

UC Berkeley

UC Berkeley Electronic Theses and Dissertations

Title

Reinforcement Learning with Action Chunking Policies

Permalink

<https://escholarship.org/uc/item/71s6q75j>

ISBN

9798189271557

Author

Li, Qiyang

Publication Date

2026-05-01

Peer reviewed|Thesis/dissertation

Reinforcement Learning with Action Chunking Policies

by

Qiyang Li

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Sergey Levine, Chair

Professor Abhishek Gupta

Professor Angjoo Kanazawa

Professor Aditi Krishnapriyan

Spring 2026

Reinforcement Learning with Action Chunking Policies

Copyright 2026

by

Qiyang Li

Abstract

Reinforcement Learning with Action Chunking Policies

by

Qiyang Li

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Sergey Levine, Chair

In robotic manipulation, action chunking, a technique that predicts and executes a sequence of actions rather than one at a time, is a powerful tool for capturing noisy and non-Markovian behavior in human data or prior experience. While action chunking policies have seen lots of successes in imitation learning, they are typically trained with supervised learning on human demonstration data that are often costly to collect. Reinforcement learning (RL) offers a promising alternative by enabling robots to autonomously collect data and continuously self-improve from a well-specified reward function, but existing RL methods rely on simple policy classes that often struggle to capture the multi-modality in prior data. In this dissertation, we discuss algorithmic and theoretical foundations for scalable RL with action chunking policies. We start by presenting practical algorithms for optimizing flow-matching policies in Q-learning. Then, we establish a theoretical framework where we identify the key open-loop consistency condition under which the learned action chunking policies are guaranteed to be near-optimal as well as the bounded optimality variability condition under which we can mitigate the open-loop bias via closed-loop execution. Inspired by these analyses, we introduce a practical algorithm that demonstrates the scalability of action chunking RL to longer-horizon tasks. Altogether, these contributions provide a foundation for scalable RL with action chunking policies and bring RL closer to real-world deployment in robotics.

To my dear parents and friends

Contents

Contents	ii
List of Figures	v
List of Tables	viii
1 Introduction	1
2 Preliminaries and Problem Definition	3
2.1 Reinforcement Learning	3
2.2 Temporal Difference Learning	3
2.3 Flow-Matching Generative Model	4
3 Related Work	7
3.1 RL with Generative Policies	7
3.2 Diffusion and Flow-Matching with Guidance	8
3.3 Offline and Offline-to-Online Reinforcement Learning	8
3.4 Action Chunking	9
3.5 Hierarchical Reinforcement Learning	10
3.6 Theoretical Analysis for Reinforcement Learning under Unobserved Confounding Variables	10
4 Q-learning with Flow-Matching Policies	12
4.1 Flow Q-Learning	13
4.2 Q-learning with Adjoint Matching	15
4.3 Q-Chunking: Leveraging Flow RL Algorithms for Action Chunking Policy Optimization	21
4.4 Comprehensive Evaluations for Flow RL Methods	29
4.5 Concluding Remark	32
5 Theoretical Foundation of Action Chunking Q-learning	34
5.1 Assumptions and Notations	35
5.2 Weak and Strong Open-loop consistency (OLC)	36

5.3	Comparing to n -step Return Q-learning	40
5.4	Closed-loop Execution of Action Chunking Policy	41
5.5	Concluding Remark	44
6	Scaling up Action Chunking Q-learning to Long-horizon Tasks	45
6.1	Decoupled Q-chunking	46
6.2	Experiments	48
6.3	Concluding Remark	52
7	Final Remarks	53
	Bibliography	54
A	Appendix for Q-learning with Adjoint-Matching	66
A.1	Additional Implementation Details for QAM	66
A.2	Full Results	66
A.3	Domain and experiment details	67
A.4	Baselines	67
A.5	Hyperparameters	76
B	Appendix for Reinforcement Learning with Action Chunking	84
B.1	Domain Details	84
B.2	Implementation Details	86
B.3	Experiment Details	90
B.4	Full Results	93
C	Appendix for Decoupled Q-Chunking	102
C.1	Full results	102
C.2	Environments and datasets	103
C.3	Hyperparameters and implementation details	104
D	Proofs for Main Theoretical Results	108
D.1	Utility Lemmata	108
D.2	Proof of Theorem 1	112
D.3	Proof of Theorem 2	114
D.4	Proof of Corollary 1	117
D.5	Proof of Corollary 2	119
D.6	Proof of Proposition 1	122
D.7	Proof of Theorem 3	126
D.8	Proof of Theorem 4	128
D.9	Proof of Proposition 3	146
D.10	Proof of Proposition 2	147
D.11	Proof of Theorem 5	148

D.12 Proof of Theorem 6	153
E Additional Theoretical Results	166
E.1 ε -deterministic dynamics is weakly open-loop consistent	166
E.2 Proof of Proposition 4	167
E.3 Conditions when n -step return policies are provably sub-optimal	169
E.4 Proof of Proposition 5	170
E.5 Closed-loop execution without stochastic shortcuts	172
E.6 Proof of Theorem 7	174

List of Figures

4.1	QAM: Q-learning with Adjoint Matching. QAM uses adjoint matching [35] that leverages the critic’s action gradient directly to fine-tune a flow policy towards the optimal behavior-constrained policy: $\pi_\theta(\cdot s) \propto \pi_\beta(\cdot s)e^{Q(s,\cdot)}$	18
4.2	Robomimic results. QC achieves strong performance across all three robomimic tasks. The first 1M steps are offline and the next 1M steps are online with one environment step per training step (5 seeds).	27
4.3	n-step return ablations on robomimic. Both Q-chunking methods consistently outperform their n -step and 1-step TD counterparts (5 seeds).	27
4.4	End-effector movements early in the training and temporal coherency analysis on cube-triple-task3. <i>Left:</i> QC covers a more diverse set of states compared to BFN in the first 1000 environment steps. <i>Right:</i> QC exhibits a higher temporal coherency in end-effector compared to BFN.	29
4.5	QAM online fine-tunes more effectively than prior methods (50 tasks, 12 seeds). We use the QAM-EDIT (QAM-E) variant for QAM and the CGQL-Linear variant for CGQL (CGQL-L) because they work well for online fine-tuning. For full results, see Figure A.1 in the Appendix.	32
6.1	Decoupled Q-chunking (DQC). <i>Left:</i> The key idea of our method is to ‘decouple’ the action chunk size of the critic Q from that of the policy π . A large critic chunk size allows for efficient value learning while a small policy chunk size makes policy learning more tractable and allows for better policy reactivity. <i>Right:</i> Our method outperforms all baselines on the six hardest environments on OGBench, an offline goal-conditioned RL benchmark with challenging long-horizon tasks. .	46
6.2	Aggregated score across six hardest OGBench environments (10 seeds): cube- $\{\text{triple/quadruple/octuple}\}$, humanoidmaze-giant, and puzzle- $\{4\times 5, 4\times 6\}$. 49	
6.3	Offline goal-conditioned RL results (10 seeds). Our method (<i>DQC</i>) uses <i>decoupled</i> critic and policy chunk sizes. <i>QC</i> : Q-chunking [81]; <i>NS</i> : n -step return backup; <i>OS</i> : 1-step TD-backup; <i>DQC-naïve</i> : same as QC but executes a partial action chunk.	50
6.4	Distilled critic ablations (10 seeds). Each group in the legend contains DQC and its non-distilled counterpart with the same configuration. Our method (DQC) performs on par with or better than the non-distilled counterpart across all configurations.	51

6.5	Hyperparameter sensitivity analysis on cube-quadruple (10 seeds). <i>Best-of-N</i> : the number of action samples drawn from $\pi_\beta(\cdot s)$ during policy evaluation; <i>Implicit loss type</i> : the implicit maximization loss function used for distillation and value backup; <i>Batch size</i> : the number of examples used in each gradient step.	52
A.1	Full training curves for our main results (12 seeds).	78
A.2	Full sensitivity analysis for QAM-E (12 seeds).	79
B.1	End-effector trajectory early in the training. Each subplot above shows the trajectory for 1000 consecutive time steps. We include up to Step 9000.	93
B.2	End-effector trajectory visualization late in the training. Each subplot above shows the trajectory for 1000 consecutive time steps (from Step 900000 to Step 909000).	94
B.3	Complete OGBench offline-to-online RL results by task. 4 seeds. 95% confidence interval.	95
B.6	Full RLPD results by task. QC-RLPD is RLPD-AC (RLPD on the temporally extended action space) where we additionally add a fixed behavior cloning coefficient of 0.01.	96
B.7	Action chunking size ablation on cube-triple-*. Increasing the action chunking size generally helps until $h = 25$. Surprisingly, $h = 25$ is the only chunk size where QC-FQL achieves non-trivial success on the hardest cube-triple-play-task5 (none of other methods can).	96
B.8	Action chunking size ablation on cube-quadruple-play-task4. The n -step return baselines (<i>i.e.</i> , FQL- n) can achieve good initial success during online learning but quickly collapse. In contrast, our method (<i>i.e.</i> , QC-FQL) does not suffer from such collapse and solves the task consistently across reasonably large chunk sizes (<i>e.g.</i> , $h \in \{5, 7, 9\}$). The results are averaged over 5 seeds.	97
B.9	Full robomimic ablation by task. We use 5 seeds for all experiments.	97
B.10	How long each method takes for one step, in milliseconds. Left: offline. Right: online (one agent training step and an environment step). The runtime is measured using the default hyperparameters from our paper on cube-triple-task1 on a single RTX-A5000.	98
B.4	Full OGBench results by task (selected baselines). <i>Top</i> : summary plots by domain; <i>Bottom</i> : individual plots by task.	100
B.5	Full OGBench results by task (selected baselines).	101
C.1	Batch size sensitivity (10 seeds). Large batch size is crucial for DQC’s performance especially on hard tasks (<i>e.g.</i> , cube-quadruple, cube-octuple, puzzle-4x5 and puzzle-4x6).	103
D.1	A $2h$-state MDP that is constructed to meet the upper bound in Theorem 1. The data distribution $\mathcal{D}$ that achieves such an upper bound is collected by the optimal policy: $\pi(X_i) = 1, \pi(\tilde{X}_i) = 0$	115

D.2	A $(3h - 1)$-state MDP that is constructed to meet the upper bound in Corollary 1.	119
D.3	A 6-state MDP that is constructed to illustrate the pathological failure mode of action chunking Q-learning under weak open-loop consistency.	122
D.4	A $(2h + 4)$-state MDP that is constructed to meet the exact upper bound optimality gap in Theorem 3. We redraw the same states Z , G , X_0 in multiple locations in the diagram above for better clarity.	132
D.5	MDP construction Part 1 for Theorem 6 (“the castle”). This diagram describes states X and Y and how actions $a = 0$ and $a = 1$ transition between them. The main purpose of this construction is to make $\hat{V}_{ac}^+(X)$ underestimate V^* by exactly $\vartheta_h^G/(1 - \gamma^h)$. This allows the action chunk that appears in the second part of the construction to be preferred (by π_{ac}^+) over the action chunks that start with $a = 0$ or $a = 1$.	154
D.6	MDP construction Part 2 for Theorem 6 (“the flower”). This diagram describes the remaining states $\tilde{C}$, $\tilde{D}$ and $\tilde{X}$, and what actions $a = 2$ and $a = 3$ do in states X and Y . The main purpose of this construction is to make $\hat{V}_{ac}^+(X)$ overestimate the optimal value of the action chunks that π_{ac}^+ , $Q^*(X, a_t^+)$, by exactly $\vartheta_h^L + \gamma^h \min(\vartheta_h^L, \vartheta_h^G)/(1 - \gamma^h)$.	158
E.1	An MDP where the learned action chunking policies are optimal and the learned n-step return policies can be arbitrarily sub-optimal.	171

List of Tables

4.1	Conceptual differences between QAM and prior methods. QAM is the only method that (1) leverages the action gradient, (2) avoids backpropagation through time, (3) recovers the optimal behavior-constrained policy, and (4) preserves the expressivity of a multi-step flow model.	20
4.2	Summary table for OGBench offline-to-online RL results. For each cell, we report the offline performance after 1M of training steps and then the online performance after 1M of additional online steps. The best method(s) for each column is highlighted in bold and color. Q-chunking methods outperform all prior methods at the end of the online training. See the full results in Table B.5 (complete table) and Figure B.3 (individual plot).	28
4.3	Offline RL performance at 1M training steps (50 tasks, 12 seeds). QAM and two of its variants, QAM-FQL (QAM-F) and QAM-EDIT (QAM-E) outperform all prior baselines. We use CGQL-M, CGQL-L, QAM-F, QAM-E as the abbreviations for CGQL-MSE, CGQL-Linex, QAM-FQL, and QAM-EDIT respectively. For results on individual tasks, see Table A.1 in Section A.2.	31
5.1	Summary of our main theoretical contributions. In this work, we introduce open-loop consistency (OLC: Definition 2) and bounded optimality variability (BOV: Definition 4). Weak OLC provides guarantees on the value estimation error of the action chunking critic but not the optimality of the learned action chunking policy. Strong OLC provides guarantees on the optimality of the learned action chunking policy and its closed-loop execution performance. BOV is an alternative condition to provide guarantees on the closed-loop execution performance. $\Omega(H)$ means that a constant factor of the maximum value gap can be achieved in the worst case.	35
6.1	Comparisons with prior methods (10 seeds). Our method outperforms SHARSA [110] (the previous state-of-the-art method on this benchmark) on most environments.	51
A.1	Full offline results at 1M training steps (12 seeds).	77
A.2	Domain metadata.	80
A.3	Common hyperparameters.	80
A.4	Domain-specific hyperparameters. The best hyperparameter configuration obtained from our tuning runs. We use the same hyperparameter configuration for all tasks in each domain.	81

A.5	Domain-specific hyperparameter tuning range. For QAM-F, $\alpha = \infty$ is equivalent to the original QAM. Similarly, for QAM-E, $\sigma_a = 0$ is equivalent to the original QAM. For methods with more than one hyperparameter, we tune all possible combinations within the specified sweep ranges. For example, for CGQL, we sweep over all $3 \times 3 = 9$ configurations with 3 possible values of ϑ and 3 possible values of τ	82
A.6	Training speed and parameter count for each method on cube-triple.	83
B.1	Domain metadata. Dataset size (number of transitions), episode length, and the action dimension. For OGBench tasks, the action dimension is 5 (x position, y position, z position, gripper yaw and gripper opening). For robomimic tasks, the action dimension is 7 for square (3 for translation, 3 for rotation, and one final dimension for the gripper opening).	85
B.2	Common hyperparameters.	91
B.3	Behavior regularization coefficient (α).	91
B.4	Number of actions sampled for the expected-max Q operator (N) for BFN methods.	92
B.5	Complete OGBench offline-to-online RL results. Averaged over 4 seeds with 95% confidence interval.	99
B.6	Parameter count for each method. RLPD has a much larger parameter count because it uses $K = 10$ critic networks whereas all the other methods and baselines use only $K = 2$. . .	101
C.1	Complete results for all hyperparameter configurations across different combinations of h, n and h_a (10 seeds). We adopt the following abbreviations: c3=cube-triple, c4=cube-quadruple, c8=cube-octuple, hg=humanoidmaze-giant, p45=puzzle-4x5, p46=puzzle-4x6. The hyperparameters used are specified in Tables C.5 and C.6.	102
C.2	Environment metadata. For both humanoidmaze-giant and puzzle-4x5, we use the default dataset that is released in the original OGBench benchmark [111]. For the other environments, we use larger datasets as we find them to be essential for achieving good performance on these environments.	104
C.3	Common hyperparameters. For the GCRL goal-sampling distribution we follow the same hyperparameters used in [110].	105
C.4	Oracle goal representation description for each environment. Following [110], we assume access to an oracle goal representation for each environment. More detailed definitions of these oracle goal representations are available in OGBench [111].	106
C.5	Environment-specific hyperparameters for DQC, QC, NS, OS, SHARSA, HIQL, IQL, and HFBC. For SHARSA, HIQL, IQL, and HFBC, we follow the hyperparameters in the original paper [110].	106
C.6	Environment-specific hyperparameters under different h, n, h_a configurations for DQC, QC, NS, OS. For DQC-naïve, we use the same hyperparameter as the corresponding QC baseline.	107

C.7	Hyperparameter tuning range for all methods.	For NS, we only tune κ_b and n because the policy chunk size is always 1 and there is no distilled critic. Similarly, for QC, we only tune κ_b and $h = h_a$ because the policy chunk size is the same as the critic chunk size and there is no distilled critic. For OS, we only tune κ_b .	107
-----	---	---	-----

Acknowledgments

First, I would like to thank my advisor, Sergey Levine, for his guidance throughout my PhD journey. He is the best advisor I could have asked for. His dedication to mentoring, incredible work ethic, and deep insights across a wide range of research topics have inspired me to become a better researcher. Beyond his unusual ability to come up with unconventional research ideas, one thing that I appreciate most is that he always pushed me to ask the right questions. Many times during my PhD I would spend days thinking about research ideas that would have never had practical impacts in a million years. Sergey would always pull me back to reality and encourage me to think about what research questions truly matter.

Next, I would also like to thank my other committee members Abhishek Gupta, Angjoo Kanazawa, and Aditi Krishnapriyan. Abhishek was one of my first academic mentors and collaborators in graduate school. I still remember struggling to make progress in my first year, and he was willing to spend hours debugging a SAC implementation with me despite being a senior PhD student at the time. Abhishek inspired me to be as hands-on and helpful as possible with students I mentored later in my PhD. Angjoo and Aditi gave very valuable feedback to my research talks and I am deeply thankful for their guidance and insights.

I have also had the fortune to work with many wonderful collaborators. A special thank you to Toru Lin, Ajay Jain, Michael Janner, Brent Yi, Dibya Ghosh, Yuexiang Zhai, Kevin Frans, Seohong Park, Paul Zhou, Amy Zhang, Chelsea Finn, Ilya Kostrikov, and Aviral Kumar. I would especially like to thank Amy and Aviral for mentoring me and teaching me how to do research when I was a junior PhD student. I also had the fortune to work with many talented undergraduate students: Jason Zhang, Max Wilcoxson, Derek Guo, Jennifer Zhao, Andria Xu, and Ishir Garg. I am also grateful to my BAIR buddy Yeshwanth Cherapanamjeri who is one of the kindest people I have ever met. He helped me through many difficult times during my PhD, and I am really thankful for all the support he has provided over the years.

To all current and past members of RAIL and BAIR, I am grateful for everyone who has helped make our research environment so open and collaborative. A special shoutout to my cohort: Dibya Ghosh, Katie Kang, Ameesh Shah, Laura Smith, Fangchen Liu, with whom I have shared memorable times while navigating through the PhD journey together. I am thankful to all RAIL alumni, including Michael Janner, Ashvin Nair, Abhishek Gupta, Kelvin Xu, Ilya Kostrikov, Oleg Rybkin, who provided guidance along the way, as well as all current and past RAIL members I have overlapped with: Seohong Park, Paul Zhou, William Chen, Catherine Glossop, Kyle Stachowicz, Homer Walke, Kevin Black, Marwa Abdulhai, Jagdeep Bhatia, Kai Nelson, Rohin Manvi, Andrew Wagenmaker, Vivek Myers, Mitsuhiro Nakamoto, Charles Xu, Kwanyoung Park, James Bowen, Pranav Atreya, Yifei Zhou, Joey Hong, Manan Tomar. Some of my most wonderful moments during my PhD are brainstorming and exchanging research ideas with these people.

To my roommate and friend Dibya, I am thankful for all the time we have shared together. During the COVID years when I just moved to Berkeley and did not know many people, he always went out of his way to make plans and make me feel welcome. As a Berkeley local, he showed me many wonderful places and cool running routes in this little town (and I have passed them down to the next generation :)).

I am also grateful to my friend Katie. Aside from being an awesome friend to hang out with, I especially appreciate the way she always encouraged me to explore new things that I would not have done otherwise. There were so many times when I was unsure about starting something new, she would always somehow find ways to convince me into doing it (which usually paid off in the end!) Over time, I have become more willing to make bold decisions, a quality that has also shaped how I approach research.

Toru is a rare friend whom I would have never expected to meet during my PhD. She is always so optimistic and supportive, bringing out a better version of me and reminding me how nice it is to be grateful and to appreciate even the smallest things in life. Running races together with Toru became a staple during some of the most stressful times of my final PhD year, and I feel so lucky and thankful for all the moments that we have shared together.

I would also like to thank all the staff at elaichi co., a little chai shop where I have spent hours grinding through papers and deadlines. I am also grateful for Berkeley, and for all the sunsets that I got to appreciate throughout the years from the 8th floor of Berkeley Way West. Outside Berkeley, I would like to thank my old Toronto friends, David Ma, Daniel Guo, Ryan Tian, Light Zhu and Sherry Xie for all the fun times we spent together and for always being so supportive.

Lastly, I would like to express my deepest gratitude to my parents for their encouragement and support throughout my journey. I would have never been where I am without them.

Chapter 1

Introduction

Reinforcement learning (RL) is a powerful framework for autonomous online self-improvement of learning agents. Through iterative online interactions and optimization, an RL algorithm holds the promise that given any sequential decision-making task with a well-specified reward function, it can eventually find the optimal policy for the task. However, making RL algorithms practical in the real world (*e.g.*, robotics) still remains challenging because even the most sample-efficient online RL algorithms can take hours to learn new tasks. This is in stark contrast to us humans, where we can often adapt to any new task presented to us incredibly fast. We can quickly adapt to new scenarios not only because of years of evolution, but also because we have seen and learned many similar tasks in our lives before and are able to quickly leverage the extremely rich and complex prior experience to allow us to adapt quickly.

Thus, leveraging large-scale prior experience data to produce intelligent behaviors that guide the agent to explore and learn efficiently online is monumental in enabling scalable RL especially for continuous action spaces, where policy optimization is especially challenging yet critical for robotics applications. However, extracting behaviors from human prior data is not an easy task, since such data is often noisy, multi-modal, and non-Markovian. These properties are difficult to capture with Markovian Gaussian policies used by the majority of prior online RL algorithms. In the adjacent field of imitation learning (IL) for robotic manipulation, *action chunking*, a simple practical technique that predicts and executes a group of actions rather than one at a time, has been found to be able to handle non-Markovian/noisy prior data better, but this makes the distribution of action chunks even more complex and multi-modal.

Generative modeling is therefore a natural choice for handling distributions over action chunks, and its effectiveness is evident from the large number of generative policy methods proposed in recent years. Generative policies have since become a standard choice in IL. However, the story is different in RL. Oftentimes RL requires optimizing the policy to maxi-

mize an objective (*e.g.*, estimated value function). While traditionally in generative modeling literature, steering a pre-trained distribution towards samples that result in a high score under a given utility function is a standard problem, RL introduces some unique challenges. What makes RL particularly challenging is that the objective function that the generative policy needs to optimize against is constantly changing. This requires us to carefully design our algorithms such that the policy optimization is done efficiently while remaining stable when subject to a moving objective. Chapter 4 introduces new algorithms that allow us to perform efficient offline pre-training and online fine-tuning in both action chunking and non-action chunking settings.

Despite the empirical success of action chunking policies in RL, there is still a lack of theoretical understanding of action chunking. Chapter 5 lays the theoretical foundation of action chunking Q-learning where we identify the key *open-loop consistency* condition for the prior data distribution (Definition 2) under which Q-learning with action chunking critic is guaranteed to produce a near-optimal action chunking policy. On the flip side, we also show that, in the worst case, as long as the data distribution is not open-loop consistent, an action chunking policy may be arbitrarily sub-optimal. To tackle this challenge, we characterize the condition under which we can mitigate this bias by closed-loop execution (*i.e.*, only executing the first action in the predicted action chunk) of an RL-trained action chunking policy, and show that even when the data is not open-loop consistent, the closed-loop execution policy is expected to be close to the optimal closed-loop policy. Motivated by our theoretical insights into action chunking RL, Chapter 6 introduces a practical algorithm that exhibits strong scalability to long-horizon tasks.

Overall, we take a step towards scalable reinforcement learning with generative policies powered by action chunking. From algorithms that make action chunking RL practical for online sample-efficient learning, theoretical understanding of action chunking Q-learning, to scaling up practical action chunking RL to long-horizon tasks, this dissertation seeks to provide both the algorithmic and theoretical foundation for scalable reinforcement learning in the real world.

Chapter 2

Preliminaries and Problem Definition

2.1 Reinforcement Learning

We consider a Markov Decision Process (MDP), $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \gamma, R, \mu)$, where $\mathcal{S}$ is the state space, $\mathcal{A} = \mathbb{R}^A$ ($A \in \mathbb{Z}^+$) is the action space, $P : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathcal{S}}$ is the transition function, $\gamma \in [0, 1)$ is the discount factor, $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, and $\mu \in \Delta_{\mathcal{S}}$ is the initial state distribution. We have access to a dataset D consisting of a set of transitions $D = \{(s_i, a_i, s'_i, r_i)\}_{i=1}^{|D|}$, where $s' \sim P(\cdot \mid s, a)$ and $r = R(s, a)$. In this dissertation, we consider two problem settings with MDPs.

The first problem setting is *offline RL*. In this problem setting, we learn a policy $\pi_{\theta} : \mathcal{S} \rightarrow \Delta_{\mathcal{A}}$ from D to maximize its expected discounted return,

$$U_{\pi} = \mathbb{E}_{s_0 \sim \mu, s_{k+1} \sim P(\cdot \mid s_k, a_k), a_k \sim \pi(\cdot \mid s_k)} \left[\sum_{k=0}^{\infty} \gamma^k R(s_k, a_k) \right]. \quad (2.1)$$

The policy is learned purely from D *without* any environment interaction.

The second problem setting is *offline-to-online RL*. This setting can be thought of as an extension of the offline RL setting where the goal is to fine-tune an offline pre-trained policy π_{θ} by continuously interacting with the MDP through trajectory episodes with a task-dependent maximum episode length of H (*i.e.*, the maximum number of time steps before the agent is reset to μ). The central challenge of offline-to-online RL is to maximally leverage the behavior distribution in D to learn as sample-efficiently as possible online.

2.2 Temporal Difference Learning

Modern value-based reinforcement learning methods often learn a critic network, $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ parameterized by ϕ to approximate the expected return starting from state s and action

a , and ϕ is often trained using the temporal-difference (TD) loss:

$$L(\phi) = \mathbb{E}_{s,a,s' \sim \mathcal{D}} \left[(Q_\phi(s, a) - r(s, a) - \gamma \max_{a'} Q_{\bar{\phi}}(s', a'))^2 \right], \quad (2.2)$$

where $\bar{\phi}$ is often set to be an exponential moving average of ϕ .

Multi-step return backup. TD learning can sometimes struggle with long-horizon tasks due to the bootstrapping bias problem, where regressing the value network towards its own potentially inaccurate value estimates amplifies the value estimation errors further. To tackle this challenge, we can instead sample a trajectory segment, $(s_t, a_t, s_{t+1}, \dots, a_{t+n-1}, s_{t+n})$, to construct an n -step return backup target from states n steps ahead:

$$L_{\text{ns}}(\phi) = \mathbb{E}_{s_t, a_t, \dots, s_{t+n}} \left[(Q_\phi(s_t, a_t) - R_{t:t+n} - \gamma^n Q_{\bar{\phi}}(s_{t+n}, a_{t+n}^*))^2 \right], \quad (2.3)$$

where $a_{t+n}^* = \arg \max_{a_{t+n}} Q(s_{t+n}, a_{t+n})$, $R_{t:t+n} := \sum_{t'=t}^{t+n-1} \gamma^{t'-t} r(s_{t'}, a_{t'})$. The n -step return backup reduces the effective horizon by a factor of n , alleviating the bootstrapping bias problem. However, such a value estimate is always biased towards the off-policy data distribution, and is also commonly referred to as the *uncorrected n -step return estimator* [44, 76]. While there are ways to correct this value estimator via importance sampling [117, 98, 123], they require additional tricks (*e.g.*, importance ratio truncation) for numerical stability and re-introduce biases into the estimator, resulting in a delicate trade-off between variance and bias that needs to be carefully balanced.

Implicit value backup. Instead of using $\max_{a'} Q(s', a')$ as the TD target, we can use an *implicit maximization* loss function f_{imp} to learn $V_\xi(s)$ to approximate it [75]:

$$L(\xi) = \mathbb{E}_{s,a \sim \mathcal{D}} [f_{\text{imp}}^\kappa(\bar{Q}(s, a) - V_\xi(s))]. \quad (2.4)$$

Two popular choices of f_{imp}^κ are (1) expectile: $f_{\text{expectile}}^\kappa(c) = |\kappa - \mathbb{I}_{c < 0}|c^2$, and (2) quantile: $f_{\text{quantile}}^\kappa(c) = |\kappa - \mathbb{I}_{c < 0}||c|$, for any real value $\kappa \in [0.5, 1)$. At the optimum of $L(\xi)$, $V_\xi(s)$ approximates the κ -expectile/quantile of the distribution of the TD target for $Q(s, a)$, induced by the data distribution $\mathcal{D}$. With such a technique, we no longer need to explicitly find the action a that maximizes $Q(s, a)$ and can use $V_\xi(s)$ as the backup target:

$$L(\phi) = \mathbb{E}_{s,a,s' \sim \mathcal{D}} [(Q_\phi(s, a) - r(s, a) - \gamma V_\xi(s'))^2]. \quad (2.5)$$

2.3 Flow-Matching Generative Model

A flow model uses a time-varying velocity field $v : \mathbb{R}^d \times [0, 1] \rightarrow \mathbb{R}^d$ to estimate the marginal distribution of a denoising process from noise, $X_0 = \mathcal{N}(0, I_d)$, to data, $X_1 = D$, at each intermediate time $t \in [0, 1]$:

$$X_t = (1 - t)X_0 + tX_1. \quad (2.6)$$

In particular, the flow model approximates the intermediate X_t via an ordinary differential equation (ODE) starting from the noise: $X_0 = \mathcal{N}$:

$$d\hat{X}_t = v(\hat{X}_t, t)dt. \quad (2.7)$$

Flow models are typically trained with a *flow matching* objective [87]:

$$L_{\text{FM}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}[0,1], x_0 \sim \mathcal{N}, x_1 \sim D} [\|v_\theta((1-t)x_0 + tx_1, t) - x_1 + x_0\|_2^2], \quad (2.8)$$

where the optimal velocity field, v_{θ^*} , results in $\hat{X}_t$ whose marginal distribution $p_v(x_t)$ exactly recovers the marginal distribution of the original denoising process X_t , $p_D(x_t)$, for each $t \in [0, 1]$ [85]. Furthermore, one may use the Fokker-Planck equations to construct a family of stochastic differential equations (SDE) that admits the same marginals as well:

$$d\hat{X}_t = \left(v(\hat{X}_t, t) + \frac{\sigma_t^2 t}{2(1-t)} \left(v(\hat{X}_t, t) + X_t/t \right) \right) dt + \sigma_t dW_t, \quad (2.9)$$

with W_t being a Brownian motion and $\sigma_t > 0$ being any noise schedule.

Adjoint matching is a technique developed by [35] with the goal of modifying a base flow-matching generative model v_β such that the resulting flow model v_θ generates the following *tilt* distribution:

$$p_\theta(X_1) \propto p_\beta(X_1) e^{Q(X_1)}, \quad (2.10)$$

where p_θ is the tilt distribution induced by v_θ and p_β is the base distribution induced by v_β , and $Q : \mathbb{R}^d \rightarrow \mathbb{R}$ is any value function that up-weights or down-weights the probability of each example in the domain $\mathbb{R}^d$. [35] uses a marginal-preserving SDE with a ‘memoryless’ noise schedule (*i.e.*, X_0 and X_1 are independent), $\sigma_t = \sqrt{2(1-t)/t}$.¹

$$dX_t = (2v(X_t, t) - X_t/t) dt + \sqrt{2(1-t)/t} dW_t, \quad (2.11)$$

where the minimizer of the following stochastic optimal control equation (with X_t sampling from the joint distribution defined by the SDE in Equation (2.11)),

$$L(\theta) = \mathbb{E}_{\mathbf{X}=\{X_t\}_t} \left[\int_0^1 \left(\frac{2}{\sigma_t^2} \|v_\theta(X_t, t) - v_\beta(X_t, t)\|_2^2 \right) dt - Q(X_1) \right] \quad (2.12)$$

gives the correct marginal *tilt* distribution for X_1 [36]:

$$p(X_1) \propto p_\beta(X_1) e^{Q(X_1)}. \quad (2.13)$$

¹The adjoint-matching paper [35] uses a more general framework with $X_t = \beta_t X_0 + \alpha_t X_1$. For simplicity, we assume the special case where $\alpha_t = t, \beta_t = 1 - t$, and $\sigma_t = \sqrt{2(1-t)/t}$.

Let the adjoint state be the gradient of the tilt function applied at the denoised X_1 :

$$g(\mathbf{X}, t) := \nabla_{X_t} \left[\int_t^1 \left(\frac{2}{\sigma_{t'}^2} \|v_\theta(X_{t'}, t') - v_\beta(X_{t'}, t')\|_2^2 \right) dt' - Q(X_1) \right], \quad (2.14)$$

where $\mathbf{X} = \{X_t\}_t$ is the random variable that represents the SDE trajectory, and g satisfies

$$\frac{dg(\mathbf{X}, t)}{dt} = -\nabla_{X_t} [2v_\theta(X_t, t) - X_t/t] g(\mathbf{X}, t) - \frac{2}{\sigma_t^2} \nabla_{X_t} \|v_\theta(X_t, t) - v_\beta(X_t, t)\|_2^2, \quad (2.15)$$

with the boundary condition $g(\mathbf{X}, t = 1) = -\nabla_{X_1} Q(X_1)$. We can compute the adjoint states by stepping through the reverse ODE (which can be efficiently computed with the vector-Jacobian product (VJP) in most modern deep learning frameworks). Then, it can be shown that the stochastic optimal control loss in Equation (2.12) is equivalent to the ‘basic’ adjoint matching objective below:

$$L_{\text{BAM}}(\theta) = \mathbb{E}_{\mathbf{X}} \left[\int_0^1 \|2(v_\theta(X_t, t) - v_\beta(X_t, t))/\sigma_t + \sigma_t g(\mathbf{X}, t)\|_2^2 dt \right]. \quad (2.16)$$

The optimal v_θ coincides with the optimal solution in the original SOC equation (Equation (2.12)), which gives the correct marginal distribution of X_1 as a result. However, the objective is equivalent to the objective used in the continuous adjoint method [116] with its gradient equal to that of backpropagation through the denoising process. Instead, [35] derive the ‘lean’ adjoint state where all the terms in the adjoint state that are zero at the optimum are removed from the original adjoint state. The ‘lean’ adjoint state satisfies the following ODE:

$$d\tilde{g}(\mathbf{X}, t) = -\nabla_{X_t} [2v_\beta(X_t, t) - X_t/t] \tilde{g}(\mathbf{X}, t) dt, \quad (2.17)$$

with the same boundary condition $\tilde{g}(\mathbf{X}, 1) = -\nabla_{X_1} Q(X_1)$.

Note that computing the ‘lean’ adjoint state only requires the base flow model $v_\beta(X_t, t)$ and no longer needs $v_\theta(X_t, t)$ as is needed in either the basic adjoint matching objective (Equation (2.16)) or naive backpropagation through the denoising process. The resulting adjoint matching objective is

$$L_{\text{AM}}(\theta) = \mathbb{E}_{\mathbf{X}} \left[\int_0^1 \|2(v_\theta(X_t, t) - v_\beta(X_t, t))/\sigma_t + \sigma_t \tilde{g}(\mathbf{X}, t)\|_2^2 dt \right], \quad (2.18)$$

where again $\mathbf{X}$ is sampled from the marginal preserving SDE in Equation (2.11). Because the terms omitted are zero at the optimum, they do not change the optimal solution for v_θ . Thus, the optimal solution for the adjoint matching gives the correct tilt distribution.

Chapter 3

Related Work

This chapter surveys related areas of literature adjacent to generative policies and action chunking pertaining to this dissertation.

3.1 RL with Generative Policies

Diffusion and flow policies have been explored in both policy gradient methods [120] and actor-critic methods [42, 69, 20, 18, 89, 31, 154, 59, 34, 1, 162, 57]. The key challenge of leveraging diffusion/flow policies in TD-based RL methods is to optimize these policies against the critic function (*i.e.*, $Q(s, a)$). Prior work can be largely put into three categories based on how the value function is used:

(1) *Post-processing* approaches refine the action distribution from a base diffusion/flow policy with rejection sampling based on the critic value [57, 92, 81, 37], or using additional gradient steps to hill climb the critic [92] (*i.e.*, $a \leftarrow a + \nabla_a Q(s, a)$). These approaches often reliably improve the quality of the extracted policy but at the expense of additional computation during evaluation or even training (*i.e.*, rejection sampling for value backup target [81, 37]). Alternatively, one may train a residual policy that modifies a base behavior policy in either the noise space [132, 152] or in the action space directly [161, 4, 3, 37].

(2) *Backprop-based* approaches perform direct backpropagation through both the critic and the policy [154, 59, 33, 162, 108, 41, 19]. While this is the most straightforward implementation, it requires backpropagation through the diffusion/flow policy’s denoising process, which has been observed to be unstable [108], or instead learns a distilled policy [33, 21, 108, 41, 19], at the expense of policy expressivity.

(3) *Intermediate fine-tuning* approaches mitigate the stability/expressivity trade-off in backprop-based approaches by leveraging the critic to construct an objective that provides direct step-wise supervision to the intermediate denoising process [118, 42, 32, 82, 46, 163, 90, 73].

While these approaches remove the need for backpropagation through the denoising process completely, the challenge lies in carefully crafting the step-wise objective that does not introduce additional biases and learning instability.

3.2 Diffusion and Flow-Matching with Guidance

More broadly, diffusion and flow-matching models have been used for generating data with different modalities ranging from images [122], videos [63], and text [88]. In most applications, the generative models trained on large-scale unlabeled data do not provide high-quality samples when conditioned on some context (*e.g.*, language description), and a common practice is to augment the sampling process with classifier guidance/classifier-free guidance [29, 62], with the goal of aligning the sampling distribution better with the posterior distribution conditioned on the context. However, most of the guidance methods suffer from a bias problem that is tricky to tackle, stemming from the fact that simply adding or interpolating two diffusion/flow-matching sampling processes does not lead to the correct composite distribution in general [38, 16]. One solution is to use Langevin dynamics sampling approaches [136] where only the score function for the noise-free distribution is required, but they have been known to underperform diffusion/flow models due to the challenge of accurately estimating the score functions in low-density regions [137]. Since then, a line of work has proposed solutions to generate the correct composite distribution. [38], [115], [146], [133] and [134] propose to use Sequential Monte Carlo (SMC) that uses resampling procedures to leverage additional test-time compute to correct such bias. Rather than correcting the distribution at test-time, [35] and [58] take a different perspective by formulating it as a stochastic optimal control (SOC) objective that can be efficiently optimized as a fine-tuning process while providing a guarantee that the model converges to the correct distribution at the optimum.

3.3 Offline and Offline-to-Online Reinforcement Learning

Offline and offline-to-online RL methods assume access to an offline dataset to learn a policy without interactions with the environment (offline) [78, 75, 144] or with as little online interaction with the environment as possible (offline-to-online) [79, 9, 102]. Q-learning or TD-based RL algorithms have been a popular choice for these problem settings as they naturally handle off-policy data without the need for on-policy rollouts, and also exhibit great online sample-efficiency [22, 26]. A large body of literature in these two problem settings has focused on tackling the distribution shift challenge by appropriately constraining the policies with respect to the prior offline data, and most of them use the standard 1-step TD backup for Q-learning, which has been known to suffer from the bootstrapping bias problem in the RL literature [66, 141].

3.4 Action Chunking

Action chunking is a technique popularized by roboticists for imitation learning (IL), where the policy predicts and executes a sequence of actions in an open-loop manner (“an action chunk”) [164]. Action chunking has been shown to improve policy robustness [164, 53, 13], and handle non-Markovian behavior in offline data [164]. Existing RL methods that incorporate action chunking typically focus on fine-tuning a policy pre-trained with imitation learning [120]. [147] propose to learn a critic on action chunks by integrating n -step returns with a transformer. However, their method only applies chunking to the critic, while still optimizing a single-step actor. [80] also observe that learning a critic over short action chunks removes the off-policy bias in n -step return backups, leading to more stable and effective value learning. There are key differences between their work and ours—[80] operates in the online episodic RL setting [155, 71] and use Gaussian policies to predict parameters of movement primitives (MP) [124, 106], which are then used to generate full action sequences at the beginning of each episode. In contrast, we operate in the conventional offline-to-online RL setting and leverage more expressive flow-matching-based policies to predict short action sequences directly in the raw action space. [125] also train critics on action chunks and impose a behavior cloning loss that aligns with the principles behind Q-chunking. The key distinction from our work lies in their use of a multi-level, factorized critic architecture [126], which generates and refines action chunks from coarse to fine granularity via iterative discretization. At each level, the action space is discretized into bins, and the Q-function is modeled independently for each action dimension and timestep, conditioned on the whole action chunks predicted at the previous coarser level. While this factorized critic design enables tractable value-maximizing action sampling, it imposes strong structural assumptions on the action space, limiting policy expressiveness at each refinement level. In contrast, we make no such assumptions in our recipe, which allows us to derive two general algorithms where both the critic and policy operate directly on action chunks without requiring factorization or iterative discretization/refinement.

On the theoretical side, existing analyses for action chunking focus exclusively on the imitation learning setting [149, 130]. While they laid out the theoretical foundation of action chunking policies for imitation learning, formal guarantees of action chunking RL are still an open problem. In the adjacent field of stochastic optimal control (SOC), action chunking is related to control under *intermittent observations* where the observation inputs to the controller are either unreliable (*e.g.*, with a Poissonian model [153, 39]), or partially missing [97, 159, 105, 10]. While conceptually related, these analyses are in the continuous-time setting in contrast to discrete-time transitions. To the best of our knowledge, we are the first to provide a formal analysis of action chunking in Q-learning. In particular, we identify the key open-loop consistency condition under which we quantify the exact worst-case Q-learning sub-optimality.

3.5 Hierarchical Reinforcement Learning

Hierarchical reinforcement learning methods [28, 30, 113, 121, 127, 114, 52, 158] solve tasks by typically leveraging a bi-level structure: a set of low-level/skill policies that directly interact with the environment and a high-level policy that selects among low-level policies. The low-level policies can also be learned via online RL [77, 150, 151, 100] or offline pre-training on a prior dataset [106, 95, 2, 114, 148, 104, 64, 47, 17, 107]. In the options framework, these low-level policies are often additionally associated with initiation and termination conditions that specify when and for how long these actions can be used [142, 94, 23, 131, 74, 27, 139, 45, 6, 7, 8, 72]. A long-lasting challenge in HRL is optimization stability because the high-level policy needs to optimize for an objective that is shaped by the constantly changing low-level policies [100]. Prior work [2, 114, 156] avoided this by first pre-training low-level policies and then keeping them frozen during the optimization of the high-level policy. Macro-actions [93, 40], or action chunking [164] are another form of temporally extended action, a special case of the low-level policies often considered in HRL, options literature, where a short horizon of actions is predicted all at once and executed in open loop. Such an approach collapses the bi-level structure, conveniently side-stepping optimization instability, and when combined with Q-learning, has shown great empirical successes in the offline-to-online RL setting [126, 81]. Action chunking policies need to predict multiple actions open-loop, which can be difficult to learn and can sacrifice reactivity. As we will show later in Chapter 6, we can address this downside and regain policy reactivity by predicting and executing only a partial action chunk, while still learning with the fully chunked critic for TD-backup. This design preserves the value propagation benefits of a chunked critic without relying on fully open-loop action chunking policies, allowing our approach to work well on a wider range of tasks.

3.6 Theoretical Analysis for Reinforcement Learning under Unobserved Confounding Variables

RL with action chunking policies can be seen as a special case of RL under unobserved confounding variables [68] as the action chunking policies ignore the intermediate states during the execution of an action chunk. Prior analyses are based on either causal-inference-inspired sensitivity models [67, 103, 70], confounded MDP models [12, 50, 129], or more general partially observable MDP (POMDP) models [145, 96, 128, 11] where the confounding variables are modeled as part of the partially observable states. These models largely focus on characterizing either how much confounding variables affect the policy behavior (*e.g.*, bounded odds-ratio between the policy with or without conditioning on the confounding variables [67]) or how much the observations reveal the confounding variables (*e.g.*, the full-rank emission matrix assumption [5] and the weak revealing assumption [86] in POMDP). In contrast, our analysis in Chapter 5 specializes in action chunking policies where the unobserved variables are the intermediate states during an action chunk. This allows us to establish a more specialized (and thus distinct) open-loop consistency condition under which

we can identify the exact worst-case bias (*i.e.*, with matching lower and upper bounds to the exact value) for both behavioral value estimation and sub-optimality gap of the fixed point of Bellman optimality iteration, which are usually unknown under the more general models/assumptions in the literature.

Chapter 4

Q-learning with Flow-Matching Policies

A long-standing tension in continuous-action RL, especially in the offline/offline-to-online setting, is between policy expressivity and optimization tractability with respect to the critic (*i.e.*, $Q(s, a)$). Simple policies, such as single-step Gaussian policies, are easy to train, since they can directly leverage the critic’s action gradient (*i.e.*, $\nabla_a Q(s, a)$) via the reparameterization trick [56]. This optimization tractability and stability, however, often comes at the cost of expressivity. Some of the most expressive policy classes today, such as flow or diffusion policies, generate actions through a multi-step denoising process. While this allows them to represent complex, multi-modal action distributions, leveraging the action gradient requires backpropagation through the entire denoising process, which often leads to instability [108].

One might be tempted to directly apply the critic’s action gradient to intermediate noisy actions within the denoising process, as in diffusion classifier guidance (with the critic function being the classifier) [29]. Intuitively, this blends two generative processes together: one that generates a behavior action distribution, and another that hill-climbs the critic to maximize action value. While this approach bypasses the backpropagation instability and retains full policy expressivity, it relies on the assumption that the critic’s gradient at a noisy action is a good proxy for its gradient at the corresponding denoised action. In practice, this assumption often breaks down: when the offline dataset has limited action coverage, the critic is well-trained only on a narrow distribution of noiseless actions, rendering its gradients unreliable for intermediate noisy out-of-distribution actions.

In this chapter, we introduce new TD-based RL algorithms to tackle these policy optimization challenges. First, Section 4.1 introduces Flow Q-Learning (FQL) that leverages distillation to turn a multi-step flow policy into a single-step flow map that directly maps noise to action with a single forward pass of a neural network. This allows stable backpropagation

Algorithm 1 Flow Q-Learning (FQL)

```

function ODE( $s, a_0; v$ )
  for  $t = 0, h, \dots, (T - 1)h$  do
     $a_{t+h} \leftarrow a_t + h \cdot v(s, a_t, t)$   $\triangleright$  Euler's method
  return  $a_1$ 

for each training iteration do
  Sample batch  $\{(s, a, r, s')\} \sim D$ 
  Critic update:  $L(\phi) = \mathbb{E}_{z \sim \mathcal{N}(0, I_d)} [(Q_\phi(s, a) - r - \gamma Q_\phi(s', \mu_\theta(s', z)))^2]$ 
  Policy update:  $L(\theta) = \mathbb{E}_{z \sim \mathcal{N}(0, I_d)} [-Q_\phi(s, \mu_\theta(s, z)) + \alpha \|\text{ODE}(s, z; v_\beta) - \mu_\theta(s, z)\|_2^2]$ 
return One-step policy  $\mu_\theta$ 

```

through the distilled policy while preserving the ability of the policy to represent multi-modal action distributions. Next, Section 4.2 introduces Q-learning with Adjoint Matching (QAM) where it leverages adjoint matching [35] to steer the multi-step flow policy directly towards the optimal behavior-constrained policy. Unlike FQL that may still lose policy expressiveness from distillation, QAM retains the multi-step flow policy while effectively leveraging the critic’s action gradient to fine-tune it. As an application of these flow RL approaches in the real-world, Section 4.3 discusses how they can be readily adapted to the action chunking setting. Finally, in Section 4.4, we conduct a comprehensive empirical study comparing policy extraction methods for flow/diffusion policies, including recent approaches and novel baselines, and show QAM consistently achieves strong performance across both offline RL and offline-to-online RL benchmarks in both action chunking and non-action chunking settings.

4.1 Flow Q-Learning

This section introduces **Flow Q-Learning**, a new RL algorithm that directly leverages the critic’s action gradient while maintaining the policy’s ability to express multi-modal action distributions. The idea is simple—instead of optimizing the multi-step flow policy, we directly train a one-step policy to approximate the flow map induced by this multi-step behavior flow policy while steering this one-step policy to maximize the critic value using its action gradient.

The algorithm

In this section, we describe in detail how our method leverages distillation to directly optimize a flow policy without suffering from backpropagation instability.

To start with, we train a behavior flow policy, $v_\beta : \mathcal{S} \times \mathbb{R}^A \times [0, 1] \rightarrow \mathbb{R}^A$, using the standard

flow-matching loss:

$$L_{\text{FM}}(\beta) = \mathbb{E}_{(s,a) \sim D, t \sim \text{Unif}[0,1], z \sim \mathcal{N}(0, I_d)} [\|v_\beta(s, ta + (1-t)z, t) - (a - z)\|_2^2]. \quad (4.1)$$

To avoid backpropagation through the multi-step flow, we train a one-step distilled policy $\mu_\theta : \mathcal{S} \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ that maps directly from noise to action. The loss function consists of two parts. The first part maximizes the value function under the one-step policy. The second part controls how much the one-step policy’s flow map (*i.e.*, the map from the noise to the action) can deviate from that of the multi-step behavior flow policy. The loss is formally defined as follows:

$$L_{\text{FQL}}(\theta) = \mathbb{E}_{s \sim D, z \sim \mathcal{N}(0, I_d)} \left[\underbrace{-Q_\phi(s, \mu_\theta(s, z))}_{\text{Value Maximization Loss}} + \underbrace{\alpha \|\mu_\beta(s, z) - \mu_\theta(s, z)\|_2^2}_{\text{Distillation Loss}} \right], \quad (4.2)$$

where $\mu_\beta(s, z)$ is the solution of the ODE represented by v_β with z as the initial noise:

$$\mu_\beta(s, z) = a_1, \quad a_u = \int_0^u v_\beta(s, a_t, t) dt, \quad a_0 = z. \quad (4.3)$$

In practice, this is approximated by numerical integration via Euler’s method:

$$a_{t+h} \leftarrow a_t + h \cdot v_\beta(s, a_t, t). \quad (4.4)$$

In this procedure, T is the number of numerical integration steps used in Euler’s method and $h = 1/T$ is the step size induced by T .

Intuitively, the one-step policy μ_θ retains the expressivity of the multi-step behavior flow v_β as much as possible through the distillation loss, while at the same time maximizing the value function without suffering from unstable backpropagation.

Finally, we use the standard critic loss in Q-learning:

$$L(\phi) = \mathbb{E}_{(s, z, s', r) \sim D, z \sim \mathcal{N}(0, I_d)} [(Q_\phi(s, a) - r - \gamma Q_{\bar{\phi}}(s', \mu_\theta(s', z)))^2], \quad (4.5)$$

where the one-step policy is used to directly generate action a' for the next state s' , and then fed into the target critic $Q_{\bar{\phi}}$ to provide the value target. An overview of the FQL algorithm is available in Algorithm 1.

Another benefit of FQL is that sampling actions from the policy only requires one forward pass of the one-step distilled policy μ_θ , rather than T forward passes of the velocity field. This accelerates both training and inference time.

Connection to Wasserstein regularization

FQL has an intriguing connection to Wasserstein regularization. For any state s , let $\pi_\beta(s)$ and $\pi_\theta(s)$ be the distributions over actions induced by the multi-step behavior flow and the

one-step distilled policy respectively. It is easy to see that the distillation loss is an upper bound on the squared 2-Wasserstein distance:

$$\begin{aligned} \|\mu_\beta(s, z) - \mu_\theta(s, z)\|_2^2 &\geq \inf_{p \in \Lambda(\pi_\beta(s), \pi_\theta(s))} \mathbb{E}_{a_\beta, a_\theta \sim p} [\|a_\beta - a_\theta\|_2^2] \\ &= W_2(\pi_\beta(s), \pi_\theta(s))^2, \end{aligned} \quad (4.6)$$

where $\Lambda(\pi, \pi')$ is the set of couplings of action distributions π and π' , and W_2 denotes the 2-Wasserstein distance with the Euclidean metric in the action space.

Compared to the standard KL-constrained policy optimization objective, Wasserstein distance offers a metric-aware alternative that can be beneficial when there is a support mismatch between the behavior distribution and the optimal distribution.

Discussion

Up until now, we have introduced a new RL algorithm, FQL, and discussed how we could use it to address the unstable backpropagation issue via one-step distillation. While the algorithm is simple and effective (as we will show in Section 4.4), it does require distillation that may limit expressivity. Indeed, there has been a rich literature in image generative modeling on the trade-off between generation steps and generation quality [138, 160]. While there has been a lot of progress in enhancing the generation quality with few diffusion/flow steps, distillation still leads to performance degradation. This is reflected in recent robotic learning systems [83, 49, 65] where they directly use multi-step diffusion/flow models to parameterize action chunking policies without distillation. The natural question arises—is there a way to efficiently optimize flow policies without distillation?

4.2 Q-learning with Adjoint Matching

This section introduces **Q-learning with Adjoint Matching (QAM)**, a new RL algorithm that applies adjoint matching [35], a recently developed technique in generative modeling, to policy optimization. QAM directly fine-tunes the full flow model without distillation, completely eliminating the expressivity issue in FQL. Like FQL, QAM leverages the critic’s action gradient for efficient policy optimization. The difference lies in how the gradient is being used. QAM operates under a more standard KL constraint, rather than the less commonly used geometric-aware Wasserstein constraint discussed in Section 4.1.

In general, the standard KL constrained optimization problem on a flow model can be formulated as a stochastic optimal control (SOC) objective, which can be solved by using the continuous adjoint method [116]. However, this standard formulation has the same loss landscape as directly backpropagating through the SOC objective, causing instability. Instead, we leverage a modified objective from [35] that admits the same optimal solution, but does not suffer from the instability challenge.

At a high level, QAM transforms the critic’s gradient at noiseless actions using a behavior flow policy, independent from the potentially ill-conditioned flow policy being optimized, to construct unbiased gradient estimates for optimizing the flow policy’s velocity field at intermediate denoising steps. This allows the velocity field to align directly with the optimal one implied by the critic and the KL behavior constraint, without direct and potentially unstable backpropagation, while preserving the full expressivity of multi-step flow models. By combining this policy extraction procedure with a standard temporal-difference (TD) backup for critic learning, QAM enables the flow policy to efficiently converge to the optimal policy subject to the behavior constraint. This is in contrast to most existing flow RL methods that rely on approximations that lack such optimum guarantees (see more discussion in Section 3.2).

The algorithm

In this section, we describe in detail how our method leverages adjoint matching to directly align the flow policy with the prior-regularized optimal policy without suffering from backpropagation instability.

To start with, we first define the optimal policy that we want to learn as the solution of the best policy under the standard KL behavior constraint:

$$\arg \max_{\pi} \mathbb{E}_{a \sim \pi(\cdot | s)} [Q(s, a)] \quad \text{s.t.} \quad D_{\text{KL}}(\pi \parallel \pi_{\beta}) \leq \epsilon(s). \quad (4.7)$$

or equivalently, for an appropriate $\tau(s)$, [101] show that

$$\pi^*(a | s) \propto \pi_{\beta}(a | s) e^{\tau(s) Q_{\phi}(s, a)}, \quad (4.8)$$

where $\tau : \mathcal{S} \rightarrow \mathbb{R}^+$ is the inverse temperature coefficient that controls the strength of the behavior constraint at each state.

Similar to FQL, we train a behavior flow policy using the standard flow-matching loss:

$$L_{\text{FM}}(\beta) = \mathbb{E}_{(s, a) \sim D, t \sim \text{Unif}[0, 1], z \sim \mathcal{N}(0, I_d)} [\|v_{\beta}(s, ta + (1 - t)z, t) - (a - z)\|_2^2]. \quad (4.9)$$

We then parameterize our approximation of the optimal policy as another flow model $v_{\theta} : \mathcal{S} \times \mathbb{R}^A \times [0, 1] \rightarrow \mathbb{R}^A$ and solve the following SOC equation:

$$L(\theta) = \mathbb{E}_{s \sim D, a_t} \left[\int_0^1 \frac{2}{\sigma_t^2} \|v_{\theta}(s, a_t, t) - v_{\beta}(s, a_t, t)\|_2^2 - \tau(s) Q_{\phi}(s, a_1) dt \right], \quad (4.10)$$

where $\mathbf{a} = \{a_t\}_t$ is defined by the following ‘memory-less’ SDE with $\sigma_t = \sqrt{2(1 - t)/t}$:

$$da_t = (2v_{\theta}(s, a_t, t) - a_t/t)dt + \sqrt{2(1 - t)/t}dW_t. \quad (4.11)$$

Similar to the derivation by [35], the memoryless property allows us to directly conclude that the SOC equation has the optimum at

$$\pi_\theta(\cdot | s) \propto \pi_\beta(\cdot | s) e^{\tau(s) Q_\phi(s, a)} \quad (4.12)$$

where $\pi_\theta(\cdot | s)$ and $\pi_\beta(\cdot | s)$ are the corresponding action distributions defined by v_θ and v_β .

However, directly solving the SOC equation involves backpropagation through the numerical integration of the SDE, which introduces additional instability. To mitigate this issue, we use the adjoint matching objective proposed by [35] (Equation (2.18)) to construct a similar objective for policy optimization in our case:

$$L_{\text{AM}}(\theta) = \mathbb{E}_{s \sim D, \{a_t\}_t \sim \text{SDE}} \left[\int_0^1 \|2(v_\theta(s, a_t, t) - v_\beta(s, a_t, t))/\sigma_t + \sigma_t \tilde{g}_t\|_2^2 dt \right]. \quad (4.13)$$

where $\tilde{g}_t$ is the ‘lean’ adjoint state defined by a reverse ODE constructed from $\mathbf{a} = \{a_t\}_t$:

$$d\tilde{g}_t = -\nabla_{a_t} [2v_\beta(s, a_t, t) - a_t/t] \tilde{g}_t dt, \quad (4.14)$$

with the boundary condition:

$$\tilde{g}_1 = -\tau \nabla_{a_1} Q_\phi(s, a_1). \quad (4.15)$$

Unlike the original SOC objective (Equation (4.10)) from which calculating the gradient requires backpropagating through an SDE, which suffers from stability challenges, the adjoint matching objective is constructed without direct backpropagation. Instead, it uses the behavior velocity field v_β to calculate the ‘lean’ adjoint states $\{\tilde{g}_t\}_t$ through a series of VJPs for every SDE trajectory $\mathbf{a} = \{a_t\}_t$, which are then used to form a squared loss in the adjoint matching objective. Mathematically, backpropagation can also be interpreted as calculating the adjoint states through a series of VJPs, with the key distinction that the VJPs are computed under the flow model that is being optimized (*i.e.*, v_θ). This is an important distinction because for direct backpropagation, any ill-conditioned action gradient in v_θ (*i.e.*, $\nabla_a v_\theta(s, a, t)$) would compound over the entire denoising process, contributing to the ‘ill-condition-ness’ of the overall gradient to the parameter θ , which can in turn destabilize the whole optimization process. In contrast, in adjoint matching, the action gradient of v_θ has no contribution to the overall gradient to θ , allowing the optimization to be much more stable.

Finally, we combine the policy optimization with the standard TD-learning objective similar to what was done in FQL:

$$L(\phi) = \mathbb{E}_{s, a, s', r \sim D} [(Q(s, a) - r - \gamma Q_{\bar{\phi}}(s', a'))], \quad a' \leftarrow \mu_\theta(s', z), \quad z \sim \mathcal{N}, \quad (4.16)$$

where again $\mu_\theta(s', a'_0) := \int_0^1 v_\theta(s', a'_t, t) dt$ and $\bar{\phi}$ is the exponential moving average of ϕ with a time-constant of $\lambda = 0.005$ (*i.e.*, $\bar{\phi}_{i+1} \leftarrow (1 - \lambda)\bar{\phi}_i + \lambda\phi_i$ for each training step i).

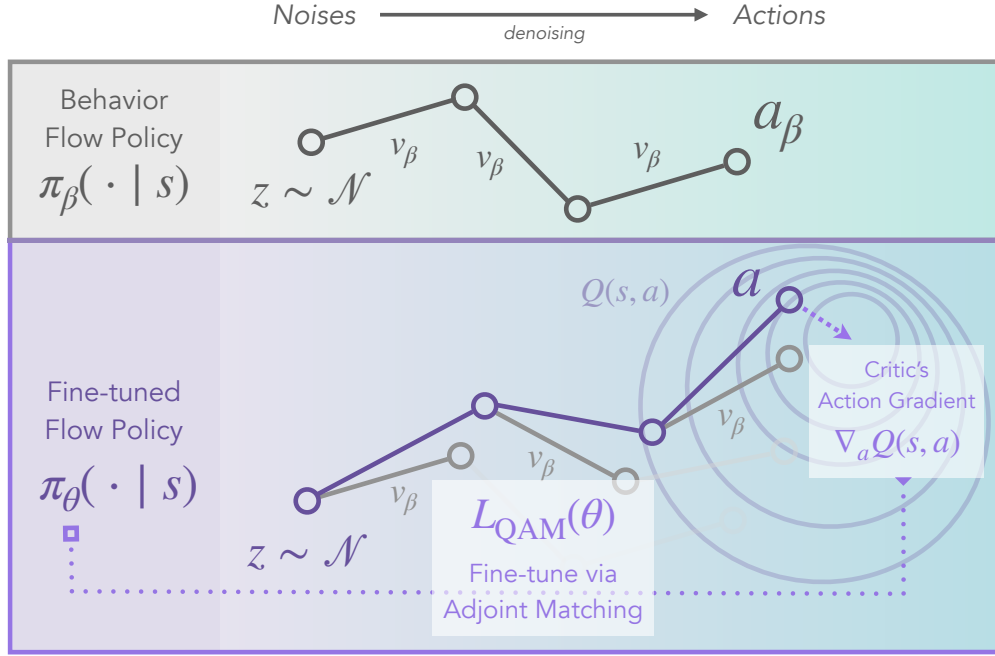


Figure 4.1: **QAM: Q-learning with Adjoint Matching.** QAM uses adjoint matching [35] that leverages the critic’s action gradient directly to fine-tune a flow policy towards the optimal behavior-constrained policy: $\pi_\theta(\cdot | s) \propto \pi_\beta(\cdot | s)e^{Q(s, \cdot)}$.

Practical Considerations

In practice, following [35], we solve both the SDE and the reverse ODE with discrete approximation and a fixed step size of $h = 1/T$, where again T is the number of integration steps. In particular, with $a_0 \sim \mathcal{N}$ and $z_t \sim \mathcal{N}, \forall t \in \{0, h, \dots, (T-1)h\}$, the forward SDE process is approximated by

$$a_{t+h} \leftarrow a_t + h \cdot (2v_\theta(s, a_t, t) - a_t/t) + \sqrt{2h(1-t)/t}z_t. \quad (4.17)$$

We set the boundary condition as $\tilde{g}_1 = -\tau \nabla_{a_1} Q_\phi(s, a_1)$, where we use a state-independent inverse temperature coefficient τ to modulate the influence of the prior π_β and we additionally clip the magnitude of the parameter gradient element-wise by 1 for numerical stability. The backward adjoint state calculation process is then approximated by

$$\tilde{g}_{t-h} \leftarrow \tilde{g}_t + h \cdot \text{VJP}(\nabla_{a_t}(2v_\beta(s, a_t, t) - a_t/t), \tilde{g}_t), \quad (4.18)$$

with $\text{VJP}(\nabla_{\mathbf{y}}b(\mathbf{y}), \mathbf{x}) = [\nabla_{\mathbf{y}}b(\mathbf{y})]\mathbf{x}$ being the vector-Jacobian product and it can be practically implemented by carrying the ‘gradient’ $\mathbf{x}$ with backpropagation through b .

For all our experiments, we do not use a separate training process for v_β and instead train it at the same time as v_θ and Q_ϕ , following [108, 81]. For all our loss functions, the transition

Algorithm 2 Q-learning with Adjoint Matching (QAM).

```

function SDE( $s; v$ )
  for  $t = 0, h, \dots, 1 - h$  do
     $z \sim \mathcal{N}(0, I_d)$ 
     $a_{t+h} \leftarrow a_t + h \cdot [2v(t, s, a_t) - a_t/t] + \sqrt{2h(1-t)/t}z$   $\triangleright$  Euler's method
  return  $\mathbf{a} = \{a_0, a_h, \dots, a_1\}$ 

function LeanAdj( $s, \mathbf{a}, \tilde{g}_1; v_\beta$ )
  for  $t = 1, 1 - h, \dots, h$  do
     $\tilde{g}_{t-h} \leftarrow \tilde{g}_t + h \cdot \text{VJP}(\nabla_{a_t}(2v_\beta(t, s, a_t) - a_t/t), \tilde{g}_t)$ 
  return  $\mathbf{g} = \{g_0, g_h, \dots, g_{1-h}\}$ 

for each training iteration do
   $(s, a, r, s') \sim D$ 
  Critic update:  $L(\phi) = \mathbb{E}_{z \sim \mathcal{N}(0, I_d)} [(Q_\phi(s, a) - r - \gamma Q_\phi(s', \text{ODE}(s', z; v_\theta)))^2]$ 
   $\mathbf{a} = \{a_0, a_h, \dots, a_1\} \sim \text{SDE}(s, v_\theta)$ 
   $\tilde{\mathbf{g}} \leftarrow \text{LeanAdj}(s, \mathbf{a}, -\tau \nabla_{a_1} Q_\phi(s, a_1); v_\beta)$ 
   $\sigma_t \leftarrow \sqrt{2(1-t)/t}$ 
  Actor update:  $L(\theta) = \sum_{t \in \{0, h, \dots, 1-h\}} \|2(v_\theta(s, a_t, t) - v_\beta(s, a_t, t))/\sigma_t + \sigma_t \tilde{g}_t\|_2^2$ 

Output:  $v_\theta$ 

```

tuple (s, a, s', r) is drawn from D uniformly. During offline training, D is the offline data. During online fine-tuning, D is a combination of the offline and online replay buffer data without any re-weighting.

Expanding Beyond KL behavior constraint

While our method, QAM, is guaranteed to converge to the optimal behavior-constrained policy (*i.e.*, $\propto \pi_\beta \exp(Q)$), it can struggle to represent any policy that has a support mismatch with the behavior policy. For tasks where the optimal actions have extremely low probability under the behavior distribution, our algorithm may have trouble to represent an optimal policy for these tasks. In practice, we often find it to be beneficial to relax the behavior constraint in QAM to allow actions that are ‘close’ (in the action space) to the behavior actions. This amounts to optimizing the policy under the following constraint:

$$\arg \max_{\pi, \tilde{\pi}} \mathbb{E}_{a \sim \pi(\cdot|s)} [Q(s, a)] \quad \text{s.t.} \quad D_{\text{KL}}(\tilde{\pi} \parallel \pi_\beta) \leq \epsilon(s), W_q(\pi, \tilde{\pi}) \leq \sigma_a, \quad (4.19)$$

where W is the q -Wasserstein distance between $\pi, \tilde{\pi}$ under some metric $d : \mathcal{A} \times \mathcal{A} \mapsto \mathbb{R}_0^+$, and σ_a is some constant that controls the strength of the constraint:

$$W_q(\pi(\cdot | s), \tilde{\pi}(\cdot | s)) := \inf_{c \in \mathcal{C}(\pi, \tilde{\pi})} \left[\mathbb{E}_{a, \tilde{a} \sim c(a, \tilde{a}|s)} [d(a, \tilde{a})^q]^{1/q} \right], \quad (4.20)$$

	Action Gradient ($\nabla_a Q(s, a)$)	No BPTT	Optimum ($\propto \pi_\beta e^Q$)	Multi-step Flow/Diffusion
ReBRAC	○	○		
FBRAC	○			○
BAM	○		○	○
FQL	○	○		
FAWAC		○	○	○
QSM/DAC/CGQL*	○	○		○
DSRL	○	○		○
FEdit	○	○		○
IFQL		○		○
QAM	○	○	○	○

Table 4.1: **Conceptual differences between QAM and prior methods.** QAM is the only method that (1) leverages the action gradient, (2) avoids backpropagation through time, (3) recovers the optimal behavior-constrained policy, and (4) preserves the expressivity of a multi-step flow model.

where $\mathcal{C}(\pi, \tilde{\pi})$ is the set of all couplings between π and $\tilde{\pi}$. With such design, π not only respects KL behavior constraint, but can also represent actions that are ‘close’ (in the geometry of the action space) to the behavior policy. While directly optimizing this objective is intractable, we approximate it by using the solution from QAM (*i.e.*, π_θ) to approximate the optimal $\tilde{\pi}$, allowing us to transpose the problem into a behavior-constrained policy optimization problem with π_θ being the base policy:

$$\pi^* \approx \pi_\omega = \arg \max_{\pi} \mathbb{E}_{a \sim \pi(\cdot|s)} [Q(s, a)] \quad \text{s.t.} \quad W_q(\pi(a|s), \pi_\theta(a|s)) \leq \sigma_a. \quad (4.21)$$

In practice, we explore two practical variants for optimizing π with different choices of q and d and approximations.

The first variant, QAM-FQL: For our first variant, we use $q = 2$ and $d(a, \tilde{a}) = \|a - \tilde{a}\|_2$, which allows us to directly use a prior flow RL method, FQL [108]. In particular, we learn a 1-step noise-conditioned policy, π_ω (as represented by $\mu_\omega(s, z) : \mathcal{S} \times \mathbb{R}^A \mapsto \mathbb{R}^A$), and optimize the following objective (with α being a tunable BC coefficient),

$$L_{\text{QAM-FQL}}(\omega) = \mathbb{E}_{z \sim \mathcal{N}} [-Q_\phi(s, \mu_\omega(s, z)) + \alpha \|\mu_\omega(s, z) - \mu_\theta(s, z)\|_2^2], \quad (4.22)$$

where again μ_θ is the solution of the ODE defined by the velocity field v_θ .

As we have discussed above in Equation (4.6), the second term in the equation above is an upper-bound on squared Wasserstein distance $W_2^2(\pi_\omega, \pi_\theta)$ between the 1-step noise-conditioned policy π_ω and the optimal behavior-constrained policy obtained via QAM, π_θ . This is exactly the desired constraint (*i.e.*, $W_2^2(\pi_\omega, \pi_\theta) \leq \epsilon_W$). We then use this 1-step policy π_ω both to interact with the environment and to compute value targets (*i.e.*, $\bar{Q}(s', a' = \mu(s, z \sim \mathcal{N}))$). Intuitively, the 1-step policy is optimized to remain close to the QAM-fine-tuned flow policy while also maximizing the action value under the current critic. We call this first variant of our method QAM-FQL.

The second variant, QAM-EDIT: Our second variant uses $q = 1$ and $d(a, \tilde{a}) = \|a - \tilde{a}\|_\infty$. To implement this, we learn an edit policy $\pi_\omega(\Delta_a \mid s, a)$ that predicts an action edit Δ_a to modify the action output from π_θ by at most σ_a in the L_∞ distance (*i.e.*, $\|\Delta_a\|_\infty \leq \sigma_a$). This ensures that the Wasserstein distance constraint (*i.e.*, $W_1(\pi_\omega, \pi_\theta) \leq \sigma_a$) is enforced by construction. In practice, we train an edit policy [37], $\pi_\omega(\cdot \mid s, \tilde{a})$, where it first predicts a Gaussian distribution, which is then squashed by a tanh function to be between -1 and 1 , and finally rescaled to be between $-\sigma_a$ and σ_a as the action edit. To avoid the degenerate case where the edit policy always outputs the boundary action edits (*i.e.*, $-\sigma_a$ or σ_a), we also include the standard entropy term from [56] with the automatic entropy tuning trick such that the policy’s entropy is close to a constant value of $\mathcal{H}(\pi_\omega) = \mathcal{H}_{\text{target}} = -A/2$.

$$\begin{aligned} L_{\text{QAM-EDIT}}(\omega) &= \mathbb{E}_{\Delta_a \sim \pi_\omega(\cdot \mid s, a)} [-Q_\phi(s, \Delta_a + a) + \eta \log \pi_\omega(\Delta_a \mid s, a)], \\ L(\eta) &= \eta [\mathbb{E}_{\Delta_a \sim \pi_\omega(\cdot \mid s, a)} [\log \pi_\omega(\Delta_a \mid s, a)] - \mathcal{H}_{\text{target}}], \end{aligned} \quad (4.23)$$

where $a \sim \pi_\theta(\cdot \mid s)$ for both losses and again Δ_a is restricted to be $\|\Delta_a\|_\infty \leq \sigma_a$. Similarly to our first variant, we use π_ω both to interact with the environment and to compute value targets. A potential side benefit of the entropy term is that it can also encourage action diversity that can be helpful for online exploration and fine-tuning. As we will show in our experiments, this QAM variant not only excels in offline RL but also fine-tunes effectively online.

4.3 Q-Chunking: Leveraging Flow RL Algorithms for Action Chunking Policy Optimization

In the adjacent field of imitation learning (IL), a widely used approach in recent years has been to employ *action chunking*, where instead of training policies to predict a single action based on the state observation from prior data, the policy is instead trained to predict a short sequence of future actions (an “action chunk”) [164, 24]. While a complete explanation for the effectiveness of action chunking in IL remains an open question, its effectiveness can be at least partially ascribed to better handling of non-Markovian behavior in the offline data, essentially providing a more powerful tool for modeling the kinds of complex distributions that might occur in (for example) human-provided demonstrations or mixtures of different

behaviors [164]. However, action chunking has not been used widely in RL, perhaps because optimal policies in fully observed MDPs are Markovian [141], and therefore chunking may appear unnecessary. More importantly policy optimization for action chunking policies is intrinsically more difficult compared to one-step policies because the network needs to predict a higher-dimensional action (*i.e.*, $h \times$ larger with h being the sequence length). This not only increases the challenge of capturing behaviors in the prior data, but also makes online exploration more challenging as now the policy needs to explore in the larger temporally extended action space.

In IL, action chunking is often paired with generative modeling to combat the high-dimensional complex action chunk distribution. Up until now in this dissertation, we have discussed how we could practically and efficiently train flow policies algorithms that leverage one-step distillation (FQL) and adjoint matching (QAM) respectively for efficient flow policy optimization. It turns out that these advances are the foundation of practical, efficient RL algorithms for action chunking policies.

We introduce a general recipe, **Q-chunking**, that allows many existing flow RL algorithms to be adapted directly to action chunking policies. The recipe consists of two simple design principles: (1) Q-learning on a temporally extended action space where each action is an action chunk, and (2) imposing behavior constraint in this extended action space. Behavior constraint allows our action chunking policies to stay close to the prior behavior in the data (*e.g.*, demonstration data, noisy experts), which is often temporally coherent. Even though we might desire a final optimal Markovian policy in most RL tasks, incorporating non-Markovian, temporally coherent behaviors in our policies make online exploration more effective, resulting better sample efficiency. Furthermore, Q-chunking improves the stability and efficiency of TD-based RL, by enabling unbiased n -step backups.

Q-learning on a temporally extended action space

The first design principle of Q-chunking is to apply Q-learning on the temporally extended action space. Unlike normal 1-step TD-based actor-critic methods, which train a Q-function $Q_\phi(s_t, a_t)$ and a policy $\pi_\theta(a_t | s_t)$, we instead train both the critic and the actor with a span of h consecutive actions:¹

$$\text{Q-Chunking Policy: } \pi_\theta(\mathbf{a}_{t:t+h} | s_t) := \pi_\theta(a_t, a_{t+1}, \dots, a_{t+h-1} | s_t) : \mathcal{S} \rightarrow \Delta_{\mathcal{A}^h}$$

$$\text{Q-Chunking Critic: } Q_\phi(s_t, \mathbf{a}_{t:t+h}) := Q_\phi(s_t, a_t, a_{t+1}, \dots, a_{t+h-1}) : \mathcal{S} \times \mathcal{A}^h \rightarrow \mathbb{R}$$

In practice, this involves updating the critic and the actor on batches of transitions consisting of a random state s_t , an action sequence $\mathbf{a}_{t:t+h}$ followed by the state, and the state h steps

¹We use $\mathbf{a}_{t:t+h}$ (or simply $\mathbf{a}_t$) to denote a concatenation of h consecutive actions: $[a_t \ \dots \ a_{t+h-1}] \in \mathbb{R}^{Ah}$ for notation convenience. This is similar for $\mathbf{s}_{t:t+h}$ and $\mathbf{r}_{t:t+h}$.

into the future, s_{t+h} . Specifically, we train Q_θ with the following TD loss,

$$L(\phi) = \mathbb{E}_{s_t, \mathbf{a}_{t:t+h}, s_{t+h} \sim D} \left[\left(Q_\phi(s_t, \mathbf{a}_{t:t+h}) - \sum_{t'=0}^{h-1} \gamma^{t'} r_{t+t'} - \gamma^h Q_{\bar{\phi}}(s_{t+h}, \mathbf{a}_{t+h:t+2h}) \right)^2 \right]. \quad (4.24)$$

with $\mathbf{a}_{t+h:t+2h} \sim \pi_\theta(\cdot \mid s_{t+h})$, and $\bar{\phi}$ being the target network parameters that are often an exponential moving average of ϕ [56].

The TD loss above shares a striking similarity to the n -step return in Equation 2.3 (with n matching h) but with a crucial difference — the Q -function used in the n -step return backup takes in only one action (at time step t) whereas our Q -function takes in the whole action sequence. The implication of this difference can be best explained after we write out the TD backup equations for standard 1-step TD, n -step return, and Q-chunking:

$$Q(s_t, a_t) \leftarrow r_t + \gamma Q(s_{t+1}, a_{t+1}), \quad (\text{standard 1-step TD}) \quad (4.25)$$

$$Q(s_t, a_t) \leftarrow \underbrace{\sum_{t'=t}^{t+h-1} \left[\gamma^{t'-t} r_{t'} \right]}_{\text{biased}} + \gamma^h Q(s_{t+h}, a_{t+h}), \quad (n\text{-step return, } n = h) \quad (4.26)$$

$$Q(s_t, \mathbf{a}_{t:t+h}) \leftarrow \underbrace{\sum_{t'=t}^{t+h-1} \left[\gamma^{t'-t} r_{t'} \right]}_{\text{unbiased}} + \gamma^h Q(s_{t+h}, \mathbf{a}_{t+h:t+2h}). \quad (\text{Q-chunking}) \quad (4.27)$$

For the standard 1-step TD, each backup step propagates the value back by only 1 time step. n -step return propagates the value back $h \times$ faster, but can suffer from a *biased* value estimation issue when $s_{t:t+h}$ and $\mathbf{a}_{t:t+h}$ are off-policy [44]. This is because the discounted sum of the n -step rewards $\mathbf{r}_{t:t+h}$ from the dataset or replay buffer is no longer an unbiased estimate of the expected n -step rewards under the current policy π . Q-chunking value backup is similar to the n -step return where each step also propagates the value back by h time steps, but *does not* suffer from this biased estimation issue. Unlike n -step return where we are propagating the value to a 1-step Q -function, Q-chunking backup propagates the value back to a h -step Q -function that takes in the exact same actions that are taken to obtain the n -step rewards $\mathbf{r}_{t:t+h}$, eliminating the biased value estimation. One caveat of this argument is that it *only holds* under some data distribution assumption (*e.g.*, when the dynamics is deterministic), which we will formalize in Chapter 5. Regardless, Q-chunking value backup does not suffer from the same bias in the naïve n -step return value backup. This allows Q-chunking to learn from arbitrarily sub-optimal data, while enjoying the value learning speedup.

Behavior constraint in the temporally extended action space

The second design principle of Q-chunking addresses the action incoherency issues by leveraging a behavior constraint in the objective for the π_ψ :

$$L(\psi) = -\mathbb{E}_{\substack{s_t \sim D \\ \mathbf{a}_{t:t+h} \sim \pi_\theta(\cdot | s_t)}} [Q_\phi(s_t, \mathbf{a}_{t:t+h})], \text{ s.t. } \text{Dist}(\pi_\theta(\mathbf{a}_{t:t+h} | s_t), \pi_\beta(\mathbf{a}_{t:t+h} | s_t)) \leq \varepsilon. \quad (4.28)$$

where we denote $\pi_\beta(\mathbf{a}_{t:t+h} | s_t)$ as the behavior distribution in the offline data D , and Dist as some distance metric that measures how different the learned policy π deviates from π_β .

Intuitively, a behavior constraint on the temporally extended action sequence allows us to leverage temporally coherent action sequences in the offline dataset. This is particularly advantageous in the temporally extended action space compared to the original action space because offline data often exhibit non-Markovian structure (*e.g.*, from scripted policies [111], human tele-operators [91], or noisy expert policies for sub-tasks [111, 48]) that cannot be well captured by a Markovian behavior constraint. Temporally coherent actions are desirable for online exploration because they resemble temporally extended skills (*e.g.*, moving in a certain direction for navigation, jumping motions for going over obstacles) that help traverse the environment in a structured way rather than using random actions that often result in data that is localized near the initial states. Imposing behavior constraint for an action chunking policy is a very simple way to approximately extract skills without the need to train policy with bi-level structure as often necessitated by skill-based methods (see more discussion in Section 3.5). We observe that Q-chunking, with such behavior constraints, can effectively explore online with temporally coherent actions (see Section 4.3).

Applying Q-chunking to existing flow RL methods

Now, we can apply these two principles to existing RL algorithms. We focus on two examples. The first is best-of-N (BFN), a commonly used policy extraction method in the flow RL literature, and the second is FQL, our new RL algorithm that uses distillation to make policy optimization tractable and efficient.

Q-chunking with Best-of-N (QC) We consider a KL constraint on our policy through the learned behavior distribution:

$$D_{\text{KL}}(\pi_\theta(\cdot | s) \| \pi_\beta(\cdot | s)) \leq \varepsilon. \quad (4.29)$$

While it is possible to include the KL as part of the loss, estimating the KL divergence or log probability for flow models is practically challenging. Best-of- N sampling [140] is a simple, convenient approach that maximizes Q -value while imposing this KL constraint implicitly. Practically, this involves first sampling N action chunks from the learned behavior policy $\pi_\beta(\cdot | s)$,

$$\{\mathbf{a}^1, \mathbf{a}^2, \dots, \mathbf{a}^N\} \sim \pi_\beta(\cdot | s),$$

and then picking the action chunk that maximizes the temporally extended Q -function:

$$\mathbf{a}^* \leftarrow \arg \max_{\mathbf{a} \in \{\mathbf{a}^1, \mathbf{a}^2, \dots, \mathbf{a}^N\}} Q(s, \mathbf{a}).$$

Best-of- N sampling implicitly optimizes the constrained KL objective because its distribution admits a closed-form upper bound on the KL divergence [60]:

$$D_{\text{KL}}(\pi_{\text{BFN}}(\cdot | s) \| \pi_\beta(\cdot | s)) \leq \log N - \frac{N-1}{N}, \quad (4.30)$$

where $\pi_{\text{BFN}}(\cdot | s)$ is the distribution over the action chunks induced by the rejection sampling procedure. Tuning the value of N directly controls the strength of the constraint. The benefit of approximating the policy optimization objective (Equation 4.28) with best-of- N sampling is that we can completely avoid separately parameterizing a policy π_θ and only sample from the behavioral policy π_β . In particular, we use best-of- N sampling to generate actions to both (1) interact with the environment, and (2) provide the action samples in the TD backup following [55]. The resulting algorithm has only one additional loss function:

$$L(\phi) = \mathbb{E}_{\substack{(\mathbf{s}_{t:t+h+1}, \mathbf{a}_{t:t+h}) \sim D \\ \{\mathbf{a}_{t+h}^i\}_{i=1}^N \sim \pi_\beta(\cdot | s_{t+h})}} \left[(Q_\phi(s_t, \mathbf{a}_t) - R_{t:t+h} - \gamma^h Q_\phi(s_{t+h}, \mathbf{a}_{t+h}^*))^2 \right] \quad (4.31)$$

where again $\mathbf{a}_{t+h}^* := \arg \max_{\mathbf{a} \in \{\mathbf{a}_{t+h}^i\}} Q(s, \mathbf{a})$ and $R_{t:t+h} = \sum_{t'=1}^h \gamma^{t'} r(s_{t'}, a_{t'})$.

While QC is simple and easy to implement, it does come with some additional computational costs associated with best-of- N sampling. The cost scales linearly with the number of samples, N , making it non-ideal for tasks where a large N is needed (*e.g.*, the behavior prior is sub-optimal). Next, we present our second example based on FQL, which avoids this costly sampling procedure.

Q-chunking with FQL (QC-FQL) As we have established in Section 4.1, FQL imposes a Wasserstein distance constraint instead of a KL constraint:

$$W_2(\pi_\theta(\cdot | s), \pi_\beta(\cdot | s)) \leq \varepsilon. \quad (4.32)$$

Recall that FQL parametrizes the policy π_θ with a noise-conditioned action prediction model. In the case of action chunking, the model, $\mu_\theta(s, \mathbf{z}) : \mathcal{S} \times \mathbb{R}^{Ah} \mapsto \mathbb{R}^{Ah}$, takes in a noise of size $\mathbb{R}^{Ah}$ instead of $\mathbb{R}^A$ and also outputs action chunks directly. This noise-conditioned policy μ_θ is trained to maximize the Q-chunking critic $Q_\phi(s_t, \mathbf{a}_{t:t+h})$ while being regularized to be close to a behavioral flow-matching policy parameterized state-conditioned velocity field $v_\beta : \mathcal{S} \times \mathbb{R}^{Ah} \times [0, 1] \rightarrow \mathbb{R}^{Ah}$ via a distillation loss:

$$L(\theta) = \mathbb{E}_{s \sim D, \mathbf{z}^0 \sim \mathcal{N}(0, \mathbf{I}_{Ah})} \left[\alpha \left\| \mu_\theta(s, \mathbf{z}^0) - \mu_\beta(s, \mathbf{z}^0) \right\|_2^2 - Q(s, \mu_\theta(s, \mathbf{z}^0)) \right], \quad (4.33)$$

where $\mu_\beta(s, \mathbf{z}^0)$ is the ODE solution from $u = 0$ to $u = 1$ following $d\mathbf{z}^u = v_\beta(s, \mathbf{z}^u, u)du$ (the initial value $\mathbf{z}^0$ is sampled from the unit Gaussian).² The hyperparameter α directly controls the magnitude of the distillation loss, which corresponds to the strength of the Wasserstein constraint. Finally, the TD loss is similar to the TD loss in QC:

$$L(\phi) = \mathbb{E}_{\substack{(\mathbf{s}_{t:t+h+1}, \mathbf{a}_{t:t+h}) \sim D \\ \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_{Ah})}} \left[\left(Q_\phi(s_t, \mathbf{a}_t) - R_{t:t+h} - \gamma^h Q_\phi(s_{t+h}, \mu_\theta(s_{t+h}, \mathbf{z})) \right)^2 \right], \quad (4.34)$$

where $R_{t:t+h} = \sum_{t'=t}^{t+h-1} \gamma^{t'} r(s_{t'}, a_{t'})$.

Does action chunking improve Q-learning?

Now, we study how effective our recipe, Q-chunking, is to enable flow RL methods to work well in manipulation tasks. In particular, we focus on the two examples we describe in the previous section (*i.e.*, QC and QC-FQL), along with an additional variant, QC-IFQL, which is the result of applying Q-chunking to an existing method introduced in [108], IFQL.

In terms of baselines, we first consider the direct ablations of QC, QC-FQL, QC-IFQL: **BFN** (**best-of- N**) is the same as QC but without using action chunking. **FQL** is the same as QC-FQL but without using action chunking. **IFQL** is a baseline introduced in [108] that combines IQL [75] with rejection sampling policy extraction. Next, we consider their corresponding n -step return baselines, BFN- n , FQL- n , and IFQL- n , where n -step backup is used instead of 1-step TD backup. These baselines enjoy the benefits of value backup speedup, but do not use chunked critic or actor, and potentially suffer from the bias issue as discussed in Section 4.3. We also compare with **RLPD** [9], **RLPD-AC**. RLPD is a sample-efficient RL algorithm that treats offline data as additional off-policy data and learn from scratch online. RLPD-AC is the same as RLPD but operates on the temporally extended action space. Both of them do not use a behavior constraint. In addition, we compare with **SUPE-GT** [156], a recently proposed skill-based method that is designed for offline-to-online RL. The original method is designed to deal with an unlabeled dataset by learning a reward model with reward bonuses. We adapt it to our setting by directly using the ground truth rewards. Finally, we compare with two Gaussian baselines **IQL** [75] and **ReBRAC** [143].

We consider six sparse reward robotic manipulation domains with tasks of varying difficulties. This includes 5 domains (5 tasks each) from OGBench [111], **scene-sparse**, **puzzle-3x3-sparse**, **cube-double/triple/quadruple** and 3 tasks from robomimic [91]. For OGBench, we use the default play-style datasets except for **cube-quadruple** where we use the larger 100M-size dataset. For robomimic, we use the multi-human datasets. See more details in Appendix B.1.

²We use u here instead of t as used in Equation (4.9) to differentiate the denoising time step from the environment step.

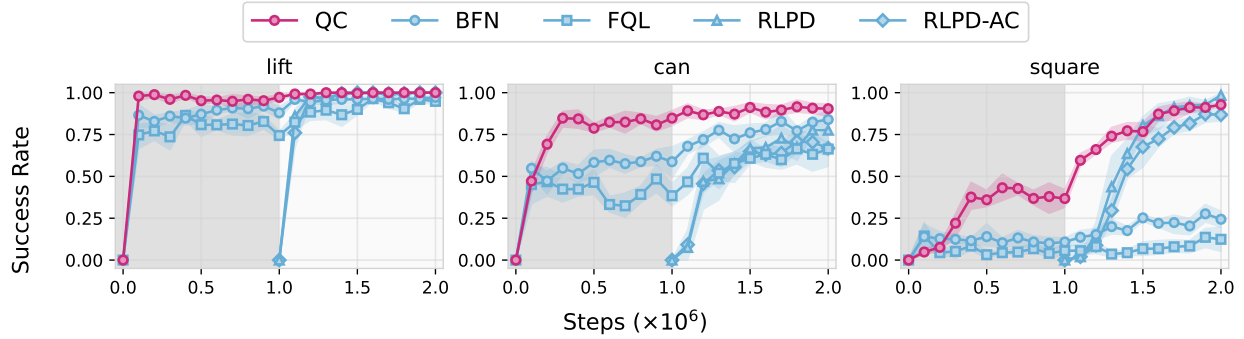


Figure 4.2: **Robomimic results.** QC achieves strong performance across all three robomimic tasks. The first 1M steps are offline and the next 1M steps are online with one environment step per training step (5 seeds).

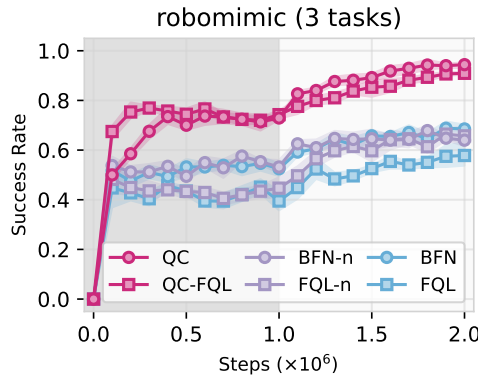


Figure 4.3: **n -step return ablations on robomimic.** Both Q-chunking methods consistently outperform their n -step and 1-step TD counterparts (5 seeds).

Q-chunking improves online sample efficiency. We report the performance of all Q-chunking methods (QC, QC-FQL, QC-IFQL) and the baselines in Table 4.2 (OGBench) and a selective (strong) subset in Figure 4.2 (robomimic). QC achieves competitive performance offline, often matching or sometimes outperforming best prior methods. In the online phase, QC shows strong sample-efficiency, especially on the two hardest OGBench domains (cube-triple/quadruple), where it outperforms all prior methods (especially on cube-quadruple) by a large margin. In addition, on OGBench domains, all Q-chunking methods (QC, QC-FQL, QC-IFQL) outperform both their corresponding 1-step TD counterpart (BFN, FQL, IFQL) and their corresponding n -step return counterpart (BFN-n, FQL-n, IFQL-n). A similar trend continues on robomimic tasks as shown in Figure 4.3. Across OGBench and robomimic, n -step return baselines, which do not use chunked critics or policies, perform significantly worse than Q-chunking methods.

		puzzle-3x3-sparse (5 tasks)	scene-sparse (5 tasks)	cube-double (5 tasks)	cube-triple (5 tasks)	cube-quadruple (5 tasks)	overall (25 tasks)
From Scratch (1-step TD)	RLPD	. → 100	. → 94	. → 99	. → 41	. → 0	. → 67
	RLPD-AC	. → 100	. → 91	. → 96	. → 11	. → 7	. → 61
	SUPE-GT	. → 100	. → 92	. → 67	. → 0	. → 0	. → 52
Gaussian (1-step TD)	IQL	0 → 20	0 → 39	0 → 0	0 → 0	0 → 0	0 → 12
	ReBRAC	55 → 100	11 → 99	3 → 30	0 → 0	1 → 20	14 → 50
Flow (1-step TD)	IFQL	98 → 97	68 → 74	11 → 59	0 → 3	0 → 0	36 → 47
	FQL	100 → 100	58 → 95	29 → 76	1 → 18	0 → 3	37 → 58
	BFN	98 → 100	85 → 99	68 → 79	4 → 23	1 → 12	51 → 63
Flow (n-step TD)	IFQL-n	94 → 100	68 → 92	5 → 29	0 → 0	0 → 0	34 → 44
	FQL-n	98 → 100	18 → 70	11 → 77	0 → 1	7 → 36	27 → 57
	BFN-n	58 → 90	57 → 97	11 → 65	0 → 0	0 → 0	25 → 50
Q-chunking (Ours)	QC-IFQL	100 → 100	83 → 98	12 → 78	0 → 0	1 → 19	39 → 59
	QC-FQL	63 → 100	84 → 99	39 → 100	4 → 53	1 → 77	38 → 86
	QC	100 → 100	84 → 99	67 → 98	6 → 64	4 → 73	52 → 86

Table 4.2: **Summary table for OGBench offline-to-online RL results.** For each cell, we report the offline performance after 1M of training steps and then the online performance after 1M of additional online steps. The best method(s) for each column is highlighted in bold and color. Q-chunking methods outperform all prior methods at the end of the online training. See the full results in Table B.5 (complete table) and Figure B.3 (individual plot).

Q-chunking improves online exploration. We hypothesize in Section 4.3 that action chunking policies produce more temporally coherent actions and thus lead to better state coverage and exploration. In this section, we study to what degree that holds empirically. We first visualize the end-effector movements early in the training for **QC** and **BFN** (Figure 4.4, left). **BFN**’s trajectory contains many pauses (as indicated by a very big and dense cluster near the center of the visualization), especially when the end-effector is being lowered to pickup a cube. In contrast, **QC** has fewer pauses (fewer and shallower clusters) and a more diverse state coverage in the end-effector space. We include additional examples in Appendix B.4, Figure B.1 and Figure B.2. To get a quantitative measure of the temporal coherency in actions, we record the 3-D end-effector position throughout training every 5 time steps: $\{\mathbf{x}_0^{\text{ef}}, \mathbf{x}_5^{\text{ef}}, \dots\}$ and compute the average L_2 norm of the difference vector of two adjacent end-effector positions. This average norm would be small if there are any pauses or jittery motions, making a good proxy for measuring the temporal coherency in actions. As shown in Figure 4.4 (right), **QC** exhibits a higher action temporal coherency throughout training compared to **BFN**. This suggests that Q-chunking improves temporal coherency in actions, which explains the improved sample-efficiency that Q-chunking brings.

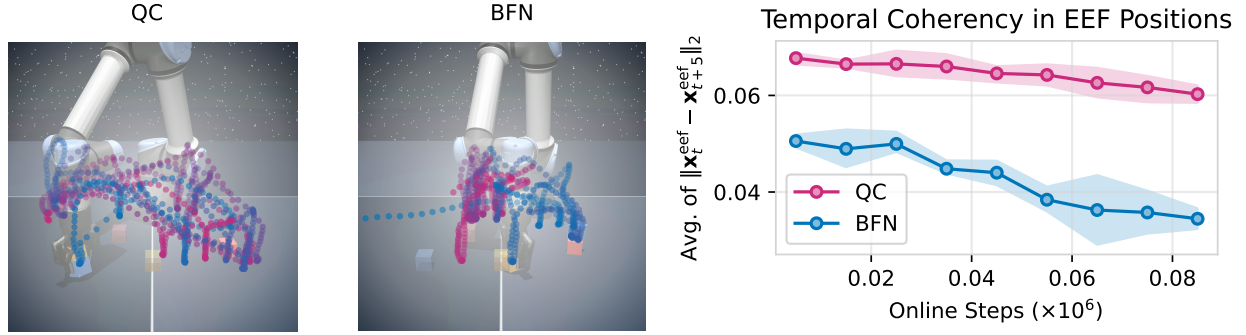


Figure 4.4: **End-effector movements early in the training and temporal coherency analysis on cube-triple-task3.** *Left:* QC covers a more diverse set of states compared to BFN in the first 1000 environment steps. *Right:* QC exhibits a higher temporal coherency in end-effector compared to BFN.

4.4 Comprehensive Evaluations for Flow RL Methods

Up to now, we have introduced two novel RL algorithms, FQL and QAM, and described an effective recipe, Q-chunking, that allow existing flow RL algorithms to work with action chunking policies. In this section, we focus on evaluating flow RL approaches in both the action chunking setting and the non-action chunking setting. For all our experiments in the action chunking setting, we apply Q-chunking to all methods considered.

Domains and datasets

For the evaluation tasks, we reuse many manipulation tasks from the previous section for the action chunking setting, and add locomotion tasks for the non-action chunking setting. We consider 10 domains from OGBench [111]: `scene`, `puzzle-3x3` (p33), `puzzle-4x4` (p44), `cube-double` (c2), `cube-triple` (c3), `cube-quadruple` (c4), `humanoidmaze-medium` (hm), `humanoidmaze-large` (hl), `antmaze-large` (al), and `antmaze-giant` (ag). For `antmaze-*` and `humanoidmaze-*`, we use the default `navigate` datasets. For `scene`, `puzzle-*`, and `cube-*`, we use the default `play` datasets except for c4 and p44 where we use the larger 100M-size dataset from [110], and use the sparse reward definition for {p33, p44, scene}, following [81]. All of these domains require the RL agent to solve long-horizon tasks from diverse offline behavior data that can only be accurately captured by expressive policies like flow/diffusion policies. Furthermore, the harder domains are difficult to solve from offline data alone, making these benchmarks great for evaluating the online fine-tuning effectiveness of our approach. In addition, for all {cube-*, scene-*, p33-*, p44-*} domains, we follow [81] to learn action chunking policies with an action chunking size of $h = 5$. Action chunking policies output high-dimensional actions that exhibits a much more complex behavior distribution, where the policy extraction becomes critical. Since our approach primarily focuses on the policy extraction aspect, these domains make an ideal testbed us to compare

our method to prior work. See Section A.3 for more details on these domains.

Baselines

We carefully select 13 representative, strong baselines that can be roughly categorized into the following 5 categories— **(1) Gaussian:** ReBRAC [143], **(2) Backprop:** FBRAC [108] (backprop through the flow policy’s denoising step directly), FQL (backprop through a 1-step distilled policy) [108]; **(3) Advantage-weighted:** FAWAC [108] (advantage weighted actor critic, AWAC [101], with flow policy); **(4) Guidance with the critic’s action gradient:** DAC [42], QSM [118], and CGQL/CGQL-MSE/CGQL-Linex (three variants of classifier guidance-based methods inspired by [29]); **(5) Post-processing-based:** DSRL [152], FEdit (flow + Gaussian edit policy from [37]), and IFQL (flow counterpart of IDQL [57]). For offline-to-online evaluations we additionally compare with RLPD [9]. Finally, we compare with BAM, a direct ablation from our method QAM where we use the ‘basic’ adjoint matching (and hence the abbreviation BAM) objective in Equation (2.16) instead of the adjoint matching objective in Equation (2.18), and keep the rest of the implementation exactly the same. We categorize it as a “backprop” method because its gradient is equivalent to that of backpropagating through the memoryless SDE as we discuss above in Section 2.3.

Among them, RLPD does not employ any behavior constraint, so we directly train from scratch online. To make the comparison fair, we use $K = 10$ critic networks, pessimistic value backup with $\rho = 0.5$ (except on `humanoidmaze-large` where we find $\rho = 0$ to work better), no best-of- N sampling (*i.e.*, $N = 1$) for both our methods and our baselines except for IFQL where best-of- N is used for policy extraction. See Section A.4 for detailed descriptions and implementation details for each of the baselines. We also include the domain-specific hyperparameters for each baseline in Section A.5.

Which flow RL method attains the best performance?

Table 4.3 reports the offline RL performance across 10 different domains (50 tasks in total). QAM outperforms all prior methods with an aggregated score of 44. In particular, among all the baselines, FAWAC also converges to the optimal behavior-constrained distribution (Equation (4.8)) but does not leverage the action gradient of the critic, resulting in a much worse aggregated score of 8; BAM also leverages the SOC objective but optimizes it with backpropagation through the SDE chain (without using the ‘lean’ adjoint), leading to a worse aggregated score of 35. It is worth noting that FBRAC also leverages backpropagation through time (BPTT) similar to BAM, but performs much worse than BAM. The main distinction between BAM and FBRAC is that BAM leverages the SOC formulation which enjoys the guarantee that the policy at the optimum recovers the optimal behavior-constrained policy, whereas FBRAC has not been found to enjoy such a guarantee. This suggests that carefully designing the loss function where its optimum coincides with the optimal behavior-constrained policy can be influential to the algorithm’s performance.

		al	ag	hm	hl	scene	p33	p44	c2	c3	c4	all
		5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	50 tasks
GAUSSIAN	ReBRAC	94	57	69	17	65	79	0	9	1	9	40
		[94.0]	[57.0]	[69.0]	[17.0]	[65.0]	[79.0]	[0.0]	[9.0]	[1.0]	[9.0]	[40.0]
BACKPROP	FBRAC	2	0	39	0	50	0	15	0	0	0	11
	BAM	84	1	60	5	98	56	0	47	3	0	35
	FQL	76	0	68	9	78	70	5	46	3	2	36
		[84.0]	[1.0]	[60.0]	[5.0]	[98.0]	[56.0]	[0.0]	[47.0]	[3.0]	[0.0]	[35.0]
		[76.0]	[0.0]	[68.0]	[9.0]	[78.0]	[70.0]	[5.0]	[46.0]	[3.0]	[2.0]	[36.0]
ADV. WEIGHTED	FAWAC	17	0	24	0	38	3	0	2	0	0	8
		[17.0]	[0.0]	[24.0]	[0.0]	[38.0]	[3.0]	[0.0]	[2.0]	[0.0]	[0.0]	[8.0]
GUIDANCE	CGQL	76	0	60	5	38	48	24	38	8	0	30
	CGQL-M	71	4	42	6	74	100	0	41	8	1	35
	CGQL-L	65	3	62	6	88	90	0	45	8	0	37
	DAC	88	16	83	0	68	68	0	35	5	3	36
	QSM	90	24	82	6	78	57	0	33	6	19	39
		[76.0]	[0.0]	[60.0]	[5.0]	[38.0]	[48.0]	[24.0]	[38.0]	[8.0]	[0.0]	[30.0]
POST-PROCESSING	DSRL	61	3	53	3	99	87	0	74	1	2	38
	FEdit	58	2	22	3	62	99	34	40	2	5	33
	IFQL	36	1	86	24	84	100	0	11	0	2	34
		[61.0]	[3.0]	[53.0]	[3.0]	[99.0]	[87.0]	[0.0]	[74.0]	[1.0]	[2.0]	[38.0]
		[58.0]	[2.0]	[22.0]	[3.0]	[62.0]	[99.0]	[34.0]	[40.0]	[2.0]	[5.0]	[33.0]
		[36.0]	[1.0]	[86.0]	[24.0]	[84.0]	[100.0]	[0.0]	[11.0]	[0.0]	[2.0]	[34.0]
ADJOINT MATCHING	QAM	81	18	67	11	97	100	0	64	3	3	44
	QAM-F	83	12	65	12	95	99	6	65	3	14	45
	QAM-E	83	1	59	2	97	100	39	65	5	6	46
		[81.0]	[18.0]	[67.0]	[11.0]	[97.0]	[100.0]	[0.0]	[64.0]	[3.0]	[3.0]	[44.0]
		[83.0]	[12.0]	[65.0]	[12.0]	[95.0]	[99.0]	[6.0]	[65.0]	[3.0]	[14.0]	[45.0]
		[83.0]	[1.0]	[59.0]	[2.0]	[97.0]	[100.0]	[39.0]	[65.0]	[5.0]	[6.0]	[46.0]

Table 4.3: **Offline RL performance at 1M training steps (50 tasks, 12 seeds).** QAM and two of its variants, QAM-FQL (QAM-F) and QAM-EDIT (QAM-E) outperform all prior baselines. We use CGQL-M, CGQL-L, QAM-F, QAM-E as the abbreviations for CGQL-MSE, CGQL-Linear, QAM-FQL, and QAM-EDIT respectively. For results on individual tasks, see Table A.1 in Section A.2.

Furthermore, BAM ends up performing similarly to FQL, which addresses the BPTT instability issue by leveraging a 1-step distilled policy. This suggests that BPTT might not be as disadvantageous as previously reported in [108] in the best case, but can be sensitive to implementation details. In addition, we find QSM, one of the earliest methods proposed in the literature of diffusion/flow-matching and RL can perform very well when augmented with a behavior cloning term.³ We also find that a Gaussian baseline, ReBRAC, while worse on manipulation tasks, often outperforms other flow/diffusion-based methods on locomotion tasks. This makes ReBRAC the fifth-best method in terms of aggregated score, falling short

³The original QSM implementation does not have a behavior cloning term. It only aligns the action gradient of the critic to the diffusion denoising process. This makes the vanilla QSM very hard to learn anything during the offline pre-training phase. To adapt it to our offline-to-online setting and make the comparison fair, we augment the original QSM loss with the standard DDPM loss on as the behavior cloning loss. See more details in Section A.4.

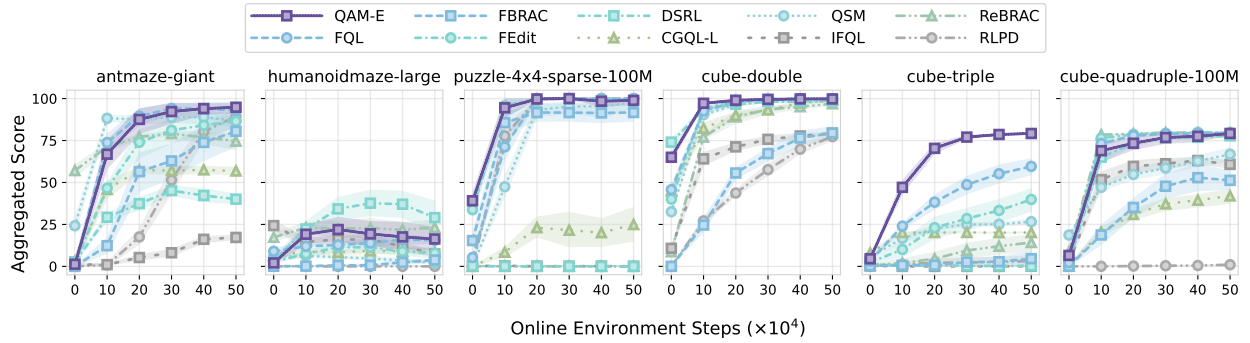


Figure 4.5: **QAM online fine-tunes more effectively than prior methods (50 tasks, 12 seeds).** We use the QAM-EDIT (QAM-E) variant for QAM and the CGQL-L_{inex} variant for CGQL (CGQL-L) because they work well for online fine-tuning. For full results, see Figure A.1 in the Appendix.

compared to QSM and all three QAM variants. Last but not least, combining QAM with FQL and FEdit (QAM-F and QAM-E respectively) can push the performance even further, achieving an aggregated score of 45 and 46 respectively.

Moving on to the online fine-tuning setting, Figure 4.5 shows the sample efficiency curve (with x -axis being the number of environment steps). QAM-E outperforms all prior methods on **cube-triple** and is the most robust method across the board. For example, compared to QAM, QSM fine-tunes better on **ag** but struggles on all other tasks except on **c2** where it performs similar to QAM. FQL fine-tunes slightly better on **ag** but much slower on both **p44** and **c3**.

4.5 Concluding Remark

In this chapter, we introduce two flow RL methods, FQL and QAM, as well as a generic recipe that allows these approaches (along with many other flow RL methods) to be effective in the action chunking setting. Our empirical results suggest that QAM-E (*i.e.*, the variant with residual Gaussian policy) is the most effective policy extraction method in both the offline RL setting and the offline-to-online RL setting, matching or outperforming prior methods. FQL is also a strong alternative, especially during online fine-tuning, where it demonstrates strong sample-efficiency in the most challenging **cube-triple** and **cube-quadruple** domains, surpassed by only QAM-E. There are still practical challenges associated with both FQL and QAM. While their effectiveness can be largely attributed to how well it is able to leverage the critic’s action gradient for efficient policy optimization, this can be a double-edge sword—for cases where the critic function is ill-conditioned, it could lead to optimization stability issue. We have found gradient clipping to help alleviate this issue in QAM, but a more principled

method that combines both value and gradient information could further improve robustness and performance.

On the action chunking side, we have shown that Q-chunking as an effective recipe that allows flow RL methods to be effective in manipulation tasks due to its value speedup and temporal coherent exploration. However, it also comes with a fundamental limitation—action chunking policy executes actions open-loop within chunk, constricting reactivity. This can especially be problematic when the environment is stochastic, where the policy is required to be reactive to compensate for environment stochasticity. Furthermore, Q-learning with action chunking Q-function is an understudied topic with little known theoretical guarantees. In Chapter 5, we will introduce and discuss the first theoretical framework for action chunking Q-learning and turn theoretical insights into a practical algorithm in Chapter 6 that allows action chunking Q-learning to be scaled up to extremely long horizon tasks that were previously impossible to solve with naïve Q-chunking.

Chapter 5

Theoretical Foundation of Action Chunking Q-learning

At the time of writing this dissertation, many methods have been proposed to adopt action chunking in value learning similar to Q-chunking [125, 80, 147]. As we have shown both conceptually and empirically, this formulation enables more efficient value backups and leads to better performance in manipulation tasks. However, theoretical guarantees of action chunking Q-learning, especially on arbitrary off-policy data, are still an open problem.

Intuitively, an action chunking policy implicitly poses an assumption that the action chunks are carried out open-loop. Mathematically, this can be roughly described as

$$\pi_{ac}(a_t, a_{t+1} \cdots, a_{t+h-1} \mid s_t) = P_{\mathcal{D}}(a_t, a_{t+1} \cdots, a_{t+h-1} \mid s_t, s_{t+1}, \cdots, s_{t+h-1}), \quad (5.1)$$

where $\mathcal{D}$ is the distribution of the trajectory as a result of rolling out an action chunking policy. This leads to a problem—when the offline dataset does not satisfy such a relationship, Q-learning on the action chunking space starts to break down. Imagine an environment where there is a random binary noise (*e.g.*, wind blowing a robot to the left or to the right) that a policy needs to react to with the appropriate binary action (*e.g.*, steering to the right or to the left). Failure to react appropriately results in an instant episode termination with negative return. If the offline data is collected by an expert policy that always reacts correctly to the noise, all the trajectories would be successful and free from the negative return. However, this deviates from the performance of any action chunking policies where they are doomed to fail with no mechanism of reacting during action chunks. The worst part is that the Q-function would incorrectly learn that executing the sequence of actions presented within the offline data open-loop results in success, resulting in a systematic open-loop bias that cannot be fixed easily.

The example above gives a glimpse of the nuances in the action chunking Q-learning, and the lack of theoretical understanding in this setting makes it unclear when and how action

Assumptions	Value Estimation Error $ \hat{V}_{ac} - V_{ac} $	AC Optimality $V^* - V_{ac}^+$	Closed-loop AC Optimality $V^* - V^\bullet$
Weak ε_h -OLC	$\Theta(\varepsilon_h H \bar{H})$ (Theorems 1 and 2)	$\Omega(H)$ (Proposition 1)	-
Strong ε_h -OLC	$O(\varepsilon_h H \bar{H})$ (Theorem 1)	$\Theta(\varepsilon_h H \bar{H})$ (Theorems 3 and 4)	$O(\varepsilon_h H^2 \bar{H})$ (Proposition 3)
$(\vartheta_h^L, \vartheta_h^G)$ -BOV	-	$\Omega(H)$ (Theorem 6)	$\Theta(\vartheta_h^L H + \vartheta_h^G H \bar{H})$ (Theorems 5 and 6)

Table 5.1: **Summary of our main theoretical contributions.** In this work, we introduce open-loop consistency (OLC: Definition 2) and bounded optimality variability (BOV: Definition 4). Weak OLC provides guarantees on the value estimation error of the action chunking critic but not the optimality of the learned action chunking policy. Strong OLC provides guarantees on the optimality of the learned action chunking policy and its closed-loop execution performance. BOV is an alternative condition to provide guarantees on the closed-loop execution performance. $\Omega(H)$ means that a constant factor of the maximum value gap can be achieved in the worst case.

chunking Q-learning might work in some environments compared to others. Indeed, what we are dealing with here is no longer a well-understood Markovian policy learning problem, but rather a special case of non-Markovian policy learning problem, where temporal dependency must be factored into the consideration.

In this chapter, we provide the *first* formal analysis of Q-learning with action chunking, focusing on characterizing the value learning bias of the Bellman backup of action chunking critic. Specifically, we introduce the *open-loop consistency* condition under which we *exactly* characterize the worst-case value estimation bias (Theorems 1 and 2) and sub-optimality gap (Theorems 3 and 4) at the fixed point of the bellman optimality equations. Moreover, we characterize (i) the conditions under which the action chunking critic backup is preferable over n -step return backup with a single-step critic (Proposition 2), and (ii) the conditions under which closed-loop execution of the action chunking policy further mitigates the open-loop bias (Theorem 5). A brief summary of the key results is available in Table 5.1.

5.1 Assumptions and Notations

To build the foundation of our analysis, we start by describing the trajectory data distribution that we use for Q-learning and the trajectory distribution induced by an action chunking policy. In particular, we assume that the trajectory data distribution obeys the transition dynamics T :

Assumption 1 (Data Obeys the Transition Dynamics) $\mathcal{D} \in \Delta_{\mathcal{T}}$ is a trajectory distribution generated by rolling out a behavior policy from a distribution of $s_t \sim \mu$. The behavior policy can be non-Markovian (*i.e.*, $\pi_\beta(a_{t+k} \mid s_{t:t+k+1}, a_{t:t+k})$). Each subsequent state is generated according to the MDP dynamics $\mathcal{M}$: $s_{t+k+1} \sim T(\cdot \mid s_{t+k}, a_{t+k})$, $\forall k \in \{0, 1, \dots, h-1\}$. The resulting trajectory is $(s_t, s_{t+1}, \dots, s_{t+h}, a_t, a_{t+1}, \dots, a_{t+h}) \in \mathcal{T} = \mathcal{S}^h \times \mathcal{A}^h$.

Next, we formally define the open-loop trajectory distribution that we would obtain if we take the same actions in the data and roll them out open-loop for h steps in the MDP.

Definition 1 (Open-loop Trajectory) From any data distribution $\mathcal{D}$, we use $\pi_{\mathcal{D}}^{\circ} : \mathcal{S} \rightarrow \Delta_{\mathcal{A}^h}$ to denote an action chunking policy that admits the same marginal distribution as $\mathcal{D}$:

$$\pi_{\mathcal{D}}^{\circ}(a_{t:t+h} \mid s_t) := P_{\mathcal{D}}(a_{t:t+h} \mid s_t). \quad (5.2)$$

Rolling out this action chunking policy by carrying out actions in chunks induces a trajectory distribution $P_{\mathcal{D}}^{\circ} \in \Delta_{\mathcal{S}^{h+1}, \mathcal{A}^h}$ that is generally different from $P_{\mathcal{D}}$:

$$P_{\mathcal{D}}^{\circ}(s_{t+1:t+h+1}, a_{t:t+h} \mid s_t) := \pi_{\mathcal{D}}^{\circ}(a_{t:t+k} \mid s_t) \prod_{k=0}^{h-1} T(s_{t+k+1} \mid s_{t+k}, a_{t+k}). \quad (5.3)$$

Next, we introduce a set of notations and conventions that we use in our theoretical analysis. We use $a_{t:t+h}$ to denote an action chunk of length h : $(a_t, a_{t+1}, \dots, a_{t+h-1})$ (not including a_{t+h}). We use the subscript $[\cdot]_{\text{ac}}$ for all action chunking policies or value functions, $\hat{[\cdot]}$ to denote the *nominal* (i.e., estimated) value (in contrast to the *actual* without the '^'), and $[\cdot]^+$ to denote something that is *learned* from the data (usually defaults to $\mathcal{D}$). For example, $\hat{V}_{\text{ac}}^+ : \mathcal{S} \rightarrow [0, 1/(1 - \gamma)]$ is the *nominal* value (i.e., expected discounted return) of an action chunking policy π_{ac}^+ learned from $\mathcal{D}$, whereas V_{ac}^+ is the *actual* value of the same action chunking policy (where the value is obtained by rolling the policy out in the MDP with open-loop action chunks). As we will elucidate in the next section, the *nominal* value and the *actual* value of a policy are usually different, and hence making this differentiation critical in our analysis. We also use $[\cdot]^*$ to denote the optimal policy or value function under the constraint of the policy class (e.g., π_{ac}^* for the optimal action chunking policy and π^* for the optimal closed-loop 1-step policy). Finally, we use $H = 1/(1 - \gamma)$, $\bar{H} = 1/(1 - \gamma^h)$ to denote the effective horizon for 1-step TD backup and h -step TD backup respectively.

5.2 Weak and Strong Open-loop consistency (OLC)

From the definition above, we have demonstrated that replaying the actions from the trajectory data distribution $P_{\mathcal{D}}$ in an open-loop manner may result in a different trajectory distribution, $P_{\mathcal{D}}^{\circ}$. This discrepancy between $P_{\mathcal{D}}^{\circ}$ and $P_{\mathcal{D}}$ has not been carefully analyzed by prior work but can play a huge role in the optimal policy that action chunking Q-learning converges to. To characterize this discrepancy, we use a notion of consistency as defined below.

Definition 2 (Open-Loop Consistency) $\mathcal{D}$ is **weakly** ε_h -open-loop consistent if for every $s_t \in \mathcal{S}$ with $P_{\mathcal{D}}(s_t) > 0$ (i.e., $s_t \in \text{supp}(P_{\mathcal{D}}(s_t))$),

$$D_{\text{TV}}(P_{\mathcal{D}}^{\circ}(s_{t+h'}, a_{t+h'} | s_t) \parallel P_{\mathcal{D}}(s_{t+h'}, a_{t+h'} | s_t)) \leq \varepsilon_h, \forall h' \in \{1, 2, \dots, h-1\}, \quad (5.4)$$

$$D_{\text{TV}}(P_{\mathcal{D}}^{\circ}(s_{t+h} | s_t) \parallel P_{\mathcal{D}}(s_{t+h} | s_t)) \leq \varepsilon_h. \quad (5.5)$$

$\mathcal{D}$ is **strongly** ε_h -open-loop consistent if additionally for every $a_{t:t+h} \in \text{supp}(P_{\mathcal{D}}(a_{t:t+h} | s_t))$,

$$D_{\text{TV}}(T(s_{t+h'} | s_t, a_{t:t+h'}) \parallel P_{\mathcal{D}}(s_{t+h'} | s_t, a_{t:t+h})) \leq \varepsilon_h, \forall h' \in \{1, 2, \dots, h\}, \quad (5.6)$$

where we use $T(s_{t+h'} | s_t, a_{t:t+h'})$ to denote the distribution of the future state $s_{t+h'}$ after carrying out the action sequence $a_{t:t+h'}$ in the environment open-loop from the current state s_t .

Intuitively, $\mathcal{D}$ is ε_h -open-loop consistent if, when executing the same sequence of actions from it open-loop from s_t , the resulting marginal distribution of the state-action h steps into the future (i.e., s_{t+h}) deviates from the corresponding distribution in the dataset by at most ε_h in total variation distance. The strong version (Equation (5.6)) requires the total variation distance bound to hold for every action sequence in the support, whereas the weak version (Equations (5.4) and (5.5)) only requires the bound to hold in expectation. See Section E.1 for examples of weakly open-loop consistent data.

Value learning bias of action chunking Q-learning

Next, we show that the *weak* open-loop consistency of $\mathcal{D}$ alone is sufficient to show that *behavior* value iteration of an action chunking critic results in a *nominal* value function (i.e., $\hat{V}_{\text{ac}}$) with a bounded bias from the *true* value (i.e., V_{ac}) of the behavior cloning action chunking policy $\tilde{\pi}_{\text{ac}}$:

Theorem 1 (AC Value Bias) Let $\hat{V}_{\text{ac}} : \mathcal{S} \rightarrow [0, 1/(1-\gamma)]$ be a solution of

$$\hat{V}_{\text{ac}}(s_t) = \mathbb{E}_{s_{t+1:t+h+1}, a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t)} \left[R_{t:t+h} + \gamma^h \hat{V}_{\text{ac}}(s_{t+h}) \right], \quad (5.7)$$

with $R_{t:t+h} = \sum_{t'=t}^{t+h} \gamma^{t'-t} r(s_{t'}, a_{t'})$ and V_{ac} is the true value of $\tilde{\pi}_{\text{ac}} : s_t \mapsto P_{\mathcal{D}}(a_{t:t+h} | s_t)$. If $\mathcal{D}$ is weakly ε_h -open-loop consistent, then for all $s_t \in \text{supp}(P_{\mathcal{D}}(s_t))$,

$$\left| V_{\text{ac}}(s_t) - \hat{V}_{\text{ac}}(s_t) \right| \leq \frac{\gamma \varepsilon_h}{(1-\gamma)(1-(1-\varepsilon_h)\gamma^h)} \leq \varepsilon_h H \bar{H}. \quad (5.8)$$

We show that this bound is *tight* for any value of $h > 1, \gamma \in [0, 1)$ and $0 \leq \varepsilon_h \leq \frac{1}{2}$:

Theorem 2 (Worst-case AC Value Bias) For any $h > 1, \gamma \in [0, 1], \varepsilon_h \in [0, 1/2]$, there exists an MDP $\mathcal{M}$ and a weakly ε_h -open-loop consistent $\mathcal{D}$ such that for some $s_t \in \text{supp}(P_{\mathcal{D}}(s_t))$,

$$V_{\text{ac}}(s_t) - \hat{V}_{\text{ac}}(s_t) = \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)}. \quad (5.9)$$

Similarly, there exists an MDP $\mathcal{M}$ and an ε_h -open-loop consistent $\mathcal{D}$ such that for some $s_t \in \text{supp}(P_{\mathcal{D}}(s_t))$,

$$\hat{V}_{\text{ac}}(s_t) - V_{\text{ac}}(s_t) = \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)}. \quad (5.10)$$

The proofs can be found in [Section D.2](#) and [Section D.3](#). A direct consequence of these results is that the *true* value of the optimal action chunking policy is close to that of the optimal closed-loop policy:

Corollary 1 (Optimality Gap for AC Policy) Let $\mathcal{D}^*$ be the data collected by any optimal policy π^* . If $\mathcal{D}^*$ is weakly ε_h -open-loop consistent, then for all $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V_{\text{ac}}^*(s_t) \leq V^*(s_t) - \tilde{V}_{\text{ac}}(s_t) \leq \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)} \leq \varepsilon_h H \bar{H}, \quad (5.11)$$

where V^* is the value of the optimal policy π^* , V_{ac}^* is the *true* value of the optimal action chunking policy, and $\tilde{V}_{\text{ac}}$ is the *true* value of the action chunking policy from cloning the data $\mathcal{D}^*$:

$$\tilde{\pi}_{\text{ac}}^{\mathcal{D}^*}(a_{t:t+h} \mid s_t) : s_t \mapsto P_{\mathcal{D}^*}(\cdot \mid s_t). \quad (5.12)$$

We show that this bound is also *tight*. The proofs are in [Section D.4](#) and [Section D.5](#).

Corollary 2 (Worst-case Optimality Gap for Action Chunking Policy) For any $h > 1, \gamma \in [0, 1], \varepsilon_h \in [0, 1/2]$, there exists an MDP $\mathcal{M}$ whose optimal policy π^* induces a data distribution $\mathcal{D}^*$ that is weakly ε_h -open-loop consistent, such that for some $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V_{\text{ac}}^*(s_t) = \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)}. \quad (5.13)$$

The key observation that enables these results is that $\hat{V}_{\text{ac}}$ obtained from value iteration on $\mathcal{D}^*$ (data collected by an optimal policy) recovers the value of the optimal policy V^* . This

allows us to use Theorem 1 to directly obtain a bound on the optimality gap for action chunking policies.

Next, we analyze the performance of the action chunking policy obtained by Q-learning. In particular, we analyze the Q-function obtained as a solution of the bellman optimality equation under $\text{supp}(\mathcal{D})$:

$$\hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}) = \mathbb{E}_{s_{t+1:t+h+1} \sim P_{\mathcal{D}}(\cdot | s_t, a_{t:t+h})} \left[R_{t:t+h} + \gamma^h \hat{Q}_{\text{ac}}^+(s_{t+h}, \pi_{\text{ac}}^+(s_{t+h})) \right], \quad (5.14)$$

where π_{ac}^+ is defined as follows:

$$\pi_{\text{ac}}^+ : s_t \mapsto \arg \max_{a_{t:t+h} \in \text{supp}(P_{\mathcal{D}}(a_{t:t+h} | s_t))} \hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}). \quad (5.15)$$

With only the weak open-loop consistency condition, the worst-case performance of the action chunking policy may be arbitrarily low, as formalized below (proof in Section D.6).

Proposition 1 (AC Q-Learning under Weak OLC) For any $h > 1, \gamma \in [0, 1), c \in [0, 1/2), \varepsilon_h \in (0, 1/2)$, there exists an MDP $\mathcal{M}$, weakly ε_h -open-loop consistent $\mathcal{D}$ and $\mathcal{D}^*$ with $\text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h})) \supseteq \text{supp}(P_{\mathcal{D}^*}(s_t, a_{t:t+h}))$, such that for some $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V_{\text{ac}}^+(s_t) = V_{\text{ac}}^*(s_t) - V_{\text{ac}}^+(s_t) = \frac{\gamma^c}{1 - \gamma}. \quad (5.16)$$

Intuitively, the chunked critic $Q(s_t, a_{t:t+h})$ has no way of differentiating a low-probability, ‘lucky’ success from a closed-loop, high-probability success. This can cause the learned policy π_{ac}^+ to erroneously prefer very low-value action chunks even when the optimal action chunks are available in the data distribution. With Proposition 1, we conclude that the weak open-loop consistency is *insufficient* for effectively bounding the sub-optimality of action chunking Q-learning. Fortunately, the strong open-loop consistency (Equation (5.6)) is sufficient as quantified by the following bound:

Theorem 3 (AC Q-Learning under Strong OLC) If $\mathcal{D}$ and $\mathcal{D}^*$ are strongly ε_h -open-loop consistent, and $\text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h})) \supseteq \text{supp}(P_{\mathcal{D}^*}(s_t, a_{t:t+h}))$, then $\forall s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V_{\text{ac}}^+(s_t) \leq \frac{\varepsilon_h \gamma}{1 - \gamma} \left[\frac{2}{1 - (1 - 2\varepsilon_h)\gamma^h} + \frac{1}{1 - (1 - \varepsilon_h)\gamma^h} \right] \leq 3\varepsilon_h H \bar{H}, \quad (5.17)$$

where V^* is the value of a closed-loop optimal policy and V_{ac}^+ is the *true* value of π_{ac}^+ .

Theorem 3 (proof in Section D.7) shows that as long as both $\mathcal{D}$ and $\mathcal{D}^*$ satisfy the strongly open-loop consistency condition and $\mathcal{D}$ contains the behavior in $\mathcal{D}^*$, Q-learning with action

chunking is guaranteed to converge to a near-optimal action chunking policy regardless of how sub-optimal the data $\mathcal{D}$ might be. Also, we show this bound is *tight* (proof in Section D.8):

Theorem 4 (Worst-case Analysis of Q-Learning with Action Chunking Policy on Off-policy Data) For any $h > 1, \gamma \in (0, 1), \varepsilon_h \in (0, 1/5), c_1 \in (0, \varepsilon_h/2)$, and $c_2 \in (0, 2\varepsilon_h\gamma)$, there exists an MDP $\mathcal{M}$ and strongly ε_h -open-loop consistent data distributions $\mathcal{D}$ and $\mathcal{D}^*$ with $\text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h})) \supseteq \text{supp}(P_{\mathcal{D}^*}(s_t, a_{t:t+h}))$, such that for some $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V_{\text{ac}}^+(s_t) = \frac{2\varepsilon_h\gamma - c_2}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)} + \frac{\varepsilon_h\gamma}{(1-\gamma)(1-(1-\varepsilon_h-c_1)\gamma^h)}, \quad (5.18)$$

where V^* is the value of an optimal policy and V_{ac}^+ is the *true* value of π_{ac}^+ .

As $c_1, c_2 \rightarrow 0$,

$$V^*(s_t) - V_{\text{ac}}^+(s_t) \rightarrow \frac{\varepsilon_h\gamma}{1-\gamma} \left[\frac{2}{1-(1-2\varepsilon_h)\gamma^h} + \frac{1}{1-(1-\varepsilon_h)\gamma^h} \right]. \quad (5.19)$$

Up to now, none of the bounds that we have shown so far depend on the sub-optimality of the data. Indeed, we can make the data arbitrarily sub-optimal while the action chunking policy learning is still guaranteed to be near optimal. As we will show in the following section, this is in contrast to the n -step return policy where its performance depends on the sub-optimality of the data.

5.3 Comparing to n -step Return Q-learning

We now characterize the condition when action chunking Q-learning should be preferred over the standard n -step return backup. We start by introducing a notion of sub-optimality:

Definition 3 (Sub-optimal Data) $\mathcal{D}$ is δ_n -sub-optimal for a backup horizon length of $n > 1$ if

$$Q^*(s_t, a_t) - \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_t)} [R_{t:t+n} + \gamma^n V^*(s_{t+n})] \geq \delta_n, \forall s_t, a_t \in \text{supp}(P_{\mathcal{D}}(s_t, a_t)). \quad (5.20)$$

Intuitively, δ_n captures how much worse the n -step return policy can be than the optimal policy incurred by the backup bias. Under such a condition, we can show that the action chunking policy is provably better than the n -step return policy as long as δ_n is large.

Proposition 2 (Comparing Q-chunking backup and n -step return backup) Let $\mathcal{D}$ be strongly ε_h -open-loop consistent and δ_n -sub-optimal, and $\text{supp}(P_{\mathcal{D}}(s_t)) \supseteq \text{supp}(P_{\mathcal{D}^*}(s_t))$. Let $\pi_n^+ : s_t \mapsto \arg \max_{a_t} \hat{Q}_n^+(s_t, a_t)$ be the policy learned from $\mathcal{D}$, via n -step return backup:

$$\hat{Q}_n^+(s_t, a_t) = \mathbb{E} \left[R_{t:t+n} + \gamma^n \hat{Q}_n^+(s_{t+n}, \pi_n^+(s_{t+n})) \right]. \quad (5.21)$$

Then, for all $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$ (and with $\bar{H}_n = 1/(1 - \gamma^n)$),

$$\begin{aligned} V_{\text{ac}}^+(s_t) - \hat{V}_n^+(s_t) &\geq \frac{\delta_n}{1 - \gamma^n} - \frac{\varepsilon_h \gamma}{1 - \gamma} \left[\frac{2}{1 - (1 - 2\varepsilon_h)\gamma^h} + \frac{1}{1 - (1 - \varepsilon_h)\gamma^h} \right], \\ &\geq \delta_n \bar{H}_n - 3\varepsilon_h H \bar{H}. \end{aligned} \quad (5.22)$$

The proof of Proposition 2 is available in Section D.10. Notably, for $n = h$, as long as $\mathcal{D}$ is more than $(3\varepsilon_h H)$ -sub-optimal, the value of the action chunking policy is provably better than the value of the n -step return policy. It is worth noting that Proposition 2 uses the *nominal* value of the n -step return, which may be lower than its *actual* value. We refer readers to Section E.3 for examples where the n -step return policy is provably worse than the action chunking policy.

Up to now, we have characterized the conditions under which action chunking policies are better than n -step return policies. However, action chunking policies are still fundamentally limited when subject to poor open-loop consistent data. To tackle this challenge, we explore *closed-loop execution* of an action chunking policy (*i.e.*, carrying out the first action of the full action chunk at every step). While this has been explored in robotic applications [164, 24, 84, 14] to reduce latency and improve smoothness, the theoretical property of closed-loop execution of action chunking policies is not well-understood, especially in the context of Q-learning.

5.4 Closed-loop Execution of Action Chunking Policy

If we reuse the same strongly ε_h -open-loop consistency assumption, we can guarantee that closed-loop execution of the action chunking policy is also near-optimal. The intuition is that in order for action chunking policy to be near-optimal, the first action in the chunk cannot be too sub-optimal:

Proposition 3 (Optimality of Closed-loop Execution of Action Chunking Policy) Let $V^\bullet$ be the value of the one-step policy, $\pi^\bullet$, as a result of the closed-loop execution of the action chunking policy π_{ac}^+ learned from $\mathcal{D}$. That is, for each $s_t \in \text{supp}(P_{\mathcal{D}}(s_t))$,

$$\pi^\bullet(s_t) = a_t^+, \quad \text{where } a_{t:t+h}^+ = \pi_{ac}^+(s_t). \quad (5.23)$$

If $\mathcal{D}$ and $\mathcal{D}^\star$ are strongly ε_h -open-loop consistent, and

$$\text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h})) \supseteq \text{supp}(P_{\mathcal{D}^\star}(s_t, a_{t:t+h})),$$

then $\forall s_t \in \text{supp}(P_{\mathcal{D}^\star}(s_t))$,

$$V^\star(s_t) - V^\bullet(s_t) \leq \frac{\varepsilon_h \gamma}{(1 - \gamma)^2} \left[\frac{2}{1 - (1 - 2\varepsilon_h)\gamma^h} + \frac{1}{1 - (1 - \varepsilon_h)\gamma^h} \right] \leq 3\varepsilon_h H^2 \bar{H}. \quad (5.24)$$

The proof is available in Section D.9. This result demonstrates that closed-loop execution is also near-optimal as long as the action chunking policy is near-optimal, though we might have to pay up to a horizon factor H (i.e., $1/(1 - \gamma)$) in sub-optimality gap in the worst case. Can we do better than this?

In practical applications, the data distributions that we are dealing with often have more structure. For example, it is common to have a dataset consisting of multiple sources where each data source is collected by either a human expert or a scripted policy that exhibits a somewhat predictable behavior (e.g., after a robot arm picks up a cube, it will always move up rather than dropping it right away). We formalize this kind of structures as a notion of optimality variability as follows:

Definition 4 (Optimality Variability) $\mathcal{D}$ exhibits ϑ_h -bounded variability in optimality conditioned on an event X if

$$\max_{\text{supp}(P_{\mathcal{D}}(\cdot|X))} [R_{t:t+h} + \gamma^h V^\star(s_{t+h})] - \min_{\text{supp}(P_{\mathcal{D}}(\cdot|X))} [R_{t:t+h} + \gamma^h V^\star(s_{t+h})] \leq \vartheta_h. \quad (5.25)$$

If we pick X to be the current state and the current action, a bounded optimality variability subject to such conditioning means that as long as we observe the initial action, the optimality of the outcome after h -steps does not vary too much. It turns out that if (1) the data distribution is a mixture of a bunch of data sources where the optimality variability conditioned on the *current actions* is bounded within each data source, and additionally (2) the optimality variability conditioned on the *current action chunks* is bounded globally across mixture, we can form a much stronger bound on the optimality of $\pi^\bullet$. It is worth noting that the second optimality variability condition is *much weaker* than the first one

because it is conditioned on the event where we observe the state s_t and the entire action chunk $a_{t:t+h}$ (rather than only the first action a_t). We now state our theorem as follows:

Theorem 5 (Closed-loop AC Policy under Bounded OV) Let $\mathcal{D}^*$ be the data distribution collected by an optimal policy. Assume $\mathcal{D}$ can be decomposed into a mixture of data distributions $\{\mathcal{D}^*, \mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_M\}$ and for some $\vartheta_h^L, \vartheta_h^G \geq 0$, they satisfy the following two conditions:

1. Locally bounded optimality variability condition: every $\mathcal{D}_i$ (including $\mathcal{D}^*$) exhibits ϑ_h^L -bounded variability in optimality conditioned on s_t, a_t for all $(s_t, a_t) \in \text{supp}(P_{\mathcal{D}_i}(s_t, a_t))$, and

2. Globally bounded optimality variability condition: $\mathcal{D}$ as a whole exhibits ϑ_h^G -variability in optimality conditioned on $s_t, a_{t:t+h}$ for all $(s_t, a_{t:t+h}) \in \text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h}))$.

Then for all $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V^\bullet(s_t) \leq \frac{\vartheta_h^L}{1-\gamma} + \frac{\vartheta_h^G + \gamma^h \min(\vartheta_h^L, \vartheta_h^G)}{(1-\gamma)(1-\gamma^h)} \leq \vartheta_h^L H + 2\vartheta_h^G H \bar{H}. \quad (5.26)$$

The proof of Theorem 5 (available in Section D.11) is made possible by observing that $V^*(s_t) - \hat{V}_{ac}^+(s_t)$ and $\hat{V}_{ac}^+(s_t) - Q^*(s_t, a_t^+)$ are bounded by $\vartheta_h^G/(1-\gamma^h)$ and $\vartheta_h^L/(1-\gamma^h)$ respectively. Combining these two bounds naïvely already allows us to derive a relatively loose bound $V^*(s_t) - Q^*(s_t, a_t^+) \leq (\vartheta_h^L + \vartheta_h^G)/(1-\gamma^h)$ which leads to $V^*(s_t) - V^\bullet(s_t) \leq (\vartheta_h^L + \vartheta_h^G)/(1-\gamma^h)/(1-\gamma)$. To obtain the tight bound in Theorem 5, we leverage a key insight that the amount of overestimation in V_{ac}^+ can *never exceed* $\vartheta_h^L + \frac{\vartheta_h^G}{1-\gamma^h}$ as otherwise the nominal value of the action chunking policy h -step into the future, $\hat{V}_{ac}^+(s_{t+h})$, would have an optimality gap higher than $\vartheta_h^G/(1-\gamma^h)$, which is impossible under the global optimality variability condition. Forming this tight bound is important because it effectively shaves off a factor of $\bar{H} = 1/(1-\gamma^h)$ from the ϑ_h^L term (the stronger local condition) and only bumps up a factor of ≈ 2 to the ϑ_h^G term (the weaker global condition).

It is worth noting that although the global optimality variability condition looks similar to the strong open-loop consistency condition, they have completely different properties. For instance, a nearly strong open-loop consistent data distribution $\mathcal{D}$ can have unbounded global optimality variability and a data distribution that exhibits zero optimality variability can also have large open-loop inconsistency. The implication of this is that while the closed-loop execution of an action chunking policy can be near-optimal, the same action chunking policy executed in chunks can be sub-optimal. We formalize this intuition as the worst-case result below:

Theorem 6 (Worst-case Closed-loop AC Policy under BOV) For any $h > 1, \gamma \in (0, 1), \vartheta_h^G, \vartheta_h^L \in \left(0, \frac{\gamma - \gamma^h}{4(1-\gamma)}\right], c \in \left[0, \frac{\gamma - \gamma^h}{4(1-\gamma)}\right), \sigma \in \left(0, \frac{\min(\vartheta_h^G, \vartheta_h^L)}{1-\gamma}\right)$, there exists $\mathcal{M}$ and $\mathcal{D}$ satisfying the assumptions in Theorem 5 such that there exists $s_t \in \text{supp}(P_{\mathcal{D}^\star}(s_t))$, where

$$V^\star(s_t) - V^\bullet(s_t) = \frac{\vartheta_h^L}{1-\gamma} + \frac{\vartheta_h^G + \gamma^h \min(\vartheta_h^L, \vartheta_h^G)}{(1-\gamma)(1-\gamma^h)} - \sigma, V^\star(s_t) - V_{\text{ac}}^+(s_t) \geq \frac{c}{1-\gamma}. \quad (5.27)$$

The examples in the proof of Theorem 6 (available in Section D.12) serve a dual purpose—they not only show that our upper bound in Theorem 5 is *tight* (since we can make $\sigma \rightarrow 0$), but also show that the sub-optimality of the action chunking policy can be made arbitrarily large. Furthermore, *both* the local optimality (ϑ_h^L) condition and the global optimality (ϑ_h^G) are *necessary* to guarantee $\pi^\bullet$ being near-optimal. When any of them is large, Theorem 6 implies that there exists an MDP where $\pi^\bullet$ is sub-optimal. As a side note, we can guarantee $\pi^\bullet$ to be near-optimal with an alternative ‘stochastic shortcut’ assumption (a weaker form of the global optimality variability assumption) and a slightly stronger data mixing assumption. We refer the readers to Section E.5 for the formal results under this alternative assumption. Overall, combining Theorem 5 and Proposition 3 shows that, compared to executing the action chunking policy in open-loop chunks, closed-loop execution attains a similar bound under the strongly ε_h -open-loop consistent assumption, and excels under the bounded optimality variability assumptions.

5.5 Concluding Remark

Up to now, we have provided a theoretical foundation for action chunking Q-learning and demonstrate how to effectively extract policies from chunked critics. In particular, we have formally analyzed action chunking Q-learning, identifying the TD backup bias that arises from open-loop inconsistency and characterizing the conditions under which action chunking Q-learning is preferred over n -step return learning and the conditions under which closed-loop execution of the action chunking policy is near-optimal.

Conceptually, closed-loop execution of the learned action chunking policy decouples the open-loop execution horizon (policy chunk length) from the value-learning horizon (critic chunk length). Such decoupling inherits the strength of action chunking TD and 1-step TD: (1) the value learning speedup of action chunking Q-learning, and (2) the reactivity of a standard, single-step policy. Furthermore, executing the first action (or more generally a partial chunk) of the original action chunk also brings practical benefits: it removes the need to explicitly train a policy to predict the full action chunk all at once, which can be especially challenging when the chunk size grows big. In Chapter 6, we develop a practical algorithm to realize such potential, which allows us to scale up action chunking Q-learning to tasks with much longer horizons.

Chapter 6

Scaling up Action Chunking Q-learning to Long-horizon Tasks

In the previous chapter, we have formally characterized that action chunking Q-learning may suffer from open-loop bias if the data distribution exhibits poor open-loop consistency, and closed-loop execution of a learned action chunking policy under such distribution can mitigate such bias. However, it is still unclear how one may design a method to make good use of this result. One naïve method is to simply learn a policy for full action chunks but only execute the first (or first couple of actions) in the predicted chunks, but this requires the policy to model increasingly more complex action chunk distributions as the action chunk size grows. This growing action chunk size may adversely impact performance, limiting the scalability of action chunking RL. Is there a practical approach that circumvents this limitation?

Our key insight is that we can avoid training the policy to predict the full action chunks and instead to only predict shorter, partial action chunks against the chunked critic. To achieve this, we use a ‘distilled’ chunked critic with a chunk size that matches the policy: it optimistically regresses to the original chunked critic to approximate the maximum value that the partial action chunk can achieve after being extended into a full action chunk. Conceptually, while the action optimization is still done for the longer, complete action chunks, the policy network is only trained to output the partial action chunk of an optimized complete action chunk. This way, the policy only needs to predict a much shorter action chunk (*e.g.*, in the extreme case, only one action), which often admits a much simpler distribution, while enjoying the value learning benefits from the use of chunked critics.

In this chapter, we instantiate this insight as a practical offline RL algorithm (Section 6.1) for action chunking Q-learning. We stress test our algorithm on some of the most challenging, long-horizon tasks in OGBench [111], and show that our algorithm outperforms the previous state-of-the-art.

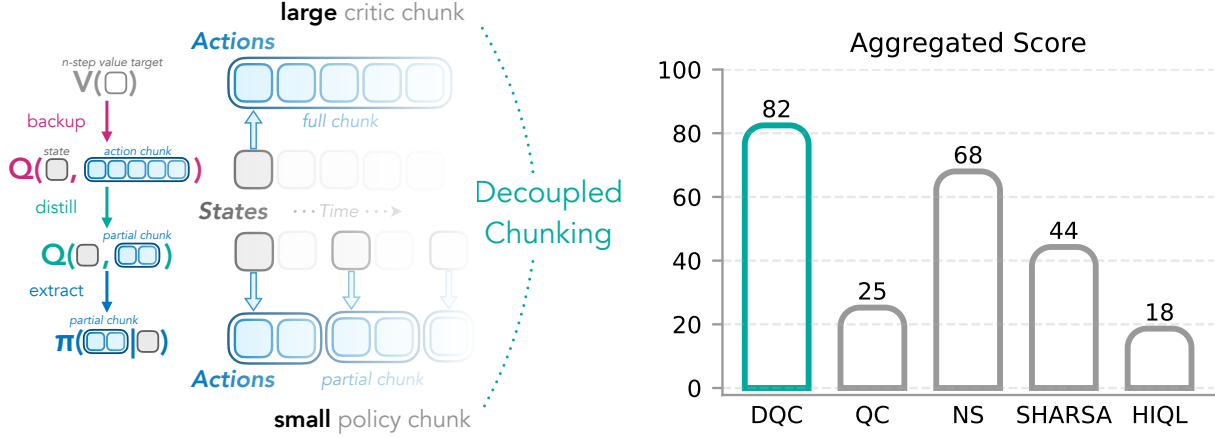


Figure 6.1: **Decoupled Q-chunking (DQC)**. *Left*: The key idea of our method is to ‘decouple’ the action chunk size of the critic Q from that of the policy π . A large critic chunk size allows for efficient value learning while a small policy chunk size makes policy learning more tractable and allows for better policy reactivity. *Right*: Our method outperforms all baselines on the six hardest environments on OGBench, an offline goal-conditioned RL benchmark with challenging long-horizon tasks.

6.1 Decoupled Q-chunking

In this section, we propose a new algorithm that enjoys the benefits of value backup speedup of critic chunking while avoiding the difficulty of learning an open-loop action chunking policy with a large chunk size. As the name of the algorithm suggests, our core idea is to decouple the chunk size of the critic from that of the policy where the policy only predicts a partial action chunk. We first describe the algorithm conceptually, and then provide detail on how the algorithm can be effectively implemented in practice for offline RL.

The Algorithm

To start with, we train a policy $\pi(a_{t:t+h_a} \mid s_t)$ to output an action chunk (with a size of $h_a \ll h$) using the following objective:

$$L(\pi) := -\mathbb{E}_{a_{t:t+h_a} \sim \pi(\cdot \mid s_t)} [Q_\phi(s_t, [a_{t:t+h_a}, a_{t+h_a:t+h}^*])], \quad (6.1)$$

where $[a_{t:t+h_a}, a_{t+h_a:t+h}^*]$ represents the concatenation of two partial action chunks (size h_a and size $h - h_a$) into a full action chunk $a_{t:t+h}$ of size h , and $a_{t+h_a:t+h}^*$ is the best ‘second-half’ of the action chunk that maximizes the critic value under Q_ϕ :

$$a_{t+h_a:t+h}^* := \arg \max_{a_{t+h_a:t+h}} Q_\phi(s_t, [a_{t:t+h_a}, a_{t+h_a:t+h}]). \quad (6.2)$$

Essentially, we want our policy to predict the partial action chunk (of size h_a) within an optimal action chunk of size h , rather than the entire optimal action chunk. This lowers the policy expressivity requirement and hence the learning challenges associated with it.

However, directly optimizing the objective in Equation (6.1) does not lead to a new algorithm because taking the maximization over $a_{t+h_a:t+h}$ seemingly requires us to learn a policy of the original chunk size anyway. To address this issue, we learn a separate partial critic Q_ψ^P , which only takes in the partial action chunk (of size h_a) as input, to approximate the maximum value this partial action chunk can achieve when it is extended to the full action chunk (of size h):

$$Q_\psi^P(s_t, a_{t:t+h_a}) \approx Q_\phi(s_t, [a_{t:t+h_a}, a_{t+h_a:t+h}^*]). \quad (6.3)$$

To train Q_ψ^P , we can use an *implicit maximization* loss function (as described in Equation (2.4)):

$$L(\psi) := f_{\text{imp}}^{\kappa_d}(\bar{Q}_\phi(s_t, a_{t:t+h}) - Q_\psi^P(s_t, a_{t:t+h_a})), \quad (6.4)$$

where $s_t, a_{t:t+h}$ are sampled from the offline dataset D . As a result, the partial critic, Q_ψ^P , is distilled from the original critic via an optimistic regression, where its optimum $Q_\psi^*(s_t, a_{t:t+h_a})$ approximates $Q_\phi(s_t, [a_{t:t+h_a}, a_{t+h_a:t+h}^*])$ in Equation (6.1), conveniently removing the need for training a policy to predict the whole optimal action chunk entirely. This allows us to simplify the policy objective as

$$L(\pi) := -\mathbb{E}_{a_{t:t+h_a} \sim \pi(\cdot | s_t)} [Q_\psi^P(s_t, a_{t:t+h_a})]. \quad (6.5)$$

In summary, DQC trains a policy to predict a partial chunk, $a_{t:t+h_a}$ (of size h_a), by hill climbing the value of a partial critic $Q_\psi^P(s_t, a_{t:t+h_a})$ that is distilled from the original chunked critic $Q_\phi(s_t, a_{t:t+h})$ via an implicit maximization loss. This allows our policy to fully leverage the chunked critic Q_ϕ (and thus the value speedup benefits associated with Q-chunking) without the need to predict the full action chunk (of size h), mitigating the learning challenge of an action chunking policy.

Practical Considerations for Offline RL

Finally, we describe several implementation details that we find to work well in the offline RL setting, which our experiments focus on. Our implementation draws inspiration from a prior method, IDQL [57].

We first train a behavior cloning flow policy π_β using a standard flow-matching objective [87] on the offline dataset D . Then, we approximate the policy optimization objective in Equation (6.5) using best-of- N sampling without explicitly modeling π :

$$a_{t:t+h_a}^* \leftarrow \arg \max_{\{a_{t:t+h_a}^i\}_{i=1}^N} Q_\psi^P(s_t, a_{t:t+h_a}), \quad \text{where } a_{t:t+h_a}^1, \dots, a_{t:t+h_a}^N \sim \pi_\beta(\cdot | s_t), \quad (6.6)$$

Algorithm 3 Decoupled Q-chunking (DQC).

Given: $D, Q_\phi(s_t, a_{t:t+h}), Q_\psi^P(s_t, a_{t:t+h_a}), V_\xi(s_t), \pi_\beta(a_{t:t+h_a} \mid s_t)$

1. Agent Update:

$(s_{t:t+h+1}, a_{t:t+h}, r_{t:t+h}) \sim D.$ $\triangleright$ sample trajectory chunk from the offline dataset

Optimize Q_ϕ with $L(\phi) = \left(Q_\phi(s_t, a_{t:t+h}) - \sum_{k=0}^{h-1} \gamma^k r_{t+k} - \gamma^h \bar{V}_\xi(s_{t+h}) \right)^2$.

Optimize Q_ψ^P with $L(\psi) = f_{\text{expectile}}^{\kappa_d}(\bar{Q}_\phi(s_t, a_{t:t+h}) - Q_\psi^P(s_t, a_{t:t+h_a}))$.

Optimize V_ξ with $L(\xi) = f_{\text{quantile}}^{\kappa_b}(\bar{Q}_\psi^P(s_t, a_{t:t+h_a}) - V_\xi(s_t))$,

2. Policy Extraction:

$a_{t:t+h_a}^1, a_{t:t+h_a}^2, \dots, a_{t:t+h_a}^N \sim \pi_\beta(\cdot \mid s_t)$ $\triangleright$ sample N actions from behavior policy

$a_{t:t+h_a}^* \leftarrow \arg \max_{\{a_{t:t+h_a}^i\}_{i=1}^N} Q_\psi^P(s_t, a_{t:t+h_a}^i)$ $\triangleright$ take the action with the highest Q-value

and $a_{t:t+h_a}^*$ is output of the policy that we extract from Q_ψ^P for state s_t . Essentially, this sampling procedure is a test-time approximation of the objective in Equation (6.5), where it outputs an action (chunk) that maximizes Q_ψ^P , subject to the behavior prior, as modeled by π_β .

For TD learning of Q_ϕ , directly computing the TD backup target from either $Q_{\bar{\phi}}$ or Q_ψ^P is computationally expensive, as either requires samples from the current policy, which is approximated via the best-of- N sampling procedure as described above. Instead, we use the implicit value backup [75] (i.e., as described in Equation (2.4)) to approximate the target:

$$L(\xi) = f_{\text{quantile}}^{\kappa_b}(\bar{Q}_\psi^P(s_t, a_{t:t+h_a}) - V_\xi(s_t)), \quad (6.7)$$

where we pick the quantile regression loss as the implicit maximization loss function. This is because the Q-value obtained from best-of- N sampling can be seen as the largest order statistic of a random batch (of size N) of the behavior Q-values. Such a statistic estimates the behavior Q-value distribution's $\frac{N-1}{N}$ -quantile, which is the same as $V_\xi(s)$ at the optimum of $L(\xi)$ if we set $\kappa_b = \frac{N-1}{N}$. In practice, we use a smaller κ_b for numerical stability (see Table C.6).

Finally, we pick the expectile regression loss for training the distilled partial critic Q_ψ^P because prior work has found it to work the best among all implicit maximization loss functions [57]. A summary of the algorithm is available in Algorithm 3.

6.2 Experiments

We conduct experiments to evaluate the benefits of decoupling the policy chunk size and the critic chunk size on OGBench [111]—a challenging long-horizon, goal-conditioned offline RL benchmark consisting of a diverse set of environments (from manipulation to locomotion).

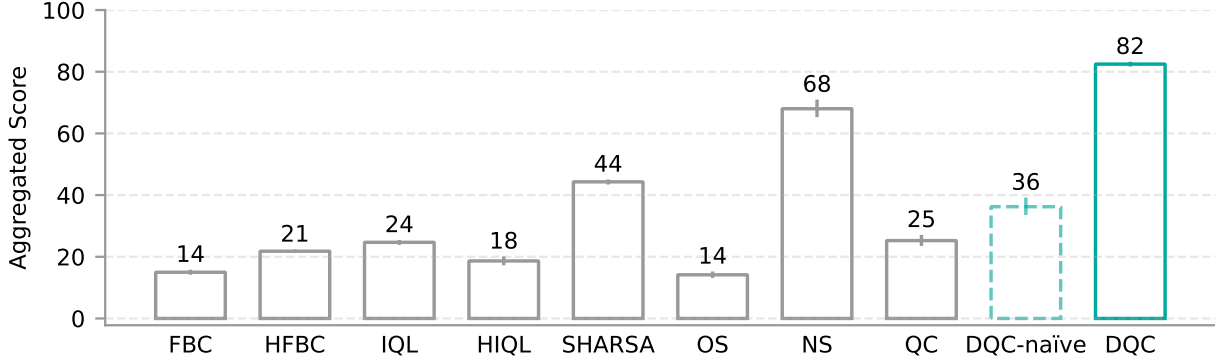


Figure 6.2: **Aggregated score across six hardest OGBench environments (10 seeds):** cube- $\{\text{triple/quadruple/octuple}\}$, humanoidmaze-giant, and puzzle- $\{4\times 5, 4\times 6\}$.

In particular, we use the more difficult environments introduced by [110], where multi-step return backups are crucial. These environments require highly complex, long-horizon reasoning. For example, the **puzzle** tasks require stitching up to 24 atomic motions to solve a combinatorial puzzle with a robot arm, and the **humanoidmaze** tasks require controlling a high-dimensional humanoid robot over 3000 environment steps to navigate a maze. These environments serve as an ideal testbed for our algorithm, which improves upon n -step returns and Q-chunking. We now describe our main comparisons. To start with, we consider several direct ablation baselines where the same algorithm backbone is being used (*i.e.*, implicit value backup and best-of- N sampling):

QC (Q-chunking [81]) uses a single critic that has the same chunk length as that of the policy (*i.e.*, $h = h_a$). This baseline tests whether having *decoupled* chunk sizes is important.

DQC-naïve is a naïve attempt at decoupling the critic chunk size from the policy chunk size, where it takes the QC policy to predict full action chunks of size h but only execute the first h_a actions.

NS: n -step return TD backup. This baseline uses a single one-step critic (*i.e.*, $Q(s_t, a_t)$). Compared to DQC with $h = n$ and $h_a = 1$, this baseline tests whether using a chunked critic is important.

OS: Standard 1-step TD backup. This is the same as NS but with $n = 1$.

Beyond the ablation baselines, we also consider the following strong goal-conditioned baselines from prior work:

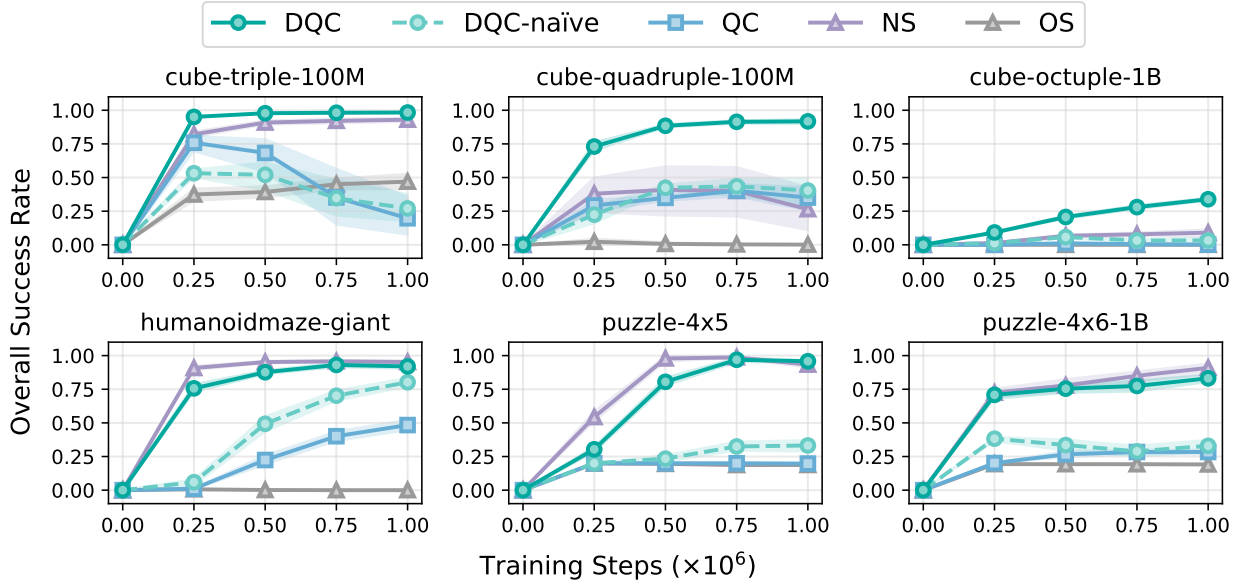


Figure 6.3: **Offline goal-conditioned RL results (10 seeds).** Our method (*DQC*) uses *decoupled* critic and policy chunk sizes. *QC*: Q-chunking [81]; *NS*: n -step return backup; *OS*: 1-step TD-backup; *DQC-naïve*: same as QC but executes a partial action chunk.

FBC/HFBC: Goal-conditioned and hierarchical goal-conditioned flow behavior cloning baselines considered in [110].

IQL/HIQL [75, 109]: These are strong goal-conditioned RL methods that train a goal-conditioned value function with implicit value backups and extract a flat (IQL) or hierarchical (HIQL) policy from the value function.

SHARSA [110]: The previous state-of-the-art method on the long-horizon environments that we evaluate on. The method uses a combination of n -step return and bi-level hierarchical policies.

In our ablation study, we also consider an additional baseline, **QC-NS**, that uses the idea of decoupled policy chunking and critic chunking ($h_a < h$), but without using a distilled critic. This baseline simply uses n -step return targets to directly train a critic with a chunk size of h_a without implicit maximization (Equation (6.4)). The performance of this baseline helps determine how important it is to learn a separate distilled critic for partial action chunks with implicit maximization. We run 10 seeds for all methods, and report the means and the 95% confidence intervals.

Task	FBC	HFBC	IQL	HIQL	SHARSA	OS	NS	QC	DQC-naïve	DQC
cube-triple-100M	54 _[51,56]	56 _[53,59]	66 _[63,67]	35 _[31,39]	83 _[81,85]	47 _[41,53]	93 _[91,94]	20 _[7,36]	27 _[18,38]	98 _[98,99]
cube-quadruple-100M	34 _[32,37]	37 _[34,40]	53 _[52,55]	24 _[21,28]	64 _[62,68]	0 _[0,0]	27 _[11,43]	35 _[26,43]	40 _[29,49]	92 _[90,93]
cube-octuple-1B	0 _[0,0]	28 _[26,29]	0 _[0,0]	20 _[17,23]	34 _[31,36]	0 _[0,0]	9 _[6,12]	0 _[0,0]	3 _[1,5]	34 _[33,35]
humanoidmaze-giant	1 _[1,2]	6 _[4,8]	3 _[2,5]	24 _[22,26]	19 _[16,23]	0 _[0,0]	95 _[94,97]	48 _[45,52]	80 _[77,83]	92 _[90,94]
puzzle-4x5	0 _[0,0]	0 _[0,0]	20 _[19,20]	0 _[0,0]	1 _[1,2]	19 _[18,19]	93 _[91,95]	20 _[20,20]	33 _[29,37]	96 _[95,97]
puzzle-4x6-1B	1 _[0,1]	4 _[3,5]	6 _[3,9]	9 _[5,13]	64 _[60,68]	19 _[19,20]	91 _[86,94]	28 _[27,30]	33 _[28,38]	83 _[80,86]

Table 6.1: **Comparisons with prior methods (10 seeds).** Our method outperforms SHARSA [110] (the previous state-of-the-art method on this benchmark) on most environments.

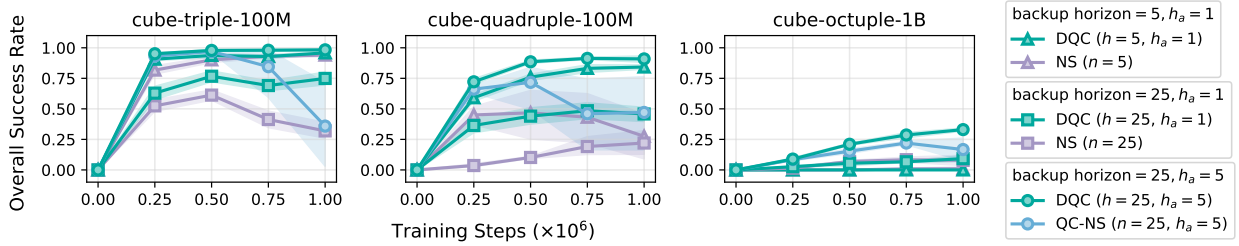


Figure 6.4: **Distilled critic ablations (10 seeds).** Each group in the legend contains DQC and its non-distilled counterpart with the same configuration. Our method (DQC) performs on par with or better than the non-distilled counterpart across all configurations.

Results

(Q1) Does DQC improve upon n -step return, Q-chunking? Figure 6.3 compares DQC (ours) to both n -step and QC across six challenging long-horizon GCRL environments, with our method performing on par with or better across the board. Table 6.1 shows DQC also consistently outperforms the previous state-of-the-art method on this benchmark, SHARSA [110], on all environments. For each environment, we tune DQC (ours), QC, NS, and OS (see the tuning range in Table C.7) and pick the best configuration (Table C.5) for hyperparameters used in Figure 6.3 and Table 6.1. For all baselines from prior work (SHARSA, HIQL, IQL, HFBC, FBC), we directly use their tuned hyperparameters and run with the same batch size (*i.e.*, 4096) as used in our method and other baselines. See the complete result table for all combinations of h, n, h_a in Section C.1.

(Q2) Is training a separate distilled critic Q_ψ^P necessary? In Figure 6.4, we compare DQC to DQC without using the distilled critic across three different (h, h_a) configurations: $(h = 25, h_a = 5)$, $(h = 25, h_a = 1)$, and $(h = 5, h_a = 1)$. For configurations with $h_a = 1$, the baseline without using the distilled critic is the same as the n -step return baseline (with $n = h$) and for the configuration with $h_a = 5$, it is the same as combining Q-chunking and n -step return (QC-NS). Across three configurations, DQC performs on par or better than its non-distilled counterpart. This highlights that a separate distilled critic for the partial

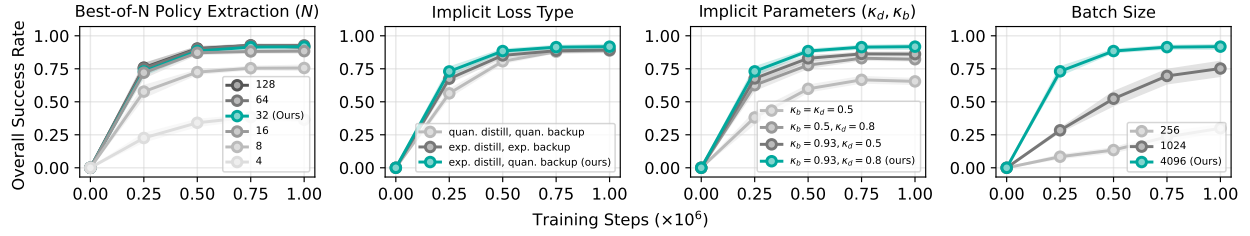


Figure 6.5: **Hyperparameter sensitivity analysis on cube-quadruple (10 seeds).** *Best-of-N*: the number of action samples drawn from $\pi_\beta(\cdot | s)$ during policy evaluation; *Implicit loss type*: the implicit maximization loss function used for distillation and value backup; *Batch size*: the number of examples used in each gradient step.

action chunk is necessary for the effectiveness of DQC.

(Q3) How sensitive is DQC to its hyperparameters? Figure 6.5 shows that our method is not sensitive to the implicit backup method (quantile or expectile), and somewhat sensitive to the implicit parameters κ_b, κ_d . In particular, DQC is still reasonably effective as long as some form of optimism is employed (*i.e.*, either $\kappa_b \neq 0.5$ or $\kappa_d \neq 0.5$). Using no optimism ($\kappa_b = \kappa_d = 0.5$) results in a big performance drop. The other important hyperparameters are N in the best-of- N policy extraction and the batch size. Having a large enough batch size (*i.e.*, 4096) and N (*e.g.*, $N = 32$) is crucial for good performance, although increasing N further (*e.g.*, $N = 128$) does not lead to better performance.

6.3 Concluding Remark

We develop a new technique that enables effective policy extraction from chunked critics with long action chunks, scaling up action chunking Q-learning to much harder environments. Together, these contributions advance the goal of tackling bootstrapping bias in TD-learning. Several challenges remain, indicating promising avenues for future research. For example, our method relies on a fixed policy action chunk size h_a and critic action chunk size h across all states, even though the optimal action chunk size may vary by state. Developing practical methods that can support flexible, state-dependent chunk sizes would be a natural next step.

Chapter 7

Final Remarks

In this dissertation, we explored algorithms for making action chunking reinforcement learning practical in the presence of prior data. In particular, we presented two algorithms, FQL and QAM, that enable efficient optimization of action chunking policies in both offline and online settings. Our experiments demonstrated the effectiveness of these approaches in simulated robotic environments, suggesting that they are promising for real-world robotic policy learning. More broadly, we also examined a fundamental limitation of action chunking Q-learning, namely open-loop inconsistency. We established worst-case performance guarantees for action chunking policies under both weak and strong open-loop consistency, as well as the worst-case performance of closed-loop execution of a learned action chunking policy under the bounded optimality variability condition. These theoretical insights, in turn, motivated an effective practical algorithm, DQC, which side-steps the open-loop inconsistency issue and enables action chunking RL to scale to horizon lengths far beyond what had previously been shown to work effectively. All of these provide a foundation for scalable reinforcement learning in real-world settings.

Looking forward, the complexity of real-world RL applications will likely require policy classes beyond action chunking policies, given their inherent limitations (*e.g.*, non-reactivity, large action spaces). We hope the insights and practical designs presented in this dissertation will help guide future algorithmic advances in RL for more general non-Markovian policies. This is especially promising because the practical algorithms introduced here are built around the very general flow-matching generative models, making them readily extensible to other policy classes that can also be parameterized in this way.

Bibliography

- [1] Suzan Ece Ada, Erhan Oztop, and Emre Ugur. “Diffusion Policies for Out-of-Distribution Generalization in Offline Reinforcement Learning”. In: *IEEE Robotics and Automation Letters (RA-L)* 9 (2024), pp. 3116–3123.
- [2] Anurag Ajay et al. “OPAL: Offline Primitive Discovery for Accelerating Offline Reinforcement Learning”. In: *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=V69LGwJ0lIN>.
- [3] Lars Ankile et al. “From imitation to refinement-residual RL for precise assembly”. In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2025, pp. 01–08.
- [4] Lars Ankile et al. “Residual off-policy rl for finetuning behavior cloning policies”. In: *arXiv preprint arXiv:2509.19301* (2025).
- [5] Kamyar Azizzadenesheli, Alessandro Lazaric, and Animashree Anandkumar. “Reinforcement learning of pomdps using spectral methods”. In: *Conference on Learning Theory*. PMLR. 2016, pp. 193–256.
- [6] Pierre-Luc Bacon, Jean Harb, and Doina Precup. “The option-critic architecture”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 31. 1. 2017.
- [7] Akhil Bagaria and George Konidaris. “Option discovery using deep skill chaining”. In: *International Conference on Learning Representations*. 2019.
- [8] Akhil Bagaria et al. “Effectively learning initiation sets in hierarchical reinforcement learning”. In: *Advances in Neural Information Processing Systems* 36 (2024).
- [9] Philip J Ball et al. “Efficient online reinforcement learning with offline data”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 1577–1594.
- [10] Christian Bayer et al. “Continuous time stochastic optimal control under discrete time partial observations”. In: *arXiv preprint arXiv:2407.18018* (2024).
- [11] Andrew Bennett and Nathan Kallus. “Proximal reinforcement learning: Efficient off-policy evaluation in partially observed markov decision processes”. In: *Operations Research* 72.3 (2024), pp. 1071–1086.

- [12] Andrew Bennett et al. “Off-policy evaluation in infinite-horizon reinforcement learning with latent confounders”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 1999–2007.
- [13] Homanga Bharadhwaj et al. “Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 4788–4795.
- [14] Kevin Black, Manuel Y Galliker, and Sergey Levine. “Real-Time Execution of Action Chunking Flow Policies”. In: *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. 2025. URL: <https://openreview.net/forum?id=UkR2z05uww>.
- [15] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: <http://github.com/google/jax>.
- [16] Arwen Bradley and Preetum Nakkiran. “Classifier-free guidance is a predictor-corrector”. In: *arXiv preprint arXiv:2408.09000* (2024).
- [17] Boyuan Chen et al. “Self-supervised reinforcement learning that transfers using random features”. In: *Advances in Neural Information Processing Systems* 36 (2024).
- [18] Huayu Chen et al. “Score regularized policy optimization through diffusion behavior”. In: *International Conference on Learning Representations (ICLR)*. 2024.
- [19] Tianyi Chen et al. “One-Step Flow Policy Mirror Descent”. In: *arXiv preprint arXiv:2507.23675* (2025).
- [20] Tianyu Chen, Zhendong Wang, and Mingyuan Zhou. “Diffusion Policies creating a Trust Region for Offline Reinforcement Learning”. In: *Neural Information Processing Systems (NeurIPS)*. 2024.
- [21] Tianyu Chen, Zhendong Wang, and Mingyuan Zhou. “Diffusion policies creating a trust region for offline reinforcement learning”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 50098–50125.
- [22] Xinyue Chen et al. “Randomized Ensembled Double Q-Learning: Learning Fast Without a Model”. In: *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=AY8zfZm0tDd>.
- [23] Nuttapong Chentanez, Andrew Barto, and Satinder Singh. “Intrinsically motivated reinforcement learning”. In: *Advances in neural information processing systems* 17 (2004).
- [24] Cheng Chi et al. “Diffusion policy: Visuomotor policy learning via action diffusion”. In: *The International Journal of Robotics Research* (2023), p. 02783649241273668.
- [25] Imre Csiszár. “On information-type measure of difference of probability distributions and indirect observations”. In: *Studia Sci. Math. Hungar.* 2 (1967), pp. 299–318.

- [26] Pierluca D’Oro et al. “Sample-Efficient Reinforcement Learning by Breaking the Replay Ratio Barrier”. In: *The Eleventh International Conference on Learning Representations*. 2023. URL: <https://openreview.net/forum?id=OpC-9aBBVJe>.
- [27] Christian Daniel et al. “Hierarchical relative entropy policy search”. In: *Journal of Machine Learning Research* 17.93 (2016), pp. 1–50.
- [28] Peter Dayan and Geoffrey E Hinton. “Feudal reinforcement learning”. In: *Advances in neural information processing systems* 5 (1992).
- [29] Prafulla Dhariwal and Alexander Nichol. “Diffusion models beat gans on image synthesis”. In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.
- [30] Thomas G Dietterich. “Hierarchical reinforcement learning with the MAXQ value function decomposition”. In: *Journal of artificial intelligence research* 13 (2000), pp. 227–303.
- [31] Shutong Ding et al. “Diffusion-based Reinforcement Learning via Q-weighted Variational Policy Optimization”. In: *Neural Information Processing Systems (NeurIPS)*. 2024.
- [32] Shutong Ding et al. “Diffusion-based reinforcement learning via Q-weighted variational policy optimization”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 53945–53968.
- [33] Zihan Ding and Chi Jin. “Consistency models as a rich and efficient policy class for reinforcement learning”. In: *arXiv preprint arXiv:2309.16984* (2023).
- [34] Zihan Ding and Chi Jin. “Consistency models as a rich and efficient policy class for reinforcement learning”. In: *International Conference on Learning Representations (ICLR)*. 2024.
- [35] Carles Domingo-Enrich et al. “Adjoint Matching: Fine-tuning Flow and Diffusion Generative Models with Memoryless Stochastic Optimal Control”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=xQBRrtQM8u>.
- [36] Carles Domingo-Enrich et al. “Stochastic Optimal Control Matching”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., 2024, pp. 112459–112504. DOI: 10.52202/079017-3573. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/cc32ec39a5073f61d38c338d963df30Paper-Conference.pdf.
- [37] Perry Dong et al. “EXPO: Stable Reinforcement Learning with Expressive Policies”. In: *arXiv preprint arXiv:2507.07986* (2025).
- [38] Yilun Du et al. “Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc”. In: *International conference on machine learning*. PMLR. 2023, pp. 8489–8510.

- [39] Paul Dupuis and Hui Wang. “Optimal stopping with random intervention times”. In: *Advances in Applied probability* 34.1 (2002), pp. 141–157.
- [40] Ishan P Durugkar et al. “Deep reinforcement learning with macro-actions”. In: *arXiv preprint arXiv:1606.04615* (2016).
- [41] Nicolas Espinosa-Dice et al. “Scaling Offline RL via Efficient and Expressive Shortcut Models”. In: *arXiv preprint arXiv:2505.22866* (2025).
- [42] Linjiajie Fang et al. “Diffusion Actor-Critic: Formulating Constrained Policy Iteration as Diffusion Noise Regression for Offline Reinforcement Learning”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=ldVKA009Km>.
- [43] Linjiajie Fang et al. “Diffusion Actor-Critic: Formulating Constrained Policy Iteration as Diffusion Noise Regression for Offline Reinforcement Learning”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=ldVKA009Km>.
- [44] William Fedus et al. “Revisiting fundamentals of experience replay”. In: *International conference on machine learning*. PMLR. 2020, pp. 3061–3071.
- [45] Roy Fox et al. “Multi-level discovery of deep options”. In: *arXiv preprint arXiv:1703.08294* (2017).
- [46] Kevin Frans et al. “Diffusion Guidance Is a Controllable Policy Improvement Operator”. In: *arXiv preprint arXiv:2505.23458* (2025).
- [47] Kevin Frans et al. “Unsupervised Zero-Shot Reinforcement Learning via Functional Reward Encodings”. In: *Proceedings of the 41st International Conference on Machine Learning*. Ed. by Ruslan Salakhutdinov et al. Vol. 235. Proceedings of Machine Learning Research. PMLR, 21–27 Jul 2024, pp. 13927–13942. URL: <https://proceedings.mlr.press/v235/frans24a.html>.
- [48] Justin Fu et al. “D4RL: Datasets for deep data-driven reinforcement learning”. In: *arXiv preprint arXiv:2004.07219* (2020).
- [49] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. “Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation”. In: *arXiv preprint arXiv:2401.02117* (2024).
- [50] Zuyue Fu et al. “Offline reinforcement learning with instrumental variables in confounded markov decision processes”. In: *arXiv preprint arXiv:2209.08666* (2022).
- [51] Scott Fujimoto, Herke Hoof, and David Meger. “Addressing function approximation error in actor-critic methods”. In: *International conference on machine learning*. PMLR. 2018, pp. 1587–1596.
- [52] Jonas Gehring et al. “Hierarchical skills for efficient exploration”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 11553–11564.

- [53] Abraham George and Amir Barati Farimani. “One act play: Single demonstration behavior cloning with action chunking transformers”. In: *arXiv preprint arXiv:2309.10175* (2023).
- [54] Kamyar Ghasemipour, Shixiang Shane Gu, and Ofir Nachum. “Why so pessimistic? estimating uncertainties for offline RL through ensembles, and why their independence matters”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 18267–18281.
- [55] Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. “EMaQ: Expected-max Q-learning operator for simple yet effective offline and online rl”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 3682–3691.
- [56] Tuomas Haarnoja et al. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.
- [57] Philippe Hansen-Estruch et al. “IDQL: Implicit Q-learning as an actor-critic method with diffusion policies”. In: *arXiv preprint arXiv:2304.10573* (2023).
- [58] Aaron Havens et al. “Adjoint sampling: Highly scalable diffusion samplers via adjoint matching”. In: *arXiv preprint arXiv:2504.11713* (2025).
- [59] Longxiang He et al. “DiffCPS: Diffusion model based constrained policy search for offline reinforcement learning”. In: *ArXiv abs/2310.05333* (2023).
- [60] Jacob Hilton. *KL divergence of max-of-n*. 2023. URL: https://www.jacobh.co.uk/bon_kl.pdf.
- [61] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.
- [62] Jonathan Ho and Tim Salimans. “Classifier-free diffusion guidance”. In: *arXiv preprint arXiv:2207.12598* (2022).
- [63] Jonathan Ho et al. “Video diffusion models”. In: *Advances in neural information processing systems* 35 (2022), pp. 8633–8646.
- [64] Hao Hu et al. “Unsupervised Behavior Extraction via Random Intent Priors”. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: <https://openreview.net/forum?id=4vGVQVz5KG>.
- [65] Physical Intelligence et al. “ $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization”. In: *arXiv preprint arXiv:2504.16054* (2025).
- [66] Tommi Jaakkola, Michael Jordan, and Satinder Singh. “Convergence of stochastic iterative dynamic programming algorithms”. In: *Advances in neural information processing systems* 6 (1993).
- [67] Nathan Kallus and Angela Zhou. “Confounding-robust policy evaluation in infinite-horizon reinforcement learning”. In: *Advances in neural information processing systems* 33 (2020), pp. 22293–22304.

- [68] Nathan Kallus and Angela Zhou. “Minimax-optimal policy learning under unobserved confounding”. In: *Management Science* 67.5 (2021), pp. 2870–2890.
- [69] Bingyi Kang et al. “Efficient diffusion policies for offline reinforcement learning”. In: *Neural Information Processing Systems (NeurIPS)*. 2023.
- [70] Chinmaya Kausik et al. “Offline policy evaluation and optimization under confounding”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2024, pp. 1459–1467.
- [71] Jens Kober and Jan Peters. “Policy search for motor primitives in robotics”. In: *Advances in neural information processing systems* 21 (2008).
- [72] Anita De Mello Koch et al. *Learning Transferable Sub-goals by Hypothesizing Generalizing Features*. 2025. URL: <https://openreview.net/forum?id=OvrnA3GMiX>.
- [73] Prajwal Koirala and Cody Fleming. “Flow-Based Single-Step Completion for Efficient and Expressive Policy Learning”. In: *arXiv preprint arXiv:2506.21427* (2025).
- [74] George Dimitri Konidaris. *Autonomous robot skill acquisition*. University of Massachusetts Amherst, 2011.
- [75] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. “Offline Reinforcement Learning with Implicit Q-Learning”. In: *International Conference on Learning Representations*. 2022. URL: <https://openreview.net/forum?id=68n2s9ZJWF8>.
- [76] Tadashi Kozuno et al. “Revisiting Peng’s $Q(\lambda)$ for Modern Reinforcement Learning”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5794–5804.
- [77] Tejas D Kulkarni et al. “Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation”. In: *Advances in neural information processing systems* 29 (2016).
- [78] Aviral Kumar et al. “Conservative Q-learning for offline reinforcement learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1179–1191.
- [79] Seunghyun Lee et al. “Offline-to-online reinforcement learning via balanced replay and pessimistic Q-ensemble”. In: *Conference on Robot Learning*. PMLR. 2022, pp. 1702–1712.
- [80] Ge Li et al. “TOP-ERL: Transformer-based Off-Policy Episodic Reinforcement Learning”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=N4NhVN30ph>.
- [81] Qiyang Li, Zhiyuan Zhou, and Sergey Levine. “Reinforcement Learning with Action Chunking”. In: *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. 2025. URL: <https://openreview.net/forum?id=XUks1Y96NR>.
- [82] Steven Li et al. “Learning multimodal behaviors from scratch with diffusion policy gradient”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 38456–38479.

- [83] Toru Lin et al. “Learning Visuotactile Skills with Two Multifingered Hands”. In: *arXiv:2404.16823* (2024).
- [84] Toru Lin et al. “Learning visuotactile skills with two multifingered hands”. In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2025, pp. 5637–5643.
- [85] Yaron Lipman et al. “Flow matching guide and code”. In: *arXiv preprint arXiv:2412.06264* (2024).
- [86] Qinghua Liu et al. “When is partially observable reinforcement learning not scary?” In: *Conference on Learning Theory*. PMLR. 2022, pp. 5175–5220.
- [87] Xingchao Liu, Chengyue Gong, and qiang liu. “Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow”. In: *The Eleventh International Conference on Learning Representations*. 2023. URL: <https://openreview.net/forum?id=XVjTT1nw5z>.
- [88] Aaron Lou, Chenlin Meng, and Stefano Ermon. “Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution”. In: *Forty-first International Conference on Machine Learning*. 2024. URL: <https://openreview.net/forum?id=CNicRIVIPA>.
- [89] Cheng Lu et al. “Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning”. In: *International Conference on Machine Learning (ICML)*. 2023.
- [90] Haitong Ma et al. “Efficient Online Reinforcement Learning for Diffusion Policy”. In: *arXiv preprint arXiv:2502.00361* (2025).
- [91] Ajay Mandlekar et al. “What Matters in Learning from Offline Human Demonstrations for Robot Manipulation”. In: *Proceedings of the 5th Conference on Robot Learning*. Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, Aug. 2022, pp. 1678–1690. URL: <https://proceedings.mlr.press/v164/mandlekar22a.html>.
- [92] Max Sobol Mark et al. “Policy-Agnostic RL: Offline RL and Online RL Fine-Tuning of Any Class and Backbone”. In: *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*. 2025. URL: <https://openreview.net/forum?id=VGOFxdc6AD>.
- [93] Amy McGovern and Richard S Sutton. “Macro-actions in reinforcement learning: An empirical analysis”. In: (1998).
- [94] Ishai Menache, Shie Mannor, and Nahum Shimkin. “Q-cut—dynamic discovery of sub-goals in reinforcement learning”. In: *Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings 13*. Springer. 2002, pp. 295–306.
- [95] Josh Merel et al. “Neural probabilistic motor primitives for humanoid control”. In: *arXiv preprint arXiv:1811.11711* (2018).

- [96] Rui Miao, Zhengling Qi, and Xiaoke Zhang. “Off-policy evaluation for episodic partially observable markov decision processes under non-parametric models”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 593–606.
- [97] Prabhat K Mishra, Debasish Chatterjee, and Daniel E Quevedo. “Stochastic predictive control under intermittent observations and unreliable actions”. In: *Automatica* 118 (2020), p. 109012.
- [98] Rémi Munos et al. “Safe and efficient off-policy reinforcement learning”. In: *Advances in neural information processing systems* 29 (2016).
- [99] Vivek Myers et al. “Offline Goal-conditioned Reinforcement Learning with Quasimetric Representations”. In: *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. 2025. URL: <https://openreview.net/forum?id=P23UMiw7iJ>.
- [100] Ofir Nachum et al. “Data-efficient hierarchical reinforcement learning”. In: *Advances in neural information processing systems* 31 (2018).
- [101] Ashvin Nair et al. “AWAC: Accelerating online reinforcement learning with offline datasets”. In: *arXiv preprint arXiv:2006.09359* (2020).
- [102] Mitsuhiro Nakamoto et al. “Cal-QL: Calibrated offline RL pre-training for efficient online fine-tuning”. In: *Advances in Neural Information Processing Systems* 36 (2024).
- [103] Hongseok Namkoong et al. “Off-policy Policy Evaluation For Sequential Decisions Under Unobserved Confounding”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 18819–18831. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/da21bae82c02d1e2b8168d57cd3fbab7-Paper.pdf.
- [104] Soroush Nasiriany et al. “Learning and Retrieval from Prior Data for Skill-based Imitation Learning”. In: *Conference on Robot Learning*. 2022.
- [105] Kei Noba and Kazutoshi Yamazaki. “On stochastic control under Poisson observations: optimality of a barrier strategy in a general Lévy model”. In: *arXiv preprint arXiv:2210.00501* (2022).
- [106] Alexandros Paraschos et al. “Probabilistic movement primitives”. In: *Advances in neural information processing systems* 26 (2013).
- [107] Seohong Park, Tobias Kreiman, and Sergey Levine. “Foundation Policies with Hilbert Representations”. In: *Forty-first International Conference on Machine Learning*. 2024. URL: <https://openreview.net/forum?id=LhNsSaAKub>.
- [108] Seohong Park, Qiyang Li, and Sergey Levine. “Flow Q-Learning”. In: *Forty-second International Conference on Machine Learning*. 2025. URL: <https://openreview.net/forum?id=KVf2SFL1pi>.
- [109] Seohong Park et al. “HIQL: Offline Goal-Conditioned RL with Latent States as Actions”. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: <https://openreview.net/forum?id=cLQCCTVDuW>.

- [110] Seohong Park et al. “Horizon Reduction Makes RL Scalable”. In: *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. 2025. URL: <https://openreview.net/forum?id=hguaupzLCU>.
- [111] Seohong Park et al. “OGBench: Benchmarking Offline Goal-Conditioned RL”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=M992mjgKzI>.
- [112] Ahmad Parsian and SNUA Kirmani. “Estimation under LINEX loss function”. In: *Handbook of applied econometrics and statistical inference*. CRC Press, 2002, pp. 75–98.
- [113] Xue Bin Peng et al. “Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning”. In: *Acm transactions on graphics (tog)* 36.4 (2017), pp. 1–13.
- [114] Karl Pertsch, Youngwoon Lee, and Joseph Lim. “Accelerating reinforcement learning with learned skill priors”. In: *Conference on robot learning*. PMLR. 2021, pp. 188–204.
- [115] Angus Phillips et al. “Particle Denoising Diffusion Sampler”. In: *Proceedings of the 41st International Conference on Machine Learning*. Ed. by Ruslan Salakhutdinov et al. Vol. 235. Proceedings of Machine Learning Research. PMLR, 21–27 Jul 2024, pp. 40688–40724. URL: <https://proceedings.mlr.press/v235/phillips24a.html>.
- [116] Lev Semenovich Pontryagin et al. *The Mathematical Theory of Optimal Processes*. Wiley, 1962.
- [117] Doina Precup, Richard S Sutton, and Satinder Singh. “Eligibility traces for off-policy policy evaluation.” In: *ICML*. Vol. 2000. Citeseer. 2000, pp. 759–766.
- [118] Michael Psenka et al. “Learning a Diffusion Model Policy from Rewards via Q-Score Matching”. In: *Proceedings of the 41st International Conference on Machine Learning*. Ed. by Ruslan Salakhutdinov et al. Vol. 235. Proceedings of Machine Learning Research. PMLR, 21–27 Jul 2024, pp. 41163–41182. URL: <https://proceedings.mlr.press/v235/psenka24a.html>.
- [119] Michael Psenka et al. “Learning a diffusion model policy from rewards via Q-score matching”. In: *International Conference on Machine Learning (ICML)*. 2024.
- [120] Allen Z. Ren et al. “Diffusion Policy Policy Optimization”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=mEpqHvbD2h>.
- [121] Martin Riedmiller et al. “Learning by playing solving sparse reward tasks from scratch”. In: *International conference on machine learning*. PMLR. 2018, pp. 4344–4353.
- [122] Robin Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

- [123] Mark Rowland, Will Dabney, and Rémi Munos. “Adaptive trade-offs in off-policy learning”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 34–44.
- [124] Stefan Schaal. “Dynamic movement primitives-A framework for motor control in humans and humanoid robotics”. In: *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.
- [125] Younggyo Seo and Pieter Abbeel. “Coarse-to-fine Q-Network with Action Sequence for Data-Efficient Reinforcement Learning”. In: *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. 2025. URL: <https://openreview.net/forum?id=VoFXUNc9Zh>.
- [126] Younggyo Seo, Jafar Uruç, and Stephen James. “Continuous Control with Coarse-to-fine Reinforcement Learning”. In: *8th Annual Conference on Robot Learning*. 2024. URL: <https://openreview.net/forum?id=WjDR48cL30>.
- [127] Tanmay Shankar and Abhinav Gupta. “Learning robot skills with temporal variational inference”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8624–8633.
- [128] Chengchun Shi et al. “A minimax learning approach to off-policy evaluation in confounded partially observable markov decision processes”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 20057–20094.
- [129] Chengchun Shi et al. “Off-policy confidence interval estimation with confounded markov decision process”. In: *Journal of the American Statistical Association* 119.545 (2024), pp. 273–284.
- [130] Max Simchowitz, Daniel Pfrommer, and Ali Jadbabaie. “The pitfalls of imitation learning when actions are continuous”. In: *Proceedings of Thirty Eighth Conference on Learning Theory*. Ed. by Nika Haghtalab and Ankur Moitra. Vol. 291. Proceedings of Machine Learning Research. PMLR, 30 Jun–04 Jul 2025, pp. 5248–5351. URL: <https://proceedings.mlr.press/v291/simchowitz25a.html>.
- [131] Özgür Şimşek and Andrew G. Barto. *Betweenness Centrality as a Basis for Forming Skills*. English. WorkingPaper. University of Massachusetts Amherst, Apr. 2007.
- [132] Avi Singh et al. “Parrot: Data-Driven Behavioral Priors for Reinforcement Learning”. In: *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=Ysuv-WOFeKR>.
- [133] Raghav Singhal et al. “A General Framework for Inference-time Scaling and Steering of Diffusion Models”. In: *Forty-second International Conference on Machine Learning*. 2025. URL: <https://openreview.net/forum?id=Jp988ELppQ>.
- [134] Marta Skreta et al. “Feynman-kac correctors in diffusion: Annealing, guidance, and product of experts”. In: *arXiv preprint arXiv:2503.02819* (2025).

- [135] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising diffusion implicit models”. In: *arXiv preprint arXiv:2010.02502* (2020).
- [136] Yang Song and Stefano Ermon. “Generative modeling by estimating gradients of the data distribution”. In: *Advances in neural information processing systems* 32 (2019).
- [137] Yang Song and Stefano Ermon. “Improved techniques for training score-based generative models”. In: *Advances in neural information processing systems* 33 (2020), pp. 12438–12448.
- [138] Yang Song et al. “Consistency models”. In: (2023).
- [139] Aravind Srinivas et al. “Option discovery in hierarchical reinforcement learning using spatio-temporal clustering”. In: *arXiv preprint arXiv:1605.05359* (2016).
- [140] Nisan Stiennon et al. “Learning to summarize with human feedback”. In: *Advances in neural information processing systems* 33 (2020), pp. 3008–3021.
- [141] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.
- [142] Richard S Sutton, Doina Precup, and Satinder Singh. “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning”. In: *Artificial intelligence* 112.1-2 (1999), pp. 181–211.
- [143] Denis Tarasov et al. “Revisiting the Minimalist Approach to Offline Reinforcement Learning”. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: <https://openreview.net/forum?id=vqGWslLeEw>.
- [144] Denis Tarasov et al. “Revisiting the minimalist approach to offline reinforcement learning”. In: *Advances in Neural Information Processing Systems* 36 (2024).
- [145] Guy Tennenholtz, Uri Shalit, and Shie Mannor. “Off-policy evaluation in partially observable environments”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 06. 2020, pp. 10276–10283.
- [146] James Thornton et al. “Controlled Generation with Distilled Diffusion Energy Models and Sequential Monte Carlo”. In: *The 28th International Conference on Artificial Intelligence and Statistics*. 2025. URL: <https://openreview.net/forum?id=6GyX0YRw8P>.
- [147] Dong Tian et al. “Chunking the critic: A transformer-based soft actor-critic with N-step returns”. In: *arXiv preprint arXiv:2503.03660* (2025).
- [148] Ahmed Touati, Jérémy Rapin, and Yann Ollivier. “Does Zero-Shot Reinforcement Learning Exist?” In: *The Eleventh International Conference on Learning Representations*. 2022.
- [149] Stephen Tu et al. “On the sample complexity of stability constrained imitation learning”. In: *Learning for Dynamics and Control Conference*. PMLR. 2022, pp. 180–191.

- [150] Alexander Vezhnevets et al. “Strategic attentive writer for learning macro-actions”. In: *Advances in neural information processing systems* 29 (2016).
- [151] Alexander Sasha Vezhnevets et al. “Feudal networks for hierarchical reinforcement learning”. In: *International conference on machine learning*. PMLR. 2017, pp. 3540–3549.
- [152] Andrew Wagenmaker et al. “Steering Your Diffusion Policy with Latent Space Reinforcement Learning”. In: *Conference on Robot Learning* (2025).
- [153] Hui Wang. “Some control problems with random intervention times”. In: *Advances in Applied Probability* 33.2 (2001), pp. 404–422.
- [154] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. “Diffusion policies as an expressive policy class for offline reinforcement learning”. In: *International Conference on Learning Representations (ICLR)*. 2023.
- [155] Darrell Whitley et al. “Genetic reinforcement learning for neurocontrol problems”. In: *Machine Learning* 13.2 (1993), pp. 259–284.
- [156] Max Wilcoxson et al. “Leveraging Skills from Unlabeled Prior Data for Efficient On-line Exploration”. In: *International Conference on Machine Learning (ICML)*. 2025.
- [157] Yihong Wu. “Lecture notes on information-theoretic methods for high-dimensional statistics”. In: *Lecture Notes for ECE598YW (UIUC)* 16 (2017), p. 15.
- [158] Kevin Xie et al. “Latent Skill Planning for Exploration and Transfer”. In: *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=jXe91kq3jAq>.
- [159] Shuhao Yan, Mark Cannon, and Paul J Goulart. “Stochastic output feedback MPC with intermittent observations”. In: *Automatica* 141 (2022), p. 110282.
- [160] Tianwei Yin et al. “One-step diffusion with distribution matching distillation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2024, pp. 6613–6623.
- [161] Xiu Yuan et al. “Policy Decorator: Model-Agnostic Online Refinement for Large Policy Model”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=e5jGTEiJMT>.
- [162] Ruoqi Zhang et al. “Entropy-regularized diffusion policy with Q-ensembles for offline reinforcement learning”. In: *Neural Information Processing Systems (NeurIPS)*. 2024.
- [163] Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. “Energy-Weighted Flow Matching for Offline Reinforcement Learning”. In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: <https://openreview.net/forum?id=HA0oLUvuGI>.
- [164] Tony Z. Zhao et al. “Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware”. In: *Proceedings of Robotics: Science and Systems*. Daegu, Republic of Korea, July 2023. DOI: 10.15607/RSS.2023.XIX.016.

Appendix A

Appendix for Q-learning with Adjoint-Matching

A.1 Additional Implementation Details for QAM

For the critic, we use an ensemble of $K = 10$ critic functions $\phi^1, \dots, \phi^K$ and use the pessimistic target value backup following [43] (originally inspired by [54]). The loss function for each ϕ^j , $j \in \{1, 2, \dots, K\}$ is

$$L(\phi^j) = \left(Q_{\phi^j}(s, a) - r - \gamma [\bar{Q}_{\text{mean}}(s', a') - \rho \bar{Q}_{\text{std}}(s', a')] \right)^2, \quad (\text{A.1})$$

where $\bar{Q}_{\text{mean}}(s', a') := \frac{1}{K} \sum_k Q_{\phi^k}(s', a')$, $\bar{Q}_{\text{std}}(s', a') = \frac{1}{K} \sqrt{\sum_k (Q_{\phi^k}(s', a') - \bar{Q}_{\text{mean}}(s', a'))^2}$, and a' is the action sampled from the fine-tuned flow model: $f_\theta(s, \cdot, \cdot)$. For all our experiments, we do not use a separate training process for f_β and instead train it at the same time as f_θ and Q_ϕ , following [108, 81]. For all our loss functions, the transition tuple (s, a, s', r) is drawn from D uniformly. During offline training, D is the offline data. During online fine-tuning, D is combination of the offline and online replay buffer data without any re-weighting.

A.2 Full Results

We provide full training curves and a result table for all methods and tasks individually. Figure A.1 shows both the offline phase (until 1M steps) and the online phase (after 1M steps). Table A.1 reports the performance at the end of the 1M-step offline training. The hyperparameters can be found in Section A.5. Figure A.2 shows the full training curves for our sensitivity analysis.

A.3 Domain and experiment details

We consider 10 domains in our experiments. The dataset size, episode length, and the action dimension for each domain is available in Table A.2. For each method and each task, we run 12 seeds. All plots and tables report the means with 95% confidence intervals computed via bootstrapping using 5000 samples. All our experiments are run on NVIDIA RTX-A5000 GPU and our code is written in JAX [15]. We use NVIDIA-A5000 GPU to run all our experiments. Each complete offline-to-online experiment run takes around 3 hours. To reproduce all our results in Table 4.3 and Figure 4.5, we estimate that it would take around

$$\underbrace{5}_{\text{hours per single run}} \times \underbrace{17}_{\text{\# of methods}} \times \underbrace{50}_{\text{\# of tasks}} \times \underbrace{12}_{\text{\# of seeds}} = 51\,000 \text{ GPU hours.}$$

A.4 Baselines

In this section, we describe in detail how each of our baselines are implemented. In all the loss functions below, unless specified otherwise, s, a, r, s' are assumed to be sampled from D and as part of the expectation even when it is not explicitly written under the expectation (*i.e.*, $\mathbb{E}_{z \sim \mathcal{N}}[\cdot] := \mathbb{E}_{(s, a, r, s') \sim D, z \sim \mathcal{N}}[\cdot]$).

1. Backprop-based.

FBRAC is a baseline considered in FQL [108] as a flow counterpart of diffusion Q-learning (DQL) [154], where the multi-step flow policy is directly optimized against the Q-function with backpropagation through time (BPTT). In addition to maximizing the Q-value, FBRAC also has a behavior cloning term where the flow policy is trained with the standard flow-matching objective on the dataset action with a BC coefficient α .

We implement this baseline by training a flow-matching policy with the following loss:

$$L_{\text{FBRAC}}(\theta) = \alpha L_{\text{FM}}(\theta) + L_{\text{BPTT}}(\theta), \quad (\text{A.2})$$

where L_{FM} is the standard flow-matching loss that clones the data behavior (*i.e.*, Equation (4.9)) and

$$L_{\text{BPTT}}(\theta) = -\mathbb{E}_{a^0 = z \sim \mathcal{N}(0, I_A)} \left[Q_\phi \left(s, \text{Clip} \left[z + h \sum_{k=0}^{T-1} f_\theta(s, a^{k-1}, kh) \right]_{-1}^1 \right) \right], \quad (\text{A.3})$$

where $\text{Clip}[\cdot]_a^b$ is an element-wise clipping function that makes sure the actions generated from the flow model f_θ are within the valid range $[-1, 1]$ and $\{a^i\}_i$ is the discrete approximation of the ODE trajectory using Euler's method with a step size of $h = 1/T$:

$$a^{i+1} := a^i + h f_\theta(s, a^{i-1}, ih), \forall i \in \{0, 1, \dots, T\}. \quad (\text{A.4})$$

We use

$$\text{ODE}(f_\theta(s, \cdot, \cdot), z) := \text{Clip} \left[z + h \sum_{i=0}^{T-1} f_\theta(s, a^{i-1}, ih) \right]_{-1}^1 \quad (\text{A.5})$$

as the abbreviation for the rest of this section, and additionally use π_{f_θ} to denote the distribution of actions generated by f_θ .

The critic loss is the standard TD backup:

$$L(\phi) = \mathbb{E}_{z \sim \mathcal{N}} [(Q_\phi(s, a) - r - Q_{\bar{\phi}}(s', \text{ODE}(f_\theta(s', \cdot, \cdot), z)))^2] \quad (\text{A.6})$$

In practice, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (A.1) and the policy we use to interact with the environment is π_{f_θ} .

FQL [108] distills a multi-step flow policy into a one-step distillation to avoid BPTT.

This baseline is implemented by training a behavior cloning flow-matching policy (*i.e.*, $f_\theta : \mathcal{S} \times \mathbb{R}^A \times [0, 1] \rightarrow \mathbb{R}^A$), and a 1-step distilled noise-conditioned policy (*i.e.*, $\Omega_\omega : \mathcal{S} \times \mathbb{R}^A \rightarrow \mathbb{R}^A$):

$$L_{\text{FQL}}(\theta, \omega) = L_{\text{FM}}(\theta) + L_{\text{onestep}}(\omega) \quad (\text{A.7})$$

where $L_{\text{FM}}(\theta)$ is the standard flow-matching loss (*i.e.*, Equation (4.9)) and

$$L_{\text{onestep}}(\omega) = \mathbb{E}_{z \sim \mathcal{Z}} [\alpha \|\Omega_\omega(s, z) - \text{ODE}(f_\theta(s', \cdot, \cdot), z)\|_2^2 - Q(s, \Omega_\omega(s, z))] \quad (\text{A.8})$$

where α is the BC coefficient that controls how close the 1-step distilled policy should be relative to the BC policy π_{f_θ} . Finally, the critic loss is the standard TD backup:

$$L(\phi) = \mathbb{E}_{z \sim \mathcal{N}} [(Q_\phi(s, a) - r - Q_{\bar{\phi}}(s', \Omega_\omega(s', z)))^2] \quad (\text{A.9})$$

In practice, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (A.1) and the policy we use to interact with the environment is π_{Ω_ω} where the action is sampled by first drawing a Gaussian noise $z \sim \mathcal{N}$ and then obtaining the action by running through the one-step distilled model: $a = \Omega_\omega(s, z)$.

2. Directly using the action gradient of the critic (*i.e.*, $\nabla_a Q(s, a)$) with approximations.

QSM [119] uses the action gradient of the critic to train a diffusion model such that it can reconstruct the policy that follows the Boltzmann distribution of the Q-function:

$$\pi(\cdot \mid s) \propto \exp(\tau Q_\phi(s, \cdot)). \quad (\text{A.10})$$

To approximate the score of the intermediate actions, $\nabla_{a^i} \log \pi^i(a^i | s)$, QSM directly uses the critic evaluated at a^i :

$$\nabla_{a^i} \log \pi^i(a^i | s) \approx \tau \nabla_{a^i} Q_\phi(s, a^i) \quad (\text{A.11})$$

where $a^i = \sqrt{\alpha^i} a + \sqrt{1 - \alpha^i} \varepsilon$ with $\varepsilon \sim \mathcal{N}$ and $\{\alpha^0, \dots, \alpha^{T-1}\}$ being any diffusion schedule with the number of diffusion steps of T . For all diffusion methods in this paper, we follow [57] to use the variance preserving diffusion schedule:

$$\beta^i = 1 - \exp \left(-\frac{b_{\min}}{T} - \frac{(b_{\max} - b_{\min})(2i + 1)}{2T^2} \right), \quad (\text{A.12})$$

$$\alpha^i = \prod_{j=0}^i (1 - \beta^j), \quad (\text{A.13})$$

for all $i \in \{0, 1, \dots, T-1\}$ with $b_{\min} = 0.1, b_{\max} = 10$ and T being the number of diffusion steps.

To train the diffusion model, QSM uses the following loss function:

$$L_{\text{QSM}}(\theta) = \mathbb{E}_{\varepsilon \sim \mathcal{N}, i \sim \mathcal{U}\{0, \dots, T-1\}} [\| -\tau \nabla_a Q_\phi(s, a^i) - f_\theta(s, a^i, i) \|_2^2], \quad (\text{A.14})$$

where again $a^i = \sqrt{\alpha^i} a + \sqrt{1 - \alpha^i} \varepsilon$. In practice, we find that using $\nabla_a Q_{\bar{\phi}}(s, a^i)$ (the action gradient of the critic network) helps learning stability, so we use that instead.

To adopt QSM into the offline-to-online RL setting, we additionally augment the loss function with the standard diffusion loss [61] as the behavior regularization:

$$L_{\text{DDPM}}(\theta) = \mathbb{E}_{\varepsilon \sim \mathcal{N}, i \sim \mathcal{U}\{0, \dots, T-1\}} [\| \varepsilon - f_\theta(s, a^i, i) \|_2^2]. \quad (\text{A.15})$$

The overall actor loss is thus

$$L(\theta) = L_{\text{QSM}}(\theta) + \eta L_{\text{DDPM}}(\theta). \quad (\text{A.16})$$

QSM then uses the standard diffusion denoising procedure (with the clipping to make sure generated actions are within $[-1, 1]^{|A|}$):

$$a^{i-1} \leftarrow \text{Clip} \left[\frac{1}{\sqrt{1 - \beta^i}} \left(a^i - \frac{\beta^i}{\sqrt{1 - \alpha^i}} f_\theta(s, a^i, i) \right) + \sqrt{\beta^i} \varepsilon^i \right]_{-1}^1, \quad \varepsilon^i \sim \mathcal{N}, \quad (\text{A.17})$$

where $a^T \sim \mathcal{N}$. We denote $\text{Diff}(f_\theta(s, \cdot, \cdot))$ as the resulting a^0 after going through the diffusion procedure above and $\pi_{f_\theta} : s \mapsto \text{Diff}(f_\theta(s, \cdot, \cdot))$ as the policy that generates actions using f_θ .

The critic loss can now be defined as follows:

$$L(\phi) = \mathbb{E}_{z \sim \mathcal{N}} [(Q_\phi(s, a) - r - Q_{\bar{\phi}}(s', a' \sim \pi_{f_\theta}(\cdot | s')))^2] \quad (\text{A.18})$$

In practice, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (A.1) and the policy we use to interact with the environment is π_{f_θ} .

DAC [42] is another method that uses a similar approximation with a behavior prior (*i.e.*, $\nabla_{a^i} \log \pi^i(a^i | s) \approx \nabla_{a^i} \log \pi_\beta^i(a^i | s) + \tau \nabla_{a^i} Q_\phi(s, a^i)$). To train the diffusion model, DAC matches f_θ to a linear combination of ε (behavior cloning) and $\nabla_{a^i} Q(s, a^i)$ (Q-maximization).

$$\begin{aligned} L_{\text{DAC}}(\theta) &= \mathbb{E}_{\varepsilon \sim \mathcal{N}, i \sim \mathcal{U}\{0, \dots, T-1\}} [\|\varepsilon - \tau \nabla_{a^i} Q_\phi(s, a^i) - f_\theta(s, a^i, i)\|_2^2] \\ &= \mathbb{E}_{\varepsilon \sim \mathcal{N}, i \sim \mathcal{U}\{0, \dots, T-1\}} [\|\varepsilon - f_\theta(s, a^i, i)\|_2^2 + \eta^{-1} (f_\theta(s, a^i, i) \cdot \nabla_{a^i} Q_\phi(s, a^i))] + C, \end{aligned} \quad (\text{A.19})$$

with $\eta = 2/\tau$. In practice, it is implemented as a combination of

$$L(\theta) = \mathbb{E}_{\varepsilon \sim \mathcal{N}, i \sim \mathcal{U}\{0, \dots, T-1\}} [\|(f_\theta(s, a^i, i) \cdot \nabla_{a^i} Q_\phi(s, a^i))\|] + \eta L_{\text{DDPM}}(\theta) \quad (\text{A.20})$$

DAC uses the following procedure:

$$\hat{a}^{i-1} \leftarrow \text{Clip} \left[\frac{1}{\sqrt{\alpha^i}} \left(a^i - \sqrt{1 - \alpha^i} f_\theta(s, a^i, i) \right) \right]_{-1}^1 \quad (\text{A.21})$$

$$a^{i-1} \leftarrow \frac{\beta^i \sqrt{\alpha^{i-1}}}{1 - \alpha^i} \hat{a}^{i-1} + \sqrt{1 - \beta^i} (1 - \alpha^{i-1}) a^i + \sqrt{\beta^i} \varepsilon^i, \quad \varepsilon^i \sim \mathcal{N} \quad (\text{A.22})$$

where $a^T \sim \mathcal{N}$ and $\hat{a}^{i-1}$ can be interpreted as an approximation of the denoised action (*e.g.*, in the DDIM sampler [135]). This approximated denoised action is first clipped to be within $[-1, 1]^{|A|}$ and then used to reconstruct a^{i-1} using the closed-form Gaussian conditional probability distribution of $p(a^{i-1} | a^0 = \hat{a}^{i-1}, a^T = a)$ (*e.g.*, Eq. (7) in [61]). In practice, DAC also uses $\sqrt{\beta^i}$ as standard deviation for this conditional distribution instead of the correct one $\tilde{\beta}^i = \frac{\sqrt{1 - \alpha^{i-1}}}{\sqrt{1 - \alpha^i}} \beta^i$ as an approximation. This is why in the expression above the multiplier before ε^i is $\sqrt{\beta^i}$.

The critic loss can now be defined as follows:

$$L(\phi) = \mathbb{E}_{z \sim \mathcal{N}} [(Q_\phi(s, a) - r - Q_{\bar{\phi}}(s', a' \sim \pi_{f_\theta}(\cdot | s')))^2], \quad (\text{A.23})$$

where π_{f_θ} is the policy that generates actions using the procedure above in Equation (A.21) and Equation (A.22). In practice, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (A.1) and the policy we use to interact with the environment is π_{f_θ} .

CGQL is a novel baseline built on top of the idea of classifier guidance [29]. In particular, we combine the velocity field of a behavior cloning flow policy and the gradient field of the Q-function to form a new velocity field that approximates the velocity field that generates the optimal behavior-constrained action distribution.

More specifically, we implement this baseline by interpreting $Q_\phi(s, \cdot)$ as the score of the optimal entropy-regularized distribution $\log \pi^*(\cdot | s)$ (where $\pi^*(\cdot | s) \propto e^{\tau Q_\phi(s, \cdot)}$). The corresponding velocity field that generates this distribution of actions can be obtained through a simple conversion (e.g., following Equation 4.79 from [85]):

$$v_\phi(s, a, t) := \frac{(1-t)\tau \nabla_a Q(s, a, t) + a}{t}, \quad (\text{A.24})$$

where

$$Q(s, a_t, t) := \frac{1}{\tau} \log \mathbb{E}_{a_1 \sim P(a_1 | a_t)} [e^{\tau Q_\phi(s, a_1)}], \quad (\text{A.25})$$

is an approximation of the score of the distribution over the noisy intermediate actions. In the classifier guidance literature, the score of the noisy examples is approximated by the score of the noise-free examples at the noisy examples [29]. In our setting this translates to

$$\hat{f}_\phi(s, a_t, t) := \frac{(1-t)\tau \nabla_a Q_\phi(s, a_t) + a_t}{t}. \quad (\text{A.26})$$

Empirically, both versions (f_ϕ and $\hat{f}_\phi$) perform similarly and we opt for a simpler design $\hat{f}$ as it does not require learning or approximating $Q(s, a, t)$ for all $u \in [0, 1]$. Finally, we add the velocity field defined by Q_ϕ directly to the behavior cloning velocity field to form our policy:

$$f = f_\beta + \vartheta \hat{f}_\phi, \quad (\text{A.27})$$

where f_β is trained with the standard flow-matching loss (i.e., $L(\beta) = L_{\text{FM}}(\beta)$) and ϑ is a coefficient that modulates influence of the guidance that we find to be helpful (e.g., $\vartheta < 1$ often works better than $\vartheta = 1$). The critic loss uses π_v to backup the target Q -value:

$$L(\phi) = \mathbb{E}_{z \sim \mathcal{N}} [(Q_\phi(s, a) - r - Q_\phi(s', \text{ODE}(f(s', \cdot, \cdot), z)))^2]. \quad (\text{A.28})$$

In practice, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (A.1) and the policy we use to interact with the environment is π_v (generated from the summation of f_β and $\hat{f}_\phi$).

CGQL-MSE/Linex. Alternatively, we can approximate Q_t in Equation (A.25) more closely with a training objective. In particular, we explore the following two regression objectives:

$$\textbf{MSE:} \quad L_{\text{MSE}}(\zeta) = \mathbb{E}_{t \sim \mathcal{U}[0,1], z \sim \mathcal{N}} \left[\left(\hat{Q}_\zeta(s, a_t, t) - Q_\phi(s, a) \right)^2 \right] \quad (\text{A.29})$$

$$\textbf{Linex:} \quad L_{\text{Linex}}(\zeta) = \mathbb{E}_{t \sim \mathcal{U}[0,1], z \sim \mathcal{N}} \left[\exp(\tau(Q_\phi(s, a) - \hat{Q}_\zeta(s, a_t, t))) + \tau \hat{Q}_\zeta(s, a_t, t) \right] \quad (\text{A.30})$$

where $a_t := (1 - t)z + ta$. The optimal solution of $\hat{Q}_\zeta$ in Equation (A.29) is not exactly the same as the desired Q_ϕ in Equation (A.25) but constitutes a lower-bound due to Jensen's inequality:

$$Q_{\text{MSE}}^*(s, a_t, t) = \mathbb{E}_{z,t} [Q_\phi(s, a_t)] \leq \frac{1}{\tau} \log \mathbb{E}_{a_1 \sim P(a_1|a_t)} [e^{\tau Q_\phi(s, a_t)}]. \quad (\text{A.31})$$

The second objective resembles the classic Linex objective [112] where the optimal solution of $\hat{Q}_\zeta$ in Equation (A.30) is the same as the desired Q in Equation (A.25):

$$Q_{\text{Linex}}^*(s, a_t, t) = \frac{1}{\tau} \log \mathbb{E}_{a_1 \sim P(a_1|a_t)} [e^{\tau Q_\phi(s, a_1)}]. \quad (\text{A.32})$$

A discussion on this can be found in [99]. For completeness, we also show this below. Without loss of generality, we just need to show that $\exp(y) = \mathbb{E}[\exp(x)] = \int p(x) \exp(x) dx$ is the unique minimum for the following loss function:

$$L(y) = \int p(x) [\exp(x - y) + y] dx \quad (\text{A.33})$$

Taking the first and second derivatives with respect to y gives

$$\frac{dL}{dy} = -\exp(-y) \mathbb{E}[\exp(x)] + 1 \quad (\text{A.34})$$

$$\frac{d^2L}{dy^2} = \exp(-y) \mathbb{E}[\exp(x)] \quad (\text{A.35})$$

Setting Equation (A.34) gives $y = \log \mathbb{E}[\exp(x)]$ at which the second derivative is 1 (*i.e.*, $\frac{d^2L}{dy^2} = 1$). Furthermore, to prevent the exponential blow up of $\exp(Q_\phi(s, a) - \hat{Q}_\xi(s, a_t, t))$, we follow [99] to use a Huber-style loss that locally behaves like a Linex loss but with a linear penalty when the exponential term is too large. In particular, we use the following loss:

$$L_{\text{Linex}^+}(\zeta) = \begin{cases} \mathbb{E} [\exp(\Delta) + \tau \hat{Q}_\xi(s, a_t, t)] , & \Delta < 5 \\ \mathbb{E} [\Delta] , & \Delta \geq 5 \end{cases} \quad (\text{A.36})$$

where $\Delta = \tau(Q_\phi(s, a) - \hat{Q}_\xi(s, a_t, t))$.

In practice, however, both the MSE and Linex objectives (L_{Linex^+} , L_{MSE^+}) can still be unstable and exhibit high variances. Instead of learning $\hat{Q}_\zeta$ and Q_ϕ separately, we find it is often better to directly learn both of them in a single network $Q_\phi(s, a, t) : \mathcal{S} \times \mathcal{A} \times [0, 1] \rightarrow \mathbb{R}$ as follows:

$$L(\phi) = \mathbb{E}_{z \sim \mathcal{N}} [(Q_\phi(s, a, 1) - r - Q_{\bar{\phi}}(s', \text{ODE}(f(s', \cdot, \cdot), z)), z))^2] + \varrho \mathbb{E}_{t \in \mathcal{U}[0, 1], z \sim \mathcal{N}} [(Q_\phi(s, (1 - t)z + ta, t) - Q_{\bar{\phi}}(s, a, 1))^2], \quad (\text{A.37})$$

where ϱ is the coefficient that balances the TD backup for $Q_\phi(s, a, 1)$ and the noisy target regression for $Q_\phi(s, a, t < 1)$. With $Q_\phi(s, \cdot, t)$, we can define our velocity field (which is also used as the TD backup above) as

$$f := f_\beta + \vartheta \hat{f}_\phi, \quad \text{where } \hat{f}_\phi(s, a_t, t) := \frac{\tau \nabla_a Q_\phi(s, a_t, t) + a_t}{t}, \quad (\text{A.38})$$

and ϑ again modulates the guidance strength.

3. Directly using the critic value (i.e., $Q(s, a)$).

FAWAC is a baseline considered in FQL [108] where it uses AWR to train the flow policy similar to QIPO [163].

We implement it by training a flow-matching policy with the weighted flow-matching loss:

$$L_{\text{FAWAC}}(\theta) = \tilde{w}(s, a) L_{\text{FM}}(\theta) \quad (\text{A.39})$$

$$= \tilde{w}(s, a) \mathbb{E}_{t \sim \mathcal{U}[0, 1], z \sim \mathcal{N}} [\|f_\theta(s, (1-t)z + ta, t) - z + a\|_2^2] \quad (\text{A.40})$$

where $\tilde{w}(s, a) = \min(e^{\tau(Q_\phi(s, a) - V_\xi(s))}, 100.0)$. The inverse temperature parameter τ controls how sharp the prior regularized optimal policy distribution is.

The critic function $Q_\phi(s, a)$ is trained with the standard TD backup and the value function $V_\xi(s)$ regresses to the same target:

$$L(\phi) = \mathbb{E}_{z \sim \mathcal{N}} [(Q_\phi(s, a) - r - Q_{\bar{\phi}}(s', \text{ODE}(f(s', \cdot, \cdot), z)))^2] \quad (\text{A.41})$$

$$L(\xi) = \mathbb{E}_{z \sim \mathcal{N}} [(V_\xi(s) - r - Q_{\bar{\phi}}(s', \text{ODE}(f(s', \cdot, \cdot), z)))^2] \quad (\text{A.42})$$

The second line can also be alternatively implemented by regressing to the critic function $Q(s, a)$ directly. We implement it in this particular way because we can re-use the Q -target computed.

In practice, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (A.1) and the policy we use to interact with the environment is π_{f_θ} .

4. Post-processing-based.

FEdit is a baseline that uses the policy edit from a recent offline-to-online RL method conceptually similar to EXPO [37]. We implement a Gaussian edit policy on top of a BC flow policy rather than a diffusion policy used in EXPO. EXPO also uses the standard sample-and-rank trick where it samples multiple actions and rank them based on the value. To keep computational cost down and comparisons fair to other methods, we only use a single edited action for both value backup and evaluation.

We implement this baseline by training a flow-matching policy (*i.e.*, $f_\theta : \mathcal{S} \times \mathbb{R}^A \times [0, 1] \rightarrow \mathbb{R}^A$), and a 1-step Gaussian edit policy (*i.e.*, $\pi_\omega : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathcal{A}}$) implemented with an entropy regularized SAC policy [56]. The loss function can be described as follows:

$$L_{\text{FEdit}}(\theta, \omega) = L_{\text{FM}}(\theta) + L_{\text{Gaussian}}(\omega), \quad \text{s.t.} \quad \mathbb{E}_{s \sim D} [\mathbb{H}(\pi_\omega(\cdot | s))] \geq H_{\text{target}} \quad (\text{A.43})$$

where L_{FM} is the standard flow-matching loss that clones the data behavior (*i.e.*, Equation (4.9)), H_{target} is the target entropy that the Gaussian policy is constrained to be above of, and

$$L_{\text{Gaussian}}(\omega) = \mathbb{E}_{\Delta a \sim \pi_\omega(\cdot | s, \tilde{a}), z \sim \mathcal{N}} [-Q_\phi(s, \text{Clip}[\sigma_a \cdot \Delta a + \tilde{a}]_{-1}^1)] \quad (\text{A.44})$$

where $\tilde{a} = \text{ODE}(f_\theta(s, \cdot, \cdot), z)$ and $\text{Clip}[\cdot]_a^b$ is an element-wise clipping function that makes sure the actions are within the valid range $[-1, 1]$.

Intuitively, the Gaussian SAC policy edits the behavior flow policy by modifying its output action where σ_a is the hyperparameter that controls how much the original behavior actions can be edited.

The critic loss is the standard TD backup:

$$L(\phi) = \mathbb{E}_{z \sim \mathcal{N}, \Delta a' \sim \pi_\omega(\cdot | s', \tilde{a}')} [(Q_\phi(s, a) - r - Q_{\bar{\phi}}(s', \text{Clip}[\tilde{a}' + \sigma_a \cdot \Delta a']_{-1}^1))^2] \quad (\text{A.45})$$

where again $\tilde{a}' = \text{ODE}(f_\theta(s', \cdot, \cdot), z)$. In practice, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (A.1) and the policy we use to interact with the environment is a combination of the BC policy π_{f_θ} and the Gaussian edit policy. We first sample $z \sim \mathcal{N}$ and then run it through the BC flow policy to obtain an initial action $\tilde{a} \leftarrow \text{ODE}(f_\theta(s, \cdot, \cdot), z)$ and then both the initial action and the state is fed into the edit policy to generate the final action $a \leftarrow \tilde{a} + \sigma_a \cdot \Delta a$ where $\Delta a \sim \pi_\omega(\cdot | s, \tilde{a})$.

DSRL [152] is a recently proposed method that performs RL directly in the noise-space of a pre-trained expressive BC policy (flow or diffusion). We use the flow-matching version of DSRL as our method is also based on flow-matching policies. The original DSRL implementation does not fine-tune the BC policy during online learning while all our baselines do fine-tune the BC policy online. To make the comparison fair, we modify the DSRL implementation such that it also fine-tunes the BC policy. One additional implementation trick that allows this modification to work well is the use of a target policy network for the noise-space policy. In general, we find that fine-tuning the BC policy yields better online performance, so we adopt this new design of DSRL in our experiments.

More specifically, we train a flow-matching policy (*i.e.*, $f_\theta : \mathcal{S} \times \mathbb{R}^A \times [0, 1] \rightarrow \mathbb{R}^A$), and a 1-step Gaussian edit policy (*i.e.*, $\pi_\omega : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathcal{A}}$) implemented with an entropy regularized SAC policy [56]. The loss function can be described as follows:

$$L_{\text{DSRL}}(\theta, \omega) = L_{\text{FM}}(\theta) + L_{\text{LatentGaussian}}(\omega), \quad \text{s.t.} \quad \mathbb{E}_{s \sim D} [\mathbb{H}(\pi_\omega(\cdot | s))] \geq H_{\text{target}} \quad (\text{A.46})$$

where L_{FM} is the standard flow-matching loss that clones the data behavior (*i.e.*, Equation (4.9)), H_{target} is the target entropy that the Gaussian policy is constrained to be above of, and

$$L_{\text{LatentGaussian}}(\omega) = \mathbb{E}_{z \sim \pi_\omega(\cdot | s)} [-Q_\psi^z(s, z)] \quad (\text{A.47})$$

where $Q_\psi^z(s, z)$ is a distilled critic function in the noise space that is regressed to the original critic function, $Q_\phi(s, a)$:

$$L(\psi) = \mathbb{E}_{z \sim \mathcal{N}} [(Q_\psi^z(s, z) - Q_\phi(s, \text{ODE}(f_\theta(s, \cdot, \cdot), z)))^2]. \quad (\text{A.48})$$

Intuitively, DSRL directly learns a policy in the noise space by hill-climbing a distilled critic that also operates in the noise space. Finally, the critic loss for the original critic function in the action space is

$$L(\phi) = \mathbb{E}_{z \sim \pi_\omega(\cdot | s')} [(Q_\phi(s, a) - r - Q_\phi(s', \text{ODE}(f_\theta(s', \cdot, \cdot), z)))^2] \quad (\text{A.49})$$

We use $K = 10$ critic functions and pessimistic target value backup as described in Equation (A.1). The policy we use to interact with the environment is a combination of the BC policy π_{f_θ} and the Gaussian noise-space policy. We first sample $z \sim \pi_\omega(\cdot | s)$ and then run it through the BC flow policy to obtain the final action $a \leftarrow \text{ODE}(f_\theta(s, \cdot, \cdot), z)$. One important implementation detail for stability in the offline-to-online setting is to use the target network for the BC flow policy $f_{\bar{\theta}}$ (instead of f_θ). Without it, DSRL can become unstable sometimes when the BC flow policy changes too fast online.

IFQL is a baseline considered in FQL [108] as a flow counterpart of implicit diffusion Q-learning (IDQL) [57], where IQL [75] is used for value learning and the policy extraction is done by sampling multiple actions from a behavior cloning diffusion policy and select the one that maximizes the Q -function value.

More specifically, we train a critic function $Q_\phi(s, a)$ and a value function $V_\xi(s)$ with implicit value backup:

$$L(\phi) = (Q_\phi(s, a) - r - V_\xi(s'))^2 \quad (\text{A.50})$$

$$L(\xi) = F_{\text{exp}}^\kappa(Q_\phi(s, a) - V_\xi(s)) \quad (\text{A.51})$$

where $F_{\text{exp}}^\kappa(u) = |\kappa - \mathbb{I}_{u < 0}|u^2$ is the expectile regression loss function.

On top of that, we also use $K = 10$ critic functions and pessimistic target value backup as described in Equation (A.1) for training the value function $V_\xi(s)$. To extract a policy from $Q_\phi(s, a)$, IFQL uses rejection sampling with a base behavior cloning flow policy that is trained with the standard flow-matching objective. In particular, the output action a^* for s is selected as the following:

$$a^* \leftarrow \arg \max_{a_1, \dots, a_N} Q(s, a_i), \quad a_1, \dots, a_N \sim \pi_{f_\beta}(\cdot | s). \quad (\text{A.52})$$

5. Gaussian.

RLPD [9] is a strong offline-to-online RL method that trains a SAC agent from scratch online with a 50/50 sampling scheme (*i.e.*, 50% of training examples in a batch comes from the offline dataset whereas the other 50% of training examples comes from the online replay buffer). In addition to the standard RLPD update, we also add a behavior cloning (BC) loss which we find to improve the performance further. We tune over the BC coefficient (α) and report the value we use for each domain in Table A.4.

ReBRAC [144] is a strong offline RL method that trains a TD3 [51] agent with behavior cloning loss. In practice, we find two hyperparameters to impact performance the most. The first hyperparameter is α which controls the strength of the behavior cloning loss. The second hyperparameter is σ which controls the magnitude of the Gaussian noise added in the TD3 policy. We set the action noise clip to 0.5 and the actor delay to 2, following the original paper.

A.5 Hyperparameters

While most methods share a common set of hyperparameters (Table A.3 for a fair comparison, most methods need to be tuned for each domain. We include the domain-specific hyperparameters in Table A.4 and their tuning ranges in Table A.5. To pick the hyperparameter for each domain for each method, we first run a sweep over all hyperparameters in the range (specified by Table A.5), or all combinations of hyperparameters if there are multiple hyperparameters involved. The hyperparameter tuning runs use 4 seeds for each method for each hyperparameter configuration for each of the two tasks per domain. For locomotion domains, we use task 1 (the default task) and task 4. For manipulation domains, we use task 2 (the default task) and task 4. We use task 4 in addition to the default task recommended by OGBench because we often find the combination of these two covers the characteristics of each domain better. We then use the combined performance of the two tuning tasks per domain per method to pick the hyperparameter configuration. We include them in Table A.4. Finally, for all our main results, we run all methods on all five tasks for each domain on 12 *new* seeds (different from the tuning seeds). We pick the hyperparameter range such that the total number of tuning runs is similar across methods. To achieve this, we use the following strategy: For methods where more than one hyperparameter needs to be tuned, we use a coarser hyperparameter range. For methods where there is only one hyperparameter, we use a more fine-grained sweep.

		ReBRAC	FBRAC	BAM	FQL	FAWAC	COQL	COQL-M	COQL-L	DAC	QSM	DSRL	FEdit	IFQL	QAM	QAM-F	QAM-E
antmaze-large	task1	98	0	90	93	6	63	56	39	88	87	62	67	41	77	85	87
	task2	88	0	60	86	1	73	49	51	71	78	75	67	14	80	64	81
	task3	98	12	94	59	40	91	90	79	98	99	81	65	54	89	96	94
	task4	94	0	85	54	18	78	78	76	91	91	18	28	25	69	81	70
	task5	95	0	88	86	22	75	80	78	92	96	69	63	45	89	87	83
	agg. (5 tasks)	94	2	84	79	17	76	71	65	88	90	61	58	36	81	83	83
antmaze-giant	task1	36	0	3	0	0	3	0	0	53	62	0	0	0	7	11	0
	task2	73	0	0	0	0	0	0	4	0	1	0	0	0	0	0	0
	task3	13	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	task4	74	0	0	0	0	0	0	0	0	0	12	10	2	34	5	0
	task5	89	0	1	0	0	0	22	11	25	58	2	0	2	49	43	6
	agg. (5 tasks)	57	0	1	0	0	0	3	3	16	24	3	2	1	18	12	1
humanoidmaze-medium	task1	38	26	49	34	18	30	8	55	87	88	49	0	86	40	32	27
	task2	91	78	69	95	44	78	99	93	96	96	91	39	92	97	98	99
	task3	83	28	75	96	20	78	0	62	92	95	36	0	93	96	95	68
	task4	37	3	22	14	1	23	0	2	43	31	0	0	60	3	0	0
	task5	96	59	83	99	36	89	100	98	99	98	90	68	98	99	99	99
	agg. (5 tasks)	69	39	60	68	24	60	42	62	83	82	53	22	86	67	65	59
humanoidmaze-large	task1	36	0	5	8	0	2	0	2	0	10	14	8	36	6	8	6
	task2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	task3	32	0	6	19	1	11	4	18	0	16	1	3	55	16	19	4
	task4	10	0	6	9	0	5	8	4	0	2	0	1	2	16	12	0
	task5	7	0	11	9	0	5	16	9	1	2	2	1	29	19	19	0
	agg. (5 tasks)	17	0	5	9	0	5	6	6	0	6	3	3	24	11	12	2
scene-sparse	task1	98	66	100	99	62	79	100	100	100	100	100	95	95	100	100	100
	task2	90	80	99	71	15	88	99	98	99	100	100	97	63	98	99	98
	task3	51	41	99	97	14	23	92	91	79	97	97	61	71	100	95	100
	task4	60	49	96	92	71	0	6	86	8	92	100	35	98	100	93	100
	task5	27	17	96	33	27	0	74	66	53	25	100	24	95	87	88	88
	agg. (5 tasks)	65	50	98	78	38	38	74	88	68	78	99	62	84	97	95	97
puzzle-3x3-sparse	task1	99	1	26	99	8	87	100	98	98	89	83	100	100	98	98	100
	task2	77	0	75	79	2	55	100	92	66	85	83	100	100	100	100	100
	task3	85	0	78	83	1	24	100	87	54	18	83	98	100	100	100	100
	task4	62	0	92	84	1	25	100	85	72	87	83	97	100	100	100	100
	task5	70	1	9	6	1	47	100	88	51	9	100	100	100	100	100	100
	agg. (5 tasks)	79	0	56	70	3	48	100	90	68	57	87	99	100	100	99	100
puzzle-4x4-100M-sparse	task1	0	29	0	17	0	56	0	0	0	0	0	71	0	0	20	85
	task2	0	13	0	1	0	12	0	0	0	0	1	13	1	0	5	7
	task3	0	16	0	3	0	41	0	0	0	0	0	39	0	0	3	56
	task4	0	6	0	2	0	11	0	0	0	0	0	14	0	0	4	29
	task5	0	13	0	2	0	3	0	0	0	0	0	32	0	0	2	19
	agg. (5 tasks)	0	15	0	5	0	24	0	0	0	0	0	34	0	0	6	39
cube-double	task1	29	0	84	81	8	55	50	62	36	80	90	77	16	85	84	89
	task2	6	0	49	46	0	39	46	52	35	29	87	28	12	79	84	79
	task3	2	0	38	42	0	44	50	52	31	33	85	44	10	54	59	55
	task4	1	0	8	10	0	13	11	18	16	5	32	14	4	22	18	22
	task5	4	0	56	50	0	42	48	41	56	16	76	39	11	82	81	83
	agg. (5 tasks)	9	0	47	46	2	38	41	45	35	33	74	40	11	64	65	65
cube-triple	task1	3	2	14	15	0	39	40	40	24	26	7	11	2	14	11	16
	task2	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	2
	task3	0	0	3	1	0	1	0	0	0	0	0	0	0	2	2	2
	task4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
	task5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	agg. (5 tasks)	1	0	3	3	0	8	8	8	5	6	1	2	0	3	3	5
cube-quadruple-100M	task1	35	0	0	11	0	0	2	2	12	90	9	19	8	13	67	32
	task2	6	0	0	0	0	0	0	0	1	0	1	2	0	0	1	0
	task3	3	0	0	0	0	0	0	0	0	0	1	2	2	1	3	0
	task4	0	0	0	0	0	0	0	0	0	4	0	1	0	0	0	0
	task5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	agg. (5 tasks)	9	0	0	2	0	0	1	0	3	19	2	5	2	3	14	6
all	agg. (50 tasks)	40	11	35	36	8	30	35	37	36	39	38	33	34	44	45	46

Table A.1: Full offline results at 1M training steps (12 seeds).

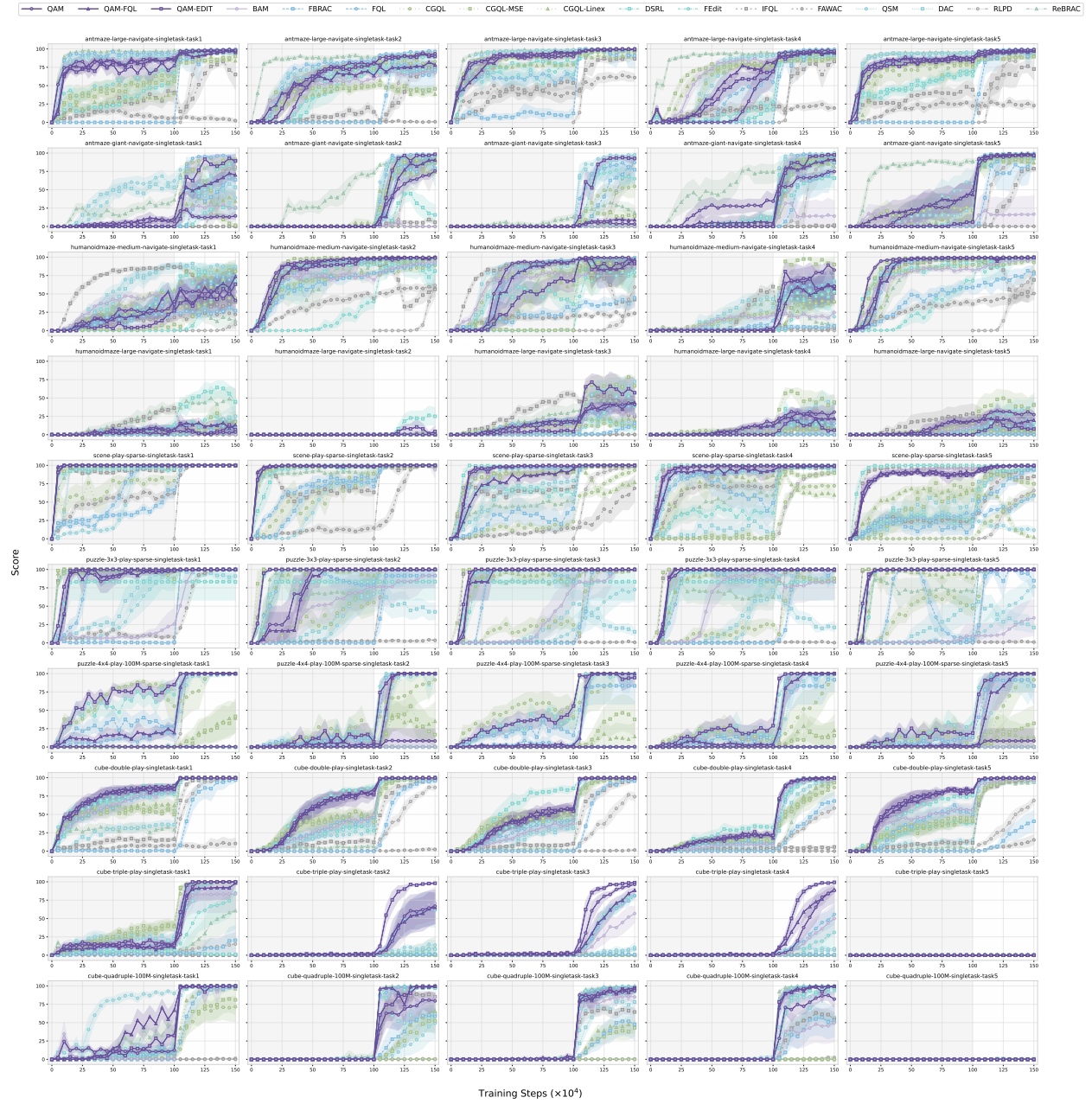


Figure A.1: Full training curves for our main results (12 seeds).

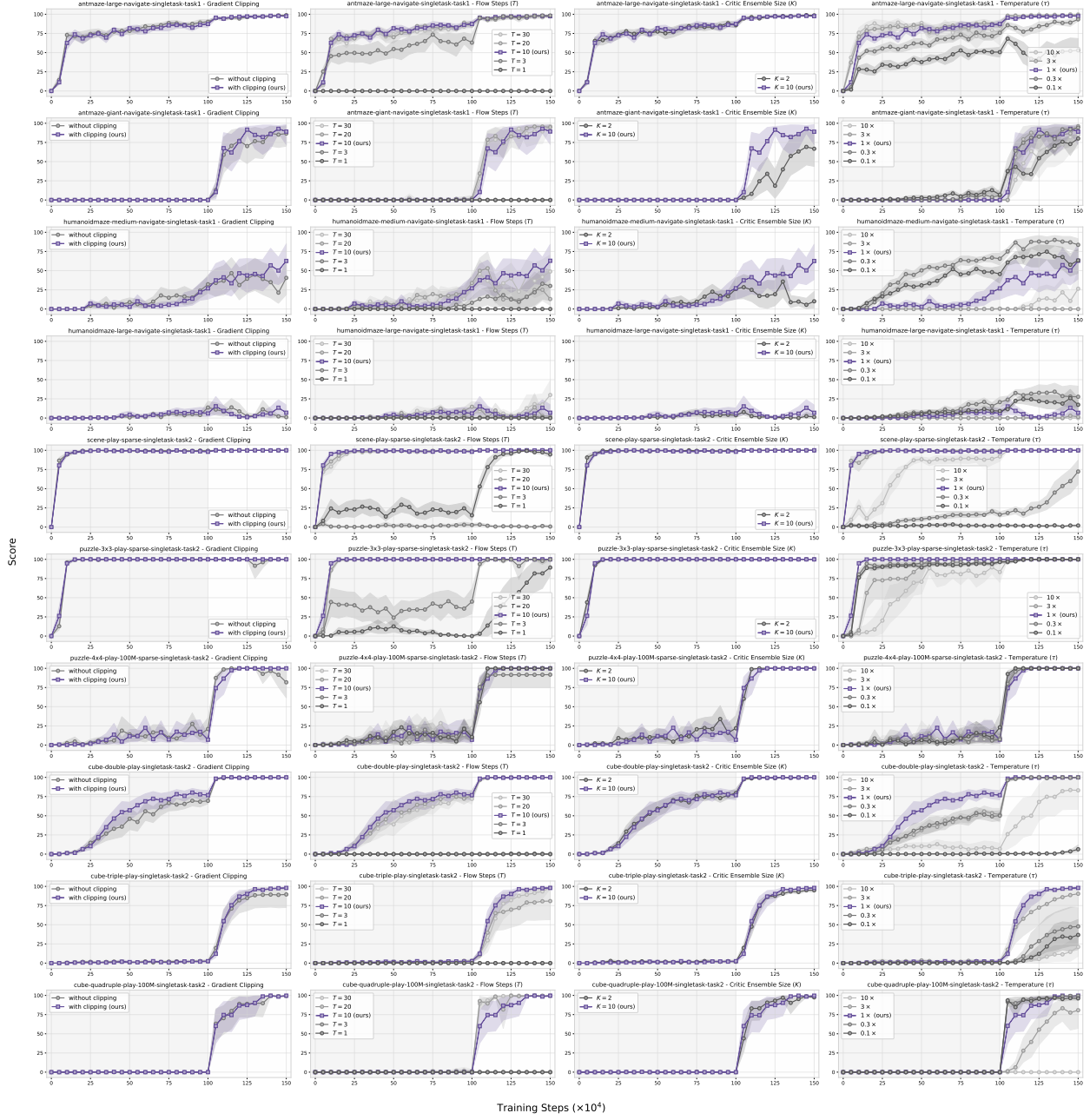


Figure A.2: Full sensitivity analysis for QAM-E (12 seeds).

Tasks	Dataset Size	Episode Length	Action Dimension (A)
cube-double-*	1M	500	5
cube-triple-*	3M	1000	5
cube-quadruple-100M-*	100M	1000	5
antmaze-large-*	1M	1000	8
antmaze-giant-*	1M	1000	8
humanoidmaze-medium-*	2M	2000	21
humanoidmaze-large-*	2M	2000	21
scene-sparse-*	1M	750	5
puzzle-3x3-sparse-*	1M	500	5
puzzle-4x4-100M-sparse-*	100M	500	5

Table A.2: Domain metadata.

Parameter	Value
Batch size	256
Discount factor (γ)	0.99 for {puzzle/scene/cube/antmaze-large}-* 0.995 for {humanoidmaze/antmaze-giant}-*
Optimizer	Adam
Learning rate	3×10^{-4}
Target network update rate (λ)	5×10^{-3}
Critic ensemble size (K)	10
Critic target pessimistic coefficient (ρ)	0.5 for {puzzle/scene/cube/antmaze}-* 0 for humanoidmaze-*
UTD ratio	1
Number of flow steps (T)	10
Number of offline training steps	10^6 ; except RLPD (0)
Number of online environment steps	0.5×10^6
Network width	512
Network depth	4 hidden layers
Optimizer gradient max norm clipping	1 for QSM, DAC, BAM and QAM*. No clipping for others.

Table A.3: Common hyperparameters.

Domains	RLPD	ReBRAC	FAWAC	QSM	DAC	FBRAC	FQL	DSRL	FEDit	IFQL	CGQL	CGQL-M	CGQL-L	BAM	QAM	QAM-E	QAM-F
	α	(α, σ)	τ	(τ, η)	η	α	α	σ_z	σ_a	τ	(θ, τ)	(θ, ϱ, τ)	(θ, ϱ, τ)	τ	τ	(τ, σ_a)	(τ, α)
scene-sparse-*	0.3	(0.03, 0)	6.4	(3, 30)	1	100	300	0.4	0.2	0.9	(10, 0.1)	(10, 0.1, 0.1)	(10, 0.1, 0.1)	3	1	(1, 0)	(1, 300)
puzzle-3x3-sparse-*	0.01	(0.1, 0)	0.8	(3, 30)	1	0.3	300	1	0.2	0.95	(10, 0.1)	(10, 0.1, 0.1)	(10, 0.001, 0.1)	10	3	(1, 0.1)	(3, ∞)
puzzle-4x4-100M-sparse-*	0	(0.01, 0.2)	0.8	(10, 1)	1	0.3	1	1	0.8	0.9	(10, 0.1)	(10, 0.001, 1)	(10, 0.001, 1)	30	30	(0.1, 0.9)	(3, 3)
cube-double-*	0.1	(0.01, 0)	0.8	(3, 10)	3	0.1	300	1	0.2	0.9	(10, 0.01)	(10, 0.001, 0.01)	(10, 0.001, 0.01)	3	1	(1, 0)	(1, ∞)
cube-triple-*	0	(0.01, 0.2)	0.8	(3, 3)	0.3	0.03	30	1.4	0.3	0.95	(10, 0.1)	(10, 0.001, 0.1)	(10, 0.001, 0.1)	30	3	(3, 0.1)	(10, 300)
cube-quadruple-100M-*	0	(0.01, 0.2)	0.8	(1, 10)	1	1	100	1.4	0.4	0.95	(10, 0.1)	(10, 0.1, 0.01)	(10, 0.1, 0.01)	10	1	(3, 0.1)	(0.3, 30)
antmaze-large-*	0	(0.01, 0)	6.4	(30, 30)	0.3	0.1	3	0.8	0.2	0.9	(10, 0.1)	(10, 0.1, 0.1)	(10, 0.001, 0.1)	30	10	(1, 0.1)	(3, 30)
antmaze-giant-*	0.01	(0.01, 0)	0.8	(30, 30)	0.3	0.1	3	1.2	0.3	0.8	(10, 0.1)	(10, 0.001, 0.1)	(10, 0.001, 0.1)	10	3	(10, 0.1)	(3, 30)
humanoidmaze-medium-*	0.03	(0.01, 0)	6.4	(10, 30)	1	30	30	0.6	0.5	0.7	(10, 0.01)	(10, 0.1, 0.1)	(10, 0.1, 0.1)	10	3	(3, 0.1)	(1, 30)
humanoidmaze-large-*	0	(0.01, 0.1)	0.8	(10, 30)	0.3	10	30	0.8	0.1	0.8	(10, 0.01)	(10, 0.1, 0.1)	(10, 0.1, 0.1)	10	3	(3, 0.1)	(0.3, 30)

Table A.4: **Domain-specific hyperparameters.** The best hyperparameter configuration obtained from our tuning runs. We use the same hyperparameter configuration for all tasks in each domain.

Method	Hyperparameter(s)	Sweep Range
RLPD	α	$\{0, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30\}$
ReBRAC	(α, σ)	$(\{0.01, 0.03, 0.1, 0.3, 1\}, \{0, 0.1, 0.2\})$
FBRAC	α	$\{0.03, 0.1, 0.3, 1.0, 3.0, 10.0, 30.0, 100.0\}$
CGQL	(ϑ, τ)	$(\{0.01, 0.1, 1\}, \{0.1, 1, 10\})$
CGQL- $\{M, L\}$	$(\vartheta, \tau, \varrho)$	$(\{0.01, 0.1, 1\}, \{0.1, 1, 10\}, \{0.001, 0.1\})$
FQL	α	$\{0.3, 1, 3, 10, 30, 100, 300, 1000\}$
DSRL	σ_z	$\{0.1, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4\}$
FEdit	σ_a	$\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$
FAWAC	τ	$\{0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 12.8\}$
IFQL	κ	$\{0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}$
DAC	η	$\{0.1, 0.3, 1, 3.0, 10, 30, 100, 300\}$
QSM	(τ, η)	$(\{1.0, 3.0, 10.0, 30\}, \{1, 3, 10, 30\})$
BAM	τ	$\{0.1, 0.3, 1, 3, 10, 30\}$
QAM	τ	$\{0.1, 0.3, 1, 3, 10, 30\}$
QAM-E	(τ, σ_a)	$(\{0.1, 0.3, 1, 3, 10\}, \{0, 0.1, 0.5, 0.9\})$
QAM-F	(τ, α)	$(\{0.1, 0.3, 1, 3, 10\}, \{3, 30, 300, \infty\})$

Table A.5: **Domain-specific hyperparameter tuning range.** For QAM-F, $\alpha = \infty$ is equivalent to the original QAM. Similarly, for QAM-E, $\sigma_a = 0$ is equivalent to the original QAM. For methods with more than one hyperparameter, we tune all possible combinations within the specified sweep ranges. For example, for CGQL, we sweep over all $3 \times 3 = 9$ configurations with 3 possible values of ϑ and 3 possible values of τ .

Method	Training Speed (milliseconds/step)	Parameter Count
ReBRAC	2.97	9 226 271
FBRAC	4.54	9 237 535
CGQL	9.62	9 242 655
CGQL- $\{M, L\}$	10.40	9 242 655
FQL	3.58	10 082 868
DSRL	5.63	18 474 580
FEdit	3.77	10 093 642
FAWAC	3.44	10 065 952
IFQL	2.66	10 065 952
DAC, QSM	4.44	10 320 447
BAM	5.52	10 083 380
QAM	5.83	10 083 380
QAM-E	6.61	10 939 487
QAM-F	6.36	10 928 713

Table A.6: Training speed and parameter count for each method on cube-triple.

Appendix B

Appendix for Reinforcement Learning with Action Chunking

B.1 Domain Details

We include the dataset size, episode length and the action dimension in Table B.1. In the following sections, we describe each domain in detail.

OGBench environments.

We consider five manipulation domains from OGBench [111] and take the publicly available single-task versions of it in our experiments. For `scene-sparse` and `puzzle-3x3-sparse`, we sparsify the reward function such that the reward values are -1 when the task is incomplete and 0 when the task is completed. For `cube-double/triple/quadruple`, the RL agent needs to command an UR-5 arm to pick and place two/three/four cubes to target locations. In particular, `cube-triple` and `cube-quadruple` are extremely difficult to solve from offline data only, and often achieve zero success rate. The RL agent must explore efficiently online in these domains to solve the tasks. The `cube-*` domains provide a great test ground for sample-efficiency of offline-to-online RL algorithms which we primarily focus on. For `cube-quadruple`, we use the 100M-size dataset. The dataset is too big to fit our CPU memory, so we periodically (after every 1000 gradient steps) load in a 1M-size chunk of the dataset for offline training. For online training of `RLPD`, `QC-RLPD`, we use the same strategy where we load in a 1M-size chunk of the dataset as the offline data and perform 50/50 sampling (e.g., 50% of the data comes from the 1M-chunk of the offline data, 50% of the data comes from the online replay buffer). For online fine-tuning of `QC-*`, `FQL`, `FQL-n`, `BFN`, and `BFN-n`, we keep a fixed 1M-size chunk of the offline dataset as the initialization of $\mathcal{D}$ and add new data to $\mathcal{D}$ directly. The remaining 99M transitions in the offline data are not being used online. We now describe each of the five domains in details:

Tasks	Dataset Size	Episode Length	Action Dimension (A)
scene-sparse-*	1M	750	5
puzzle-3x3-sparse-*	1M	500	5
cube-double-*	1M	500	5
cube-triple-*	3M	1000	5
cube-quadruple-100M-*	100M	1000	5
lift	31 127	500	7
can	62 756	500	7
square	80 731	500	7

Table B.1: **Domain metadata.** Dataset size (number of transitions), episode length, and the action dimension. For OGBench tasks, the action dimension is 5 (x position, y position, z position, gripper yaw and gripper opening). For robomimic tasks, the action dimension is 7 for square (3 for translation, 3 for rotation, and one final dimension for the gripper opening).

scene-sparse: This domain involves a drawer, a window, a cube and two button locks that control whether the drawer and the window can be opened. These tasks typically involve a sequence of actions. For example, **scene-task2** requires the robotic arm to unlock both locks, move the drawer and the window to the desired position, and then lock both locks. **scene-task4** requires the robotic arm to unlock the drawer, open the drawer, put the cube into the drawer, close the drawer. The reward is binary: -1 if the desired configuration is not yet reached and 0 if the desired configuration is reached (and the episode terminates).

puzzle-3x3-sparse: This domain contains a 3×3 grid of buttons. Each button has two states represented by its color (blue or red). Pressing any button causes its color and the color of all its adjacent buttons to flip (red $\rightarrow$ blue and blue $\rightarrow$ red). The goal is to achieve a pre-specified configuration of colors. **puzzle-3x3-task2** starts with all buttons to be blue, and the goal is to flip exactly one button (the top-left one) to be red. **puzzle-3x3-task4** starts with four buttons (top-center, bottom-center, left-center, right-center) to be blue, and the goal is to turn all the buttons to be blue. The reward is binary: -1 if the desired configuration is not yet reached and 0 if the desired configuration is reached (and the episode terminates).

cube-double/triple/quadruple: These three domains contain 2/3/4 cubes respectively. The tasks in the three domains all involve moving the cubes to their desired locations. The reward is $-n_{\text{wrong}}$ where n_{wrong} is the number of the cubes that are at the wrong position. The episode terminates when all cubes are at the correct position (reward is 0).

Robomimic environments.

We use three challenging tasks from the robomimic domain [91]. We use the multi-human datasets that were collected by six human operators. Each dataset contains 300 successful trajectories. The three tasks are as described as follows.

- **lift**: This task requires the robot arm to pick a small cube. This is the simplest task of the benchmark.
- **can**: This task requires the robot arm to pick up a coke can and place in a smaller container bin.
- **square**: This task requires the robot arm to pick a square nut and place it on a rod. The nut is slightly bigger than the rod and requires the arm to move precisely to complete the task successfully.

All of the three robomimic tasks use binary task completion rewards where the agent receives -1 reward when the task is not completed and 0 reward when the task is completed.

B.2 Implementation Details

In this section, we provide more implementation details on both of our Q-chunking methods (QC, QC-FQL) and our baselines used in our experiments. For IFQL, ReBRAC, IQL, we directly use the implementation from [108].

Algorithm 4 QC

Input: Behavior policy and critic:
 $f_\xi(\mathbf{a}_{t:t+h}|s)$ and $Q_\theta(s_t, \mathbf{a}_{t:t+h})$.
 $\mathcal{D} \leftarrow$ offline prior data.
for every environment step t **do**
 if $t \bmod h \equiv 0$ **then**
 $\mathbf{a}_{t:t+h}^1 \cdots \mathbf{a}_{t:t+h}^N \sim f_\xi(\cdot|s_t)$
 $\mathbf{a}_{t:t+h}^* \leftarrow \arg \max_{\mathbf{a}_{t:t+h}^i} Q_\theta(s, \mathbf{a}_{t:t+h}^i)$
 Act with $\mathbf{a}_t^*$ and receive s_{t+1}, r_t .
 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, \mathbf{a}_t^*, s_{t+1}, r_t)\}$
 Update f_ξ via flow-matching loss using $\mathcal{D}$.
 Update Q_θ via Eq (4.31) using $\mathcal{D}$.
Output: f_ξ, Q_θ .

Algorithm 5 QC-FQL

Input: Behavior policy, critic, one-step policy:
 $f_\xi(\mathbf{a}_{t:t+h}|s)$, $Q_\theta(s_t, \mathbf{a}_{t:t+h})$, $\mu_\psi(s, \mathbf{z})$.
 $\mathcal{D} \leftarrow$ offline prior data.
for every environment step t **do**
 if $t \bmod h \equiv 0$ **then**
 $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_{Ah})$
 $\mathbf{a}_{t:t+h} \leftarrow \mu_\psi(s_t, \mathbf{z})$
 Act with $\mathbf{a}_t$ and receive s_{t+1}, r_t .
 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, \mathbf{a}_t, s_{t+1}, r_t)\}$
 Update f_ξ via flow-matching loss using $\mathcal{D}$.
 Update μ_ψ and Q_θ via Eq. (4.33, 4.34) using $\mathcal{D}$.
Output: $f_\xi, Q_\theta, \mu_\psi(s, \mathbf{z})$.

QC-FQL

We build the implementation of our method on top of **FQL** [108], a recently proposed offline RL/offline-to-online RL method that uses a TD3+BC-style objective. It is implemented with a one-step noise-conditioned policy (instead of a Gaussian policy that is commonly used in RL) and it uses a distillation loss from a behavior flow-matching policy as the BC loss. To adapt this method to use action chunking, we simply apply **FQL** on the temporally extended action space – the behavior flow-matching policy generates a sequence of actions, the one-step noise-conditioned policy predicts a sequence of actions, and the Q -network also takes in a state and a sequence of actions. More concretely, we train three networks:

1. $Q_\theta(s, a_1, \dots, a_h) : \mathcal{S} \times \mathcal{A}^h \mapsto \mathbb{R}$ — the value function that takes in a state and a sequence of actions (action chunk). In practice, we train an ensemble of Q networks. We denote the weight of the ensemble element as $\theta = (\theta_1, \dots, \theta_K)$.
2. $\mu_\psi(s, z) : \mathcal{S} \times \mathbb{R}^{A_h} \mapsto \mathbb{R}^{A_h}$ — the one-step noise-conditioned policy that takes in a state and a noise, and outputs a sequence of actions conditioned on them.
3. $f_\xi(s, m, u) : \mathcal{S} \times \mathbb{R}^{A_h} \times [0, 1] \mapsto \mathbb{R}^{A_h}$ — the flow-matching behavior policy parameterized by a velocity prediction network. The network predicts takes in a state, an intermediate state of the flow and a time, and outputs the velocity direction that the intermediate action sequence should move in at the specified time. See Algorithm 6 for more details on how this velocity prediction network is used to generate an action from a noise vector.

We denote our policy as $\pi_\psi(\cdot|s)$. It is implemented by first sampling a Gaussian noise $z \sim \mathcal{N}(0, I_{A_h})$ and run it through the one-step noise-conditioned policy $[a_1 \dots a_h] \leftarrow \mu_\psi(s, z)$. To train these three networks, we sample a high-level transition,

$$w = (s_t, a_t, a_{t+1}, \dots, a_{t+h-1}, s_{t+h}, r_t^h) \sim \mathcal{D},$$

with $r_t^h = \sum_{t'=0}^{h-1} \gamma^{t'} r_{t+t'}$, to construct the following losses:

(1) Critic loss:

$$L(\theta_k, w) = \left(Q_{\theta_k}(s_t, a_t, \dots, a_{t+h-1}) - R_{t:t+h} - \frac{\gamma^h}{K} \sum_{k'=1}^K Q_{\bar{\theta}_{k'}}(s_{t+h}, a_{t+h}, \dots, a_{t+2h-1}) \right)^2, \quad (\text{B.1})$$

where $[a_{t+h} \dots a_{t+2h-1}] \sim \pi_\psi(\cdot|s_{t+h}), k \in \{1, 2, \dots, K\}$.

(2) Actor loss:

$$L(\psi, w) = -Q_\theta(s_t, \mu_\psi(s_t, z_t)) + \alpha \left\| \mu_\psi(s_t, z_t) - [a_t^\xi \dots a_{t+h-1}^\xi] \right\|_2^2, \quad (\text{B.2})$$

where $z_t \sim \mathcal{N}(0, I_{Ah})$ and $[a_t^\xi \cdots a_{t+h-1}^\xi]$ is the result of running the behavior policy $f_\xi(s, m, t)$ with Algorithm 6 from z_t , and $\alpha \in \mathbb{R}$ is a tunable parameter that controls the strength of the behavior regularization (higher α leads to stronger behavior regularization).

(3) Flow-matching behavior policy loss:

$$L(\xi, w) = \|f_\xi(s_t, u[a_t \cdots a_{t+h-1}] + (1-u)z_t, u) - ([a_t \cdots a_{t+h-1}] - z_t)\|_2^2, \quad (\text{B.3})$$

where $u \sim U([0, 1])$, $z_t \sim \mathcal{N}(0, I_{Ah})$.

Practically, we sample a batch of transitions $\{w_1, w_2, \dots, w_M\}$ and optimize the average loss for each network: $\mathcal{L}(\theta) = \frac{1}{M} \sum_{i=1}^M \sum_{k=1}^K \mathcal{L}(\theta_k, w_i)$, $\mathcal{L}(\psi) = \frac{1}{N} \sum_{i=1}^M \mathcal{L}(\psi, w_i)$, $\mathcal{L}(\xi) = \frac{1}{N} \sum_{i=1}^M \mathcal{L}(\xi, w_i)$.

FQL

FQL [108] is a recently proposed offline RL/offline-to-online RL method that uses TD3+BC-style objective. It is equivalent to our method with $h = 1$. For completeness, we write out the objectives for a transition sample $w = (s_t, a_t, r_t, s_{t+1})$:

$$L(\theta_k, w) = \left(Q_{\theta_k}(s_t, a_t) - r_t - \frac{\gamma}{K} \sum_{k'=1}^K Q_{\bar{\theta}_{k'}}(s_{t+1}, \mu_\psi(s_{t+1}, z_t^{k'})) \right)^2, \quad z_t^{k'} \sim \mathcal{N}(0, I_A), \quad (\text{B.4})$$

$$L(\psi, w) = -Q_\theta(s_t, \mu_\psi(s_t, z_t)) + \alpha \left\| \mu_\psi(s_t, z_t) - a_t^\xi \right\|_2^2, \quad (\text{B.5})$$

$$a_t^\xi \leftarrow \text{FlowODE_Euler}(s_t, z_t, f_\xi, T), \quad z_t \sim \mathcal{N}(0, I_A), \quad (\text{B.6})$$

$$L(\xi, w) = \|f_\xi(s_t, ua_t + (1-u)z_t, u) - (a_t - z_t)\|_2^2, \quad z_t \sim \mathcal{N}(0, I_A), \quad u \sim U([0, 1]). \quad (\text{B.7})$$

In practice, we sample a batch of transitions $\{w_1, w_2, \dots, w_N\}$ and optimize the average loss for each network: $\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K \mathcal{L}(\theta_k, w_i)$, $\mathcal{L}(\psi) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(\psi, w_i)$, $\mathcal{L}(\xi) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(\xi, w_i)$.

FQL-n

To implement the n -step return baseline, we take **FQL** and replace the 1-step TD update with the h -step TD update:

$$L(\theta_k, w) = \left(Q_{\theta_k}(s_t, a_t) - \sum_{t'=0}^{h-1} (\gamma^{t'} r_{t+t'}) - \frac{\gamma^h}{K} \sum_{k'=1}^K Q_{\bar{\theta}_{k'}}(s_{t+h}, \mu_\psi(s_{t+h}, z_t^{k'})) \right)^2, \quad (\text{B.8})$$

where $z_t^{k'} \sim \mathcal{N}(0, I_A)$ for all $k' \in \{1, 2, \dots, K\}$. The actor loss and flow-matching loss remain the same as FQL.

Algorithm 6 `FlowODE_Euler`(s_t, z_t, f_ξ, T): generate actions from the behavior flow policy $f_\xi(s, m, u)$ with Euler’s method.

Input: State s_t , noise z_t and flow model $f_\xi(s, m, u)$, number of flow steps T .

$m^0 \leftarrow z_t$

for $i \in \{1, \dots, T\}$ **do**

$m^i \leftarrow f_\xi(s_t, m^{i-1}, (i-1)/T)$

Output: m^T .

QC

The flow-matching behavior policy is trained with the same loss as used in **QC-FQL** (Equation (B.3)). On top of the flow-matching behavior policy, we simply parameterize the policy π implicitly by sampling multiple action chunks from the behavior policy and pick the one that maximizes the Q -value. Specifically, let $\{z_t^1, z_t^2, \dots, z_t^N\} \sim \mathcal{N}(0, I_{Ah})$ and

$$[a_t^i \ \dots \ a_{t+h-1}^i] = \text{FlowODE_Euler}(s_t, z_t^i, f_\xi, T)$$

The policy outputs the one action chunk out of N that maximizes the Q -value, $[a_t^{i^*} \ \dots \ a_{t+h-1}^{i^*}]$, where

$$i^* \leftarrow \arg \max_{i \in [N]} Q(s, [a_t^i \ \dots \ a_{t+h-1}^i]).$$

Finally, we directly use this implicitly parameterize policy to generate actions for computing the TD target for our TD loss:

$$L(\theta_k, w) = \left(Q_{\theta_k}(s_t, a_t, \dots, a_{t+h-1}) - R_{t:t+h} - \frac{\gamma^h}{K} \sum_{k'=1}^K Q_{\bar{\theta}_{k'}}(s_{t+h}, a_{t+h}^{i^*}, \dots, a_{t+2h-1}^{i^*}) \right)^2 \quad (\text{B.9})$$

where $a_{t+h}^{i^*}, \dots, a_{t+2h-1}^{i^*} \sim \pi(\cdot | s_{t+h})$.

The baselines **BFN-n** and **BFN** are implemented similarly to **FQL-n** and **FQL** by operating in the original action space.

RLPD, RLPD-AC, QC-RLPD

All the **RLPD** baseline results are obtained by running the official codebase (as linked in [9]) with additional modification to incorporate action chunking and behavior cloning. This baseline runs online RL from scratch using off-policy transitions where 50% of them come from the offline dataset and the other 50% come from the online replay buffer. It essentially up-weights the online data more, allowing the online RL agent to learn more quickly. This is different from how **QC-***, **BFN**, **BFN-n**, **FQL**, **FQL-n** samples off-policy transitions

(where we sample from the dataset that combines the offline dataset and online replay buffer data with no weighting). **RLPD** baselines all use Gaussian policy. This is also different from our method as our method uses noise-conditioned policy that can represent a wider range of distributions. For **RLPD-AC**, we change all the actor and critic networks such that they work with an action chunk rather than a single action. The baseline is exactly the same as our method except that actor and the critic are updated the same as how **RLPD** updates its actor and critic. For **QC-RLPD**, we add a behavior cloning loss in the actor loss as follows (highlighted in red below):

$$\mathcal{L}(\psi) = \mathbb{E}_{a'_t \sim \pi_\psi(\cdot|s_t)} \left[-\frac{1}{K} \sum_{k=1}^K Q_{\theta_k}(s_t, a'_t) - \alpha \log \pi_\psi(a_t|s_t) \right]. \quad (\text{B.10})$$

SUPE-GT

This baseline is based on a recent offline-to-online RL method [156]. The original method has additional modules such as reward models and random distillation networks to deal with the problem setting where the offline dataset is unlabeled (*e.g.*, reward information is missing). We remove these two modules as we have ground truth reward information in our offline dataset. The method has two stages: (1) skill-pretraining and (2) online RL with skills. In the first stage, the baseline uses a trajectory VAE to learn a latent-conditioned skill policy. In the second stage, the baseline uses RLPD to learn a high-level policy that outputs a latent vector to command the latent-conditioned skill policy after every h steps. Such a skill-based method in theory also allows temporally coherent actions for exploration and value backup speedup.

B.3 Experiment Details

Computational resources

We use NVIDIA RTX-A5000 GPU to run all our experiments. Each complete offline-to-online experiment run takes around 4-7 hours. To reproduce all our results in Table B.5, we estimate that it would take around $\underbrace{6}_{\text{hours per single run}} \times \underbrace{15}_{\text{\# of methods}} \times \underbrace{25}_{\text{\# of tasks}} \times \underbrace{4}_{\text{\# of seeds}} = 9\,000$ GPU hours. Each robomimic experiment run takes longer (around 8-12 hours). While each single run takes longer, we only run a subset of strong methods/baselines on robomimic tasks. We estimate it would take around $\underbrace{10}_{\text{hours per single run}} \times \underbrace{8}_{\text{\# of methods}} \times \underbrace{3}_{\text{\# of tasks}} \times \underbrace{5}_{\text{\# of seeds}} = 1\,200$ GPU hours. In total, we estimate that it would take around 10 200 GPU hours to reproduce all the main results in our paper.

Evaluation protocol

Unless specified otherwise, for all methods, we run 4 seeds on each OGBench task and 5 seeds on each robomimic task. All plots use 95% confidence interval with stratified sampling (5000 samples). The success rate is computed by running the policy in the environment for 50 episodes and record the number of times that the policy succeeds at solving the task (and divide it by 50).

Hyperparameter tuning

Parameter	Value
Batch size (M)	256
Discount factor (γ)	0.99
Optimizer	Adam
Learning rate	3×10^{-4}
Target network update rate (τ)	5×10^{-3}
Critic ensemble size (K)	10 for RLPD , RLPD-AC , QC-RLPD , and SUPE-GT 2 for QC-FQL , FQL , FQL-n , QC-BFN , BFN , BFN-n
UTD Ratio	1
Number of flow steps (T)	10
Number of offline training steps	10^6 except RLPD -based approaches (0)
Number of online environment steps	1×10^6
Network width	512
Network depth	4 hidden layers

Table B.2: Common hyperparameters.

Environments	FQL	FQL-n	QC-FQL	ReBRAC
scene-sparse-*	300	100	300	0.1
puzzle-3x3-sparse-*	100	100	300	0.1
cube-double-*	300	100	300	0.1
cube-triple-*	300	100	100	0.1
cube-quadruple-100M-*	300	100	100	0.1
lift	10000	10000	10000	-
can	10000	10000	10000	-
square	10000	10000	10000	-

Table B.3: Behavior regularization coefficient (α).

QC, **BFN**, **BFN-n**. We tune the number of actions sampled, N , for the expected-max Q operator. On OGBench domains, we sweep over $\{2, 4, 8, 16, 32, 64, 128\}$ and select the best parameter for each domain and for each method on **task2**. We report the performance of

Environments	BFN	BFN-n	QC-BFN
scene-sparse*	4	4	32
puzzle-3x3-sparse-*	4	4	64
cube-*	4	4	32
lift, can, square	4	4	16

Table B.4: Number of actions sampled for the expected-max Q operator (N) for BFN methods.

each method with the best α in Table 4.2 (on all tasks). Table B.4 summarizes the α value we use for each task.

QC-FQL, FQL, FQL-n. We tune the behavior regularization coefficient α . On OGBench domains, we take the default hyperparameter of **FQL** for each domain α_{default} and tune all methods on **task2** of each domain with three choices of α : $\{\alpha_{\text{default}}/3, \alpha_{\text{default}}, 3\alpha_{\text{default}}\}$ (our α_{default} comes from Table 6 in [108]). On robomimic domain, we sweep over much larger α values: $\{100, 1000, 10000\}$. We report the performance of each method with the best α in Table 4.2 (on all tasks). Table B.3 summarizes the α value we use for each task.

RLPD, RLPD-AC, QC-RLPD. We sweep over (1) whether or not to use clipped double Q-learning (CDQ), and (2) whether or not to use entropy backup. We find that not using CDQ and not using entropy backup performs the best for all of the RLPD baselines and use that across all domains. Even though our method and the other FQL baselines use $K = 2$ critic ensemble size, we use $K = 10$ critic ensemble size for **RLPD** to keep it the same as the hyperparameter in the original paper [9]. For **QC-RLPD**, we sweep over behavior regularization coefficient $\alpha \in \{0.001, 0.01, 0.1\}$ and pick 0.01 since it works the best.

SUPE-GT. We tune the KL coefficient for the VAE skill-pretraining from a set of five values: $\{0.001, 0.003, 0.01, 0.03, 0.1\}$. We pick 0.003 as it works the best.

ReBRAC. We tune the behavior regularization coefficient from $\{0.01, 0.03, 0.1, 0.3, 1.0\}$ and pick 0.1 as it works the best.

IFQL. We directly use the default hyperparameter from [108] ($N = 32, \tau = 0.9$) for this baseline. The first parameter, $N = 32$, means that for both online policy rollout and policy evaluation, we sample 32 actions and pick the action that has the highest Q-value. The second parameter, $\tau = 0.9$ is the expectile coefficient.

IQL. We directly use the default hyperparameter from [108] for this baseline. For **cube-***, we use $\alpha = 0.3$. For **scene** and **puzzle-3x3**, we use $\alpha = 10.0$. For the expectile coefficient, we use $\tau = 0.9$.

B.4 Full Results

End-effector visualization

We provide more examples of the trajectory rollouts from **QC** and **BFN** over the course of online training on **cube-triple-task3**. In Figure B.1, we show the first 9000 time steps (broken down into 9 subplots where each visualizes 1000 time steps). In Figure B.2, we show another 9000 time steps but late in the training (from environment step 9×10^5). The first example is the same as the one used in Figure 4.4.

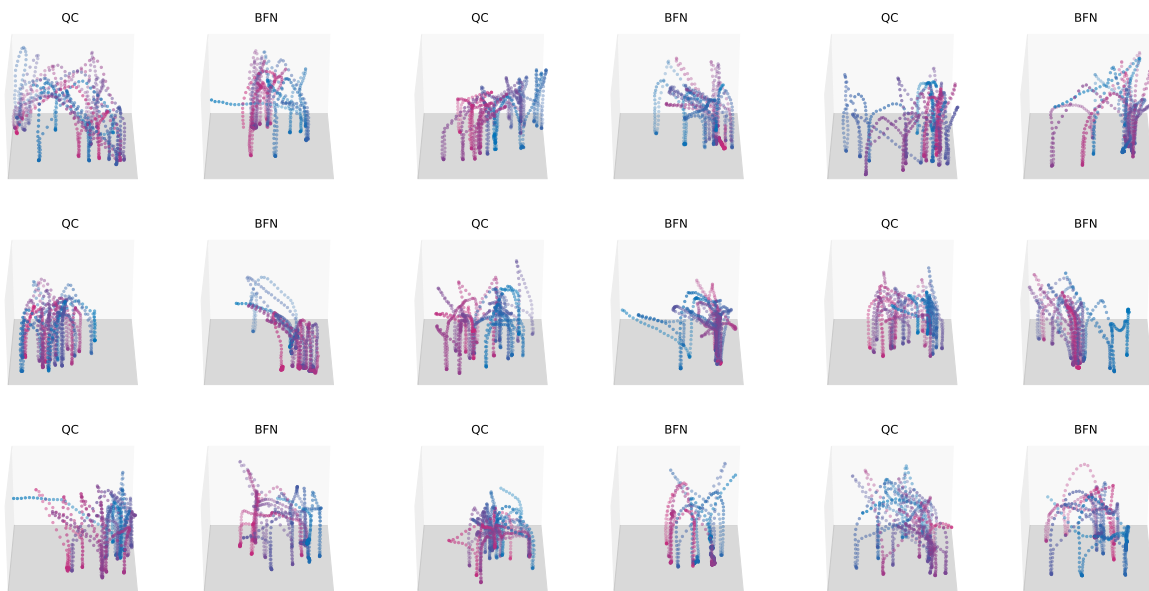


Figure B.1: **End-effector trajectory early in the training.** Each subplot above shows the trajectory for 1000 consecutive time steps. We include up to Step 9000.

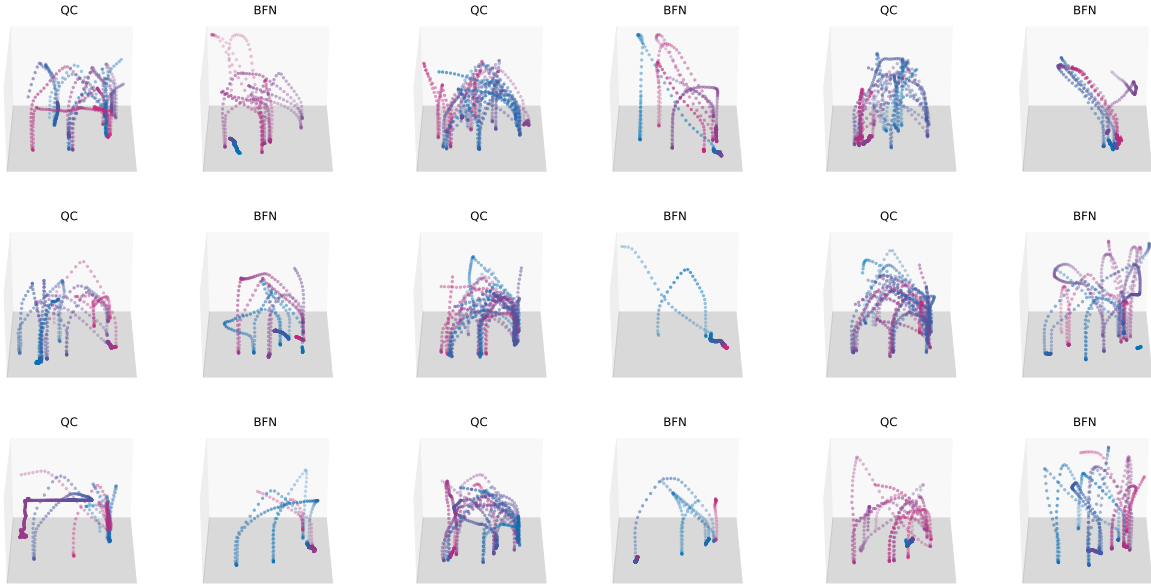


Figure B.2: **End-effector trajectory visualization late in the training.** Each subplot above shows the trajectory for 1000 consecutive time steps (from Step 900000 to Step 909000).

OGBench results by individual task

Main results by task. Table 4.2 and Figure B.3 show the performance breakdown for all methods.

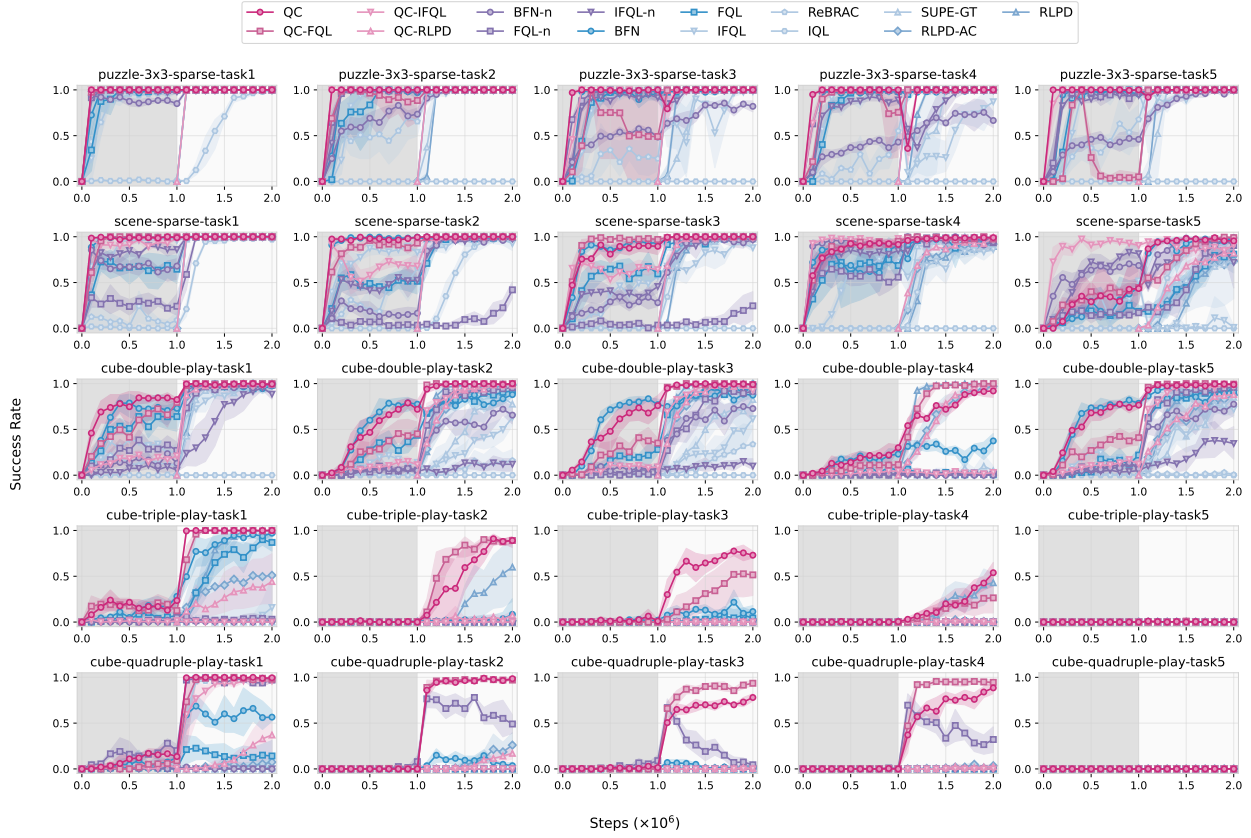


Figure B.3: Complete OGBench offline-to-online RL results by task. 4 seeds. 95% confidence interval.

Selected baselines. Figure B.4 shows the results for selected baselines (*e.g.*, BFN, FQL, RLPD and RLPD-AC).

n -step return ablation results by task. Figure B.5 shows the performance breakdown for each baseline on each OGBench task.

Q-chunking with Gaussian policies. The following plot shows the performance of RLPD with and without action chunking. In addition, we include a new method for comparison, **QC-RLPD**, where we add a behavior cloning loss to **RLPD-AC** (**RLPD** with action chunking).

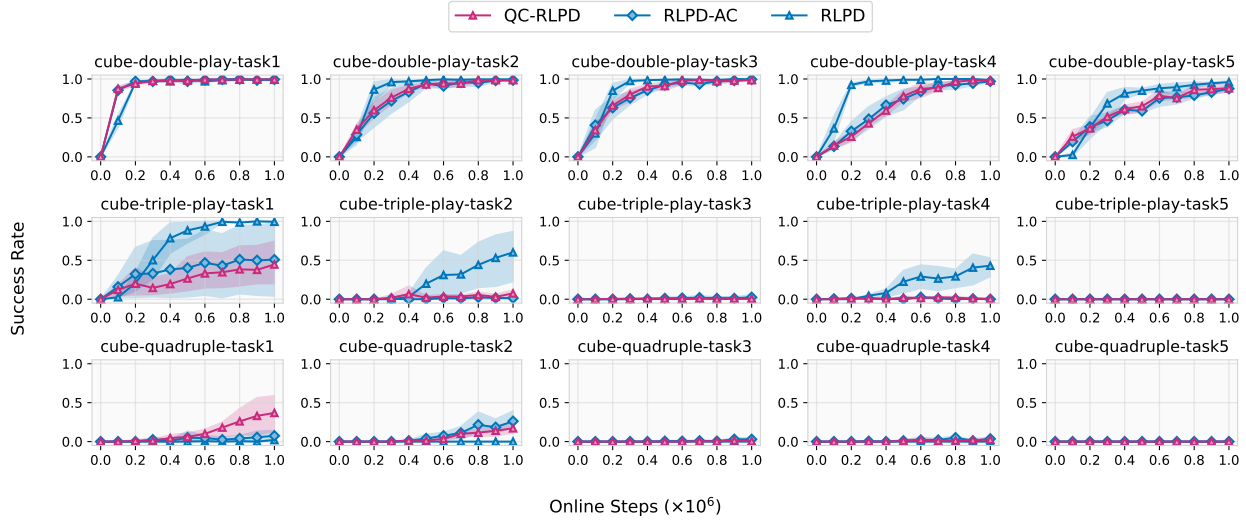


Figure B.6: **Full RLPD results by task.** **QC-RLPD** is **RLPD-AC** (**RLPD** on the temporally extended action space) where we additionally add a fixed behavior cloning coefficient of 0.01.

Action chunking size ablations. Figure B.7 includes individual task results for the action chunking size ablation study. We also include results on the hardest cube-quadruple task in Figure B.8.

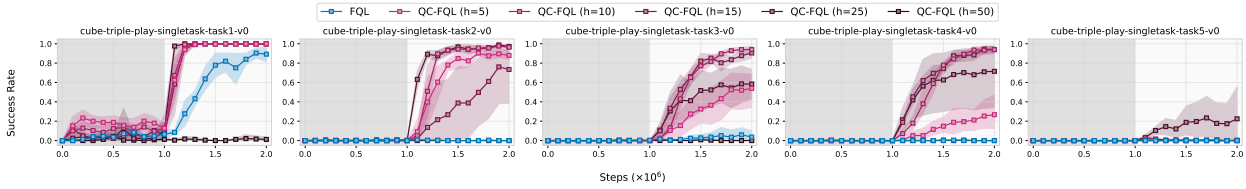


Figure B.7: **Action chunking size ablation on cube-triple-***. Increasing the action chunking size generally helps until $h = 25$. Surprisingly, $h = 25$ is the only chunk size where **QC-FQL** achieves non-trivial success on the hardest cube-triple-play-task5 (none of other methods can).

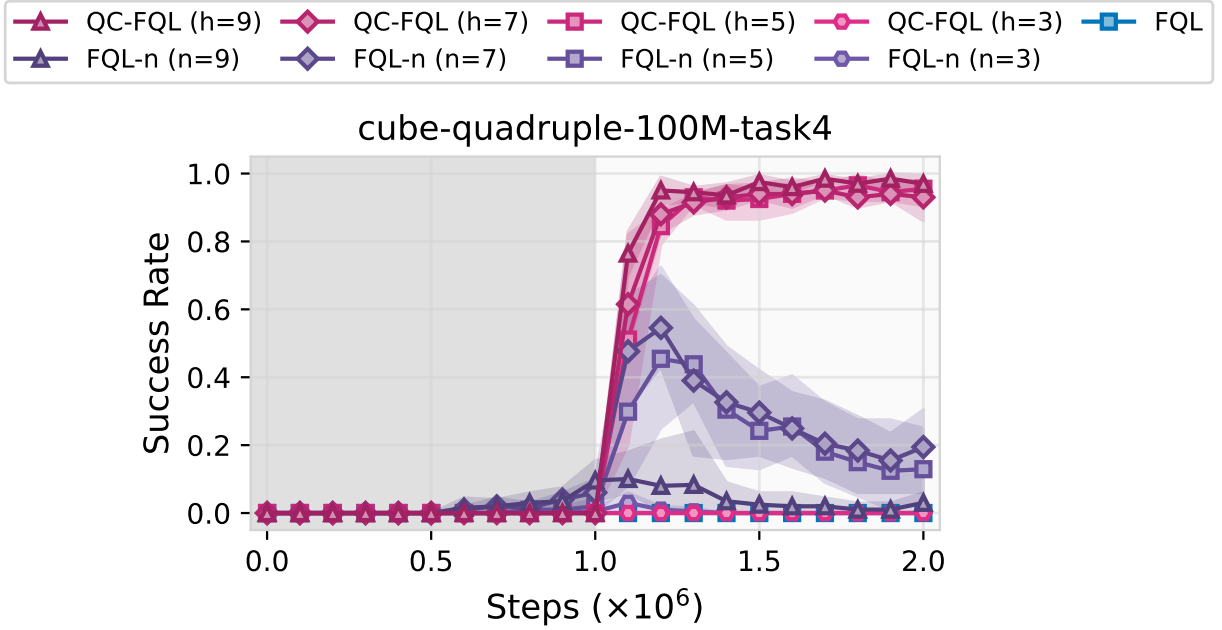


Figure B.8: **Action chunking size ablation on cube-quadruple-play-task4.** The n -step return baselines (*i.e.*, **FQL-n**) can achieve good initial success during online learning but quickly collapse. In contrast, our method (*i.e.*, **QC-FQL**) does not suffer from such collapse and solves the task consistently across reasonably large chunk sizes (*e.g.*, $h \in \{5, 7, 9\}$). The results are averaged over 5 seeds.

Robomimic ablation results

Figure B.9 shows the performance of **QC**, **QC-FQL**, **BFN-n**, **FQL-n**, **BFN**, **FQL** on three robomimic tasks.

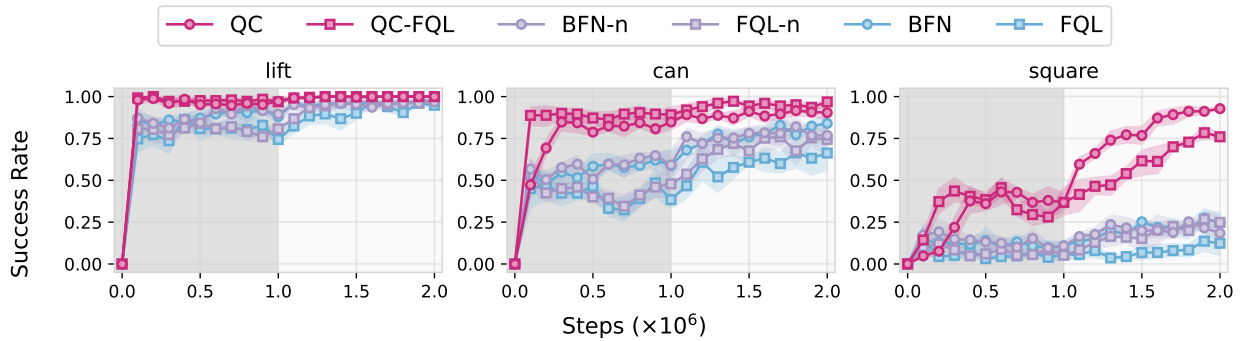


Figure B.9: **Full robomimic ablation by task.** We use 5 seeds for all experiments.

How resource efficient is Q-chunking?

In Figure B.10, we report the runtime for our approach and our baselines on a representative task `cube-triple-task1`. In general, **QC-FQL** has a comparable run-time as our baselines (e.g., **FQL** and **RLPD**) for both offline and online. **QC** is slower for offline training as it requires sampling 32 actions for each training example for the agent update (**BFN** is faster because it only needs to sample 4 actions). For online training, we are doing one gradient update per environment step, and it makes **QC** only around 50% more expensive than other methods.

Finally, we include the parameter count of each of these method on the representative domain `cube-triple-*` in Table B.6 below (assuming $h = 5$ for all Q-chunking methods).

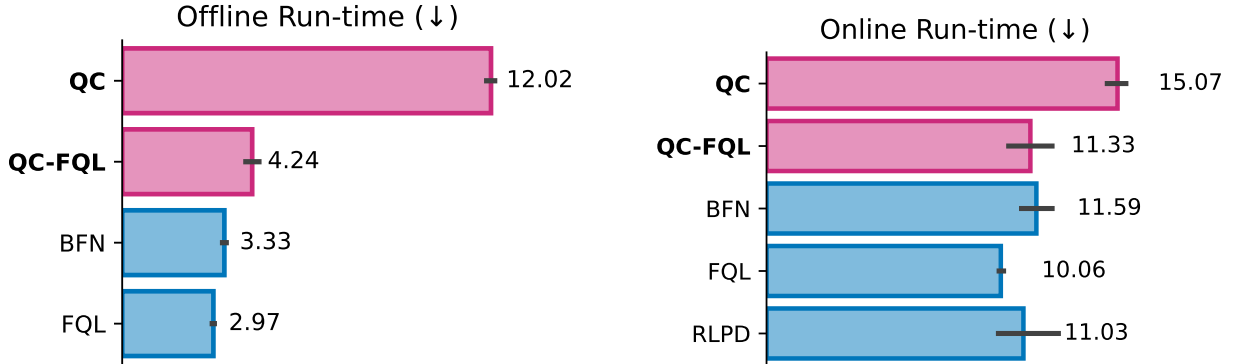


Figure B.10: **How long each method takes for one step, in milliseconds.** Left: offline. Right: online (one agent training step and an environment step). The runtime is measured using the default hyperparameters from our paper on `cube-triple-task1` on a single RTX-A5000.

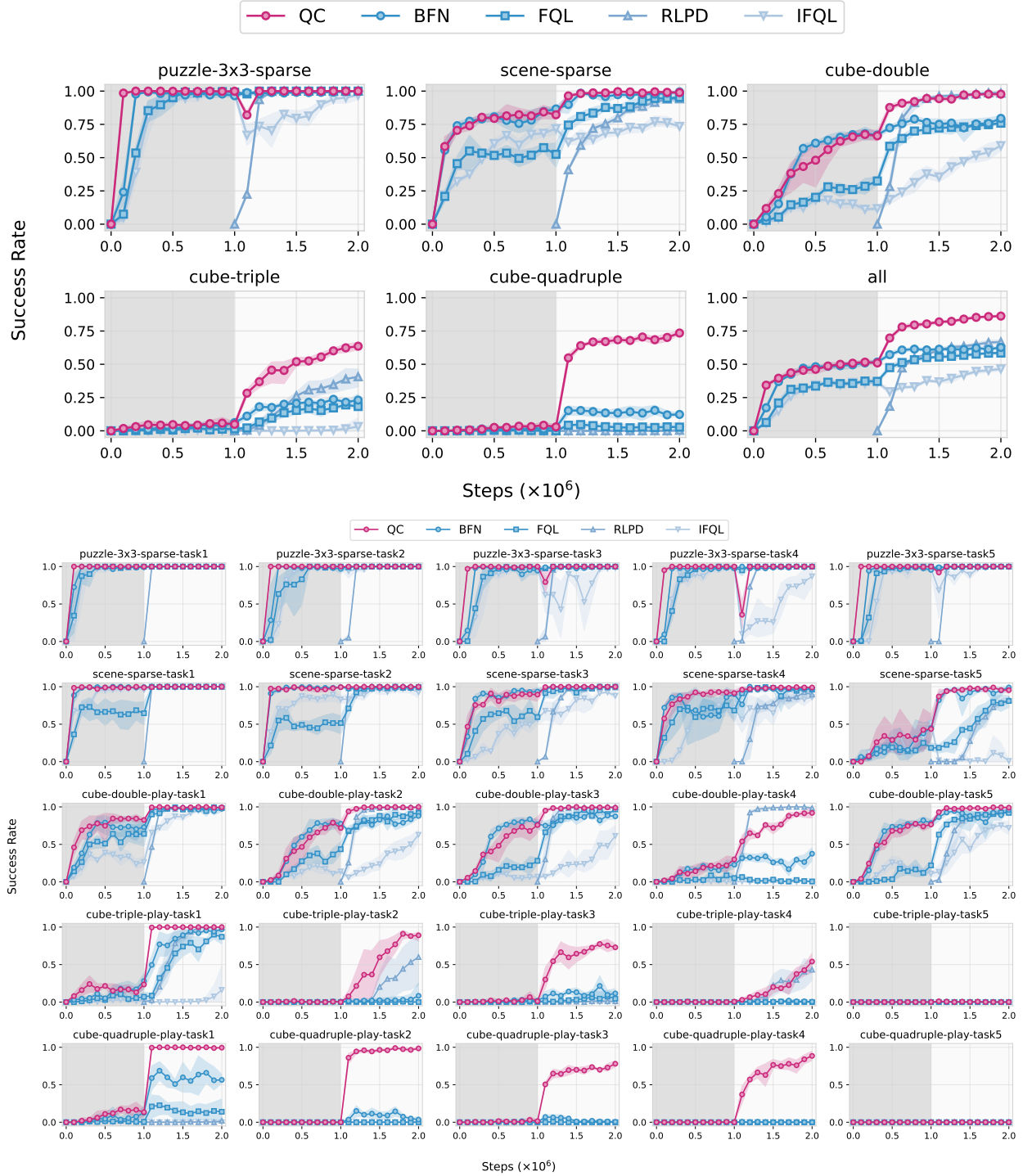


Figure B.4: **Full OGBench results by task (selected baselines).** *Top:* summary plots by domain; *Bottom:* individual plots by task.

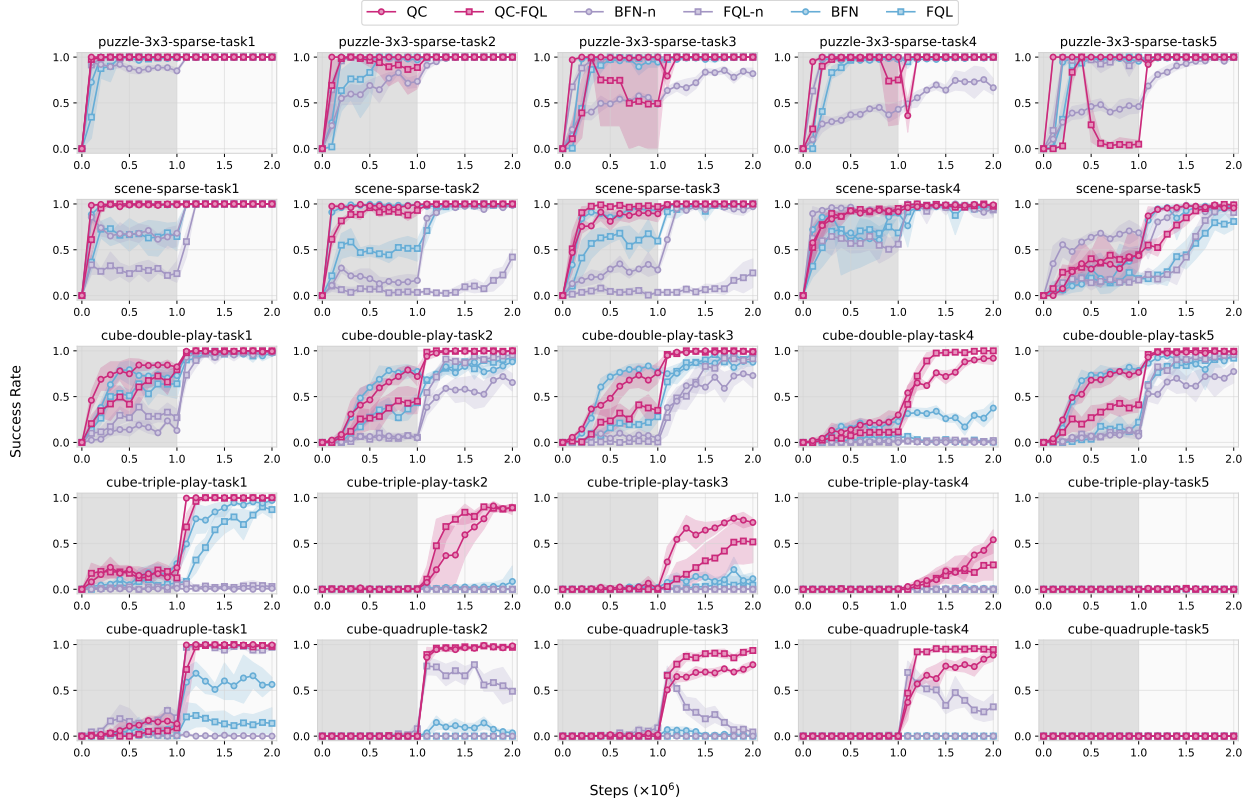


Figure B.5: Full OGBench results by task (selected baselines).

Methods	Parameter Count (in millions)
QC	≈ 4.2
BFN	≈ 4.1
QC-FQL	≈ 5.0
FQL	≈ 4.9
RLPD	≈ 17.2

Table B.6: **Parameter count for each method.** RLPD has a much larger parameter count because it uses $K = 10$ critic networks whereas all the other methods and baselines use only $K = 2$.

Appendix C

Appendix for Decoupled Q-Chunking

C.1 Full results

Table C.1 reports the performance of our method (DQC) and baselines for all hyperparameter configurations. All of them use the same hyperparameters in Table C.3 with the only exception that SHARSA, HIQL, IQL, FBC, and HFBC handle goal-sampling for training behavior cloning policies differently. We discuss this in more details in Section C.3.

Method			c3-100M	c4-100M	c8-1B	hg	p45	p46-1B
DQC	$h = 25$	$h_a = 1$	76 _[73,80]	45 _[41,49]	10 _[8,11]	92 _[90,94]	91 _[89,92]	83 _[80,86]
DQC	$h = 25$	$h_a = 5$	98 _[98,99]	92 _[90,93]	34 _[33,35]	51 _[48,54]	96 _[95,97]	68 _[66,71]
DQC	$h = 5$	$h_a = 1$	95 _[94,97]	84 _[83,86]	0 _[0,0]	19 _[15,22]	90 _[88,92]	44 _[42,47]
DQC-naïve	$h = 25$	$h_a = 1$	14 _[8,22]	16 _[9,23]	1 _[0,2]	22 _[20,24]	32 _[28,36]	33 _[29,37]
DQC-naïve	$h = 25$	$h_a = 5$	27 _[18,38]	27 _[15,39]	3 _[1,5]	0 _[0,1]	33 _[29,37]	33 _[28,38]
DQC-naïve	$h = 5$	$h_a = 1$	16 _[7,30]	40 _[29,49]	0 _[0,0]	80 _[77,83]	20 _[20,20]	26 _[25,28]
QC	$h = 25$	$h_a = 25$	21 _[13,31]	12 _[7,18]	0 _[0,0]	0 _[0,0]	30 _[27,33]	37 _[33,42]
QC	$h = 5$	$h_a = 5$	20 _[7,36]	35 _[26,43]	0 _[0,0]	48 _[45,52]	20 _[20,20]	28 _[27,30]
QC-NS	$n = 25$	$h_a = 5$	51 _[22,80]	53 _[28,77]	18 _[10,25]	60 _[58,61]	95 _[94,96]	95 _[93,97]
NS	$n = 25$	$h_a = 1$	30 _[26,35]	19 _[11,28]	9 _[6,12]	95 _[94,97]	89 _[87,91]	91 _[86,94]
NS	$n = 5$	$h_a = 1$	93 _[91,94]	27 _[11,43]	1 _[0,3]	89 _[87,91]	93 _[91,95]	56 _[48,63]
OS	$n = 1$	$h_a = 1$	47 _[41,53]	0 _[0,0]	0 _[0,0]	0 _[0,0]	19 _[18,19]	19 _[19,20]
FBC			54 _[51,56]	34 _[32,37]	0 _[0,0]	1 _[1,2]	0 _[0,0]	1 _[0,1]
HFBC			56 _[53,59]	37 _[34,40]	28 _[26,29]	6 _[4,8]	0 _[0,0]	4 _[3,5]
IQL			66 _[63,67]	53 _[52,55]	0 _[0,0]	3 _[2,5]	20 _[19,20]	6 _[3,9]
HIQL			35 _[31,39]	24 _[21,28]	20 _[17,23]	24 _[22,26]	0 _[0,0]	9 _[5,13]
SHARSA			83 _[81,85]	64 _[62,68]	34 _[31,36]	19 _[16,23]	1 _[1,2]	64 _[60,68]

Table C.1: Complete results for all hyperparameter configurations across different combinations of h , n and h_a (10 seeds). We adopt the following abbreviations: c3=cube-triple, c4=cube-quadruple, c8=cube-octuple, hg=humanoidmaze-giant, p45=puzzle-4x5, p46=puzzle-4x6. The hyperparameters used are specified in Tables C.5 and C.6.

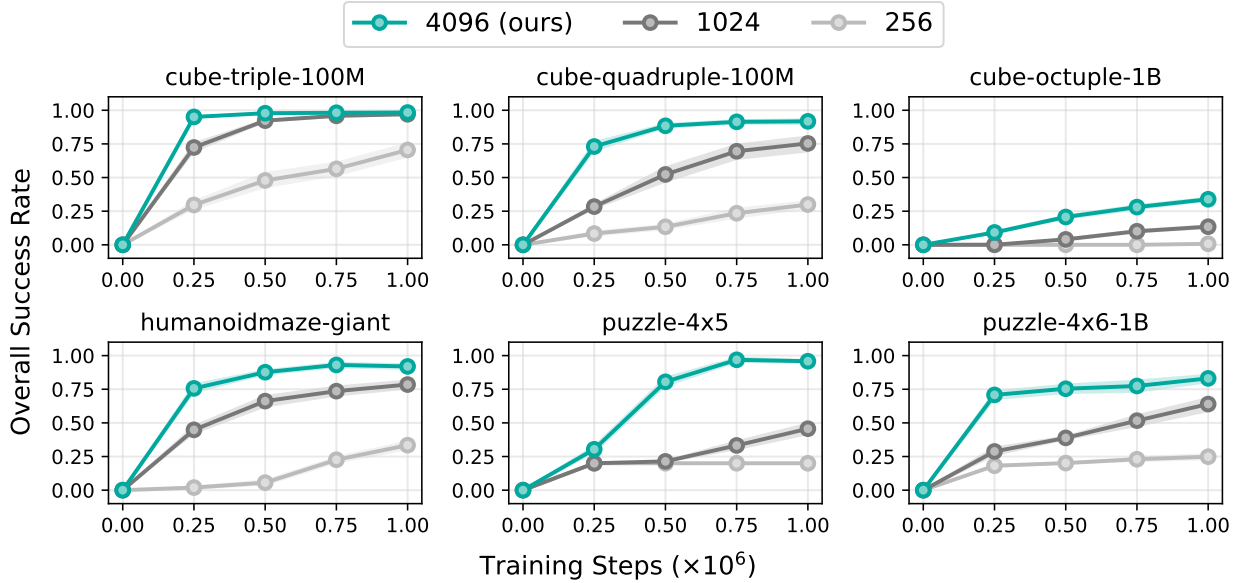


Figure C.1: **Batch size sensitivity (10 seeds)**. Large batch size is crucial for DQC’s performance especially on hard tasks (*e.g.*, `cube-quadruple`, `cube-octuple`, `puzzle-4x5` and `puzzle-4x6`).

C.2 Environments and datasets

To evaluate our method, we consider 8 goal-conditioned environments in OGBench with varying difficulties. The dataset size, episode length, and the action dimension for each environment is available in Table C.2. We describe each of the environments and the datasets we use as follows.

Environment cube-*: We consider three cube environments (`cube-triple`, `cube-quadruple`, `cube-octuple`). As the names suggest, the goal of these environments involve using a robot arm to manipulate 3/4/8 cubes from some initial configuration to some specified goal configuration. We use the same five evaluation tasks used in OGBench [111] for `cube-triple` and `cube-quadruple` and the same five evaluation tasks used in [110] for `cube-octuple`.

Environment humanoidmaze-*: We also consider the hardest locomotion environment available in OGBench. The goal of the environment is to control and navigate a humanoid agent from some initial location to some specified goal location in a 16×12 maze. This environment also has the longest episode length (4000, more than twice as long as the second longest episode length as used in `cube-octuple`).

Environment puzzle-*: Finally, we consider two environments that involve solving a combinatorial puzzle with a robot arm. The puzzle consists of a board of 4×5 or 4×6

Environment	Dataset Size	Episode Length	Action Dimension (A)
cube-triple-100M	100M	1000	5
cube-quadruple-100M	100M	1000	5
cube-octuple-1B	1B	1500	5
humanoidmaze-giant	4M (default)	4000	21
puzzle-4x5	3M (default)	1000	5
puzzle-4x6-1B	1B	1000	5

Table C.2: **Environment metadata.** For both `humanoidmaze-giant` and `puzzle-4x5`, we use the default dataset that is released in the original OGBench benchmark [111]. For the other environments, we use larger datasets as we find them to be essential for achieving good performance on these environments.

buttons, organized as a regular grid (4 rows and 5 or 6 columns). Each button has a binary state. Whenever the end-effector of the arm touches a button, the button and all its adjacent four buttons (three or two if the button is on the edge of the grid or in the corner) flip its binary state. The goal of the environment is to transform the board from some initial state to some specified goal state.

At the test-time/evaluation-time, the goal-conditioned agent is tested on five evaluation tasks for each of the six environments we consider. The overall success rate is the average over 5 tasks with 50 evaluation trials each. For the prior baselines, SHARSA, HIQL, IQL, HFBC and FBC, we run 15 evaluation trials for each task, following [110].

Datasets. We use `play` datasets for all `cube-*` and `puzzle-*` environments and `navigate` dataset for `humanoidmaze-*`. We use the original datasets available for `humanoidmaze-giant` and `puzzle-4x5` because they are sufficient for solving the environments. Using larger datasets on these environments do not help differentiate among different methods/baselines. For each of the other environments, we use the largest dataset available from [110] as we find it to be necessary to solve these environments (or achieve non-trivial performance on `cube-octuple`).

C.3 Hyperparameters and implementation details

Hyperparameters. Table C.3 describes the common hyperparameters used in all our experiments. Tables C.5 and C.6 describe the environment-specific hyperparameters and Table C.7 describes the range of hyperparameters we use for tuning each method.

Goal-conditioned RL implementation details. While we have described in the main

Parameter	Value
Batch size	4096
Discount factor (γ)	0.999
Optimizer	Adam
Learning rate	3×10^{-4}
Target network update rate (λ)	5×10^{-3}
Critic ensemble size (K)	2
Critic target	$\min(Q_1, Q_2)$ for cube-* $(Q_1 + Q_2)/2$ for puzzle-* and humanoid-*
Value loss type	binary cross entropy
Best-of-N sampling (N)	32
Number of flow steps	10
Number of training steps	10^6
Network width	1024
Network depth	4 hidden layers
Value goal sampling ($w_{\text{cur}}^v, w_{\text{geom}}^v, w_{\text{traj}}^v, w_{\text{rand}}^v$)	(0.2, 0, 0.5, 0.3)
DQC/QC/NS/OS: π_β is not goal-conditioned	
Actor goal sampling ($w_{\text{cur}}^p, w_{\text{geom}}^p, w_{\text{traj}}^p, w_{\text{rand}}^p$)	SHARSA (cube): (0, 1, 0, 0)
	SHARSA (puzzle): (0, 0, 1, 0)
	SHARSA (humanoidmaze): (0, 0, 1, 0)

Table C.3: **Common hyperparameters.** For the GCRL goal-sampling distribution we follow the same hyperparameters used in [110].

body of the paper how DQC works as a general RL algorithm, we have not touched on how DQC and similarly all our baselines work with the goal-conditioned RL (GCRL) setting. We consider the setting where we have access to an oracle goal representation $\Psi : \mathcal{S} \rightarrow \mathcal{G}$ where $\mathcal{G}$ is the goal space (see Table C.4 for the oracle goal representation description for each environment). The goal-conditioned reward function $r : (s, g) \mapsto \mathbb{I}_{\Psi(s)=g}$ is a binary reward function where its output is 1 if the goal g is reached by the current state s . We can treat g as part of an extended state $\tilde{s} = [s, g] \in \tilde{\mathcal{S}} = \mathcal{S} \times \mathcal{G}$ and learn value functions (*e.g.*, $Q_\phi(\tilde{s}, a)$) normally with such extended state.

A common practical trick in the GCRL setting is goal relabeling. That is, during training for each (s, a) pair in the training batch, a goal g is sampled from some distribution (*i.e.*, $p^{\mathcal{D}}(\cdot \mid s, a)$) and the reward of the transition is relabeled with the goal-conditioned reward function. Following [110], the goal distribution $P^g(\cdot \mid s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathcal{G}}$ is a mixture of

Environment	Goal Representation (Ψ)	Goal Domain ($\mathcal{G}$)
cube-triple	(x, y, z) of three cubes (rel. to center)	$\mathbb{R}^9$
cube-quadruple	(x, y, z) of four cubes (rel. to center)	$\mathbb{R}^{12}$
cube-octuple	(x, y, z) of eight cubes (rel. to center)	$\mathbb{R}^{24}$
humanoidmaze-giant	(x, y) of the humanoid	$\mathbb{R}^2$
puzzle-4x5	the binary state for each button	$\{0, 1\}^{20}$
puzzle-4x6	the binary state for each button	$\{0, 1\}^{24}$

Table C.4: **Oracle goal representation description for each environment.** Following [110], we assume access to an oracle goal representation for each environment. More detailed definitions of these oracle goal representations are available in OGBench [111].

Environment	DQC ($h, h_a, \kappa_b, \kappa_d$)	DQC-naïve (h, h_a, κ_b)	QC-NS (h, h_a, κ_b)	QC ($h = h_a, \kappa_b$)	NS (n, κ_b)	OS (κ_b)	SHARSA (n)	HIQL (h, κ, α)	IQL (α)	HFBC (h)
cube-triple-100M	(25, 5, 0.93, 0.8)	(25, 5, 0.93)	(25, 5, 0.93)	(5, 0.93)	(5, 0.5)	0.5	25	(25, 0.5, 10)	3	25
cube-quadruple-100M	(25, 5, 0.93, 0.8)	(5, 1, 0.93)	(25, 5, 0.93)	(5, 0.93)	(5, 0.7)	0.7	25	(25, 0.5, 10)	3	25
cube-octuple-1B	(25, 5, 0.93, 0.5)	(25, 5, 0.93)	(25, 5, 0.93)	(25, 0.93)	(25, 0.97)	0.7	25	(50, 0.5, 10)	10	50
humanoidmaze-giant	(25, 1, 0.5, 0.8)	(5, 1, 0.9)	(25, 5, 0.5)	(5, 0.5)	(25, 0.7)	0.5	50	(50, 0.5, 3)	0.3	50
puzzle-4x5	(25, 5, 0.9, 0.5)	(25, 5, 0.9)	(25, 5, 0.7)	(5, 0.9)	(25, 0.7)	0.7	50	(25, 0.7, 3)	1	25
puzzle-4x6-1B	(25, 1, 0.7, 0.5)	(25, 5, 0.7)	(25, 5, 0.5)	(5, 0.7)	(25, 0.5)	0.7	50	(25, 0.7, 3)	1	25

Table C.5: **Environment-specific hyperparameters for DQC, QC, NS, OS, SHARSA, HIQL, IQL, and HFBC.** For SHARSA, HIQL, IQL, and HFBC, we follow the hyperparameters in the original paper [110].

four distributions, conditioned on the training state-action example:

$$P^g = w_{\text{cur}} P_{\text{cur}}^g + w_{\text{geom}} P_{\text{geom}}^g + w_{\text{traj}} P_{\text{traj}}^g + w_{\text{rand}} P_{\text{rand}}^g, \quad (\text{C.1})$$

where

1. $P_{\text{cur}}^g(\cdot \mid s, a) = \delta_{\Psi(s)}$: the goal is the same as the current state;
2. $P_{\text{geom}}^g(\cdot \mid s, a)$: geometric distribution over the future states in the same trajectory that (s, a) is from;
3. $P_{\text{traj}}^g(\cdot \mid s, a)$: uniform distribution over the future states in the same trajectory that (s, a) is from; and finally
4. $P_{\text{rand}}^g(\cdot \mid s, a) = \Psi(\mathcal{U}_{\mathcal{D}(s)})$: uniform distribution over the dataset ($\mathcal{D}(s)$ is the distribution of states in the dataset).

Environment	DQC	DQC	DQC	QC-NS	NS	NS	QC	QC	OS
	$h = 25, h_a = 5$ (κ_b, κ_d)	$h = 25, h_a = 1$ (κ_b, κ_d)	$h = 5, h_a = 1$ (κ_b, κ_d)	$n = 25, h_a = 5$ κ_b	$n = 25$ κ_b	$n = 5$ κ_b	$h = 25$ κ_b	$h = 5$ κ_b	κ_b
cube-triple-100M	(0.93, 0.8)	(0.93, 0.8)	(0.5, 0.8)	0.93	0.5	0.5	0.93	0.93	0.5
cube-quadruple-100M	(0.93, 0.8)	(0.93, 0.8)	(0.5, 0.8)	0.93	0.5	0.7	0.93	0.93	0.7
cube-octuple-1B	(0.93, 0.5)	(0.93, 0.5)	(0.93, 0.5)	0.93	0.97	0.5	0.93	0.93	0.7
humanoidmaze-giant	(0.5, 0.8)	(0.5, 0.8)	(0.5, 0.5)	0.5	0.5	0.5	0.5	0.5	0.5
puzzle-4x5	(0.9, 0.5)	(0.9, 0.5)	(0.5, 0.5)	0.7	0.7	0.5	0.9	0.9	0.7
puzzle-4x6-1B	(0.7, 0.5)	(0.7, 0.5)	(0.5, 0.5)	0.5	0.7	0.5	0.7	0.7	0.7

Table C.6: **Environment-specific hyperparameters under different h, n, h_a configurations for DQC, QC, NS, OS.** For DQC-naïve, we use the same hyperparameter as the corresponding QC baseline.

Environment	Backup Quantile	Distillation Expectile	Backup Horizon	Policy Chunk Size
	(κ_b)	(κ_d)	(h) or (n)	(h_a)
cube-*	{0.5, 0.7, 0.9, 0.93, 0.95, 0.97, 0.99}	{0.5, 0.8}	{5, 25}	{1, 5, 25}
Others	{0.5, 0.7, 0.9}	{0.5, 0.8}	{5, 25}	{1, 5, 25}

Table C.7: **Hyperparameter tuning range for all methods.** For NS, we only tune κ_b and n because the policy chunk size is always 1 and there is no distilled critic. Similarly, for QC, we only tune κ_b and $h = h_a$ because the policy chunk size is the same as the critic chunk size and there is no distilled critic. For OS, we only tune κ_b .

and $w_{\text{cur}}, w_{\text{geom}}, w_{\text{traj}}, w_{\text{rand}} > 0$ are the corresponding weights for each of the distribution components with $w_{\text{cur}} + w_{\text{geom}} + w_{\text{traj}} + w_{\text{rand}} = 1$.

In practice, it has been found to be beneficial to use a separate set of goal sampling weights for TD backup [111] (*i.e.*, $(w_{\text{cur}}^v, w_{\text{geom}}^v, w_{\text{traj}}^v, w_{\text{rand}}^v)$) and for policy learning (*i.e.*, $(w_{\text{cur}}^p, w_{\text{geom}}^p, w_{\text{traj}}^p, w_{\text{rand}}^p)$). However, in our implementation of DQC/QC/NS/OS, we do not train a goal-conditioned policy, as our policy extraction is done entirely at test-time by best-of-N sampling from an *unconditional* (*i.e.*, not goal-conditioned) behavior policy π_β . In particular, we use an unconditioned flow policy $\pi_\beta(\cdot | s)$ that is parameterized by a velocity field $v_\beta : \mathcal{S} \times \mathbb{R}^A \times [0, 1] \rightarrow \mathbb{R}^A$ that is trained with the standard flow-matching objective:

$$L_{\text{FM}}(\beta) = \mathbb{E}_{u \sim \mathcal{U}[0,1], z \sim \mathcal{N}_{(s,a)} \sim \mathcal{D}} [\|v_\beta(s, (1-u)z + ua, u) - a + z\|_2^2] \quad (\text{C.2})$$

For SHARSA, we use the official implementation where both flow policies (high-level and low-level) are goal-conditioned (and thus are trained with the goal distribution mixture specified by $w_{\text{cur}}^p, w_{\text{geom}}^p, w_{\text{traj}}^p, w_{\text{rand}}^p$). The goal sampling distribution for training the value networks (for all methods) and the goal sampling distribution for the policy networks (for SHARSA only) are provided in Table C.3.

Appendix D

Proofs for Main Theoretical Results

The theoretical results in this section rely on the following assumption:

Assumption 2 (Data Obeys the Transition Dynamics) $\mathcal{D} \in \Delta_{\mathcal{T}}$ is a trajectory distribution generated by rolling out a behavior policy from a distribution of $s_t \sim \mu$. The behavior policy can be non-Markovian (*i.e.*, $\pi_{\beta}(a_{t+k} \mid s_{t:t+k+1}, a_{t:t+k})$). Each subsequent state is generated according to the MDP dynamics $\mathcal{M}$: $s_{t+k+1} \sim T(\cdot \mid s_{t+k}, a_{t+k}), \forall k \in \{0, 1, \dots, h-1\}$. The resulting trajectory is $(s_t, s_{t+1}, \dots, s_{t+h}, a_t, a_{t+1}, \dots, a_{t+h}) \in \mathcal{T} = \mathcal{S}^h \times \mathcal{A}^h$.

D.1 Utility Lemmata

We start by introducing a few lemmata for the proofs of our main results.

Lemma 1 (Mean value theorem for conditional probabilities) Let $P_1, P_2 \in \Delta_{\mathcal{X} \times \mathcal{Y}}$ and $P(x, y) := \hat{\alpha}(y)P_1(x, y) + (1 - \hat{\alpha}(y))P_2(x, y)$ and there exists $\alpha > 0$ such that $\hat{\alpha}(y) \leq \alpha, \forall y \in \mathcal{Y}$. Then, there exists $y \in \mathcal{Y}$ and $\tilde{\alpha} \leq \alpha$ such that

$$P(\cdot \mid y) = \tilde{\alpha}P_1(\cdot \mid y) + (1 - \tilde{\alpha})P_2(\cdot \mid y) \quad (\text{D.1})$$

Proof.

$$\begin{aligned} \frac{P(x, y)}{P(y)} &= \frac{\hat{\alpha}(y)P_1(y)P_1(x \mid y) + (1 - \hat{\alpha}(y))P_2(x \mid y)}{\hat{\alpha}(y)P_1(y) + (1 - \hat{\alpha}(y))P_2(y)} \\ &= \beta(y)P_1(x \mid y) + (1 - \beta(y))P_2(x \mid y) \end{aligned} \quad (\text{D.2})$$

where $\beta(y) := \frac{\hat{\alpha}(y)P_1(y)}{\hat{\alpha}(y)P_1(y) + (1 - \hat{\alpha}(y))P_2(y)}$. We now prove $\exists y \in \mathcal{Y}, \tilde{\alpha} \leq \alpha$ for Equation (D.1) to hold by contradiction.

We first assume $\tilde{\alpha} = \beta(y) > \alpha, \forall y \in \mathcal{Y}$. Now, substitute $\beta(y)$ in and integrate both side by y to obtain

$$\hat{\alpha}(y)P_1(y) > \alpha\hat{\alpha}(y)P_1(y) + \alpha(1 - \hat{\alpha}(y))P_2(y) \quad (\text{D.3})$$

$$\hat{\alpha}(y) > \alpha\hat{\alpha}(y) + \alpha - \alpha\hat{\alpha}(y) = \alpha, \quad (\text{D.4})$$

which is a contradiction to the condition $\hat{\alpha}(y) \leq \alpha$.

Therefore, there must exist $y \in \mathcal{Y}$ with $\tilde{\alpha} \leq \alpha$ such that Equation (D.1) holds. $\square$

Lemma 2 (Expectation difference for bounded function and TV) For two distributions $P, Q \in \Delta_{\mathcal{X}}$ and two bounded functions $f, g : \mathcal{X} \rightarrow [0, 1]$, if the TV distance between P and Q is no larger than ε and $\|f - g\|_{\infty} \leq \delta$ under $\text{supp}(P) \cap \text{supp}(Q)$, then

$$|\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{x \sim Q}[g(x)]| \leq (1 - \varepsilon)\delta + \varepsilon. \quad (\text{D.5})$$

Proof. Let's decompose the probability mass of P and Q in terms of $d_P, d_{PQ}, d_Q : \mathcal{X} \rightarrow \mathbb{R}$ as the following:

$$P(x) = d_P(x) + d_{PQ}(x), \quad (\text{D.6})$$

$$Q(x) = d_{PQ}(x) + d_Q(x). \quad (\text{D.7})$$

The $\int d_P(x)dx$ maximizing solution is

$$d_P(x) = \max(P(x), Q(x)) - Q(x) \quad (\text{D.8})$$

$$d_Q(x) = \max(P(x), Q(x)) - P(x) \quad (\text{D.9})$$

$$d_{PQ}(x) = P(x) + Q(x) - \max(P(x), Q(x)). \quad (\text{D.10})$$

It is clear that under this decomposition,

$$\int d_P(x)dx = \int d_Q(x)dx = \hat{\varepsilon} \leq \varepsilon, \quad (\text{D.11})$$

$$\int d_{PQ}(x)dx = 1 - \hat{\varepsilon} \geq 1 - \varepsilon. \quad (\text{D.12})$$

Now we are ready to bound the expectation difference:

$$\begin{aligned}
& |\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{x \sim Q}[g(x)]| \\
&= \left| \left(\int d_P(x) f(x) dx - \int d_Q(x) g(x) dx \right) + \left(\int d_{PQ}(x) (f(x) - g(x)) dx \right) \right| \\
&\leq \left| \int d_P(x) f(x) dx - \int d_Q(x) g(x) dx \right| + \left| \int d_{PQ}(x) (f(x) - g(x)) dx \right| \\
&\leq \max \left(\sup_x f(x) \int d_P(x) dx - \inf_x g(x) \int d_Q(x) dx, \sup_x g(x) \int d_Q(x) dx - \inf_x f(x) \int d_P(x) dx \right) \\
&\quad + \left| \left(\sup_{x: d_{PQ}(x) > 0} |f(x) - g(x)| \right) \int d_{PQ}(x) dx \right| \\
&\leq \hat{\varepsilon} + \left(\sup_{x \in \text{supp}(P) \cap \text{supp}(Q)} |f(x) - g(x)| \right) (1 - \hat{\varepsilon}) \\
&= \hat{\varepsilon} + \|f - g\|_\infty (1 - \hat{\varepsilon}) \\
&\leq \hat{\varepsilon} (1 - \delta) + \delta \\
&\leq \varepsilon (1 - \delta) + \delta \\
&= (1 - \varepsilon) \delta + \varepsilon
\end{aligned} \tag{D.13}$$

as desired. $\square$

Lemma 3 (Total variation under event conditioning) For two random variables $X \in \Delta_{\mathcal{X}}$ and $Y \in \Delta_{\mathcal{Y}}$ and any $y \in \mathcal{Y}$,

$$D_{\text{TV}}(P(X | Y = y) \parallel P(X)) \leq 1 - P(Y = y) \tag{D.14}$$

Proof. Let $p = P(Y = y)$

$$\begin{aligned}
& D_{\text{TV}}(P(X | Y = y) \parallel P(X)) \\
&= \frac{1}{2} \int |P(x) - P(x | y)| dx \\
&= \frac{1}{2} \int |P(x | Y = y)(P(Y = y) - 1) + P(x | Y \neq y)P(Y \neq y)| dx \\
&= \frac{1-p}{2} \int |(P(x | Y \neq y) - P(x | Y = y))| dx \\
&= (1-p) D_{\text{TV}}(P(X | Y = y) \parallel P(X | Y \neq y)) \\
&\leq 1 - p
\end{aligned} \tag{D.15}$$

$\square$

Lemma 4 (Data Processing Inequality for f -divergence [25]) For two random variables $A, B \in \Delta_{\mathcal{X}}$ and a deterministic function $f : \mathcal{X} \rightarrow \mathcal{Y}$, and $C := g(A), D := g(B)$

$$D_f(P_A \parallel P_B) \geq D_f(P_C \parallel P_D). \quad (\text{D.16})$$

Since TV-distance is a f -divergence with $f = |x - 1|$, we have

$$D_{\text{TV}}(P_A \parallel P_B) \geq D_{\text{TV}}(P_C \parallel P_D). \quad (\text{D.17})$$

Proof from [157].

$$\begin{aligned} D_f(P_A \parallel P_B) &= \mathbb{E}_{x \sim P_B} [f(P_A(x)/P_B(x))] \\ &= \mathbb{E}_{P_{BD}} [f(P_{AC}/P_{BD})] \\ &= \mathbb{E}_{(x,y) \sim P_D} \left[\mathbb{E}_{P_{B|D}} [f(P_{AC}(x,y)/P_{BD}(x,y))] \right] \\ &\geq \mathbb{E}_{y \sim P_D} \left[f \left(\mathbb{E}_{x \sim P_{B|D=y}} [P_{AC}(x,y)/P_{BD}(x,y)] \right) \right] \\ &= \mathbb{E}_{y \sim P_D} \left[f \left(\mathbb{E}_{x \sim P_{B|D=y}} [P_C(y)/P_D(y)] \right) \right] \\ &= \mathbb{E}_{y \sim P_D} [f(P_C(y)/P_D(y))] \\ &= D_f(P_C \parallel P_D). \end{aligned} \quad (\text{D.18})$$

□

D.2 Proof of Theorem 1

In this section, we provide the proof for Theorem 1 (restated below).

Theorem 1 (AC Value Bias) Let $\hat{V}_{\text{ac}} : \mathcal{S} \rightarrow [0, 1/(1 - \gamma)]$ be a solution of

$$\hat{V}_{\text{ac}}(s_t) = \mathbb{E}_{s_{t+1:t+h+1}, a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t)} \left[R_{t:t+h} + \gamma^h \hat{V}_{\text{ac}}(s_{t+h}) \right], \quad (5.7)$$

with $R_{t:t+h} = \sum_{t'=t}^{t+h} \gamma^{t'-t} r(s_{t'}, a_{t'})$ and V_{ac} is the true value of $\tilde{\pi}_{\text{ac}} : s_t \mapsto P_{\mathcal{D}}(a_{t:t+h} | s_t)$. If $\mathcal{D}$ is weakly ε_h -open-loop consistent, then for all $s_t \in \text{supp}(P_{\mathcal{D}}(s_t))$,

$$\left| V_{\text{ac}}(s_t) - \hat{V}_{\text{ac}}(s_t) \right| \leq \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)} \leq \varepsilon_h H \bar{H}. \quad (5.8)$$

Proof. Since $\mathcal{D}$ is $\varepsilon_{h'}$ -open-loop consistent in state-action for $h' < h$, the state-action distribution leading up to step h admits the following bound:

$$D_{\text{TV}}(P_{\mathcal{D}}(s_{t+h}, a_{t+h} | s_t) \parallel P_{\mathcal{D}}^{\circ}(s_{t+h}, a_{t+h} | s_t)) \leq \varepsilon_h \quad (\text{D.19})$$

Let $R_{t:t+h} = \sum_{k=0}^{h-1} \gamma^k r(s_{t+k}, a_{t+k})$ be the h -step reward distribution. Then the difference in h -step reward is bounded by

$$\begin{aligned} & \left| \mathbb{E}_{P_{\mathcal{D}}(\cdot | s_t)}[R_{t:t+h}] - \mathbb{E}_{P_{\mathcal{D}}^{\circ}(\cdot | s_t)}[R_{t:t+h}] \right| \\ & \leq \sum_{h'=1}^{h-1} \left[\gamma^{h'} \mathbb{E}_{P_{\mathcal{D}}(s_{t+h'}, a_{t+h'} | s_t)}[r(s_{t+h'}, a_{t+h'})] - \mathbb{E}_{P_{\mathcal{D}}^{\circ}(s_{t+h'}, a_{t+h'} | s_t)}[r(s_{t+h'}, a_{t+h'})] \right] \\ & \leq \sum_{h'=1}^{h-1} \gamma^{h'} \varepsilon_h. \end{aligned} \quad (\text{D.20})$$

where the first inequality uses Lemma 2 and the fact that TV distance is bounded (Equation (D.19)).

Since $\mathcal{D}$ is ε_h -open-loop consistent for h in state, we have

$$D_{\text{TV}}(P_{\mathcal{D}}(s_{t+h} | s_t) \parallel P_{\mathcal{D}}^{\circ}(s_{t+h} | s_t)) \leq \varepsilon_h, \quad (\text{D.21})$$

which can then be used to bound the estimation error using Lemma 2:

$$\begin{aligned} & \left| \mathbb{E}_{s_{t+h} \sim P_{\mathcal{D}}(s_{t+h} | s_t)} \left[\hat{V}_{\text{ac}}(s_{t+h}) \right] - \mathbb{E}_{s_{t+h} \sim P_{\mathcal{D}}^{\circ}(s_{t+h} | s_t)} [V_{\text{ac}}(s_{t+h})] \right| \\ & \leq \frac{\varepsilon_h}{1 - \gamma} + (1 - \varepsilon_h) \sup_{s_{t+h} \in \text{supp}(P_{\mathcal{D}}(s_{t+h} | s_t))} \left[|\hat{V}_{\text{ac}}(s_{t+h}) - V_{\text{ac}}(s_{t+h})| \right] \end{aligned} \quad (\text{D.22})$$

For all $s_t \in \text{supp}(P_{\mathcal{D}}(s_t))$,

$$\begin{aligned}
& \left| \hat{V}_{\text{ac}}(s_t) - V_{\text{ac}}(s_t) \right| \\
& \leq \left| \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t)}[R_{t:t+h}] - \mathbb{E}_{P_{\mathcal{D}}^{\circ}(\cdot|s_t)}[R_{t:t+h}] \right| \\
& \quad + \gamma^h \left| \mathbb{E}_{s_{t+h} \sim P_{\mathcal{D}}(s_{t+h}|s_t)} \left[\hat{V}_{\text{ac}}(s_{t+h}) \right] - \mathbb{E}_{s_{t+h} \sim P_{\mathcal{D}}^{\circ}(s_{t+h}|s_t)} \left[V_{\text{ac}}(s_{t+h}) \right] \right| \\
& \leq \sum_{h'=0}^{h-1} \left[\gamma^{h'} \varepsilon_h \right] + \frac{\gamma^h \varepsilon_h}{1 - \gamma} + \gamma^h (1 - \varepsilon_h) \sup_{s_{t+h} \in \text{supp}(P_{\mathcal{D}}(s_{t+h}|s_t))} \left[|\hat{V}_{\text{ac}}(s_{t+h}) - V_{\text{ac}}(s_{t+h})| \right].
\end{aligned} \tag{D.23}$$

Since the support of $s_{t+h} \mid s_t$ is a subset of the support for s_t by Assumption 2, we can recursively apply the inequality to obtain,

$$\begin{aligned}
\left| \hat{V}_{\text{ac}}(s_t) - V_{\text{ac}}(s_t) \right| & \leq \frac{1}{1 - (1 - \varepsilon_h) \gamma^h} \left(\sum_{h'=1}^{h-1} \left[\gamma^{h'} \varepsilon_h \right] + \frac{\gamma^h \varepsilon_h}{1 - \gamma} \right) \\
& = \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h) \gamma^h)},
\end{aligned} \tag{D.24}$$

as desired. □

D.3 Proof of Theorem 2

In this section, we provide the proof for Theorem 2 (restated below).

Theorem 2 (Worst-case AC Value Bias) For any $h > 1, \gamma \in [0, 1), \varepsilon_h \in [0, 1/2]$, there exists an MDP $\mathcal{M}$ and a weakly ε_h -open-loop consistent $\mathcal{D}$ such that for some $s_t \in \text{supp}(P_{\mathcal{D}}(s_t))$,

$$V_{\text{ac}}(s_t) - \hat{V}_{\text{ac}}(s_t) = \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)}. \quad (5.9)$$

Similarly, there exists an MDP $\mathcal{M}$ and an ε_h -open-loop consistent $\mathcal{D}$ such that for some $s_t \in \text{supp}(P_{\mathcal{D}}(s_t))$,

$$\hat{V}_{\text{ac}}(s_t) - V_{\text{ac}}(s_t) = \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)}. \quad (5.10)$$

Proof. Let $\delta \in [0, 1]$ be any value that satisfies $\varepsilon_h = 2\delta(1 - \delta)$. δ must exist because $\varepsilon_h \in [0, 1/2]$. Let us define a MDP that has $S = 2h$ states, $\mathcal{S} = \{X_0, X_1, \tilde{X}_1, \dots, X_{h-1}, \tilde{X}_{h-1}, Z\}$, and $A = 2$ actions, $\mathcal{A} = \{0, 1\}$, and the following transition function T and reward function r (see a diagram in Figure D.1):

$$\begin{aligned} T(\tilde{X}_{i+1} \mid X_i, a) &= T(\tilde{X}_{i+1} \mid \tilde{X}_i, a) = \delta, & \forall a \in \{0, 1\}, i \in \{1, \dots, h-2\} \\ T(X_{i+1} \mid X_i, a) &= T(X_{i+1} \mid \tilde{X}_i, a) = 1 - \delta, & \forall a \in \{0, 1\}, i \in \{0, \dots, h-2\} \\ T(Z \mid \tilde{X}_{h-1}, a = 1) &= T(Z \mid X_{h-1}, a = 0) = 1 \\ T(X_0 \mid \tilde{X}_{h-1}, a = 0) &= T(X_0 \mid X_{h-1}, a = 1) = 1 \\ r(\tilde{X}_i, a = 0) &= r(X_i, a = 1) = 1, & \forall i \in \{0, \dots, h-1\} \\ r(\tilde{X}_i, a = 1) &= r(X_i, a = 0) = 0, & \forall i \in \{0, \dots, h-1\} \\ r(Z, a = 1) &= r(Z, a = 0) = 0 \\ T(Z \mid Z, a = 0) &= T(Z \mid Z, a = 1) = 1 \end{aligned} \quad (\text{D.25})$$

Now, we assume that the data $\mathcal{D}$ is collected by the optimal closed-loop policy where

$$\pi(X_i) = 1, \pi(\tilde{X}_i) = 0. \quad (\text{D.26})$$

First, we check $\mathcal{D}$ is ε_h -open-loop consistent.

We can show that by computing the distribution for $P_{\mathcal{D}}(s_{t+i}, a_{t+i} \mid s_t = X_0)$ and $P_{\mathcal{D}}^{\circ}(s_{t+i}, a_{t+i} \mid$

Now, we are ready to compute $V_{\text{ac}}(X_0)$:

$$\begin{aligned} V_{\text{ac}}(X_0) &= \frac{(1 - \gamma^h) - \varepsilon_h(\gamma - \gamma^h)}{(1 - \gamma)} + \gamma^h [(1 - \varepsilon_h)V_{\text{ac}}(X_0) + \varepsilon_h V_{\text{ac}}(Z)] \\ &= \frac{1 - \gamma^h - \varepsilon_h(\gamma - \gamma^h)}{(1 - \gamma)(1 - \gamma^h(1 - \varepsilon_h))} \end{aligned} \quad (\text{D.32})$$

Finally, with $X_0 \in \text{supp}(\mathcal{D})$, we obtain the desired value difference

$$\hat{V}_{\text{ac}}(X_0) - V_{\text{ac}}(X_0) = \frac{\varepsilon_h \gamma}{(1 - \gamma)(1 - \gamma^h(1 - \varepsilon_h))}. \quad (\text{D.33})$$

By symmetry, we can flip the reward value (*i.e.*, $0 \rightarrow 1$ and $1 \rightarrow 0$) to construct the example such that

$$V_{\text{ac}}(X_0) - \hat{V}_{\text{ac}}(X_0) = \frac{\varepsilon_h \gamma}{(1 - \gamma)(1 - \gamma^h(1 - \varepsilon_h))}. \quad (\text{D.34})$$

□

D.4 Proof of Corollary 1

In this section, we provide the proof for Corollary 1 (restated below).

Corollary 1 (Optimality Gap for AC Policy) Let $\mathcal{D}^*$ be the data collected by any optimal policy π^* . If $\mathcal{D}^*$ is weakly ε_h -open-loop consistent, then for all $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V_{\text{ac}}^*(s_t) \leq V^*(s_t) - \tilde{V}_{\text{ac}}(s_t) \leq \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)} \leq \varepsilon_h H \bar{H}, \quad (5.11)$$

where V^* is the value of the optimal policy π^* , V_{ac}^* is the *true* value of the optimal action chunking policy, and $\tilde{V}_{\text{ac}}$ is the *true* value of the action chunking policy from cloning the data $\mathcal{D}^*$:

$$\tilde{\pi}_{\text{ac}}^{\mathcal{D}^*}(a_{t:t+h} \mid s_t) : s_t \mapsto P_{\mathcal{D}^*}(\cdot \mid s_t). \quad (5.12)$$

Proof. Let $\hat{V}_{\text{ac}}$ be the fixed point of the following equation:

$$\hat{V}_{\text{ac}}(s_t) = \mathbb{E}_{s_{t+1:t+h+1}, a_{t:t+h} \sim P_{\mathcal{D}^*}(\cdot \mid s_t)} \left[R_{t:t+h} + \gamma^h \hat{V}_{\text{ac}}(s_{t+h}) \right] \quad (\text{D.35})$$

where again $R_{t:t+h} = \sum_{t'=t}^{t+h} \gamma^{t'-t} r(s_{t'}, a_{t'})$. The value of the optimal policy is the fixed point of the following equation:

$$\begin{aligned} V^*(s_t) &= \mathbb{E}_{s_{t+1}, a_t \sim P_{\mathcal{D}^*}(\cdot \mid s_t)} [r(s_t, a_t) + \gamma V^*(s_{t+1})] \\ &= \mathbb{E}_{s_{t:t+2}, a_{t:t+1} \sim P_{\mathcal{D}^*}(\cdot \mid s_t)} [r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma V^*(s_{t+2})] \\ &\dots \\ &= \mathbb{E}_{s_{t+1:t+h+1}, a_{t:t+h} \sim P_{\mathcal{D}^*}(\cdot \mid s_t)} [R_{t:t+h} + \gamma^h V^*(s_{t+h})] \end{aligned} \quad (\text{D.36})$$

which is equivalent to the fixed-point equation for $\hat{V}_{\text{ac}}$. Therefore $\hat{V}_{\text{ac}} = V^*$. By Theorem 1, we know that the true value V_{ac} of the action chunking policy $\tilde{\pi}_{\text{ac}}$ that clones $\mathcal{D}^*$ is close to $\hat{V}_{\text{ac}}$. More specifically, for all $s_t \in \text{supp}(\mathcal{D}^*)$,

$$\left| \hat{V}_{\text{ac}}(s_t) - \tilde{V}_{\text{ac}}(s_t) \right| \leq \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)}, \quad (\text{D.37})$$

which means that

$$V^*(s_t) - \tilde{V}_{\text{ac}}(s_t) \leq \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)}, \quad (\text{D.38})$$

where we can remove the absolute value operator because $V^*(s_t)$ is by definition always at least as large as $\tilde{V}_{\text{ac}}(s_t)$. Since the optimal action chunking policy, by definition, attains

equally good or better values (over $\mathcal{S}$) represented by V_{ac} , and the optimal policy π^* also attains equally good or better value (*i.e.*, V^*) compared to that of the optimal action chunking policy π_{ac}^* (*i.e.*, V_{ac}^*), the following inequality holds for all $s_t \in \text{supp}(\mathcal{D}^*)$:

$$V^*(s_t) \geq V_{\text{ac}}^*(s_t) \geq \tilde{V}_{\text{ac}}(s_t). \quad (\text{D.39})$$

Therefore,

$$V_{\text{ac}}^*(s_t) - V^*(s_t) \leq \tilde{V}_{\text{ac}}(s_t) - V^*(s_t) \leq \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)}, \quad (\text{D.40})$$

as desired. □

D.5 Proof of Corollary 2

In this section, we provide the proof for Corollary 2 (restated below).

Corollary 2 (Worst-case Optimality Gap for Action Chunking Policy) For any $h > 1, \gamma \in [0, 1), \varepsilon_h \in [0, 1/2]$, there exists an MDP $\mathcal{M}$ whose optimal policy π^* induces a data distribution $\mathcal{D}^*$ that is weakly ε_h -open-loop consistent, such that for some $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V_{\text{ac}}^*(s_t) = \frac{\gamma \varepsilon_h}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)}. \quad (5.13)$$

Proof. To show this, we need a slightly more complicated MDP (compared to the $2h$ -state MDP we use in the proof Section D.3). The MDP we construct for this proof is a $(3h - 1)$ -state MDP as illustrated in Figure D.2.

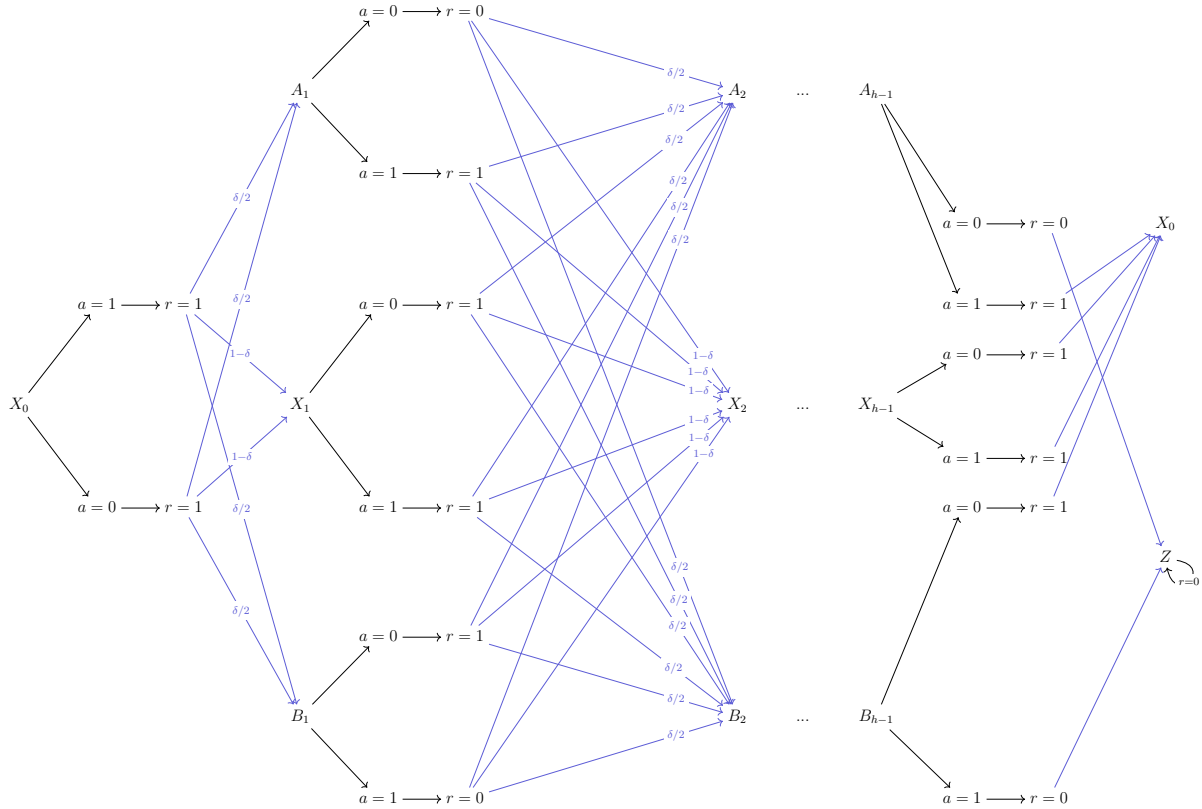


Figure D.2: A $(3h - 1)$ -state MDP that is constructed to meet the upper bound in Corollary 1.

The optimal policy we pick is described as follows:

$$\begin{aligned}
\pi^*(a = 0 \mid X_i) &= 1/2 \\
\pi^*(a = 1 \mid X_i) &= 1/2 \\
\pi^*(a = 1 \mid A_i) &= 1 \\
\pi^*(a = 0 \mid B_i) &= 1/2
\end{aligned} \tag{D.41}$$

This induces the following state distribution,

$$\begin{aligned}
P_{\mathcal{D}^*}(s_{t+i} = A_i \mid s_t = X_0) &= P_{\mathcal{D}^*}(s_{t+i} = B_i \mid s_t = X_0) \\
&= P_{\mathcal{D}^*}^\circ(s_{t+i} = A_i \mid s_t = X_0) = P_{\mathcal{D}^*}^\circ(s_{t+i} = B_i \mid s_t = X_0) = \delta/2, \\
P_{\mathcal{D}^*}(s_{t+i} = X_i \mid s_t = X_0) &= P_{\mathcal{D}^*}^\circ(s_{t+i} = X_i \mid s_t = X_0) = 1 - \delta,
\end{aligned} \tag{D.42}$$

and a fully factorized distribution for the action chunk,

$$P_{\mathcal{D}^*}^\circ(a_{t+i} = 0 \mid s_t) = P_{\mathcal{D}^*}^\circ(a_{t+i} = 0 \mid s_t, a_{t:t+i}) = \frac{1}{2}(\delta_{a=0} + \delta_{a=1}). \tag{D.43}$$

Now, we derive the condition on δ when the optimal data $\mathcal{D}^*$ is ε_h -open-loop consistent. We start by calculating the TV distance discrepancy for the future state-action distribution:

$$\begin{aligned}
&D_{\text{TV}}(P_{\mathcal{D}^*}^{\text{open}}(s_{t+i}, a_{t+i} \mid s_t) \parallel P_{\mathcal{D}^*}(s_{t+i}, a_{t+i} \mid s_t)) \\
&= \frac{1}{2} \left\| \begin{bmatrix} 0 & \delta/2 \\ (1-\delta)/2 & (1-\delta)/2 \\ \delta/2 & 0 \end{bmatrix} - \begin{bmatrix} \delta/4 & \delta/4 \\ (1-\delta)/2 & (1-\delta)/2 \\ \delta/4 & \delta/4 \end{bmatrix} \right\|_{1,1} \\
&= \delta/2.
\end{aligned} \tag{D.44}$$

In the second line of the equations above, each row in the matrix corresponds to a distinct action $a_{t+i} \in \{0, 1\}$ and each row in the matrix corresponds to a distinct state $s_{t+i} \in \{A_i, X_i, B_i\}$.

Next, we calculate the TV distance discrepancy for s_{t+h} :

$$\begin{aligned}
&D_{\text{TV}}(P_{\mathcal{D}^*}^{\text{open}}(s_{t+h} \mid s_t) \parallel P_{\mathcal{D}^*}(s_{t+h} \mid s_t)) \\
&= \frac{1}{2} \left\| \begin{bmatrix} 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 - \delta/2 & \delta/2 \end{bmatrix} \right\|_1 \\
&= \delta/2.
\end{aligned} \tag{D.45}$$

In the second line of the equations above, each element in the vector corresponds to a distinct state $s_{t+h} \in \{X_0, Z\}$. Up to now, we have concluded that $\mathcal{D}^*$ is $(\delta/2)$ -open-loop consistent.

Due to the symmetric structure of this MDP, it is clear that any action chunking policy $\pi_{\text{ac}}(X_0) = a_{t:t+h}$ with $a_{t:t+h} \in \{0, 1\}$ is optimal and achieves the following value:

$$\begin{aligned} V_{\text{ac}}^*(X_0) &= 1 + (1 - \delta/2) \left[\frac{\gamma - \gamma^h}{1 - \gamma} + \gamma^h V_{\text{ac}}^*(X_0) \right] \\ &= \frac{(1 - \gamma) + (1 - \delta/2)(\gamma - \gamma^h)}{(1 - \gamma)(1 - (1 - \delta/2)\gamma^h)}. \end{aligned} \tag{D.46}$$

The optimal closed-loop policy can achieve the maximum possible return

$$V^*(X_0) = \frac{1}{1 - \gamma}. \tag{D.47}$$

Therefore, with $\varepsilon_h = \delta/2$, the optimality gap achieved by this $(3h - 1)$ -state MDP is

$$V^*(X_0) - V_{\text{ac}}^*(X_0) = \frac{\varepsilon_h \gamma}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)}, \tag{D.48}$$

as desired. □

D.6 Proof of Proposition 1

In this section, we provide the proof for Proposition 1 (restated below).

Proposition 1 (AC Q-Learning under Weak OLC) For any $h > 1, \gamma \in [0, 1), c \in [0, 1/2), \varepsilon_h \in (0, 1/2)$, there exists an MDP $\mathcal{M}$, weakly ε_h -open-loop consistent $\mathcal{D}$ and $\mathcal{D}^*$ with $\text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h})) \supseteq \text{supp}(P_{\mathcal{D}^*}(s_t, a_{t:t+h}))$, such that for some $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V_{\text{ac}}^+(s_t) = V_{\text{ac}}^*(s_t) - V_{\text{ac}}^+(s_t) = \frac{\gamma^c}{1 - \gamma}. \quad (5.16)$$

Proof. To prove this theorem, we show an example where the optimal action chunking policy defined in Equation (5.15) can be highly sub-optimal in the absence of the strong open-loop consistency condition.

We define an MDP as follows. Let $\mathcal{S} = \{A, B, C, D, E, Z\}$ and $\mathcal{A} = \{0, 1\}$. Define the transition dynamics and reward function as shown in the diagram below (Figure D.3):

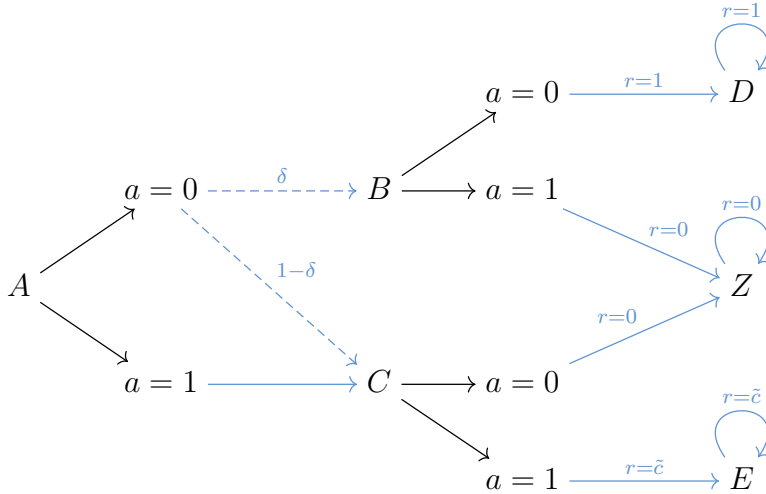


Figure D.3: A 6-state MDP that is constructed to illustrate the pathological failure mode of action chunking Q-learning under weak open-loop consistency.

where $\delta, \tilde{c} \in [0, 1)$ are real numbers and dotted lines denote stochastic transitions. The reward function depends only on states ($r(A) = r(B) = r(C) = r(F) = 0$, $r(D) = 1$, and $r(G) = c$). Assume that the dataset is collected by a policy $\pi_{\mathcal{D}}$ defined as $\pi_{\mathcal{D}}(A) = 0$ (with probability θ) or 1 (with probability $(1 - \theta)$), $\pi_{\mathcal{D}}(B) = 0$ (with probability 1), $\pi_{\mathcal{D}}(C) = 1$ (with probability 1), and $\pi_{\mathcal{D}}(D) = \pi_{\mathcal{D}}(Z) = \pi_{\mathcal{D}}(E) = 0$ (with probability 1).

Let π^* be the deterministic policy with $\pi(A) = 1, \pi(C) = 1, \pi(E) = 0$. Then, it is clear that $\text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h})) \supseteq \text{supp}(P_{\mathcal{D}^*}(s_t, a_{t:t+h})) = \{(s_t = A, a_t = 1, a_{t+1} = 1, a_{t+2:t+h} = 0)\}$. Now we are left to show that $\mathcal{D}$ is ε_h -open-loop consistent.

Since D, E, Z are all self-loops, we only need to analyze the first two actions in the action chunk; the rest is all $a_{t+2:t+h} = 0$ and does not affect the value.

We can compute the following:

$$\begin{aligned} P_{\mathcal{D}}(A, (0, 0)) &= D, \quad R(A, (0, 0)) = \gamma, \\ P_{\mathcal{D}}(A, (0, 1)) &= E, \quad R(A, (0, 1)) = 2c\gamma, \\ P_{\mathcal{D}}(A, (1, 1)) &= E, \quad R(A, (1, 1)) = 2c\gamma, \end{aligned} \tag{D.49}$$

where we denote action chunks as a tuple and slightly abuse notation to denote deterministic outputs of $P_{\mathcal{D}}(\cdot \mid s_0, a_{0:2})$ (e.g., $P_{\mathcal{D}}(A, (0, 0)) = D$ indicates that all length-2 trajectories in $\mathcal{D}$ from state A with $a_0 = a_1 = 0$ have $s_2 = D$ with probability 1).

We can similarly compute the marginal state distributions as follows:

$$\begin{bmatrix} P_{\mathcal{D}}(D \mid A) \\ P_{\mathcal{D}}(Z \mid A) \\ P_{\mathcal{D}}(E \mid A) \end{bmatrix} = \begin{bmatrix} \theta\delta \\ 0 \\ 1 - \theta\delta \end{bmatrix}, \tag{D.50}$$

and

$$\begin{bmatrix} P_{\mathcal{D}}(s_{t+1} = B, a_t = 0 \mid s_t = A) & P_{\mathcal{D}}(s_{t+1} = B, a_t = 1 \mid s_t = A) \\ P_{\mathcal{D}}(s_{t+1} = C, a_t = 0 \mid s_t = A) & P_{\mathcal{D}}(s_{t+1} = C, a_t = 1 \mid s_t = A) \end{bmatrix} = \begin{bmatrix} \theta\delta & 0 \\ 0 & 1 - \theta\delta \end{bmatrix}. \tag{D.51}$$

The marginal probability distribution of the action chunks is

$$\begin{aligned} P_{\mathcal{D}}(a_{0,1} = (0, 0) \mid A) &= \theta\delta, \\ P_{\mathcal{D}}(a_{0,1} = (0, 1) \mid A) &= (1 - \delta)\theta, \\ P_{\mathcal{D}}(a_{0,1} = (1, 1) \mid A) &= 1 - \theta. \end{aligned} \tag{D.52}$$

The induced $P_{\mathcal{D}}^{\circ}$ is then

$$\begin{bmatrix} P_{\mathcal{D}}^{\circ}(s_{t+h} = D \mid s_t = A) \\ P_{\mathcal{D}}^{\circ}(s_{t+h} = Z \mid s_t = A) \\ P_{\mathcal{D}}^{\circ}(s_{t+h} = E \mid s_t = A) \end{bmatrix} = \begin{bmatrix} \theta\delta^2 \\ (1 - \delta)\theta\delta \\ (1 - \delta)^2\theta + (1 - \theta) \end{bmatrix}, \tag{D.53}$$

and

$$\begin{aligned} &\begin{bmatrix} P_{\mathcal{D}}^{\circ}(s_{t+1} = B, a_{t+1} = 0 \mid s_t = A) & P_{\mathcal{D}}^{\circ}(s_{t+1} = B, a_{t+1} = 1 \mid s_t = A) \\ P_{\mathcal{D}}^{\circ}(s_{t+1} = C, a_{t+1} = 0 \mid s_t = A) & P_{\mathcal{D}}^{\circ}(s_{t+1} = C, a_{t+1} = 1 \mid s_t = A) \end{bmatrix} \\ &= \begin{bmatrix} \delta^2\theta & (1 - \delta)\delta\theta \\ \delta(1 - \delta)\theta & (1 - \delta)^2\theta + 1 - \theta \end{bmatrix}. \end{aligned} \tag{D.54}$$

We can then compute

$$\begin{aligned} D_{\text{TV}}(P_{\mathcal{D}}^{\circ}(s_{t+1}, a_{t+1} \mid s_t = A) \parallel P_{\mathcal{D}}(s_{t+1}, a_{t+1} \mid s_t = A)) &= 2\theta\delta(1 - \delta) \\ D_{\text{TV}}(P_{\mathcal{D}}^{\circ}(s_{t+h} \mid s_t = A) \parallel P_{\mathcal{D}}(s_{t+h} \mid s_t = A)) &= \frac{3}{2}\theta\delta(1 - \delta) \end{aligned} \quad (\text{D.55})$$

Note that we only need to check $t + 1$ time step for the state-action distribution because

$$\begin{aligned} D_{\text{TV}}(P_{\mathcal{D}}^{\circ}(s_{t+h'}, a_{t+h'} \mid s_t = A) \parallel P_{\mathcal{D}}(s_{t+h'}, a_{t+h'} \mid s_t = A)) &= \\ D_{\text{TV}}(P_{\mathcal{D}}^{\circ}(s_{t+h} \mid s_t = A) \parallel P_{\mathcal{D}}(s_{t+h} \mid s_t = A)), \quad \forall h' > 1, \end{aligned} \quad (\text{D.56})$$

coming from the fact that the rest of the action chunks are all constant 0 (*i.e.*, $a_{t+2:t+h} = 0$).

Now, we can set δ to be solution of $\varepsilon_h = 2(1 - \delta)\delta\theta$, such that $\mathcal{D}$ is ε_h -open-loop consistent as required by the assumption.

Next, we analyze the action chunking policy π_{ac}^+ learned from this ε_h -open-loop consistent data distribution. We start by calculating $\hat{Q}_{\text{ac}}^+$ as follows:

$$\begin{aligned} \hat{Q}_{\text{ac}}^+(A, (0, 0)) &= \frac{\gamma}{1 - \gamma}, \\ \hat{Q}_{\text{ac}}^+(A, (0, 1)) &= \frac{\tilde{c}\gamma}{1 - \gamma}, \\ \hat{Q}_{\text{ac}}^+(A, (1, 1)) &= \frac{\tilde{c}\gamma}{1 - \gamma}. \end{aligned} \quad (\text{D.57})$$

The optimal action chunking policy is then $\hat{\pi}_{\text{ac}}^+(A) = (0, 0)$ (Equation (5.15)) because $c < 1$.

The true value of this action chunking policy is

$$V_{\text{ac}}^+(A) = \frac{\delta\gamma}{1 - \gamma} \quad (\text{D.58})$$

As long as $\tilde{c} > \delta$, the optimal strategy in this MDP is to always choose $(a_0, a_1) = (1, 1)$, in which case the agent receives a constant return:

$$V^*(A) = V_{\text{ac}}^*(A) = \frac{\tilde{c}\gamma}{1 - \gamma}. \quad (\text{D.59})$$

The optimality gap in this example is therefore

$$V^*(A) - V_{\text{ac}}^+(A) = (\tilde{c} - \delta)\frac{\gamma}{1 - \gamma}. \quad (\text{D.60})$$

Finally, we solve $\varepsilon_h = 2(1 - \delta)\delta\theta$ and pick the smaller solution:

$$\delta = \frac{1 - \sqrt{1 - 2\varepsilon_h/\theta}}{2}. \quad (\text{D.61})$$

If we set $\theta = 2\varepsilon_h$ and $\tilde{c} = c + \frac{1}{2}$, then we get

$$V^*(A) - V_{\text{ac}}^+(A) = \frac{c\gamma}{1 - \gamma}, \quad (\text{D.62})$$

as desired.

As a small extra note, the last step is where the assumption $\varepsilon_h > 0$ becomes necessary, since otherwise the term $2\varepsilon_h/\theta$ (with $\theta = 2\varepsilon_h$) would be undefined. $\square$

D.7 Proof of Theorem 3

In this section, we provide the proof for Theorem 3 (restated below).

Theorem 3 (AC Q-Learning under Strong OLC) If $\mathcal{D}$ and $\mathcal{D}^*$ are strongly ε_h -open-loop consistent, and $\text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h})) \supseteq \text{supp}(P_{\mathcal{D}^*}(s_t, a_{t:t+h}))$, then $\forall s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V_{\text{ac}}^+(s_t) \leq \frac{\varepsilon_h \gamma}{1 - \gamma} \left[\frac{2}{1 - (1 - 2\varepsilon_h)\gamma^h} + \frac{1}{1 - (1 - \varepsilon_h)\gamma^h} \right] \leq 3\varepsilon_h H \bar{H}, \quad (5.17)$$

where V^* is the value of a closed-loop optimal policy and V_{ac}^+ is the *true* value of π_{ac}^+ .

Proof of Theorem 3. We start by constructing a bound between $\hat{Q}_{\text{ac}}^+$ and Q_{ac}^* , the solution of the following bellman equation:

$$Q_{\text{ac}}^*(s_t, a_{t:t+h}) = \mathbb{E}_{s_{t+1:t+h+1} \sim P_{\mathcal{D}}^o(\cdot | s_t, a_{t:t+h})} \left[R_{t:t+h} + \gamma^h \max_{a_{t+h:t+2h}} Q_{\text{ac}}^*(s_{t+h}, a_{t+h:t+2h}) \right]. \quad (\text{D.63})$$

Intuitively, Q_{ac}^* is the Q-function of the optimal action chunking policy π_{ac}^* that can be learned from $\mathcal{D}$. Because $\text{supp}(\mathcal{D}) \supseteq \text{supp}(\mathcal{D}^*)$, π_{ac}^* is at least as good as $\tilde{\pi}_{\text{ac}}$, the action chunking policy obtained by behavior cloning $\mathcal{D}^*$. Bounding the difference between $\hat{Q}_{\text{ac}}^+$ and Q_{ac}^* allows us to leverage the bound in Corollary 1 to form a bound between $\hat{V}_{\text{ac}}^+$ and V^* .

Since $\mathcal{D}$ is strongly ε_h -open-loop consistent,

$$D_{\text{TV}}(T(s_{t+h'} | s_t, a_{t:t+h'}) \parallel P_{\mathcal{D}}(s_{t+h'} | s_t, a_{t:t+h})) \leq \varepsilon_h, \forall h' \in \{1, \dots, h-1\}. \quad (\text{D.64})$$

Since $\mathcal{D}^*$ is also strongly ε_h -open-loop consistent,

$$D_{\text{TV}}(T(s_{t+h'} | s_t, a_{t:t+h'}) \parallel P_{\mathcal{D}^*}(s_{t+h'} | s_t, a_{t:t+h})) \leq \varepsilon_h, \forall h' \in \{1, \dots, h-1\}. \quad (\text{D.65})$$

Using the transitive property of TV distance, we have

$$D_{\text{TV}}(P_{\mathcal{D}}(s_{t+h'} | s_t, a_{t:t+h}) \parallel P_{\mathcal{D}^*}(s_{t+h'} | s_t, a_{t:t+h})) \leq 2\varepsilon_h, \forall h' \in \{1, \dots, h-1\}. \quad (\text{D.66})$$

Now, for the h -step reward, we have

$$\begin{aligned} & \left| \mathbb{E}_{P_{\mathcal{D}}(\cdot | s_t, a_{t:t+h})} [R_{t:t+h}] - \mathbb{E}_{P_{\mathcal{D}^*}(\cdot | s_t, a_{t:t+h})} [R_{t:t+h}] \right| \\ & \leq \sum_{h'=1}^{h-1} \left[\gamma^{h'} D_{\text{TV}}(P_{\mathcal{D}}(s_{t+h'} | s_t, a_{t:t+h}) \parallel P_{\mathcal{D}^*}(s_{t+h'} | s_t, a_{t:t+h})) \right] \\ & \leq \frac{2(\gamma - \gamma^h)\varepsilon_h}{1 - \gamma}. \end{aligned} \quad (\text{D.67})$$

Similarly, for the value h -step into the future, we can use Lemma 2 to obtain the following bound:

$$\begin{aligned} & \left| \mathbb{E}_{s_{t+h} \sim P_{\mathcal{D}}(s_{t+h}|s_t)} [V^*(s_{t+h})] - \mathbb{E}_{s_{t+h} \sim P_{\mathcal{D}^*}(s_{t+h}|s_t)} [\hat{V}_{\text{ac}}^+(s_{t+h})] \right| \\ & \leq 2\varepsilon_h + (1 - 2\varepsilon_h) \sup_{s_{t+h} \in \mathcal{D}^*} |V^*(s_{t+h}) - \hat{V}_{\text{ac}}^+(s_{t+h})|. \end{aligned} \quad (\text{D.68})$$

We define $Q^*(s_t, a_{t:t+h})$ to be

$$Q^*(s_t, a_{t:t+h}) := \mathbb{E}_{P_{\mathcal{D}^*}(\cdot|s_t, a_{t:t+h})} [R_{t:t+h} + \gamma^h V^*(s_{t+h})]. \quad (\text{D.69})$$

It is clear that

$$V^*(s_t) = \mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}^*}} [Q^*(s_t, a_{t:t+h})]. \quad (\text{D.70})$$

Combining the bound for the h -step reward and the bound on the value for s_{t+h} , for all $s_t, a_{t:t+h} \in \text{supp}(P_{\mathcal{D}^*}(s_t, a_{t:t+h}))$,

$$\begin{aligned} \Delta(s_t, a_{t:t+h}) &= Q^*(s_t, a_{t:t+h}) - \hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}) \\ &\leq 2\varepsilon_h \gamma^h + \frac{2(\gamma - \gamma^h)\varepsilon_h}{1 - \gamma} + (1 - 2\varepsilon_h) \gamma^h (V^*(s_{t+h}) - \hat{V}_{\text{ac}}^+(s_{t+h})) \\ &\leq \frac{2\varepsilon_h \gamma}{1 - \gamma} + (1 - 2\varepsilon_h) \gamma^h \left(\mathbb{E}_{P_{\mathcal{D}^*}} [Q^*(s_{t+h}, a_{t+h:t+2h})] - \sup_{a_{t+h:t+2h}} \hat{Q}_{\text{ac}}^+(s_{t+h}, a_{t+h:t+2h}) \right) \\ &\leq \frac{2\varepsilon_h \gamma}{1 - \gamma} + (1 - 2\varepsilon_h) \gamma^h \left(\mathbb{E}_{P_{\mathcal{D}^*}} [\hat{Q}_{\text{ac}}^+(s_{t+h}, a_{t+h:t+2h}) + \Delta(s_{t+h}, a_{t+h:t+2h})] - \sup_{a_{t+h:t+2h}} \hat{Q}_{\text{ac}}^+(s_{t+h}, a_{t+h:t+2h}) \right) \\ &\leq \frac{2\varepsilon_h \gamma}{1 - \gamma} + (1 - 2\varepsilon_h) \gamma^h \sup_{s_{t+h}, a_{t+h:t+2h}} [\Delta(s_{t+h}, a_{t+h:t+2h})], \end{aligned} \quad (\text{D.71})$$

which can be recursively expanded to obtain

$$V^*(s_t) - \hat{V}_{\text{ac}}^+(s_t) \leq \frac{2\varepsilon_h \gamma}{(1 - \gamma)(1 - (1 - 2\varepsilon_h)\gamma^h)}. \quad (\text{D.72})$$

By Theorem 1, for all $s_t \in \text{supp}(\mathcal{D})$,

$$\left| \hat{V}_{\text{ac}}^+(s_t) - V_{\text{ac}}^+(s_t) \right| \leq \frac{\varepsilon_h \gamma}{(1 - \gamma)(1 - (1 - \varepsilon_h)\gamma^h)}. \quad (\text{D.73})$$

Combining the two inequalities above, for all $s_t \in \text{supp}(\mathcal{D}^*)$,

$$V^*(s_t) - V_{\text{ac}}^+(s_t) \leq \frac{\varepsilon_h \gamma}{1 - \gamma} \left[\frac{2}{1 - (1 - 2\varepsilon_h)\gamma^h} + \frac{1}{1 - (1 - \varepsilon_h)\gamma^h} \right]. \quad (\text{D.74})$$

□

D.8 Proof of Theorem 4

In this section, we provide the proof for Theorem 4 (restated below).

Theorem 4 (Worst-case Analysis of Q-Learning with Action Chunking Policy on Off-policy Data) For any $h > 1, \gamma \in (0, 1), \varepsilon_h \in (0, 1/5), c_1 \in (0, \varepsilon_h/2)$, and $c_2 \in (0, 2\varepsilon_h\gamma)$, there exists an MDP $\mathcal{M}$ and strongly ε_h -open-loop consistent data distributions $\mathcal{D}$ and $\mathcal{D}^*$ with $\text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h})) \supseteq \text{supp}(P_{\mathcal{D}^*}(s_t, a_{t:t+h}))$, such that for some $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V_{\text{ac}}^+(s_t) = \frac{2\varepsilon_h\gamma - c_2}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)} + \frac{\varepsilon_h\gamma}{(1-\gamma)(1-(1-\varepsilon_h-c_1)\gamma^h)}, \quad (5.18)$$

where V^* is the value of an optimal policy and V_{ac}^+ is the *true* value of π_{ac}^+ .

As $c_1, c_2 \rightarrow 0$,

$$V^*(s_t) - V_{\text{ac}}^+(s_t) \rightarrow \frac{\varepsilon_h\gamma}{1-\gamma} \left[\frac{2}{1-(1-2\varepsilon_h)\gamma^h} + \frac{1}{1-(1-\varepsilon_h)\gamma^h} \right]. \quad (5.19)$$

The examples in the following proof of Theorem 4 (available in Section D.8) provide insights on the factor of 3 in $V^* - V_{\text{ac}}^+ \leq 3\varepsilon_h H \bar{H}$ (with $H = 1/(1-\gamma), \bar{H} = 1/(1-\gamma^h)$) is necessary. In particular, the worst case can be roughly seen as a combination of the two main results that we have presented so far:

1. $V^* - V_{\text{ac}}^* \approx \varepsilon_h H \bar{H}$ (Corollary 1, Corollary 2): the optimal action chunking policy is $(\varepsilon_h H^2)$ -sub-optimal due to its inability to react to environment stochasticity, quantified by the strongly- ε_h open-loop consistency of $\mathcal{D}^*$.
2. $V_{\text{ac}}^* - \hat{V}_{\text{ac}}^+ \approx \varepsilon_h H \bar{H}$ (a transformation of Theorem 1 and Theorem 2 on the optimal action chunking policy π_{ac}^*): the value *under-estimation* bias can incur another factor of $\varepsilon_h H \bar{H}$ bringing up the sub-optimality of $\hat{V}_{\text{ac}}^+$ to at most $2\varepsilon_h H \bar{H}$, and finally,
3. $\hat{V}_{\text{ac}}^+ - V_{\text{ac}}^+ \approx \varepsilon_h H \bar{H}$ (Theorem 1, Theorem 2): the action chunking value function may prefer an *overestimated* action chunking policy π_{ac}^+ where its actual value is again $\varepsilon_h H \bar{H}$ from its estimated value, resulting in a total sub-optimality of $3\varepsilon_h H \bar{H}$.

Our construction (in the proof of Theorem 4) directly builds on the above insights by using a 2-part MDP where the first part corresponds to an $(\varepsilon_h H \bar{H})$ -underestimated action chunking policy that has a $(\varepsilon_h H \bar{H})$ -optimality gap from the optimal closed-loop policy and the second part corresponds to an $(\varepsilon_h H \bar{H})$ -overestimated action chunking policy that has a $(3\varepsilon_h H \bar{H})$ -optimality gap that is preferred by the value function.

Before we start our main proof, we first introduce a Lemma that helps simplifies the inequalities.

Lemma 5 (Optimality gap comparator) For any $\tilde{\gamma} \in [0, 1]$ and $0 < \varepsilon_1 < \varepsilon_2 < 1$,

$$\frac{\varepsilon_1}{1 - (1 - \varepsilon_1)\tilde{\gamma}} < \frac{\varepsilon_2}{1 - (1 - \varepsilon_2)\tilde{\gamma}}. \quad (\text{D.75})$$

Proof.

$$\begin{aligned} 0 &< (1 - \gamma)(\varepsilon_2 - \varepsilon_1) \\ &= \varepsilon_2 - \varepsilon_2\tilde{\gamma} - \varepsilon_1 + \varepsilon_1\tilde{\gamma} \\ &= \varepsilon_2 - \varepsilon_2\tilde{\gamma} + \varepsilon_1\varepsilon_2\tilde{\gamma} - \varepsilon_1 + \varepsilon_1\tilde{\gamma} - \varepsilon_1\varepsilon_2\tilde{\gamma} \\ &= \varepsilon_2(1 - (1 - \varepsilon_1)\tilde{\gamma}) - \varepsilon_1(1 - (1 - \varepsilon_2)\tilde{\gamma}) \end{aligned} \quad (\text{D.76})$$

Since $1 - (1 - \varepsilon_1)\tilde{\gamma} > 0$ and $1 - (1 - \varepsilon_2)\tilde{\gamma} > 0$, we can divide both sides by $(1 - (1 - \varepsilon_1)\tilde{\gamma})(1 - (1 - \varepsilon_2)\tilde{\gamma})$ to get

$$0 < \frac{\varepsilon_2}{1 - (1 - \varepsilon_2)\tilde{\gamma}} - \frac{\varepsilon_1}{1 - (1 - \varepsilon_1)\tilde{\gamma}}, \quad (\text{D.77})$$

as desired. $\square$

Now, we begin the main proof as follows.

Proof of Theorem 4. We prove by constructing the following $(2h + 4)$ -state MDP where the agent can take any of the three actions $\{0, 1, 2\}$ at each state (see a diagram in Figure D.4).

Notations: we start by introducing some abbreviations for all action chunks that appear in this proof:

$$\begin{aligned} a_{t:t+h}^* &= (0, 0, 0, \dots, 0) \\ a_{t:t+h}^\diamond &= (0, 1, 0, \dots, 0) \\ a_{t:t+h}^\bullet &= (0, 2, 0, \dots, 0) \\ a_{t:t+h}^\triangle &= (1, 1, 1, \dots, 1) \\ a_{t:t+h}^\circ &= (1, 0, 1, \dots, 1) \\ a_{t:t+h}^\times &= (1, 2, 1, \dots, 1) \end{aligned} \quad (\text{D.78})$$

The first three action chunks $a_{t:t+h}^*, a_{t:t+h}^\diamond, a_{t:t+h}^\bullet$ are only possible in the top branch and the last three action chunks $a_{t:t+h}^\triangle, a_{t:t+h}^\circ, a_{t:t+h}^\times$ are only possible in the bottom branch because the first action in the action chunk deterministically divides it into the two branches.

Among these action chunks, it is clear by inspection that $\pi_{ac}(X_0) = (0, 0, \dots, 0)$ is the optimal action chunking policy, and thus we directly use ‘ $\star$ ’ to denote $a_{t:t+h}^* = (0, 0, \dots, 0)$.

$a_{t:t+h}^\Delta$ is also of great importance: as we will show later, $\pi_{ac}^+(X_0) = a_{t:t+h}^\Delta$. The *actual* values and *nominal/estimated* values for these action chunks are $(V_{ac}^\star, V_{ac}^\diamond, V_{ac}^\bullet, V_{ac}^\Delta, V_{ac}^\circ, V_{ac}^\times)$ and $(\hat{V}_{ac}^\star, \hat{V}_{ac}^\diamond, \hat{V}_{ac}^\bullet, \hat{V}_{ac}^\Delta, \hat{V}_{ac}^\circ, \hat{V}_{ac}^\times)$ respectively. Much of the focus of this proof is to calculate the optimality gap, which is the difference between the optimal closed-loop value and the action chunking policy value (either estimated or actual):

$$\text{actual optimality gap: } V^\star(X_0) - V_{ac}^{[\cdot]}(X_0) \quad (\text{D.79})$$

$$\text{nominal optimality gap: } V^\star(X_0) - \hat{V}_{ac}^{[\cdot]}(X_0) \quad (\text{D.80})$$

High-level proof sketch: The MDP contains two branches: a top branch where (as we will show) both the optimal policy $\pi^\star$ and the optimal action chunking policy $\pi_{ac}^\star$ take, and a bottom branch where (as we will also show) the learned action chunking policy π_{ac}^+ takes. The key idea of the construction is that for the top branch, we have

$$V^\star(X_0) - \hat{V}_{ac}^\star(X_0) \approx \frac{2\varepsilon_h\gamma}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)}, \quad (\text{D.81})$$

and for the bottom branch, we have

$$\hat{V}_{ac}^\star(X_0) < \hat{V}_{ac}^+(X_0) \approx V_{ac}^+(X_0) + \frac{\varepsilon_h\gamma}{(1-\gamma)(1-(1-\varepsilon_h)\gamma^h)}. \quad (\text{D.82})$$

Combining these two together gives

$$V^\star(X_0) - V_{ac}^+(X_0) \approx \frac{\varepsilon_h\gamma}{1-\gamma} \left[\frac{2}{1-(1-2\varepsilon_h)\gamma^h} + \frac{1}{1-(1-\varepsilon_h)\gamma^h} \right]. \quad (\text{D.83})$$

We use ‘ $\approx$ ’ because the equalities are not strictly achievable but (as we will show) can be made arbitrarily close.

The proof can be roughly divided into the following steps (we use ‘ $\approx$ ’ to help illustrate the high-level idea below and use more precise argument in the actual proof):

1. **MDP description:** we formally describe the transition dynamics T and the reward function r for each state-action pair for both the top and the bottom branches.
2. **Strong ε_h -open-loop consistency of $\mathcal{D}^\star$:** we then check the strong open-loop consistency assumption for $\mathcal{D}^\star$.
3. **Data distribution $\mathcal{D}_{\text{top}}$ for the top branch:** we use a mixture data distribution from two policies to construct $\mathcal{D}_{\text{top}}$.
4. **Strong ε_h -open-loop consistency of $\mathcal{D}_{\text{top}}$:** we then check that the constructed data distribution of the top branch satisfies the strongly open-loop consistency assumption. Note that we can do so separately for the top and the bottom because these two distributions have non-overlapping support in $a_{t:t+h}$.

5. **The optimality gap and value estimation error for the top branch:** we prove that $V^*(X_0) - V_{ac}^*(X_0) = \frac{\varepsilon_h \gamma}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)}$ and $V^*(X_0) - \hat{V}_{ac}^*(X_0) = \frac{2\varepsilon_h \gamma}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)}$ and the other two possible action chunks $a_{t:t+h}^\diamond = (0, 1, 0, \dots)$ and $a_{t:t+h}^\bullet = (0, 2, 0, \dots)$ both admit lower estimated values compared to $a_{t:t+h}^*$: $\hat{V}_{ac}^\diamond(X_0) < \hat{V}_{ac}^*(X_0)$ and $\hat{V}_{ac}^\bullet(X_0) < \hat{V}_{ac}^*(X_0)$.
6. **Data distribution $\mathcal{D}_{\text{bottom}}$ for the bottom branch:** we again use a mixture data distribution from two different policies to construct $\mathcal{D}_{\text{bottom}}$.
7. **Strong ε_h -open-loop consistency of $\mathcal{D}_{\text{bottom}}$:** we then check that the constructed data distribution of the bottom branch satisfies the strongly open-loop consistency assumption.
8. **The optimality gap and value estimation error for the bottom branch:** we prove that $V^*(X_0) - \hat{V}_{ac}^\Delta(X_0) \approx \frac{2\varepsilon_h \gamma}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)}$ and $\hat{V}_{ac}^\Delta(X_0) - V_{ac}^\Delta(X_0) = \frac{\varepsilon_h \gamma}{(1-\gamma)(1-(1-\varepsilon_h)\gamma^h)}$, and the other two possible action chunks $a_{t:t+h}^\circ = (1, 0, 0, \dots)$ and $a_{t:t+h}^\times = (1, 2, 0, \dots)$ both admit lower estimated values compared to $a_{t:t+h}^\Delta$: $\hat{V}_{ac}^\circ(X_0) < \hat{V}_{ac}^\Delta(X_0)$ and $\hat{V}_{ac}^\times(X_0) < \hat{V}_{ac}^\Delta(X_0)$. Moreover $a_{t:t+h}^*$ also admits a lower estimated value compared to $a_{t:t+h}^\Delta$: $\hat{V}_{ac}^*(X_0) < \hat{V}_{ac}^\Delta(X_0)$ which proves $\pi_{ac}^+(X_0) = a_{t:t+h}^\Delta$ and thus concluding our proof: $V^*(X_0) - V_{ac}^+(X_0) \approx \frac{2\varepsilon_h \gamma}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)} + \frac{\varepsilon_h \gamma}{(1-\gamma)(1-(1-\varepsilon_h)\gamma^h)}$.

Now we begin our proof as follows.

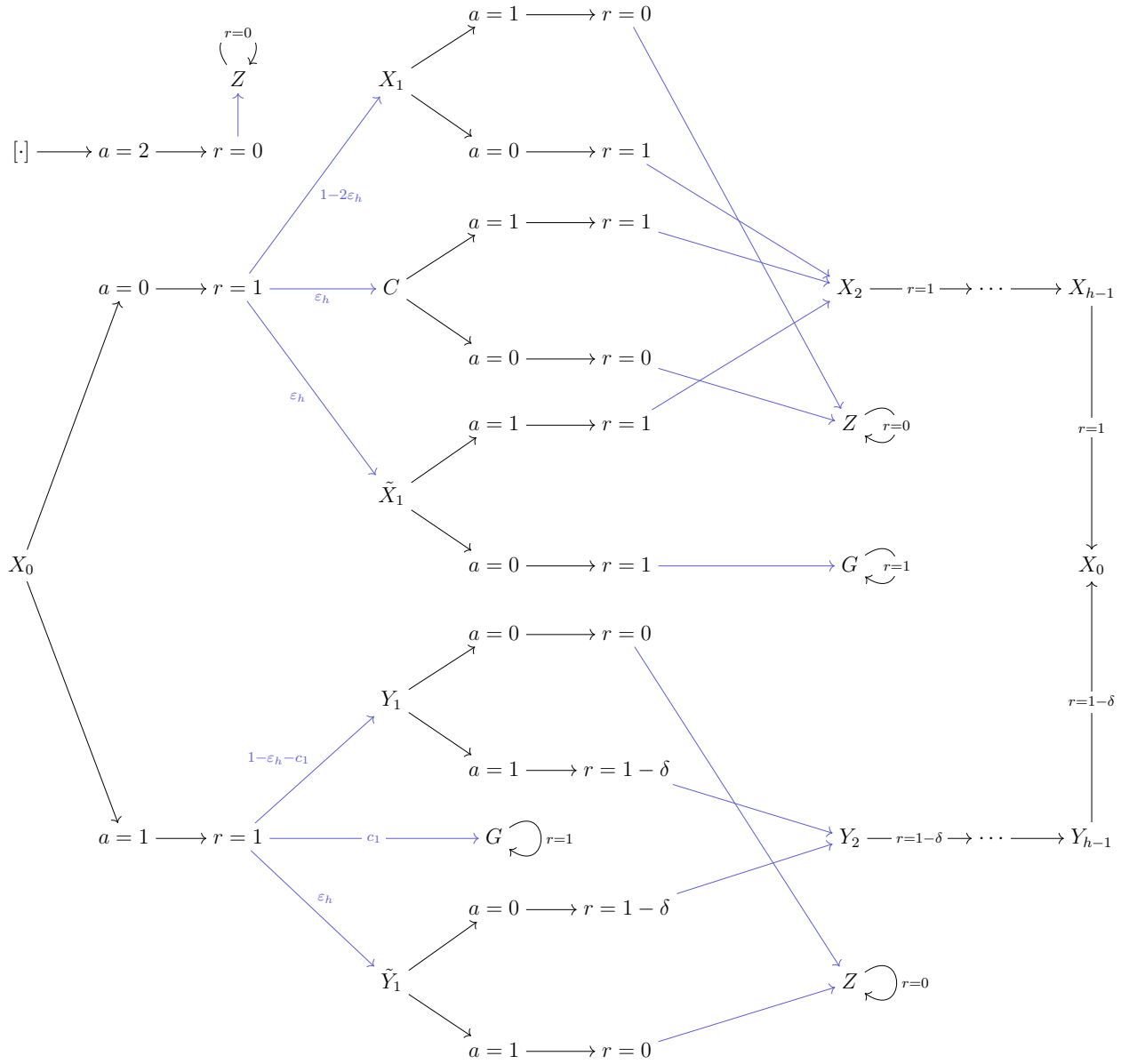


Figure D.4: A $(2h + 4)$ -state MDP that is constructed to meet the exact upper bound optimality gap in Theorem 3. We redraw the same states Z , G , X_0 in multiple locations in the diagram above for better clarity.

Step 1. MDP description (Figure D.4).

The transition function T of the MDP is defined as follows (from left to right):

$$\begin{aligned}
T(Z \mid Z, a) &= T(G \mid G, a) = 1, \quad \forall a, \\
T(Z \mid s, a = 2) &= 1, \quad \forall a, \forall s : s \neq G \\
T(X_1 \mid X_0, a = 0) &= 1 - 2\varepsilon_h \\
T(\tilde{X}_1 \mid X_0, a = 0) &= \varepsilon_h \\
T(C \mid X_0, a = 0) &= \varepsilon_h \\
T(Y_1 \mid X_0, a = 1) &= 1 - \varepsilon_h - c_1 \\
T(\tilde{Y}_1 \mid X_0, a = 1) &= \varepsilon_h \\
T(G \mid X_0, a = 1) &= c_1 \\
T(X_2 \mid X_1, a = 0) &= 1 \\
T(X_2 \mid \tilde{X}_1, a = 1) &= 1 \\
T(X_2 \mid C, a = 1) &= 1 \\
T(Z \mid X_1, a = 1) &= 1 \\
T(Z \mid C, a = 0) &= 1 \\
T(G \mid \tilde{X}_1, a = 0) &= 1 \\
T(Y_2 \mid Y_1, a = 1) &= 1 \\
T(Y_2 \mid \tilde{Y}_1, a = 0) &= 1 \\
T(Z \mid Y_1, a = 0) &= 1 \\
T(Z \mid \tilde{Y}_1, a = 1) &= 1 \\
T(X_{i+1} \mid X_i, a \in \{0, 1\}) &= T(Y_{i+1} \mid Y_i, a \in \{0, 1\}) = 1, \quad \forall i \in \{2, \dots, h-2\} \\
T(X_0 \mid X_{h-1}, a \in \{0, 1\}) &= T(Y_0 \mid Y_{h-1}, a \in \{0, 1\}) = 1
\end{aligned} \tag{D.84}$$

The reward function is defined as

$$\begin{aligned}
r(Z, a) &= 0, \quad \forall a \\
r(G, a) &= 1, \quad \forall a \\
r(s, a = 2) &= 0, \quad \forall s : s \neq G \\
r(X_0, a = 0) &= r(X_0, a = 1) = 1, \\
r(C, a = 1) &= r(X_1, a = 0) = r(\tilde{X}_1, a \in \{0, 1\}) = 1, \\
r(C, a = 0) &= r(X_1, a = 1) = 0, \\
r(Y_1, a = 1) &= r(\tilde{Y}_1, a = 0) = 1 - \delta, \\
r(Y_1, a = 0) &= r(\tilde{Y}_1, a = 1) = 0, \\
r(X_i, a \in \{0, 1\}) &= 1, \quad \forall i \in \{2, \dots, h-1\} \\
r(Y_i, a \in \{0, 1\}) &= 1 - \delta, \quad \forall i \in \{2, \dots, h-1\}
\end{aligned} \tag{D.85}$$

Notably, there are some special states:

- State Z : a self-looping “black hole” state that always gets 0 reward at each time step and thus has a constant value of 0.
- State G : a self-looping “black hole” state that always gets 1 reward at each time step and thus has a constant value of $1/(1 - \gamma)$.
- State X_0 : the special state that branches out based on the action taken. The agent periodically visit this state every h steps unless it has been trapped in either Z or G . As we proceed in the proof, we will encounter factors in the form of $\frac{1}{1-b\gamma^h}$ in the calculation of the optimality gap. These factors come from the agent looping around and revisiting X_0 with b -probability each cycle.

These two absorbing states are important because their values sit at the boundary of the value range of our value function $V(s) \in [0, 1/(1 - \gamma)]$. Shifting the reaching probability from Z to G or the other way around results in the biggest possible difference in the policy value. Our construction hinges on the constructing $\mathcal{D}$ such that

1. $P_{\mathcal{D}}(\cdot \mid s_t, \pi^*(s_t))$ and $T(\cdot \mid s_t, \pi^*(s_t))$ differs by only ε_h (in TV distance as required by the strongly open-loop consistency assumption) where precisely ε_h probability mass is moved from reaching state Z to reaching state G . This causes the $\hat{V}_{ac}^*$ to precisely underestimates the value of V_{ac}^* by $\frac{\varepsilon_h \gamma^h}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)}$. It is worth noting that we cannot make the $2\varepsilon_h$ in the denominator ε_h because $\hat{V}_{ac}^*$ needs to simultaneously maintain a value gap with V^* . If we were to construct an example where $\hat{V}_{ac}^*(X_0) - V_{ac}^*(X_0) = V_{ac}^*$ be $\frac{\varepsilon_h \gamma^h}{(1-\gamma)(1-(1-\varepsilon_h)\gamma^h)}$, it would enforce $V_{ac}^*(X_0) = V^*(X_0)$ because there would be no probability mass left to create the gap between V^* and V_{ac}^* . With an extra ε_h in the denominator, we can also make the optimality gap of V_{ac}^* precisely $\frac{\varepsilon_h \gamma^h}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)}$, bringing the combined value gap (between V^* and $\hat{V}_{ac}^*$) up to $\frac{2\varepsilon_h \gamma^h}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)}$.
2. $P_{\mathcal{D}}(\cdot \mid s_t, \pi^+(s_t))$ and $T(\cdot \mid s_t, \pi^+(s_t))$ differs by only ε_h (again in TV distance as required by the strongly open-loop consistency assumption) where precisely ε_h probability mass is moved from reaching state G to reaching state Z . This causes the $\hat{V}_{ac}^+$ to precisely overestimates the value of V_{ac}^+ by $\frac{\varepsilon_h \gamma^h}{(1-\gamma)(1-(1-\varepsilon_h)\gamma^h)}$.

We use a special action $a = 2$ where upon taking the action the agent immediately transitions to Z and receives a reward of 0 (except in G). As we will see soon, this action is useful for constructing a data distribution with an easily ‘controllable’ probability of reaching Z for the top branch and an easily ‘controllable’ probability of reaching G for the bottom branch. Before we start constructing $\mathcal{D}$, we first check the condition that $\mathcal{D}^*$ is strongly ε_h -open-loop consistent.

Step 2. Strong ε_h -open-loop consistency of $\mathcal{D}^*$: It is clear that one possible π^* that achieves $1/(1-\gamma)$ value is

$$\begin{aligned}\pi^*(X_i) &= 0 \\ \pi^*(C) &= 1 \\ \pi^*(\tilde{X}) &= 0\end{aligned}\tag{D.86}$$

We can easily check that $\mathcal{D}^*$ collected by π^* is strongly ε_h -open-loop consistent by observing that the only path that π^* outputs $(0, 1, 0, 0, \dots)$ has ε_h probability, which causes the state distribution of s_{t+1} to differ by at most ε_h under the TV distance (subject to $a = (0, 1, 0, 0, \dots)$ or $a = (0, 0, 0, \dots)$ conditioning). This concludes that $\mathcal{D}^*$ generated by π^* above is strongly ε_h -open-loop consistent.

Now, depending on the first action a_t , the MDP can be decomposed into two parts: *the top* ($a = 0$) and *the bottom* ($a = 1$). We construct the data distribution for each branch and analyze the *actual* and *nominal* optimality gap for each branch in the following steps.

Step 3. Data distribution $\mathcal{D}_{\text{top}}$ for the top branch: we use a mixture of the following two policies to construct a strongly ε_h -open-loop consistent $\mathcal{D}_{\text{top}}$.

Policy π_{top}^1 :

$$\begin{aligned}\pi_{\text{top}}^1(X_0) &= \pi_{\text{top}}^1(C) = \pi_{\text{top}}^1(Z) = 0, \\ \pi_{\text{top}}^1(X_1) &= \pi_{\text{top}}^1(\tilde{X}_1) = 2.\end{aligned}\tag{D.87}$$

π_{top}^1 always take $a = 2$ unless it is in state X_0 , C or Z where it always takes $a = 0$. It is clear that this policy only produces two possible action chunks: $a_{t:t+h} = (0, 0, \dots, 0)$ or $a_{t:t+h} = a_{t:t+h}^\bullet := (0, 2, 0, \dots)$. We note that the $a_{t:t+h}$ policy always leads to state Z :

$$P_{\mathcal{D}_{\pi_{\text{top}}^1}}(s_{t+h} = Z \mid s_t, a_{t:t+h} = 0) = 1.\tag{D.88}$$

Policy π_{top}^2 :

$$\begin{aligned}\pi_{\text{top}}^2(X_0) &= 0, \\ \pi_{\text{top}}^2(\tilde{X}_1) &= \pi_{\text{top}}^2(\tilde{C}) = 1, \\ \pi_{\text{top}}^2(a = 0 \mid X_1) &= 1 - \delta_G, \\ \pi_{\text{top}}^2(a = 1 \mid X_1) &= \delta_G,\end{aligned}\tag{D.89}$$

with some $\delta_G \in (0, 1)$. π_{top}^2 can also only produce two possible action chunks: $a_{t:t+h} = (0, 0, \dots, 0)$ or $a_{t:t+h} = a_{t:t+h}^\diamond := (0, 1, 0, \dots, 0)$.

The distribution of s_{t+h} conditioned on $a_{t:t+h} = 0$ is

$$\begin{aligned} P_{\mathcal{D}_{\pi_{\text{top}}^2}}(s_{t+h} = Z \mid s_t, a_{t:t+h} = 0) &= 0, \\ P_{\mathcal{D}_{\pi_{\text{top}}^2}}(s_{t+h} = G \mid s_t, a_{t:t+h} = 0) &= 0, \\ P_{\mathcal{D}_{\pi_{\text{top}}^2}}(s_{t+h} = X_0 \mid s_t, a_{t:t+h} = 0) &= 1. \end{aligned} \tag{D.90}$$

Mixing π_{top}^1 and π_{top}^2 : Let $\mathcal{D}_{\text{top}}$ be a mixture of $\mathcal{D}_{\pi_{\text{top}}^1}$ and $\mathcal{D}_{\pi_{\text{top}}^2}$:

$$P_{\mathcal{D}_{\text{top}}} = (1 - \varsigma)P_{\mathcal{D}_{\pi_{\text{top}}^1}} + \varsigma P_{\mathcal{D}_{\pi_{\text{top}}^2}}, \tag{D.91}$$

where

$$\varsigma = \frac{1}{2(1 - \delta_G) + 1}. \tag{D.92}$$

It is clear that $0 < \varsigma < 1$ (because $\delta_G \in (0, 1)$), making it valid mixing ratio.

We now compute the marginal state distribution of the mixture by first analyzing the action probability:

$$\begin{aligned} P_{\mathcal{D}_{\text{top}}^1}(a_{t:t+h}^* \mid s_t) &= \varepsilon_h, \\ P_{\mathcal{D}_{\text{top}}^2}(a_{t:t+h}^* \mid s_t) &= (1 - 2\varepsilon_h)(1 - \delta_G). \end{aligned} \tag{D.93}$$

The state marginals are then

$$\begin{aligned} P_{\mathcal{D}_{\text{top}}}(s_{t+h} = Z \mid s_t, a_{t:t+h}^*) &= \frac{P_{\mathcal{D}_{\text{top}}}(s_{t+h} = Z, a_{t:t+h}^* \mid s_t)}{P_{\mathcal{D}_{\text{top}}}(a_{t:t+h}^* \mid s_t)} \\ &= \frac{(1 - \varsigma)P_{\mathcal{D}_{\pi_{\text{top}}^1}}(a_{t:t+h}^* \mid s_t)}{(1 - \varsigma)P_{\mathcal{D}_{\pi_{\text{top}}^1}}(a_{t:t+h}^* \mid s_t) + \varsigma P_{\mathcal{D}_{\pi_{\text{top}}^2}}(a_{t:t+h}^* \mid s_t)} \\ &= \frac{\varepsilon_h(1 - \varsigma)}{\varepsilon_h(1 - \varsigma) + (1 - 2\varepsilon_h)(1 - \delta_G)\varsigma} \\ &= 2\varepsilon_h. \end{aligned} \tag{D.94}$$

Therefore,

$$\begin{aligned} P_{\mathcal{D}_{\text{top}}}(s_{t+h} = Z \mid s_t, a_{t:t+h}^*) &= 2\varepsilon_h, \\ P_{\mathcal{D}_{\text{top}}}(s_{t+h} = X_0 \mid s_t, a_{t:t+h}^*) &= 1 - 2\varepsilon_h, \\ P_{\mathcal{D}_{\text{top}}}(s_{t+h} = G \mid s_t, a_{t:t+h}^*) &= 0. \end{aligned} \tag{D.95}$$

Step 4. Strong ε_h -open-loop consistency of $\mathcal{D}_{\text{top}}$: Now, we check for strong open-loop consistency for the three possible action chunks on the top branch:

$$\begin{aligned} a_{t:t+h}^* &= (0, 0, 0, \dots) \\ a_{t:t+h}^\diamond &= (0, 1, 0, \dots) \\ a_{t:t+h}^\bullet &= (0, 2, 0, \dots) \end{aligned} \tag{D.96}$$

For $a_{t:t+h}^* = 0$, we can compute open-loop marginal state distribution as follows:

$$\begin{aligned} T(s_{t+h} = Z \mid s_t, a_{t:t+h}^*) &= \varepsilon_h, \\ T(s_{t+h} = X_0 \mid s_t, a_{t:t+h}^*) &= 1 - 2\varepsilon_h, \\ T(s_{t+h} = G \mid s_t, a_{t:t+h}^*) &= \varepsilon_h. \end{aligned} \quad (\text{D.97})$$

Combining this with the data distribution calculated in Equation (D.95), it is clear that

$$D_{\text{TV}}(T(s_{t+h} \mid s_t, a_{t:t+h} = 0) \parallel P_{\mathcal{D}_{\text{top}}}(s_{t+h} \mid s_t, a_{t:t+h} = 0)) = \varepsilon_h. \quad (\text{D.98})$$

We can repeat the same procedure to show that

$$D_{\text{TV}}(T(s_{t+h'} \mid s_t, a_{t:t+h'} = 0) \parallel P_{\mathcal{D}_{\text{top}}}(s_{t+h'} \mid s_t, a_{t:t+h} = 0)) = \varepsilon_h, \quad \forall h' \in \{1, \dots, h-1\} \quad (\text{D.99})$$

because the only difference in these distributions is that they occupy $s_{t+h'} = X_{h'}$ with $2\varepsilon_h$ probability instead of $s_{t+h} = X_0$ with $2\varepsilon_h$ probability.

For $a_{t:t+h}^\bullet = (0, 2, 0, \dots)$, it is clear that

$$D_{\text{TV}}(T(s_{t+h'} \mid s_t, a_{t:t+h} = a_{t:t+h}^\bullet) \parallel P_{\mathcal{D}_{\text{top}}}(s_{t+h'} \mid s_t, a_{t:t+h} = a_{t:t+h}^\bullet)) = \varepsilon_h \quad (\text{D.100})$$

holds for any $h' \in \{1, 2, \dots, h\}$ since the only difference between these two distributions is the ε_h -probability path (*i.e.*, $X_0 \rightarrow C \rightarrow Z$ where the probability is under $T(\cdot \mid s_t, a_{t:t+h}^\bullet)$).

For $a_{t:t+h}^\diamond = (0, 1, 0, \dots)$, we first compute the marginal state distributions:

$$\begin{aligned} P_{\mathcal{D}_{\text{top}}}(s_{t+h} = Z \mid s_t, a_{t:t+h}^\diamond) &= \frac{(1 - 2\varepsilon_h)\delta_G}{2\varepsilon_h + (1 - 2\varepsilon_h)\delta_G}, \\ P_{\mathcal{D}_{\text{top}}}(s_{t+h} = X_0 \mid s_t, a_{t:t+h}^\diamond) &= \frac{2\varepsilon_h}{2\varepsilon_h + (1 - 2\varepsilon_h)\delta_G}, \\ P_{\mathcal{D}_{\text{top}}}(s_{t+h} = G \mid s_t, a_{t:t+h}^\diamond) &= 0. \\ P_{\mathcal{D}_{\text{top}}}(s_{t+1} = X_1 \mid s_t, a_{t:t+h}^\diamond) &= \frac{(1 - 2\varepsilon_h)\delta_G}{2\varepsilon_h + (1 - 2\varepsilon_h)\delta_G}. \\ P_{\mathcal{D}_{\text{top}}}(s_{t+1} = \tilde{X}_1 \mid s_t, a_{t:t+h}^\diamond) &= \frac{\varepsilon_h}{2\varepsilon_h + (1 - 2\varepsilon_h)\delta_G}. \\ P_{\mathcal{D}_{\text{top}}}(s_{t+1} = C \mid s_t, a_{t:t+h}^\diamond) &= \frac{\varepsilon_h}{2\varepsilon_h + (1 - 2\varepsilon_h)\delta_G}. \end{aligned} \quad (\text{D.101})$$

We can also compute the open-loop marginal state distribution as follows:

$$\begin{aligned} T(s_{t+h} = Z \mid s_t, a_{t:t+h}^\diamond) &= 1 - 2\varepsilon_h \\ T(s_{t+h} = X_0 \mid s_t, a_{t:t+h}^\diamond) &= 2\varepsilon_h \\ T(s_{t+h} = G \mid s_t, a_{t:t+h}^\diamond) &= 0. \\ T(s_{t+1} = X_1 \mid s_t, a_{t:t+h}^\diamond) &= 1 - 2\varepsilon_h. \\ T(s_{t+1} = \tilde{X}_1 \mid s_t, a_{t:t+h}^\diamond) &= \varepsilon_h. \\ T(s_{t+1} = C \mid s_t, a_{t:t+h}^\diamond) &= \varepsilon_h. \end{aligned} \quad (\text{D.102})$$

Let c_3 be any value that satisfies $c_3 \in (0, \varepsilon_h/2)$, we can set

$$\delta_G = \frac{\varepsilon_h(1 - 2\varepsilon_h - 2c_3)}{(\varepsilon_h + c_3)(1 - 2\varepsilon_h)}, \quad (\text{D.103})$$

such that

$$\begin{aligned} P_{\mathcal{D}_{\text{top}}}(s_{t+h} = Z \mid s_t, a_{t:t+h}^\diamond) &= 1 - 2\varepsilon_h - 2c_3, \\ P_{\mathcal{D}_{\text{top}}}(s_{t+h} = X_0 \mid s_t, a_{t:t+h}^\diamond) &= 2\varepsilon_h + 2c_3, \\ P_{\mathcal{D}_{\text{top}}}(s_{t+h} = G \mid s_t, a_{t:t+h}^\diamond) &= 0, \\ P_{\mathcal{D}_{\text{top}}}(s_{t+1} = X_1 \mid s_t, a_{t:t+h}^\diamond) &= 1 - 2\varepsilon_h - 2c_3, \\ P_{\mathcal{D}_{\text{top}}}(s_{t+1} = \tilde{X}_1 \mid s_t, a_{t:t+h}^\diamond) &= \varepsilon_h + c_3, \\ P_{\mathcal{D}_{\text{top}}}(s_{t+1} = C \mid s_t, a_{t:t+h}^\diamond) &= \varepsilon_h + c_3. \end{aligned} \quad (\text{D.104})$$

It is easy to check that $0 < \delta_G < 1$ (a valid probability) because in Equation (D.103), each term in the numerator has a larger term in the denominator (*i.e.*, $\varepsilon_h < \varepsilon_h + c_3$ and $1 - 2\varepsilon_h - 2c_3 < 1 - 2\varepsilon_h$).

Now, for all $h' \in \{1, 2, \dots, h\}$, using the values calculated in Equations (D.102) and (D.104), we have

$$D_{\text{TV}}(T(s_{t+h'} \mid s_t, a_{t:t+h'} = a_{t:t+h'}^\diamond) \parallel P_{\mathcal{D}_{\text{top}}}(s_{t+h'} \mid s_t, a_{t:t+h} = a_{t:t+h}^\diamond)) = 2c_3. \quad (\text{D.105})$$

Since $c_3 < \varepsilon_h/2$, the strong open-loop consistency assumption holds for $a_{t:t+h}^\diamond$ as well.

Step 5. The optimality gap and value estimation error for the top branch: Now we can compute the optimality gap for the estimated value for $a_{t:t+h}^\diamond$:

$$V^*(X_0) - \hat{V}_{\text{ac}}^\diamond(X_0) = \frac{(1 - 2\varepsilon_h - 2c_3)\gamma}{(1 - \gamma)(1 - 2(\varepsilon_h + c_3)\gamma^h)}, \quad (\text{D.106})$$

where the h -step reward sub-optimality gap is due to the agent reaching Z with $(1 - 2\varepsilon_h - 2c_3)$ probability, and the h -step distribution gap is reflected in the $(1 - 2(\varepsilon_h + c_3)\gamma^h)$ term at bottom because the probability of reaching X_0 after h steps is $2(\varepsilon_h + c_3)$.

Similarly, we can compute the optimality gap for V_{ac}^* and $\hat{V}_{\text{ac}}^*$:

$$\begin{aligned} V^*(X_0) - V_{\text{ac}}^*(X_0) &= \varepsilon_h \frac{\gamma - \gamma^h}{1 - \gamma} + \frac{\varepsilon_h \gamma^h}{1 - \gamma} + \gamma^h(1 - 2\varepsilon_h)(V^* - \hat{V}_{\text{ac}}) \\ &= \frac{\varepsilon_h \gamma}{(1 - \gamma)(1 - (1 - 2\varepsilon_h)\gamma^h)}, \end{aligned} \quad (\text{D.107})$$

$$\begin{aligned} V^*(X_0) - \hat{V}_{\text{ac}}^*(X_0) &= \frac{2\varepsilon_h(\gamma - \gamma^h)}{1 - \gamma} + \frac{2\varepsilon_h \gamma^h}{1 - \gamma} \gamma^h(1 - 2\varepsilon_h)(V^* - \hat{V}_{\text{ac}}) \\ &= \frac{2\varepsilon_h \gamma}{(1 - \gamma)(1 - (1 - 2\varepsilon_h)\gamma^h)}. \end{aligned} \quad (\text{D.108})$$

Now, we observe that

$$1 - 2\varepsilon_h - 2c_3 > 1 - 3\varepsilon_h > 2\varepsilon_h, \quad (\text{D.109})$$

where the first inequality is due to $c_3 \in (0, \varepsilon_h/2)$ and the second inequality is due to $\varepsilon_h \in (0, 1/5)$ in our assumption. This allows us to lower bound the estimated optimality gap for $a_{t:t+h}^\diamond$ as follows:

$$\begin{aligned} V^*(X_0) - \hat{V}_{ac}^\diamond(X_0) &= \frac{(1 - 2\varepsilon_h - 2c_3)\gamma}{(1 - \gamma)(1 - 2(\varepsilon_h + c_3)\gamma^h)} \\ &> \frac{2\varepsilon_h\gamma}{(1 - \gamma)(1 - (1 - 2\varepsilon_h)\gamma^h)} \\ &= V^*(X_0) - \hat{V}_{ac}^*(X_0), \end{aligned} \quad (\text{D.110})$$

where the inequality is obtained by triggering Lemma 5 (e.g., by setting $\varepsilon_1 = 2\varepsilon_h, \varepsilon_2 = (1 - 2\varepsilon_h - 2c_3), \tilde{\gamma} = \gamma^h$). The bound above rules out the possibility of $a_{t:t+h}^\diamond$ being picked by $\hat{\pi}_{ac}^+$ because it has a lower estimated value compared to $a_{t:t+h}^*$.

Finally, for $a_{t:t+h}^\bullet$, since it is correlated with $s_{t+h} = Z$ and receives no reward except the first step in $\mathcal{D}_{\text{top}}$, the estimated value is just 1, being trivially smaller than $\hat{V}_{ac}^*(X_0)$ and would never get picked by $\hat{\pi}_{ac}^+$. Up to now, we have finished our data distribution construction and analysis for the top branch. We summarize the key intermediate results as the remark below:

Remark 1 (Intermediate results from Step 1-4) The optimal action chunk is $a_{t:t+h}^*$ and the estimated values for the two other possible action chunks $a_{t:t+h}^\bullet$, $a_{t:t+h}^\diamond$ are smaller than that of $a_{t:t+h}^*$:

$$\hat{V}_{ac}^\bullet(X_0) < \hat{V}_{ac}^\diamond(X_0) < \hat{V}_{ac}^*(X_0) = V^*(X_0) - \frac{2\varepsilon_h\gamma}{(1 - \gamma)(1 - (1 - 2\varepsilon_h)\gamma^h)}. \quad (\text{D.111})$$

In addition, both $\mathcal{D}_{\text{top}}$ and $\mathcal{D}^*$ are strongly ε_h -open-loop consistent.

Next, we move on to the bottom branch.

Step 6. Data distribution $\mathcal{D}_{\text{bottom}}$ for the bottom branch: For the bottom, we again use two policies.

Policy π_{bottom}^1 :

$$\begin{aligned} \pi_{\text{bottom}}^1(X_0) &= \pi_{\text{bottom}}^1(G) = \pi_{\text{bottom}}^1(Z) = 1, \\ \pi_{\text{bottom}}^1(Y_1) &= \pi_{\text{bottom}}^1(\tilde{Y}_1) = 2. \end{aligned} \quad (\text{D.112})$$

π_{bottom}^1 takes $a = 1$ at X_0 and G and Z , and takes $a = 2$ otherwise (at $Y_1, \tilde{Y}_1$). It is clear that this policy only produces two possible action chunks: $a_{t:t+h}^\Delta = (1, 1, 1, \dots)$ or $a_{t:t+h}^\times = (1, 2, 1, \dots)$.

Policy π_{bottom}^2 :

$$\begin{aligned}\pi_{\text{bottom}}^2(X_0) &= 1, \\ \pi_{\text{bottom}}^2(a = 0 \mid Y_1) &= \delta_Z, \\ \pi_{\text{bottom}}^2(a = 1 \mid Y_1) &= 1 - \delta_Z, \\ \pi_{\text{bottom}}^2(\tilde{Y}_1) &= 0, \\ \pi_{\text{bottom}}^2(Y_i) &= 1, \quad \forall i \in \{2, \dots, h-1\},\end{aligned}\tag{D.113}$$

where $\delta_Z \in (0, 1)$ and we shall specify the exact value of δ_Z shortly.

π_{bottom}^2 takes $a = 1$ when it is at Y_i and takes either $a = 0$ (with δ_Z probability) or $a = 1$ (with $1 - \delta_Z$ probability) when it is at $\tilde{Y}_1$. It is clear that this policy only produces two possible action chunks: $a_{t:t+h}^\Delta = (1, 1, 1, \dots)$ or $a_{t:t+h}^\circ = (1, 0, 1, \dots)$.

Now, we observe that the marginal state distributions for both policies conditioned on $a_{t:t+h}^\Delta$ are independent of c_1 and δ_Z because the action chunk only appears when π_{bottom}^1 reaches G and when π_{bottom}^2 reaches X_0 . More specifically,

$$P_{\mathcal{D}_{\text{bottom}}^1}(s_{t+1} = G \mid s_t, a_{t:t+h}^\Delta) = P_{\mathcal{D}_{\text{bottom}}^1}(s_{t+h} = G \mid s_t, a_{t:t+h}^\Delta) = 1,\tag{D.114}$$

$$P_{\mathcal{D}_{\text{bottom}}^2}(s_{t+i} = X_i \mid s_t, a_{t:t+h}^\Delta) = P_{\mathcal{D}_{\text{bottom}}^2}(s_{t+h} = X_0 \mid s_t, a_{t:t+h}^\Delta) = 1, \forall i \in \{1, \dots, h-1\}.\tag{D.115}$$

We can now mix $\mathcal{D}_{\text{bottom}}^1$ and $\mathcal{D}_{\text{bottom}}^2$ with an appropriate ratio to control the state marginals for $s_{t:t+h} = G$ and $s_{t:t+h} = X_0$ arbitrarily ($s_{t:t+h} = Z$ stays at 0 because none of the policies take/have taken $a_{t:t+h}^\Delta$ when they reach Z).

Mixing π_{bottom}^1 and π_{bottom}^2 : Let $\mathcal{D}_{\text{bottom}}$ be a mixture of $\mathcal{D}_{\text{bottom}}^1$ and $\mathcal{D}_{\text{bottom}}^2$:

$$P_{\mathcal{D}_{\text{bottom}}} = (1 - \vartheta)P_{\mathcal{D}_{\text{bottom}}^1} + \vartheta P_{\mathcal{D}_{\text{bottom}}^2},\tag{D.116}$$

where we set the mixing ratio to be

$$\vartheta = \frac{c_1}{c_1 + (1 - \delta_Z)(\varepsilon_h + c_1)}.\tag{D.117}$$

This mixing ratio helps the calculations to be simpler later on.

We can now compute the marginal state distribution of the mixture. We start by analyzing the action probability:

$$\begin{aligned}P_{\mathcal{D}_{\text{bottom}}^1}(a_{t:t+h}^\Delta \mid s_t) &= c_1, \\ P_{\mathcal{D}_{\text{bottom}}^2}(a_{t:t+h}^\Delta \mid s_t) &= (1 - \varepsilon_h - c_1)(1 - \delta_Z).\end{aligned}\tag{D.118}$$

The state marginal is then

$$\begin{aligned}
P_{\mathcal{D}_{\text{bottom}}}(s_{t+h} = X_0 \mid s_t, a_{t:t+h}^\Delta) &= \frac{P_{\mathcal{D}_{\text{bottom}}}(s_{t+h} = X_0, a_{t:t+h}^\Delta \mid s_t)}{P_{\mathcal{D}_{\text{bottom}}}(a_{t:t+h}^\Delta \mid s_t)} \\
&= \frac{\vartheta P_{\mathcal{D}_{\text{bottom}}^2}(a_{t:t+h}^\Delta \mid s_t)}{(1 - \vartheta)P_{\mathcal{D}_{\text{bottom}}^1}(a_{t:t+h}^\Delta \mid s_t) + \vartheta P_{\mathcal{D}_{\text{bottom}}^2}(a_{t:t+h}^\Delta \mid s_t)} \quad (\text{D.119}) \\
&= \frac{(1 - \varepsilon_h - c_1)(1 - \delta_Z)\vartheta}{c_1(1 - \vartheta) + (1 - \varepsilon_h - c_1)(1 - \delta_Z)\vartheta} \\
&= 1 - \varepsilon_h - c_1.
\end{aligned}$$

We can use it to deduce the rest of the marginals as follows:

$$\begin{aligned}
P_{\mathcal{D}_{\text{bottom}}}(s_{t+h} = G \mid s_t, a_{t:t+h}^\Delta) &= \varepsilon_h + c_1, \quad \forall h' \in \{1, \dots, h-1\}, \\
P_{\mathcal{D}_{\text{bottom}}}(s_{t+h} = X_0 \mid s_t, a_{t:t+h}^\Delta) &= 1 - \varepsilon_h - c_1, \\
P_{\mathcal{D}_{\text{bottom}}}(s_{t+h} = Z \mid s_t, a_{t:t+h}^\Delta) &= 0, \\
P_{\mathcal{D}_{\text{bottom}}}(s_{t+h'} = Y_{h'} \mid s_t, a_{t:t+h}^\Delta) &= 1 - \varepsilon_h - c_1, \quad \forall h' \in \{1, \dots, h-2\}, \\
P_{\mathcal{D}_{\text{bottom}}}(s_{t+1} = \tilde{Y}_1 \mid s_t, a_{t:t+h}^\Delta) &= 0.
\end{aligned} \quad (\text{D.120})$$

Up to now, we have established $\mathcal{D}_{\text{bottom}}$ and we are ready to check the strong open-loop consistency.

Step 7. Strong ε_h -open-loop consistency of $\mathcal{D}_{\text{bottom}}$:

For $a_{t:t+h}^\Delta = (1, 1, \dots)$, we can compute the open-loop marginals as follows:

$$\begin{aligned}
T(s_{t+h'} = G \mid s_t, a_{t:t+h}^\Delta) &= c_1, \quad \forall h' \in \{1, \dots, h-1\}, \\
T(s_{t+h} = X_0 \mid s_t, a_{t:t+h}^\Delta) &= 1 - \varepsilon_h - c_1, \\
T(s_{t+h} = Z \mid s_t, a_{t:t+h}^\Delta) &= \varepsilon_h. \\
T(s_{t+h'} = Y_{h'} \mid s_t, a_{t:t+h}^\Delta) &= 1 - \varepsilon_h - c_1, \quad \forall h' \in \{1, \dots, h-2\} \\
T(s_{t+1} = \tilde{Y}_1 \mid s_t, a_{t:t+h}^\Delta) &= \varepsilon_h.
\end{aligned} \quad (\text{D.121})$$

Combining it with the marginals calculated in Equation (D.120), it is clear that for all $h' \in \{1, \dots, h-1\}$,

$$D_{\text{TV}}(T(s_{t+h'} \mid s_t, a_{t:t+h'} = a_{t:t+h'}^+) \parallel P_{\mathcal{D}_{\text{bottom}}}(s_{t+h'} \mid s_t, a_{t:t+h} = a_{t:t+h}^+)) = \varepsilon_h, \quad (\text{D.122})$$

satisfying the open-loop consistency.

For $a_{t:t+h}^\times = (1, 2, 1, \dots)$, the data and open-loop state marginals are

$$\begin{aligned}
P_{\mathcal{D}_{\text{bottom}}}(s_{t+h} = Z \mid s_t, a_{t:t+h}^\times) &= 1, \\
P_{\mathcal{D}_{\text{bottom}}}(s_{t+1} = Y_1 \mid s_t, a_{t:t+h}^\times) &= \frac{1 - \varepsilon_h - c_1}{1 - c_1}, \\
P_{\mathcal{D}_{\text{bottom}}}(s_{t+1} = \tilde{Y}_1 \mid s_t, a_{t:t+h}^\times) &= \frac{\varepsilon_h}{1 - c_1}, \\
T(s_{t+h} = Z \mid s_t, a_{t:t+h}^\times) &= 1 - c_1, \\
T(s_{t+h} = G \mid s_t, a_{t:t+h}^\times) &= c_1, \\
T(s_{t+1} = Y_1 \mid s_t, a_{t:t+h}^\times) &= 1 - \varepsilon_h - c_1, \\
T(s_{t+1} = \tilde{Y}_1 \mid s_t, a_{t:t+h}^\times) &= \varepsilon_h, \\
T(s_{t+1} = G \mid s_t, a_{t:t+h}^\times) &= c_1.
\end{aligned} \tag{D.123}$$

This allows us to bound the TV distance for all $h' \in \{1, \dots, h-1\}$ as

$$D_{\text{TV}}(T(s_{t+h'} \mid s_t, a_{t:t+h'} = a_{t:t+h'}^\times) \parallel P_{\mathcal{D}_{\text{bottom}}}(s_{t+h'} \mid s_t, a_{t:t+h} = a_{t:t+h}^\times)) \leq \frac{c_1}{1 - c_1}. \tag{D.124}$$

Since $c_1 < \varepsilon_h/2 < 1/10$,

$$\frac{c_1}{1 - c_1} < \frac{10}{9}c_1 < 5\varepsilon_h/9 < \varepsilon_h, \tag{D.125}$$

satisfying the strong open-loop consistency assumption.

For $a_{t:t+h}^\circ = (1, 0, 1, \dots)$, we first compute the state marginals in $\mathcal{D}_{\text{bottom}}$ as follows:

$$\begin{aligned}
P_{\mathcal{D}_{\text{bottom}}}(s_{t+h} = Z \mid s_t, a_{t:t+h}^\circ) &= \frac{(1 - \varepsilon_h - c_1)\delta_Z}{\varepsilon_h + (1 - \varepsilon_h - c_1)\delta_Z}, \\
P_{\mathcal{D}_{\text{bottom}}}(s_{t+h} = X_0 \mid s_t, a_{t:t+h}^\circ) &= \frac{\varepsilon_h}{\varepsilon_h + (1 - \varepsilon_h - c_1)\delta_Z}, \\
P_{\mathcal{D}_{\text{bottom}}}(s_{t+1} = Y_1 \mid s_t, a_{t:t+h}^\circ) &= \frac{(1 - \varepsilon_h - c_1)\delta_Z}{\varepsilon_h + (1 - \varepsilon_h - c_1)\delta_Z}, \\
P_{\mathcal{D}_{\text{bottom}}}(s_{t+1} = \tilde{Y}_1 \mid s_t, a_{t:t+h}^\circ) &= \frac{\varepsilon_h}{\varepsilon_h + (1 - \varepsilon_h - c_1)\delta_Z}.
\end{aligned} \tag{D.126}$$

We can also compute the open-loop marginal state distribution as follows:

$$\begin{aligned}
T(s_{t+h} = Z \mid s_t, a_{t:t+h}^\circ) &= 1 - \varepsilon_h - c_1, \\
T(s_{t+h} = X_0 \mid s_t, a_{t:t+h}^\circ) &= \varepsilon_h, \\
T(s_{t+h} = G \mid s_t, a_{t:t+h}^\circ) &= c_1, \\
T(s_{t+1} = Y_1 \mid s_t, a_{t:t+h}^\circ) &= 1 - \varepsilon_h - c_1, \\
T(s_{t+1} = \tilde{Y}_1 \mid s_t, a_{t:t+h}^\circ) &= \varepsilon_h, \\
T(s_{t+1} = G \mid s_t, a_{t:t+h}^\circ) &= c_1.
\end{aligned} \tag{D.127}$$

Let $c_4 \in (c_1, \varepsilon_h)$, and we set

$$\delta_Z = \frac{\varepsilon_h(1 - \varepsilon_h - c_4)}{(\varepsilon_h + c_4)(1 - \varepsilon_h - c_1)}. \quad (\text{D.128})$$

Then, we have

$$\begin{aligned} P_{\mathcal{D}_{\text{bottom}}}(s_{t+h} = Z \mid s_t, a_{t:t+h}^\circ) &= 1 - \varepsilon_h - c_4, \\ P_{\mathcal{D}_{\text{bottom}}}(s_{t+h} = X_0 \mid s_t, a_{t:t+h}^\circ) &= \varepsilon_h + c_4, \\ P_{\mathcal{D}_{\text{bottom}}}(s_{t+h} = G \mid s_t, a_{t:t+h}^\circ) &= 0, \\ P_{\mathcal{D}_{\text{bottom}}}(s_{t+1} = Y_1 \mid s_t, a_{t:t+h}^\circ) &= 1 - \varepsilon_h - c_4, \\ P_{\mathcal{D}_{\text{bottom}}}(s_{t+1} = \tilde{Y}_1 \mid s_t, a_{t:t+h}^\circ) &= \varepsilon_h + c_4. \end{aligned} \quad (\text{D.129})$$

The TV distance is then

$$D_{\text{TV}}(T(s_{t+h'} \mid s_t, a_{t:t+h'} = a_{t:t+h'}^\circ) \parallel P_{\mathcal{D}_{\text{bottom}}}(s_{t+h'} \mid s_t, a_{t:t+h} = a_{t:t+h}^\circ)) = c_4. \quad (\text{D.130})$$

Since $c_4 < \varepsilon_h$, the strong open-loop consistency is also satisfied for $a_{t:t+h}^\circ$.

Up to now, we have checked that all three possible action chunks in the bottom branch satisfy the strong open-loop consistency assumption. Since $\mathcal{D}_{\text{top}}$ and $\mathcal{D}_{\text{bottom}}$ have non-overlapping supports for $a_{t:t+h}$, and they are both strongly ε_h -open-loop consistent on their own, we can construct $\mathcal{D}$ as

$$P_{\mathcal{D}}(\cdot \mid s_t) = (1 - \varrho)P_{\mathcal{D}_{\text{top}}}(\cdot \mid s_t) + \varrho P_{\mathcal{D}_{\text{bottom}}}(\cdot \mid s_t), \quad (\text{D.131})$$

for any $\varrho \in (0, 1)$, and conclude that

Remark 2 (Intermediate result from Step 5-7) $\mathcal{D}$ is strongly ε_h -open-loop consistent.

Up to now, we have checked both $\mathcal{D}$ and $\mathcal{D}^*$ are strongly ε_h -open-loop consistent.

As the final step, we calculate the optimality gap and value estimation error for these action chunks.

Step 8. The optimality gap and value estimation error for the bottom branch:

We first note that similar to $a_{t:t+h}^\bullet$, $a_{t:t+h}^\times$ is correlated with $s_{t+h} = Z$ and always receives 0 reward except the first step in $\mathcal{D}$. Thus, the estimated value $\hat{V}^\times$ is just 1, being trivially smaller than $\hat{V}_{\text{ac}}^*$ and would never get picked by $\hat{\pi}_{\text{ac}}^\Delta$. The only top contenders are $a_{t:t+h}^+$, $a_{t:t+h}^\circ$ and $a_{t:t+h}^*$ (which we already analyzed in Step 5 above).

We start with $a_{t:t+h}^\circ$ where we can compute optimality gap as follows:

$$V^*(X_0) - \hat{V}_{\text{ac}}^\circ(X_0) = \frac{(1 - \varepsilon_h - c_4)\gamma + \delta(1 - \gamma) + (\varepsilon_h + c_4)\delta(\gamma - \gamma^h)}{(1 - \gamma)(1 - (\varepsilon_h + c_4)\gamma^h)}. \quad (\text{D.132})$$

Now, observe that

$$\varepsilon_h + c_4 < 2\varepsilon_h < 1 - 2\varepsilon_h, \quad (\text{D.133})$$

where again the last inequality comes from the fact that $\varepsilon_h < 1/4$.

We can now lower bound the optimality gap as follows:

$$\begin{aligned} V^*(X_0) - \hat{V}_{\text{ac}}^\circ(X_0) &> \frac{2\varepsilon_h\gamma + \delta(1-\gamma) + (\varepsilon_h + c_4)\delta(\gamma - \gamma^h)}{(1-\gamma)(1 - (1-2\varepsilon_h)\gamma^h)} \\ &> \frac{2\varepsilon_h\gamma}{(1-\gamma)(1 - (1-2\varepsilon_h)\gamma^h)} \\ &= V^*(X_0) - \hat{V}_{\text{ac}}^*(X_0). \end{aligned} \quad (\text{D.134})$$

where the first inequality is obtained by triggering Lemma 5 (e.g., by setting $\varepsilon_1 = 2\varepsilon_h, \varepsilon_2 = (1 - \varepsilon_h - c_4), \tilde{\gamma} = \gamma^h$).

With this lower bound, we can conclude that $a_{t:t+h}^\circ$ would not be picked by π_{ac}^+ as well because $\hat{V}_{\text{ac}}^\circ(X_0) < \hat{V}_{\text{ac}}^*(X_0)$.

Up to now, we have eliminated both $a_{t:t+h}^\circ$ and $a_{t:t+h}^\times$ (for the possibility of being picked by π_{ac}^+) and the only remaining contender left is $a_{t:t+h}^\Delta$.

We can also compute the estimated and the actual values for $a_{t:t+h} = a_{t:t+h}^\Delta = 1$ in terms of their optimality gaps:

$$V^*(X_0) - \hat{V}_{\text{ac}}^\Delta(X_0) = \frac{\delta(1 - \varepsilon_h - c_1)\gamma}{(1-\gamma)(1 - (1 - \varepsilon_h - c_1)\gamma^h)}, \quad (\text{D.135})$$

$$V^*(X_0) - V_{\text{ac}}^\Delta(X_0) = \frac{[\delta(1 - \varepsilon_h - c_1) + \varepsilon_h]\gamma}{(1-\gamma)(1 - (1 - \varepsilon_h - c_1)\gamma^h)}. \quad (\text{D.136})$$

Let

$$\delta = \frac{2\varepsilon_h\gamma - c_2}{(1 - \varepsilon_h - c_1)\gamma} \frac{1 - (1 - \varepsilon_h - c_1)\gamma^h}{1 - (1 - 2\varepsilon_h)\gamma^h}. \quad (\text{D.137})$$

We first check $1 - \delta$ is a valid reward value (within $[0, 1]$):

$$\begin{aligned} \delta &< \frac{2\varepsilon_h}{1 - \varepsilon_h - c_1} \frac{1 - (1 - \varepsilon_h - c_1)\gamma^h}{1 - (1 - 2\varepsilon_h)\gamma^h} \\ &< \frac{2\varepsilon_h}{1 - 2\varepsilon_h} \frac{1 - (1 - 2\varepsilon_h)\gamma^h}{1 - (1 - 2\varepsilon_h)\gamma^h} \\ &= \frac{2\varepsilon_h}{1 - 2\varepsilon_h} \\ &\leq 1, \end{aligned} \quad (\text{D.138})$$

where the first inequality is because $c_2 > 0$, the second inequality is due to $c_1 < \varepsilon_h$, and the final inequality is due to $\varepsilon_h < 1/4$.

It is also clear that $\delta > 0$ because all terms are positive in the fraction (Equation (D.137)).

Next, we substitute δ in to obtain

$$V^*(X_0) - \hat{V}_{ac}^\Delta(X_0) = \frac{2\varepsilon_h\gamma - c_2}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)}, \quad (D.139)$$

$$V^*(X_0) - V_{ac}^\Delta(X_0) = \frac{2\varepsilon_h\gamma - c_2}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)} + \frac{\varepsilon_h\gamma}{(1-\gamma)(1-(1-\varepsilon_h-c_1)\gamma^h)}, \quad (D.140)$$

where intuitively the second term in $V^*(X_0) - V_{ac}^\Delta(X_0)$ is due to the fact that from $P_{\mathcal{D}}(\cdot | s_t, a_{t:t+h}^\Delta)$ to $T(\cdot | s_t, a_{t:t+h}^\Delta)$, there is a shift in ε_h probability mass from $s_{t:t+h} = (X_0, G, \dots)$ to $s_{t:t+h} = (X_0, \tilde{Y}_1, Z, \dots)$ incurring an additional $\frac{\varepsilon_h\gamma}{1-\gamma}$ sub-optimality in terms of the h -step reward, and then amplified by the value recursion by an additional factor of $\frac{1}{1-(1-\varepsilon_h-c_1)\gamma^h}$ (where $1-\varepsilon_h-c_1$ is the probability that $a_{t:t+h}^\Delta$ reaches X_0 for the value recursion to occur).

Since $c_2 > 0$, we can now show that $a_{t:t+h}^\Delta$ achieves the highest estimated value among six possible action chunks:

$$V^* - \hat{V}_{ac}^\Delta < V^* - \hat{V}_{ac}^* = \frac{2\varepsilon_h\gamma}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)}, \quad (D.141)$$

which means that $\pi_{ac}^+(X_0) = a_{t:t+h}^\Delta = (1, 1, \dots)$, or equivalently $\hat{V}_{ac}^\Delta = \hat{V}_{ac}^+$!

Finally, putting everything together, we have

$$V^*(X_0) - V_{ac}^+(X_0) = \frac{2\varepsilon_h\gamma - c_2}{(1-\gamma)(1-(1-2\varepsilon_h)\gamma^h)} + \frac{\varepsilon_h\gamma}{(1-\gamma)(1-(1-\varepsilon_h-c_1)\gamma^h)}, \quad (D.142)$$

as desired. $\square$

D.9 Proof of Proposition 3

In this section, we provide the proof for Proposition 3 (restated below).

Proposition 3 (Optimality of Closed-loop Execution of Action Chunking Policy) Let $V^\bullet$ be the value of the one-step policy, $\pi^\bullet$, as a result of the closed-loop execution of the action chunking policy π_{ac}^+ learned from $\mathcal{D}$. That is, for each $s_t \in \text{supp}(P_{\mathcal{D}}(s_t))$,

$$\pi^\bullet(s_t) = a_t^+, \quad \text{where } a_{t:t+h}^+ = \pi_{ac}^+(s_t). \quad (5.23)$$

If $\mathcal{D}$ and $\mathcal{D}^*$ are strongly ε_h -open-loop consistent, and

$$\text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h})) \supseteq \text{supp}(P_{\mathcal{D}^*}(s_t, a_{t:t+h})),$$

then $\forall s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V^\bullet(s_t) \leq \frac{\varepsilon_h \gamma}{(1-\gamma)^2} \left[\frac{2}{1 - (1-2\varepsilon_h)\gamma^h} + \frac{1}{1 - (1-\varepsilon_h)\gamma^h} \right] \leq 3\varepsilon_h H^2 \bar{H}. \quad (5.24)$$

Proof. We observe that

$$\begin{aligned} V_{ac}^+(s_t) &= Q_{ac}^+(s_t, a_{t:t+h}^+) \\ &\leq Q^*(s_t, a_t^+). \end{aligned} \quad (D.143)$$

Combining this with Theorem 3, we get

$$Q^*(s_t, a_t^+) \geq V^*(s_t) - \Delta, \quad (D.144)$$

where $\Delta = \frac{\varepsilon_h \gamma}{1-\gamma} \left[\frac{2}{1-(1-2\varepsilon_h)\gamma^h} + \frac{1}{1-(1-\varepsilon_h)\gamma^h} \right]$.

Now, we can bound $V^\bullet$ as follows:

$$\begin{aligned} V^*(s_t) - V^\bullet(s_t) &\leq Q^*(s_t, a_t^+) - Q^\bullet(s_t, a_t^+) + \Delta \\ &\leq \gamma \mathbb{E}_{T(\cdot|s_t, a_t^+)} [V^*(s_{t+1}) - V^\bullet(s_{t+1})] + \Delta \\ &\leq \frac{\varepsilon_h \gamma}{(1-\gamma)^2} \left[\frac{2}{1 - (1-2\varepsilon_h)\gamma^h} + \frac{1}{1 - (1-\varepsilon_h)\gamma^h} \right]. \end{aligned} \quad (D.145)$$

□

D.10 Proof of Proposition 2

In this section, we provide the proof for Proposition 2 (restated below).

Proposition 2 (Comparing Q-chunking backup and n -step return backup) Let $\mathcal{D}$ be strongly ε_h -open-loop consistent and δ_n -sub-optimal, and $\text{supp}(P_{\mathcal{D}}(s_t)) \supseteq \text{supp}(P_{\mathcal{D}^*}(s_t))$. Let $\pi_n^+ : s_t \mapsto \arg \max_{a_t} \hat{Q}_n^+(s_t, a_t)$ be the policy learned from $\mathcal{D}$, via n -step return backup:

$$\hat{Q}_n^+(s_t, a_t) = \mathbb{E} \left[R_{t:t+n} + \gamma^n \hat{Q}_n^+(s_{t+n}, \pi_n^+(s_{t+n})) \right]. \quad (5.21)$$

Then, for all $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$ (and with $\bar{H}_n = 1/(1 - \gamma^n)$),

$$\begin{aligned} V_{\text{ac}}^+(s_t) - \hat{V}_n^+(s_t) &\geq \frac{\delta_n}{1 - \gamma^n} - \frac{\varepsilon_h \gamma}{1 - \gamma} \left[\frac{2}{1 - (1 - 2\varepsilon_h)\gamma^h} + \frac{1}{1 - (1 - \varepsilon_h)\gamma^h} \right], \\ &\geq \delta_n \bar{H}_n - 3\varepsilon_h H \bar{H}. \end{aligned} \quad (5.22)$$

To prove Proposition 2, we first prove the following helper Lemma 6 to quantify sub-optimality for n -step return policy.

Lemma 6 Let Q_n^* be the solution of the uncorrected n -step return backup equation:

$$Q_n^*(s_t, a_t) = \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_t)} \left[R_{t:t+n} + \gamma^n \max_{a_{t+n}} Q_n^*(s_{t+n}, a_{t+n}) \right] \quad (D.146)$$

The following inequality holds as long as $\mathcal{D}$ is δ_n -sub-optimal:

$$Q^*(s_t, a_t) \geq Q_n^*(s_t, a_t) + \frac{\delta_n}{1 - \gamma^n}, \forall s_t \in \mathcal{S}, a_t \in \mathcal{A} \quad (D.147)$$

where Q^* is the Q-function of the optimal policy in $\mathcal{M}$. For the n -step return policy

$$\pi_n^* : s_t \mapsto \arg \max_{a_t} Q_n^*(s_t, a_t), \quad (D.148)$$

its corresponding value admits a similar bound:

$$V^*(s_t) \geq V_n^*(s_t) + \frac{\delta_n}{1 - \gamma^n}, \forall s_t \quad (D.149)$$

Proof. Using the definition of sub-optimal data (Definition 3), we have

$$\begin{aligned} Q_n^*(s_t, a_t) &= \mathbb{E}_{P_D(\cdot|s_t, a_t)} \left[R_{t:t+n} + \gamma^n \max_{a_{t+n}} Q_n^*(s_{t+n}, a_{t+n}) \right] \\ &\leq Q^*(s_t, a_t) - \delta_n + \gamma^n \mathbb{E}_{P_D(\cdot|s_t, a_t)} \left[\max_{a_{t+n}} Q_n^*(s_{t+n}, a_{t+n}) - V^*(s_{t+h}) \right] \end{aligned} \quad (\text{D.150})$$

Rearranging the inequality above yields

$$Q_n^*(s_t, a_t) - Q^*(s_t, a_t) \leq -\delta_n + \gamma^n \mathbb{E}_{P_D(\cdot|s_t)} [V_n^*(s_{t+n}) - V^*(s_{t+n})], \forall s_t \in \mathcal{S}, a_t \in \mathcal{A} \quad (\text{D.151})$$

By recursively applying the inequality above, we have

$$Q^*(s_t, a_t) \geq Q_n^*(s_t, a_t) + \frac{\delta_n}{1 - \gamma^n}, \forall s_t \in \mathcal{S}, a_t \in \mathcal{A} \quad (\text{D.152})$$

By choosing $a_t^* = \pi_n^*(s_t)$, we see that

$$\begin{aligned} V^*(s_t) &\geq Q^*(s_t, a_t) \\ &\geq Q_n^*(s_t, a_t^*) + \frac{\delta_n}{1 - \gamma^n} \\ &= V_n^*(s_t) + \frac{\delta_n}{1 - \gamma^n} \end{aligned} \quad (\text{D.153})$$

□

Now we are ready to prove the main Proposition 2.

Proof of Proposition 2. From Lemma 6 and Theorem 3, we have

$$V_n^*(s) + \frac{\delta_n}{1 - \gamma^n} \leq V^*(s) \leq V_{ac}^+(s) + \frac{\varepsilon_h \gamma}{1 - \gamma} \left[\frac{2}{1 - (1 - 2\varepsilon_h)\gamma^h} + \frac{1}{1 - (1 - \varepsilon_h)\gamma^h} \right]. \quad (\text{D.154})$$

Rearranging the terms gives

$$V_{ac}^+(s) - V_n^*(s) \geq \frac{\delta_n}{1 - \gamma^n} - \frac{\varepsilon_h \gamma}{1 - \gamma} \left[\frac{2}{1 - (1 - 2\varepsilon_h)\gamma^h} + \frac{1}{1 - (1 - \varepsilon_h)\gamma^h} \right]. \quad (\text{D.155})$$

□

D.11 Proof of Theorem 5

In this section, we provide the proof for Theorem 5 (restated below).

Theorem 5 (Closed-loop AC Policy under Bounded OV) Let $\mathcal{D}^*$ be the data distribution collected by an optimal policy. Assume $\mathcal{D}$ can be decomposed into a mixture of data distributions $\{\mathcal{D}^*, \mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_M\}$ and for some $\vartheta_h^L, \vartheta_h^G \geq 0$, they satisfy the following two conditions:

1. Locally bounded optimality variability condition: every $\mathcal{D}_i$ (including $\mathcal{D}^*$) exhibits ϑ_h^L -bounded variability in optimality conditioned on s_t, a_t for all $(s_t, a_t) \in \text{supp}(P_{\mathcal{D}_i}(s_t, a_t))$, and

2. Globally bounded optimality variability condition: $\mathcal{D}$ as a whole exhibits ϑ_h^G -variability in optimality conditioned on $s_t, a_{t:t+h}$ for all $(s_t, a_{t:t+h}) \in \text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h}))$.

Then for all $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s_t) - V^\bullet(s_t) \leq \frac{\vartheta_h^L}{1-\gamma} + \frac{\vartheta_h^G + \gamma^h \min(\vartheta_h^L, \vartheta_h^G)}{(1-\gamma)(1-\gamma^h)} \leq \vartheta_h^L H + 2\vartheta_h^G H \bar{H}. \quad (5.26)$$

Proof of Theorem 5. Consider any $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$. Let $a_{t:t+h}^+ = \pi_{\text{ac}}^+(s_t)$ and

$$a_{t:t+h}^\circ := \arg \max_{a_{t:t+h} \in \text{supp}(P_{\mathcal{D}^*}(a_{t:t+h}|s_t))} [\mathbb{E}_{P_{\mathcal{D}^*}(\cdot|s_t, a_{t:t+h})} [R_{t:t+h} + \gamma^h V^*(s_{t+h})]]. \quad (D.156)$$

We first observe that

$$\mathbb{E}_{P_{\mathcal{D}^*}(\cdot|s_t, a_{t:t+h}^\circ)} [R_{t:t+h} + \gamma^h V^*(s_{t+h})] \geq V^*(s_t), \quad (D.157)$$

because

$$V^*(s_t) = \mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}^*}(\cdot|s_t)} [\mathbb{E}_{P_{\mathcal{D}^*}(\cdot|s_t, a_{t:t+h})} [R_{t:t+h} + \gamma^h V^*(s_{t+h})]], \quad (D.158)$$

and the maximum value of a random variable is no less than its expectation.

Let

$$\tilde{Q}_{\min}(s_t, a_{t:t+h}^\circ) := \min_{\text{supp}(P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^\circ))} [R_{t:t+h} + V^*(s_{t+h})], \quad (D.159)$$

$$\tilde{Q}_{\max}(s_t, a_{t:t+h}^\circ) := \max_{\text{supp}(P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^\circ))} [R_{t:t+h} + V^*(s_{t+h})]. \quad (D.160)$$

Since $\mathcal{D}$ exhibits ϑ_h^G -variability in optimality, we have

$$\tilde{Q}_{\min}(s_t, a_{t:t+h}^\circ) \geq \tilde{Q}_{\max}(s_t, a_{t:t+h}^\circ) - \vartheta_h^G. \quad (D.161)$$

$$\begin{aligned}
& V^*(s_t) - Q^*(s_t, a_t^+) \\
&= V^*(s_t) - \hat{V}_{\text{ac}}^+(s_t) + \hat{V}_{\text{ac}}^+(s_t) - Q^*(s_t, a_t^+) \\
&= V^*(s_t) - \hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}^+) + \hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}^+) - Q^*(s_t, a_t^+) \\
&\leq V^*(s_t) - \hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}^\circ) + \vartheta_h^L + \gamma^h \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^+)} \left[\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h}) \right] \\
&= V^*(s_t) - \hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}^\circ) + \vartheta_h^L + \gamma^h \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^+)} \left[\hat{V}_{\text{ac}}^+(s_{t+h}) - Q^*(s_{t+h}, a_{t+h}^+) \right] - \\
&\quad \gamma^h \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^+)} \left[V^*(s_{t+h}) - Q^*(s_{t+h}, a_{t+h}^+) \right].
\end{aligned} \tag{D.162}$$

We can use it to lower bound $\hat{V}_{\text{ac}}^+(s_t)$ as follows:

$$\begin{aligned}
\hat{V}_{\text{ac}}^+(s_t) &= \hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}^+) \\
&\geq \hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}^\circ) \\
&= \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^\circ)} \left[R_{t:t+h} + \gamma^h \hat{V}_{\text{ac}}^+(s_{t+h}) \right] \\
&= \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^\circ)} \left[R_{t:t+h} + \gamma^h V^*(s_{t+h}) \right] + \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^\circ)} \left[\gamma^h (\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h})) \right] \\
&\geq \tilde{Q}_{\min}(s_t, a_{t:t+h}^\circ) + \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^\circ)} \left[\gamma^h (\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h})) \right] \\
&\geq \tilde{Q}_{\max}(s_t, a_{t:t+h}^\circ) - \vartheta_h^G + \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^\circ)} \left[\gamma^h (\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h})) \right] \\
&\geq \mathbb{E}_{P_{\mathcal{D}^*}(\cdot|s_t, a_{t:t+h}^\circ)} \left[R_{t:t+h} + \gamma^h V^*(s_{t+h}) \right] - \vartheta_h^G + \gamma^h \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^\circ)} \left[(\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h})) \right] \\
&\geq V^*(s_t) - \vartheta_h^G + \gamma^h \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^\circ)} \left[(\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h})) \right] \\
&\geq V^*(s_t) - \frac{\vartheta_h^G}{1 - \gamma^h}.
\end{aligned} \tag{D.163}$$

Let $\mathbb{M}^+ = \{\tilde{\mathcal{D}}_1, \dots, \tilde{\mathcal{D}}_{M^+}\}$ be all data distributions from $\{\mathcal{D}^*, \mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_M\}$ where $(s_t, a_{t:t+h}^+)$ is in the support. Let $\tilde{\mathcal{D}}^+$ be any mixture of $\mathbb{M}^+$ where each mixture component has non-zero weight:

$$P_{\tilde{\mathcal{D}}^+} = \sum_{i=1}^M w_i P_{\tilde{\mathcal{D}}_i}, \tag{D.164}$$

where $w_i > 0, \sum_i w_i = 1$.

Let

$$\tilde{Q}_{\min}^*(s_t, a_t) := \min_{\text{supp}(P_{\mathcal{D}^*}(\cdot|s_t, a_t))} [R_{t:t+h} + V^*(s_{t+h})], \quad (\text{D.165})$$

$$\tilde{Q}_{\max}^*(s_t, a_t) := \max_{\text{supp}(P_{\mathcal{D}^*}(\cdot|s_t, a_t))} [R_{t:t+h} + V^*(s_{t+h})], \quad (\text{D.166})$$

$$\tilde{Q}_{\min}^i(s_t, a_t) := \min_{\text{supp}(P_{\mathcal{D}^i}(\cdot|s_t, a_t))} [R_{t:t+h} + V^*(s_{t+h})], \quad (\text{D.167})$$

$$\tilde{Q}_{\max}^i(s_t, a_t) := \max_{\text{supp}(P_{\mathcal{D}^i}(\cdot|s_t, a_t))} [R_{t:t+h} + V^*(s_{t+h})], \quad (\text{D.168})$$

$$\tilde{Q}_{\max}^+(s_t, a_t^+) := \max_{\text{supp}(P_{\tilde{\mathcal{D}}^+}(\cdot|s_t, a_t^+))} [R_{t:t+h} + V^*(s_{t+h})], \quad (\text{D.169})$$

$$\tilde{Q}_{\max}^+(s_t, a_{t:t+h}^+) := \max_{\text{supp}(P_{\tilde{\mathcal{D}}^+}(\cdot|s_t, a_{t:t+h}^+))} [R_{t:t+h} + V^*(s_{t+h})]. \quad (\text{D.170})$$

The minimum and the maximum is over the remaining trajectory conditioned on s_t, a_t or $s_t, a_{t:t+h}$ that is still in the support of the corresponding data distribution.

From the ϑ_h^L -bounded variability in optimality and the Assumption 2 of each data mixture, we observe that

$$Q^*(s_t, a_t) \geq \tilde{Q}_{\min}^i(s_t, a_t) \geq \tilde{Q}_{\max}^i(s_t, a_t) - \vartheta_h^L, \quad \forall i \in \{1, 2, \dots, M\} \quad (\text{D.171})$$

$$Q^*(s_t, a_t) \geq \tilde{Q}_{\min}^*(s_t, a_t) \geq \tilde{Q}_{\max}^*(s_t, a_t) - \vartheta_h^L. \quad (\text{D.172})$$

We can then derive that

$$\begin{aligned} \tilde{Q}_{\max}^+(s_t, a_t^+) &= \max(\tilde{Q}_{\max}^*(s_t, a_t^+), \tilde{Q}_{\max}^1(s_t, a_t^+), \dots, \tilde{Q}_{\max}^M(s_t, a_t^+)) \\ &\leq Q^*(s_t, a_t) + \vartheta_h^L. \end{aligned} \quad (\text{D.173})$$

With this, we can now upper-bound $\hat{V}_{\text{ac}}^+(s_t)$ as follows:

$$\begin{aligned} \hat{V}_{\text{ac}}^+(s_t) &= \hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}^+) \\ &= \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^+)} [R_{t:t+h} + \gamma^h \hat{V}_{\text{ac}}^+(s_{t+h})] \\ &= \mathbb{E}_{P_{\tilde{\mathcal{D}}^+}(\cdot|s_t, a_{t:t+h}^+)} [R_{t:t+h} + \gamma^h \hat{V}_{\text{ac}}^+(s_{t+h})] \\ &= \mathbb{E}_{P_{\tilde{\mathcal{D}}^+}(\cdot|s_t, a_{t:t+h}^+)} [R_{t:t+h} + \gamma^h V^*(s_{t+h})] + \gamma^h \mathbb{E}_{P_{\tilde{\mathcal{D}}^+}(\cdot|s_t, a_{t:t+h}^+)} [\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h})] \\ &\leq \tilde{Q}_{\max}^+(s_t, a_{t:t+h}^+) + \gamma^h \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^+)} [\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h})] \\ &\leq \tilde{Q}_{\max}^+(s_t, a_t^+) + \gamma^h \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^+)} [\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h})] \\ &\leq Q^*(s_t, a_t^+) + \vartheta_h^L + \gamma^h \mathbb{E}_{P_{\mathcal{D}}(\cdot|s_t, a_{t:t+h}^+)} [\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h})]. \end{aligned} \quad (\text{D.174})$$

Let

$$\Delta(s_t) := V^*(s_t) - Q^*(s_t, a_t^+). \quad (\text{D.175})$$

$$\hat{\Delta}(s_t) := \hat{V}_{\text{ac}}^+(s_t) - Q^*(s_t, a_t^+). \quad (\text{D.176})$$

From the inequalities above, we have

$$\hat{\Delta}(s_t) \leq \vartheta_h^L + \gamma^h \sup_{s_{t+h}} [\hat{\Delta}(s_{t+h}) - \Delta(s_{t+h})], \quad (\text{D.177})$$

$$0 \leq \Delta(s_t) \leq \frac{\vartheta_h^G}{1 - \gamma^h} + \hat{\Delta}(s_t), \quad (\text{D.178})$$

$$\hat{\Delta}(s_t) - \Delta(s_t) \leq \min \left\{ \frac{\vartheta_h^G}{1 - \gamma^h}, \hat{\Delta}(s_t) \right\}. \quad (\text{D.179})$$

The minimum operator allows us to obtain two upper bounds on Δ :

$$\Delta(s_t) \leq \vartheta_h^L + \frac{(1 + \gamma^h)\vartheta_h^G}{1 - \gamma^h}, \quad (\text{D.180})$$

$$\Delta(s_t) \leq \frac{\vartheta_h^G}{1 - \gamma^h} + \hat{\Delta}(s_t) \leq \frac{\vartheta_h^L + \vartheta_h^G}{1 - \gamma^h}. \quad (\text{D.181})$$

Finally, combining these two upper bounds together and recursively applying the inequality yields our desired results:

$$V^*(s_t) - Q^\bullet(s_t, a_t^+) \leq \frac{\vartheta_h^L}{1 - \gamma} + \frac{\vartheta_h^L}{(1 - \gamma)(1 - \gamma^h)} + \frac{\gamma^h \min(\vartheta_h^G, \vartheta_h^L)}{(1 - \gamma)(1 - \gamma^h)}. \quad (\text{D.182})$$

□

D.12 Proof of Theorem 6

In this section, we provide the proof for Theorem 6 (restated below).

Theorem 6 (Worst-case Closed-loop AC Policy under BOV) For any $h > 1, \gamma \in (0, 1), \vartheta_h^G, \vartheta_h^L \in \left(0, \frac{\gamma - \gamma^h}{4(1-\gamma)}\right], c \in \left[0, \frac{\gamma - \gamma^h}{4(1-\gamma^h)}\right), \sigma \in \left(0, \frac{\min(\vartheta_h^G, \vartheta_h^L)}{1-\gamma}\right)$, there exists $\mathcal{M}$ and $\mathcal{D}$ satisfying the assumptions in Theorem 5 such that there exists $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$, where

$$V^*(s_t) - V^\bullet(s_t) = \frac{\vartheta_h^L}{1-\gamma} + \frac{\vartheta_h^G + \gamma^h \min(\vartheta_h^L, \vartheta_h^G)}{(1-\gamma)(1-\gamma^h)} - \sigma, V^*(s_t) - V_{\text{ac}}^+(s_t) \geq \frac{c}{1-\gamma}. \quad (5.27)$$

To show that our upper bound is achievable, we need to carefully design both the MDP and the data distribution. For clarity of the proof, we divide up the construction into two parts. The first part (Lemma 7) focuses on designing part of the MDP and two data distributions $\mathcal{D}^*$ and $\mathcal{D}^\diamond$ such that any action chunk that has a value bigger than $V^* - \frac{\vartheta_h^G}{1-\gamma^h}$ is preferred over the action chunks in $\mathcal{D}^*$ and $\mathcal{D}^\diamond$. The second part (Lemma 8) focuses on constructing the remaining MDP and the $\mathcal{D}^\Delta$ that contains the action chunk that π_{ac}^+ picks where $\hat{V}_{\text{ac}}^+$ overestimates the value of this action chunk by $\vartheta_h^L + \frac{\gamma^h \min(\vartheta_h^L, \vartheta_h^G)}{1-\gamma^h}$. Finally, we assemble these two results (combining $\mathcal{D}^*, \mathcal{D}^\diamond, \mathcal{D}^\Delta$) to show that the MDP and the mixture data achieve our upper bound *exactly*.

Lemma 7 (“The Castle”) For $\delta \in (0, 1), \vartheta_h^G < \frac{\gamma - \gamma^h}{2(1-\gamma)}$, consider a 2-state, 2-action MDP in Figure D.5. Let there be two data distributions, $\mathcal{D}^*$ and $\mathcal{D}^\diamond$. $\mathcal{D}^*$ is collected by the following optimal closed-loop policy from X and Y :

$$\pi^*(X) = 0, \pi^*(Y) = 1. \quad (D.183)$$

$\mathcal{D}^\diamond$ is collected by the following optimal closed-loop policy from X and Y :

$$\pi^\diamond(X) = 1, \pi^\diamond(Y) = 0. \quad (D.184)$$

Let $\mathcal{D}$ be a mixture of $\mathcal{D}^*$ and $\mathcal{D}^\diamond$ with

$$P_{\mathcal{D}} = (1 - \varsigma)P_{\mathcal{D}^*} + \varsigma P_{\mathcal{D}^\diamond}. \quad (D.185)$$

There exists $c_1 \in (0, 1/2)$ such that

1. $\mathcal{D}^*$ and $\mathcal{D}^\diamond$ both individually exhibits 0-variability in optimality conditioned on s_t, a_t for all $s_t, a_t \in \text{supp}(P_{\mathcal{D}}(s_t, a_t))$,

2. $\mathcal{D}$ exhibits ϑ_h^G -variability in optimality conditioned on $s_t, a_{t:t+h}$ for all $s_t, a_{t:t+h} \in \text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h}))$,

and

$$\hat{V}_{\text{ac}}^+(X) = \hat{V}_{\text{ac}}^+(Y) = \frac{1 - \gamma + \varsigma(\gamma - \gamma^h)}{2(1 - \gamma^h)(1 - \gamma)} - \frac{\varsigma\vartheta_h^G}{1 - \gamma^h}. \quad (\text{D.186})$$

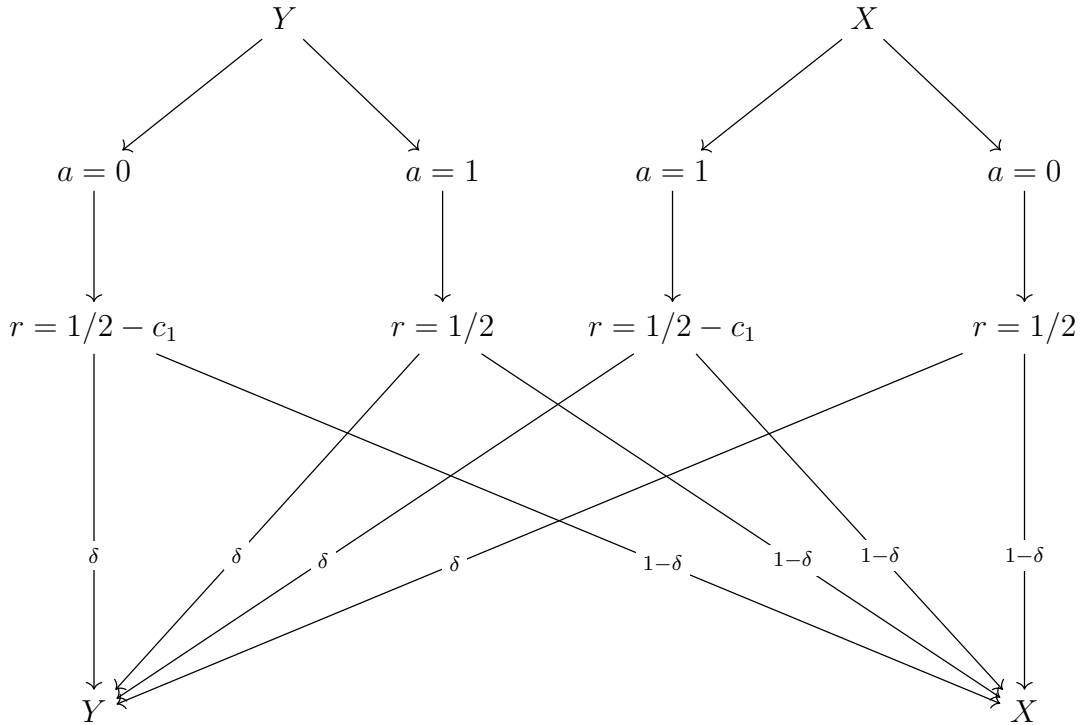


Figure D.5: **MDP construction Part 1 for Theorem 6 (“the castle”)**. This diagram describes states X and Y and how actions $a = 0$ and $a = 1$ transition between them. The main purpose of this construction is to make $\hat{V}_{\text{ac}}^+(X)$ underestimate V^* by exactly $\vartheta_h^G/(1-\gamma^h)$. This allows the action chunk that appears in the second part of the construction to be preferred (by π_{ac}^+) over the action chunks that start with $a = 0$ or $a = 1$.

Proof. Set

$$c_1 = \frac{(1 - \gamma)\vartheta_h^G}{\gamma - \gamma^h}. \quad (\text{D.187})$$

We first check whether $c_1 \in (0, 1/2)$. For the upper bound, it is clear that $c_1 < 1/2$ because $\vartheta_h^G < \frac{\gamma - \gamma^h}{2(1 - \gamma)}$. For the lower bound, $c > 0$ because all terms in the fraction are positive.

We now check the two optimality variability conditions. The first (local) one is trivial because $\pi^\diamond$ always receives $r = 1/2 - c_1$ and $\pi^\star$ always receives $r = 1/2$, and the optimal value for X and Y are both $V^\star(X) = V^\star(Y) = \frac{1}{2(1-\gamma)}$.

Next, we check the second (global) condition by analyzing all possible states and action chunks in $\mathcal{D}$. We observe that for any $a_{t:t+h}$ that starts with $a_t = 0$, we have

$$\tilde{Q}_{\min}(X, a_{t:t+h}) = \frac{1 - 2c_1(\gamma - \gamma^h)}{2(1 - \gamma)}, \quad (\text{D.188})$$

$$\tilde{Q}_{\max}(X, a_{t:t+h}) = \frac{1}{2(1 - \gamma)}, \quad (\text{D.189})$$

which gives

$$\tilde{Q}_{\max}(X, a_{t:t+h}) - \tilde{Q}_{\min}(X, a_{t:t+h}) = \vartheta_h^G. \quad (\text{D.190})$$

By symmetry, we also have

$$\tilde{Q}_{\max}(Y, a_{t:t+h}) - \tilde{Q}_{\min}(Y, a_{t:t+h}) = \vartheta_h^G. \quad (\text{D.191})$$

for all $a_{t:t+h}$ that starts with $a_t = 1$.

Now, for any $a_{t:t+h}$ that starts with $a_t = 1$, we have

$$\tilde{Q}_{\min}(X, a_{t:t+h}) = \frac{\gamma - 2c_1(\gamma - \gamma^h)}{2(1 - \gamma)}, \quad (\text{D.192})$$

$$\tilde{Q}_{\max}(X, a_{t:t+h}) = \frac{\gamma}{2(1 - \gamma)}, \quad (\text{D.193})$$

which admits the same gap as the case when $a_t = 0$. The same also holds for Y with $a_t = 1$. Thus, $\mathcal{D}$ exhibits ϑ_h^G -variability in optimality conditioned on $s_t, a_{t:t+h}$ for all $s_t, a_{t:t+h} \in \text{supp}(P_{\mathcal{D}}(s_t, a_{t:t+h}))$.

Finally, we check for the value,

$$\begin{aligned} \hat{V}_{\text{ac}}^+(X) &= \hat{V}_{\text{ac}}^+(Y) = (1 - \varsigma)/2 + \varsigma(1/2 + (1 - 2c_1)\frac{\gamma - \gamma^h}{2(1 - \gamma)}) \\ &= \frac{1}{1 - \gamma^h} \left[1/2 + \varsigma \frac{(1 - 2c_1)(\gamma - \gamma^h)}{2(1 - \gamma)} \right] \\ &= \frac{1}{2(1 - \gamma^h)} \left[1 + \varsigma \frac{\gamma - \gamma^h - 2(1 - \gamma)\vartheta_h^G}{1 - \gamma} \right] \\ &= \frac{1 - \gamma + \varsigma(\gamma - \gamma^h)}{2(1 - \gamma^h)(1 - \gamma)} - \frac{\varsigma\vartheta_h^G}{1 - \gamma^h}, \end{aligned} \quad (\text{D.194})$$

as desired. $\square$

Lemma 8 (“The Flower”) Assume $\vartheta_h^G \in \left(0, \frac{1-\gamma^h}{8}\right]$, $\vartheta_h^L \in \left(0, \frac{\gamma-\gamma^h}{4(1-\gamma)}\right]$, $\gamma \in (0, 1)$, and Consider a 5-state, 3-action MDP in Figure D.6 building on top of the transitions that already in Figure D.5. Let $\mathcal{D}^\Delta$ be a data distribution induced by a cycling, time-dependent (with a time cycle length of h) policy π^Δ (we use the subscript to indicate the time step from 0 to $h-1$):

$$\pi_0^\Delta(s_t = X) = \pi_0^\Delta(s_t = \tilde{X}) = 2, \quad (\text{D.195})$$

$$\pi_0^\Delta(s_t = Y) = 3 \quad (\text{D.196})$$

$$\pi_k^\Delta(s_{t+k} = \tilde{C}) = \pi_k^\Delta(s_{t+k} = \tilde{D}) = 2, \quad \forall k \in \{1, 2, \dots, h-2\}, \quad (\text{D.197})$$

$$\pi_k^\Delta(s_{t+h-1} = \tilde{C}) = \pi_k^\Delta(s_{t+h-1} = \tilde{D}) = 0, \quad (\text{D.198})$$

$$\pi_k^\Delta(s_{t+k} = X) = 0, \quad \forall k \in \{1, 2, \dots, h-1\}, \quad (\text{D.199})$$

$$\pi_k^\Delta(s_{t+k} = Y) = 1, \quad \forall k \in \{1, 2, \dots, h-1\}. \quad (\text{D.200})$$

Let $\hat{V}_{\text{ac}}^+$ be the nominal value of the action chunking policy π_{ac}^+ learned from $\mathcal{D}^\Delta$ and let

$$\Delta = \vartheta_h^L + \frac{\vartheta_h^G}{1-\gamma^h} + \frac{\gamma^h \min(\vartheta_h^G, \vartheta_h^L)}{1-\gamma^h}. \quad (\text{D.201})$$

For any $c \in \left[0, \frac{\gamma-\gamma^h}{4(1-\gamma^h)}\right)$, there exists some $0 < c_2 \leq 1/2$, $0 < c_3 \leq 1/2$, $\delta, \delta_2 \in (0, 1)$, such that for every $0 < \tilde{\Delta} < \min\left(\Delta, \frac{2\vartheta_h^G}{1-\gamma^h}\right)$,

1. $\mathcal{D}^\Delta$ exhibits 0-variability in optimality conditioned on $s_t, a_{t:t+h}$ for all $s_t, a_{t:t+h} \in \text{supp}(P_{\mathcal{D}^\Delta}(s_t, a_{t:t+h}))$,
2. $\mathcal{D}^\Delta$ exhibits ϑ_h^L -variability in optimality conditioned on s_t, a_t for all $s_t, a_t \in \text{supp}(P_{\mathcal{D}^\Delta}(s_t, a_t))$,

and

$$\hat{V}_{\text{ac}}^+(X) = \frac{1}{2(1-\gamma)} - \frac{\vartheta_h^G}{1-\gamma^h} + \tilde{\Delta}, \quad (\text{D.202})$$

$$V^*(X) - V^\bullet(X) = \frac{\Delta - \tilde{\Delta}}{1-\gamma}, \quad (\text{D.203})$$

$$V^*(X) - V_{\text{ac}}^+(X) \geq \frac{c}{1-\gamma}, \quad (\text{D.204})$$

$$V^*(X) - V_{\text{ac}}^*(X) \geq \frac{c}{1-\gamma}. \quad (\text{D.205})$$

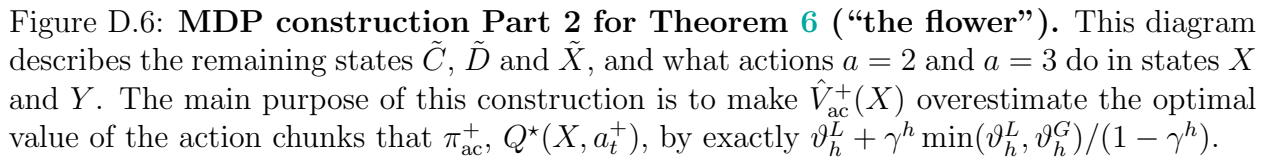
Proof. Without the loss of generality, we assume we always start from state X . Due to symmetry, the same analysis applies to state Y (with the first action being $a_t = 3$ rather than $a_t = 2$).

Due to cycling nature of the data collection policy, we observe that all action chunks starting from X are in the form of $a_{t:t+h} = (2, \underbrace{\cdots}_{0\text{'s and } 1\text{'s}})$ or $a_{t:t+h} = (2, 2, \dots, 2, 0)$. These two possibilities correspond to two different paths that the data collection policy takes:

- $a_{t:t+h}^\circ = (2, \underbrace{\cdots}_{0\text{'s and } 1\text{'s}})$: Stay in either X or Y . The agent going on this path receives a constant reward of $1/2$ except the first step where it receives a reward of $(1 - c_2)/2$.
- $a_{t:t+h}^\Delta = (2, 2, \dots, 2, 0)$: Visit $\tilde{C}$ and then stays there for $h - 1$ until it goes out with $a = 0$ to visit $\tilde{X}$. The agent going on this path receives a constant reward of $(1 + c_3)/2$ except the first step where it receives a reward of $(1 - c_2)/2$.

Similarly, all action chunks starting from $\tilde{X}$ are in the form of $a_{t:t+h} = (2, \underbrace{\cdots}_{0\text{'s and } 1\text{'s}})$ or $a_{t:t+h} = (2, 2, \dots, 2, 0)$. These two possibilities correspond to two different paths that the data collection policy takes:

- $a_{t:t+h}^\circ = (2, \underbrace{\cdots}_{0\text{'s and } 1\text{'s}})$: Stay in either X or Y . The agent going on this path receives a constant reward of $1/2$.
- $a_{t:t+h}^\Delta = (2, 2, \dots, 2, 0)$: Visit $\tilde{C}$ and then stays there for $h - 1$ until it goes out with $a = 0$ to visit $\tilde{X}$. The agent going on this path receives a constant reward of $(1 + c_4)/2$ except the first step where it receives a reward of $1/2$.



Now, we divide up the problem into two cases depending on the relative values of ϑ_h^L and ϑ_h^G .

1. *Case $\vartheta_h^L \geq \vartheta_h^G$:*

Set

$$c_2 = 2 \left[\vartheta_h^L + \frac{(1 + \gamma^h)\vartheta_h^G}{1 - \gamma^h} \right] - 2\tilde{\Delta} > 0, \quad (\text{D.206})$$

$$c_3 = \frac{2(1 - \gamma)\vartheta_h^L}{\gamma - \gamma^h} > 0, \quad (\text{D.207})$$

$$c_4 = \frac{2(1 - \gamma)\vartheta_h^G}{\gamma - \gamma^h} > 0. \quad (\text{D.208})$$

Next, we check that $c_2, c_3, c_4 \leq 1/2$.

We first observe that

$$(1 - \gamma)(1 - \gamma^h) - 2(\gamma - \gamma^h) = 1 - 3\gamma + \gamma^h(\gamma + 1) \leq 1 - 3\gamma + \gamma(\gamma + 1) = (1 - \gamma)^2 \geq 0. \quad (\text{D.209})$$

Dividing both sides by $8(1 - \gamma)$ yields

$$\frac{1 - \gamma^h}{8} \geq \frac{\gamma - \gamma^h}{4(1 - \gamma)} \geq \vartheta_h^L \geq \vartheta_h^G. \quad (\text{D.210})$$

Now, using the inequality above, we have

$$\begin{aligned} c_2 &= 2 \left[\vartheta_h^L + \frac{(1 + \gamma^h)\vartheta_h^G}{1 - \gamma^h} \right] - \tilde{2}\Delta \\ &\leq 2 \left[\vartheta_h^L + \frac{(1 + \gamma^h)\vartheta_h^L}{1 - \gamma^h} \right] \\ &\leq \frac{4\vartheta_h^L}{1 - \gamma^h} \\ &\leq 1/2. \end{aligned} \quad (\text{D.211})$$

Furthermore,

$$c_4 \leq c_3 = \frac{2(1 - \gamma)\vartheta_h^L}{\gamma - \gamma^h} \leq 1/2. \quad (\text{D.212})$$

Next, we check the data distribution $\mathcal{D}^\Delta$ satisfies both optimality variability conditions. We first note that we only need to check for $s_t \in \{X, \tilde{X}\}$ because all other states are out

of the support due to the cycling nature of the data collection policies. The first (global) optimality condition is trivial because the h -step reward received is deterministic conditioned on $a_{t:t+h} \in \{a_{t:t+h}^\circ, a_{t:t+h}^\Delta\}$, and the optimal value of $V^*(s_{t+h})$ is always $\frac{1}{2(1-\gamma)}$. This leads to 0-variability in optimality conditioned on $s_t, a_{t:t+h}$. For the second (local) optimality condition, we check the difference in optimality for two paths from $s_t, a_t = 2$ for both $s_t = X$ and $s_t = \tilde{X}$.

For $s_t = X$, the optimality gap is

$$c_3 \frac{\gamma - \gamma^h}{2(1 - \gamma^h)} = \vartheta_h^L. \quad (\text{D.213})$$

For $s_t = \tilde{X}$, the optimality gap is

$$c_4 \frac{\gamma - \gamma^h}{2(1 - \gamma^h)} = \vartheta_h^G \leq \vartheta_h^L. \quad (\text{D.214})$$

This concludes that the second (local) optimality condition is also satisfied.

Next, we first analyze which action chunk π_{ac}^+ prefers by computing $\hat{Q}_{\text{ac}}^+$'s:

$$\hat{Q}_{\text{ac}}^+(X, a_{t:t+h}^\circ) = \frac{1}{2} \left[(1 - c_2) + \frac{\gamma - \gamma^h}{1 - \gamma} \right] + \gamma^h \hat{V}_{\text{ac}}^+(X), \quad (\text{D.215})$$

$$\hat{Q}_{\text{ac}}^+(X, a_{t:t+h}^\Delta) = \frac{1}{2} \left[(1 - c_2) + (1 + c_3) \frac{\gamma - \gamma^h}{1 - \gamma} \right] + \gamma^h \hat{V}_{\text{ac}}^+(\tilde{X}), \quad (\text{D.216})$$

$$\hat{Q}_{\text{ac}}^+(\tilde{X}, a_{t:t+h}^\circ) = \frac{1}{2} \left[\frac{1 - \gamma^h}{1 - \gamma} \right] + \gamma^h \hat{V}_{\text{ac}}^+(X), \quad (\text{D.217})$$

$$\hat{Q}_{\text{ac}}^+(\tilde{X}, a_{t:t+h}^\Delta) = \frac{1}{2} \left[1 + (1 + c_4) \frac{\gamma - \gamma^h}{1 - \gamma} \right] + \gamma^h \hat{V}_{\text{ac}}^+(\tilde{X}). \quad (\text{D.218})$$

We first observe that

$$\begin{aligned} \hat{Q}_{\text{ac}}^+(\tilde{X}, a_{t:t+h}^\Delta) - \hat{Q}_{\text{ac}}^+(X, a_{t:t+h}^\Delta) &= \frac{1}{2} \left[c_2 - (c_3 - c_4) \frac{\gamma - \gamma^h}{1 - \gamma} \right] \\ &= \vartheta_h^L + \frac{(1 + \gamma^h) \vartheta_h^G}{1 - \gamma^h} - \vartheta_h^L + \vartheta_h^G - \tilde{\Delta} \\ &= \frac{2\vartheta_h^G}{1 - \gamma^h} - \tilde{\Delta} \\ &> 0. \end{aligned} \quad (\text{D.219})$$

Also,

$$\hat{Q}_{\text{ac}}^+(\tilde{X}, a_{t:t+h}^\circ) - \hat{Q}_{\text{ac}}^+(X, a_{t:t+h}^\circ) = c_2 > 0 \quad (\text{D.220})$$

Therefore,

$$\begin{aligned}\hat{V}_{\text{ac}}^+(X) &= \max(\hat{Q}_{\text{ac}}^+(X, a_{t:t+h}^\circ), \hat{Q}_{\text{ac}}^+(X, a_{t:t+h}^\Delta)) \\ &< \max(\hat{Q}_{\text{ac}}^+(\tilde{X}, a_{t:t+h}^\circ), \hat{Q}_{\text{ac}}^+(\tilde{X}, a_{t:t+h}^\Delta)) \\ &= \hat{V}_{\text{ac}}^+(\tilde{X}).\end{aligned}\tag{D.221}$$

Now, we can compare the values for the action chunks for X and $\tilde{X}$:

$$\hat{Q}_{\text{ac}}^+(X, a_{t:t+h}^\Delta) - \hat{Q}_{\text{ac}}^+(X, a_{t:t+h}^\circ) = c_3 \frac{\gamma - \gamma^h}{2(1 - \gamma)} + \gamma^h (\hat{V}_{\text{ac}}^+(\tilde{X}) - \hat{V}_{\text{ac}}^+(X)) > 0,\tag{D.222}$$

$$\hat{Q}_{\text{ac}}^+(\tilde{X}, a_{t:t+h}^\Delta) - \hat{Q}_{\text{ac}}^+(\tilde{X}, a_{t:t+h}^\circ) = c_4 \frac{\gamma - \gamma^h}{2(1 - \gamma)} + \gamma^h (\hat{V}_{\text{ac}}^+(\tilde{X}) - \hat{V}_{\text{ac}}^+(X)) > 0,\tag{D.223}$$

since $c_3, c_4 > 0$ and $h > 1, 0 < \gamma < 1$ (and thus $\frac{\gamma - \gamma^h}{1 - \gamma} > 0$).

This concludes that $\pi_{\text{ac}}^+(X) = \pi_{\text{ac}}^+(\tilde{X}) = a_{t:t+h}^\Delta = (2, 2, \dots, 2, 0)$ and thus

$$\hat{V}_{\text{ac}}^+(\tilde{X}) = \frac{1 - \gamma + (\gamma - \gamma^h)(1 + c_4)}{2(1 - \gamma^h)(1 - \gamma)},\tag{D.224}$$

and

$$\begin{aligned}\hat{V}_{\text{ac}}^+(X) &= \frac{1}{2} \left[(1 - c_2) + (1 + c_3) \frac{\gamma - \gamma^h}{1 - \gamma} \right] + \frac{\gamma^h}{1 - \gamma^h} \hat{V}_{\text{ac}}^+(\tilde{X}) \\ &= \frac{1}{2(1 - \gamma)} - \frac{\vartheta_h^G}{1 - \gamma^h} + \frac{\tilde{\Delta}}{2}.\end{aligned}\tag{D.225}$$

We can now compute the remaining values as follows:

$$V^\star(X) = \frac{1}{2(1 - \gamma)},\tag{D.226}$$

$$Q^\star(X, a = 2) = \frac{(1 - c_2)(1 - \gamma) + \gamma}{2(1 - \gamma)},\tag{D.227}$$

$$Q^\bullet(X, a = 2) = \frac{1 - c_2}{2(1 - \gamma)}.\tag{D.228}$$

Substituting the value of c_2 yields

$$V^\star(X) - V^\bullet(X) = \frac{\vartheta_h^L}{1 - \gamma} + \frac{(1 + \gamma^h)\vartheta_h^G}{(1 - \gamma)(1 - \gamma^h)} - \frac{\tilde{\Delta}}{2(1 - \gamma)}.\tag{D.229}$$

2. Case $\vartheta_h^L < \vartheta_h^G$:

Set

$$\Delta = 2 \left[\frac{\vartheta_h^L + \vartheta_h^G}{1 - \gamma^h} \right] \quad (\text{D.230})$$

$$c_2 = 2 \left[\frac{\vartheta_h^L + \vartheta_h^G}{1 - \gamma^h} \right] - \tilde{\Delta} > 0, \quad (\text{D.231})$$

$$c_3 = c_4 = \frac{2(1 - \gamma)\vartheta_h^L}{\gamma - \gamma^h} > 0 \quad (\text{D.232})$$

where again $\tilde{\Delta}$ is any value that satisfies $0 < \tilde{\Delta} \leq \Delta$.

From the definitions above and the value range of ϑ_h^G ($\vartheta_h^G \leq \frac{1-\gamma^h}{4}$), it is clear that

$$c_3 = c_4 < c_2 \leq \frac{4\vartheta_h^G}{1 - \gamma^h} \leq \frac{2(1 - \gamma)}{\gamma - \gamma^h} \leq 1/2. \quad (\text{D.233})$$

Next, we check the data distribution $\mathcal{D}^\Delta$ satisfies both optimality variability conditions. With the same argument as the previous case, we can quickly conclude that the global optimality condition is satisfied. We just need to show the remaining local optimality condition. We repeat the procedure from the previous case.

For $s_t = X$, the local optimality gap is

$$c_3 \frac{\gamma - \gamma^h}{2(1 - \gamma^h)} = \vartheta_h^L. \quad (\text{D.234})$$

For $s_t = \tilde{X}$ the local optimality gap is the same because $c_4 = c_3$:

$$c_4 \frac{\gamma - \gamma^h}{2(1 - \gamma^h)} = \vartheta_h^L. \quad (\text{D.235})$$

This concludes that the second (local) optimality condition is also satisfied for the second case.

Now, we can follow the same procedure as the previous case to show that $\hat{Q}_{\text{ac}}^+(X, a_{t:t+h}^\Delta) - \hat{Q}_{\text{ac}}^+(X, a_{t:t+h}^\circ) > 0$ and $\hat{Q}_{\text{ac}}^+(\tilde{X}, a_{t:t+h}^\Delta) - \hat{Q}_{\text{ac}}^+(\tilde{X}, a_{t:t+h}^\circ) > 0$.

This concludes that $\pi_{\text{ac}}^+(X) = \pi_{\text{ac}}^+(\tilde{X}) = a_{t:t+h}^\Delta = (2, 2, \dots, 2, 0)$, and thus

$$\hat{V}_{\text{ac}}^+(\tilde{X}) = \frac{1}{2} \left[\frac{1 - \gamma + (1 + c_3)(\gamma - \gamma^h)}{(1 - \gamma)(1 - \gamma^h)} \right], \quad (\text{D.236})$$

and

$$\begin{aligned}\hat{V}_{\text{ac}}^+(X) &= \frac{1}{2} \left[(1 - c_2) + (1 + c_3) \frac{\gamma - \gamma^h}{1 - \gamma} \right] + \frac{\gamma^h}{1 - \gamma^h} \hat{V}_{\text{ac}}^+(\tilde{X}) \\ &= \frac{1}{2(1 - \gamma)} - \frac{\vartheta_h^G}{1 - \gamma^h} + \frac{\tilde{\Delta}}{2}.\end{aligned}\tag{D.237}$$

Repeating the same procedure as the previous case, we obtain

$$V^*(X) - Q^*(X, a = 2) = \frac{\vartheta_h^L + \vartheta_h^G}{1 - \gamma^h} - \tilde{\Delta},\tag{D.238}$$

resulting in an optimality of

$$V^*(X) - V^\bullet(X) = \frac{\vartheta_h^L + \vartheta_h^G}{(1 - \gamma)(1 - \gamma^h)} - \frac{\tilde{\Delta}}{1 - \gamma}.\tag{D.239}$$

3. Sub-optimality of V_{ac}^+ :

Finally, we can use a pretty crude upper bound on the actual value of the action chunking policy π_{ac}^+ (reparameterizing $\tilde{\delta}_2 = 1 - (1 - \delta_2)^h$):

$$V_{\text{ac}}^+(X) \leq (1 - \tilde{\delta}_2) \left[(1 - c_2)/2 + \frac{\delta(\gamma - \gamma^h)}{2(1 - \gamma)} + \gamma^h V_{\text{ac}}^+(X) \right] + \frac{\tilde{\delta}_2}{1 - \gamma}\tag{D.240}$$

$$\leq \frac{1 - \tilde{\delta}_2}{2(1 - \gamma^h)(1 - \gamma)} [1 - \gamma + \delta(\gamma - \gamma^h)] + \frac{\tilde{\delta}_2}{1 - \gamma}.\tag{D.241}$$

Set $\delta = 1/2$, we have

$$V_{\text{ac}}^+(X) \leq \frac{1 - \tilde{\delta}_2}{2(1 - \gamma^h)(1 - \gamma)} [1 - \gamma/2 - \gamma^h/2] + \frac{\tilde{\delta}_2}{1 - \gamma}.\tag{D.242}$$

We set

$$\delta_2 = 1 - \left[1 - \frac{\gamma - \gamma^h - 4c(1 - \gamma^h)}{2 - 3\gamma^h + \gamma} \right]^{1/h},\tag{D.243}$$

which results in

$$\tilde{\delta}_2 = \frac{\gamma - \gamma^h - 4c(1 - \gamma^h)}{2 - 3\gamma^h + \gamma}.\tag{D.244}$$

It is clear that $0 < \delta_2 < 1$ because $c < \frac{\gamma - \gamma^h}{4(1 - \gamma^h)}$ and $\frac{\gamma - \gamma^h}{2 - 3\gamma^h + \gamma} < 1$.

Substituting $\tilde{\delta}_2$ in the bound of $V_{\text{ac}}^+(X)$ above, we obtain

$$V^*(X) - V_{\text{ac}}^+(X) \geq \frac{c}{1 - \gamma}. \quad (\text{D.245})$$

□

Proof of Theorem 6. Let

$$\Delta = \vartheta_h^L + \frac{\vartheta_h^G}{1 - \gamma^h} + \frac{\gamma^h \min(\vartheta_h^G, \vartheta_h^L)}{1 - \gamma^h}. \quad (\text{D.246})$$

Consider the 5-state, 3-action MDP constructed in Lemma 7 and Lemma 8 and a data distribution consisting of a mixture of three data distributions $\mathcal{D}^*$, $\mathcal{D}^\diamond$ (from Lemma 7) and $\mathcal{D}^\Delta$ (from Lemma 8):

$$P_{\mathcal{D}} = \alpha(1 - \varsigma)P_{\mathcal{D}^*} + \varsigma P_{\mathcal{D}^\diamond} + (1 - \alpha)P_{\mathcal{D}^\Delta}. \quad (\text{D.247})$$

We set α to be any value between 0 and 1 (non-inclusive) and set ς as any positive value such that

$$\varsigma < \frac{(\gamma - \gamma^h) - 2\vartheta_h^G(1 - \gamma) + 2\tilde{\Delta}(1 - \gamma)(1 - \gamma^h)}{(\gamma - \gamma^h) - 2\vartheta_h^G(1 - \gamma)}, \quad (\text{D.248})$$

where $\tilde{\Delta} = \sigma(1 - \gamma) < \min(\vartheta_h^L, \vartheta_h^G) < \min(\Delta, \frac{2\vartheta_h^G}{1 - \gamma^h})$ (satisfying the condition for $\tilde{\Delta}$ in Lemma 8).

The numerator and the denominator are both positive:

$$(\gamma - \gamma^h) - 2\vartheta_h^G(1 - \gamma) + 2\tilde{\Delta}(1 - \gamma)(1 - \gamma^h) > (\gamma - \gamma^h) - 2\vartheta_h^G(1 - \gamma) > 0, \quad (\text{D.249})$$

meaning such ς always exists.

Substituting the inequality to the result of Lemma 7 results in

$$\frac{1 - \gamma + \varsigma(\gamma - \gamma^h)}{2(1 - \gamma^h)(1 - \gamma)} - \frac{\varsigma\vartheta_h^G}{1 - \gamma^h} < \frac{1}{2(1 - \gamma)} - \frac{\vartheta_h^G}{1 - \gamma^h} + \tilde{\Delta}, \quad (\text{D.250})$$

which shows that π_{ac}^+ will always prefer $a_{t:t+h}^\Delta$ over action chunks in $\mathcal{D}^*$ and $\mathcal{D}^\diamond$.

This means that the value $\hat{V}_{\text{ac}}^+$ and the action chunking policy π_{ac}^+ we learn from $\mathcal{D}$ coincides with those of $\mathcal{D}^\Delta$, allowing us to directly use the results of Lemma 8.

Thus, we can conclude that

$$V^*(s_t) - V_{\text{ac}}^+(s_t) \geq \frac{c}{1 - \gamma}, \quad (\text{D.251})$$

and

$$V^*(X) - V^\bullet(X) = \frac{\Delta - \tilde{\Delta}}{1 - \gamma} = \frac{\vartheta_h^L}{1 - \gamma} + \frac{\vartheta_h^G}{(1 - \gamma)(1 - \gamma^h)} + \frac{\gamma^h \min(\vartheta_h^L, \vartheta_h^G)}{(1 - \gamma)(1 - \gamma^h)} - \sigma, \quad (\text{D.252})$$

as desired.

□

Appendix E

Additional Theoretical Results

E.1 ε -deterministic dynamics is weakly open-loop consistent

To provide some intuitions on what this open-loop consistency implies, we discuss a concrete family of MDPs where any data distribution from these MDPs is (weakly) ε_h -open-loop consistent (Proposition 4, with proof available in [Section E.2](#)).

Definition 5 (Near-deterministic Dynamics) A transition dynamics T is ε -deterministic if there exists a deterministic transition dynamics represented by function $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ and another arbitrary transition dynamics $\tilde{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathcal{S}}$, and T is a combination of f and $\tilde{T}$:

$$T(s' | s, a) = (1 - \varepsilon)\delta_{f(s,a)}(s') + \varepsilon\tilde{T}(s' | s, a), \forall s, s' \in \mathcal{S}, a \in \mathcal{A}. \quad (\text{E.1})$$

Proposition 4 (Deterministic Dynamics are Weakly Open-loop Consistent) If a transition dynamics $\mathcal{M}$ is ε -deterministic, then any data $\mathcal{D}$ collected from $\mathcal{M}$ is weakly ε_h -open-loop consistent with respect to $\mathcal{M}$ for any $h \in \mathbb{N}^+$ as long as $\varepsilon_h \geq 3(1 - (1 - \varepsilon)^{h-1})$.

An ε -deterministic dynamics acts like a deterministic one most of the time (with $1 - \varepsilon$ probability) and a non-deterministic one occasionally (with ε probability). This bounded stochasticity allows the results of taking an action sequence (of length h) open-loop to be deterministically determined in the event that the deterministic dynamics is ‘triggered’ (with a joint $(1 - \varepsilon)^{h-1}$ probability across h time steps). It is clear that under such event, there is no gap between the ‘replayed’ open-loop data $P_{\mathcal{D}}^o$ and the original data distribution $P_{\mathcal{D}}$, and as result there is also no value estimation bias under this event, and thus intuitively we

can bound the value estimation error by a function of the probability that the stochastic dynamics is ‘triggered’ (*i.e.*, with $1 - (1 - \varepsilon)^{h-1}$ probability).

E.2 Proof of Proposition 4

In this section, we provide the proof for Proposition 4 (restated below).

Proposition 4 (Deterministic Dynamics are Weakly Open-loop Consistent) If a transition dynamics $\mathcal{M}$ is ε -deterministic, then any data $\mathcal{D}$ collected from $\mathcal{M}$ is weakly ε_h -open-loop consistent with respect to $\mathcal{M}$ for any $h \in \mathbb{N}^+$ as long as $\varepsilon_h \geq 3(1 - (1 - \varepsilon)^{h-1})$.

Proof. Since T is ε -deterministic, it can be represented as $T(\cdot \mid s, a) = (1 - \varepsilon)\delta_{f(s, a)} + \varepsilon\tilde{T}(\cdot \mid s, a)$ for some $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ and $\tilde{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathcal{S}}$. Let $f(s, a_1, \dots, a_h) = f(\dots f(f(s, a_1), a_2) \dots a_h)$

Let $I \in \{0, 1\}$ a binary indicator variable that is 1 if and only if

$$s_{t+k+1} = f(s_{t+k}, a_{t+k}), \forall k \in \{0, 1, 2, \dots, h-1\} \quad (\text{E.2})$$

Intuitively $I = 1$ when the trajectory is generated deterministically until but not including the last state s_h in the trajectory chunk.

From the fact that T is ε -deterministic, we know that

$$P_{\mathcal{D}}(I_h = 1) \geq (1 - \varepsilon)^{h-1} \quad (\text{E.3})$$

We also have

$$P_{\mathcal{D}}(a_{t:t+h} \mid s_t) = P_{\mathcal{D}}(I_h = 1)P_{\mathcal{D}}(a_{t:t+h} \mid s_t, I_h = 1) + P_{\mathcal{D}}(I_h = 0)P_{\mathcal{D}}(a_{t:t+h} \mid s_t, I_h = 0) \quad (\text{E.4})$$

Then we have

$$D_{\text{TV}}(P_{\mathcal{D}}(a_{t:t+h} \mid s_t) \parallel P_{\mathcal{D}}(a_{t:t+h} \mid s_t, I_h = 1)) \leq (1 - (1 - \varepsilon)^{h-1}) \quad (\text{E.5})$$

If we transform each distribution of $a_{t:t+h}$ deterministically by $f(s_t, \cdot)$, by data processing inequality (DPI; Lemma 4), we have

$$D_{\text{TV}}(\mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot \mid s_t)} [\delta_{f(s_t, a_{t:t+h})}] \parallel \mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot \mid s_t, I_h = 1)} [\delta_{f(s_t, a_{t:t+h})}]) \leq (1 - (1 - \varepsilon)^{h-1}) \quad (\text{E.6})$$

Similarly, we have

$$D_{\text{TV}}(P_{\mathcal{D}}(a_{t:t+h+1} \mid s_t) \parallel P_{\mathcal{D}}(a_{t:t+h+1} \mid s_t, I_{h+1} = 1)) \leq (1 - (1 - \varepsilon)^h) \quad (\text{E.7})$$

which can be also deterministically transformed by taking $a_{t:t+h+1} \mapsto (f(s_t, \cdot), a_{t+h})$ (again with DPI, Lemma 4) to obtain

$$D_{\text{TV}}\left(\mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t)} \left[\pi_{\mathcal{D}}^{\circ}(a_{t+h} | s_t, a_{t:t+h}) \mathbb{I}_{f(s_t, a_{t:t+h})} \right] \parallel \mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t, I_{h+1}=1)} \left[\pi_{\mathcal{D}}^{\circ}(a_{t+h} | s_t, a_{t:t+h}, I_{h+1}=1) \mathbb{I}_{f(s_t, a_{t:t+h})} \right] \right) \leq (1 - (1 - \varepsilon)^h) \quad (\text{E.8})$$

Now, if we analyze the distribution of s_{t+h} subject to the open-loop execution of the action sequence from $P_{\mathcal{D}}(\cdot | s_t)$ and break it up into the deterministic and the non-deterministic case, we get

$$\mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t)} [T_{a_{t:t+h}}(\cdot | s_t)] = P_T(I=1) \mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t)} [\delta_{f(s_t, a_{t:t+h})}] + P_T(I=0) \mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t)} [T_{a_{t:t+h}}(\cdot | s_t, I_h=0)] \quad (\text{E.9})$$

Note that $P_T(I=1)$ denotes the probability that an open-loop executed trajectory using $a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t)$ is deterministic. This is different from $P_{\mathcal{D}}(I_h=1)$ because the latter is based on $P_{\mathcal{D}}(s_{t:t+h+1}, a_{t:t+h})$ whereas $P_T(I_h=1)$ is based on the open-loop trajectory distribution: $P_{\mathcal{D}}(\cdot | s_t) \prod_{k=0}^{h-1} T(s_{t+k} | s_t, a_{t:t+k})$. They both admit the same lower bound of $2(1 - (1 - \varepsilon)^{h-1})$.

Therefore,

$$D_{\text{TV}}(\mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t)} [T_{a_{t:t+h}}(\cdot | s_t)] \parallel \mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t)} [\delta_{f(s_t, a_{t:t+h})}]) \leq (1 - (1 - \varepsilon)^{h-1}) \quad (\text{E.10})$$

Similarly for the state-action case, we can multiply both side by the same conditional distribution $\pi_{\mathcal{D}}^{\circ}(a_{t+h} | s_t, a_{t:t+h})$ which preserves the TV bound. For the left-hand side, we have

$$P_{\mathcal{D}}^{\circ}(s_{t+h}, a_{t+h} | s_t) = \mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t)} [\pi_{\mathcal{D}}^{\circ}(a_{t+h} | s_t, a_{t:t+h}) T_{a_{t:t+h}}(s_{t+h} | s_t)] \quad (\text{E.11})$$

Therefore, we get

$$D_{\text{TV}}(P_{\mathcal{D}}^{\circ}(s_{t+h}, a_{t+h} | s_t) \parallel \mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t)} [\pi_{\mathcal{D}}^{\circ}(a_{t+h} | s_t, a_{t:t+h}) \mathbb{I}_{f(s_t, a_{t:t+h})}]) \leq (1 - (1 - \varepsilon)^{h-1}) \quad (\text{E.12})$$

We also have

$$P_{\mathcal{D}}(s_{t+h} | s_t) = (1 - \varepsilon)^{h-1} P_{\mathcal{D}}(s_{t+h} | s_t, I=1) + (1 - (1 - \varepsilon)^{h-1}) P_{\mathcal{D}}(s_{t+h} | s_t, I_h=0) \quad (\text{E.13})$$

Similarly, we have

$$\begin{aligned} D_{\text{TV}}(P_{\mathcal{D}}(s_{t+h} | s_t) \parallel P_{\mathcal{D}}(s_{t+h} | s_t, I_h=1)) \\ = D_{\text{TV}}(P_{\mathcal{D}}(s_{t+h} | s_t) \parallel \mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot | s_t, I_h=1)} [\delta_{f(s_t, a_{t:t+h})}]) \leq (1 - (1 - \varepsilon)^{h-1}) \end{aligned} \quad (\text{E.14})$$

For state-action, we can also get

$$P_{\mathcal{D}}(s_{t+h}, a_{t+h} \mid s_t) = (1 - \varepsilon)^h P_{\mathcal{D}}(s_{t+h}, a_{t+h} \mid s_t, I_{h+1} = 1) + (1 - (1 - \varepsilon)^h) P_{\mathcal{D}}(s_{t+h}, a_{t+h} \mid s_t, I_{h+1} = 0) \quad (\text{E.15})$$

which can be turned into the TV distance bound:

$$\begin{aligned} & D_{\text{TV}}(P_{\mathcal{D}}(s_{t+h}, a_{t+h} \mid s_t) \parallel P_{\mathcal{D}}(s_{t+h}, a_{t+h} \mid s_t, I_{h+1} = 1)) \\ &= D_{\text{TV}}\left(P_{\mathcal{D}}(s_{t+h}, a_{t+h} \mid s_t) \parallel \right. \\ & \quad \left. \mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot \mid s_t, I_{h+1}=1)} [\pi_{\mathcal{D}}^{\circ}(a_{t+h} \mid s_t, a_{t:t+h}, I_{h+1} = 1) \mathbb{I}_{f(s_t, a_{t:t+h})}] \right) \\ & \leq (1 - (1 - \varepsilon)^h) \end{aligned} \quad (\text{E.16})$$

Connecting all three total variation inequality (Equations (E.6), (E.10) and (E.14)) together, we get

$$D_{\text{TV}}(P_{\mathcal{D}}(s_{t+h} \mid s_t) \parallel \mathbb{E}_{a_{t:t+h} \sim P_{\mathcal{D}}(\cdot \mid s_t)} [T_{a_{t:t+h}}(\cdot \mid s_t)]) \leq 3(1 - (1 - \varepsilon)^{h-1}) \leq \varepsilon_h \quad (\text{E.17})$$

Connecting all three total variation inequalities for state-action (Equations (E.8), (E.11) and (E.16)) together, we get

$$\begin{aligned} D_{\text{TV}}(P_{\mathcal{D}}^{\circ}(s_{t+h-1}, a_{t+h-1} \mid s_t) \parallel P_{\mathcal{D}}(s_{t+h}, a_{t+h} \mid s_t)) & \leq 3 - 2(1 - \varepsilon)^{h-1} - (1 - \varepsilon)^{h-2} \\ & \leq 3(1 - (1 - \varepsilon)^{h-1}) \\ & \leq \varepsilon_h \end{aligned} \quad (\text{E.18})$$

Therefore, $\mathcal{D}$ is ε_h -open-loop consistent as desired. $\square$

E.3 Conditions when n -step return policies are provably sub-optimal

In this section, we provide the conditions under which the n -step return policies are provably sub-optimal.

Definition 6 (Near Optimal Data) We say $\mathcal{D}$ is $\tilde{\delta}_n$ -optimal for backup horizon length $n \in \mathbb{N}^+$ if

$$Q^*(s_t, a_t) - \mathbb{E}_{P_{\mathcal{D}}(\cdot \mid s_t, a_t)} [R_{t:t+n} + \gamma^n V^*(s_{t+n})] \leq \tilde{\delta}_n, \forall s_t, a_t \in \text{supp}(P_{\mathcal{D}}(s_t, a_t)). \quad (\text{E.19})$$

In Proposition 2, we have shown that the value of the learned action chunking policy is better than the nominal value of n -step return policy with a value gap of $3\varepsilon_h H$. However, the actual value of the n -step return policy maybe better. Here, we analyze the worst-case performance of n -step return policies.

Proposition 5 (Worst-case analysis of n -step return backup) For any $n \in \mathbb{N}^+$, $\tilde{\delta}_n \in (0, \gamma - \gamma^n)$ and $\sigma \in \left(0, \tilde{\delta}_n/(1 - \gamma)\right)$, there exists an MDP $\mathcal{M}$, and a $\tilde{\delta}_n$ -optimal data distribution $\mathcal{D}$ with $\text{supp}(P_{\mathcal{D}}(s_t, a_t)) \supseteq \text{supp}(P_{\mathcal{D}^*}(s_t, a_t))$ such that for some $s \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V_{\text{ac}}^+(s) - V_n^+(s) = \frac{\tilde{\delta}_n}{1 - \gamma} - \sigma, \quad (\text{E.20})$$

and for all $s \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s) = V_{\text{ac}}^+(s). \quad (\text{E.21})$$

The proof (available in Section E.4) provides concrete examples where n -step return policies are worse than action chunking policies. The implication of this result is that the suboptimality of the data distribution (as characterized by δ_n and $\tilde{\delta}_n$) is generally independent from the open-loop consistency (as characterized by ε_h).

E.4 Proof of Proposition 5

In this section, we provide the proof for Proposition 5 (restated below).

Proposition 5 (Worst-case analysis of n -step return backup) For any $n \in \mathbb{N}^+$, $\tilde{\delta}_n \in (0, \gamma - \gamma^n)$ and $\sigma \in \left(0, \tilde{\delta}_n/(1 - \gamma)\right)$, there exists an MDP $\mathcal{M}$, and a $\tilde{\delta}_n$ -optimal data distribution $\mathcal{D}$ with $\text{supp}(P_{\mathcal{D}}(s_t, a_t)) \supseteq \text{supp}(P_{\mathcal{D}^*}(s_t, a_t))$ such that for some $s \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V_{\text{ac}}^+(s) - V_n^+(s) = \frac{\tilde{\delta}_n}{1 - \gamma} - \sigma, \quad (\text{E.20})$$

and for all $s \in \text{supp}(P_{\mathcal{D}^*}(s_t))$,

$$V^*(s) = V_{\text{ac}}^+(s). \quad (\text{E.21})$$

Proof. Consider an MDP in Figure E.1. Let $\mathcal{D}$ be the data collected by the following policy:

$$\pi(a = 0 \mid X) = \pi(a = 1 \mid X) = 1/2, \quad (\text{E.22})$$

$$\pi(a = 0 \mid Y) = \alpha, \quad (\text{E.23})$$

$$\pi(a = 1 \mid Y) = 1 - \alpha. \quad (\text{E.24})$$

It is clear that the optimal policy is $\pi^*(X) = 1, \pi^*(Y) = 0$.

Since the dynamics are deterministic, the data distribution $\mathcal{D}$ is strongly open-loop consistent and thus by Theorem 3, $V_{\text{ac}}(s_t) = V^*(s_t)$ for all $s_t \in \text{supp}(P_{\mathcal{D}^*}(s_t))$.

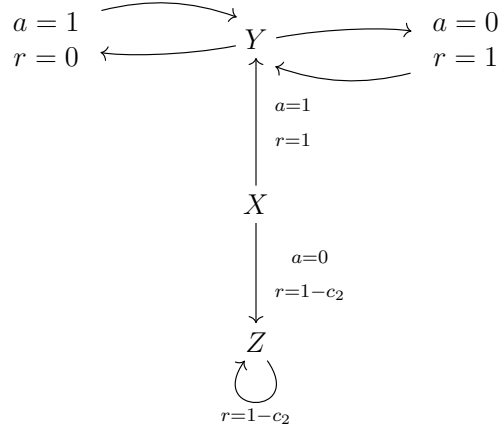


Figure E.1: **An MDP where the learned action chunking policies are optimal and the learned n -step return policies can be arbitrarily sub-optimal.**

To make sure the data distribution satisfies the $\tilde{\delta}_n$ -optimal condition, we set

$$\alpha = 1 - \frac{\tilde{\delta}_n}{\gamma - \gamma^n}, \quad c_2 = \tilde{\delta}_n - (1 - \gamma)\sigma. \quad (\text{E.25})$$

It is clear that at state Y , $\pi_n^+(Y) = 0$. We can then calculate the optimality gap for $\hat{Q}_n^+$ exactly as follows:

$$V^*(X) - \hat{Q}_n^+(X, a=0) = \frac{c_2}{1 - \gamma} = \frac{\tilde{\delta}_n}{1 - \gamma} - \sigma, \quad (\text{E.26})$$

$$V^*(X) - \hat{Q}_n^+(X, a=1) = \frac{1}{1 - \gamma} - \frac{1 - \gamma + \alpha(\gamma - \gamma^n)}{(1 - \gamma)(1 - \gamma^n)} = \frac{\tilde{\delta}_n}{1 - \gamma}. \quad (\text{E.27})$$

Since $\sigma > 0$, $\pi_n^+(X) = 0$. Now, we can compute $V_n(X)$ as follows:

$$V^*(X) - V_n(X) = \frac{c_2}{1 - \gamma} = \frac{\tilde{\delta}_n}{1 - \gamma} - \sigma, \quad (\text{E.28})$$

as desired.

□

E.5 Closed-loop execution without stochastic shortcuts

In this section, we provide an alternative way of bounding the sub-optimality of $\pi^\bullet$, the closed-loop execution of the learned action chunking policy π_{ac}^+ . In particular, we characterize two conditions when closed-loop execution of an action chunking policy can help mitigate open-loop biases.

Our first condition is based on the key observation that only a certain type of value overestimation is harmful for closed-loop execution of the action chunking policy. The source of this type of value overestimation comes from *stochastic shortcuts*:

Definition 7 (Stochastic Shortcuts) We say $\mathcal{M}$ is free of ϑ_h -stochastic shortcuts for a horizon h if

$$\begin{aligned} V^*(s_{t+h}) + R_{t:t+h} - V^*(s_t) &\leq \vartheta_h, \\ \forall s_{t:t+h+1}, a_{t:t+h} : \prod_{k=0}^{h-1} P(s_{t+k+1} \mid s_{t+k}, a_{t+k}) &> 0, \end{aligned} \tag{E.29}$$

where V^* is the value function of optimal policy in $\mathcal{M}$.

Intuitively, stochastic shortcuts are low-probability (but plausible) paths (*i.e.*, $s_t, a_t, \dots, s_{t+h}$) in the MDP that lead to returns that are much higher than the optimal expected value (*i.e.*, V^*). These stochastic shortcuts are particularly problematic for action chunking value backup because the chunked critic/Q-function cannot distinguish between a low-probability stochastic shortcut and an optimal (or near-optimal) closed-loop trajectory, leading it to erroneously favor the shortcut.

Our second condition is that our data distribution is a mixture of some data distribution that is collected by some optimal closed-loop policy ($\mathcal{D}^\star$) and some data distribution that is collected by an open-loop policy ($\mathcal{D}^\circ$, and thus is open-loop consistent). Intuitively, this condition makes sure that any non-optimal trajectory can be accurately estimated by the action chunking value function $\hat{V}_{ac}^+$ and the bounded mixing ratio restricts the amount of bias that the $\hat{V}_{ac}^+$ has on the estimation of the optimal trajectories when the open-loop action chunks (*e.g.*, in $\mathcal{D}^\circ$) coincide with the action chunks in the optimal data (*e.g.*, in $\mathcal{D}^\star$). We formally define the second condition as follows:

Definition 8 (Open-loop Data Mix) We say $\mathcal{D}$ is α -open-loop mixed if for some $\beta \in [0, 1)$, $\mathcal{D}$ can be decomposed into two data distributions $\mathcal{D}^\star, \mathcal{D}^\circ$ as

$$P_{\mathcal{D}}(\cdot | s_t) = \beta P_{\mathcal{D}^\star}(\cdot | s_t) + (1 - \beta) P_{\mathcal{D}^\circ}(\cdot | s_t), \quad (\text{E.30})$$

where $\mathcal{D}^\star$ is any data distribution collected by an optimal closed-loop policy $\pi^\star$ and $\mathcal{D}^\circ$ is any strongly open-loop consistent data distribution, and

$$P_{\mathcal{D}^\circ} [a_{t:t+h} \in \text{supp}(P_{\mathcal{D}^\star}(a_{t:t+h} | s_t)) | s_t] \leq \frac{\alpha\beta}{(1-\alpha)(1-\beta)}, \quad \forall s_t \quad (\text{E.31})$$

With such data mixing assumption and in the absence of stochastic shortcuts, we can show that closed-loop execution of the action chunking policy (*i.e.*, only executing the first action of the action chunk) recovers a near-optimal closed-loop policy:

Theorem 7 (Closed-loop Execution in the Absence of Stochastic Shortcuts) If $\mathcal{D}$ is α -open-loop mixed and $\mathcal{M}$ is free of ϑ_h -stochastic shortcut, the value ($V^\bullet$) of the one-step policy ($\pi^\bullet$) as a result of the closed-loop execution of the action chunking policy π_{ac}^+ learned from $\mathcal{D}$ admits the following bound for all $s_t \in \text{supp}(P_{\mathcal{D}^\star}(s_t))$:

$$V^\star(s_t) - V^\bullet(s_t) \leq \frac{\alpha}{(1-\gamma)^2(1-\gamma^h(1-\alpha))} + \frac{\vartheta_h \gamma^h}{(1-\gamma)(1-\gamma^h)}. \quad (\text{E.32})$$

A proof is available in Section E.6. Intuitively, the second condition measures how much the support of the open-loop data overlaps with the support of the optimal data. With some algebraic manipulation, assuming the worst case of Equation (E.31), we can rewrite the data mixture as

$$\mathcal{D} = \hat{\beta} [(1-\alpha)\mathcal{D}^\star + \alpha\mathcal{D}_{\text{in}}^\circ] + (1-\hat{\beta})\mathcal{D}_{\text{out}}^\circ, \quad (\text{E.33})$$

where $\hat{\beta} = \frac{\beta}{1-\alpha}$, $\text{supp}(P_{\mathcal{D}_{\text{in}}^\circ}(\cdot | s_t)) \subseteq \text{supp}(P_{\mathcal{D}^\star}(\cdot | s_t))$ and $\text{supp}(P_{\mathcal{D}_{\text{out}}^\circ}(\cdot | s_t)) \cap \text{supp}(P_{\mathcal{D}^\star}(\cdot | s_t)) = \emptyset$. As the bound is independent of $\hat{\beta}$, it becomes clear that $\mathcal{D}_{\text{out}}^\circ$ makes no contribution to the optimality of action chunking policy learning. The only harmful portion of the open-loop data distribution is $\mathcal{D}_{\text{in}}^\circ$, as the action chunking Q-function cannot differentiate these open-loop actions in $\mathcal{D}_{\text{in}}^\circ$ from the closed-loop optimal actions in $\mathcal{D}^\star$. This is reflected as the first term in our bound. The implication is that even if the data $\mathcal{D}$ is arbitrarily sub-optimal (with $\hat{\beta} \rightarrow 0$, and hence arbitrarily bad for n -step return policies), $\pi^\bullet$ remains near-optimal as long as the ‘in-distribution’ open-loop data $\mathcal{D}_{\text{in}}^\circ$ is relatively low in density compared to the optimal closed-loop data $\mathcal{D}^\star$ (*i.e.*, α is small).

Furthermore, our bound is independent of the open-loop consistency of the data $\mathcal{D}$. As $\alpha, \vartheta \rightarrow 0$, closed-loop execution of the action chunking policy exactly recovers the optimal

policy. In contrast, even when $\alpha, \vartheta \rightarrow 0$, open-loop execution of the original action chunking policy (*i.e.*, π_{ac}^+) can suffer from the open-loop inconsistency of the data $\mathcal{D}$: its value error can only be bounded by $\frac{\varepsilon_h}{(1-\gamma)(1-\gamma^h)}$ (as shown in Theorem 1), a function of ε_h (the strong open-loop consistency of $\mathcal{D}$).

E.6 Proof of Theorem 7

In this section, we provide the proof for Theorem 7 (restated below).

Theorem 7 (Closed-loop Execution in the Absence of Stochastic Shortcuts) If $\mathcal{D}$ is α -open-loop mixed and $\mathcal{M}$ is free of ϑ_h -stochastic shortcut, the value ($V^\bullet$) of the one-step policy ($\pi^\bullet$) as a result of the closed-loop execution of the action chunking policy π_{ac}^+ learned from $\mathcal{D}$ admits the following bound for all $s_t \in \text{supp}(P_{\mathcal{D}^\star}(s_t))$:

$$V^\star(s_t) - V^\bullet(s_t) \leq \frac{\alpha}{(1-\gamma)^2(1-\gamma^h(1-\alpha))} + \frac{\vartheta_h \gamma^h}{(1-\gamma)(1-\gamma^h)}. \quad (\text{E.32})$$

Before we start proving the main theorem, we first prove the following Lemma, which shows that the overestimation of $\hat{V}_{ac}^+$ is bounded when the advantage of the stochastic shortcut (as defined in Definition 7) is bounded.

Lemma 9 (Lack of stochastic shortcut bounds overestimation) If $\mathcal{M}$ is free of ϑ_h -advantageous stochastic shortcuts for a horizon h , then

$$V_{ac}^+(s_t) - V^\star(s_t) \leq \frac{\vartheta_h}{1-\gamma^h}, \quad (\text{E.34})$$

where V_{ac}^+ is the value function of the action chunking policy learned from any data distribution $\mathcal{D}$ from $\mathcal{M}$ and $V^\star$ is the optimal value function in $\mathcal{M}$.

Proof.

$$\begin{aligned} \hat{V}_{ac}^+(s_t) - V^\star(s_t) &= \mathbb{E}_{P_{\mathcal{D}}} \left[R_{t:t+h} + \gamma^h \hat{V}_{ac}^+(s_{t+h}) \right] - V^\star \\ &\leq \mathbb{E}_{P_{\mathcal{D}}} \left[\gamma^h (\hat{V}_{ac}^+(s_{t+h}) - V^\star(s_{t+h})) \right] + \vartheta_h \\ &\leq \frac{\vartheta_h}{1-\gamma^h}. \end{aligned} \quad (\text{E.35})$$

□

Lemma 10 (Monotonicity of optimality) Let $\mathcal{D}^\circ$ be any data distribution that is collected by an open-loop policy. Then, for all $s_t, a_{t:t+h}$,

$$V^\star(s_t) \geq \mathbb{E}_{s_{t+1} \sim T(\cdot | s_t, a_t)} [r_t + \gamma V^\star(s_{t+1})] \geq \mathbb{E}_{P_{\mathcal{D}^\circ}(\cdot | s_t, a_{t:t+h})} [R_{t:t+h} + \gamma^h V^\star(s_{t+h})] \quad (\text{E.36})$$

Proof. The first inequality is clear from the definition of $Q^\star(s_t, a_t)$ and $V^\star(s_t)$:

$$Q^\star(s_t, a_t) := \mathbb{E}_{s_{t+1} \sim T(\cdot | s_t, a_t)} [r_t + \gamma V^\star(s_{t+1})] \quad (\text{E.37})$$

$$V^\star(s_t) := \max_{a_t^\star} Q^\star(s_t, a_t^\star) \geq Q^\star(s_t, a_t) \quad (\text{E.38})$$

For the second inequality, we observe that

$$\begin{aligned} & \mathbb{E}_{P_{\mathcal{D}^\circ}(\cdot | s_t, a_{t:t+h-1})} [R_{t:t+h-1} + \gamma^{h-1} V^\star(s_{t+h-1})] \\ &= \mathbb{E}_{P_{\mathcal{D}^\circ}(\cdot | s_t, a_{t:t+h-1})} \left[R_{t:t+h-1} + \gamma^{h-1} \max_{a_{t+h-1}^\star} Q^\star(s_{t+h-1}, a_{t+h-1}^\star) \right] \\ &\geq \mathbb{E}_{P_{\mathcal{D}^\circ}(\cdot | s_t, a_{t:t+h-1})} [R_{t:t+h-1} + \gamma^{h-1} Q^\star(s_{t+h-1}, a_{t+h-1})] \\ &= \mathbb{E}_{P_{\mathcal{D}^\circ}(\cdot | s_t, a_{t:t+h})} [R_{t:t+h-1} + \gamma^{h-1} [r(s_{t+h-1}, a_{t+h-1}) + \gamma V^\star(s_{t+h})]] \\ &= \mathbb{E}_{P_{\mathcal{D}^\circ}(\cdot | s_t, a_{t:t+h})} [R_{t:t+h} + \gamma^h V^\star(s_{t+h})] \end{aligned} \quad (\text{E.39})$$

By induction,

$$\mathbb{E}_{T(\cdot | s_t, a_t)} [r_t + \gamma V^\star(s_{t+1})] \geq \mathbb{E}_{P_{\mathcal{D}^\circ}(\cdot | s_t, a_{t:t+h})} [R_{t:t+h} + \gamma^h V^\star(s_{t+h})], \quad (\text{E.40})$$

as desired. $\square$

With these two lemmata, we are now ready to present our main proof.

Proof of Theorem 7. The key idea of our proof is to analyze the action chunk taken by the policy π_{ac}^+ (i.e., $\pi_{\text{ac}}^+ : s_t \mapsto \arg \max_a \hat{Q}(s_t, a_{t:t+h})$). Due to our construction of $\mathcal{D}$, the action chunk learned by π_{ac}^+ either comes from $\mathcal{D}^\star$ or $\mathcal{D}^\circ$. We can express $\mathcal{D}^\circ$ as the aggregation of two open-loop data distributions, $\mathcal{D}_{\text{in}}^\circ$ and $\mathcal{D}_{\text{out}}^\circ$:

$$P_{\mathcal{D}^\circ}(\cdot | s_t) := \hat{\alpha} P_{\mathcal{D}_{\text{in}}^\circ}(\cdot | s_t) + (1 - \hat{\alpha}) P_{\mathcal{D}_{\text{out}}^\circ}(\cdot | s_t) \quad (\text{E.41})$$

where $\text{supp}(P_{\mathcal{D}_{\text{out}}^\circ}(a_{t:t+h} | s_t)) \cap \text{supp}(P_{\mathcal{D}^\star}(a_{t:t+h} | s_t)) = \emptyset$ and

$$\hat{\alpha} \leq \frac{\alpha\beta}{(1-\alpha)(1-\beta)}. \quad (\text{E.42})$$

Case 1, from $\mathcal{D}^\star$: If $\pi_{\text{ac}}^+(\cdot | s_t) \cap \text{supp}(P_{\mathcal{D}^\circ}(a_{t:t+h} | s_t)) = \emptyset$, then we know that

$$a_{t:t+h}^\diamond \in \text{supp}(P_{\mathcal{D}^\star}(a_{t:t+h} | s_t)), \quad (\text{E.43})$$

for any $a_{t:t+h}^\diamond \in \pi_{\text{ac}}^+$. Thus, the closed-loop execution policy $\pi^\bullet$ takes the optimal action at s_t . This leads to the following equalities:

$$\begin{aligned} V^*(s_t) - V^\bullet(s_t) &= V^*(s_t) - Q^\bullet(s_t, a_t^\diamond) \\ &= \gamma \mathbb{E}_{T(\cdot | s_t, a_t^\diamond)} [(V^*(s_{t+1}) - V^\bullet(s_{t+1}))] \end{aligned} \quad (\text{E.44})$$

$$\begin{aligned} \hat{V}_{\text{ac}}^+(s_t) - V^*(s_t) &= \mathbb{E}_{P_{\mathcal{D}^*}} [\hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}^\diamond)] - V^*(s_t) \\ &= \gamma^h \mathbb{E}_{P_{\mathcal{D}^*}} [\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h})] \end{aligned} \quad (\text{E.45})$$

Case 2, from $\mathcal{D}^\circ$: If $\pi_{\text{ac}}^+(\cdot | s_t) \cap \text{supp}(P_{\mathcal{D}^\circ}(a_{t:t+h} | s_t)) \neq \emptyset$, then there exists $a_{t:t+h}^\diamond \in \pi_{\text{ac}}^+(\cdot | s_t)$ such that

$$a_{t:t+h}^\diamond \in \text{supp}(P_{\mathcal{D}^\circ}(a_{t:t+h} | s_t)). \quad (\text{E.46})$$

For any $a_{t:t+h}^\diamond \in \pi_{\text{ac}}^+(\cdot | s_t)$ such that $a_{t:t+h}^\diamond \notin \text{supp}(P_{\mathcal{D}^\circ}(a_{t:t+h} | s_t))$, we know from the previous case that $V^*(s_t) - Q^\bullet(s_t, a_t^\diamond) = \gamma \mathbb{E}_{T(\cdot | s_t, a_t^\diamond)} [(V^*(s_{t+1}) - V^\bullet(s_{t+1}))]$. We are left with analyzing $V^* - Q^\bullet(s_t, a_t^\diamond)$ for the second case.

Since $P_{\mathcal{D}}(\cdot | s_t) = \beta P_{\mathcal{D}^*}(\cdot | s_t) + (1 - \beta)\hat{\alpha} P_{\mathcal{D}_{\text{in}}^\circ}(\cdot | s_t) + (1 - \beta)(1 - \hat{\alpha}) P_{\mathcal{D}_{\text{out}}^\circ}(\cdot | s_t)$ with $\text{supp}(P_{\mathcal{D}_{\text{out}}^\circ}(a_{t:t+h} | s_t)) \cap \text{supp}(P_{\mathcal{D}^*}(a_{t:t+h} | s_t)) = \emptyset$, we can isolate the contribution of $\mathcal{D}_{\text{out}}^\circ$ in $\mathcal{D}^\circ$ by an event I that is 1 when $a_{t:t+h} \in \text{supp}(P_{\mathcal{D}^*}(a_{t:t+h} | s_t))$ and 0 otherwise. Now, we can remove $\mathcal{D}_{\text{out}}^\circ$ when conditioned on $I = 1$ as follows:

$$\begin{aligned} P_{\mathcal{D}}(\cdot | s_t, I = 1) &= \frac{\beta}{\beta + (1 - \beta)\hat{\alpha}} P_{\mathcal{D}^*}(\cdot | s_t, I = 1) + \frac{(1 - \beta)\hat{\alpha}}{\beta + (1 - \beta)\hat{\alpha}} P_{\mathcal{D}^\circ}(\cdot | s_t, I = 1) \\ &= (1 - \bar{\alpha}) P_{\mathcal{D}^*}(\cdot | s_t, I = 1) + \bar{\alpha} P_{\mathcal{D}^\circ}(\cdot | s_t, I = 1) \end{aligned} \quad (\text{E.47})$$

Since $\hat{\alpha} \leq \frac{\alpha\beta}{(1-\alpha)(1-\beta)}$, with some algebraic manipulation, we can obtain $\bar{\alpha} \leq \alpha$.

By Lemma 1, there exists $a_{t:t+h}^\diamond \in \text{supp}(P_{\mathcal{D}^*}(a_{t:t+h} | s_t))$ such that

$$P_{\mathcal{D}}(\cdot | s_t, a_{t:t+h}^\diamond) = (1 - \tilde{\alpha}) P_{\mathcal{D}^*}(\cdot | s_t, a_{t:t+h}^\diamond) + \tilde{\alpha} P_{\mathcal{D}^\circ}(\cdot | s_t, a_{t:t+h}^\diamond) \quad (\text{E.48})$$

with $\tilde{\alpha} \leq \bar{\alpha} \leq \alpha$.

We are now ready to bound $\hat{V}_{\text{ac}}^+$ as follows:

$$\begin{aligned} \hat{V}_{\text{ac}}^+(s_t) - V^*(s_t) &\geq \hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}^\diamond) - V^*(s_t) \\ &= (1 - \tilde{\alpha}) \mathbb{E}_{P_{\mathcal{D}^*}} [\hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}^\diamond)] + \tilde{\alpha} \mathbb{E}_{P_{\mathcal{D}^\circ}} [\hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}^\diamond)] - V^*(s_t) \\ &\geq (1 - \alpha) \mathbb{E}_{P_{\mathcal{D}^*}} [\gamma^h (\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h}))] - \alpha V^*(s_t) \\ &\geq (1 - \alpha) \gamma^h \mathbb{E}_{P_{\mathcal{D}^*}} [\hat{V}_{\text{ac}}^+(s_{t+h}) - V^*(s_{t+h})] - \frac{\alpha}{1 - \gamma} \end{aligned} \quad (\text{E.49})$$

where we drop the term $\tilde{\alpha}\mathbb{E}_{\mathcal{D}^\circ} [\hat{Q}_{\text{ac}}^+(s_t, a_{t:t+h}^\circ)] \geq 0$ for the second inequality.

By combining Equation (E.49) (when $\pi_{\text{ac}}^+(\cdot | s_t) \cap \text{supp}(P_{\mathcal{D}^\circ}(\cdot | s_t)) \neq \emptyset$) with Equation (E.45) (when $\pi_{\text{ac}}^+(\cdot | s_t) \cap \text{supp}(P_{\mathcal{D}^\circ}(\cdot | s_t)) = \emptyset$), we can now recursively bound $\hat{V}_{\text{ac}}^+(s_t) - V^*(s_t)$ as follows:

$$\hat{V}_{\text{ac}}^+(s_t) - V^*(s_t) \geq -\frac{\alpha}{(1-\gamma)(1-\gamma^h(1-\alpha))}. \quad (\text{E.50})$$

Combining this with the result from Lemma 9, we have both a lower bound and an upper bound on $\hat{V}_{\text{ac}}^+$ in terms of V^* :

$$V^*(s_t) - C_\alpha \leq \hat{V}_{\text{ac}}^+(s_t) \leq V^*(s_t) + C_\vartheta. \quad (\text{E.51})$$

where $C_\alpha := \frac{\alpha}{(1-\gamma)(1-\gamma^h(1-\alpha))}$, $C_\vartheta := \frac{\vartheta_h}{1-\gamma^h}$.

Now, we are ready to bound $V^* - Q^\bullet(s_t, a_t^\circ)$ for the second case (when $\pi_{\text{ac}}^+(\cdot | s_t) \cap \text{supp}(P_{\mathcal{D}^\circ}(\cdot | s_t)) \neq \emptyset$). For notation convenience, for the following equation, we use $P_{\mathcal{D}^\circ}$ as an abbreviation for $P_{\mathcal{D}^\circ}(\cdot | s_t, a_{t:t+h}^\circ)$.

$$\begin{aligned} & V^*(s_t) - Q^\bullet(s_t, a_t^\circ) \\ &= V^* - \mathbb{E}_{P_{\mathcal{D}^\circ}} [r_t + \gamma V^\bullet(s_{t+1})] \\ &\leq V^* - \mathbb{E}_{P_{\mathcal{D}^\circ}} [R_{t:t+h} + \gamma^h V^*(s_{t+h}) - \gamma(V^*(s_{t+1}) - V^\bullet(s_{t+1}))] \\ &\leq \hat{V}_{\text{ac}}^+(s_t) + C_\alpha - \mathbb{E}_{P_{\mathcal{D}^\circ}} [R_{t:t+h} + \gamma^h(\hat{V}_{\text{ac}}^+(s_{t+h}) - C_\vartheta) - \gamma(V^*(s_{t+1}) - V^\bullet(s_{t+1}))] \\ &= \hat{V}_{\text{ac}}^+(s_t) - \mathbb{E}_{P_{\mathcal{D}^\circ}} [R_{t:t+h} + \gamma^h(\hat{V}_{\text{ac}}^+(s_{t+h}))] + C_\alpha + \gamma^h C_\vartheta + \mathbb{E}_{T(\cdot | s_t, a_t^\circ)} [\gamma(V^*(s_{t+1}) - V^\bullet(s_{t+1}))] \\ &= C_\alpha + \gamma^h C_\vartheta + \gamma \mathbb{E}_{T(\cdot | s_t, a_t^\circ)} [V^*(s_{t+1}) - V^\bullet(s_{t+1})], \end{aligned} \quad (\text{E.52})$$

where the first inequality uses Lemma 10,

$$\mathbb{E}_{T(\cdot | s_t, a_t^\circ)} [r_t + \gamma V^*(s_{t+1})] \geq \mathbb{E}_{P_{\mathcal{D}^\circ}(\cdot | s_t, a_{t:t+h}^\circ)} [R_{t:t+h} + \gamma^h V^*(s_{t+h})], \quad (\text{E.53})$$

and the second inequality uses the lowerbound for $\hat{V}_{\text{ac}}^+(s_t)$ and the upperbound for $\hat{V}_{\text{ac}}^+(s_{t+h})$.

Combining the bounds of $V^* - Q^\bullet(s_t, a_t^\circ)$ for both cases (Equation (E.44) and Equation (E.52)),

we have

$$\begin{aligned}
V^\star - V^\bullet(s_t) &\leq \max \left(\gamma \mathbb{E}_{T(\cdot|s_t, a_t^\circ)} [(V^\star(s_{t+1}) - V^\bullet(s_{t+1}))], \right. \\
&\quad \left. C_\alpha + \gamma^h C_\vartheta + \gamma \mathbb{E}_{T(\cdot|s_t, a_t^\circ)} [V^\star(s_{t+1}) - V^\bullet(s_{t+1})] \right) \\
&= C_\alpha + \gamma^h C_\vartheta + \gamma \mathbb{E}_{T(\cdot|s_t, a_t^\circ)} [V^\star(s_{t+1}) - V^\bullet(s_{t+1})] \\
&\leq \frac{C_\alpha + \gamma^h C_\vartheta}{1 - \gamma} \\
&= \frac{\alpha}{(1 - \gamma)^2(1 - \gamma^h(1 - \alpha))} + \frac{\vartheta \gamma^h}{(1 - \gamma)(1 - \gamma^h)}.
\end{aligned} \tag{E.54}$$

□