D4RL: Datasets for Deep Data-Driven Reinforcement Learning

Justin Fu
UC Berkeley
justinjfu@eecs.berkeley.edu

Aviral Kumar
UC Berkeley
aviralk@berkeley.edu

Ofir Nachum
Google Brain
ofirnachum@google.com

George Tucker
Google Brain
gjt@google.com

Sergey Levine
UC Berkeley, Google Brain
svlevine@eecs.berkeley.edu

Abstract

The offline reinforcement learning (RL) problem, also known as batch RL, refers to the setting where a policy must be learned from a static dataset, without additional online data collection. This setting is compelling as potentially it allows RL methods to take advantage of large, pre-collected datasets, much like how the rise of large datasets has fueled results in supervised learning in recent years. However, existing *online* RL benchmarks are not tailored towards the *offline* setting, making progress in offline RL difficult to measure. In this work, we introduce benchmarks specifically designed for the offline setting, guided by key properties of datasets relevant to real-world applications of offline RL. Examples of such properties include: datasets generated via hand-designed controllers and human demonstrators, multi-objective datasets where an agent can perform different tasks in the same environment, and datasets consisting of a mixtures of policies. To facilitate research, we release our benchmark tasks and datasets with a comprehensive evaluation of existing algorithms and an evaluation protocol together with an open-source codebase. We hope that our benchmark will focus research effort on methods that drive improvements not just on simulated tasks, but ultimately on the kinds of real-world problems where offline RL will have the largest impact.

1 Introduction

The last decade has seen impressive progress across a range of machine learning application domains, driven in large part by high-capacity deep neural network models together with large and diverse training datasets [8]. While reinforcement learning (RL) algorithms have also benefited from deep learning [25], the active data collection is typically required for these algorithms to succeed, limiting the extent to which large, static datasets can be leveraged. Offline RL [19], also referred to as *full batch* RL, where agents must learn from fixed datasets, provides a bridge between RL and data-driven supervised learning. The promise of offline RL is potentially enormous: much like how deep neural networks with large datasets can enable powerful pattern recognition, offline RL methods equipped with high-capacity

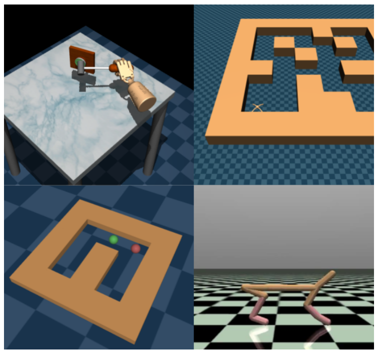


Figure 1: A selection of tasks contained within the benchmark.

Website with tasks and datasets is available at https://sites.google.com/view/d4rl/home.

Preprint. Under review.

models can train powerful decision making models entirely from static datasets. This could have profound implications for a range of application domains, such as robotics, autonomous driving, and healthcare.

The offline RL setting also addresses several major limitations of the standard online RL formulation. First, by leveraging offline data, RL algorithms can reduce sample complexity. While online methods might require millions of time steps of experience to learn one task, many settings, such as autonomous driving, natural language interfaces, and recommender systems *already* offer abundant sources of logged data. Although this data may not correspond to any specific task that a practitioner may be interested in solving, utilizing it in an offline RL framework may enable these tasks to be solved with minimal or no additional data collection. Second, the offline RL setting alleviates many of the safety concerns associated with online RL. In many domains, from robotics to medical diagnosis, the cost of failure is unacceptable. The offline RL setting allows policies to be pre-trained on large datasets, such that they may achieve an acceptable baseline level of performance the first time they are deployed.

Unfortunately, current offline RL methods have not yet lived up to the full promise of enabling reinforcement learning from large datasets. While recent work has investigated a number of possible technical reasons for this [37, 7, 18], a major challenge in addressing these issues has been the lack of realistic evaluation benchmarks. Most recent works in this area have used data collected from full or partial training runs of standard online RL methods. However, such datasets are not necessarily representative of the kinds of scenarios in which offline RL might be used in practice.

The key contribution of this work is the introduction of Datasets for Deep Data-Driven Reinforcement Learning (D4RL): a suite of tasks and datasets for benchmarking progress in offline RL. We focus our design around two principles: the tasks should be realistic but conducive to experimentation, and the set of tasks and datasets should exercise dimensions of the offline RL problem where current offline RL algorithms may struggle. These dimensions include: data from human demonstrations, passively collected logs of multiple different tasks distinct from the task being learned, and data from non-learned "scripted" controllers. We provide tasks with different types of data distributions, such as data from behavior policies that cannot be represented precisely by Markovian policies (e.g., demonstrations or stateful hand-designed controllers), and tasks with strict safety considerations such as autonomous driving. Finally, we benchmark several state-of-the-art algorithms [13, 18, 37, 2, 7, 28, 29], demonstrating that while these algorithms perform well in the settings they were designed for, they can perform poorly on tasks such as data collected from hand-designed controllers and multi-task behavior. We hope that our work provides insight into existing shortcomings in offline RL methods, and that our benchmark provides a meaningful metric for progress in this emerging area.

2 Related Work

Recently proposed evaluations for offline or batch RL algorithms have primarily been instantiated as learning from a fixed dataset of behaviors generated by a previously trained behavior policy. The quality of this agent may range from the random behavior of an initial policy to near-expert behavior from a fully trained policy. This evaluation protocol has been used in domains such as continuous control for robotics [7, 18, 37], navigation [20], industrial control [14], and Atari video games [2]. While this method may be adequate for demonstrating progress towards the more traditional, policy improvement-centric goal of offline RL, our focus in this work is primarily to use offline RL as a means to scale RL to large datasets. Thus, these benchmarks lack properties that might be seen in large, cheaply collected datasets, such as behavior from multiple tasks and human demonstrations, which can adversely affect algorithm performance. Gulcehre et al. [10] recently proposed a benchmark for offline reinforcement learning which focuses on problems with partial observability, memory, and exploration challenges. D4RL focuses on a wider range of dataset generation procedures, such as human demonstrations, exploratory agents, and hand-coded controllers.

Offline reinforcement learning using large datasets has also been used in real-world systems where evaluation using a simulator is not possible, such as in robotics [3] and dialogue systems [15, 30, 16]. Moreover, significant efforts have been made to incorporate large-scale datasets into off-policy RL [17, 26, 9], but these works generally use large numbers of robots to collect online interaction during training. We believe these to be promising directions for future research, but the primary goal of this work is to provide an effective platform for developing algorithms, and simulated environments to enable cheap and reliable evaluation and comparative benchmarking.

3 Background

The offline reinforcement learning problem statement is formalized within a Markov decision process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \rho_0, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $P(s'|s, a)$ denotes the transition distribution, $\rho_0(s)$ denotes the initial state distribution, $R(s, a)$ denotes the reward function, and $\gamma \in (0, 1)$ denotes the discount factor. The goal in RL is to find a policy $\pi(a|s)$ that maximizes the expected cumulative discounted rewards $J(\pi) = E_{\pi,P,\rho_0}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$, also known as the discounted returns.

In episodic RL, the algorithm is given access to the MDP via trajectory samples for arbitrary π of the algorithm's choosing. Off-policy methods may use experience replay [22] to store these trajectories in a *replay buffer* $\mathcal{D}$ of transitions (s_t, a_t, s_{t+1}, r_t), and use an off-policy algorithm such as Q-learning [36] to optimize π. However, these methods still iteratively collect additional data, and mitting this collection step can produce poor results. For example, running state-of-the-art off-policy RL algorithms on trajectories collected from an expert policy can result in diverging Q-values [18].

In *offline RL*, the algorithm no longer has access to a simulator, and is instead presented with a fixed *dataset* of transitions $\mathcal{D}$, akin to supervised learning. In the special case that the dataset is generated by sampling trajectories from a single policy, the sampling policy is referred to as a *behavior policy* π_B. While off-policy RL algorithms can in principle be used in the offline setting, issues such as distribution shift can cause undesirable performance in practice, as mentioned previously. Thus, approaches to offline RL have focused on techniques such as safe policy improvement [33] or adding regularizers in to mitigate effects of distribution shift [37]. In order to be effective, offline RL algorithms must be able to handle extensive distribution shift, as well as data collected via unconventional means, such as through human demonstration or hand-designed controllers, which may not be representable by the chosen policy class. A more comprehensive discussion on the problems affecting offline RL and what available techniques can be found in Levine et al. [21].

4 Task Design Factors

In order to design a benchmark that provides a meaningful measure of progress towards realistic applications of offline RL, we choose datasets and tasks to cover a range of practical properties. In practice, one may not have control over the type of data available, so we outline several properties to explore which we believe may be problematic for existing RL algorithms and representative of real-world datasets.

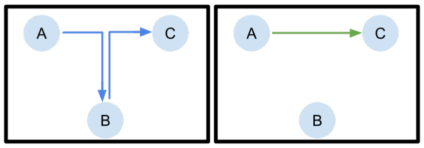


Figure 2: An example of stitching together subtrajectories to solve a task.

Undirected and multitask data naturally arises when data is passively logged, such as recording user interactions on the internet or recording videos of a car for autonomous driving. This data may not necessarily be directed towards the specific task one is trying to accomplish. However, parts of trajectories in such a dataset can still provide useful information for the policy we are trying to learn. For example, one may be able to combine sub-optimal sub-trajectories to form a shortest path to a goal. In the figure to the upper-right, if an agent is given trajectories from A-B and B-C in a dataset (left image), it can form a trajectory from A-C by combining the corresponding halves of the original trajectories. We refer to this property as *stitching*, since the agent can use portions of existing trajectories in order to solve a task, rather than relying on generalization outside of the dataset. Undirected data may also be collected by exploratory agents optimizing a different objective from evaluation or an objective such as intrinsic motivation.

Narrow data distributions, such as those from deterministic policies, are problematic for offline RL algorithms and may cause divergence both empirically [7, 18] and theoretically [27, 6, 18, 1, 5]. Narrow datasets may arise in human demonstrations, or when using hand-crafted policies. An important challenge in offline RL is to be able to gracefully handle diverse data distributions without algorithms diverging or producing performance worse than the provided behavior.

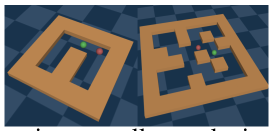
Data generated from a non-RL policy. Real-life behavior may not originate from learned policies, which can cause issues for RL algorithms. For example, human demonstrators may utilize external cues that are not observable to the policy. This may make the dataset generation process non-Markovian, making it impossible to represent with Markovian policies. Hand-designed controllers may not be representable by the learner's policy class, introducing bias into the learning process [23]. Additionally, these issues can problematic for methods which rely on importance sampling [31] as it may be difficult to estimate the probability of the observed actions.

Suboptimal data. For tasks with a clear objective, the datasets may not contain behaviors from optimal agents. This represents a challenge for approaches such as imitation learning, which generally requires expert demonstrations. In contrast, with offline RL, we can still potentially improve over suboptimal data. We note that this type of data is currently the predominant method for benchmarking offline RL algorithms within the field of deep RL [7, 18, 37].

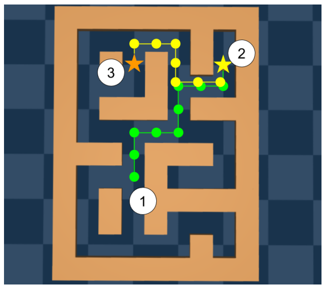
In addition to the specific distributional properties discussed above, there are several additional benchmark-wide design considerations. First, we strived to provide **realistic tasks** which were still amenable to efficient simulation. Second, we wished to include a variety of qualitatively different tasks, in order to provide diversity in the domains tested. Therefore, we include both locomotion, autonomous driving, and robotic manipulation tasks. In terms of representation, we include domains with state-based representations as well as those with challenging image-based observations. We also provide tasks with a wide range of difficulty, from simple baseline tasks and tasks current algorithms can already solve to harder problems that are currently out of reach. Finally, for the purpose of comparability with prior works, we also include the OpenAI Gym robotic locomotion tasks used by Fujimoto et al. [7], Kumar et al. [18], Wu et al. [37].

5 Tasks and Datasets

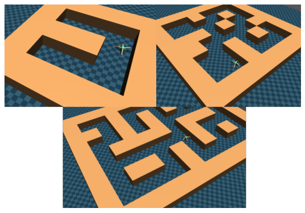
Given the properties outlined in Section 4, and taking into account ease of experimentation, we designed the following tasks and datasets. All tasks consist of a large offline *dataset* (typically 10^6 samples) of transition samples for training, and a *simulator* for evaluation. The mapping is not one-to-one – several tasks use the same simulator with different datasets. A tabular organization of domains, and dataset statistics such as size can be found in Appendix A. Our code is available at `https://github.com/rail-berkeley/d4rl`



Maze2D. *(Non-RL policies, undirected and multitask data)* The Maze2D domain is a navigation task requiring a 2D agent to reach a fixed goal location. The tasks are designed to provide a simple test of the ability of offline RL algorithms to be able to stitch together parts of different trajectories in order to find the shortest path to a new goal, while keeping overall complexity and dimensionality low. Three maze layouts are provided. The "umaze" and "medium" mazes are shown to the right, and the "large" maze is shown below.



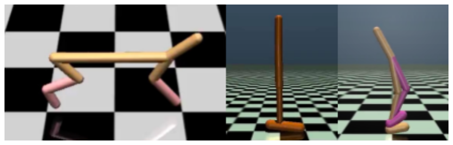
The data is generated by selecting goal location at random and then using a planner that generates sequences of waypoints that are followed using a PD controller. In the figure on the left, the waypoints, represented by green circles, are planned from the starting location (1) along the path to a goal (2). Upon reaching a threshold distance to a waypoint, the controller updates its internal state to track the next waypoint along the path to the goal. Once a goal is reached, a new goal is selected (3) and the process continues. The trajectories in the dataset are visualized in Appendix B.



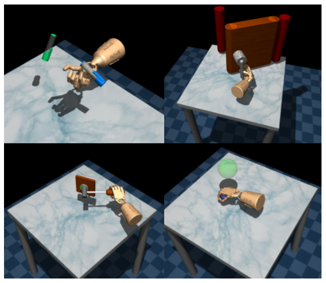
AntMaze. *(Non-RL policies, undirected and multitask data)* The AntMaze domain is a navigation domain that replaces the 2D ball from Maze2D with the more complex 8-DoF "Ant" quadraped robot. We introduce this domain in order to test the stitching property with multitask data using a more morphologically complex robot that could mimic real-world robotic navigation tasks.

The data is generated by training a goal reaching policy and using it in conjunction with the same high-level waypoint generator from **Maze2D** to provide subgoals that guide the agent to the goal. The same 3 maze layouts are used: "umaze", "medium", and "large". We introduce three flavors of datasets. The first dataset commands the ant to reach a specific goal from a fixed start location (`antmaze-umaze-v0`). Next, the "diverse" datasets command the ant to a randomly sampled goal from a randomly sampled start location. Finally, the "play" datasets [24] commands to specific hand-picked locations in the maze (which are not necessarily the goal at evaluation), starting from a different set of hand-picked start locations. As in Maze2D, the trajectories in the dataset are visualized in Appendix B.

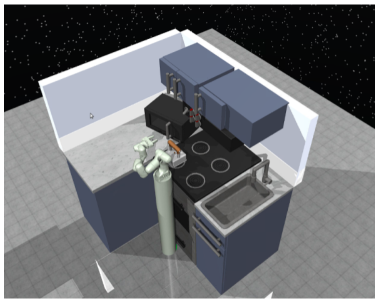
Gym-MuJoCo. *(Suboptimal agents, narrow data distributions)* The Gym-MuJoCo tasks (Hopper, HalfCheetah, Walker2d) are popular benchmarks used in prior work in offline deep RL [7, 18, 37]. Therefore, we introduce standardized datasets, and propose a new task by mixing data between datasets to test the impact of heterogenous policy mixtures.



The "medium" dataset consists of data generated by executing suboptimal. This dataset is generated by first training a policy online using Soft Actor-Critic [12] and but early-stopping the training and collecting 1M samples from this partially trained policy. The "random" datasets are generated by unrolling a randomly initialized policy on these three domains. The "medium-replay" dataset consists of recording all samples in the replay buffer observed during training until the policy reaches the "medium" level of performance. Datasets similar to these three have been used in prior work, but in order to evaluate algorithms on mixtures of policies, we further introduce a "medium-expert" dataset by mixing equal amounts of expert demonstrations and suboptimal data, generated via a partially trained policy or by unrolling a uniform-at-random policy.

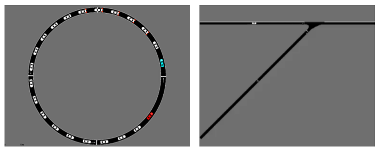


Adroit. *(Non-RL policies, narrow data distributions, realism)* The Adroit domain [32] (pictured left) involves controlling a 24-DoF simulated hand tasked with hammering a nail, opening a door, twirling a pen, or picking up and moving a ball. This domain is designed to measure the effect of a narrow expert data distributions and human demonstrations on a high-dimensional robotic manipulation task. While [32] propose utilizing human demonstrations, in conjunction with online RL fine-tuning, our benchmark adapts these tasks for evaluating the fully offline RL setting. We introduce three types of datasets for each task, two of which are included from the original paper: a small amount of demonstration data from a human ("human"), a large amount of expert data from a fine-tuned RL policy ("expert"). Because the original demonstration dataset only contains 25 trajectories per task, we also introduce a third dataset generated by imitating the human data, running the policy, and mixing data at a 50-50 ratio with the demonstrations, referred to as "cloned." The mixing is performed because the cloned policies themselves do not successfully complete the task, making the dataset otherwise difficult to learn from. The Adroit domain has several unique properties that make it qualitatively different from the Gym MuJoCo tasks. First, the data is collected in the real-world from human demonstrators. Second, each task is difficult to solve with online RL, due to sparse rewards and exploration challenges, which make cloning and online RL alone insufficient. Lastly, the tasks have high dimensionality, presenting a representation learning challenge.



FrankaKitchen. *(Undirected and multitask data, realism)* The Franka Kitchen domain, first proposed by Gupta et al. [11], involves controlling a 9-DoF Franka robot in a kitchen environment containing several common household items: a microwave, a kettle, an overhead light, cabinets, and an oven. The goal of each task is to interact with the items in order to reach a desired goal configuration. For example, one such state is to have the microwave and sliding cabinet door open with the kettle on the top burner and the overhead light on. This domain benchmarks the effect of multitask behavior on a realistic, non-navigation environment which in which the "stitching" property is less obvious since the trajectories are less constrained to simple paths through the state space. This means the algorithms will have to rely on some degree of generalization to unseen states in order to solve the task, rather than relying purely on trajectories seen during training.

In order to study the effect of “stitching” and generalization, we introduce 3 datasets: complete, partial, and mixed, in increasing order of difficulty. The “complete” dataset consists of the robot performing all of the desired tasks in order. This provides data that is easy for an imitation learning method to solve. The “partial” and “mixed” datasets consist of undirected data, where the robot performs subtasks that are not necessarily related to the goal configuration. In the “partial” dataset, a subset of the dataset is guaranteed to solve the task, meaning an imitation learning agent may learn by selectively choosing the right subsets of the data. The “mixed” dataset contains no trajectories which solve the task completely, and the RL agent must learn to assemble the relevant sub-trajectories. This dataset requires the highest degree of generalization in order to succeed.



Flow. *(Non-RL policies, realism)* The Flow benchmark [35] is a framework for studying traffic control using deep reinforcement learning. We use two tasks in the Flow benchmark which involve controlling autonomous vehicles to maximize the flow of traffic through a ring or merge road configuration (left).

We use the Flow domain in order to provide a task that simulates real-world traffic dynamics. A large challenge in autonomous driving is to be able to directly learn from human behavior. Thus, we include “human” data from agents controlled by the intelligent driver model (IDM) [34], a hand-designed model of human driving behavior. In order to provide data with a wider distribution as a reference, we also include “random” data generated from an agent that commands random vehicle accelerations.

Offline CARLA. *(Non-RL policies, undirected and multitask data, realism)* CARLA [4] is a high-fidelity autonomous driving simulator, where the agent controls the throttle (gas pedal), the steering, and the break pedal for the car, and receives 48x48 RGB images from the driver’s perspective as observations. We propose two tasks for offline RL: one is a lane following task within a figure eight path (shown to the right, top picture), and a navigation task within a small town (bottom picture). The principle challenge of the CARLA domain is visual complexity, as all observations are provided as a first-person RGB images.

The datasets in both tasks are generated via hand-designed controllers meant to emulate human driving - the lane-following task uses simple heuristics to avoid cars and keep the car within lane boundaries, whereas the navigation task layers an additional high-level controller on top that takes turns randomly at intersections. Like the Maze2D and AntMaze domains, this dataset consists of undirected navigation data in order to test the “stitching” property, except in a more perceptually challenging domain.

5.1 Evaluation Protocol

The simplest way to evaluate the performance of an algorithm is to perform online execution inside the simulator. However, using online evaluation as a validation method for tuning hyperparameters or model and algorithm selection is not a realistic scenario. As done by Gulcehre et al. [10], we also designate a subset of tasks in each domain as “training” tasks, where hyperparameter tuning is allowed, and another subset as “evaluation” tasks on which final performance is measured. Our recommended split between train and evaluation tasks is outlined in Appendix D Table 4.

We additionally provide reference values to normalize scores for each environment roughly to the range between 0 and 100, by computing $\texttt{normalized score} = 100 * \frac{\texttt{score}-\texttt{random score}}{\texttt{expert score}-\texttt{random score}}$. A normalized score of 0 corresponds to the average returns (over 100 episodes) of an agent taking actions uniformly at random across the action space. A score of 100 corresponds to the average returns of a domain-specific expert. For Maze2D, and Flow domains, this corresponds to the performance of the hand-designed controller used to collect data. For CARLA, AntMaze, and FrankaKitchen, we used an estimate of the maximum score possible. For Adroit, this corresponds to a policy trained with behavioral cloning on human-demonstrations and fine-tuned with RL. For Gym-MuJoCo, this corresponds to a soft-actor critic [13] agent.

6 Benchmarking Prior Methods

We now present results for a number of recently proposed offline RL algorithms, as well as several baselines, on our proposed offline RL benchmarks. The purpose of this evaluation is to a) provide a useful baseline to gauge the difficulty of each task, and b) identify areas of shortcomings in existing offline RL algorithms in order to guide future research. Aggregated results for all algorithms, normalized to lie approximately between 0 and 100, are reported in Table 1. The raw, unnormalized scores can be found in Appendix Table 3, and further details on the experiments in Appendix C.

For baseline algorithms, we evaluate behavioral cloning (BC), online and offline soft actor-critic (SAC) [13], bootstrapping error reduction (BEAR) [18], and behavior-regularized actor-critic (BRAC) [37], advantage-weighted regression (AWR) [29], batch-constrained Q-learning (BCQ) [7], continuous action random ensemble mixtures (cREM) [2], and AlgaeDICE [28]. In most domains, we expect online SAC to outperform offline algorithms when given the same amount of data, since the algorithm is able to collect on-policy data. There are a few exceptions, such as for environments where exploration challenges (such as the Adroit or various maze domains) make it difficult for RL algorithms to find reward signal.

Overall, we find the most success on datasets generated from an RL-trained policy, such as in the Adroit and Gym-MuJoCo domains. In these domains, offline RL algorithms are able to match the behavior policy when given expert data, and outperform when given suboptimal data. This positive result is expected, as it is the predominant setting where algorithms have been benchmarked. However, datasets generated by a mixture of policies of varying quality, such as the medium-expert datasets or the replay datasets often tend to be challenging for some algorithms.

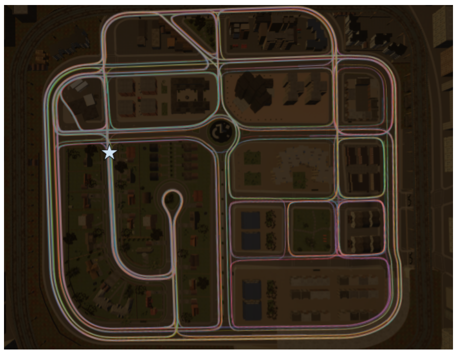
We find that tasks with undirected data, such as the Maze2D, FrankaKitchen, CARLA and AntMaze domains, are challenging for existing methods. Even in the simpler Maze2D domain, the large maze provides a surprising challenge for most methods. However, the smaller instances of Maze2D and AntMaze are very much within reach of current algorithms.

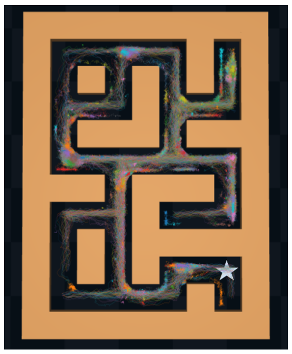
We find that many algorithms are able to handle data generated from non-RL policies in the domains tested. Many algorithms were able to succeed to some extent on tasks with controller-generated data such as Flow and "carla-lane". We also note that the human demonstration setting (Adroit) is inconclusive. The evaluated methods performed poorly, but the datasets contain very few samples (in the order of 10^4 samples) as human demonstrations for a particular task are expensive to collect.

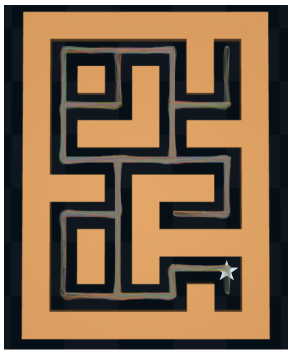
6.1 Assessing the Feasibility of Difficult Tasks

None of the prior offline RL methods were able to successfully solve the AntMaze or carla-town tasks. We therefore took two measures to maximize the chance that these tasks are actually solvable. First, we ensured that the trajectories observed in these tasks has adequate coverage of the state space. An illustration of the trajectories in the CARLA and AntMaze tasks are shown below, where trajectories are shown as different colored lines and the goal state is marked with a star.

Second, for the AntMaze task, the data was generated by having the ant follow the same high-level planner in the maze as in the 2D mazes. While the dynamics of the ant itself are much more complex,


(a) carla-town


(b) antmaze-large-diverse


(c) maze2d-large

	Task Name	SAC	BC	SAC-off	BEAR	BRAC-p	BRAC-v	AWR	cREM	BCQ	aDICE
Maze 2D	maze2d-umaze	62.7	3.8	88.2	3.4	4.7	-16.0	1.0	-15.8	12.8	-15.7
	maze2d-medium	21.3	30.3	26.1	29.0	32.4	33.8	7.6	0.9	8.3	10.0
	maze2d-large	2.7	5.0	-1.9	4.6	10.4	40.6	23.7	-2.2	6.2	-0.1
AntMaze	antmaze-umaze	0.0	65.0	0.0	73.0	50.0	70.0	56.0	0.0	78.9	0.0
	antmaze-umaze-diverse	0.0	55.0	0.0	61.0	40.0	70.0	70.3	0.0	55.0	0.0
	antmaze-medium-play	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	antmaze-medium-diverse	0.0	0.0	0.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0
	antmaze-large-play	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.7	0.0
	antmaze-large-diverse	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.2	0.0
Gym	halfcheetah-random	100.0	2.1	30.5	25.1	24.1	31.2	2.5	-2.6	2.2	-0.3
	walker2d-random	100.0	1.6	4.1	7.3	-0.2	1.9	1.5	-0.3	4.9	0.5
	hopper-random	100.0	9.8	11.3	11.4	11.0	12.2	10.2	0.7	10.6	0.9
	halfcheetah-medium	100.0	36.1	-4.3	41.7	43.8	46.3	37.4	-2.6	40.7	-2.2
	walker2d-medium	100.0	6.6	0.9	59.1	77.5	81.1	17.4	-0.2	53.1	0.3
	hopper-medium	100.0	29.0	0.8	52.1	32.7	31.1	35.9	0.6	54.5	1.2
	halfcheetah-medium-replay	100.0	38.4	-2.4	38.6	45.4	47.7	40.3	-3.0	38.2	-2.1
	walker2d-medium-replay	100.0	11.3	1.9	19.2	-0.3	0.9	15.5	-0.2	15.0	0.6
	hopper-medium-replay	100.0	11.8	3.5	33.7	0.6	0.6	28.4	0.8	33.1	1.1
	halfcheetah-medium-expert	100.0	35.8	1.8	53.4	44.2	41.9	52.7	-2.6	64.7	-0.8
	walker2d-medium-expert	100.0	6.4	-0.1	40.1	76.9	81.6	53.8	-0.2	57.5	0.4
	hopper-medium-expert	100.0	111.9	1.6	96.3	1.9	0.8	27.1	0.7	110.9	1.1
Adroit	pen-human	21.6	34.4	6.3	-1.0	8.1	0.6	12.3	3.5	68.9	-3.3
	hammer-human	0.2	1.5	0.5	0.3	0.3	0.2	1.2	0.2	0.5	0.3
	door-human	-0.2	0.5	3.9	-0.3	-0.3	-0.3	0.4	-0.1	-0.0	-0.0
	relocate-human	-0.2	0.0	0.0	-0.3	-0.3	-0.3	-0.0	-0.2	-0.1	-0.1
	pen-cloned	21.6	56.9	23.5	26.5	1.6	-2.5	28.0	-3.4	44.0	-2.9
	hammer-cloned	0.2	0.8	0.2	0.3	0.3	0.3	0.4	0.2	0.4	0.3
	door-cloned	-0.2	-0.1	0.0	-0.1	-0.1	-0.1	0.0	-0.1	0.0	0.0
	relocate-cloned	-0.2	-0.1	-0.2	-0.3	-0.3	-0.3	-0.2	-0.2	-0.3	-0.3
	pen-expert	21.6	85.1	6.1	105.9	-3.5	-3.0	111.0	0.3	114.9	-3.5
	hammer-expert	0.2	125.6	25.2	127.3	0.3	0.3	39.0	0.2	107.2	0.3
	door-expert	-0.2	34.9	7.5	103.4	-0.3	-0.3	102.9	-0.2	99.0	0.0
	relocate-expert	-0.2	101.3	-0.3	98.6	-0.3	-0.4	91.5	-0.1	41.6	-0.1
Flow	flow-ring-controller	100.7	-57.0	9.2	62.7	-12.3	-91.2	75.2	-47.4	76.2	15.2
	flow-ring-random	100.7	94.9	70.0	103.5	95.7	78.6	80.4	-87.4	94.6	83.6
	flow-merge-controller	121.5	114.1	111.6	150.4	129.8	143.9	152.7	183.2	114.8	196.4
	flow-merge-random	121.5	-17.1	-40.1	-20.6	146.2	27.3	99.6	-31.2	28.2	4.7
Franka Kitchen	kitchen-complete	0.0	33.8	15.0	0.0	0.0	0.0	0.0	0.0	8.1	0.0
	kitchen-partial	0.6	33.8	0.0	13.1	0.0	0.0	15.4	0.0	18.9	0.0
	kitchen-mixed	0.0	47.5	2.5	47.2	0.0	0.0	10.6	0.0	8.1	2.5
CARLA	carla-lane	-0.8	31.8	0.1	-0.2	18.2	19.6	-0.4	0.1	-0.1	-1.2
	carla-town	1.4	-1.8	-1.8	-2.7	-4.6	-2.6	1.9	-0.9	1.9	-11.2

Table 1: Normalized results comparing online & offline SAC (SAC, SAC-off), bootstrapping error reduction (BEAR), behavior-regularized actor critic with policy (BRAC-p) or value (BRAC-v) regularization, behavioral cloning (BC), advantage-weighted regression (AWR), batch-constrained Q-learning (BCQ), continuous random ensemble mixtures (cREM), and AlgaeDICE (aDICE). Average results are reported over 3 seeds, and normalized to a score between 0 (random) and 100 (expert).

its walking gait is a relatively regular periodic motion, and since the high-level waypoints are similar, we would expect the AntMaze data to provide similar coverage as in the 2D mazes. The coverage of Maze2D is shown below the AntMaze on the right on the large maze layout. While the Ant has more erratic motion, both datasets cover the the majority of the maze thoroughly. A comparison of the state coverage between Maze2D and AntMaze on all domains is shown in Appendix B.

7 Discussion

We have proposed an open-source benchmark for offline reinforcement learning. The selection of the benchmark tasks were motivated by properties that we believe realistic data is likely to have, such as narrow data distributions and undirected or multitask behavior. Existing benchmarks have largely been concentrated on robotic control using data generated by previously optimized RL algorithms [7, 18, 37], which in our view, can give a misleading sense of progress as this is a particularly narrow application of offline RL. Indeed, our evaluations reveal a lack of ability for existing offline RL algorithms to handle properties such as undirected data, which may be crucial for the success of offline RL in many real-world domains.

Ultimately, we would like to see offline RL applications move from simulated domains to real-world domains where significant amounts of offline data is easily obtainable. This includes exciting areas such as recommender systems and natural language interfaces, where user behavior can be easily logged, and medical diagnoses, where doctors must record symptoms, diagnoses, and treatments in complete medical records for each patient. A key challenge for developing algorithms on these domains is that reliable evaluation must be done in a *real system*, which significantly slows down the pace at which one can iteratively improve an algorithm.

Offline RL holds great promise as a potential paradigm to leverage vast amounts of existing sequential data within the flexible decision making framework of reinforcement learning. We hope that providing a benchmark that is representative of potential problems in offline RL, but that still can be cheaply evaluated in simulation, will greatly accelerate progress in this field and create new opportunities to apply RL in many real-world application areas.

Acknowledgements

We would like to thank Abhishek Gupta, Aravind Rajeswaran, Eugene Vinitsky, and Rowan McAllister for providing implementations and assistance in setting up tasks, Michael Janner for informative discussions, and Kelvin Xu and Yinlam Chow for feedback on an earlier draft of this paper. We would like to thank NVIDIA, Google Cloud Platform and Amazon Web Services for providing computational resources. This research was supported by the Office of Naval Research and the DARPA Assured Autonomy Program.

References

[1] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. *arXiv preprint arXiv:1908.00261*, 2019.

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. Striving for simplicity in off-policy deep reinforcement learning. *CoRR*, abs/1907.04543, 2019. URL `http://arxiv.org/abs/1907.04543`.

[3] Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Żołna, Yusuf Aytar, David Budden, Mel Vecerik, et al. A framework for data-driven robotics. *arXiv preprint arXiv:1909.12200*, 2019.

[4] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[5] Simon S. Du, Sham M. Kakade, Ruosong Wang, and Lin F. Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=r1genAVKPB`.

[6] Amir-massoud Farahmand, Csaba Szepesvári, and Rémi Munos. Error propagation for approximate policy and value iteration. In *Advances in Neural Information Processing Systems*, pages 568–576, 2010.

[7] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[9] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[10] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gómez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, Gabriel Dulac-Arnold, Jerry Li, Mohammad Norouzi, Matt Hoffman, Ofir Nachum, George Tucker, Nicolas Heess, and Nando de Freitas. Rl unplugged: Benchmarks for offline reinforcement learning, 2020.

[11] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018. URL `http://arxiv.org/abs/1801.01290`.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[14] Daniel Hein, Steffen Udluft, Michel Tokic, Alexander Hentschel, Thomas A Runkler, and Volkmar Sterzing. Batch reinforcement learning on the industrial benchmark: First experiences. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4214–4221. IEEE, 2017.

[15] James Henderson, Oliver Lemon, and Kallirroi Georgila. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 34(4):487–511, 2008.

[16] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Àgata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind W. Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *CoRR*, abs/1907.00456, 2019. URL `http://arxiv.org/abs/1907.00456`.

[17] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673, 2018.

[18] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, 2019. URL `http://arxiv.org/abs/1906.00949`.

[19] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[20] Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning (ICML)*, 2019.

[21] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[22] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[23] Tyler Lu, Dale Schuurmans, and Craig Boutilier. Non-delusional q-learning and value-iteration. In *Advances in neural information processing systems*, pages 9949–9959, 2018.

[24] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. *Conference on Robot Learning (CoRL)*, 2019. URL `https://arxiv.org/abs/1903.01973`.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836.

[26] Kaichun Mo, Haoxiang Li, Zhe Lin, and Joon-Young Lee. The adobeindoornav dataset: Towards deep reinforcement learning based real-world indoor robot visual navigation. 2018.

[27] Rémi Munos. Error bounds for approximate policy iteration. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, pages 560–567. AAAI Press, 2003.

[28] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

[29] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[30] Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3):1–21, 2011.

[31] Doina Precup, Richard S Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. In *ICML'00 Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[32] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems*, 2018.

[33] Philip S Thomas. *Safe reinforcement learning*. PhD thesis, University of Massachusetts Libraries, 2015.

[34] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[35] Eugene Vinitsky, Aboudy Kreidieh, Luc Le Flem, Nishant Kheterpal, Kathy Jang, Cathy Wu, Fangyu Wu, Richard Liaw, Eric Liang, and Alexandre M Bayen. Benchmarks for reinforcement learning in mixed-autonomy traffic. In *Conference on Robot Learning*, pages 399–409, 2018.

[36] Christopher J.C.H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[37] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Appendices

A Task Properties

The following is a full list of task properties and dataset statistics for all tasks in the benchmark. Note that the full dataset for "carla-town" requires over 30GB of memory to store, so we also provide a subsampled version of the dataset which we used in our experiments.

Domain	**Task Name**	**Controller Type**	**# Samples**
Maze2D	maze2d-umaze	Planner	10^6
	maze2d-medium	Planner	$2 * 10^6$
	maze2d-large	Planner	$4 * 10^6$
AntMaze	antmaze-umaze	Planner	10^6
	antmaze-umaze-diverse	Planner	10^6
	antmaze-medium-play	Planner	10^6
	antmaze-medium-diverse	Planner	10^6
	antmaze-large-play	Planner	10^6
	antmaze-large-diverse	Planner	10^6
Gym-MuJoCo	hopper-random	Policy	10^6
	hopper-medium	Policy	10^6
	hopper-medium-replay	Policy	200920
	hopper-medium-expert	Policy	2×10^6
	halfcheetah-random	Policy	10^6
	halfcheetah-medium	Policy	10^6
	halfcheetah-medium-replay	Policy	101000
	halfcheetah-medium-expert	Policy	2×10^6
	walker2d-random	Policy	10^6
	walker2d-medium	Policy	10^6
	walker2d-medium-replay	Policy	100930
	walker2d-medium-expert	Policy	2×10^6
Adroit	pen-human	Human	5000
	pen-cloned	Policy	$5 * 10^5$
	pen-expert	Policy	$5 * 10^5$
	hammer-human	Human	11310
	hammer-cloned	Policy	10^6
	hammer-expert	Policy	10^6
	door-human	Human	6729
	door-cloned	Policy	10^6
	door-expert	Policy	10^6
	relocate-human	Human	9942
	relocate-cloned	Policy	10^6
	relocate-expert	Policy	10^6
Flow	flow-ring-random	Policy	10^6
	flow-ring-controller	Policy	10^6
	flow-merge-random	Policy	10^6
	flow-merge-controller	Policy	10^6
FrankaKitchen	kitchen-complete	Policy	3680
	kitchen-partial	Policy	136950
	kitchen-mixed	Policy	136950
CARLA	carla-lane	Planner	10^5
	carla-town	Planner	$2 * 10^6$ full
			10^5 subsampled

Table 2: Statistics for each task in the benchmark. For the controller type, "planner" refers to a hand-designed navigation planner, "human" refers to human demonstrations, and "policy" refers to random or neural network policies. The number of samples refers to the number of environment transitions recorded in the dataset.

Domain	Task Name	SAC	BC	SAC-off	BEAR	BRAC-p	BRAC-v	AWR	cREM	BCQ	AlgaeDICE
Maze2D	maze2d-umaze	110.4	29.0	145.6	28.6	30.4	1.7	25.2	2.1	41.5	2.2
	maze2d-medium	69.5	93.2	82.0	89.8	98.8	102.4	33.2	15.6	35.0	39.6
	maze2d-large	14.1	20.1	1.5	19.0	34.5	115.2	70.1	0.8	23.2	6.5
AntMaze	antmaze-umaze	0.0	0.7	0.0	0.7	0.5	0.7	0.6	0.0	0.8	0.0
	antmaze-umaze-diverse	0.0	0.6	0.0	0.6	0.4	0.7	0.7	0.0	0.6	0.0
	antmaze-medium-play	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	antmaze-medium-diverse	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
	antmaze-large-play	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0
	antmaze-large-diverse	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Gym	halfcheetah-random	12135.0	-17.9	3502.0	2831.4	2713.6	3590.1	36.3	-597.6	-1.3	-318.0
	walker2d-random	4592.3	73.0	192.0	336.3	-7.2	87.4	71.5	-12.7	228.0	26.5
	hopper-random	3234.3	299.4	347.7	349.9	337.5	376.3	312.4	4.0	323.9	10.0
	halfcheetah-medium	12135.0	4196.4	-808.6	4897.0	5158.8	5473.8	4366.1	-600.0	4767.9	-551.6
	walker2d-medium	4592.3	304.8	44.2	2717.0	3559.9	3725.8	800.7	-9.3	2441.0	15.5
	hopper-medium	3234.3	923.5	5.7	1674.5	1044.0	990.4	1149.5	0.3	1752.4	17.5
	halfcheetah-medium-replay	12135.0	4492.1	-581.3	4517.9	5350.8	5640.6	4727.4	-655.7	4463.9	-540.8
	walker2d-medium-replay	4592.3	518.6	87.8	883.8	-11.5	44.5	712.5	-8.7	688.7	31.0
	hopper-medium-replay	3234.3	364.4	93.3	1076.8	0.5	-0.8	904.0	5.8	1057.8	14.0
	halfcheetah-medium-expert	12135.0	4169.4	-55.7	6349.6	5208.1	4926.6	6267.3	-600.8	7750.8	-377.6
	walker2d-medium-expert	4592.3	297.0	-5.1	1842.7	3533.1	3747.5	2469.7	-8.0	2640.3	19.7
	hopper-medium-expert	3234.3	3621.2	32.9	3113.5	42.6	5.1	862.0	3.1	3588.5	15.5
Adroit	pen-human	739.3	1121.9	284.8	66.3	339.0	114.7	463.1	199.7	2149.0	-0.7
	hammer-human	-248.7	-82.4	-214.2	-242.0	-239.7	-243.8	-115.3	-243.2	-210.5	-234.8
	door-human	-61.8	-41.7	57.2	-66.4	-66.5	-66.4	-44.4	-58.7	-56.6	-56.5
	relocate-human	-13.7	-5.6	-4.5	-18.9	-19.7	-19.7	-7.2	-14.1	-8.6	-10.8
	pen-cloned	739.3	1791.8	797.6	885.4	143.4	22.2	931.3	-6.5	1407.8	8.6
	hammer-cloned	-248.7	-175.1	-244.1	-241.1	-236.7	-236.9	-226.9	-245.8	-224.4	-233.1
	door-cloned	-61.8	-60.7	-56.3	-60.9	-58.7	-59.0	-56.1	-58.0	-56.3	-56.4
	relocate-cloned	-13.7	-10.1	-16.1	-17.6	-19.8	-19.4	-16.6	-16.9	-17.5	-18.8
	pen-expert	739.3	2633.7	277.4	3254.1	-7.8	6.4	3406.0	104.0	3521.3	-6.9
	hammer-expert	-248.7	16140.8	3019.5	16359.7	-241.4	-241.1	4822.9	-247.1	13731.5	-235.2
	door-expert	-61.8	969.4	163.8	2980.1	-66.4	-66.6	2964.5	-60.9	2850.7	-56.5
	relocate-expert	-13.7	4289.3	-18.2	4173.8	-20.6	-21.4	3875.5	-11.7	1759.6	-8.7
Flow	flow-ring-controller	25.8	-273.3	-147.7	-46.3	-188.5	-338.2	-22.6	-255.0	-20.7	-136.3
	flow-ring-random	25.8	14.7	-32.4	31.0	16.2	-16.2	-12.7	-330.9	14.2	-6.8
	flow-merge-controller	375.4	359.8	354.6	436.5	392.9	422.9	441.4	505.8	361.4	533.9
	flow-merge-random	375.4	82.6	33.9	75.1	427.6	176.3	329.3	52.7	178.3	128.7
FrankaKitchen	kitchen-complete	0.0	1.4	0.6	0.0	0.0	0.0	0.0	0.0	0.3	0.0
	kitchen-partial	0.0	1.4	0.0	0.5	0.0	0.0	0.6	0.0	0.8	0.0
	kitchen-mixed	0.0	1.9	0.1	1.9	0.0	0.0	0.4	0.0	0.3	0.1
Offline CARLA	carla-lane	-8.6	324.7	-0.3	-3.0	186.1	199.6	-4.5	-0.0	-1.4	-13.4
	carla-town	-79.7	-161.5	-159.9	-182.5	-231.6	-181.5	-65.8	-137.4	-66.6	-400.2

Table 3: The raw, un-normalized scores for each task and algorithm are reported in the table below. These scores represent the undiscounted return obtained from executing a policy in the simulator, averaged over 3 random seeds.

B Maze Domain Trajectories

In this section, we visualized trajectories for the datasets in the Maze2D and AntMaze domains. Each image plots the states visited along each trajectory as a different colored line, overlaid on top of the maze. The goal state is marked as a white star.

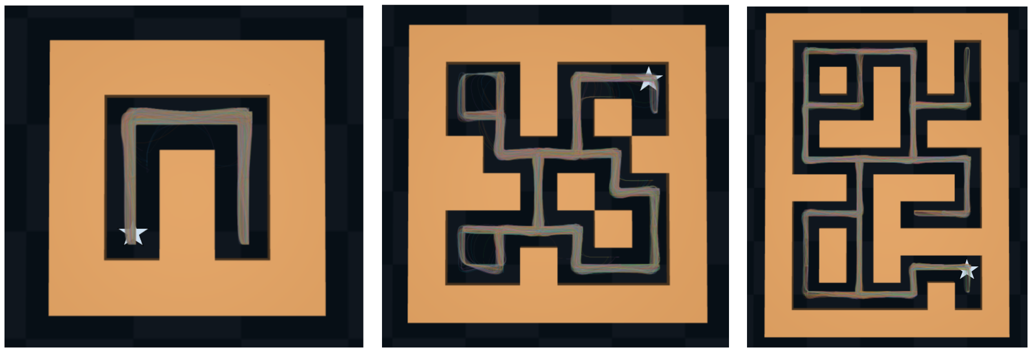


Figure 4: Trajectories visited in the Maze2D domain. From left-to-right: maze2d-umaze, maze2d-medium, and maze2d-large.

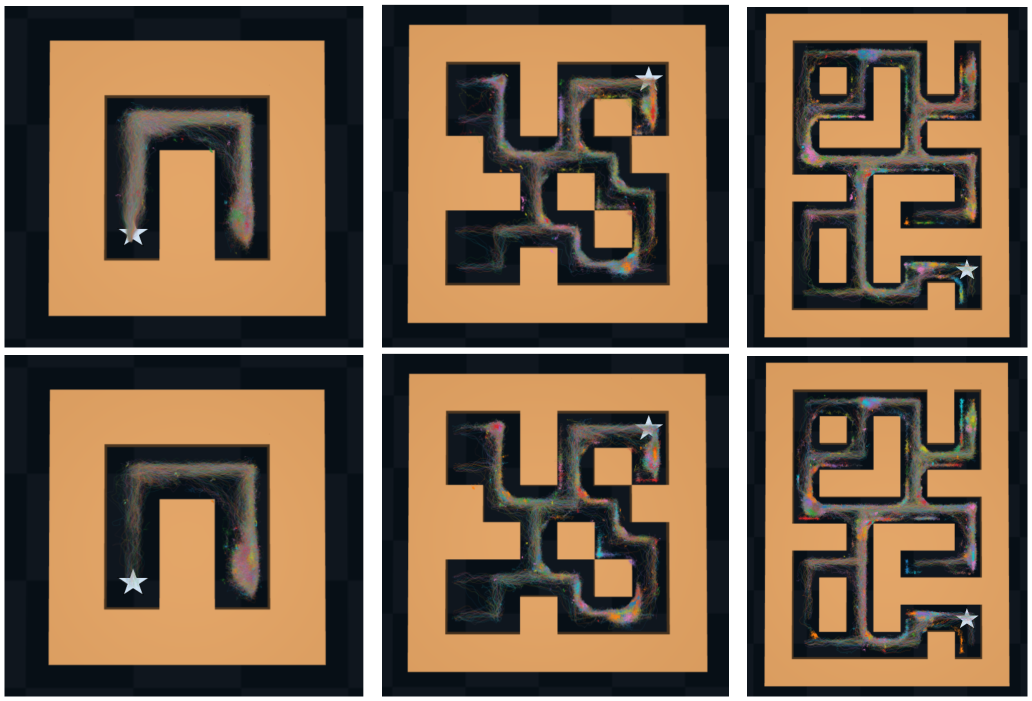


Figure 5: Trajectories visited in the AntMaze domain. Top row, from left-to-right: antmaze-umaze, antmaze-medium-play, and antmaze-large-play. Bottom row, from left-to-right: antmaze-umaze-diverse, antmaze-medium-diverse, and antmaze-large-diverse.

C Experiment Details

For all experiments, we used default hyperparameter settings and minimal modifications to public implementations wherever possible, using 500K training iterations or gradient steps. The code bases we used for evaluation are listed below. The most significant deviation from original published algorithms was that we used an unofficial continuous-action implementation of REM [2], which was originally implemented for discrete action spaces. We ran our experiments using Google cloud platform (GCP) on `n1-standard-4` machines.

- BRAC and AlgaeDICE: `https://github.com/google-research/google-research`
- AWR: `https://github.com/xbpeng/awr`
- SAC: `https://github.com/vitchyr/rlkit`
- BEAR: `https://github.com/aviralkumar2907/BEAR`
- Continuous-action REM: `https://github.com/theSparta/off_policy_mujoco`
- BCQ: `https://github.com/sfujim/BCQ`

D Training and Evaluation Task Split

The following table lists our recommended protocol for hyperparameter tuning. Hyperparameters should be tuned on the tasks listed on the left in the "Training" column, and algorithms should be evaluated without tuning on the tasks in the right column labeled "Evaluation".

Domain	**Training**	**Evaluation**
Maze2D	maze2d-umaze	maze2d-eval-umaze
	maze2d-medium	maze2d-eval-medium
	maze2d-large	maze2d-eval-large
AntMaze	ant-umaze	ant-eval-umaze
	ant-umaze-diverse	ant-eval-umaze-diverse
	ant-medium-play	ant-eval-medium-play
	ant-medium-diverse	ant-eval-medium-diverse
	ant-large-play	ant-eval-large-play
	ant-large-diverse	ant-eval-large-diverse
Adroit	pen-human	hammer-human
	pen-cloned	hammer-cloned
	pen-expert	hammer-expert
	door-human	relocate-human
	door-cloned	relocate-cloned
	door-expert	relocate-expert
Gym	halfcheetah-random	hopper-random
	halfcheetah-medium	hopper-medium
	halfcheetah-mixed	hopper-mixed
	halfcheetah-medium-expert	hopper-medium-expert
	walker2d-random	ant-random
	walker2d-medium	ant-medium
	walker2d-mixed	ant-mixed
	walker2d-medium-expert	and-medium-expert

Table 4: Our recommended partition of tasks into "training" tasks where hyperparameter tuning is allowed, and "evaluation" tasks where final algorithm performance should be reported.