Leveraging Affordances for Accelerating Online Reinforcement Learning

Aman Mehra

CMU-RI-TR-24-45

July 15, 2024

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Prof. Jeff Schneider (*Chair*)
Prof. Katerina Fragkiadaki
Swaminathan Gurumurthy

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my parents and grandparents.

Abstract

A long-standing problem in online reinforcement learning (RL) is that of ensuring sample efficiency. This stems from the inability of a learning agent to explore environments efficiently. Most attempts at efficient exploration tackle this problem in a setting where no prior information is utilized to bootstrap learning, thus failing to leverage additional affordances that may be cheaply available at the time of learning. These could include expert demonstrations, simulators that can reset to arbitrary states and domain specific inductive biases. Such affordances are valuable resources that offer enormous potential to guide exploration and speed up learning. As such their use in facilitating accelerated exploration is under explored in existing literature.

Consequently, in this thesis, we study different ways to utilize such affordances to facilitate faster online learning. We first incorporate affordances into the output end of a policy by describing a method that re-parameterizes the action space of driving policies through Bezier curves to induce an inductive bias towards natural vehicular trajectories. This enables us to learn challenging driving behaviour over 12$\times$ faster than traditional instantaneous action spaces. Subsequently, we study how affordances influencing the input end of a policy can improve learning efficiency. When provided with an arbitrarily resettable simulator, we find that training with a suitable choice of an auxiliary start state distribution that may differ from the true start state distribution of the underlying Markov Decision Process can significantly improve sample efficiency. Using a notion of safety to inform the choice of this auxiliary distribution significantly accelerates learning. We empirical demonstrate the effectiveness of this approach on the suite of MuJoCo continuous control tasks and a hard-exploration sparse-reward navigation task.

Acknowledgments

This work would not have been possible without the help and support of a large number of people and I would like to take this opportunity to thank them all for the invaluable part they have played in my journey.

Firstly, I am grateful to my advisor, Jeff Schneider, for giving me an opportunity to explore a new field where I had little prior experience. I really appreciate the freedom and flexibility I was given to find my feet and discover for myself the problems I would most enjoy tackling in this domain. His emphasis on zooming out and looking at the big picture has helped me stay on track and it has improved my ability to ask the right questions and define the scope of the research problem. This academic atmosphere has helped me develop a taste for the kind of research I like and I am incredibly grateful for his role in helping develop these skills and perspectives.

I have also been very lucky to collaborate with Aditi Raghunathan. Working with her, I got first hand experience of what it takes to make a meaningful contribution at the cutting edge of a rapidly evolving field. I learnt to be comprehensive with experimentation and the importance of conveying findings and research impact in an appropriate manner. My biggest takeaway was from observing her derive deep insights from seemingly disparate experimental results. This is something that I am yet to fully develop but I have gotten a glimpse of what to strive for in this regard and I am very grateful to have this north star to work towards.

I also want to give a big shout-out to all the labmates that I have had the chance to interact with at Auton Lab and FORUM. I am incredibly grateful for the numerous technical discussions we have got to have over the past two years. These discussions have been very helpful and I have often left with new questions, new insights and a burning desire to explore a new topic that someone brought up! All of this has helped me progress my work further. One of the biggest things that I will miss about CMU is this fabulous intellectual environment that has been so instrumental in my professional growth. A special shout-out to Vedant, Xintong and Alex for not just being great labmates but even better friends. I will miss our road trips, complete with deep philosophical discussions and our weekly culinary escapades. I hope we continue to have more such adventures in the future!

Alongside all the professional growth that the last two years at CMU brought about, I have gotten to meet some of the most amazing people that I am lucky to call friends, starting off with the Machas - Bharath, Prachi and Jay! First a big thank you to Bharath, with his questionable taste in sweets, for being my partner in crime for all things food and entertainment, for always being willing to make time, and for being the calm and level-headed complement to my crazy! Prachi, with her authoritarian views on what constitutes good chaat, for being my go to person for deep conversations, for adding that bubbly energy that makes the Machas Machao, for being an inspiration in standing for yourself and for being the spontaneous complement to my plan-it-to-the-tee self. And lastly Jay, with his foolproof remedy for tough times - *lite be*, for being the fun-loving culinarily-challenged brother who has inspired me to be a fitter and more well rounded person! Also, a big shout-out to my old friend and roommate Saujas, for broadening my palette, surviving the daily culinary shocks that living with me entails and for being a good sport about everything from food, travel to random 3am chats before conference deadlines! A big thank you to Peya, for being an amazing friend. Your ability to plan and execute trips alongside the intense demands of work has always inspired me to be more proactive in being more organized with my work so that I can travel! Lastly, a massive shout out to Adi, Nikhil, Christian, Ani, Ravi, Srujan and all my other RI buddies for the fond memories. I will miss our weekend hikes, rafting adventure, road trips and numerous parties. Each of you have made these two years at CMU truly memorable and I leave Pittsburgh with a feeling that I am part of a brand new family that will stay connected for life!

Finally, a huge thank you to my family. Words cannot express my gratitude. Everything that I have achieved, not just at CMU, but from day I was born is all because you have been there with me for the past two and a half decades! A special mention to my family in Pittsburgh who helped make the move to a new country feel completely seamless and ensured that I never felt homesick these past two years!

Funding

This material is based upon work supported by the U.S. Army Research Office and the U.S. Army Futures Command under Contracts W519TC-23-C-0031 and W519TC-23-C-0030 and by Stack AV.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Online reinforcement learning algorithms facilitate learning general behaviors without inductive biases and domain expertise through trial and error. By learning from environmental interaction such methods hold the potential to exceed the performance of supervised learning alternatives, reaching superhuman levels of performance on tasks such as Atari [23], Chess [33] and Go [32]. Despite such successes, these algorithms find it challenging to explore environments efficiently, resulting in long training times [8, 26, 35].

There has been a considerable amount of work on making online RL more efficient by promoting exploratory behaviors that are novelty-seeking [26] and state space-covering [10, 14, 31]. Although such approaches have the potential to learn robust policies, the lack of task-informed exploratory cues [22] and a tendency to forget how to revisit promising exploration frontiers [8] make them inefficient at learning to solve hard-exploration tasks. Moreover, these methods have been designed to improve exploration efficiency in the absence of any other prior information. Consequently, when affordances such as expert demonstrations, simulators with arbitrary reset conditions and domain specific inductive biases are available, these approaches fail to adequately leverage these additional resources to accelerate exploration.

On the other extreme, methods like imitation learning [9, 12] and offline RL [18, 19] can learn task-specific behavior from purely offline data. These methods perform well within the distribution of the training data but fail to be robust in an out-of-distribution (OOD) setting, making them unsuitable for use in the real world.

Hybrid Reinforcement Learning approaches mix offline data with online interactions to bridge this gap and learn robust policies efficiently. Bootstrapping online training with offline data isn't straightforward and naively finetuning a policy learned offline leads to sub-optimal performance [37]. In particular, offline experience can be quickly forgotten during online training if not handled appropriately [35, 37]. Successful hybrid methods ensure the persistence of information acquired offline during online training through various methods. Broadly they fall into two categories. The first category of methods utilize offline data to alter the output of a policy by learning action abstractions [34] to constrain the action space of the policy and thereby reduce the complexity of the search problem. Another line of approaches devise interventions at the input stage by either freezing a part of the replay buffer or learning fixed reference policies using the offline data to serve as manner of generating a curriculum online.

In this thesis, we revisit this hybrid RL setup and conduct two studies. In Chapter 4 we study interventions at the output end of policies by investigating action space abstractions to facilitate efficient learning of driving policies. We observe that navigation problems possess natural inductive biases with regards to what feasible trajectories look like and these inductive biases can be modelled without needing any offline data. Consequently, we show that reparameterizing an instantaneous control action space with a hand crafted temporally abstract trajectory representation improves the sample efficiency of learning. We empirically demonstrate this on the CARLA [7] simulator through a challenging overtaking scenario, where using Bezier curves to achieve such a reparameterization brings about a 12× improvement in sample efficiency.

In Chapter 5 we investigate hybrid interventions at the input end of a policy. We show that we can considerably accelerate online learning in a setting where the environment can be reset to arbitrary states by using expert offline data to construct auxiliary start state distributions. We find that using a notion of safety, approximated via episode length information is crucial for forming auxiliary start state distributions that accelerate training. Unlike other hybrid approaches, the proposed approach requires access to just the states from expert demonstration data. Access to an arbitrarily resetable simulator enables relaxing the need to access expert rewards and actions. This makes it much easier to collect data offline. We demonstrate

the efficacy of this approach by presenting empirical results on continuous control robotic tasks from MuJoCo [36] where the improved explorations allows us to obtain reward that matches or exceeds state-of-the-art performance. Additionally, we use a hard-exploration sparse-reward navigation task to showcase that training with a suitable choice of auxiliary start state distribution enables learning policies that are more robust to distribution shifts induced by evaluating at out-of-distribution (OOD) start states.

Chapter 2

Related Work

We explore related literature in this space through three broad category of methods: i) purely online RL, ii) purely offline learning and iii) hybrid RL methods.

2.1 Exploration in purely online RL

Exploration is an age-old problem in reinforcement learning that is crucial to building efficient and effective algorithms. In order to effectively explore, an algorithm must be adept at doing four kinds of tasks - i) choosing where to explore from, i.e., choosing an exploration frontier, ii) efficiently navigating to an exploration frontier, iii) choosing an exploration target to explore towards after arriving at an exploration frontier and iv) navigating efficiently to an exploration target.

In current literature, there exist many methods that aim to incentivize exploration but do not explicitly optimize for these properties. Several methods treat exploration as incidental to the central goal of reward maximization. These include methods that inject additive noise to the actions [30] or network parameters [4] to perform incidental exploration and are not very efficient at exploring the state space [35]. Many approaches incentivize exploratory behaviour through exploration bonuses such as surprise-maximizing intrinsic motivation [26], surprise-minimizing intrinsic motivation [3], and action [10], state [31] and trajectory [15] entropy maximizing rewards. Entropy maximization approaches fail to distinguish exploration in unseen regions (navigating to exploration targets) from exploration in regions of the state

space where the policy is already proficient (navigating to exploration frontiers). This makes them inefficient. Intrinsic motivation based methods use the notion of surprise as a mechanism to choose exploration targets, however, their inability to reason about navigation makes them struggle in hard-exploration sparse-reward environments [8]. Moreover, all these methods are unable to leverage affordances like offline data and resetable simulators when available.

Go-explore [8] is a conceptual framework that disentangles the question of selecting and reaching an exploration frontier. It is reminiscent of classical planners that first choose an exploration frontier, navigate to it quickly without exploring (or by resetting the simulator to that frontier state) and then initiate exploration after arriving at the frontier. Go-explore maintains an archive of visited states and chooses an exploration frontier from this archive either uniformly at random or using a domain specific heuristic. In Chapter 5 we expand on this theme by investigating what is a good way of picking an exploration frontier. We present generic properties that are desirable to have in this selection procedure alongside a broadly applicable mechanism to select exploration frontiers. Furthermore, in Chapter 4 we study how to efficiently navigate after reaching an exploration frontier.

BARL [22] is an information theoretic exploration method that uses a classical planner and a learnt posterior model to sample transitions that are maximally informative for the policy to learn a given task. This enables it to solve tasks very efficiently. The setting used by its authors bears close resembles to ours since they assume access to a simulator that supports arbitrary resets. Moreover, their use of a Gaussian process (GP) to model the posterior is amenable to utilizing expert demonstrations during training. While very effective on small scale problems with dense reward functions, BARL unfortunately does not scale to higher dimensions and sparse-reward settings. This is confirmed by us in our experiments.

2.2 Learning policies efficiently offline

Another way to efficiently learn policies is by training on a purely offline dataset of experiences. This sidesteps the issue of online exploration and efficiently recovers a policy based on the offline dataset. Methods such as behaviour cloning, GAIL [12] and AIRL [9] (an inverse RL method) fall under the broad class of imitation learning

algorithms that model policy learning as a supervised learning problem and learn a mapping from states to actions. A key issue with imitation learning methods is that they are highly brittle and require access to large amounts of high quality expert data to succeed [29].

Offline reinforcement learning [21] is another offline learning paradigm capable of efficiently learning policies from demonstration data of mixed quality while requiring good state coverage in the offline dataset [20, 29]. Consequently, a key challenge with this approach is that the lack of online interactions leaves offline RL susceptible to distribution shift. Wrongly estimating values for actions beyond the support of the dataset can hamper training [21]. Conservative Q-learning (CQL) [19] is a recent offline RL method that attempts to tackle this problem by maintaining pessimism within the Q-value function towards actions that are absent from the offline dataset. Implicit Q-learning (IQL) [18] completely avoids predicting value estimates for unseen actions by learning a distributional state-value function and computing an upper expectile over it to obtain the value estimate of the best action in that state. Real world deployment of these algorithms invariably lead to encounters with OOD states and actions making online finetuning a necessity for practical deployment of such algorithms.

2.3 Hybrid Reinforcement Learning

Hybrid reinforcement learning leverages a combination of offline data with online interaction to learn policies. The main challenge in hybrid reinforcement learning is to devise methods that effectively bootstrap online learning from offline data. Several approaches [11, 28] do this by using imitation to learn a policy from offline demonstrations before finetuning it with RL. However, most modern state-of-the-art online RL algorithms are value based [10, 30]. Naively finetuning an offline acquired policy with value based RL can cause significant performance degradation as a value function of similar quality to the pretrained policy is not available at the start of online finetuning [37]. Though Monte Carlo return estimate based algorithms exist, their online finetuning is known to be less efficient [25].

Offline RL presents a transferable paradigm to train a policy and value function with identical objectives in both offline and online setups. However, not all offline RL

methods are well suited for online finetuning due to their inherent pessimism towards distribution shift [25]. Even better suited offline RL methods like IQL [18] result in weaker policies after finetuning especially when limited offline data is available [37].

An alternative line of work [24, 35, 38] avoids finetuning a pretrained policy all together by pre-filling replay buffers at the start of training with transitions from the offline dataset. These transitions persist through training and policy learning happens from scratch. JSRL [37] presents an alternative approach to the finetuning-free idea of hybrid RL. It learns a guide policy from offline data and uses it to impart a curriculum for a freshly initialized policy by rolling out a part of the online episode before handing over control to the freshly initialized policy for completing the roll-out. The handover point is altered over the course of training and all the captured experience is used to train the freshly initialized policy.

Hierarchical RL [2, 6] is another formulation that has been used in the hybrid setup. In hierarchical RL, there exist two policies - a manager and a worker. The manager policy conveys intents that encapsulate higher level behaviours that serve as an abstract representation for a set of action sequences. These intents are passed to the worker policy that translate high level intents to low level action sequences. Parrot [34] is a method that leverages large scale offline data to learn a worker policy conditioned on the input observation. This serves as a *behavioural prior* that facilitates exploration via realistic trajectories from the start of training. This reduces the complexity of the search problem facilitating efficient online learning. One drawback of such a method is the need for large-scaling pre-training of behavioural priors on tasks with similar behaviour patterns. This may not always be feasible.

The work presented in this thesis delves into three of the four aspects of building an effective exploration algorithm. Chapter 4 tackles the question of efficiently getting to an exploration frontier or exploration target. It is akin to Parrot [34] in this regard, however, unlike Parrot we show that for tasks such as driving, effective behavioural priors can be encoded through simple reparameterizations. These priors are unconditional on the input observation and do not require any data or learning to induce good inductive biases on the manner of exploration. The work proposed in Chapter 5 lies in the finetuning-free hybrid setting and shares similarities with both Go-explore [8] and JSRL [37]. All these methods choose an exploration frontier and reach them efficiently. The proposed work and Go-explore use environmental

resets to reach the desired frontier while JSRL uses a learnt policy to navigate there. The proposed work and JSRL go one step further and induce a specific kind of reset distribution. JSRL does this through its gradual hastening of the handover point over the course of training. In contrast we choose the reset distribution with more care and through our experiments show that this is an important choice that influences the sample-efficiency and performance of the learnt policy.

Chapter 3

Preliminaries

3.1 Reinforcement Learning

We model the reinforcement learning problem by defining a finite-horizon discrete-time MDP $\mathcal{M}$ to be a $(\mathcal{S}, \mathcal{A}, r, p_0, H, \mathcal{T}, \gamma)$ tuple where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, p_0 is a probability distribution defined over $\mathcal{S}$ corresponding to the start state distribution of $\mathcal{M}$, $H \in \mathbb{N}$ is the finite time horizon, γ is the discount factor and $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to P(\mathcal{S})$ describes a transition function capturing the distribution of next states when an action $a \in \mathcal{A}$ is taken at a state $s \in \mathcal{S}$.

The goal is to obtain a policy $\pi(a|s)$ that maximizes the expected sum of future discounted rewards from p_0. Concretely, the objective is to maximize -

$$\mathcal{J}_{p_0}(\pi) = \mathbb{E}_{s_0 \sim p_0, s_{t+1} \sim \mathcal{T}(s_t, a_t), a_t \sim \pi(s_t)}[\Sigma_{t=0}^{H} \gamma^t r(s_t, a_t, s_{t+1})] \tag{3.1}$$

Equation 3.1 presents the commonly desired metric to be maximized during reinforcement learning. The quality of this metric is contingent upon the state visitation distribution induced by the choice of p_0. The real world may often present situations that are not well covered by the induced state distribution arising from p_0 [27], resulting the policy having to navigate novel situations that it may not have encountered often enough while training. Consequently, for real world deployment, it is desirable for a policy to be robust in addition to maximizing $\mathcal{J}_{p_0}$. While there

are many to benchmark robustness, we formulate it as desiring a policy learnt to maximize $\mathcal{J}_{p_0}$ to also do well on a broader start state distribution which we call μ_{OOD}, i.e., maximize $\mathcal{J}_{\mu_{OOD}}$ (OOD refers to Out-Of-Distribution). The state distribution induced by a policy starting from μ_{OOD} may be different from the induced distribution of states when starting from p_0. As a result proficiency starting from p_0 does not imply proficiency starting from μ_{OOD}. For policies designed for real world deployment, it is important for them to generalize beyond the training distribution making evaluation from μ_{OOD} a useful benchmark to consider.

In Chapter 5, we investigate how sample-efficiency of online RL can be enhanced through a suitable choice of μ by leveraging a small amount of expert data and a simulator permitting arbitrary state resets. Through a sparse-reward navigation task we show how this not only enables accelerated learning on the MDP start state distribution, p_0, but also on a broader set of start states μ_{OOD} through the construction of robustness benchmark.

3.2 Bezier Curves

In Chapter 4 we reparameterize the action space using a family of parameterizable curves known as Bezier curves. Bezier curves have been used in a variety of domains, particularly graphics, to compactly represent smooth curves. More formally, an order n Bezier curve $\mathcal{B}(t)$ is defined over the domain $t \in [0, 1]$ as -

$$\mathcal{B}(t) = \binom{n}{0}(1-t)^n P_0 + \binom{n}{1}(1-t)^{n-1} t P_1 + ... + \binom{n}{x}(1-t)^{n-x} t^x P_x + ... + \binom{n}{n} t^n P_n \tag{3.2}$$

Here, $P_0, ...P_n$ are a set of points in the d dimensional space that the bezier curve resides in. Points P_0 and P_n are the start and end point of the bezier curve respectively. The remaining points reweigh the curve and help obtain curves in a variety of shapes. The higher the order of a bezier curve, the more complex a curve that can be represented and the order n of the curve ensures the curve is n times differentiable over the closed interval $t \in [0, 1]$. These characteristics allow Bezier curves to represent a large variety of naturally occurring curves. Vehicular trajectories is one such type of naturally occurring curve. These trajectories are fairly simple and

as we will show in Chapter 4 can be represented by cubic (order 3) bezier curves.

Chapter 4

Temporal Action Abstractions via Bezier Curves

4.1 Introduction

Finding good policies for solving robotic tasks such as manipulation and navigation entails learning in continuous action spaces to learn effective trajectories of actuators. Performing exploration for doing reinforcement learning in such action spaces can be challenging especially in safety critical domains such as autonomous driving. It has been shown that using online RL to learn simple driving policies that predict instantaneous steer and throttle can take extremely long to train (over 10 million iterations [13]) on simple benchmark like NoCrash [5] in CARLA [7].

The exploration challenges of such continuous action spaces can get further exacerbated in robotic domains since effective policies must take feasible and temporally consistent actions over a time horizon. Random exploration through instantaneous control can generate a vast amount of unrealistic trajectories that would not take place in the real world. Figure 4.1 illustrates this by showing two possible trajectories that instantaneous steer can generate. One of these is jittery and unlike real world trajectories observed on the road. The other is a much more probable trajectory that can be encountered in the wild.

By allowing a policy to apply an instantaneous steer at each timestep over a time

Figure 4.1: Two examples of trajectories that can be generated by a policy that predicts instantaneous steer. On the left is a jittery trajectory that is unlikely to be seen in the wild, while on the right is a smooth trajectory that resembles commonly seen real world trajectories.

horizon H, our search problem over the space of possible trajectories spans the entirety of the $\mathbb{R}^H$ space. However, realistic trajectories as shown in Figure 4.1 can potentially be expressed in a much lower dimensional action space. Thus the use of instantaneous steer causes an unnecessary increase in the amount of exploration needed resulting in policies that are slower to train. This motivates the need for compact temporal abstractions of actions that can reduce the search space and facilitate accelerated learning.

In this chapter, we dive deeper in the pathology of inefficient exploration in safety critical domains, specifically onroad autonomous driving, and demonstrate how using general inductive biases to constrain the set of actions can dramatically shrink the exploration problem and lead to accelerated learning,

4.2 Temporal Re-parameterization of Action Space

Real world trajectories are known to exist in low dimensional manifolds of the $\mathbb{R}^H$ space spanned by a sequence of steering angles or waypoints [16]. Moreover, real

world motion is generally smooth. Such properties make it feasible to compactly capture sequences of instantaneous control through parameterized curves without sacrificing the diversity of feasible trajectories attainable by the policy. As described in Chapter 3.2, Bezier curves are a form of smooth parameterized curves that can represent a diverse set of trajectories compactly. The complexity of curves (representing the degree of a polynomial that produces the curve) can be controlled via the degree of the Bezier curve. By altering the control points of the curve, it is possible to obtain trajectories of different shapes. Thus predicting these control points as part of the action allows the policy to directly reason in the trajectory space while at the same time restricting the search space to smooth trajectories of some bounded degree of complexity.

We find that using cubic Bezier curves as an action representation provides us with sufficient diversity to capture a large variety of potential curves that can appear in the real world. Specifically, a cubic Bezier curve $B_3(t)$ is described via a start point $P_0 = (x_0, y_0)$, an end point $P_3 = (x_3, y_3)$ and two control points $P_1 = (x_1, x_1)$ and $P_2 = (x_2, y_2)$. To use these curves as an action representation, we have the policy $\pi(s)$ learn to predict the following parameter values in stead of steering angle.

$$\pi(s) = [x_1(s), y_1(s), x_2(s), y_2(s), y_3(s)] \tag{4.1}$$

Note that we don't predict all parameters of this curve since (x_0, y_0) is the origin of the policy's frame of reference and hence always $(0, 0)$ and x_3 is set to be a fixed constant distance away.

By predicting the parameters of a Bezier curve, we enable the policy to additionally reason in time. This would not be possible to do with instantaneous steer unless a history of states is passed as input to the policy. Using a history of states would not only increase the size of the observation space but also fail to constrain the search space in actions thereby not contributing to sample-efficiency.

Once a policy predicts the parameters of the Bezier curve, we discretize the curve into H uniformly spaced waypoints. We pick a contiguous set of k waypoints and use these waypoints to provide the *next waypoint orientation* to the policy's observation for the next k time-steps. This k is referred to as the action horizon. As we will show subsequently, in order to gains training speedups and improved performance, it is

crucial to utilize $k > 1$ while training with Bezier curves as $k = 1$ is equivalent to predicting the next target waypoint and prevents the policy from learning non-trivial curves.

4.3 Evaluations

We focus on navigation as a testbed for understanding the impact of temporal action abstractions through Bezier curves. Specifically we study the problem in the context of onroad autonomous driving. In this problem, safety is of paramount importance. Consequently, a good policy must thoroughly explore the set of achievable trajectories and learn to only generate the small subset that ensure the task is safely and successfully completed.

4.3.1 Setup

We perform evaluations of the proposed method in CARLA[7] which is a photorealistic simulator for autonomous driving research. Specifically, we use Town 1 from CARLA version 9.10 to create an overtaking scenario as shown in Figure 4.2. In this scenario, we have a stationary vehicle in the path of the ego vehicle. In order to complete the task, the ego vehicle must exit its lane, get around the stationary vehicle and re-enter the lane to progress towards the target waypoint. The position of the stationary vehicle is randomly chosen along the episode trajectory. We train all our policies using PPO [30].

For each experiment we use a 24 dimensional observation space which is as follows

Figure 4.2: The overtake scenario in CARLA comprising an ego vehicle and a stationary vehicle parked in the path of the ego vehicle.

Table 4.1: Influence of Action Horizon on policy performance for different action parameterizations. All policies are trained for 1.5 million iterations

Action Space	Action Horizon	Success Rate (%)
Steer	1	60
Bezier	1	60
Steer	4	2
Bezier	4	**75**

$$\begin{pmatrix} \text{next waypoint orientation} \\ \text{ego vehicle speed} \\ \text{ego vehicle steering angle} \\ \text{distance to waypoint trajectory} \\ \text{front obstacle displacement and relative velocity} \times 4 \text{ dims} \\ \text{front right obstacle displacement and relative velocity} \times 4 \text{ dims} \\ \text{front left obstacle displacement and relative velocity} \times 4 \text{ dims} \\ \text{back right obstacle displacement and relative velocity} \times 4 \text{ dims} \\ \text{back left obstacle displacement and relative velocity} \times 4 \text{ dims} \end{pmatrix}, \tag{4.2}$$

For control, we predict a trajectory representation which is the control points of a cubic Bezier curve if using a temporally abstracted action representation or the steering angle when using instantaneous control. Simultaneously a speed value is also predicted.

4.3.2 Do Bezier Action Spaces learn more temporally accurate action sequences?

We evaluate both instantaneous control and Bezier action spaces at two granularities of action horizons - 1 and 4. As described above, action horizons of $k > 1$ are achieved through the usage of frame skipping where the policy is used to predict an action every k frames. We train all policies for 1.5 million iterations. In order to avoid confounders, we fix the speed to a constant value (20 km/h) and ask the policy to

Table 4.2: Sample efficiency and final performance for different action parameterizations when trained to convergence

Action Space	Training Steps (millions)	Success Rate (%)	Top Speed (km/h)
Steer	10	85	20
Steer + Speed	10	0	-
Bezier	2.5	**90**	20
Bezier + Speed	**0.8**	**90**	**40**

only predict the trajectory component of the action space.

The results of this study are summarized in Table 4.1. We find that for an action horizon of 1, policies trained via instantaneous steer and Bezier reparameterized actions both achieve the same performance. This intuitively makes sense since at an action horizon of 1, only a single waypoint is being chosen on the Bezier curve for each policy prediction. This is equivalent to a policy that learns to predict the next waypoint at a fixed distance away from the ego agent, which in turn is equivalent to predicting the steering angle.

When the action horizon is raised to 4 we begin to see significant differences in the performance of the two policies. The instantaneous control policies performance deteriorated catastrophically down to 2%, indicating that steering angle by itself is a poor representation of trajectory over longer time horizons. Meanwhile the Bezier reparameterized policy sees its performance improve to 75% indicating its effectiveness at modeling temporally accurate action sequences.

4.3.3 Do Bezier Action Spaces improve sample-efficiency?

To evaluate sample-efficiency, we train four policies to convergence - two instantaneous steer policies (with and without learnable speed) and two Bezier reparameterized policies (with and without learnable speed). We report the success rate of each policy at convergence as well as the number of training steps taken to attain that performance.

Table 4.2 presents the results of this evaluation. We see that when learning with fixed speed, the reduced search space of the Bezier action space enables learning a better policy than instantaneous control based policy 4× faster. More surprisingly,

we find that learning both speed and trajectory actions with an instantaneous control policy ends up failing to learn the task as the policy learns to get the ego agent to a complete stop rather than risk a collision. On the other hand the Bezier action spaces facilitate learning both speed and trajectory control achieving even larger sample efficiency gains (12.5×) in addition to learning a policy with 2× higher max achieved velocity.

4.4 Limitations and Discussion

We have seen that temporally reparameterized action spaces facilitate accelerated learning online. In this work we used Bezier curves as a parameterization roughly capturing curves that constitute real world trajectories. However, Bezier Curves are not the only class of parameterized curves that can be used and it would be interesting to see how other smooth parameterizations of curves, such as common splines used in computer graphics, can be utilized for robotic tasks.

While the results of this re-parameterization are promising, we have observed that training with larger action horizons such as $k \in 8, 12, 16$ is still very challenging. We find that as larger action horizons are used, training times increase owing to larger frame-skips and higher amounts of simulator interaction needed to collect the same quantity of training data. In addition, training becomes more unstable since there are a larger set of next states that can be achieved by rolling out a policy for more time steps. This makes learning accurate action value functions harder and results in diverging training runs. In the overtake scenario, this results in agents exploring limited regions of the state space. For example, when training with larger action horizons, the agent will spend most of its time attempting to exit its lane in order to get around the stationary vehicle, while spending almost no time in returning to its lane after passing the stationary vehicle. This indicates that the original start state distribution is perhaps not the best at ensuring good coverage of the state space. In the next chapter we will explore this observation further and present a method to tackle this issue.

Lastly, we find that in challenging variants of larger scale driving benchmarks such as NoCrash [5], the presence of dynamic actors makes learning harder and results in instabilities that aren't fully resolved by this re-parameterization. Improving sample

efficiency in such setups with other dynamic actors is a key unsolved problem that would greatly enhance the applicability of online reinforcement learning in large scale multi-agent robotic tasks such as autonomous driving.

Chapter 5

Accelerated Online RL via Auxiliary Start State Distributions

5.1 Introduction

In Chapter 3.1, we introduced the standard reward maximization objective ($\mathcal{J}_{p_0}(\pi)$) in RL. $\mathcal{J}_{p_0}(\pi)$ is maximized with respect to a pre-specified start state distribution p_0. While p_0 is part of the maximization objective in Equation 3.1, it is not necessary to train with this start state distribution in order to maximize the objective function. As discussed in [17], a narrow p_0 can down-weight the influence of unlikely but important states during policy improvement by having the induced state visitation distribution visit such states infrequently. The presence of such unlikely but important states along the optimal trajectory, which we refer to as *task critical states* ($\mathcal{C}$), can result in slow learning progress. It would instead be beneficial to have a start state distribution (μ) that is distributed more uniformly across some subset $\mathcal{S}_{sub}$ of the state space (i.e., $\mathcal{S}_{sub} \subseteq \mathcal{S}$) emphasizing visitation of *task critical states* while asymptotically obtaining comparable performance to the optimal policy as measured by $\mathcal{J}_{p_0}(\pi)$.

5.2 Constructing Auxiliary Start State Distributions

Emphasizing visitation of *task critical states* ($\mathcal{C}$) is an appealing concept but requires us to address the challenge of efficiently identifying such task critical states and subsequently determining a suitable distribution to implement this emphasis in a tractable manner.

Directly computing the visitation distribution over *task critical states* ($\mathcal{C}$) is challenging in continuous state spaces owing to the associated computational infeasibility of calculating this quantity. Moreover, this is a policy dependant quantity and will need to be continuously recomputed over the course of training. As a result, a suitable μ should be an easily computable dynamic distribution that accounts for the changing visitation distribution of the policy over time.

We observe that episode termination is a powerful and ubiquitous signal available in a variety of RL tasks. It is especially prevalent in robotic tasks such as autonomous driving and robot locomotion where eventual real world deployment is the end goal and safety of the agent and its surroundings are of paramount importance.

Intuitively, for a state s, if the proportion of actions that cause the agent to land in a terminal state is high, then a larger exploratory budget is required to learn a feasible action for this state by the policy. Moreover, the chance of navigating through this state likely hinges on the repeated selection of a small set of safe and feasible actions in the neighbouring regions of the state space. Therefore, there is a high likelihood that such states belong to $\mathcal{C}$ and sampling them more frequently can accelerate learning. More formally, we define the notion of state safety for any state $s \in \mathcal{S}$ as -

$$\Omega_\pi(s) = \int_{a_{0:k-1}} P(a_{0:k-1}|s,\pi) \int_{s_k} P(s_k|s, a_{0:k-1}, \mathcal{T}, \pi)\mathcal{Z}(s_k)\, ds_k\, da_{0:k-1} \tag{5.1}$$

Here, $\mathcal{Z}(s) \in \{0,1\}\, \forall s \in \mathcal{S}$ and denotes whether or not state s causes episode termination. $\mathcal{Z}(s) = 0$ if episode termination is caused by being in state s and 1 otherwise. $a_{0:k-1}$ is the sequence of k actions induced by the policy π from state s

Algorithm 1 Online RL with Auxiliary Start States

Inputs: Task Horizon H, Offline Demonstration States $\mathcal{S}_{demo}$, Algorithm $\mathcal{A}$, Training Timesteps T_{max}, Environment $\mathcal{E}$, replay buffer $\mathcal{B}$
Sampling distribution $\mathcal{W} \leftarrow \overbrace{[1, 1 \dots 1]}^{\text{len}(\mathcal{S}_{demo})}$ ▷ Initialization incentivizes visiting states atleast once
Sampling distribution norm $\mathcal{N} \leftarrow \text{SUM}(\mathcal{W})$
$t \leftarrow 0$
while $t \leq T_{max}$ **do**
    $i \leftarrow \text{SAMPLESTARTSTATE}(\frac{\mathcal{W}}{\mathcal{N}})$
    $s_0 \leftarrow \mathcal{S}_{demo}[i]$
    $L_{ep} \leftarrow \text{TRAINFORONEEPISODE}(\mathcal{A}, \mathcal{B}, \mathcal{E}, s_0))$ ▷ Return value is episode length
    $t \leftarrow t + L_{ep}$
    $\mathcal{W}, \mathcal{N} \leftarrow \text{UPDATESAMPLER}(\mathcal{W}, \mathcal{N}, L_{ep}, i, H, \mathcal{S}_{demo})$ ▷ See Algorithm 2
end while

under the transition model given by $\mathcal{T}$. s_k is the state that is reached when policy π takes action sequence $a_{0:k-1}$ starting from state s in an environment with transition model $\mathcal{T}$.

Exactly computing $\Omega_\pi(s)$ is still computationally expensive. Instead we leverage the time to termination or episode length from a given start state, which is a freely available metric at training time, as a Monte Carlo approximation of the true state safety for a given policy at that state. By maintaining a parameterized distribution over a set of desirable start states (*candidate task critical states* or $\widetilde{\mathcal{C}}$) we can exploit local smoothness in the majority of the state space to quickly propagate these approximations across the start state distribution (see Algorithm 2 for details)

Since we are taking a hybrid approach to RL, we have access to a limited amount of expert demonstration data. Since this data comprises successful demonstrations of the task, the demonstration trajectories will likely contain *task critical states* if they exist. As a result, we can simply set $\widetilde{\mathcal{C}}$ to be the states from the demonstration data and identify $\mathcal{C}$ from amongst these states over the course of training. Putting everything together we get our proposed method *AuxSS* that we describe in Algorithm 1.

Algorithm 2 Updating Auxiliary Start State Distribution via Episode Length (*AuxSS*)

Inputs: Sampling distribution $\mathcal{W}$, Sampling distribution norm $\mathcal{N}$, Episode Length L_{ep}, Update index i, Task Horizon H, Offline Demonstration States $\mathcal{S}_{demo}$, Weight Threshold δ, Smoothing Variance σ^2
Outputs: Sampling distribution $\mathcal{W}$, Sampling distribution norm $\mathcal{N}$
$\mathcal{W}[i] \leftarrow \textsc{Max}(\frac{H-L_{ep}}{H}, \delta)$ ▷ δ ensures probability of sampling ≥ 0
$\lambda \leftarrow \frac{1}{\sqrt{2\pi\sigma}}\textsc{exp}(\frac{(S_{demo}-S_{demo}[i])^2}{2\sigma^2})$ ▷ λ is used for smoothing updates to $\mathcal{W}$
$\mathcal{W} \leftarrow (1-\lambda)\mathcal{W} + \lambda\mathcal{W}[i]$
$\mathcal{N} \leftarrow \textsc{sum}(\mathcal{W})$

5.3 Evaluations

In this section we first highlight the affinity of using auxiliary start state distributions with the hybrid RL setup by demonstrating state-of-the-art sample-efficiency on a sparse-reward hard-exploration task. We show that *AuxSS* is better suited at assimilating information from limited amounts of expert offline data by demonstrating better sample-efficiency than competing approaches that have access to 15× more offline expert data available to them. We simultaneously show *AuxSS* also facilitates learning more robust policies. Finally, we empirically demonstrate that approximating a more uniform visitation distribution over $\mathcal{C}$ through Ω facilitates accelerated learning. We showcase how *AuxSS* is a good way to approximate Ω while other distributions not motivated in the same way are not.

5.3.1 Setup

Environments: We conduct our experiments on two testbeds. The first is a suite of four continuous control robotic tasks from MuJoCo [36] and the second is a 2D maze setup shown in Figure 5.4. The maze consists of two large regions connected by a narrow traversable passage with obstacles on either side (shown in blue). The obstacles are pits of of lava and the episode terminates if an agent encroaches this area. We refer to this environment as Lava Bridge. It has a continuous $4D$ state space comprising 2 dimensions of position and 2 dimensions of velocity. The action space is a continuous $2D$ force vector. All evaluations are performed in a sparse-reward setup

where the agent only gets a non-zero reward on reaching the goal state or entering a terminal state.

Evaluations: On the both the MuJoCo tasks and Maze navigation task, we track the episodic returns and present plots showing the evolution of these returns over the course of training. On the maze environment, we in addition track the success of the learning policy on two start state state distributions. The first is the MDP start state distribution (p_0) and the second is an out-of-distribution start state distribution (μ_{OOD}) to serve as a robustness benchmark. Both these distributions are shown in Figure 5.4.

Affordances: In all our evaluations, we assume access to two affordances - a small quantity of offline demonstration data and a simulator that supports arbitrary resets. For MuJoCo tasks, we assume access to a 1000 transitions of demonstration data from a medium policy trained to a third of the performance of a converged SAC policy. This is equivalent to 1 demonstration trajectory worth of data. On the maze navigation task, unless otherwise specified, we use 500 transitions of offline data, which is equivalent to the maximum episode length allowed for the task.

Note: Since the distribution of offline data μ_{off} is generated via an expert starting from the distribution p_0, the state distribution of μ_{off} will be similar to the induced state distribution of a well performing policy starting from p_0. As a result, resetting the simulator to a μ derived from μ_{off} will not compromise the robustness benchmark. At the start of training, novel states emerging from changing reset distributions from p_0 to μ_{OOD} will remain novel when changing reset distributions from μ to μ_{OOD}. Over the course of training, resetting to μ may simply alter the likelihood of visiting some novel states. The impact of this altered likelihood will be observable through the robustness evaluations.

5.3.2 Do Auxiliary Start State Distributions accelerate learning of robust policies?

In this section we study the efficacy of *AuxSS* at improving the sample efficiency of online learning by making use of affordances such as arbitrary resetting of the environment and access to a limited quantity of expert demonstration data. We compare *AuxSS* with a variety of offline, online and hybrid methods. In addition

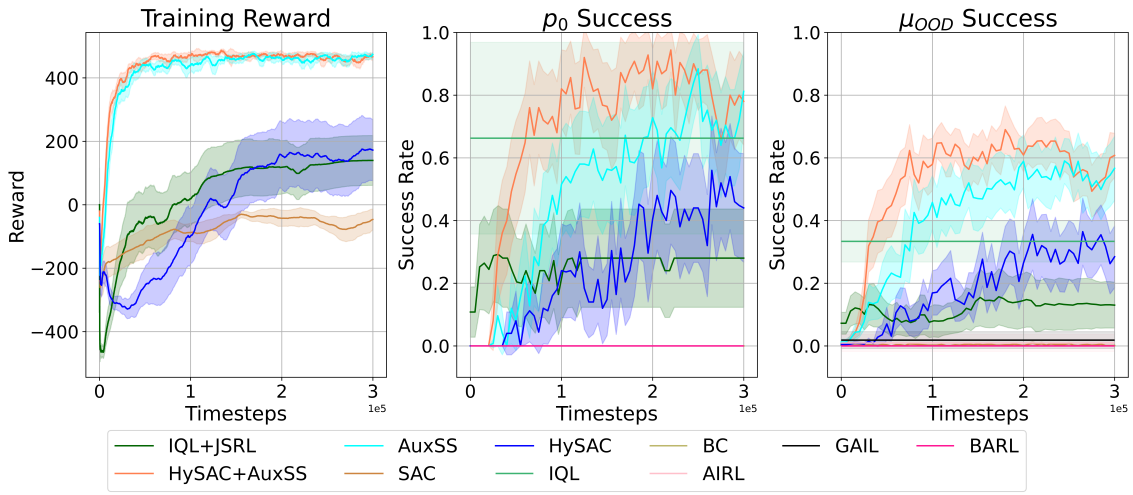

Figure 5.1: Task Completion Rate of Various methods on the Lava Bridge Environment. Each method is evaluated on an In Distribution (ID) and Out-of-Distribution (OOD) benchmark of starting states where the ID start state distribution is the start state distribution of the MDP while the OOD benchmark comprises a different distribution of start states.

to tracking sample-efficiency we also track the robustness of the learnt policy. The choice of the MDP start state distribution (p_0) and robustness benchmark start state distribution (μ_{OOD}), are shown in Figure 5.3.

Figure 5.1 presents the findings of this study on the Lava Bridge environment. A standardized training setup has been used across methods (except BARL [22]) where the number of offline demonstration transitions is set to 10 million, number of online learning steps is 300000, replay buffer size is 10000, max episode length is 500 and experiments are evaluated across 25 seeds. All hybrid methods and offline methods have access to 500 transitions of expert demonstration data.

We use the example of SAC [10], a purely online RL method, to highlight the exploration challenges that the Lava Bridge environment poses to standard online RL. SAC uses an undirected entropy based bonuses to promote exploration but struggles to efficiently explore in our environment. Its failure to reach the goal within the stipulated training budget and learn a robust policy highlights the exploration challenges posed by the Lava Bridge environment. In addition we evaluate BARL [22], an information theoretic method for sample-efficient online exploration. We evaluated BARL both with and without access to the demonstration data and found

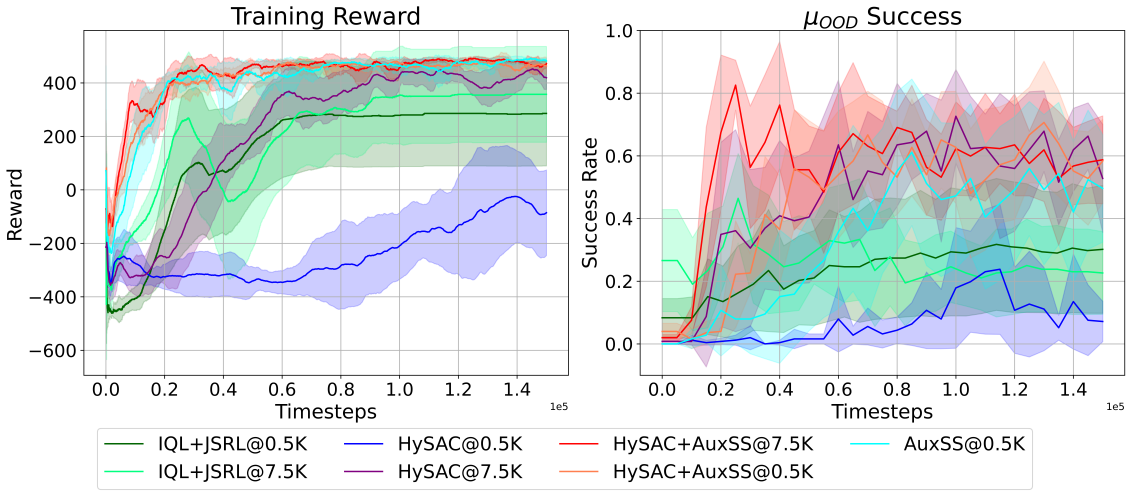

Figure 5.2: A study of how sample-efficiency and robustness vary for hybrid RL methods when provided with different amounts of demonstration data.

it unable to solve our task. BARL is reliant on a classical planner which is designed to work with dense rewards. Therfore the sparse-reward nature of our task prevents the BARL planner from finding solutions in a feasible amount of time, resulting in the method's failure.

We compare our proposed approach with two hybrid RL approaches - HySAC (an adaptation of HyQ [35] where a DQN is replaced with SAC) and JSRL [37]. It can be seen that across training reward and success rates, using a good auxiliary start state distribution yields state-of-the-art sample efficiency and performance as both HySAC and JSRL struggle to make full use of the limited offline demonstration data. Moreover, the proposed approach is complementary to HySAC's persistent storage of offline demonstration in the replay buffer throughout training and as a result the two approaches can be coupled (HySAC+*AuxSS*) to obtain better robustness in fewer training steps.

It can be noted that the approach taken by JSRL of handing over episode rollout from a guide policy to the learning policy is conceptually similar to having an auxiliary start state distribution that monotonically recedes towards p_0 over the course of training. Unlike our proposed auxiliary distribution, JSRL cannot reemphasize visiting previously learnt regions of the state space that may have been forgotten over the course of training. Even accounting for some sample-efficiency gains that

JSRL may obtain by directly resetting to the handover point (rather than using the guide policy to get there) withing an episode, its inability to reemphasize visitation of previously learnt regions prevent it from learning very robust policies as can be seen in Figure 5.1.

We also compare against purely offline methods such as imitation learning and offline RL. We find that imitation learning methods like behaviour cloning (BC), GAIL [12] and AIRL [9] fail to succeed on both p_0 and μ_{OOD}. This stems from their inability to learn meaningful policies from just 500 transitions. Infact, we provide these methods with 7.5K transitions to learn from in Figure 5.1 and they still fail to learn a reasonable policy as measured by both start state distributions. Offline RL fairs a lot better. IQL [18] is a popular offline RL method that we compare against. We find that IQL performs well on p_0 but suffers a large drop in performance when evaluated on μ_{OOD}. This arises from the fact that offline RL methods are incentivized to learn a good policy only within the distribution of their training data which in this case is μ_{off}. For a policy that performs well starting from p_0, the induced state distribution of this policy will be close to μ_{off} and as a result a policy learnt using IQL does well on p_0. On the other hand, the induced state distribution of the policy learnt via IQL when starting from μ_{OOD} would present a distribution shift with respect to μ_{off} resulting the in the poor robust performance of IQL.

5.3.3 Influence of offline demonstration set size on performance and sample-efficiency

In Figure 5.2 we plot the training reward and evaluate robustness when different quantities of expert demonstration data ($0.5K$ and $7.5K$ expert samples) are available prior to the online learning phase. We find that by accessing $15\times$ fewer expert samples *AuxSS* and HySAC+*AuxSS* can match and exceed the robustness and sample efficiency of policies learnt via other hybrid RL methods. When provided access to a resetable simulator, this demonstrates that a good auxiliary start state distribution can more effectively assimilate data to guide exploration and accelerate learning than other approaches to hybrid RL. Unlike other methods, having a good start state distribution prevents the need to collect large quantities of expert data through ability to bootstrap online learning off of very limited demonstration trajectories (the 500

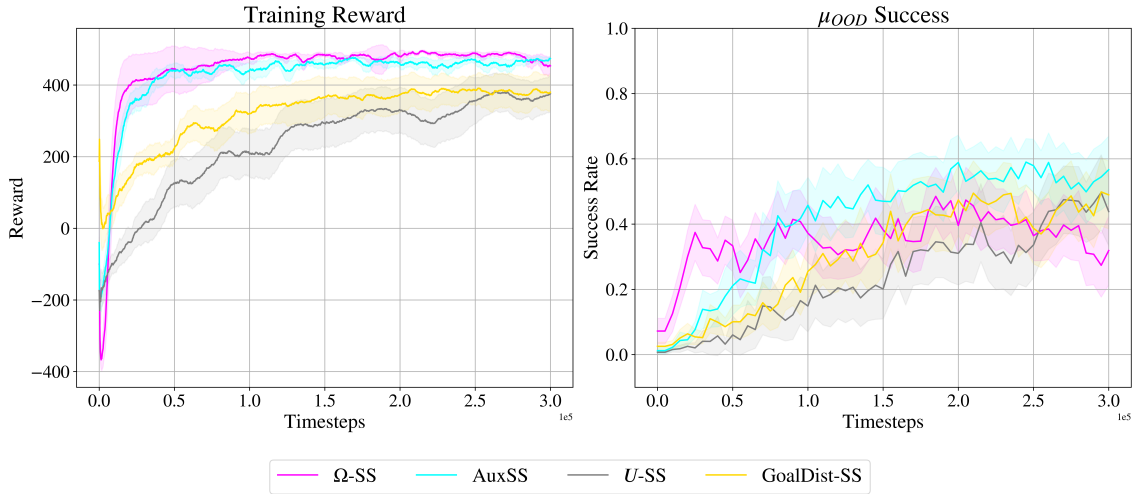

Figure 5.3: Sample-efficiency and robustness trends when simulator resets are selected using different start state distributions.

expert transitions come from 3 demonstration trajectories. Reported *AuxSS* trends sample from a random subset of 150 transitions. This is approximately one trajectory worth of expert data).

5.3.4 State safety inspired start state sampling for sample efficiency

In Chapter 5.2, we connect the notion of state safety Ω with *task critical states* ($\mathcal{C}$) and discuss how this can influence sample-efficiency. In this section, we empirically validate our claims. We modify *AuxSS* by constructing a static distribution (Ω-SS) that sample start states with respect to a random policy. Concretely, we sample start states inversely proportional to $\Omega_{\pi_{rand}}(s)$ where $\pi_{rand}(.|s) \sim \mathcal{U}^{|\mathcal{A}|}$. In practice, we use Monte Carlo sampling of actions for a fixed time horizon (= 4 time steps) to approximate this quantity for each state. Since the policy at the start of online training is initialized randomly, this mimics the state safety distribution with respect to the policy at the start of training. Therefore if our claims hold we expect to see matching sample-efficiency trends to *AuxSS* in the early stages of training.

Figure 5.3 presents the findings of this study. We see that as expected, Ω-SS demonstrates matching sample-efficiency and robustness trends as *AuxSS* early in

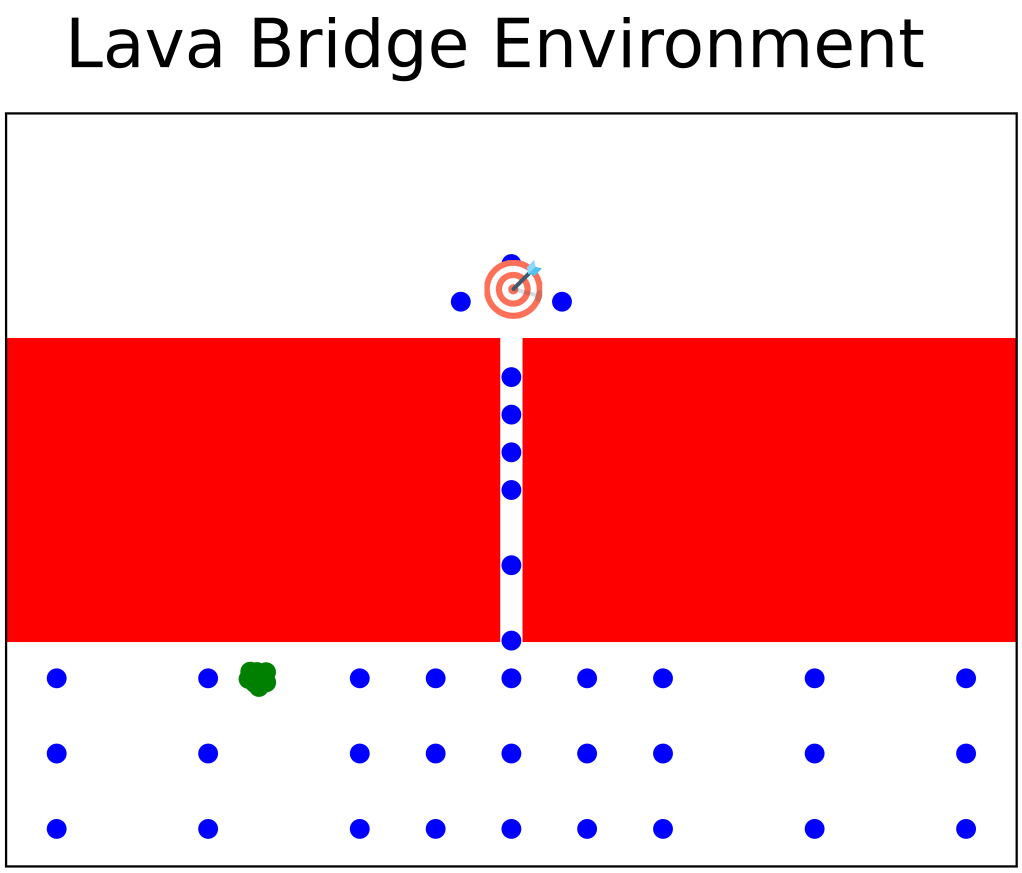

Figure 5.4: Sample-efficiency and robustness trends when simulator resets are selected using different start state distributions. An illustration of the Lava Bridge environment. The red regions are the lava pits, the green blobs denote p_0 and the blue spots correspond to the distribution μ_{OOD}. The red target marks the goal location.

training. In fact, since Ω-SS is the correct state safety distribution with respect to the initialized policy from the start of training it learns even faster than *AuxSS*, since *AuxSS* must gradually approximate this state safety distribution over the course of multiple training episodes.

We note here the divergence in robustness trends seen later in training. This is caused by the static nature of Ω-SS which fails to adapt to the morphing $\mathcal{C}$ induced by the policy as it trains. This causes the resulting loss of robustness. The dynamic nature of *AuxSS* helps prevent this degradation as its able to adapt its start state distribution based on changes in the policy.

5.3.5 Do start state distributions not deriving from state safety fail to be sample efficient?

To study this inverse logical question, we construct two start state distributions, $\mathcal{U}$-SS and *GoalDist*-SS, that do not try to incentivize visitation of *task critical states*. $\mathcal{U}$-SS is a static distribution that uniformly samples states from the provided demonstrations. *GoalDist*-SS is a dynamic distribution that exponentially weights states based on their distance from the task goal. States closer to the goal are assigned a higher probability to be sampled. The time varying component of this distribution arises from temperature scaling of the distribution with the temperature gradually rising over the course of training. This promotes sampling near goal states early on in training and sampling more uniformly from the demonstration data later on in training.

Figure 5.3 contains the findings of this study. It can be seen that both $\mathcal{U}$-SS and *GoalDist*-SS are far slower to train than state safety inspired distribution demonstrating that not all start state distributions will accelerate learning. As a consequence of the poor choice of their state visitation, these methods fail to learn good policies in the stipulated training budget and thus also have much lower robust performance than *AuxSS* and Ω-SS (before its static nature causes robustness to degrade).

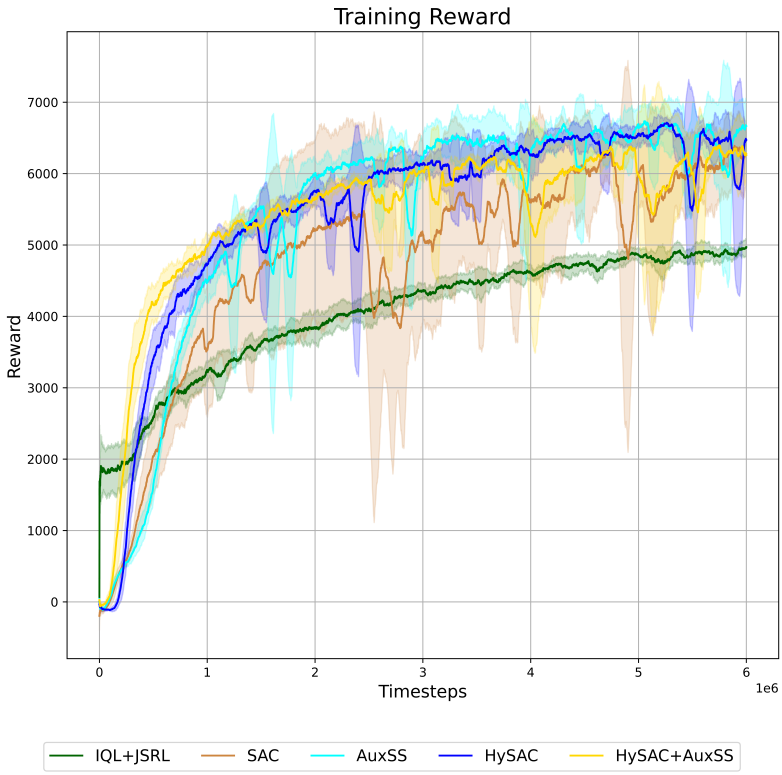

Figure 5.5: Performance comparison of various online and hybrid RL methods on the MuJoCo Ant-v4 continuous control task.

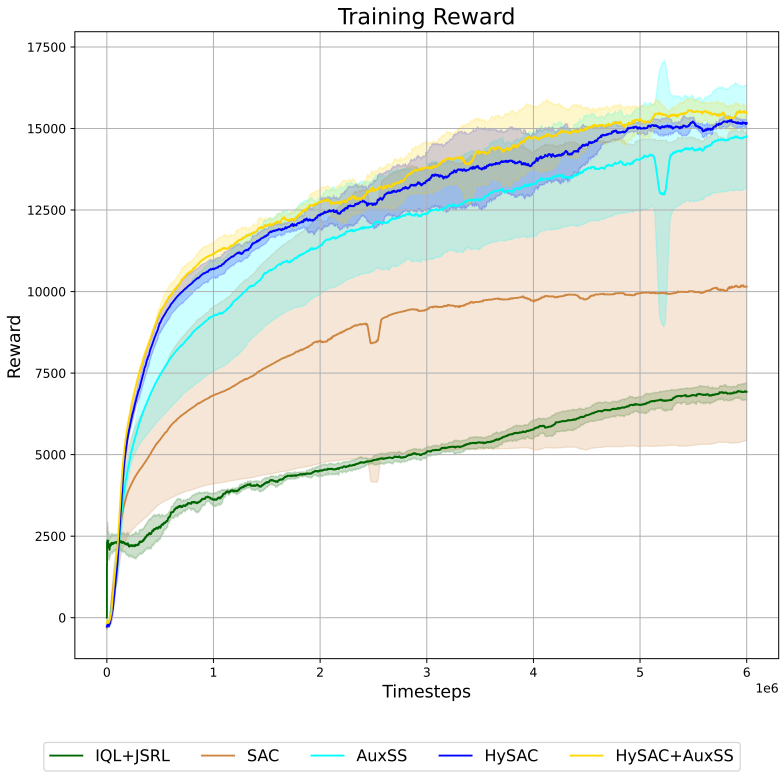

Figure 5.6: Performance comparison of various online and hybrid RL methods on the MuJoCo HalfChetaah-v4 continuous control task.

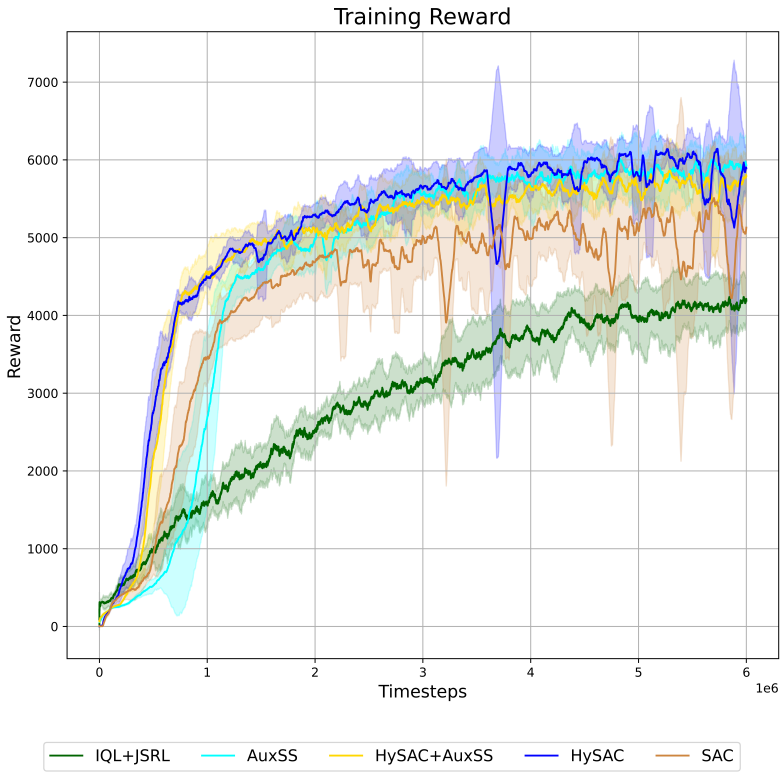

Figure 5.7: Performance comparison of various online and hybrid RL methods on the MuJoCo Walker2D-v4 continuous control task.

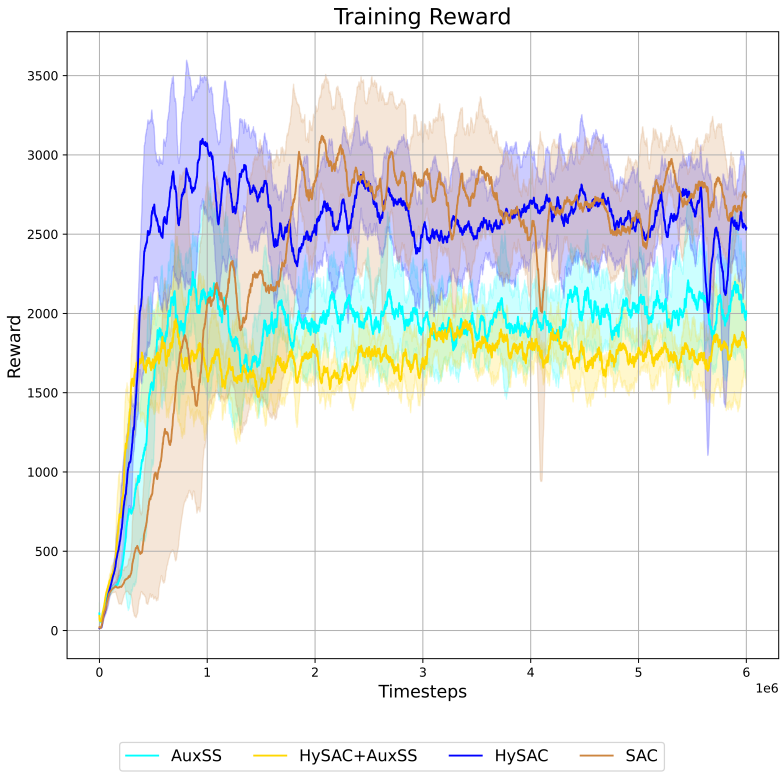

Figure 5.8: Performance comparison of various online and hybrid RL methods on the MuJoCo Hopper-v4 continuous control task.

5.3.6 Performance on MuJoCo Continuous Control Tasks

In this section we compare the performance of the proposed approach with existing methods on a suite of three MuJoCo continuous continuous tasks - Ant, Walker2D and HalfCheetah. We train each policy for 6 million timesteps (Walker2D has been trained for 3 million and will be updated in the final version of the thesis. In addition a fourth task, Hopper, will be added). As seen in Figures 5.5 - 5.8, both HySAC and AuxSS consistently achieve higher reward than JSRL and SAC across these tasks indicating that both methods enhance exploration and enable learning more performant policies. It should be noted that while HySAC and AuxSS perform similarly, AuxSS only requires access to an unordered set of demonstration states while HyAC requires an ordered set of state, action and reward tuples as it needs to populate a replay buffer. Despite this mismatch, AuxSS closely matches the performance of HySAC.

5.4 Limitations and Discussion

In this chapter, we demonstrate the importance of auxiliary start state distributions, constructed by utilizing small quantities of expert demonstration comprising only state information, in facilitating sample-efficient learning of robust policies. We find that this is a crucial design choice in environments that allow arbitrary state resetting and we observe that deriving start state distributions from notions of state safety can dramatically accelerate policy learning online.

Naturally, the need for a simulator that supports arbitrary state resets is also a limitation. As shown by JSRL, it is possible to use guide policies learnt offline to reach exploration frontiers. However, learning such policies require having access to expert actions and rewards in addition to state information. Consequently, an interesting future direction would be to explore how goal conditioned policies can be learnt efficiently from expert data comprising only state information and thereby facilitating the use of auxiliary start states in broader set of tasks.

Finally, it can be noted that the manner in which safety cues have been incorporated into the proposed method solely utilize variable episode length, which is a consequence of episode terminations within the environment. Not all safety constraints need to be modelled through hard terminations and instead they can be

incorporated as costs that need to be minimized or constrained to remain below a certain value [1]. Generalizing the notion of safety to account for such formulations would be a natural extension to the proposed idea.

Chapter 6

Conclusion

In this thesis, we leverage commonly available affordances such as domain specific inductive biases, simulator resets and expert data to help shrink the search space while performing RL online. Designing algorithms that seamlessly incorporate these affordances, help guide online exploration and learn performant policies efficiently. To this end, we present to contrasting approaches to leverage such affordances. First, we highlight the efficacy of re-parameterizing the action space of navigation policies from instantaneous steer to a temporally abstract trajectory representation. This reduces the space of trajectories to reason over and results in learning better policies over an order of magnitude faster. Subsequently, we show how small quantities of expert demonstration comprising only state information, can facilitating sample-efficient learning of robust policies when presented with ability to reset environments to arbitrary states. We extensively evaluate these methods and demonstrate some promising results on a variety of environments spanning autonomous driving, navigation and robotic control task. Simultaneously, we identify the limitations of current methods and present promising future directions to take this work forwards.

Bibliography

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning*, 2017. 5.4

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. 2017. 2.3

[3] Glen Berseth, Daniel Geng, Coline Manon Devin, Nicholas Rhinehart, Chelsea Finn, Dinesh Jayaraman, and Sergey Levine. {SM}irl: Surprise minimizing reinforcement learning in unstable environments. In *International Conference on Learning Representations*, 2021. 2.1

[4] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. 2.1

[5] Felipe Codevilla, Eder Santana, Antonio M. L´opez, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. 2019. 4.1, 4.4

[6] Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. 1992. 2.3

[7] 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*, 2017. 1, 4.1, 4.3.1

[8] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return, then explore. *Nature*, 2021. 1, 2.1, 2.3

[9] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018. 1, 2.2, 5.3.2

[10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. 1, 2.1, 2.3, 5.3.2

[11] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John Agapiou, Joel Leibo, and Audrunas Gruslys. Deep q-learning from demonstrations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 2.3

[12] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 2016. 1, 2.2, 5.3.2

[13] Zhe Huang. Distributed reinforcement learning for autonomous driving. 2022. 4.1

[14] Arnav Kumar Jain, Lucas Lehnert, Irina Rish, and Glen Berseth. Maximum state entropy exploration using predecessor and successor representations. In *Advances in Neural Information Processing Systems*, 2023. 1

[15] Arnav Kumar Jain, Lucas Lehnert, Irina Rish, and Glen Berseth. Maximum state entropy exploration using predecessor and successor representations. In *Neural Information Processing Systems*, 2023. 2.1

[16] Chiyu Max Jiang, Andre Cornman, Cheolho Park, Ben Sapp, Yin Zhou, and Dragomir Anguelov. Motiondiffuser: Controllable multi-agent motion prediction using diffusion, 2023. 4.2

[17] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, 2002. 5.1

[18] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. 1, 2.2, 2.3, 5.3.2

[19] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020. 1, 2.2

[20] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. Should i run offline reinforcement learning or behavioral cloning? In *International Conference on Learning Representations*, 2022. 2.2

[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. 2.2

[22] Viraj Mehta, Biswajit Paria, Jeff Schneider, Willie Neiswanger, and Stefano Ermon. An experimental design perspective on model-based reinforcement learning. In *International Conference on Learning Representations*, 2022. 1, 2.1, 5.3.2

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 2015. 1

[24] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *IEEE international conference on robotics and automation*, 2018. 2.3

[25] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020. 2.3

[26] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, 2017. 1, 2.1

[27] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham Kakade. Towards Generalization and Simplicity in Continuous Control. In *Neural Information Processing Systems*, 2017. 3.1

[28] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *Proceedings of Robotics: Science and Systems*, 2018. 2.3

[29] Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. In *Advances in Neural Information Processing Systems*, 2021. 2.2

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2.1, 2.3, 4.3.1

[31] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration. In *International Conference on Machine Learning*, 2021. 1, 2.1

[32] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 2016. 1

[33] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017. 1

[34] Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. In *International Conference on Learning Representations*, 2021. 1, 2.3

[35] Yuda Song, Yifei Zhou, Ayush Sekhari, Drew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid RL: Using both offline and online data can make RL efficient. In *The Eleventh International Conference on Learning Representations*, 2023. 1, 2.1, 2.3, 5.3.2

[36] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012. 1, 5.3.1

[37] Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, Sergey Levine, and Karol Hausman. Jump-start reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. 1, 2.3, 5.3.2

[38] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017. 2.3