Guided Data Augmentation for Offline Reinforcement Learning and Imitation Learning

Nicholas E. Corrado[1], **Yuxiao Qu**[2], **John U. Balis**[1], **Adam Labiosa**[1], **Josiah P. Hanna**[1]
University of Wisconsin–Madison[1], Carnegie Mellon University[2]
{ncorrado, balis, labiosa}@wisc.edu, yuxaioq@andrew.cmu.edu, jphanna@cs.wisc.edu

Abstract

In offline reinforcement learning (RL), RL agents learn to solve a task using only a fixed dataset of previously collected data. While offline RL has proven to be a viable method for learning real-world robot control policies, it typically requires large amounts of expert-quality data to learn effective policies that generalize to out-of-distribution states. Unfortunately, such data is often difficult and expensive to acquire in real-world tasks. Several recent works have leveraged data augmentation (DA) to inexpensively generate additional data, but most DA works apply augmentations in a random fashion and ultimately produce highly suboptimal augmented data. In this work, we propose **Gu**ided **D**ata **A**ugmentation (GuDA), a human-guided DA framework that generates expert-quality augmented data. The key insight behind GuDA is that while it may be difficult to demonstrate the sequence of actions required to produce expert data, a user can often easily characterize when an augmented trajectory segment represents progress toward task completion. Thus, a user can restrict the space of possible augmentations to automatically reject suboptimal augmented data. To extract a policy from GuDA, we use off-the-shelf offline reinforcement learning and behavior cloning algorithms. We evaluate GuDA on a physical robot soccer task as well as simulated D4RL navigation tasks, a simulated autonomous driving task, and a simulated soccer task. Empirically, GuDA enables learning given a small initial dataset of potentially suboptimal experience and outperforms a random DA strategy as well as a model-based DA strategy. We include videos and code at https://nicholascorrado.github.io/projects/GuDA/.

1 Introduction

Offline reinforcement learning (RL) is a learning paradigm in which RL agents learn to solve a task using only a static dataset of previously collected data. While offline RL algorithms can produce effective real-world robot control policies without the expense or danger of active task interaction (Levine et al., 2020), their performance and generalization capabilities depend greatly on the size and quality of the provided dataset. Ideally, we would provide large amounts of high-coverage, near expert-quality trajectories, but acquiring such data in real-world tasks is often challenging: the expense of data collection often limits us to just a few trajectories, and their quality depends on the performance of the data collection policy. Although prior works have shown that offline RL algorithms can perform well even with highly suboptimal data (Kumar et al., 2019; Fujimoto et al., 2019; Kumar et al., 2020; Fujimoto & Gu, 2021), these same works show that these algorithm learn far more effective policies with expert-quality data. As such, we focus on developing methods that produce expert-quality data without requiring a human to demonstrate expert behavior.

To improve performance and generalization of RL agents, a number of works have leveraged *data augmentation* (Laskin et al., 2020) (DA), a technique in which agents generate additional synthetic experience without the expense of task interaction by applying transformations to previously collected experience. These transformations – or *data augmentation functions* (DAFs) – often leverage

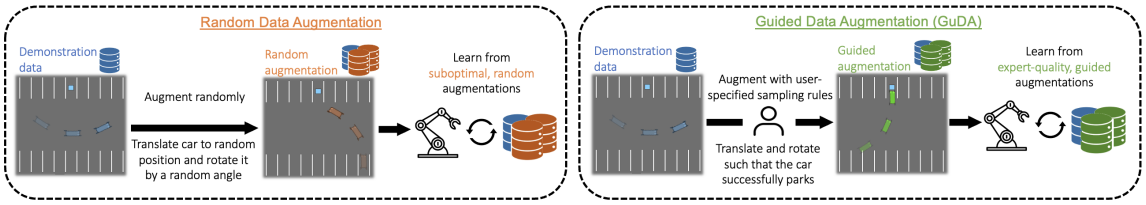


Figure 1: An overview of GuDA applied to a parking task given DAFs that translate and rotate a trajectory segment τ. A user first defines a *sampling procedure* describing how to translate and rotate τ to produce expert-quality data: translate τ so that the agent's final position is at the parking spot, and then rotate τ such that the agent is aligned with the parking spot. We augment our dataset using this sampling procedure and then learn a policy with offline RL or imitation learning.

task-specific invariances and symmetries inherent to many real-world tasks (*e.g.* translational invariance (Pitis et al., 2020; 2022), gait symmetry (Abdolhosseini et al., 2019; Mikhail Pavlov & Plis, 2018)). Viewing DA as a means to improve dataset coverage, most prior works generate highly diverse augmented data by sampling data uniformly at random from a DAF (Sinha et al., 2022a; Pitis et al., 2020; Joo et al., 2022; Cho et al., 2022; Lu et al., 2020) or from a learned dynamics model (Hepburn & Montana, 2022; Wang et al., 2022; Han & Kim, 2022). However, these random DA strategies generally produce highly suboptimal experience. Thus, we aim to develop a DA strategy that produces both high-coverage and high-quality augmented data.

We propose **Gu**ided **D**ata **A**ugmentation (GuDA), a human-guided DA framework that generates large amounts of expert-quality data from a limited set of potentially suboptimal data. The key insight behind GuDA is that a human can often determine if an augmented trajectory segment resembles expert data by simply checking if its sequence of states brings the agent closer to solving the task. Thus, a user can restrict the space of DAF transformations to only generate augmented data that represents progress toward task completion. To make this concept more concrete, imagine training an autonomous vehicle to park in a parking lot given a single suboptimal trajectory (Fig. 1). Since a parking lot has a relatively uniform surface, we can generate augmented experience by translating and rotating the agent. Sampling augmented data uniformly at random will most often produce data in which the agent drives away from the parking spot or approaches it at an unfavorable angle. However, we can generate expert-quality augmented data by translating and rotating trajectory segments such that the agent successfully parks.

GuDA enables practitioners to generate expert data from potentially suboptimal experience without the expense of task interaction. Additionally, instead of requiring that an expert provide an optimal sequence of actions solving a task, GuDA simply requires the user to characterize when an augmented trajectory segment represents progress toward task completion. We evaluate GuDA with off-the-shelf offline RL algorithms on simulated navigation, autonomous driving, and soccer tasks as well as a physical robot soccer task. Since GuDA is also compatible with imitation learning algorithms (which require expert data), we also evaluate GuDA with behavior cloning. Empirically, GuDA produces effective policies given a small amount of data – even highly suboptimal data – while a model-based DA strategy often fails due to poor model generalization. Moreover, polices trained under GuDA achieve larger returns than policies trained under a DA strategy that samples augmented data uniformly at random, emphasizing the importance of generating high-quality augmented data. In summary, our core contributions are

1. We demonstrate how a human can guide data augmentation to inexpensively produce expert-quality data from potentially suboptimal experience.
2. We show that GuDA yields effective policies even when provided a small initial dataset.
3. We show that GuDA outperforms the most widely used DA strategy of sampling augmented data randomly, highlighting the benefits of generating expert-quality augmented data.

2 Related Work

2.1 Data Augmentation

Data augmentation (DA) refers to techniques that generate synthetic data by transforming previously collected experience and has been applied a variety of tasks, including algorithm discovery (Fawzi et al., 2022), locomotion (Mikhail Pavlov & Plis, 2018; Abdolhosseini et al., 2019), and physical robot manipulation (George et al., 2023; Mitrano & Berenson, 2022).

DA is often used to generate perturbed data with the same semantic meaning as the original data. Many vision-based RL works have trained agents to be robust to visual augmentations (Laskin et al., 2020; Guan et al., 2021; Wang et al., 2020; Yarats et al., 2021; Raileanu et al., 2021; Hansen & Wang, 2021; Hansen et al., 2021), and similar approaches have been applied to non-visual tasks (Sinha et al., 2022b; Weissenbacher et al., 2022; Qiao et al., 2021). These approaches are orthogonal to GuDA; they use DA to improve policy robustness, while GuDA uses DA to improve dataset coverage and quality. Perturbation-based DA methods more closely relate to domain randomization (Sadeghi & Levine, 2016; Tobin et al., 2017; Peng et al., 2018) which also aims for policy robustness.

Other works exploit invariances and symmetries in a task's dynamics to generate data that is semantically different from the original data. Hindsight experience replay (HER) (Andrychowicz et al., 2017; Fang et al., 2018) counter-factually relabels a trajectory's goal. Counterfactual Data Augmentation (CoDA) (Pitis et al., 2020) and Model-based CoDA (MoCoDA) (Pitis et al., 2022) exploit local causal independence in a task's dynamics to generate additional data. Several works use a learned model to generate augmented data (Lu et al., 2020; Wang et al., 2022; Hepburn & Montana, 2022; Sutton, 1990; Gu et al., 2016; Venkatraman et al., 2016; Racanière et al., 2017). Most of these works focus on developing new DAFs and simply generate augmented experience in a random fashion. In contrast, GuDA focuses on the importance of sampling expert-quality augmentations.

Two prior works closely relate to GuDA in that they aim to sample task-relevant augmented data: EXPAND (Guan et al., 2020), which applies visual augmentations to image regions identified by human feedback, and MoCoDA (Pitis et al., 2022), which generates augmented data by sampling $(\boldsymbol{s}, \boldsymbol{a})$ pairs from a user-defined *parent distribution* $P(\boldsymbol{s}, \boldsymbol{a})$ and then computes $\boldsymbol{s}'$ from a learned dynamics model. GuDA differs from EXPAND in that GuDA focuses on non-visual tasks with DAFs more relevant to robotics. While MoCoDA can in principle generate expert data using an appropriately defined parent distribution, the user must specify the distribution over expert actions. In contrast, GuDA requires no knowledge of the expert actions and simply requires the user to characterize data that represent task progress. Moreover, GuDA is a model-free DA framework and can be used when data is too scarce to model the task's dynamics, as is common in physical tasks.

2.2 Offline Reinforcement Learning

Offline RL (Levine et al., 2020) methods learn a reward-maximizing policy from reward labels provided with a fixed dataset of task interactions. These methods are designed such that, in principle, they can learn even with suboptimal data, though they are generally far more successful with expert data (Kumar et al., 2019; Fujimoto et al., 2019; Kumar et al., 2020; Fujimoto & Gu, 2021).

One core challenge with offline RL is extrapolation error: state-action pairs outside of the dataset's support can attain arbitrarily inaccurate state-action values during training, causing learning instabilities and poor generalization during deployment (Gulcehre et al., 2020). This challenge is especially problematic for real-world robotics tasks in which offline data is scarce. Offline RL algorithms typically mitigate extrapolation error with policy parameterizations that only consider state-action pairs within the dataset (Fujimoto et al., 2019; Ghasemipour et al., 2021; Zhou et al., 2021) or with behavioral cloning regularization (Nair et al., 2020; Fujimoto & Gu, 2021; Xu et al., 2021). GuDA, like other DA strategies, can be viewed as a technique to mitigate extrapolation error by simply generating more data to improve dataset coverage without further task interaction. However, GuDA also improves dataset quality by generating expert-quality augmented data.

3 Preliminaries

3.1 Offline Reinforcement Learning

We consider finite-horizon Markov decision processes (MDPs) (Puterman, 2014) defined by $(\mathcal{S}, \mathcal{A}, p, r, d_0, \gamma)$ where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action space, respectively; $p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a})$ denotes the probability density of the next state $\boldsymbol{s}'$ after taking action $\boldsymbol{a}$ in state $\boldsymbol{s}$; and $r(\boldsymbol{s}, \boldsymbol{a})$ denotes the reward for taking action $\boldsymbol{a}$ in state $\boldsymbol{s}$.[1] We write d_0 as the initial state distribution, $\gamma \in [0, 1)$ as the discount factor, and H the episode length. We consider stochastic policies $\pi_{\boldsymbol{\theta}} : \mathcal{S} \times \mathcal{A} \to [0, 1]$ parameterized by $\boldsymbol{\theta}$. The RL objective is to find a policy that maximizes the expected sum of discounted rewards $J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]$. In the offline RL paradigm, the agent cannot collect data through environment interaction and must instead learn from a static dataset $\mathcal{D}$ of transitions collected by a different policy.

3.2 Data Augmentation Functions

In this section, we introduce a general notion of a data augmentation function (DAF). At a high level, a DAF generates augmented data by applying transformations to an input trajectory segment. More formally, let $\mathcal{T}$ denote the set of all possible trajectory segments and let $\Delta(\mathcal{T})$ denote the set of distributions over $\mathcal{T}$. A DAF is a stochastic function $f : \mathcal{T} \to \Delta(\mathcal{T})$ mapping a trajectory segment $((\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}'_i))_{i=1}^k$ of length k to an augmented trajectory segment $((\tilde{\boldsymbol{s}}_i, \tilde{\boldsymbol{a}}_i, \tilde{r}_i, \tilde{\boldsymbol{s}}'_i))_{i=1}^k$. In this work, we focus on *dynamics invariant* DAFs which produce realistic data that respect the task's dynamics and reward function, *i.e.* $p(\tilde{\boldsymbol{s}}' \mid \tilde{\boldsymbol{s}}, \tilde{\boldsymbol{a}}) > 0$, and $\tilde{r} = r(\tilde{\boldsymbol{s}}, \tilde{\boldsymbol{a}})$ (Corrado & Hanna, 2024). As in most prior works, we assume a user can specify a DAF f for a given domain (Pitis et al., 2020).

4 Guided Data Augmentation

In this section, we introduce Guided Data Augmentation (GuDA), a DA framework that automatically generates expert-quality augmented data. We provide a high-level overview of GuDA in Section 4.1, and then describe how we implement GuDA in Section 4.2.

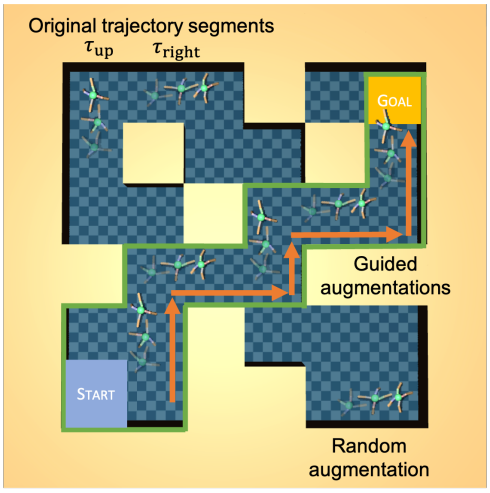


Figure 2: GuDA translates trajectory segments $\tau_{\text{up}}, \tau_{\text{right}}$ to demonstrate the agent walking to the goal. A random translation (bottom right) may be highly suboptimal.

4.1 Method Overview

We assume access to a dataset $\mathcal{D}$ of task interactions and DAFs $f_1, \ldots, f_m$. Prior to offline training, GuDA generates an augmented dataset $\widetilde{\mathcal{D}}$ consisting of the original dataset plus n augmented samples generated from the composition of DAFs $f = f_1 \circ \cdots \circ f_m$. Afterwards, an agent learns from $\widetilde{\mathcal{D}}$ using an off-the-shelf offline RL or imitation learning algorithm. The core difference between GuDA and previous DA works lies in how GuDA samples augmented data from f. Prior works typically sample augmented data uniformly at random, but most transformations under f produce highly suboptimal experience. However, a user can often easily characterize when an augmented trajectory segment represents progress toward task completion. Thus, to generate augmented data that closely resembles expert data, GuDA has the user define a *sampling procedure* that describes how to sample augmentations from f to produce data in which the agent makes task progress.

[1] If a reward function is unavailable, GuDA can be used with imitation learning methods such as behavior cloning which only assume access to expert data and do not require access to a reward function.

To illustrate how a user might identify a sampling procedure, consider a maze navigation task in which a legged robot must reach a fixed goal state from a fixed initial position (Fig. 2). We assume access to a DAF that translates the agent to a new position. While it is difficult to demonstrate the precise sequence of leg movements required to optimally solve the maze, we can easily identify when a trajectory segment progresses the agent toward its goal. A randomly sampled augmentation from our DAF will most likely have the agent visit maze regions that an expert would never visit and may even show the agent moving *away* from the goal rather than toward it. To ensure we generate expert augmented data, we can simply restrict our DAF to only sample new positions near the shortest path to the goal (green region) for which the agent's displacement is closely aligned with the shortest path (orange arrows). This approach shifts the burden from the user having to demonstrate optimal actions to the user simply having to understand when augmented data represent progress toward task completion. In the next section, we describe the DAFs we use and the sampling procedures we define to generate augmented data that shows task progress.

4.2 Implementation

GuDA's sampling procedures are domain-specific and depend on which DAFs are available as well as what task progress looks like in a given domain. In this work, we consider four DAFs that transform an input trajectory segment τ using invariances and symmetries common to many physical tasks:

1. **`Translate`**$(\tau; \mathcal{P})$**:** Since the dynamics of agents and objects are often independent of their position, we can translate them to a new position (x, y) sampled from a distribution $\mathcal{P}$.

2. **`Rotate`**$(\tau; \Theta)$**:** Since the dynamics of agents and objects are often independent of their orientation, we can rotate the direction the agent and/or object faces by an angle θ sampled from a distribution Θ to produce motion in a different direction.

3. **`Reflect`**$(\tau; \mathcal{R})$**:** An agent that moves to the left often produces a mirror image of an agent moving to the right, so we can reflect the agent's left-right motion with probability $\mathcal{R}(\tau)$.

4. **`RelabelGoal`**$(\tau; \mathcal{G})$**:** In goal-conditioned tasks, dynamics are generally independent of the desired goal state (Andrychowicz et al., 2017). Thus, we can replace the true goal with a new goal $\boldsymbol{g}$ sampled from the task's goal distribution $\mathcal{G}$.

We focus on navigation and manipulation tasks which have intuitive notions of task progress: an agent makes progress if it moves closer to a goal position (navigation) or if it moves an object closer to a goal position (manipulation). Given these notions of task progress, the user must specify how to apply these DAFs to generate expert-quality augmented data. Formally, the user specifies distributions over translations $\mathcal{P}(x, y|\tau)$, rotations $\Theta(\theta|\tau)$, and/or reflections $\mathcal{R}(\tau)$ that produce data showing task progress. To provide a concrete example of one such distribution, we return to the quadruped maze example in Fig. 2. A human can easily identify task-relevant maze positions (x, y) (green region) and a near-optimal displacement directions $\theta^*(x, y)$ for these positions (orange arrows). Thus, to generate expert-quality augmented data using only the `Translate` DAF, we can sample new positions from $\mathcal{P}(x, y|\tau) = \mathsf{Unif}(\{(x, y) : |\theta(\tau) - \theta^*(x, y)| \leq \frac{\pi}{4}$ and (x, y) is within the green region$\})$, a uniform distribution over task-relevant maze positions for which the agent's original displacement

Algorithm 1: Guided Data Augmentation

$\mathcal{G} \leftarrow$ distribution over task-relevant goals.
$\mathcal{P}(x, y|\tau) \leftarrow$ distribution over task-relevant positions for trajectory segment τ.
$\Theta(\theta|\tau) \leftarrow$ distribution over task-relevant rotation angles for trajectory segment τ.
$\mathcal{R}(\tau) \leftarrow$ probability of reflecting τ.

```
function GuidedDAF(τ_0)
    τ ← copy(τ_0)
    τ ← RelabelGoal(τ; G)
    τ ← Translate(τ; P(x, y|τ))
    τ ← Reflect(τ; R(τ))
    τ ← Rotate(τ; Θ(θ|τ))
    for (s, a, r, s') ∈ τ do
        r ← r(s, a) // Recompute rewards
    return τ
```

Task Name	Initial Dataset Size	GuDA Sampling Procedures (τ = input trajectory segment)
maze2d-umaze maze2d-medium maze2d-large	5 trajectories 5 trajectories 5 trajectories	`Translate` a partial trajectory τ to a random maze position, and then `Rotate` τ such that the agent moves along the shortest path to the goal. See Fig. 4a.
antmaze-umaze antmaze-medium antmaze-large	1 trajectory 2 trajectories 5 trajectories	`Translate` a partial trajectory τ to a random maze position such that the agent moves along the shortest path to the goal. See Fig. 2.
parking	10 trajectories	Here, τ is a *full trajectory*. First, use `RelabelGoal` to change τ's goal to randomly sampled goal (parking spot). Then, `Translate` τ such that the agent's final position is at the goal, and `Rotate` τ such that the car is within the parking spot. See Fig. 1.
soccer-sim	3 trajectories	Here, τ is a *full trajectory*. `Reflect` τ with probability 0.5, `Translate` τ such that the ball's final position is at the goal, and then `Rotate` τ randomly. See Fig. 4b.
soccer-physical	1 trajectory	See Section 5.2.

Table 1: GuDA sampling procedures for tasks in our empirical analysis. We provide task descriptions in Appendix A and describe how we implement these sampling procedures in Appendix B.

direction $\theta(\tau)$ is closely aligned with $\theta^*(x, y)$. If $\mathcal{P}(x, y|\tau)$, $\Theta(\theta|\tau)$, and $\mathcal{R}(\tau)$ are uniform distributions independent of τ over all valid position, rotations, and reflections, then GuDA reduces to the standard DA strategy that samples augmented data uniformly at random.

Algorithm 1 provides pseudocode for our implementation of GuDA assuming access to all four DAFs.[2] Table 1 describes high-level sampling procedures for tasks in our empirical analysis: D4RL maze2d and antmaze navigation tasks (Fu et al., 2020), a parking task (Leurent, 2018), a simulated robot soccer task, and a physical robot soccer task. Assuming access to all DAFs, the sampling procedures generally proceed as follows. First, we randomly sample a new goal. If τ is a full trajectory, we `Translate` τ such that the agent or object's final position is at the goal, and then `Reflect` and/or `Rotate` τ randomly about the goal. If τ is a partial trajectory, we `Translate` τ to a new position that would likely be observed by an expert policy, and then `Reflect` and/or `Rotate` τ so that the agent or object moves as close as possible to the goal. We provide task descriptions in Appendix A and a more formal description of our sampling procedures in Appendix B.

5 Experiments

We design an empirical study to evaluate two core hypotheses:

H1: GuDA enables learning from a small dataset of potentially suboptimal data.

H2: GuDA yields larger returns than a random DA strategy.

H1 implies that GuDA is well-suited to offline learning for real-world tasks where expert data is often scarce, and **H2** emphasizes the importance of sampling expert-quality augmented data. We note that support for **H2** implicitly provides support for **H1**.

5.1 Simulated Experiments

We first evaluate GuDA on simulated tasks described in Table 1. In all tasks, we start with a small initial dataset containing at least one successful – though not necessarily expert-level – trajectory

[2]GuDA can be implemented in many different ways and can be adapted depending on which DAFs are available. For instance, it is possible to guide DA by applying a subset of these four DAFs in a different order.

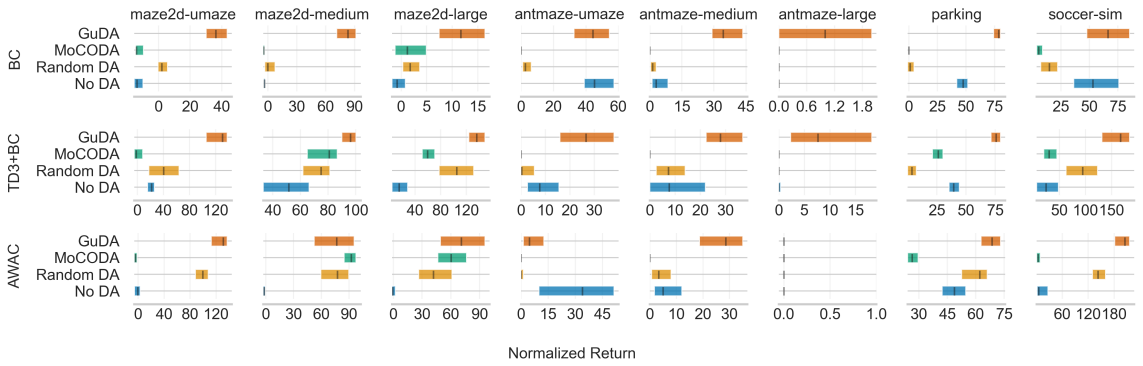
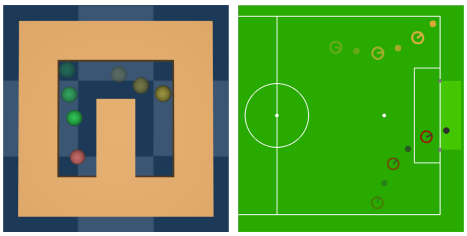


Figure 3: IQM normalized returns over 10 independent runs with 95% stratified bootstrap confidence intervals for different DA strategies and algorithms. We compute normalized returns computed as $= 100 \cdot \frac{R - R_{\text{random}}}{R_{\text{expert}} - R_{\text{random}}}$ where R_{expert} and R_{random} denote the average return of the demonstrator and a policy that chooses actions uniformly at random, respectively, computed over 100 trajectories.

(Table 1). These datasets contain failures and suboptimal behaviors as well: maze2d datasets contain data in which the agent moves away from the goal, soccer datasets contain trajectories where the agent kicks the ball out of bounds, and parking datasets contain trajectories where the car fails to park. For maze2d and antmaze tasks, we hand-pick a small number of trajectory segments from the original '-v1' and '-diverse-v1' D4RL datasets, respectively. For the remaining tasks, we use pre-trained policies to generate datasets. Dataset visualizations can be found in Fig. 7 of Appendix A.

We consider three baselines: the model-based DA strategy MoCoDA (Pitis et al., 2022), a DA strategy that randomly samples augmented data (Random DA), and no augmentation (No DA). MoCoDA is a well-suited model-based baseline for our experiments; it exploits causal independence in the task's dynamics to efficiently learn a dynamics model that generalizes outside of the support of the dataset, which is particularly important when data is scarce. To improve the quality of MoCoDA data, we sample augmented states from a *parent distribution* that closely matches the distribution of augmented states under GuDA.[3] We provide further details on how we apply MoCoDA to each task in Appendix C. With each DA strategy, we generate 1 million augmented transitions and then perform offline learning with BC, TD3+BC (Fujimoto & Gu, 2021), and AWAC (Nair et al., 2020) for 1 million policy updates. We tune hyperparameters for each algorithm and DA strategy separately using a hyperparameter sweep described in Appendix D. We report the inter-quartile mean (IQM) return with 95% bootstrap confidence intervals over 10 independent runs.[4]



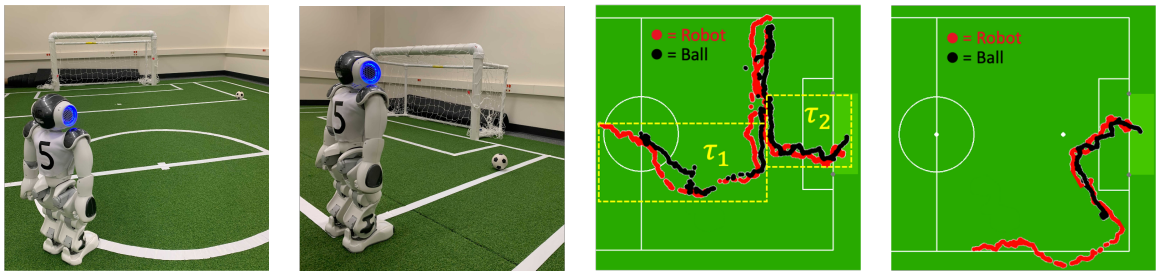
(a) maze2d (b) soccer-sim

Figure 4: Example augmentations under GuDA. The original trajectory segment is shown in yellow.

Fig. 3 shows IQM normalized returns for each algorithm in each task. GuDA almost always outperforms all baselines – and often by a large margin (supporting **H1**). For instance, in antmaze-medium, GuDA yields returns 3x larger than the next best strategy for all algorithms. GuDA with TD3+BC is also the only strategy that can solve antmaze-large with significance. Moreover, we emphasize that BC often achieves much larger returns with GuDA than with Random DA or MoCoDA, indicating that GuDA indeed generates expert data. MoCoDA is unable to solve the more complex antmaze, parking, and soccer-sim tasks because it does not have enough data to learn an accurate, generalizable dynamics model, emphasizing the utility of GuDA in data-scarce settings.

[3]Since we cannot identify expert state-action pairs, we only specify a parent distribution over task-relevant states.

[4]We choose to report the IQM because it is less biased and more statistically efficient than the median, and it is more robust to outliers than the mean (Agarwal et al., 2021).



(a) Easy initialization (b) Hard initialization (c) Initial dataset (d) Example GuDA data

Figure 5: (5a, 5b) Task initializations. (5c) Initial data with relevant segments τ_1 and τ_2. (5d) An illustration of GuDA data generated by translating, rotating, and/or reflecting τ_1 and τ_2.

While Random DA is often beneficial in maze2d and soccer-sim tasks, it often performs *worse* than No DA in other tasks. For instance, Random DA harms performance with all algorithms in antmaze-umaze, with BC and AWAC in antmaze-medium, and with BC and TD3+BC in parking. Since BC mimics the provided data, it is understandable that Random DA may harm performance with BC. However, since offline RL algorithms can learn from suboptimal data, these findings emphasize the importance of generating expert augmented data even for offline RL (supporting **H2**).

5.2 Physical Experiments

We further evaluate GuDA in a physical robot soccer task in which a NAO V6 robot must dribble a ball to the goal from the Easy and Hard initializations shown in Fig. 5a and 5b. The agent observes its position and orientation as well as the ball's position using vision-based state estimation. The ball's dynamics depend on how the robot's feet contact the ball, and since foot positions are not observed, the ball's dynamics appear highly stochastic to the agent. This stochasticity coupled with noisy state estimation makes this task notably difficult. We collect data using a policy pre-trained in a low-fidelity soccer simulator with simplified dynamics and perfect state estimation (soccer-sim, Fig. 4b). Our dataset contains a single physical trajectory of the agent dribbling the ball from the center of the field to the goal (Fig. 5c). This data is highly suboptimal for two reasons: (1) we trained the demonstrator in a low-fidelity simulator, and (2) the robot fumbled the ball and had to take an indirect route to the goal.

Method	Easy	Hard
GuDA	8/10	7/10
MoCoDA	0/10	0/10
Random DA	4/10	0/10
No DA	4/10	0/10
Demonstrator	9/10	2/10

Table 2: Success rates for our physical robot soccer experiments.

To apply GuDA, we first identify two task-relevant behaviors in our initial dataset (Fig. 5c): the robot executing a tight turn to the ball (τ_1), and the robot scoring with the ball away from the sideline (τ_2). We then define a sampling procedure to generate augmented trajectories that trace out the path an expert might take to successfully score (Fig. 5d): we `Translate` and `Rotate` τ_1 to demonstrate the agent approaching the ball at a favorable angle, and then we `Translate`, `Rotate`, and `Reflect` τ_2 to demonstrate the agent scoring with the ball away from the sideline.

We generate 1 million augmented samples using GuDA, MoCoDA, and Random DA, and we train agents using IQL (Kostrikov et al., 2021) for 1 million policy updates. We also compare agents to the demonstrator we used to collect our physical trajectory. Table 2 and Fig. 6 show the success rate and IQM time to score for each agent over 10 attempts at each initialization. With the Easy initialization, GuDA scores faster and more frequently than MoCoDA, Random DA, and No DA. GuDA and the demonstrator policy have similar success rates, but GuDA scores significantly faster than the demonstrator as well. We attribute this speedup to how the GuDA policy trained on augmented data that matches the physical world's dynamics (since our DAFs are dynamics-invariant) whereas our demonstrator policy trained in a low-fidelity simulator. With the Hard initialization,

only the GuDA agent can consistently score; MoCoDA, Random DA, No DA policies always kick the ball out of bounds. Even the demonstrator policy almost always fails. Our results show that GuDA not only outperforms MoCoDA and Random DA (**H2**) but also enables an agent to surpass its demonstrator in a difficult physical task with just a single suboptimal trajectory (**H1**).

6 Conclusion

In this work, we introduced Guided Data Augmentation (GuDA), a human-guided data augmentation (DA) framework that generates expert-quality augmented data without the expense of real-world task interaction. In GuDA, a user imposes a series of simple rules on the DA process to automatically generate augmented samples that approximate expert behavior. GuDA serves as a intuitive way to integrate human expertise into offline RL; instead of requiring that an expert demonstrate a near-optimal sequence of actions to solve a task, GuDA simply requires the user to understand what augmented data represents progress toward task completion. Empirically, we demonstrate that GuDA outperforms a widely-applied random DA strategy as well as a model-based DA strategy and enables offline learning from a limited set of potentially suboptimal data. Furthermore, we show how GuDA yields an effective policy in a physical robot soccer task when given a single highly suboptimal trajectory. Our findings emphasize how a more intentional approach to DA can yield substantial performance gains.

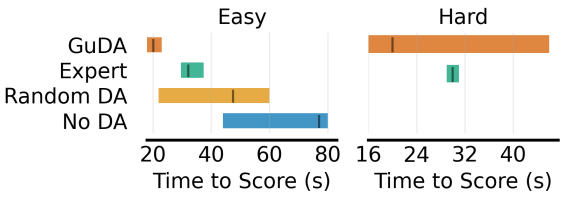


Figure 6: IQM time to score over 10 attempts with 95% stratified bootstrap confidence intervals. Lower times are better. GuDA's confidence interval in Hard is wide because of a single trial in which the agent scored after an unusually hard kick moved the ball to the opposite end of the field. Since MoCoDA failed to score in both tasks, we exclude it from this figure.

The core limitation of GuDA is that it requires domain knowledge to specify sampling procedures. Since the sampling procedures required to generate expert augmented data are task dependent, GuDA must be implemented separately for each task. In many navigation and object manipulation tasks, these rules can be derived from basic intuitions on what task progress looks like and are simple to implement. However, GuDA is less applicable to tasks in which it is difficult to assess the quality of a trajectory segment (*e.g.* chess). While our empirical analysis focuses on offline RL and behavior cloning, GuDA can in principle be applied to other learning methods – both offline and online. Future work should study how GuDA interacts with other learning methods such as inverse RL and online RL. Furthermore, a broader analysis investigating the the most effective way to integrate augmented data into offline RL – similar to the analysis of Corrado & Hanna (2024) for online RL – would further strengthen the effectiveness of GuDA as well as other DA techniques.

Broader Impact Statement

Our work focuses on fundamental RL research, and we thus see no direct negative societal consequences. In this work, we propose a data augmentation framework (GuDA) that generates expert-quality augmented data and improves the performance of offline RL and behavior cloning methods. Since GuDA outperforms existing data augmentation methods on both simulated and physical tasks and yields effective policies even when given a small amount of suboptimal data, it can be applied to real-world tasks (where expert data is often scarce) and positively impact society.

References

Farzad Abdolhosseini, Hung Yu Ling, Zhaoming Xie, Xue Bin Peng, and Michiel Van de Panne. On learning symmetric locomotion. In *Motion, Interaction and Games*, pp. 1–10. 2019.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Daesol Cho, Dongseok Shim, and H. Jin Kim. S2p: State-conditioned image synthesis for data augmentation in offline reinforcement learning. *Advances in Neural Information Processing Systems*, 2022.

Nicholas E. Corrado and Josiah P. Hanna. Understanding when dynamics-invariant data augmentations benefit model-free reinforcement learning updates. In *International Conference on Learning Representations (ICLR)*, 2024.

Meng Fang, Cheng Zhou, Bei Shi, Boqing Gong, Jia Xu, and Tong Zhang. Dher: Hindsight experience replay for dynamic goals. In *International Conference on Learning Representations*, 2018.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Abraham George, Alison Bartsch, and Amir Barati Farimani. Minimizing human assistance: Augmenting a single demonstration for deep reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5027–5033, 2023. doi: 10.1109/ICRA48891.2023.10161119.

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*, pp. 3682–3691. PMLR, 2021.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International conference on machine learning*, pp. 2829–2838. PMLR, 2016.

L. Guan, Mudit Verma, Sihang Guo, Ruohan Zhang, and Subbarao Kambhampati. Widening the pipeline in human-guided reinforcement learning with explanation and context-aware data augmentation. In *Neural Information Processing Systems*, 2020. URL https://api.semanticscholar.org/CorpusID:238227077.

L. Guan, Mudit Verma, Sihang Guo, Ruohan Zhang, and Subbarao Kambhampati. Widening the pipeline in human-guided reinforcement learning with explanation and context-aware data augmentation. In *NeurIPS*, 2021.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259, 2020.

Jungwoo Han and Jinwhan Kim. Selective data augmentation for improving the performance of offline reinforcement learning. In *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*, pp. 222–226, 2022. doi: 10.23919/ICCAS55662.2022.10003747.

Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13611–13617. IEEE, 2021.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. *Advances in Neural Information Processing Systems*, 34: 3680–3693, 2021.

Charles A Hepburn and Giovanni Montana. Model-based trajectory stitching for improved offline reinforcement learning. *arXiv preprint arXiv:2211.11603*, 2022.

Ho-Taek Joo, In-Chang Baek, and Kyung-Joong Kim. A swapping target q-value technique for data augmentation in offline reinforcement learning. *IEEE Access*, 10:57369–57382, 2022. doi: 10.1109/ACCESS.2022.3178194.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *ArXiv*, abs/2110.06169, 2021.

Aviral Kumar, Justin Fu, G. Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Neural Information Processing Systems*, 2019. URL https://api.semanticscholar.org/CorpusID:173990380.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. Should I run offline reinforcement learning or behavioral cloning? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=AP1MKT37rJ.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020.

Edouard Leurent. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

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.

C. Lu, B. Huang, K. Wang, J. M. Hernández-Lobato, K. Zhang, and B. Schölkopf. Sample-efficient reinforcement learning via counterfactual-based data augmentation. In *Offline Reinforcement Learning - Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Sergey Kolesnikov Mikhail Pavlov and Sergey M. Plis. Run, skeleton, run: skeletal model in a physics-based simulation. *AAAI Spring Symposium Series*, 2018.

Peter Mitrano and Dmitry Berenson. Data augmentation for manipulation. *arXiv preprint arXiv:2205.02886*, 2022.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810. IEEE, 2018.

Silviu Pitis, Elliot Creager, and Animesh Garg. Counterfactual data augmentation using locally factored dynamics. *Advances in Neural Information Processing Systems*, 33:3976–3990, 2020.

Silviu Pitis, Elliot Creager, Ajay Mandlekar, and Animesh Garg. Mocoda: Model-based counterfactual data augmentation. *Advances in Neural Information Processing Systems*, 35:18143–18156, 2022.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. Efficient differentiable simulation of articulated bodies. In *International Conference on Machine Learning*, pp. 8661–8671. PMLR, 2021.

Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 34:5402–5415, 2021.

Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

Samarth Sinha, Ajay Mandlekar, and Animesh Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 907–917. PMLR, 08–11 Nov 2022a. URL https://proceedings.mlr.press/v164/sinha22a.html.

Samarth Sinha, Ajay Mandlekar, and Animesh Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 907–917. PMLR, 08–11 Nov 2022b.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pp. 216–224. Elsevier, 1990.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Arun Venkatraman, Roberto Capobianco, Lerrel Pinto, Martial Hebert, Daniele Nardi, and J Andrew Bagnell. Improved learning of dynamics models for control. In *International Symposium on Experimental Robotics*, pp. 703–713. Springer, 2016.

Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. *Advances in Neural Information Processing Systems*, 33: 7968–7978, 2020.

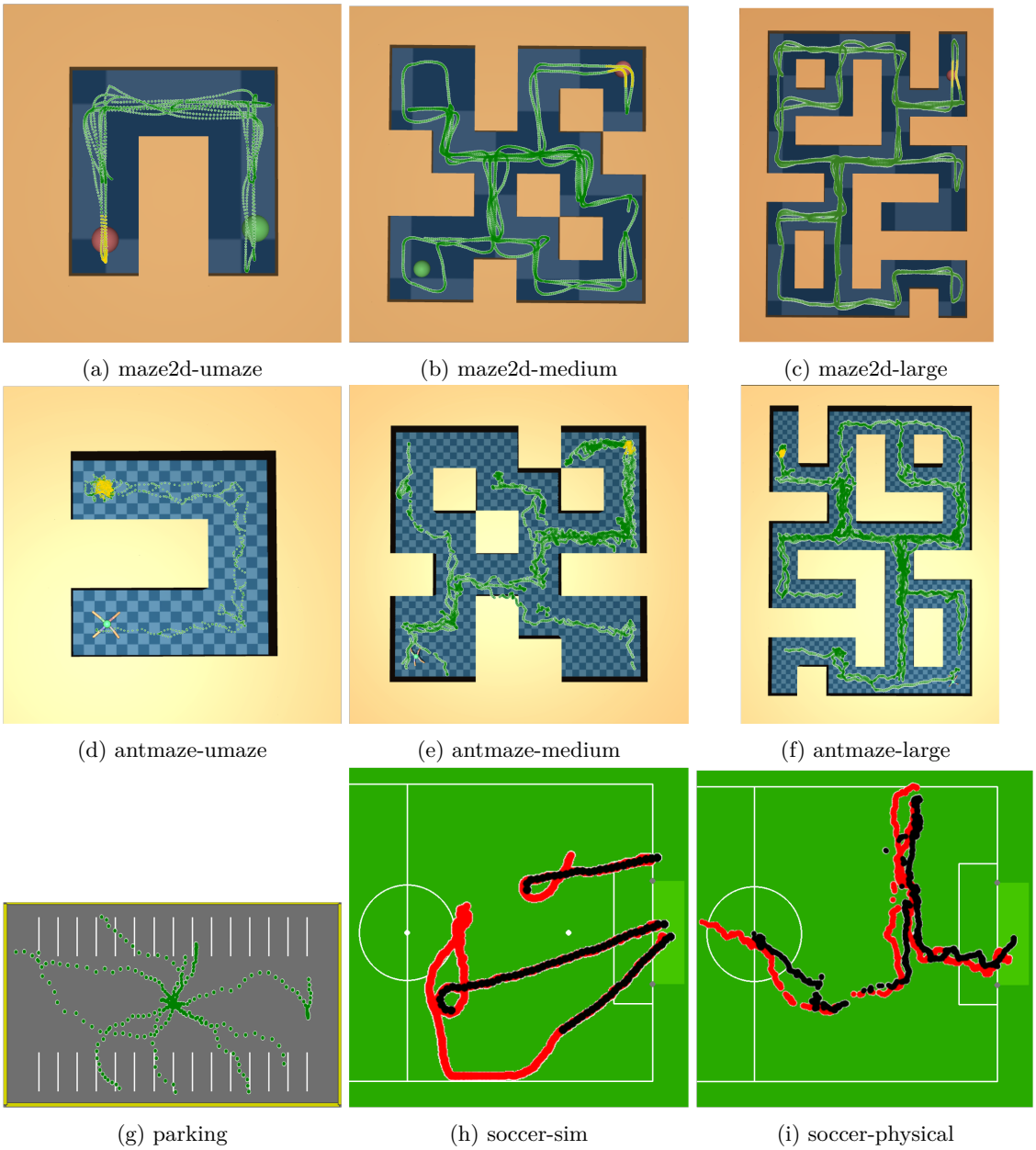
Kerong Wang, Hanye Zhao, Xufang Luo, Kan Ren, Weinan Zhang, and Dongsheng Li. Bootstrapped transformer for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 2022.

Matthias Weissenbacher, Samarth Sinha, Animesh Garg, and Kawahara Yoshinobu. Koopman q-learning: Offline reinforcement learning via symmetries of dynamics. In *International Conference on Machine Learning*, pp. 23645–23667. PMLR, 2022.

Haoran Xu, Xianyuan Zhan, Jianxiong Li, and Honglei Yin. Offline reinforcement learning with soft behavior regularization. *arXiv preprint arXiv:2110.07395*, 2021.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, pp. 1719–1735. PMLR, 2021.



(a) maze2d-umaze (b) maze2d-medium (c) maze2d-large

(d) antmaze-umaze (e) antmaze-medium (f) antmaze-large

(g) parking (h) soccer-sim (i) soccer-physical

Figure 7: Visualizations of initial datasets. In maze2d and antmaze tasks, gold indicate data points for which the agent receives a nonzero reward. In soccer-sim and soccer-physical tasks, red denotes the agent and black denotes the ball.

A Task Descriptions

In this section, we describe each task in our empirical analysis. Fig. 7 visualizes the initial datasets used in each task.

A.1 Maze2d

A force-actuated point-mass must navigate to a fixed goal from a random initial position. The agent observes its position (x, y) and velocity (v_x, v_y), and the agent's actions take the form $\boldsymbol{a} = (f_x, f_y)$

where f_x and f_y are linear forces applied to the agent in the x and y directions, respectively. The agent receives $+1$ reward for being in positions within a disk of radius 0.5 centered at the the goal and 0 reward otherwise.

A.2 Antmaze

This task is essentially the same as the maze2d task except the agent is replaced with a quadruped "ant". The agent must navigate to a fixed goal from a fixed initial maze position. The agent observes its position (x, y), its height above the ground z, its orientation expressed as a quaternion, as well as the angle and angular velocities or all eight of its joints. The agent's action consists of torques to apply to each of the agent's joint. The agent receives $+1$ reward for being in positions within a disk of radius 0.5 centered at the the goal and 0 reward otherwise.

A.3 Parking

An autonomous vehicle must park front-first into a designated parking spot. The agent observes it's current position (x, y), velocity (v_x, v_y), as well as the sin and cos of its heading θ (*i.e.* the direction the front of the car is facing). The agent's state is thus

$$\boldsymbol{s}_{\text{agent}} = (x, y, v_x, v_y, \sin\theta, \cos\theta).$$

The agent also observes a goal $\boldsymbol{g}$ consisting of the (x_g, y_g) position of the parking spot, the desired velocity $(v_{g,x}, v_{g,y})$ at the parking spot (which is always set to $(0, 0)$), as well as the sine and cosine of car's desired heading θ_g at the parking spot (which is either $\theta_g = +\pi/2$ or $\theta_g = -\pi/2$) :

$$\boldsymbol{g} = (x_g, y_g, 0, 0, \sin\theta_g, \cos\theta_g).$$

The full state is $\boldsymbol{s} = (\boldsymbol{s}_{\text{agent}}, \boldsymbol{g})$. The agent selects actions $\boldsymbol{a} = (a_{\text{acc}}, a_{\text{steer}})$ where a_{acc} is the agent's acceleration in direction θ and a_{steer} controls the agent's change in direction.

The agent receives a dense reward based on its distance to the parking spot and how closely the car aligns with the spot. If the agent crashes into one of the walls surrounding the parking lot, it receives a -5 reward penalty:

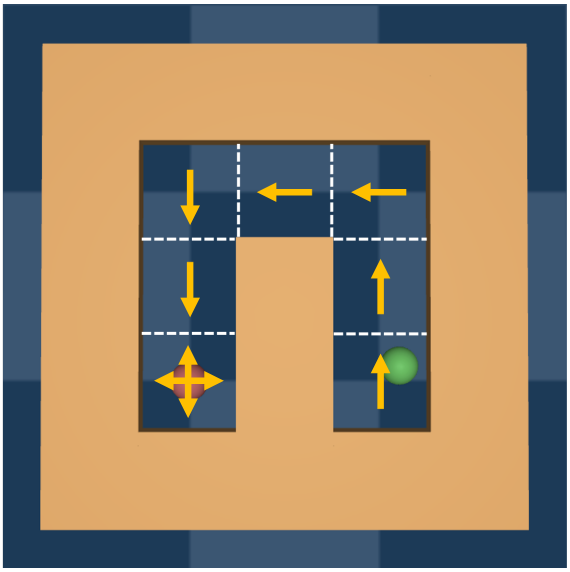
$$r = -\sqrt{|\boldsymbol{s} - \boldsymbol{g}|} - 5 \cdot \mathbb{1}_{\text{crash}} \tag{1}$$

A.4 Soccer-sim

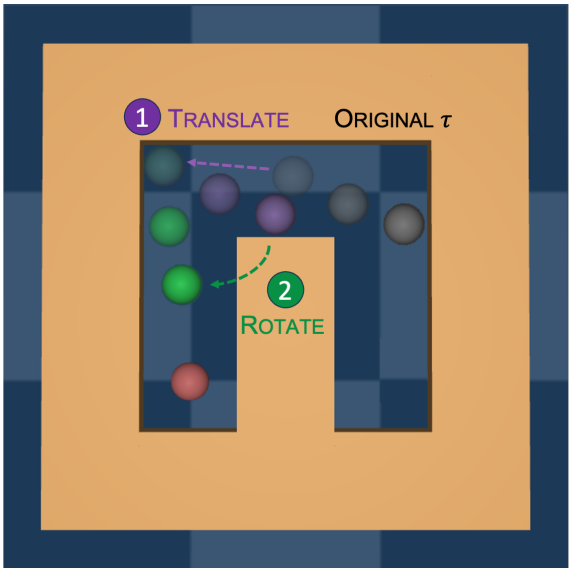
A robot (agent) must kick a ball to a fixed goal location. Robot and ball positions are initialized uniformly at random across the entire field. The observation contains the the following features:

- $(x_{\text{robot to ball}}, y_{\text{robot to ball}}) = (x_{\text{robot}} - x_{\text{ball}}, y_{\text{robot}} - y_{\text{ball}})$, the vector difference between the robot and ball positions.
- $(x_{\text{ball to goal}}, y_{\text{ball to goal}}) = (x_{\text{ball}} - x_{\text{goal}}, y_{\text{ball}} - y_{\text{goal}})$, the vector difference between the ball and goal positions.
- $(\sin(\theta_{\text{robot to ball}}), \cos(\theta_{\text{robot to ball}}))$, where $\theta_{\text{robot to ball}}$ denotes the angle between the direction the robot is facing and the ball.
- $(\sin(\theta_{\text{ball to goal}}), \cos(\theta_{\text{ball to goal}}))$, where $\theta_{\text{ball to goal}}$ denotes the angle between the ball and the goal.

The action $\boldsymbol{a} = (a_\theta, a_x, a_y)$ has three components: a_θ rotates the direction the robot is facing, and (a_x, a_y) controls the robot's change in (x, y) position. The agent receives reward based on its distance to the ball and the ball's distance to the goal:



(a) An illustration of the near-optimal displacement directions $\theta^*(\boldsymbol{x})$ in the maze2d-umaze-v1 task. Near the goal, (red ball), we sample rotation angles uniformly at random.



(b) An illustration of the near-optimal displacement directions $\theta^*(\boldsymbol{x})$ in the maze2d-umaze-v1 task. Near the goal, (red ball), we sample rotation angles uniformly at random.

$$r = \begin{cases} \frac{0.9}{d_{\text{agent to ball}}} + \frac{0.1}{d_{\text{ball to goal}}} + \mathbb{1}_{\text{ball at goal}}, & \text{if the agent is facing the ball} \\ \mathbb{1}_{\text{ball at goal}}, & \text{if the agent is not facing the ball} \end{cases} \tag{2}$$

where $d_{\text{agent to ball}}$ is the Euclidean distance between the agent and the ball, $d_{\text{ball to goal}}$ is the Euclidean distance between the ball and the goal, and $\mathbb{1}_{\text{ball at goal}}$ is an indicator function that returns 1 when ball is at the goal and 0 otherwise. We say the agent is facing the ball if $|\theta_{\text{robot to ball}}| < 30°$

A.5 Soccer-physical

An agent must kick a ball to a fixed goal location. Agent and ball positions are initialized as shown in Fig. 5a and Fig. 5b. The agent receives reward based on its distance to the ball and the ball's distance to the goal. This task uses the same observation space, action space, and reward function used in the soccer-sim task.

B Sampling Procedures

In this appendix, we provide a formal description of the sampling procedures we use to guide DA in our empirical analysis. More concretely, we define distributions over translations $\mathcal{P}(x, y|\tau)$, rotations $\Theta(\theta|\tau)$, and reflections $\mathcal{R}(\tau)$ for each task. We refer the reader to Table 1 for a high-level description of guided data augmentations for each task. In all descriptions, we let τ denote an input trajectory to be augmented.

B.1 Maze2d

In this task, we use the `Translate` and `Rotate` DAFs. Since the agent is initialized to a random position in the maze, an expert policy will visit all maze position. Thus, we let $\mathcal{P}(x, y|\tau)$ be a uniform distribution over all valid maze positions for all τ. We note that this distribution over maze positions is identical to the distribution used in a random DA strategy.

Before describing how we sample rotation angles, we introduce a few relevant quantities. Let $(\Delta x, \Delta y)$ denote τ's displacement (the difference between τ's final and initial positions), and let $\theta(\tau) = \arctan \frac{\Delta y}{\Delta x}$ be the displacement angle of the original trajectory segment. A user can easily compute $\theta(\tau)$ given τ. Additionally, let $\theta^*(x, y)$ be a function returning a near-optimal displacement direction at position (x, y). Fig. 8a illustrates how we define $\theta^*(\tau)$ in the maze2d-umaze-v1 task. We divide the maze into cells (indicated by white dashed lines) and then label each cell with a desired displacement direction (arrows). Note that for the cell containing the goal, have no preferred displacement direction.

After translating the agent, we sample rotation angles for `Rotate` from $\Theta(\theta|\tau) = (\theta^*(x, y) + \varepsilon) - \theta(\tau)$ where (x, y) is the agents initial position in τ, and ε is a noise parameter distributed according to $\mathsf{Unif}([-\pi/6, +\pi/6])$. We add noise to the rotation angle because offline RL methods learn to follow expert trajectories more effectively when expert data is noisy (Kumar et al., 2022). Intuitively, $(\theta^*(x, y) + \varepsilon) - \theta(\tau)$ is the rotation angle required to rotate the agent's displacement direction from $\theta(\tau)$ to $(\theta^*(x, y) + \varepsilon)$. A visualization of this sampling procedure can be found in Fig. 8b.

B.2 Antmaze

In this task, we use the `Translate` DAF. Since the agent is initialized to a *fixed* position in the maze, an expert policy will only visit a subset of maze position near the optimal path toward the goal. Thus, using the same notation established in the previous section for maze2d, we sample new positions from $\mathcal{P}(x, y|\tau) = \mathsf{Unif}(\{(x, y) : |\theta(\tau) - \theta^*(x, y)| < \frac{\pi}{4}, (x, y)$ is near the optimal path to the goal$\})$, a uniform distribution of task-relevant maze positions for which the agent's original displacement angle is within $\pi/4$ of the optimal displacement angle. Similar to maze2d, we define $\theta^*(x, y)$ by dividing the maze into cells and then labeling each cell with a desired displacement angle. We then fetch all cells whose optimal displacement angle closely aligns with the agent's original displacement, and then randomly sample a new position from one of these cells.

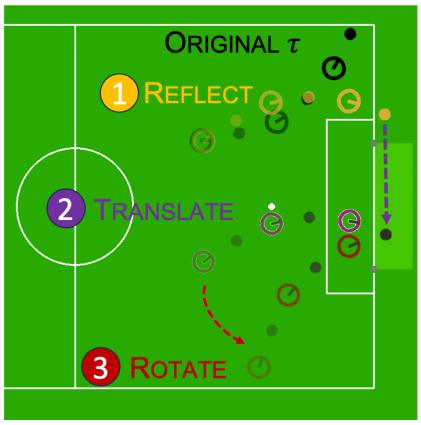


Figure 9: Visualization of the sampling procedure we use in soccer-sim.

B.3 Parking

In this task, we use the `Translate` and `Rotate` DAFs. We first sample a new goal $\boldsymbol{g}$ uniformly at random from $\mathcal{G}$, and then translate τ such that its final position is a $\boldsymbol{g} = (x_g, y_g)$, *i.e.*, $\mathcal{P}(x, y|\tau)$ places probability 1 on the goal's position (x_g, y_g).

After translating, we rotate the agent such that the car's heading at the final step in τ is closely aligned with the parking spot's heading θ_g. We sample rotation angles from $\Theta(\theta|\tau) = (\theta_g + \varepsilon) - \theta(\tau)$, where ε is a noise parameter distributed according to $\mathsf{Unif}([-\pi/6, +\pi/6])$, and θ_τ is heading angle of the last transitions in τ. Intuitively, $(\theta_g + \varepsilon) - \theta(\tau)$ is the rotation angle required to rotate agent's heading from $\theta(\tau)$ to $(\theta_g + \varepsilon)$.

B.4 Soccer-sim

In this task, we first reflect τ with probability $\mathcal{R}(\tau) = 0.5$. Then, we translate the τ so the ball's final position is at the goal. Thus, we sample a new position from $\mathcal{P}(x, y|\tau) = \mathsf{Unif}(\{(x, y) : (x, y)$ is inside the goal$\})$. Last, we rotate τ by a rotation angle sampled uniformly at random from while making sure the rotate trajectory remains in-bounds. A visualization of this sampling procedure can be found in Fig. 9.

Algorithm	Hyperparameter	Values
BC	MLP network hidden layers	$(64, 64), (256, 256)$
	learning rate	$10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$
TD3+BC (Fujimoto & Gu, 2021)	MLP network hidden layers	$(64, 64), (256, 256)$
	actor/critic learning rates	$10^{-3}, 10^{-4}, 10^{-5}$
	α	$2.5, 5, 7.5, 10$
AWAC (Nair et al., 2020)	MLP network hidden layers	$(64, 64), (256, 256)$
	actor/critic learning rates	$10^{-3}, 10^{-4}, 10^{-5}$
	inverse advantage weight λ	$0.5, 1, 2$ (and 0.1 for antmaze)
IQL (Kostrikov et al., 2021)	MLP network hidden layers	$(64, 64)$
	actor/critic learning rates	$10^{-4}, 10^{-5}, 10^{-6}$
	inverse temperature β	$1, 5, 10$
	expectile τ	$0.5, 0.7, 0.9$

Table 3: Hyperparameter values we considered for each algorithm.

C MoCoDA Baseline

In this section, we provide additional details regarding MoCoDA experiments. We use the author's original implementation pitis2022mocoda.

MoCoDA can in principle generate expert-quality augmented data if we specify a *parent distribution* $P(\boldsymbol{s}, \boldsymbol{a})$ that is distributed according to the $(\boldsymbol{s}, \boldsymbol{a})$ distribution an expert might observe, but doing so requires us to explicitly describe the distribution of expert actions. We do not have access to this information. However, it is nevertheless fairly simple to specify a distribution $P(\boldsymbol{s}, \cdot)$ over task-relevant states. In all tasks, we choose a parent distribution that is uniform over task-relevant agent positions, corresponding to MoCoDA-U in the original paper (Pitis et al., 2022). At the implementation level, MoCoDA-U fits a Gaussian mixture model $P_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{a})$ parameterized by $\boldsymbol{\theta}$ to the provided dataset (while exploiting causal independence to generalize beyond the dataset's support). Then, this $P_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{a})$ is reweighed to be uniform over agent positions and used as the parent distribution. We note that this choice of parent distribution closely aligns with our sampling procedures detailed in Appendix B, which sample transformations uniformly at random over a small subset of task-relevant positions.

D Hyperparameter Tuning

We tune all algorithms and DA strategies separately using a hyperparameter sweep over values listed in Table 3. In the main paper, we report the hyperparameters yielding the largest IQM return over 10 seeds.

Since it would be time-consuming and costly to evaluate all IQL hyperperameter settings in the physical robot soccer tasks, we first evaluate IQL hyperparameter settings in soccer-sim. We identify four IQL policies with the largest IQM return in soccer-sim, and then evaluate each of these four policies in the physical task. We report results for the IQL policy yielding the largest success rate in both the Easy and Hard initializations.