Coarse-to-fine Q-Network with Action Sequence for Data-Efficient Robot Learning

Younggyo Seo Pieter Abbeel
University of California, Berkeley
mail@younggyo.me

Abstract

In reinforcement learning (RL), we train a value function to understand the long-term consequence of executing a single action. However, the value of taking each action can be ambiguous in robotics as robot movements are typically the aggregate result of executing multiple small actions. Moreover, robotic training data often consists of noisy trajectories, in which each action is noisy but executing a series of actions results in a meaningful robot movement. This further makes it difficult for the value function to understand the effect of individual actions. To address this, we introduce Coarse-to-fine Q-Network with Action Sequence (CQN-**AS**), a novel value-based RL algorithm that learns a critic network that outputs Q-values over *a sequence of actions*, i.e., explicitly training the value function to learn the consequence of executing action sequences. We study our algorithm on 53 robotic tasks with sparse and dense rewards, as well as with and without demonstrations, from BiGym, HumanoidBench, and RLBench. We find that CQN-**AS** outperforms various baselines, in particular on humanoid control tasks.
Project webpage: younggyo.me/cqn-as

1. Introduction

Reinforcement learning (RL) holds the promise of continually improving policies through online experiences (Sutton & Barto, 2018). However, training RL agents on robotic tasks often requires a prohibitively large number of training samples (Kalashnikov et al., 2018; Herzog et al., 2023), which is problematic as deploying robots incurs a huge cost.

We posit that one cause for the poor data-efficiency of RL for robotics is the credit assignment problem (Minsky, 1961). In value-based RL, we typically train a value function to understand the long-term consequences of taking individual actions. However, this becomes challenging in robotics as robot movements are the aggregate result of a sequence of low-level actions, where the effect of each action is often ambiguous. For instance, moving a robot arm forward involves multiple steps of changing joint angles. While one may easily understand the consequence of executing a series of actions, it is often difficult to learn how each low-level action, such as changing the joint angle by certain degrees, contributes to the resulting movement.

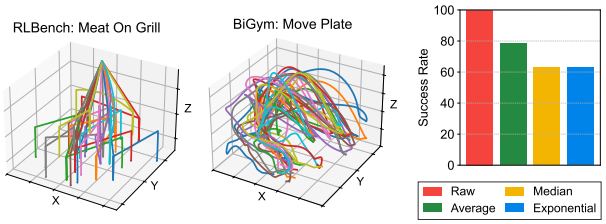


Figure 1. **Noisy trajectories in robotic data.** We visualize the (x, y, z) coordinates of a gripper in demonstrations from RLBench (James et al., 2020) that uses motion-planning for generating demonstrations and BiGym (Chernyadev et al., 2024) that provides human-collected demonstrations. (Right) We report the success rates of replaying BiGym demonstrations with various action smoothing schemes. We find that naïve approach of smoothing actions can make demonstrations be invalid as smoothed actions often lose precision, highlighting the need for developing RL algorithms that can learn from noisy robotic training data.

Moreover, robotic training data often consists of *noisy* trajectories, amplifying the credit assignment problem. For instance, we often initialize training with human-collected demonstrations that consist of noisy multi-modal trajectories (Chernyadev et al., 2024). In such trajectories, each action may be noisy because of human errors; however, executing a sequence of actions results in meaningful robot movement. One may think that this issue can be resolved by smoothing actions, but it often makes demonstrations be invalid by reducing action precision (see Figure 1). Moreover, when training RL agents, we typically inject some noise into actions for exploration (Sehnke et al., 2010; Lillicrap et al., 2016), which induces trajectories with jerky motions. This characteristic of robotic training data further makes it difficult to learn the value function that can properly evaluate the long-term consequence of each individual action.

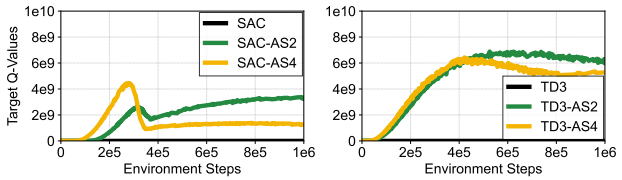


Figure 2. **Training actor-critic algorithms with action sequences leads to value overestimation.** We report target Q-values recorded throughout training actor-critic algorithms, i.e., SAC (Haarnoja et al., 2018) and TD3 (Fujimoto et al., 2018), with action sequence on `stand` task from HumanoidBench (Sferrazza et al., 2024).

On the other hand, recent behavior cloning (BC; Pomerleau 1988) approaches have shown that *predicting a sequence of actions* enables policies to effectively approximate the noisy, multi-modal distribution of expert demonstrations (Zhao et al., 2023; Chi et al., 2023). Inspired by this, we investigate whether we can similarly train a value function to explicitly learn the consequence of executing action sequences.

As a straightforward implementation of this idea, in Figure 2, we train actor-critic algorithms (Haarnoja et al., 2018; Fujimoto et al., 2018) with action sequence on `stand` task from HumanoidBench (Sferrazza et al., 2024). Specifically, we train the actor to output action sequence and the critic to take action sequence as inputs instead of single-step actions. Unfortunately, we observe that actor-critic algorithms with action sequence suffer from a severe value overestimation. This is because a wider action space makes the critic more vulnerable to function approximation error (Fujimoto et al., 2018) and the actor can easily exploit the estimation error.

These results motivate us to design our RL algorithm with action sequence upon a recent value-based algorithm, i.e., Coarse-to-fine Q-Network (CQN; Seo et al. 2024), which solves continuous control tasks with discrete actions. We find that training this critic-only algorithm with action sequence enables us to achieve the benefits of RL with action sequence yet avoid the value overestimation problem. In particular, we introduce Coarse-to-fine Q-Network with **A**ction **S**equence (CQN-**AS**), which learns a critic network that outputs *Q-values over a sequence of actions* (see Figure 3). By training the critic network to explicitly learn the consequence of taking a sequence of current and future actions, CQN-**AS** enables the RL agents to effectively learn useful value functions on challenging robotic tasks.

We show that CQN-**AS** outperforms various baselines on diverse setups with sparse and dense rewards, as well as with or without demonstrations: (i) mobile bi-manual manipulation tasks from BiGym (Chernyadev et al., 2024) that provides human-collected demonstrations, (ii) densely-rewarded humanoid control tasks from HumanoidBench (Sferrazza et al., 2024), and (iii) table-top manipulation tasks from RLBench (James et al., 2020) that provides synthetic demonsrations generated via motion-planning.

2. Preliminaries

Problem setup We formulate a robotic control problem as a partially observable Markov decision process (Kaelbling et al., 1998; Sutton & Barto, 2018). At time step t, an RL agent encounters an observation $\mathbf{o}_t$, executes an action a_t, receives a reward r_{t+1}, and encounters a new observation $\mathbf{o}_{t+1}$ from the environment. We aim to train a policy π that maximizes the expected sum of rewards through RL while using as few online samples as possible, optionally with access to a modest amount of expert demonstrations.

Inputs and encoding Given visual observations $\mathbf{o}_t^v = \{\mathbf{o}_t^{v_1}, ..., \mathbf{o}_t^{v_M}\}$ from M cameras, we encode each $\mathbf{o}_t^{v_i}$ using convolutional neural networks (CNN) into $\mathbf{h}_t^{v_i}$. We then process them through a series of linear layers to fuse them into $\mathbf{h}_t^v$. If low-dimensional observations $\mathbf{o}_t^{\texttt{low}}$ are available along with visual observations, we process them through a series of linear layers to obtain $\mathbf{h}_t^{\texttt{low}}$. We then use concatenated features $\mathbf{h}_t = [\mathbf{h}_t^v, \mathbf{h}_t^{\texttt{low}}]$ as inputs to the critic network. In domains without vision sensors, we simply use $\mathbf{o}_t^{\texttt{low}}$ as $\mathbf{h}_t$ without encoding the low-dimensional observations.

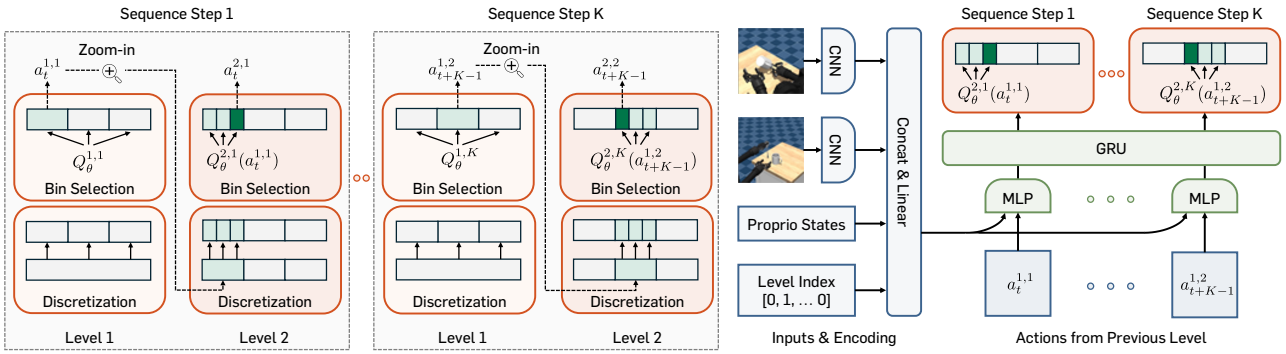
Coarse-to-fine Q-Network Coarse-to-fine Q-Network (CQN; Seo et al. 2024) is a value-based RL algorithm that solves continuous control tasks with discrete actions. The main idea of CQN is to train an RL agent to learn to select coarse discrete actions in shallower levels with larger bin sizes, and then refine their choices by selecting finer-grained actions in deeper levels with smaller bin sizes. Specifically, CQN iterates the procedures of (i) discretizing the continuous action space into multiple bins and (ii) selecting the bin with the highest Q-value to further discretize. This reformulates the continuous control problem as a multi-level discrete control problem, allowing for the use of ideas from sample-efficient discrete RL algorithms (Mnih et al., 2015; Silver et al., 2017) for continuous control.

Formally, let a_t^l be an action at level l with a_t^0 being the zero vector.[1] We then define the coarse-to-fine critic to consist of multiple Q-networks which compute Q-values for actions at each level a_t^l, given the features $\mathbf{h}_t$ and actions from the previous level a_t^{l-1}, as follows:

$$Q_\theta^l(\mathbf{h}_t, a_t^{l-1}) = \left[Q_\theta^l(\mathbf{h}_t, a_t^{l,b}, a_t^{l-1})\right]_{b=1}^{B} \in \mathbb{R}^B$$

where B is the number of bins for each level. We note that CQN uses scalar values representing the center of each bin for a_t^{l-1}, enabling the network to locate itself without access to all previous levels' actions. We optimize each Q-network at level l with the following objective:

[1]For simplicity, we describe CQN and CQN-**AS** with a single-dimensional action in the main section. See Appendix B for full description with N-dimensional actions.



(a) Coarse-to-fine inference procedure (b) Architecture

Figure 3. **Coarse-to-fine Q-Network with Action Sequence (CQN-AS).** We build our algorithm upon Coarse-to-fine Q-Network (CQN; Seo et al. 2024), a recent critic-only RL algorithm that solves continuous control tasks with discrete actions. (a) In CQN framework, we train RL agents to *zoom-into* the continuous action space by iterating the procedures of (i) discretizing continuous action space into B bins and (ii) find the bin with the highest Q-value to further discretize at the next level. We then use the last level's action sequence for controlling robots. CQN-**AS** extends this idea to action sequences by computing actions for all sequence steps $k \in [1, ..., K]$ in parallel. (b) We train a critic network to output Q-values over *a sequence of actions*. We design our architecture to first obtain features for each sequence step and aggregate features from multiple sequence steps with a recurrent network. We then project these outputs into Q-values.

$$\mathcal{L}^l = \big(Q^l_\theta(\mathbf{h}_t, a^l_t, a^{l-1}_t) - r_{t+1} - \gamma \max_{a'} Q^l_{\bar{\theta}}(\mathbf{h}_{t+1}, a', \pi^l(\mathbf{h}_{t+1}))\big),$$

where $\bar{\theta}$ are delayed parameters for a target network (Polyak & Juditsky, 1992) and π^l is a policy that outputs the action a^l_t at each level l via the inference steps with our critic, *i.e.*, $\pi^l(\mathbf{h}_t) = a^l_t$. Specifically, to output actions at time step t, CQN first initializes constants $a^{\texttt{low}}_t$ and $a^{\texttt{high}}_t$ with -1 and 1. Then the following steps are repeated for $l \in \{1, ..., L\}$:

- Step 1 (Discretization): Discretize an interval $[a^{\texttt{low}}_t, a^{\texttt{high}}_t]$ into B uniform intervals, and each of these intervals become an action space for Q^l_θ
- Step 2 (Bin selection): Find a bin with the highest Q-value and set a^l_t to the centroid of the bin.
- Step 3 (Zoom-in): Set $a^{\texttt{low}}_t$ and $a^{\texttt{high}}_t$ to the minimum and maximum of the selected bin, which intuitively can be seen as zooming-into each bin.

We then use the last level's action a^L_t as the action at time step t. For more details, including the inference procedure for computing Q-values, we refer readers to Appendix B.

3. Method

We present Coarse-to-fine Q-Network with **A**ction **S**equence (CQN-**AS**), a value-based RL algorithm that learns a critic network that outputs Q-values for *a sequence of actions* $a_{t:t+K} = \{a_t, ..., a_{t+K-1}\}$ for a given observation $\mathbf{o}_t$. Our main motivation comes from one of the key ideas in recent BC approaches: predicting *action sequences*, which helps resolve ambiguity when approximating noisy distributions of expert demonstrations (Zhao et al., 2023; Chi et al., 2023). Similarly, by explicitly learning Q-values of a sequence of actions from the given state, our approach mitigates the challenge of learning Q-values with noisy trajectories. We provide the overview of CQN-**AS** in Figure 3.

3.1. Coarse-to-fine Critic with Action Sequence

Objective Let $a^l_{t:t+K} = \{a^l_t, ..., a^l_{t+K-1}\}$ be an action sequence at level l and $a^0_{t:t+K}$ be a zero vector. We design our coarse-to-fine critic network to consist of multiple Q-networks that compute Q-values for each action at sequence step $k \in \{1, ..., K\}$ and level $l \in \{1, ..., L\}$:

$$Q^{l,k}_\theta(\mathbf{h}_t, a^{l-1}_{t:t+K}) = \Big[Q^{l,k}_\theta(\mathbf{h}_t, a^{l,b}_{t+k-1}, a^{l-1}_{t:t+K})\Big]^B_{b=1}$$

where B is the number of bins for each level. We optimize our critic network with the following objective:

$$\sum_k \sum_l \Big(Q^{l,k}_\theta(\mathbf{h}_t, a^l_{t+k-1}, a^{l-1}_{t:t+K}) - \sum_{i=1}^N r_{t+i} - \gamma \max_{a'} Q^{l,k}_{\bar{\theta}}(\mathbf{h}_{t+1}, a', \pi^l_K(\mathbf{h}_{t+1}))\Big)^2, \quad (1)$$

where N is a hyperparameter for n-step return and π^l_K is an action sequence policy that outputs the action sequence $\mathbf{a}^l_{t:t+K}$ by following the similar inference procedure as in Section 2 (see Figure 3a). In practice, we compute Q-values for all sequence step $k \in \{1, ..., K\}$ in parallel, which is possible as Q-values for future actions depend only on features $\mathbf{h}_t$ but not on previous actions.

Remarks on objective with N-step return We note that any N-return can be used in Equation 1 because the network can learn the long-term value of outputting action a_{t+k} from bootstrapping. There is a trade-off: if one considers a short N-step return, it can cause a challenge as the setup

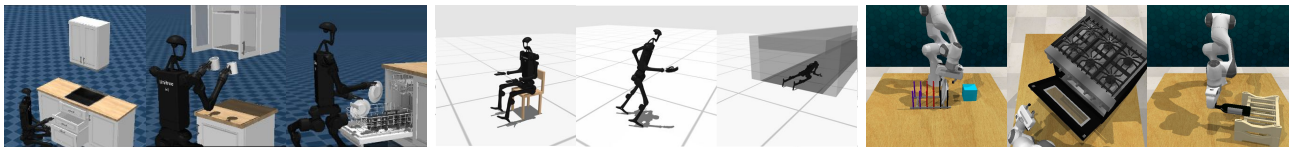


Figure 4. **Examples of robotic tasks.** We study CQN-**AS** on 53 tasks spanning mobile bi-manual manipulation, whole-body control, and tabletop manipulation from BiGym (Chernyadev et al., 2024), HumanoidBench (Sferrazza et al., 2024), and RLBench (James et al., 2020).

becomes a delayed reward setup; but training with higher N-step return may introduce variance (Sutton & Barto, 2018). In our considered setups, we empirically find that using common values $N \in \{1, 3\}$ works the best. We provide empirical analysis on the effect of N in Figure 8c.

Architecture Our critic network initially extracts features for each sequence step k and aggregates features from multiple steps with a recurrent network (see Figure 3b). This architecture is helpful in cases where a single-step action is already high-dimensional so that concatenating them make inputs too high-dimensional. Specifically, let $\mathbf{e}_k$ denote an one-hot encoding for k. At each level l, we construct features for each sequence step k as $\mathbf{h}^l_{t,k} = \left[\mathbf{h}_t, a^{l-1}_{t+k-1}, \mathbf{e}_k\right]$. We then encode each $\mathbf{h}^l_{t,k}$ with a shared MLP network and process them through GRU (Cho et al., 2014) to obtain $\mathbf{s}^l_{t,k} = f^{\texttt{GRU}}_\theta(f^{\texttt{MLP}}_\theta(\mathbf{h}^l_{t,1}), ..., f^{\texttt{MLP}}_\theta(\mathbf{h}^l_{t,k})$. We find that this design empirically performs better than directly giving actions as inputs to GRU. We then use a shared projection layer to map each $\mathbf{s}^l_{t,k}$ into Q-values at each sequence step k, *i.e.,* $Q^{l,k}_\theta(\mathbf{o}_t, a^l_{t+k-1}, a^{l-1}_{t:t+K}) = f^{\texttt{proj}}_\theta(\mathbf{s}^l_{t,k})$.

3.2. Action Execution and Training Details

Executing action with temporal ensemble With the policy that outputs an action sequence $a_{t:t+K}$, one question is how to execute actions at time step $i \in \{t, ..., t+K-1\}$. For this, we use *temporal ensemble* (Zhao et al., 2023) that computes $a_{t:t+K}$ every time step, saves it to a buffer, and executes a weighted average $\sum_i w_i \bar{a}^i_t / \sum w_i$ where $\bar{a}^i_t$ denotes an action for step t computed at step $t-i$, $w_i = \exp(-m*i)$ denotes a weight that assigns higher value to more recent actions, with m as a hyperparameter that adjusts the weighting magnitude. We find this scheme outperforms the alternative of computing $a_{t:t+K}$ every K steps and executing each action for subsequent K steps on most tasks we considered, except on several tasks that need reactive control.

Storing training data When storing samples from the environment, we store a transition $(\mathbf{o}_t, \hat{a}_t, r_{t+1}, \mathbf{o}_{t+1})$ where $\hat{a}_t$ denotes an action executed at time step t. For instance, if we use temporal ensemble for action execution, $\hat{a}_t$ is a weighted average of action outputs obtained from previous K time steps, i.e., $\hat{a}_t = \sum_i w_i \bar{a}^i_t / \sum w_i$.

Sampling training data from a replay buffer When sampling training data from the replay buffer, we sample a transition with action sequence, *i.e.,* $(\mathbf{o}_t, \hat{a}_{t:t+K}, r_{t+1}, \mathbf{o}_{t+1})$. If we sample time step t near the end of episode so that we do not have enough data to construct a full action sequence, we fill the action sequence with *null* actions. In particular, in position control where we specify the position of joints or end effectors, we repeat the action from the last step so that the agent learns not to change the position. In torque control where we specify the force to apply, we set the action after the last step to zero so that agent learns to not to apply force.

4. Experiment

We study CQN-**AS** on 53 robotic tasks spanning mobile bi-manual manipulation, whole-body control, and tabletop manipulation tasks from BiGym (Chernyadev et al., 2024), HumanoidBench (Sferrazza et al., 2024), and RLBench (James et al., 2020) environments (see Figure 4 for examples of robotic tasks). These tasks with sparse and dense rewards, with or without vision sensors, and with or without demonstrations, allow for evaluating the capabilities and limitations of our algorithm. In particular, our experiments are designed to investigate the following questions:

- Can CQN-**AS** quickly match the performance of a recent BC algorithm (Zhao et al., 2023) and surpass it through online learning? How does CQN-**AS** compare to previous model-free RL algorithms (Haarnoja et al., 2018; Yarats et al., 2022; Seo et al., 2024)?
- What is the effect of each component in CQN-**AS**?
- Under which conditions is CQN-**AS** effective?

Baselines for tasks with demonstrations For manipulation tasks from BiGym and RLBench, we consider model-free RL baselines that learn deterministic policies, as we find that stochastic policies struggle to solve such fine-grained control tasks. Specifically, we consider (i) Coarse-to-fine Q-Network (CQN; Seo et al. 2024), our backbone algorithm and (ii) DrQ-v2+, an optimized demo-driven variant of an actor-critic algorithm DrQ-v2 (Yarats et al., 2022) that uses a deterministic policy algorithm and data augmentation. We further consider (iii) Action Chunking Transformer (ACT; Zhao et al. 2023), a BC algorithm that trains a transformer (Vaswani et al., 2017) policy to predict action sequence and utilizes temporal ensemble, as our BC baseline.

Baselines for humanoid control tasks with dense reward For HumanoidBench tasks, we consider (i) Soft Actor-Critic (SAC; Haarnoja et al. 2018), a model-free actor-critic RL algorithm that maximizes action entropy and (ii) CQN.

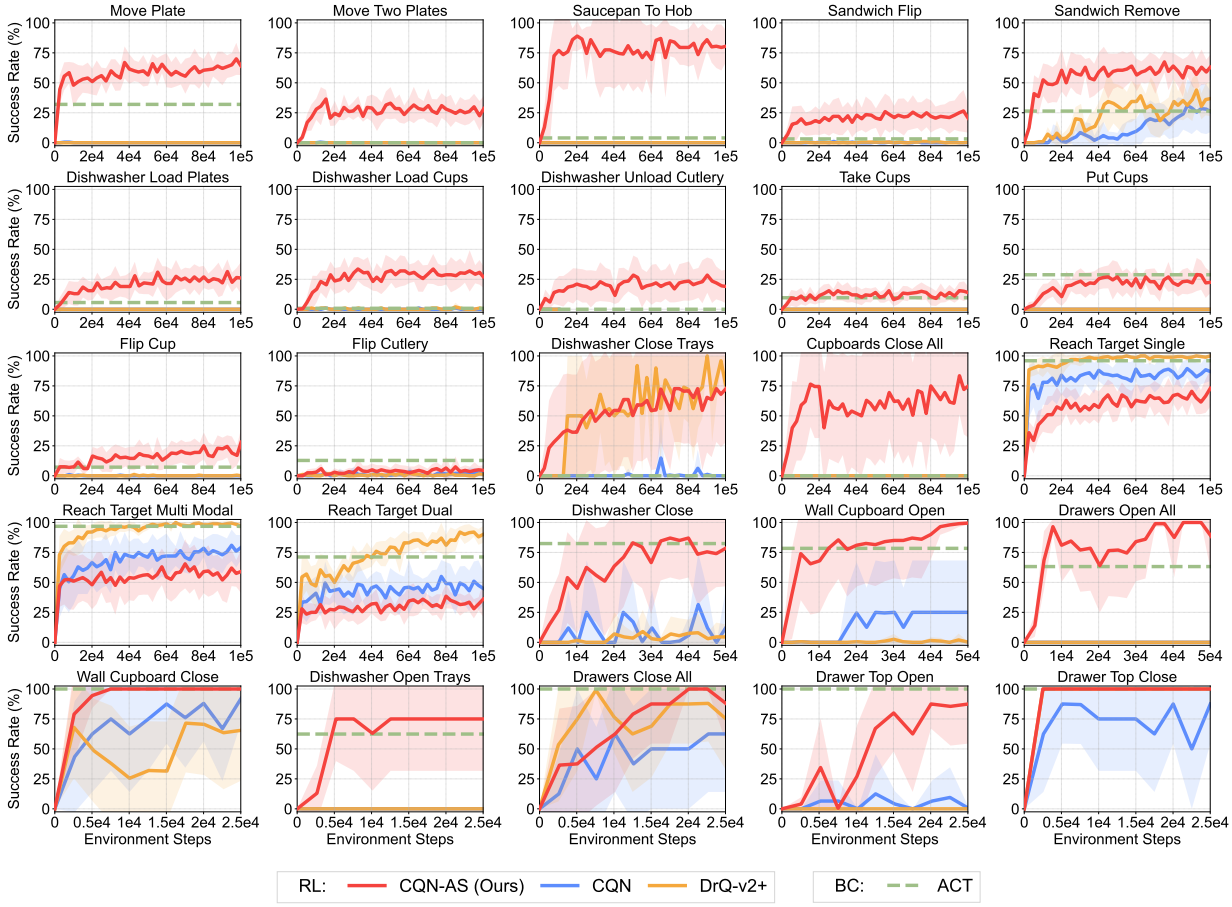


Figure 5. **BiGym results** on 25 sparsely-rewarded mobile bi-manual manipulation tasks. All experiments are initialized with 17 to 60 *human-collected* demonstrations, and RL methods are trained with an auxiliary BC objective. On many tasks, CQN-**AS** quickly matches the performance of ACT (Zhao et al., 2023) and surpasses it through online learning, also outperforming other RL baselines. We report the success rate over 25 episodes. The solid line and shaded regions represent the mean and confidence intervals, respectively, across 8 runs.

Implementation details For training with expert demonstrations, we follow the setup of Seo et al. (2024). We keep a separate replay buffer that stores demonstrations and sample half of training data from demonstrations. We also relabel successful online episodes as demonstrations and store them in the demonstration replay buffer. For CQN-**AS**, we use an auxiliary BC loss from Seo et al. (2024) based on large margin loss (Hester et al., 2018). For actor-critic baselines, we use an auxiliary BC loss that minimizes L2 loss between the policy outputs and expert actions.

4.1. BiGym Experiments

We study CQN-**AS** on mobile bi-manual manipulation tasks from BiGym (Chernyadev et al., 2024). BiGym's *human-collected* demonstrations are often noisy and multi-modal, posing challenges to RL algorithms which should leverage demonstrations for solving sparsely-rewarded tasks.

Setup We consider 25 BiGym tasks with 17 to 60 demonstrations. We use RGB observations with 84×84 resolution from `head`, `left_wrist`, and `right_wrist` cameras. We also use low-dimensional proprioceptive states. We use (i) absolute joint position control action mode and (ii) floating base that replaces locomotion with classic controllers. We use the same set of hyperparameters for all the tasks. Details on BiGym experiments are available in Appendix A.

Comparison to baselines Figure 5 shows that CQN-**AS** quickly matches the performance of ACT and outperforms it through online learning on most tasks, while other RL algorithms fail to do so especially on challenging long-horizon tasks such as `Move Plate` and `Saucepan To Hob`. A notable result here is that CQN-**AS** *enables* solving challenging BiGym tasks while other RL baselines completely fail as they achieve 0% success rate on many tasks.

Limitation However, CQN-**AS** struggles to achieve meaningful success rate on some of the long-horizon tasks that require interaction with delicate objects such as cup or cutlery. This leaves room for future work to incorporate advanced vision encoders (He et al., 2016) or critic architectures (Chebotar et al., 2023; Springenberg et al., 2024).

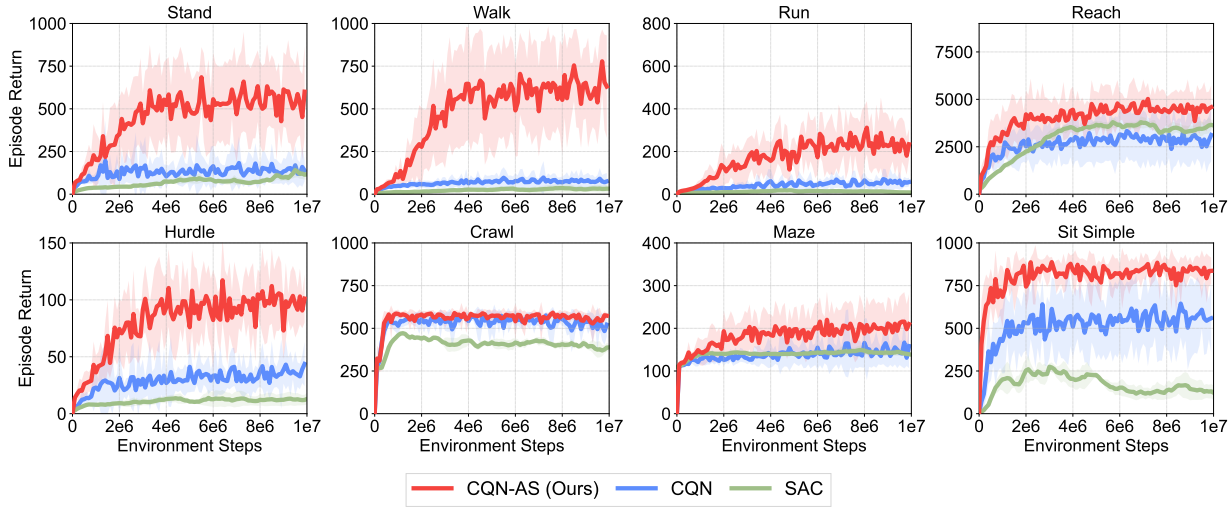


Figure 6. **HumanoidBench results** on 8 densely-rewarded humanoid control tasks (Sferrazza et al., 2024). All the experiments start from scratch and RL methods do not have an auxiliary BC objective. CQN-**AS** significantly outperforms other model-free RL baselines on most tasks. For CQN-**AS** and CQN, we report the results aggregated over 8 runs. For SAC, we report the results aggregated over 3 runs available from public website. The solid line and shaded regions represent the mean and confidence intervals.

4.2. HumanoidBench Experiments

To show that CQN-**AS** is generally applicable to tasks without demonstrations, we study CQN-**AS** on densely-rewarded tasks from HumanoidBench (Sferrazza et al., 2024).

Setup We follow a standard setup that trains RL agents from scratch. We use low-dimensional states consisting of proprioception and privileged task information as inputs. For tasks, we simply select the first 8 locomotion tasks in the benchmark. For baselines, we use the results available from the public repository, which are evaluated on *tasks with dexterous hands*, and we also evaluate our algorithm on tasks with hands. We use the same set of hyperparameters for all the tasks. More details are available in Appendix A.

Comparison to baselines Figure 6 shows that, by learning the critic network with action sequence, CQN-**AS** outperforms other model-free RL baselines, *i.e.,* CQN and SAC, on most tasks. In particular, the difference between CQN-**AS** and baselines becomes larger as the task gets more difficult, *e.g.,* baselines fail to achieve high episode return on `Walk` and `Run` tasks but CQN-**AS** achieves strong performance. This result shows that our idea of using action sequence can be applicable to generic setup without demonstrations.

4.3. RLBench Experiments

To investigate whether CQN-**AS** can also be effective in leveraging *clean* demonstrations, we study CQN-**AS** on RLBench (James et al., 2020) with synthetic demonstrations.

Setup We use the official CQN implementation for collecting demonstrations and reproducing the baseline results on the same set of tasks. We use RGB observations with 84×84 resolution from `front`, `wrist`, `left_shoulder`, and `right_shoulder` cameras. We also use low-dimensional proprioceptive states consisting of 7-dimensional joint positions and a binary value for gripper open. We use 100 demonstrations and delta joint position control action mode. We use the same set of hyperparameters for all the tasks, in particular, we use action sequence of length 4. More details on RLBench experiments are available in Appendix A.

CQN-AS is also effective with *clean* demonstrations Because RLBench provides synthetic *clean* demonstrations, as we expected, Figure 7 shows that CQN-**AS** achieves *similar* performance to CQN on most tasks, except 2/25 tasks where it hurts the performance. But we still find that CQN-**AS** achieves quite superior performance to CQN on some challenging long-horizon tasks such as `Open Oven` or `Take Plate Off Colored Dish Rack`. These results show that CQN-**AS** can be used in various benchmark with different characteristics.

4.4. Ablation Studies, Analysis, Failure Cases

Effect of action sequence length Figure 8a shows the performance of CQN-**AS** with different action sequence lengths. We find that training the critic network with longer action sequences improves performance.

RL objective is crucial for strong performance Figure 8b shows the performance of CQN-**AS** without RL objective that trains the model only with BC objective on successful demonstrations. We find this baseline significantly underperforms CQN-**AS**, which shows that RL objective enables the agent to learn from trial-and-error experiences.

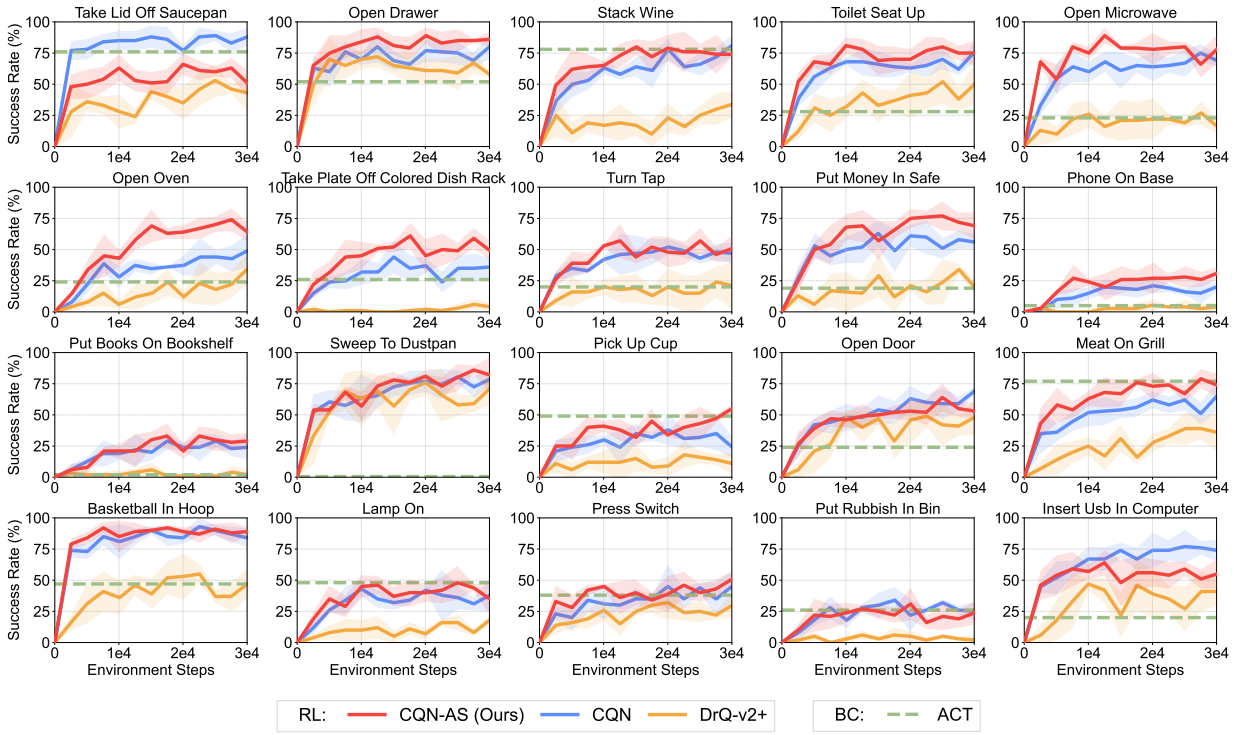


Figure 7. **RLBench results** on 20 sparsely-rewarded tabletop manipulation tasks from RLBench (James et al., 2020). All experiments are initialized with 100 *synthetic* demonstrations generated via motion-planning and RL methods are trained with an auxiliary BC objective. *As expected*, with synthetic demonstrations, CQN-**AS** achieves *similar* performance to CQN on most tasks. However, CQN-**AS** often significantly outperforms baselines on several challenging, long-horizon tasks such as `Open Oven`. We report the success rate over 25 episodes. The solid line and shaded regions represent the mean and confidence intervals, respectively, across 4 runs.

Effect of N-step return Figure 8c shows experimental results with varying N-step returns. We find that too high N-step return significantly degrades performance. We hypothesize this is because the variance from N-step return makes it difficult to learn useful value functions.

Effect of temporal ensemble Figure 8d shows that performance degrades without temporal ensemble on `Saucepan To Hop` as temporal ensemble induces a smooth motion and thus improves performance in fine-grained control tasks. But we also find that temporal ensemble can be harmful on `Reach Target Single`. We hypothesize this is because temporal ensemble uses predictions from previous steps and thus makes it difficult to refine behaviors based on recent visual observations. Nonetheless, we use temporal ensemble for all the tasks as we find it helps on most tasks and we aim to use the same set of hyperparameters.

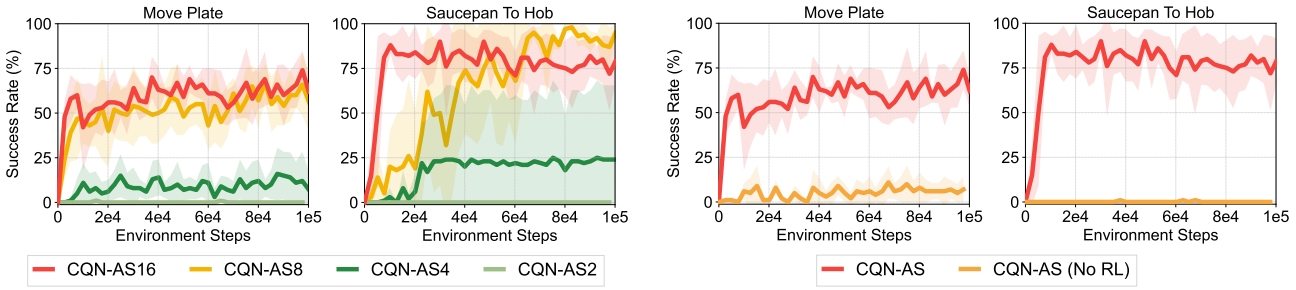
Effect of temporal ensemble magnitude We further provide results with different temporal ensemble magnitudes by adjusting a hyperparameter m in Figure 8e. Here, higher m puts higher weights on recent actions and thus very high m corresponds to using only first action. Similarly to previous paragraph, we find that higher m leads to better performance on `Reach Target Single` that needs fast reaction, but degrades performance on `Saucepan To Hop`.

Failure case: Torque control Figure 8f shows that CQN-**AS** underperforms CQN on locomotion tasks with torque control. We hypothesize this is because a sequence of joint positions usually has a semantic meaning in joint spaces, making it easier to learn with, when compared to learning how to apply a sequence of torques. Addressing this failure case is an interesting future direction.

5. Related Work

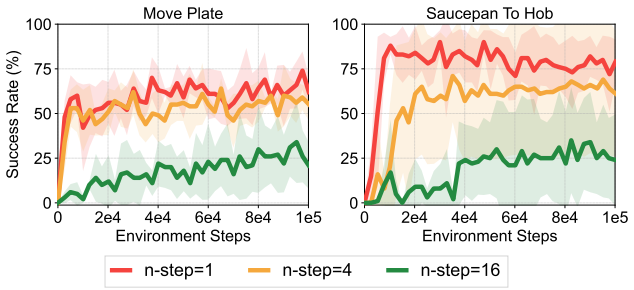
Behavior cloning with action sequence Recent behavior cloning approaches have shown that predicting a sequence of actions enables the policy to imitate noisy expert trajectories and helps in dealing with idle actions from human pauses during data collection (Zhao et al., 2023; Chi et al., 2023). Zhao et al. (2023) train a transformer model (Vaswani et al., 2017) that predicts action sequence and Chi et al. (2023) train a denoising diffusion model (Ho et al., 2020) that approximates the action distributions. This idea has been extended to multi-task setup (Bharadhwaj et al., 2024), mobile manipulation (Fu et al., 2024b) and humanoid control (Fu et al., 2024a). Our work is inspired by this line of work and proposed to learn RL agents with action sequence.

Reinforcement learning with action sequence In the context of RL, Medini & Shrivastava (2019) propose to

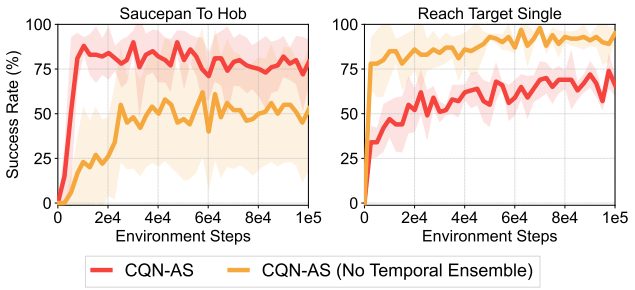


(a) Effect of action sequence length

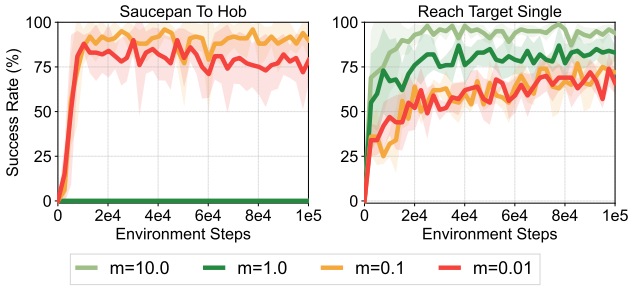
(b) Effect of RL objective



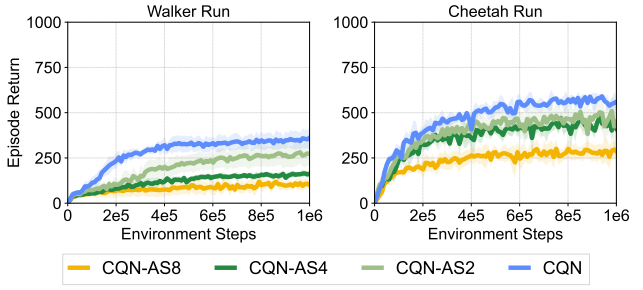
(c) Effect of N-step



(d) Effect of temporal ensemble



(e) Effect of temporal ensemble magnitude m



(f) Failure mode: Torque control

Figure 8. **Ablation studies and analysis** on the effect of (a) action sequence, (b) RL objective, (c) N-step return, and (d & e) temporal ensemble. (f) We also provide results on locomotion tasks from DeepMind Control Suite (Tassa et al., 2020), where CQN-**AS** fails to improve performance. The solid line and shaded regions represent the mean and confidence intervals, respectively, across 4 runs.

pre-compute frequent action sequences from expert demonstrations and augment the action space with these sequences. However, this idea introduces additional complexity and is not scalable to setups without demonstrations. One recent work relevant to ours is Saanum et al. (2024) that encourage a sequence of actions from RL agents to be predictable and smooth. But this differs from our work in that it uses action sequence only for computing the penalty term. Recently, Ankile et al. (2024) point out that RL with action sequence is challenging and instead propose to use RL for learning a single-step policy that corrects action sequence predictions from BC. In contrast, we show that RL with action sequence is feasible and improves performance of RL algorithms.

6. Conclusion

In this paper, we presented Coarse-to-fine Q-Network with **A**ction **S**equence (CQN-**AS**), a value-based RL algorithm that trains a critic network that outputs Q-values over action sequences. Extensive experiments in benchmarks with various setups show that our idea not only improves the performance of the base algorithm but also allows for solving complex tasks where prior RL algorithms completely fail.

We believe our work will be strong evidence that shows RL can realize its promise to develop robots that can continually improve through online trial-and-error experiences, surpassing the performance of BC approaches. We are excited about future directions, including real-world RL with humanoid robots, incorporating advanced critic architectures (Kapturowski et al., 2023; Chebotar et al., 2023; Springenberg et al., 2024), bootstrapping RL agents from imitation learning (Hu et al., 2023; Xing et al., 2024) or offline RL (Nair et al., 2020; Lee et al., 2021), extending the idea to recent model-based RL approaches (Hafner et al., 2023; Hansen et al., 2024), fine-tuning vision-language-action models that use action sequence (Team et al., 2024; Doshi et al., 2024; Kim et al., 2024) with our algorithm, to name but a few.

Acknowledgements

We thank Stephen James and Richie Lo for the discussion on the initial idea of this project. This work was supported in part by Multidisciplinary University Research Initiative (MURI) award by the Army Research Office (ARO) grant No. W911NF-23-1-0277, an ONR DURIP grant, and BAIR Industrial Consortium. Pieter Abbeel holds concurrent appointments as a Professor at UC Berkeley and as an Amazon Scholar. This paper describes work performed at UC Berkeley and is not associated with Amazon.

References

Ankile, L., Simeonov, A., Shenfeld, I., Torne, M., and Agrawal, P. From imitation to refinement–residual rl for precise visual assembly. *arXiv preprint arXiv:2407.16677*, 2024.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, 2017.

Bharadhwaj, H., Vakil, J., Sharma, M., Gupta, A., Tulsiani, S., and Kumar, V. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

Chebotar, Y., Vuong, Q., Hausman, K., Xia, F., Lu, Y., Irpan, A., Kumar, A., Yu, T., Herzog, A., Pertsch, K., et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, 2023.

Chernyadev, N., Backshall, N., Ma, X., Lu, Y., Seo, Y., and James, S. Bigym: A demo-driven mobile bi-manual manipulation benchmark. In *Conference on Robot Learning*, 2024.

Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Doshi, R., Walke, H., Mees, O., Dasari, S., and Levine, S. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *Conference on Robot Learning*, 2024.

Fu, Z., Zhao, Q., Wu, Q., Wetzstein, G., and Finn, C. Humanplus: Humanoid shadowing and imitation from humans. In *Conference on Robot Learning*, 2024a.

Fu, Z., Zhao, T. Z., and Finn, C. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning*, 2024b.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Hansen, N., Su, H., and Wang, X. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Herzog, A., Rao, K., Hausman, K., Lu, Y., Wohlhart, P., Yan, M., Lin, J., Arenas, M. G., Xiao, T., Kappler, D., et al. Deep rl at scale: Sorting waste in office buildings with a fleet of mobile manipulators. *arXiv preprint arXiv:2305.03270*, 2023.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 2020.

Hu, H., Mirchandani, S., and Sadigh, D. Imitation bootstrapped reinforcement learning. *arXiv preprint arXiv:2311.02198*, 2023.

James, S., Freese, M., and Davison, A. J. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

James, S., Ma, Z., Arrojo, D. R., and Davison, A. J. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 1998.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, 2018.

Kapturowski, S., Campos, V., Jiang, R., Rakićević, N., van Hasselt, H., Blundell, C., and Badia, A. P. Human-level atari 200x faster. In *International Conference on Learning Representations*, 2023.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., et al. Openvla: An open-source vision-language-action model. In *Conference on Robot Learning*, 2024.

Lee, S., Seo, Y., Lee, K., Abbeel, P., and Shin, J. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, 2021.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Medini, T. and Shrivastava, A. Mimicking actions is a good strategy for beginners: Fast reinforcement learning with expert action sequences, 2019. URL https://openreview.net/forum?id=HJfxbhR9KQ.

Minsky, M. Steps toward artificial intelligence. *Proceedings of the IRE*, 1961.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Nair, A., Gupta, A., Dalal, M., and Levine, S. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Polyak, B. T. and Juditsky, A. B. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 1992.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, 1988.

Rohmer, E., Singh, S. P., and Freese, M. V-rep: A versatile and scalable robot simulation framework. In *IEEE/RSJ international conference on intelligent robots and systems*, 2013.

Saanum, T., Éltető, N., Dayan, P., Binz, M., and Schulz, E. Reinforcement learning with simple sequence priors. *Advances in Neural Information Processing Systems*, 2024.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

Seo, Y., Uruç, J., and James, S. Continuous control with coarse-to-fine reinforcement learning. In *Conference on Robot Learning*, 2024.

Seyde, T., Werner, P., Schwarting, W., Gilitschenski, I., Riedmiller, M., Rus, D., and Wulfmeier, M. Solving continuous control via q-learning. In *International Conference on Learning Representations*, 2023.

Sferrazza, C., Huang, D.-M., Lin, X., Lee, Y., and Abbeel, P. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. In *Robotics: Science and Systems*, 2024.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 2017.

Springenberg, J. T., Abdolmaleki, A., Zhang, J., Groth, O., Bloesch, M., Lampe, T., Brakel, P., Bechtle, S., Kapturowski, S., Hafner, R., et al. Offline actor-critic reinforcement learning scales to large models. In *International Conference on Machine Learning*, 2024.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tassa, Y., Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., and Heess, N. dm_control: Software and tasks for continuous control. *arXiv preprint arXiv:2006.12983*, 2020.

Team, O. M., Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Kreiman, T., Xu, C., et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 2016.

Xing, J., Romero, A., Bauersfeld, L., and Scaramuzza, D. Bootstrapping reinforcement learning with imitation for vision-based agile flight. In *Conference on Robot Learning*, 2024.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2022.

Zhao, T. Z., Kumar, V., Levine, S., and Finn, C. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.

A. Experimental Details

BiGym BiGym[2] (Chernyadev et al., 2024) is built upon MuJoCo (Todorov et al., 2012). We use Unitree H1 with two parallel grippers. We find that demonstrations available in the recent version of BiGym are not all successful. Therefore we adopt the strategy of replaying all the demonstrations and only use the successful ones as demonstrations. instead of discarding the failed demonstrations, we still store them in a replay buffer as failure experiences. To avoid training with too few demonstrations, we exclude the tasks where the ratio of successful demonstrations is below 50%. Table 1 shows the list of 25 sparsely-rewarded mobile bi-manual manipulation tasks used in our experiments.

Table 1. **BiGym tasks** with their maximum episode length and number of successful demonstrations.

Task	Length	Demos	Task	Length	Demos
Move Plate	300	51	Cupboards Close All	620	53
Move Two Plates	550	30	Reach Target Single	100	30
Saucepan To Hob	440	28	Reach Target Multi Modal	100	60
Sandwich Flip	620	34	Reach Target Dual	100	50
Sandwich Remove	540	24	Dishwasher Close	375	44
Dishwasher Load Plates	560	17	Wall Cupboard Open	300	44
Dishwasher Load Cups	750	58	Drawers Open All	480	45
Dishwasher Unload Cutlery	620	29	Wall Cupboard Close	300	60
Take Cups	420	32	Dishwasher Open Trays	380	57
Put Cups	425	43	Drawers Close All	200	59
Flip Cup	550	45	Drawer Top Open	200	40
Flip Cutlery	500	43	Drawer Top Close	120	51
Dishwasher Close Trays	320	62			

HumanoidBench HumanoidBench[3] (Sferrazza et al., 2024) is built upon MuJoCo (Todorov et al., 2012). We use Unitree H1 with two dexterous hands. We consider the first 8 locomotion tasks in the benchmark: `Stand`, `Walk`, `Run`, `Reach`, `Hurdle`, `Crawl`, `Maze`, `Sit Simple`. We use proprioceptive states and privileged task information instead of visual observations. Unlike BiGym and RLBench experiments, we do not utilize dueling network (Wang et al., 2016) and distributional critic (Bellemare et al., 2017) in HumanoidBench for faster experimentation.

RLBench RLBench[4] (James et al., 2020) is built upon CoppeliaSim (Rohmer et al., 2013) and PyRep (James et al., 2019). We use a 7-DoF Franka Panda robot arm and a parallel gripper. Following the setup of Seo et al. (2024), we increase the velocity and acceleration of the arm by 2 times. For all experiments, we use 100 demonstrations generated via motion-planning. Table 2 shows the list of 20 sparsely-rewarded visual manipulation tasks used in our experiments.

Table 2. **RLBench tasks** with their maximum episode length used in our experiments.

Task	Length	Task	Length
Take Lid Off Saucepan	100	Put Books On Bookshelf	175
Open Drawer	100	Sweep To Dustpan	100
Stack Wine	150	Pick Up Cup	100
Toilet Seat Up	150	Open Door	125
Open Microwave	125	Meat On Grill	150
Open Oven	225	Basketball In Hoop	125
Take Plate Off Colored Dish Rack	150	Lamp On	100
Turn Tap	125	Press Switch	100
Put Money In Safe	150	Put Rubbish In Bin	150
Phone on Base	175	Insert Usb In Computer	100

[2] https://github.com/chernyadev/bigym
[3] https://github.com/carlosferrazza/humanoid-bench
[4] https://github.com/stepjam/RLBench

Hyperparameters We use the same set of hyperparameters across the tasks in each domain. For hyperparameters shared across CQN and CQN-**AS**, we use the same hyperparameters for both algorithms for a fair comparison. We provide detailed hyperparameters for BiGym and RLBench experiments in Table 3 and HumanoidBench experiments in Table 4

Table 3. Hyperparameters for demo-driven vision-based experiments in BiGym and RLBench

Hyperparameter	**Value**
Image resolution	$84 \times 84 \times 3$
Image augmentation	RandomShift (Yarats et al., 2022)
Frame stack	4 (BiGym) / 8 (RLBench)
CNN - Architecture	Conv (c=[32, 64, 128, 256], s=2, p=1)
MLP - Architecture	Linear (c=[512, 512, 64, 512, 512], bias=False) (BiGym) Linear (c=[64, 512, 512], bias=False) (RLBench)
CNN & MLP - Activation	SiLU (Hendrycks & Gimpel, 2016) and LayerNorm (Ba et al., 2016)
GRU - Architecture	GRU (c=[512], bidirectional=False)
Dueling network	True
C51 - Atoms	51
C51 - v_{min}, v_{max}	-2, 2
Action sequence	16 (BiGym) / 4 (RLBench)
Temporal ensemble weight m	0.01
Levels	3
Bins	5
BC loss ($\mathcal{L}_{\text{BC}}$) scale	1.0
RL loss ($\mathcal{L}_{\text{RL}}$) scale	0.1
Relabeling as demonstrations	True
Data-driven action scaling	True
Action mode	Absolute Joint (BiGym), Delta Joint (RLBench)
Exploration noise	$\epsilon \sim \mathcal{N}(0, 0.01)$
Target critic update ratio (τ)	0.02
N-step return	1
Batch size	256
Demo batch size	256
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Learning rate	5e-5
Weight decay	0.1

Computing hardware For BiGym and Humanoid experiments, we use NVIDIA A5000 GPU with 24GB VRAM. With A5000, each BiGym experiment with 100K environment steps take 16 hours, and each HumanoidBench experiment with 10M environment steps take 40 hours. For RLBench experiments, we use NVIDIA RTX 2080Ti GPU, with which each experiment with 30K environment steps take 6.5 hours. We find that CQN-**AS** is around 33% slower than running CQN because larger architecture slows down both training and inference.

Baseline implementation For CQN (Seo et al., 2024) and DrQ-v2+ (Yarats et al., 2022), we use the implementation available from the official CQN implementation[5]. For ACT (Zhao et al., 2023), we use the implementation from RoboBase repository[6]. For SAC (Haarnoja et al., 2018), DreamerV3 (Hafner et al., 2023), and TD-MPC2 (Hansen et al., 2024), we use results provided in HumanoidBench[7] repository (Sferrazza et al., 2024).

[5] https://github.com/younggyoseo/CQN
[6] https://github.com/robobase-org/robobase
[7] https://github.com/carlosferrazza/humanoid-bench

Table 4. Hyperparameters for state-based experiments in HumanoidBench

Hyperparameter	**Value**
MLP - Architecture	Linear (c=[512, 512], bias=False)
CNN & MLP - Activation	SiLU (Hendrycks & Gimpel, 2016) and LayerNorm (Ba et al., 2016)
GRU - Architecture	GRU (c=[512], bidirectional=False)
Dueling network	True
Action sequence	4
Temporal ensemble weight m	0.01
Levels	3
Bins	5
RL loss ($\mathcal{L}_{\text{RL}}$) scale	1.0
Action mode	Absolute Joint
Exploration noise	$\epsilon \sim \mathcal{N}(0, 0.01)$
Target critic update ratio (τ)	1.0
Target critic update interval (τ)	100
Update-to-data ratio (UTD)	0.5
N-step return	3
Batch size	128
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Learning rate	5e-5
Weight decay	0.1

B. Full description of CQN and CQN-AS

This section provides the formulation of CQN and CQN-**AS** with n-dimensional actions.

B.1. Coarse-to-fine Q-Network

Let $a_t^{l,n}$ be an action at level l and dimension n and $\mathbf{a}_t^l = \{a_t^{l,1}, ..., a_t^{l,N}\}$ be actions at level l with $\mathbf{a}_t^0$ being zero vector. We then define coarse-to-fine critic to consist of multiple Q-networks:

$$Q_\theta^{l,n}(\mathbf{h}_t, a_t^{l,n}, \mathbf{a}_t^{l-1}) \text{ for } l \in \{1, ..., L\} \text{ and } n \in \{1, ..., N\} \tag{2}$$

We optimize the critic network with the following objective:

$$\sum_n \sum_l \left(Q_\theta^{l,n}(\mathbf{h}_t, a_t^{l,n}, \mathbf{a}_t^{l-1}) - r_{t+1} - \gamma \max_{a'} Q_{\bar{\theta}}^{l,n}(\mathbf{h}_{t+1}, a', \pi^l(\mathbf{h}_{t+1})) \right)^2, \tag{3}$$

where $\bar{\theta}$ are delayed parameters for a target network (Polyak & Juditsky, 1992) and π^l is a policy that outputs the action $\mathbf{a}_t^l$ at each level l via the inference steps with our critic, *i.e.,* $\pi^l(\mathbf{h}_t) = \mathbf{a}_t^l$.

Action inference To output actions at time step t with the critic, CQN first initializes constants $a_t^{n,\texttt{low}}$ and $a_t^{n,\texttt{high}}$ with -1 and 1 for each n. Then the following steps are repeated for $l \in \{1, ..., L\}$:

- Step 1 (Discretization): Discretize an interval $[a_t^{n,\texttt{low}}, a_t^{n,\texttt{high}}]$ into B uniform intervals, and each of these intervals become an action space for $Q_\theta^{l,n}$
- Step 2 (Bin selection): Find the bin with the highest Q-value, set $a_t^{l,n}$ to the centroid of the selected bin, and aggregate actions from all dimensions to $\mathbf{a}_t^l$
- Step 3 (Zoom-in): Set $a_t^{n,\texttt{low}}$ and $a_t^{n,\texttt{high}}$ to the minimum and maximum of the selected bin, which intuitively can be seen as zooming-into each bin.

We then use the last level's action $\mathbf{a}_t^L$ as the action at time step t.

Computing Q-values To compute Q-values for given actions $\mathbf{a}_t$, CQN first initializes constants $a_t^{n,\texttt{low}}$ and $a_t^{n,\texttt{high}}$ with -1 and 1 for each n. We then repeat the following steps for $l \in \{1, ..., L\}$:

- Step 1 (Discretization): Discretize an interval $[a_t^{n,\texttt{low}}, a_t^{n,\texttt{high}}]$ into B uniform intervals, and each of these intervals become an action space for $Q_\theta^{l,n}$
- Step 2 (Bin selection): Find the bin that contains input action $\mathbf{a}_t$, compute $a_t^{l,n}$ for the selected interval, and compute Q-values $Q_\theta^{l,n}(\mathbf{h}_t, a_t^{l,n}, \mathbf{a}_t^{l-1})$.
- Step 3 (Zoom-in): Set $a_t^{n,\texttt{low}}$ and $a_t^{n,\texttt{high}}$ to the minimum and maximum of the selected bin, which intuitively can be seen as zooming-into each bin.

We then use a set of Q-values $\{Q_\theta^{l,n}(\mathbf{h}_t, a_t^{l,n}, \mathbf{a}_t^{l-1})\}_{l=1}^L$ for given actions $\mathbf{a}_t$.

B.2. Coarse-to-fine Critic with Action Sequence

Let $\mathbf{a}_{t:t+K}^l = \{\mathbf{a}_t^l, ..., \mathbf{a}_{t+K-1}^l\}$ be an action sequence at level l and $\mathbf{a}_{t:t+K}^0$ be zero vector. Our critic network consists of multiple Q-networks for each level l, dimension n, and sequence step k:

$$Q_\theta^{l,n,k}(\mathbf{h}_t, a_{t+k-1}^{l,n}, \mathbf{a}_{t:t+K}^{l-1}) \text{ for } \; l \in \{1, ..., L\}, \; n \in \{1, ..., N\} \text{ and } k \in \{1, ..., K\} \tag{4}$$

We optimize the critic network with the following objective:

$$\sum_n \sum_l \sum_k \left(Q_\theta^{l,n,k}(\mathbf{h}_t, a_t^{l,n}, \mathbf{a}_{t:t+K}^{l-1}) - r_{t+1} - \gamma \max_{a'} Q_{\bar{\theta}}^{l,n,k}(\mathbf{h}_{t+1}, a', \pi_K^l(\mathbf{h}_{t+1})) \right)^2, \tag{5}$$

where π_K^l is an action sequence policy that outputs the action sequence $\mathbf{a}_{t:t+K}^l$. In practice, we compute Q-values for all sequence step $k \in \{1, ..., K\}$ and all action dimension $n \in \{1, ..., N\}$ in parallel. This can be seen as extending the idea of Seyde et al. (2023), which learns decentralized Q-networks for action dimensions, into action sequence dimension. As we mentioned in Section 3.1, we find this simple scheme works well on challenging tasks with high-dimensional action spaces.

Architecture Let $\mathbf{e}_k$ denote an one-hot encoding for k. For each level l, we construct features for each sequence step k as $\mathbf{h}_{t,k}^l = \left[\mathbf{h}_t, \mathbf{a}_{t+k-1}^{l-1}, \mathbf{e}_k\right]$. We encode each $\mathbf{h}_{t,k}^l$ with a shared MLP network and process them through GRU (Cho et al., 2014) to obtain $\mathbf{s}_{t,k}^l = f_\theta^{\texttt{GRU}}(f_\theta^{\texttt{MLP}}(\mathbf{h}_{t,1}^l), ..., f_\theta^{\texttt{MLP}}(\mathbf{h}_{t,k}^l))$. We use a shared projection layer to map each $\mathbf{s}_{t,k}^l$ into Q-values at each sequence step k, *i.e.,* $\{Q_\theta^{l,k}(\mathbf{o}_t, a_{t+k-1}^{l,n}, \mathbf{a}_{t:t+K}^{l-1})\}_{n=1}^N = f_\theta^{\texttt{proj}}(\mathbf{s}_{t,k}^l)$. We note that we compute Q-values for all dimensions $n \in \{1, ..., N\}$ at the same time with a big linear layer, which follows the design of Seo et al. (2024).