Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model

CS330 Student Presentation

Table of Contents

Motivation

- We would like to use reinforcement learning algorithms to solve tasks using only low-level observations, such as learning **robotic control** using only unstructured raw **image data**
- The standard approach relies on sensors to obtain information that would be helpful for learning
- Learning from only image data is hard because the RL algorithm must learn both a **useful representation of the data** and the **task itself**
- This is called the **representation learning problem**

Approach

The approach of the paper is a two-fold approach:

1. Learn a **predictive stochastic latent variable model** for given high-dimensional data (i.e., images)
2. Perform learning in the **latent space** of the latent variable model

The Stochastic Latent Variable model

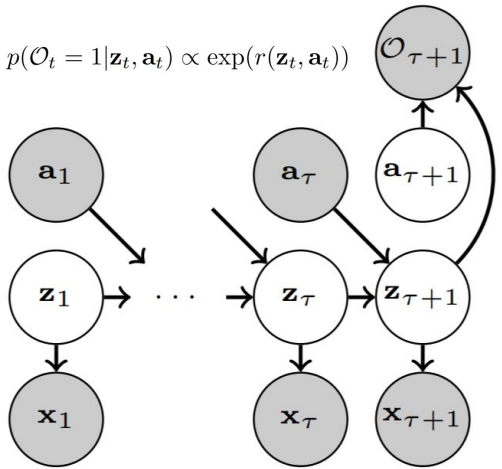
- We would like our latent variable model to represent a **partially-observable Markov Decision Process (POMDP)**
- The authors choose a graphical model for the latent variable model
- Previous work have used mixed deterministic-stochastic models, but SLAC's model is purely stochastic
- The graphical model will be trained using amortized variational inference

$$J_M(\psi) = \mathop{\mathbb{E}}_{(\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}, r_{1:\tau}) \sim \mathcal{D}} \Bigg[\mathop{\mathbb{E}}_{\mathbf{z}_{1:\tau+1} \sim q_\psi} \Bigg[\sum_{t=1}^{\tau+1} \log p_\psi(\mathbf{x}_t | \mathbf{z}_t) - \mathrm{D}_{\mathrm{KL}} \left(q_\psi(\mathbf{z}_1 | \mathbf{x}_1) \parallel p_\psi(\mathbf{z}_1) \right) - \sum_{t=1}^{\tau} \mathrm{D}_{\mathrm{KL}} \left(q_\psi(\mathbf{z}_{t+1} | \mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t) \parallel p_\psi(\mathbf{z}_{t+1} | \mathbf{z}_t, \mathbf{a}_t) \right) \Bigg] \Bigg] .$$

Graphical model representation of POMDP

- Since we can only observe part of the true state, we need past information to infer the next latent state
- We can derive an **evidence lower bound (ELBO)** for POMDP:

$$\log p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T}|\mathbf{a}_{1:\tau}) \geq \mathop{\mathbb{E}}_{(\mathbf{z}_{1:\tau+1}, \mathbf{a}_{1:\tau})\sim q} \Bigg[\sum_{t=1}^{\tau+1} \log p(\mathbf{x}_t|\mathbf{z}_t) - \mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{z}_1|\mathbf{x}_1) \parallel p(\mathbf{z}_1)\right) - \sum_{t=1}^{\tau} \mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t) \parallel p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)\right) + r(\mathbf{z}_{\tau+1}, \mathbf{a}_{\tau+1}) - \log \pi(\mathbf{a}_{\tau+1}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) + V(\mathbf{z}_{\tau+1}) \Bigg]$$



Learning in the Latent Space

- The SLAC algorithm can be viewed as an extension of the Soft Actor-Critic algorithm (SAC)
- Learning is done in the **maximum entropy setting**, where we seek to maximize the entropy along with the expected reward:

$$\phi^* = \arg\max_{\phi} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} [r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi_\phi(\cdot|\mathbf{s}_t))],$$

- The entropy term encourages exploration

Soft Actor-Critic (SAC)

- As an actor-critic method, SAC learns both value function approximators (the **critic**) and a policy (the **actor**)
- SAC is trained using alternating policy evaluation and policy improvement
- Training is done in the latent space (i.e., in the state space **z**)

$$J_Q(\theta) = \mathop{\mathbb{E}}_{(\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}, r_\tau) \sim \mathcal{D}} \left[\mathop{\mathbb{E}}_{\mathbf{z}_{1:\tau+1} \sim q_\psi} \left[\frac{1}{2} \left(Q_\theta(\mathbf{z}_\tau, \mathbf{a}_\tau) - \left(r_\tau + \gamma \mathop{\mathbb{E}}_{\mathbf{a}_{\tau+1} \sim \pi_\phi} [Q_{\bar{\theta}}(\mathbf{z}_{\tau+1}, \mathbf{a}_{\tau+1}) - \alpha \log \pi_\phi(\mathbf{a}_{\tau+1} | \mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau})] \right) \right)^2 \right] \right]$$

$$J_\pi(\phi) = \mathop{\mathbb{E}}_{(\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) \sim \mathcal{D}} \left[\mathop{\mathbb{E}}_{\mathbf{z}_{1:\tau+1} \sim q_\psi} \left[\mathop{\mathbb{E}}_{\mathbf{a}_{\tau+1} \sim \pi_\phi} [\alpha \log \pi_\phi(\mathbf{a}_{\tau+1} | \mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) - Q_\theta(\mathbf{z}_{\tau+1}, \mathbf{a}_{\tau+1})] \right] \right]$$

Soft Actor Critic (SAC), con’t

- SAC learns two Q-networks, a V-network, and a policy network
- Two Q-networks are used to mitigate overestimation bias
- A V-network is used to stabilize training
- Taking gradients through the expectations is done using the reparametrization trick

Putting it all Together

- Finally, both the latent variable model and agent are trained together
- The full SLAC model has two layers of latent variables

$$
\begin{aligned}
p_\psi(\mathbf{z}_1) &= p_\psi(\mathbf{z}_1^2|\mathbf{z}_1^1)p(\mathbf{z}_1^1), \\
p_\psi(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t) &= p_\psi(\mathbf{z}_{t+1}^2|\mathbf{z}_{t+1}^1, \mathbf{z}_t^2, \mathbf{a}_t)p_\psi(\mathbf{z}_{t+1}^1|\mathbf{z}_t^2, \mathbf{a}_t), \\
q_\psi(\mathbf{z}_1|\mathbf{x}_1) &= p_\psi(\mathbf{z}_1^2|\mathbf{z}_1^1)q_\psi(\mathbf{z}_1^1|\mathbf{x}_1), \\
q_\psi(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t) &= p_\psi(\mathbf{z}_{t+1}^2|\mathbf{z}_{t+1}^1, \mathbf{z}_t^2, \mathbf{a}_t)q_\psi(\mathbf{z}_{t+1}^1|\mathbf{x}_{t+1}, \mathbf{z}_t^2, \mathbf{a}_t).
\end{aligned}
$$

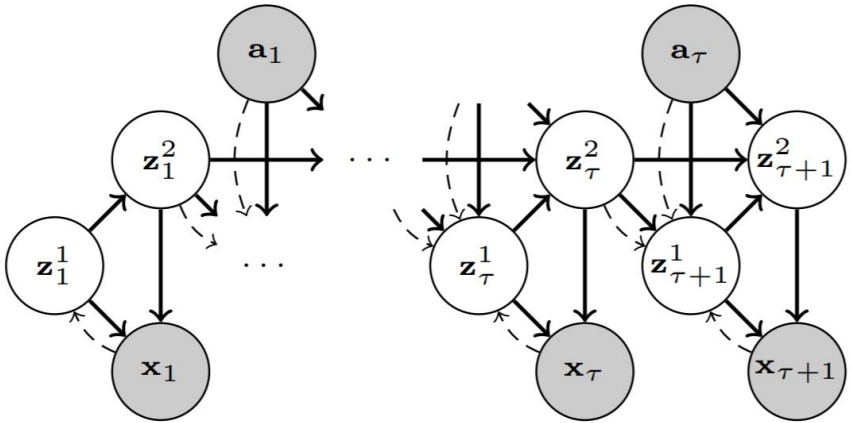


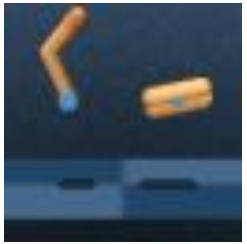
Image-based Continuous Control Tasks

- Four tasks from DeepMind Control Suite

Cheetah run

Walker walk

Ball-in-cup catch



Finger spin

- Four tasks from OpenAI Gym

Cheetah

Walker

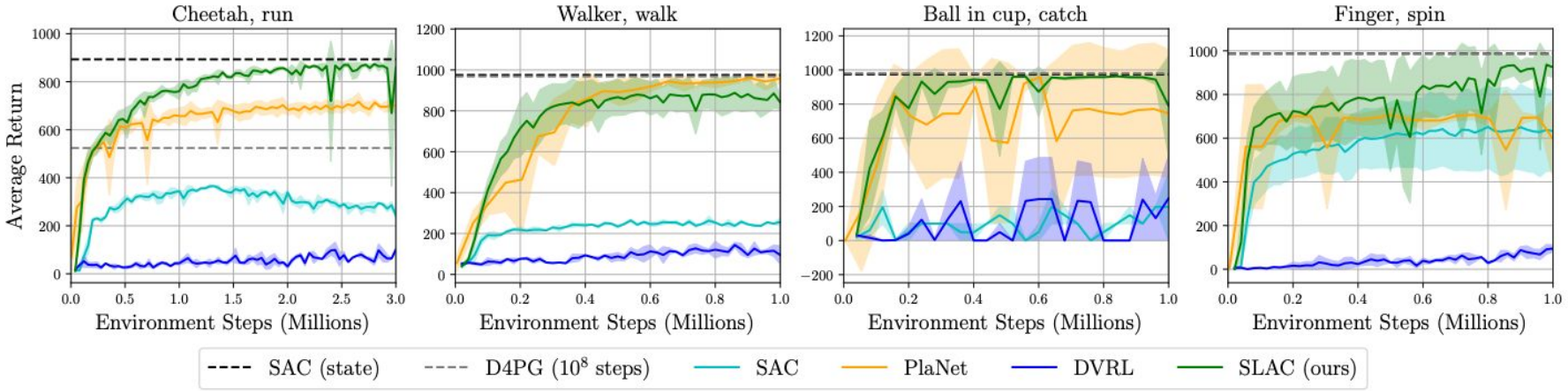
Hopper

Ant

Comparison with other models

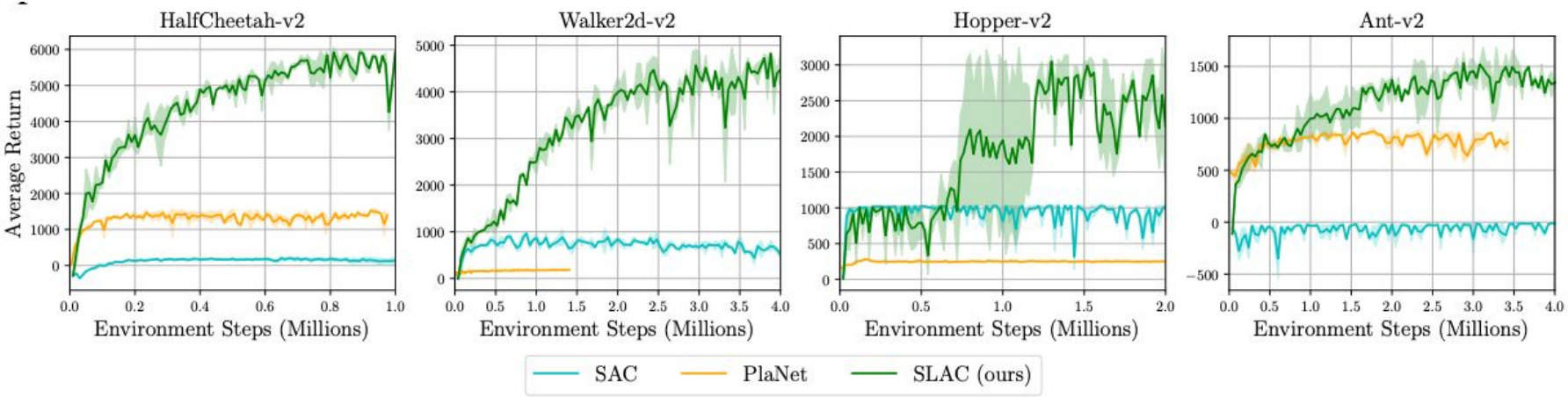
- SAC
  - Off-policy actor-critic algorithm, learning directly from images or true states
- D4PG
  - Off-policy actor-critic algorithm, learning directly from images
- PlaNet
  - Model-based RL method for learning directly from images
  - Mixed deterministic/stochastic sequential latent variable model
  - No explicit policy learning yet used model predictive control (MPC)
- DVRL
  - On-policy model-free RL algorithm
  - Mixed deterministic/stochastic latent-variable POMDP model

Results on DeepMind Control Suite (4 tasks)



- Sample efficiency of SLAC is comparable or better than both model-based and model-free
- Outperforms DVRL
  - Efficient off-policy RL algorithm take advantage of the learned representation

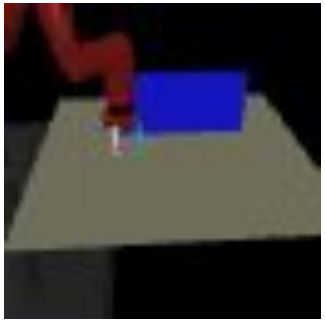
Results on OpenAI Gym (4 tasks)



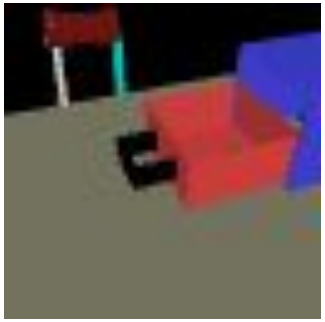
- Tasks are more challenging than DeepMind Control Suite tasks
  - Rewards not shaped, not bounded between 0 and 1
  - More complex dynamics
  - Episode terminate on failure
- PlaNet unable to solve last three tasks but obtains sub-optimal policy on cheetah

Robotic Manipulation Tasks

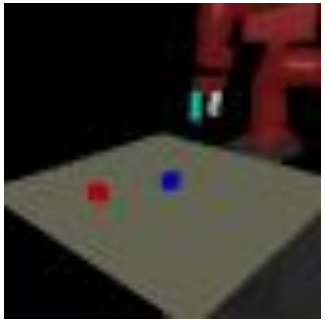
- 9-DoF 3-fingered DClaw robot



Push a door



Close a drawer



Reach out and pick up an object

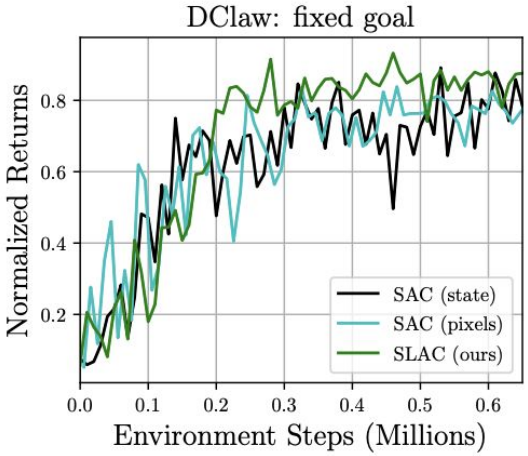
*Note: SLAC algorithm achieves above actions

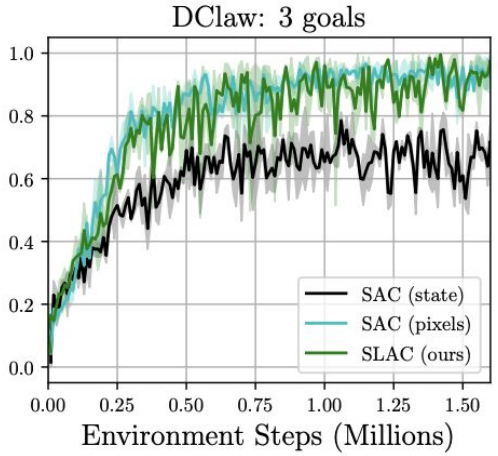
Robotic Manipulation Tasks (continued)

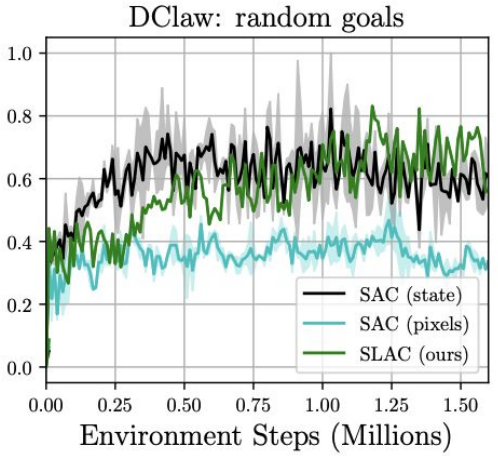
- **9-DoF 3-fingered DClaw robot**
- **Goal**: rotate a valve from various starting positions to various desired goal locations
- **Three** different settings:

1. Fixed goal position
2. Random goal from 3 options : $\text{goal} \in \{-\frac{\pi}{2}, 0, \frac{\pi}{2}\}$
3. Random goal : $\text{goal} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$

Results







Goal: **Turning a valve to a desired location**

Takeaways:

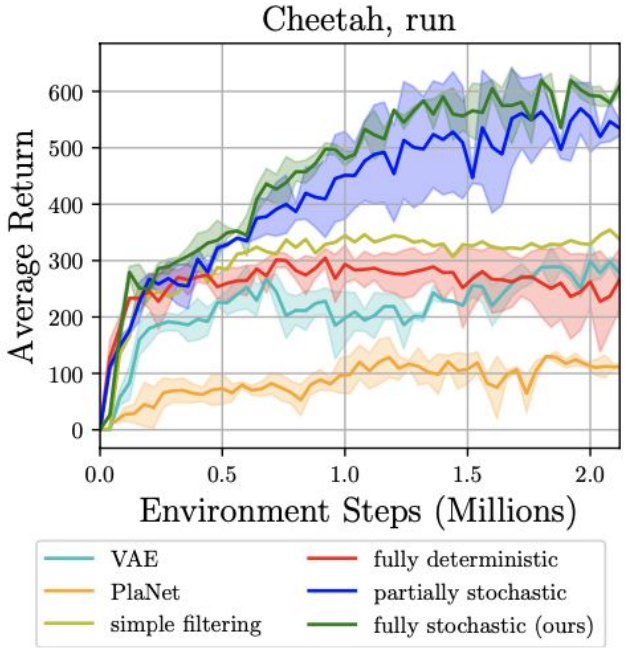
- For **fixed goal** setting, all performances are similar
- For **three random goal** setting, SLAC and SAC from raw images performs well
- For **random goal setting**, SLAC performs better than SAC from raw images / comparable to SAC from states

Latent Variable Models

- **Six** different models:
  - Non-sequential VAE
  - PlaNet (Mixed deterministic/stochastic Model)
  - Simple Filtering (without factoring model)
  - Fully deterministic
  - Mixed deterministic/stochastic Model
  - Fully stochastic
- Under **fixed RL framework of SLAC**

Takeaway:

- **Fully stochastic** model outperforms others



SLAC paper summary

- Propose a **SLAC RL algorithm** for learning from high-dimensional image inputs
- Combined off-policy model-free RL with representation learning via a **sequential stochastic state space model**
- SLAC's **fully stochastic model** outperforms other latent variable models
- Achieved improved sample efficiency and final task performance
    - Four DeepMind Control Suite tasks and four OpenAI Gym tasks
    - Simulation on robotic manipulation tasks (9-DoF 3-fingered DClaw robot on four tasks)

Limitations

- For **fairness**, performance evaluations for other models seems necessary
  - not just SLAC RL framework, compare on different latent variable models
- States benefits of using two layers of latent variables
  - Insufficient explanation on why it brings **good balance**
- **Reward function** choice for simulated robotics tasks **are not well explained**
- Insufficient explanation on weak performances of SAC from true states on three random goal setting (refer to previous slide)
- Performance on other image-based continuous control tasks

Appendix A (reward functions)

C Reward Functions for the Manipulation Tasks

Sawyer Door Open

$$r = -|\theta_{\text{door}} - \theta_{\text{desired}}| \tag{12}$$

Sawyer Drawer Close

$$\begin{aligned} d_{\text{handToHandle}} &= ||xyz_{\text{hand}} - xyz_{\text{handle}}||_2^2 \\ d_{\text{drawerToGoal}} &= |x_{\text{drawer}} - x_{\text{goal}}| \\ r &= -d_{\text{handToHandle}} - d_{\text{drawerToGoal}} \end{aligned} \tag{13}$$

Sawyer Pick-up

$$\begin{aligned} d_{\text{toObj}} &= ||xyz_{\text{hand}} - xyz_{\text{obj}}||_2^2 \\ r_{\text{reach}} &= 0.25(1 - \tanh(10.0 * d_{\text{toObj}})) \\ r_{\text{lift}} &= \begin{cases} 1 & \text{if object lifted} \\ 0 & \text{else} \end{cases} \end{aligned}$$

$$r = \max(r_{\text{reach}}, r_{\text{lift}}) \tag{14}$$

Appendix B (SLAC algorithm)

Algorithm 1 Stochastic Latent Actor-Critic (SLAC)

Require: $E, \psi, \theta_1, \theta_2, \phi$ ▷ Environment and initial parameters for model, actor, and critic
  $\mathbf{x}_1 \sim E_{\text{reset}}()$ ▷ Sample initial observation from the environment
  $\mathcal{D} \leftarrow (\mathbf{x}_1)$ ▷ Initialize replay buffer with initial observation
  for each iteration **do**
    for each environment step **do**
      $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1})$ ▷ Sample action from the policy
      $r_t, \mathbf{x}_{t+1} \sim E_{\text{step}}(\mathbf{a}_t)$ ▷ Sample transition from the environment
      $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{a}_t, r_t, \mathbf{x}_{t+1})$ ▷ Store the transition in the replay buffer
    for each gradient step **do**
      $\psi \leftarrow \psi - \lambda_M \nabla_\psi J_M(\psi)$ ▷ Update model weights
      $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$ ▷ Update the Q-function weights
      $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_\pi(\phi)$ ▷ Update policy weights
      $\bar{\theta}_i \leftarrow \nu\theta_i + (1-\nu)\bar{\theta}_i$ for $i \in \{1, 2\}$ ▷ Update target critic network weights

Benchmark	Task	Action repeat	Original control time step	Effective control time step
DeepMind Control Suite	cheetah run	4	0.01	0.04
	walker walk	2	0.025	0.05
	ball-in-cup catch	4	0.02	0.08
	finger spin	2	0.02	0.04
OpenAI Gym	HalfCheetah-v2	1	0.05	0.05
	Walker2d-v2	4	0.008	0.032
	Hopper-v2	2	0.008	0.016
	Ant-v2	4	0.05	0.2

Table 1: Action repeats and the corresponding agent's control time step used in our experiments.

Log-likelihood of the observations can be bounded

$$\log p(\mathbf{x}_{1:\tau+1}|\mathbf{a}_{1:\tau}) \geq \mathop{\mathbb{E}}_{\mathbf{z}_{1:\tau+1}\sim q}\left[\sum_{t=1}^{\tau+1}\log p(\mathbf{x}_t|\mathbf{z}_t) - \mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{z}_1|\mathbf{x}_1)\parallel p(\mathbf{z}_1)\right)\right.$$
$$\left. + \sum_{t=1}^{\tau} -\mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{z}_{t+1}|\mathbf{x}_{t+1},\mathbf{z}_t,\mathbf{a}_t)\parallel p(\mathbf{z}_{t+1}|\mathbf{z}_t,\mathbf{a}_t)\right)\right]$$