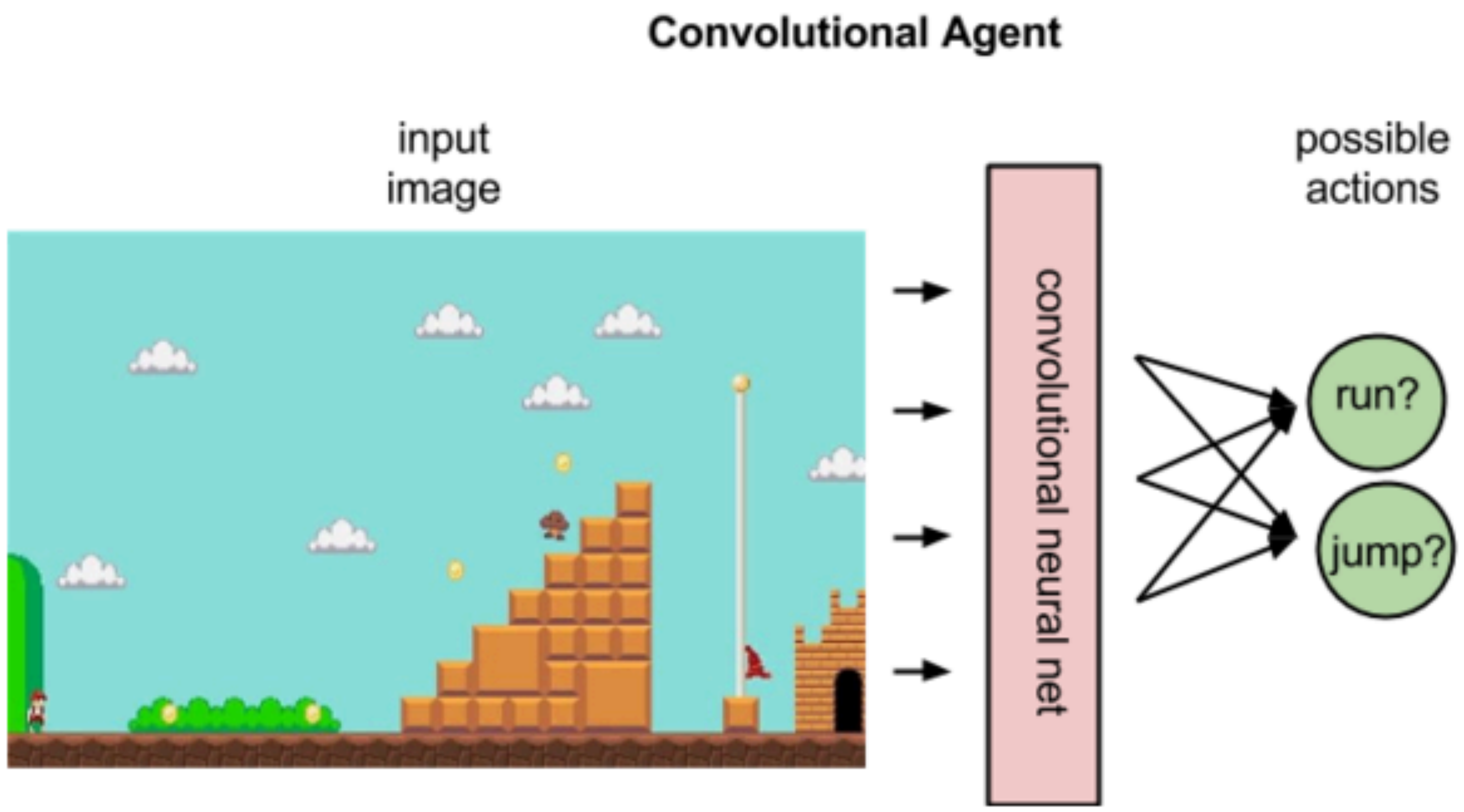


Q-learning

DQN

Q-Learning framework with a neural net

- Solves problems with high-dimensional observation space
- Limited to low-dimensional discrete action space
- Physical control -> continuous high-dimensional action spaces
- Discretization: number of actions exponential w.r.t DOF
- Fine-control also increases number



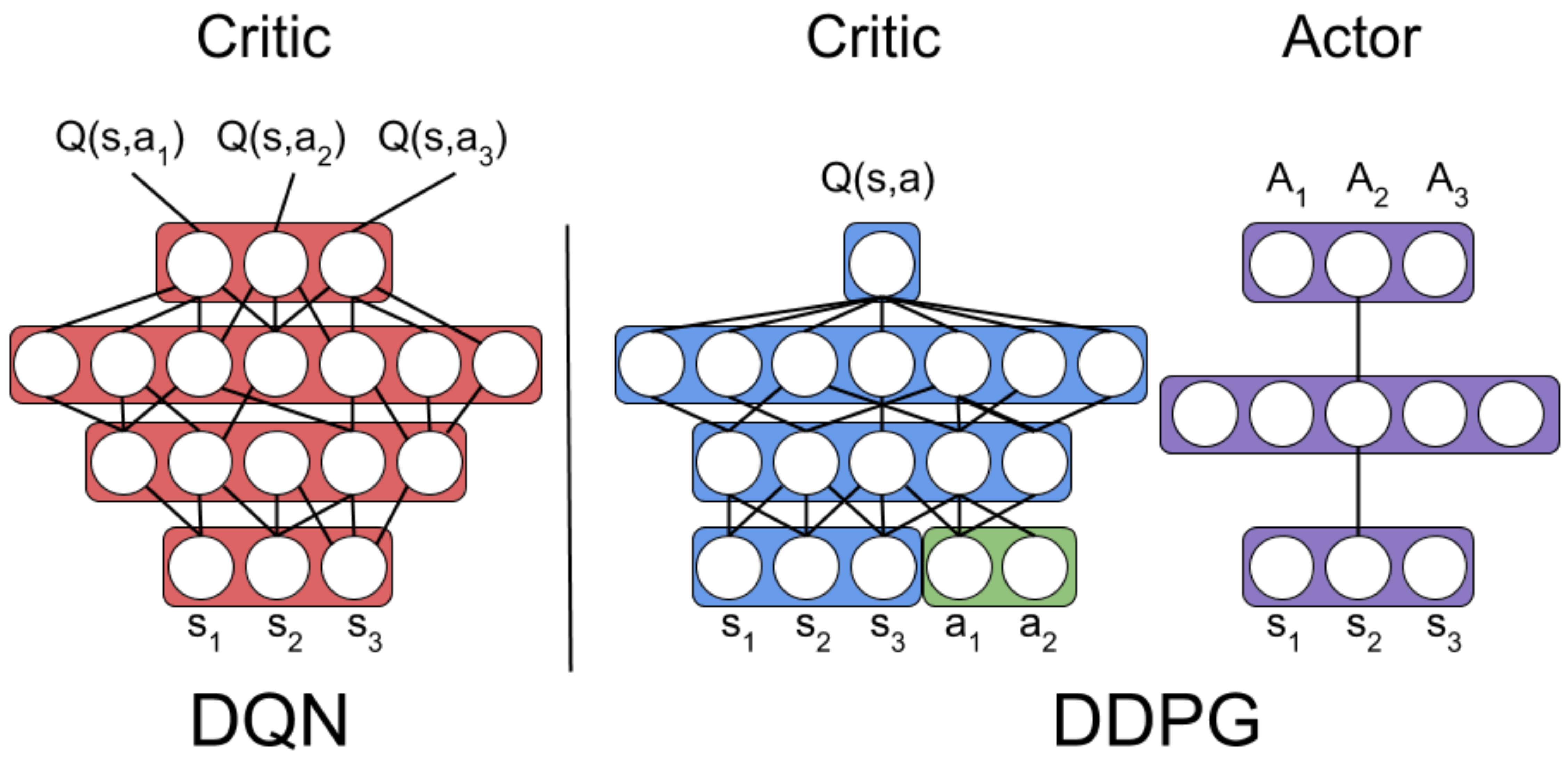
DQN

Continuous case

- Naive discretization may throw away action space structure
- To find action that maximizes action-value function, iterative optimization at each step is needed
- Also uses replay buffer

Deep Deterministic Policy Gradient

- DPG -> actor-critic
- DDPG <- DPG + DQN
- Model-free, off-policy, actor-critic also using deep function approximators
- Learns policies in continuous high-dimensional action spaces



DPG

Deterministic Policy Gradient

- $$\begin{aligned} \nabla_{\theta^\mu} J &\approx \mathbb{E}_{s_t \sim \rho^\beta} \left[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^\mu)} \right] \\ &= \mathbb{E}_{s_t \sim \rho^\beta} \left[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s|\theta^\mu)|_{s=s_t} \right] \end{aligned}$$ [Actor update]
- maintains a parameterized actor function $\mu(s|\theta^\mu)$ which specifies the current policy by deterministically mapping states to a specific action
- The critic Q(s, a) is learned using the Bellman equation as in Q-learning with L2 weight decay
- Non-linear function approximators -> converge no longer guaranteed

Function approximation

Learning value functions using large, non-linear function approximators is difficult and unstable

- Network is trained off-policy with samples from a replay buffer to minimize correlations between samples
- Network is trained with a target Q network to give consistent targets during temporal difference backups
- Batch normalization

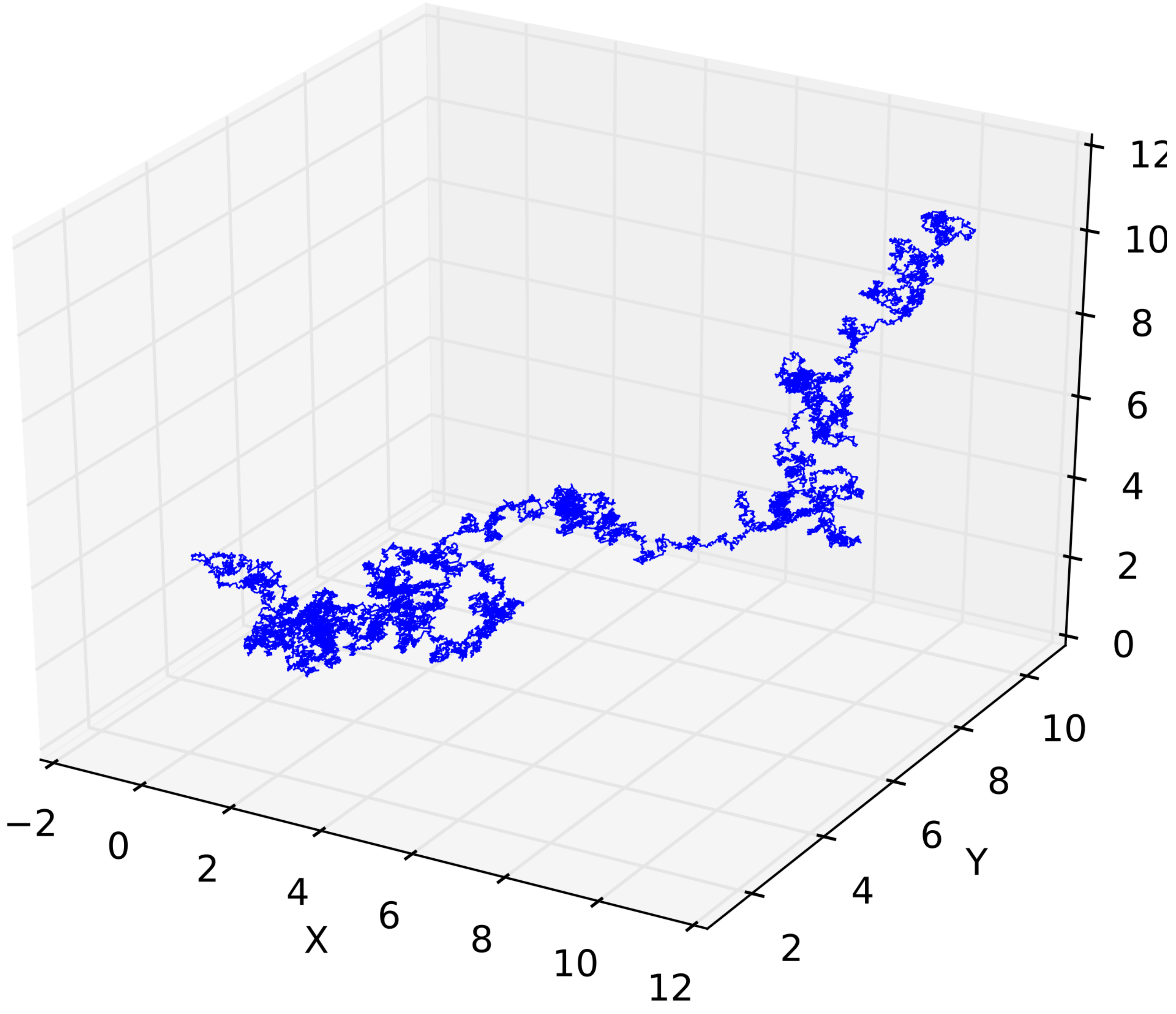
DDPG

- Replay buffer like DQN
- Copies actor/critic networks to a target network and updates target network slowly
- Normalizes scales of input by batch normalization. Normalize each dim across samples in each mini batch to have unit mean and variance
- Action repeats of length 3

Behavioral policy

Ornstein-Uhlenbeck process

- $\mu'(s_t) = \mu(s_t|\theta^{\mu}{}_t) + N$
- Exploration policy = actor policy + noise
- Used to generate temporally correlated exploration for exploration efficiency in physical control problems with inertia
- A 3D simulation with θ = 1.0, σ = 3, μ = (0, 0, 0) and the initial position (10, 10, 10)



DDPG

Continued

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 $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    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:

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

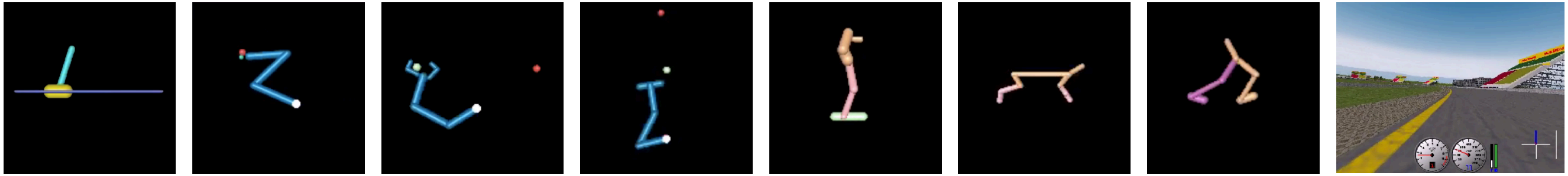
    Update the target networks:

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

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

  end for
end for

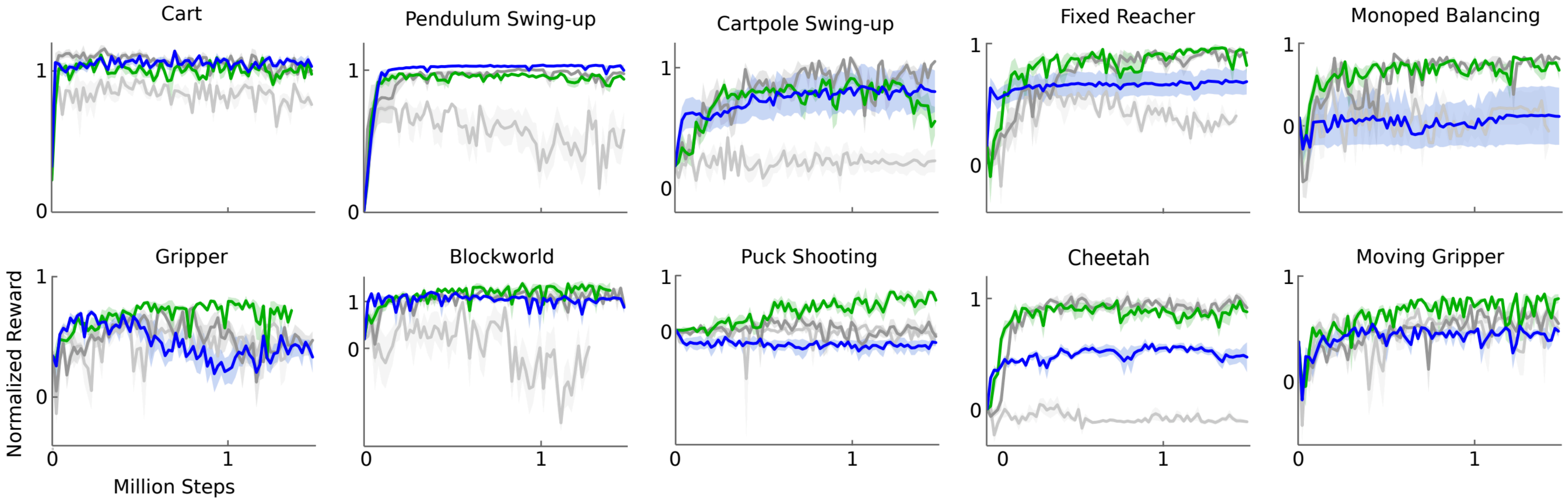
Tasks



Experiments & Results

- 20 classic physics tasks
- Cartpole swing-up, dexterous manipulation, legged locomotion and car driving
- Involve complex multi-joint movements, unstable and rich contact dynamics, and gait behavior
- Policies can be learnt end-to-end
- Policy performance competitive to planning with full access, sometimes exceed planning

Results



Batch normalization
Target network
Target net and batch normalization
Target net from pixel input only

Conclusion

- Target network + batch normalization necessary
- Learning from pixels can be as good as from states. Conv layers might provide a separable state space. NN learns the necessary transformation
- Expanded model-free RL to continuous domain


Cheetah
Low Dimensional Features

Discussion