

# Mutual Information for Exploration in Reinforcement Learning

*Abhinav Gopal*



Electrical Engineering and Computer Sciences  
University of California, Berkeley

Technical Report No. UCB/EECS-2022-54

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-54.html>

May 10, 2022

Copyright © 2022, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

### Acknowledgement

Many thanks to Professor John Canny for his guidance over the last three years. He has been a mentor, teacher, and facilitator without whom I would not be able to complete this report. I would also like to thank Professor Ken Goldberg for his reading this report. Many thanks to Daniel Seita for his mentorship over this project, and also to Dhruv Jhamb, Darren Hsu, Arbaaz Muslim, and Dustin Luong, for their additional support over this project.

---

# Mutual Information for Exploration in Reinforcement Learning

by Abhinav Gopal

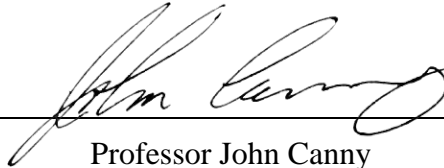
---

## Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences,  
University of California at Berkeley, in partial satisfaction of the requirements for the  
degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

### Committee:



---

Professor John Canny  
Research Advisor

5/6/2022

---

(Date)

\* \* \* \* \*



---

Professor Ken Goldberg  
Second Reader

5/5/2022

---

(Date)

Abstract

Mutual Information for Exploration in Reinforcement Learning

by

Abhinav Gopal

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Reinforcement Learning (RL) is a rapidly growing area of interest in the Artificial Intelligence community, with tremendous applications. As a result, there is a need to improve efficiency and exploration in RL algorithms to promote quicker and improved learning. We introduce MIREL: Mutual Information for Beneficial Exploration in RL, which considers the use of the mutual information between an action and the expected “future” from a given state as an additional reward to improve exploration. Using MIREL, agents learn to exploit “decision states” that lead to highly specialized futures.

# Contents

<b>Contents</b>	<b>i</b>
<b>List of Figures</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 A Note on Reinforcement Learning . . . . .	1
1.2 Motivation . . . . .	1
1.3 Problem Statement . . . . .	2
1.4 Hypothesis . . . . .	2
<b>2 Related Work</b>	<b>3</b>
2.1 Exploration . . . . .	3
2.2 Mutual Information . . . . .	4
2.3 Twin Delayed Deep Deterministic Policy Gradients . . . . .	4
2.4 Preliminaries . . . . .	5
<b>3 Approach</b>	<b>6</b>
3.1 Motivation . . . . .	6
3.2 MIRL: High Level Overview . . . . .	6
3.3 Mathematical Simplification of $MI(A, F S = s_i)$ . . . . .	6
3.4 Required Architecture/Models for $MI(A, F S)$ . . . . .	7
3.5 Implementation of $MI(A, F S = s_i)$ . . . . .	8
3.6 Determining Whether A State is A Decision State . . . . .	9
3.7 The actual algorithm . . . . .	10
3.8 Environments . . . . .	10
<b>4 Experiments</b>	<b>12</b>
<b>5 Experiment Analysis</b>	<b>14</b>
5.1 Proof of Concept for $\phi$ predictors . . . . .	14
5.2 Proof of Concept – Understanding $MI(A; F S = s_i)$ . . . . .	14
5.3 MIRL-TD3 vs. Traditional TD3 . . . . .	16
5.4 MIRL-TD3 vs. Traditional TD3 with Exploration Bonus . . . . .	16

<b>6 Conclusion</b>	<b>21</b>
<b>7 Limitations</b>	<b>22</b>
<b>8 Future Work</b>	<b>23</b>
<b>9 Appendix</b>	<b>24</b>
9.1 Predictors– Additional Information . . . . .	24
9.2 Mutual Information Calculation– Additional Information . . . . .	24
<b>Bibliography</b>	<b>26</b>

# List of Figures

3.1	This diagram shows a high level overview of the algorithm MIRL. . . . .	7
4.1	D4RL’s Large Maze Environment. . . . .	12
4.2	D4RL’s Medium Maze Environment. . . . .	13
4.3	D4RL’s U-Maze Environment. . . . .	13
5.1	Comparison between future predictor $\phi_{future}$ which predicts $P(A F, S)$ and past predictor $\phi_{past}$ which predicts $P(A S)$ in D4RL’s Large Maze environment. The future predictor is trained using the state, and the state 5 timesteps later. As we can see, the future predictor has an overall lower MSE than the past predictor. We report our average over 3 seeds. . . . .	15
5.2	Visualization of mutual information in D4RL’s large maze. Here, we cut off the values to be within the allowable range of -1 to 4. Some points are between 0 and -1 because the mutual information is an approximation and may come out to be slightly negative at times. . . . .	15
5.3	D4RL’s maze environment, with annotations for areas discussed in the paper. . . . .	16
5.4	Comparison of performance of MIRL-TD3 versus TD3 in D4RL’s fixed-goal medium maze environment without exploration bonuses. We smooth out the graph using a 15-epoch moving average. We report our average over 3 seeds. . . . .	17
5.5	Comparison of performance of MIRL-TD3 versus TD3 in D4RL’s fixed-goal large maze environment. We smooth out the graph using a 15-epoch moving average. We report our average over 3 seeds. . . . .	18
5.6	Comparison of performance of MIRL-TD3 versus TD3 in D4RL’s fixed-goal umaze environment without exploration bonuses. We smooth out the graph using a 15-epoch moving average. We report our average over 3 seeds. . . . .	19
5.7	Comparison of performance of MIRL-TD3 versus TD3 in D4RL’s random-goal medium maze environment. We smooth out the graph using a 15-epoch moving average. We report our average over 3 seeds. . . . .	19
5.8	Comparison of performance of MIRL-TD3 versus TD3 in D4RL’s random-goal large maze environment. We smooth out the graph using a 15-epoch moving average. We report our average over 3 seeds. . . . .	20



5.9	Comparison of performance of MIRL-TD3 versus TD3 in D4RL’s random-goal unmaze environment. We smooth out the graph using a 15-epoch moving average. We report our average over 3 seeds. . . . .	20
-----	---	----

## **Acknowledgments**

Many thanks to Professor John Canny for his guidance over the last three years. He has been a mentor, teacher, and facilitator without whom I would not be able to complete this report. I would also like to thank Professor Ken Goldberg for his reading this report. Many thanks to Daniel Seita for his mentorship over this project, and also to Dhruv Jhamb, Darren Hsu, Arbaaz Muslim, and Dustin Luong, for their additional support over this project.

# Chapter 1

## Introduction

### 1.1 A Note on Reinforcement Learning

Artificial Intelligence is a booming field in computer science today, aimed at creating autonomous agents to do human tasks, or even superhuman tasks, improving humanity broadly. Today, various methods of learning exist to help agents learn how to do tasks, and a popular field of interest is Reinforcement Learning.

Reinforcement Learning relies on experience-driven autonomous learning, similar to the way humans learn to do tasks from scratch. Through interactions with the actual environment/task of interest, agents learn in a self-critical manner, understanding optimal and sub-optimal behavior. Agents seek to learn a policy that strategically maximizes the expected “return” from an environment. Today, with the existence of deep neural networks, agents are able to also use function approximation in order for deep RL to be able to tackle complex, high-dimensional problems.

### 1.2 Motivation

Deep reinforcement learning has grown to be able to solve a variety of tasks in complicated environments in high dimensions. However, learning to perform well and efficiently explore has remained a difficult problem. Specifically, RL is still waiting for a clear and efficient method for exploration to discover effective policies. Currently, when faced with the choice of determining which states are worth exploring, agents frequently struggle, using seemingly ad hoc methods of exploration [4, 5, 1].

We focus on addressing the problem by taking advantage of the overall goal of exploration in RL: visiting states that have the possibility of learning new approaches that may yield high reward. To identify these new approaches, it is important to visit states that have diverse options for trajectories agents can take, each leading to a completely different

approach to solving the problem. These states are called “decision states.” Keeping this in mind, we introduce MIREL, Mutual Information for Beneficial Exploration in RL, which explores states based on whether they are decision states.

Our approach extends the work of [3] by using a more generic approach instead of goal conditioning.

### 1.3 Problem Statement

We use D4RL’s large-maze environment <sup>1</sup> as seen in ???. The agent has to reach the goal position from its initial starting position. We run experiments with a fixed goal position and randomized goal position. The benefit of using such maze environments is that they contain both paths of low mutual information and high mutual information. Put simply, they contain “branch points,” which are regions that have high mutual information since there are many possible paths/directions.

### 1.4 Hypothesis

We hypothesize that encouraging agents to take low mutual information paths unless the state is a local maximum of mutual information will result in efficient exploration that leads to faster learning and improved performance. We test this hypothesis using MIREL on top of Twin Delayed Deep Deterministic Policy Gradients (TD3), and comparing against vanilla TD3 [2].

Our hypothesis follows from the approach of incentivizing agents to learn task structure by training policies that will learn to perform well for a variety of end points, not specializing to any specific approach. At the same time, our approach incentivizes deviating from the default policy when visiting “decision states” that might lead to highly specialized futures.

---

<sup>1</sup><https://github.com/rail-berkeley/d4rl>

# Chapter 2

## Related Work

### 2.1 Exploration

[4] proposes a simple unified ensemble method to address standard off-policy RL algorithm issues such as unstable Q-learning and balancing exploration and exploitation. In terms of addressing exploration, SUNRISE uses an inference method that selects actions using the highest upper-confidence bounds for efficient exploration. This is done by defining an upper-confidence bound (UCB) based on the mean and variance of Q-functions. The inference method then selects actions with the highest UCB for efficient exploration as it provides a bonus for visiting unseen state-action pairs, where ensembles produce high uncertainty.

[5] proposes a “plan online and learn offline” (POLO) framework for the setting where an agent needs to continually act and learn in the world. They use model predictive control (MPC) for the agent to explore the space of trajectories. The agent considers a hypothesis of potential reward regions in the state space (generated by taking a Bayesian view and approximately tracking a posterior over value functions), and then execute the optimal trajectory conditioned on this belief. This results in a temporally coordinated sequence of actions which leads the agent to cover the state space more rapidly and intentionally (efficient exploration).

[1] shows that an ensemble of  $Q^*$ -functions can be used for effective exploration in deep RL by achieving significant gains on the Atari benchmark. Instead of using posterior sampling for exploration, the uncertainty estimates from the  $Q$ -ensemble are used. [1] proposes the UCB exploration strategy, which constructs uncertainty estimates of the  $Q$ -values. Agents take actions with the highest UCB.

[Tang:2017] introduces count-based exploration in reinforcement learning. In the paper, Tang shows how the generalization of traditional count based approaches can help RL agents reach state-of-the-art performance. Specifically, states are mapped to “hashes” which are then counted whenever the hashes are reached. Simple hashing results in excellent results.

With static hashing, Tang discretizes the state space, then adding an exploration bonus in the following fashion:

$$r^+(s) = \frac{\beta}{\sqrt{n(\phi(s))}}$$

Here,  $n(s)$  represents the number of times the "s" hash has occurred, or has been visited.

## 2.2 Mutual Information

Mutual information is a way to measure the dependence between two random variables. If  $(X, Y)$  are random variables with their joint distribution defined as  $P_{(X,Y)}$  and the marginal distributions  $P_X$  and  $P_Y$ , then mutual information can be defined as follows. The operation  $\otimes$  refers to the tensor product.

$$I(X; Y) = D_{KL}(P_{X,Y} || P_X \otimes P_Y)$$

For discrete distributions, we can rewrite this as

$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p_{(X,Y)}(x, y) \log\left(\frac{p_{(X,Y)}(x, y)}{p_X(x)p_Y(y)}\right)$$

In [3], agents are incentivized to learn task structure by training policies that perform well under a variety of goals, while not overfitting to any individual goal. The policy dependence on the individual goal is minimized using the conditional mutual information  $I(A; G|S)$ .

The conditional mutual information  $I(A; G|S)$  from [3] is defined as follows:

$$I(A; G|S) = \sum_{a,g,s} p(a, g, s) \log \frac{p(a|g, s)}{p(a|s)}$$

[3] also coins the term "decision states," which refer to states in which the agent learns to deviate from its habits. These states can be used as the basis for learning an exploration bonus that leads to more effective training than other task-agnostic exploration methods.

## 2.3 Twin Delayed Deep Deterministic Policy Gradients

[2] introduces Twin Delayed Deep Deterministic Policy Gradients (TD3), which builds on Double Q-Learning. TD3 addresses overestimation bias and the accumulation of error in temporal difference methods are present in an actor-critic setting. TD3 does this by using a pair of critic networks, delaying updates of the actor, and using action noise regulation. TD3 outperforms the state of the art in continuous control tasks from OpenAI gym, making it a suitable benchmark for us to use.

## 2.4 Preliminaries

In this paper, we investigate a method of solving maze-based Markov Decision Processes (MDPs) defined as follows:

$$M = (S, A, T, r, \gamma, \mu_0),$$

where  $S$  and  $A$  represent the state and action spaces,  $r$  denotes the reward function,  $T(s'|s, a)$  is dynamics transition function,  $\gamma \in [0, 1]$  is the discount factor, and  $\mu_0(s)$  is an initial state distribution. Actions  $a \in A$  can be performed in states  $s \in S$ , resulting in a reward  $r$ .

In this paper, we are also interested in the *future* of a particular state at timestep  $t$ ,  $s(t)$ . To represent the *future*, we take the state that exists a certain “lookahead” timesteps ahead of time  $t$ . This state,  $s(t+\text{lookahead})$  is referred to as the *future*.

The notation  $MI(A, F|S = s_i)$  refers to the mutual information between an action  $A$  and a future state  $F$  conditioned on the current state  $S$  being  $s_i$ .

# Chapter 3

## Approach

### 3.1 Motivation

With continuous reinforcement learning tasks, agents have a theoretically "infinite" number of decisions that they can make at any given state. To decrease the search space, as well as train time, minimizing the number of decisions that agents have to make is very important. Without this minimization, agents have more difficulty staying in "optimal paths" as they have to continue making correct decisions on every single small step. Through the use of MI, we intend on inducing the agent into traversing certain canonical paths in free space, regardless of where the agent is coming from. Agents only need to make decisions when they are required to, at "decision" or "branch" points. In effect, the infinite decision problem is reduced to a finite graph search problem.

### 3.2 MIRL: High Level Overview

At a high level, MIRL is implemented on top of existing reinforcement learning algorithms (in this case specifically TD3), and uses replay-buffer memory and gradient updates from buffer sampling. MIRL first takes samples from the buffer, calculates the mutual information  $MI(A, F|S = s_i)$ , and calculates whether each state is a "local maximum." When the state is not a local maximum,  $MI(A, F|S)$  is subtracted from the reward (using a factor). MIRL incentivizes beneficial exploration through reward shaping.

Refer to 3.1 for a high level overview of MIRL.

### 3.3 Mathematical Simplification of $MI(A, F|S = s_i)$

From the definition of mutual information,

$$MI(A, F|S = s_i)$$



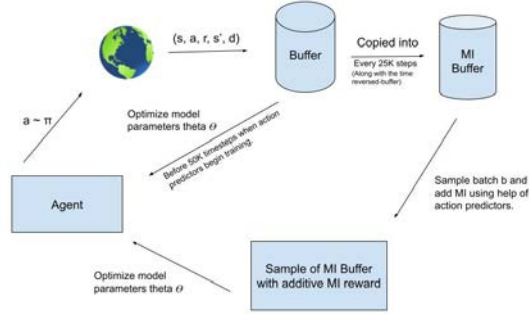


Figure 3.1: This diagram shows a high level overview of the algorithm MRL.

$$\begin{aligned}
 &= \sum_{a \in A} \sum_{f \in F} P(F, A|S = s_i) \log\left(\frac{P(F, A|S = s_i)}{P(F|S = s_i)P(A|S = s_i)}\right) \\
 &= \sum_{a \in A} \sum_{f \in F} P(F, A|S = s_i) \log\left(\frac{P(A|F, S = s_i)}{P(A|S = s_i)}\right) \\
 &= E_{A,F}(\log\left(\frac{P(A|F, S = s_i)}{P(A|S = s_i)}\right))
 \end{aligned}$$

We perform this simplification so that we can compute the quantity  $MI(A, F|S = s_i)$  using predictor models, as explained in the next section.

### 3.4 Required Architecture/Models for $MI(A, F|S)$

MRL requires the modification of existing replay buffers that keep track of states, next states, rewards, and done signals. Specifically, MRL also requires buffers to maintain the  $1.5 * k$  closest neighbors for each state. The necessity of these neighbors will become evident in the next section.

In addition to this, MRL requires the maintenance of the "time-reversed" observations as well. This time-reversal incorporates the knowledge that the agent in the maze could easily go in the exact opposite trajectory of one it took (i.e. backwards). In a maze with a state containing (position, velocity), an action ( $a$ ), and a next state (position', velocity'), the time reversal would amount to negating the action, reversing the velocities, and flipping the initial and next states.

To understand the need for the time reversal, consider a maze with a particular branch point that has 3 possible entrances. If an agent only encounters 1 entrance and exit in its episodes, it will not register the branch point as an area of high mutual information. With time reversal, this situation is always avoided!

Given a particular state, for the calculation of  $P(A|S)$ , MIRL also requires the maintenance of a predictor,  $\phi_{past}$ , which will calculate the probability of an action given a particular state.  $\phi_{past}$  is trained using data from the replay buffer that is generated with training.

Similarly, given a particular state and a future (represented as another state  $l$  steps ahead), MIRL requires the maintenance of a predictor,  $\phi_{future}$ , which will calculate  $P(A|F, S)$ .  $\phi_{future}$  is also trained using data from the replay buffer that is generated with training.

Both  $\phi_{past}$  and  $\phi_{future}$  are 2-layer MLPs with 32 neurons in each hidden layer. The MLPs output a singular scalar value to represent the mean of the action distribution, which has a constant variance of 0.1 for the past predictor, and 0.09 for the future predictor. In future work, we hope to eliminate these hard-coded variances entirely. When we trained the predictors using a network to estimate standard deviations, it resulted in poor performance for the  $MI$  calculation.

### 3.5 Implementation of $MI(A, F|S = s_i)$

The calculation of  $MI(A, F|S = s_i)$  comes from an expectation over the futures and actions at a particular state. To get the states, we randomly sample  $(s_i, a_i)$  tuples from the replay buffer to update on.

For each  $(s_i, a_i)$  tuple, we first calculate  $P(A|S)$  by directly using the predictor  $\phi_{past}$ .  $\phi_{past}$  generates a gaussian distribution over the action space, from which we can generate the probability density function (pdf) of the action of interest ( $a_i$ ).

Now, we move to the calculation of  $P(A|F, S = s_i)$ . In order to get a series of samples for the computation over the “expected” future, we calculate the  $k$  nearest neighbors for each state  $s_i$  in the sample from the replay buffer.

Since these states are very similar to the states  $s_i$ , the futures and actions taken at these states can be used to estimate  $MI(A, F|S = s_i)$ . For each state  $s_i$ , we generate  $s_i^{\prime 1} \dots s_i^{\prime k}$  matches. We get the action taken at these matches, and we also get the future from these matches, by checking  $l$  steps in the future. Specifically, for each match  $s_i^{\prime j}$ , which is found at the  $p$ 'th position in the replay buffer, we get the action by checking the  $p$ 'th position in the replay buffer's action buffer, and we get the future by finding the  $p + l$ 'th position in the

replay buffer’s observation buffer.

Now that we have  $k$  different  $(s'_i, a'_i)$  matches for each state  $s_i$ , we can calculate the pdf for each of the  $a'_i$  by passing them into  $\phi_{future}$ . After this, we have  $k$  values for  $P(A|F, S)$ . We combine these terms into one term by weighing them according to their closeness to the original state  $S = s_i$ .

Specifically, we compute the following:

$$P(A|F, S_i) = \frac{\sum_{q=1}^k (\Phi(s_i'^q) * \phi_{future}(a_i'^q))}{\sum_{q=1}^k (\Phi(s_i'^q))}$$

Where  $\Phi(s_i'^q)$  is the normal pdf of a gaussian centered at  $S_i$  with a predefined variance of 0.01 evaluated at  $\Phi(s_i'^q)$ .

Ultimately, putting together the calculated probabilities from our two predictors  $\phi_{past}$  and  $\phi_{future}$ , we can compute:

$$MI(A, F|S = s_i) = E_{A,F}(\log(\frac{P(A|F, S = s_i)}{P(A|S = s_i)}))$$

This quantity is the mutual information estimate  $MI(F, A|S = s_i)$ . Since mutual information is measured in terms of bits, we expect no more than 0 to 5 nats (units with a natural log, 1 bit = 0.693 nats) of mutual information at any given state. Since we are using an estimation, there may be some slightly negative mutual information points. In our several experiments, we have found that roughly 80% of  $MI$  values found in our method lie between -1 to 5 nats. For negative mutual information, we take those values to be 0.

## 3.6 Determining Whether A State is A Decision State

A decision state is a junction/state in the maze environment, wherein a specific action results in a highly specialized future. Importantly, different actions result in vastly different futures.

In order to determine whether a particular state is a “decision state” or “branch point,” we determine whether the state is a local maximum of mutual information. In the determination of whether a state is the local maximum, we compare the  $MI$  of the state with the  $MI$ s of its  $k$  closest neighbors. If it is larger than each of its neighbors, we consider it to be a local maximum.

### 3.7 The actual algorithm

The pseudocode for the algorithm is defined in 1. For our experiments, we use a value of 10 for  $k$ . We train for a total of 1M steps, and we begin training our predictor after 50K steps. We update our  $MI$  buffer every 25K steps.

---

**Algorithm 1:** MIRL
 

---

```

for  $t \in 1000000steps$  do – –
  Take environment steps by executing action according to algorithm’s distribution centered
  at  $\mu_\theta$ 
  observe next state  $s'$ , reward  $r$ , and done signal to indicate whether next state is terminal
  add  $(s', r, done)$  to replay buffer
  if time to update  $mi$  buffer or time to begin training predictors then
    copy buffer into  $mi$  buffer.
    calculate closest indices for  $mi$  buffer
  end if
  if time to do gradient updates then
    if predictors have already begun training then
      sample batch  $b$  from  $mi$  buffer.
      Update  $\phi_{past}$  and  $\phi_{future}$  using  $b$ 
      Calculate  $MI(A, F|S = s_i)$  for each state in  $b$ 
      if state  $s_i$  is not a local maximum of  $MI$ , and  $MI(A, F|S = s_i)$  is positive then
        subtract  $\alpha * MI(A, F|S = s_i)$  from  $r_i$ 
      end if
      Perform gradient updates using modified reward
    else
      sample batch  $b$  from replay buffer
      Perform gradient updates using  $b$ 
    end if
  end if

```

---

### 3.8 Environments

Environments for this paper were developed using D4RL’s pointmaze environments. Traditionally, pointmaze has a fixed goal with a varying start position. However, in order to show the utility of identifying ”decision points” with mutual information, the goal needs to be varied– if the goal is not varied, there is (almost) always one correct path that the agent will choose to take (the shortest distance to the goal). By varying the goal, we teach the agent to ensure that it can do a ”searching” operation. We proceeded to fix the agent’s initial state

to decrease the complexity of the Q function that the agent needs to model. We design a “large”, “umaze” and a “medium” maze to consider in this setting.

We use the sparse setting of the environment, wherein the agent receives no reward unless it is within a 0.5 unit radius of the goal, at which time the episode ends in success. Using a sparse reward setting makes the setting more realistic to moving goals in the real world, and also helps underscore the importance of MIRL having the ability to generate reward signals for learning in the absence of them.

Since traditional algorithms without any exploration bonus struggle to escape local maxima (well documented in literature), we proceed to use a count-based exploration bonus. This bonus is detailed in the appendix.

# Chapter 4

## Experiments



Figure 4.1: D4RL’s Large Maze Environment.

Our experiments are as follows:

1. We first confirm that predictors for actions given a particular state (for  $\phi_{past}$ ), or given a state and its future state (for  $\phi_{future}$ ) perform well.
2. We then investigate a proof of concept for our *MI* approach, checking whether mutual information quantities are clearly high at decision points, and lower in other areas. We visualize a *MI*-heatmap for D4RL’s maze environment with a fixed goal.

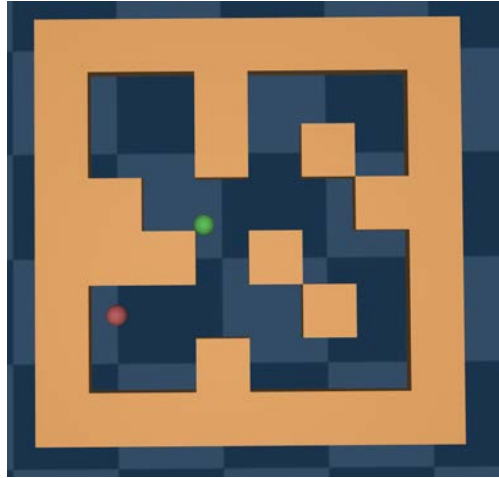


Figure 4.2: D4RL’s Medium Maze Environment.

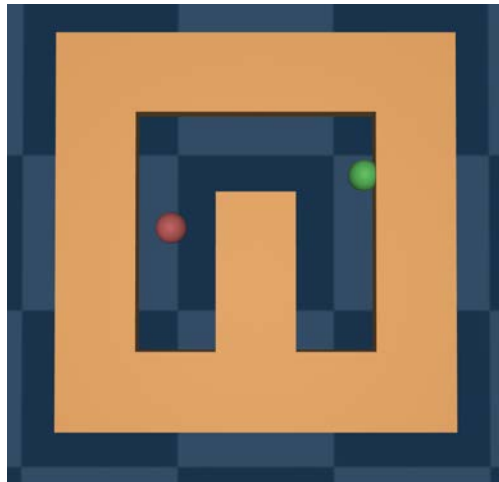


Figure 4.3: D4RL’s U-Maze Environment.

3. Next, we test MIRL using traditional TD3 against MIRL-TD3, which adds reward shaping on top of the traditional TD3 algorithm in the same fashion as highlighted in section IV: Approach and 1. We use openAI’s spinningup’s <sup>1</sup> implementation of TD3 for our experiments.
4. Finally, we test MIRL-TD3 with the additional count-based reward for exploration against vanilla TD3 with count-based reward exploration.

---

<sup>1</sup><https://github.com/openai/spinningup>

# Chapter 5

## Experiment Analysis

We break down our analysis into the following three parts:

1. Proof of concept for the predictors  $\phi_{past}$  and  $\phi_{future}$
2. Proof of concept – understanding  $MI(A, F|S = s_i)$
3. MIRL-TD3 vs. Traditional TD3

### 5.1 Proof of Concept for $\phi$ predictors

In order to calculate  $MI(A, F|S = s_i)$ , we need to ensure that we have usable methods of calculating  $P(A|F, S = s_i)$  and  $P(A|S = s_i)$ . To investigate this, we fully train a model using traditional TD3, adding 0.01 variance of noise on every dimension, and then sample from the replay buffer to train the predictors  $\phi_{past}$  and  $\phi_{future}$ . Our training curves for the mean squared error between the actions predicted and the real actions are presented in 5.1. We observe that as expected, the future predictor generally performs slightly better than the past predictor during the entire period of training. Both predictors are able to predict the action with 0.03 mean squared error.

### 5.2 Proof of Concept – Understanding $MI(A; F|S = s_i)$

First, we consider the proof of concept – to show that  $MI$  is high in “decision states,” and low in other areas. Visually, in a maze environment, we should see higher mutual information at branch points, and lower mutual information along certain corridors/walls. 5.2 is a visualization for the mutual information calculated in the maze based environment.

It is worth considering why the mutual information is distributed as shown in 5.2. As we can see, branch points, or “decision states,” seem to have extremely high values of mutual information. This is because at branch points, agents have many possible directions to move. At branch point 1, for example, as seen in 5.3, the agent has the option of moving in three



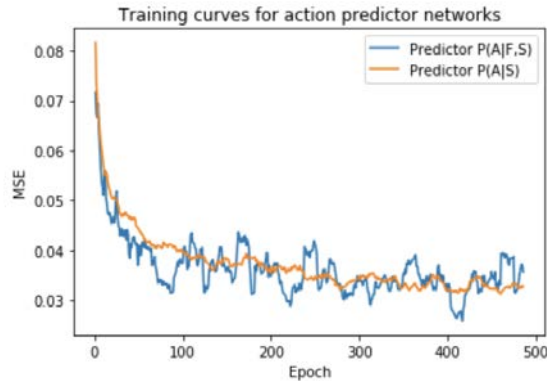


Figure 5.1: Comparison between future predictor  $\phi_{future}$  which predicts  $P(A|F, S)$  and past predictor  $\phi_{past}$  which predicts  $P(A|S)$  in D4RL’s Large Maze environment. The future predictor is trained using the state, and the state 5 timesteps later. As we can see, the future predictor has an overall lower MSE than the past predictor. We report our average over 3 seeds.

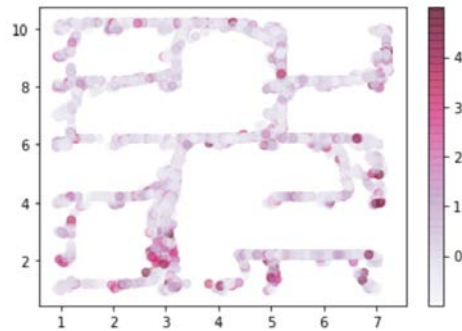


Figure 5.2: Visualization of mutual information in D4RL’s large maze. Here, we cut off the values to be within the allowable range of -1 to 4. Some points are between 0 and -1 because the mutual information is an approximation and may come out to be slightly negative at times.

distinct directions: up, left, or right. Given the future state, the agent knows the direction that was chosen. However, without this future state knowledge, the agent has three different options. As a result,  $P(A|F, S = s_i)$  is much higher than  $P(A|S = s_i)$  for these decision states, making the mutual information quantity  $MI(A, F|S = s_i)$  higher.

At states that are not branch points, and more like corridors, like point 2 in 5.3, the mutual information is extremely low, and that is because given the velocity of the agent (provided as a part of the state), the future of the agent is relatively known because of the

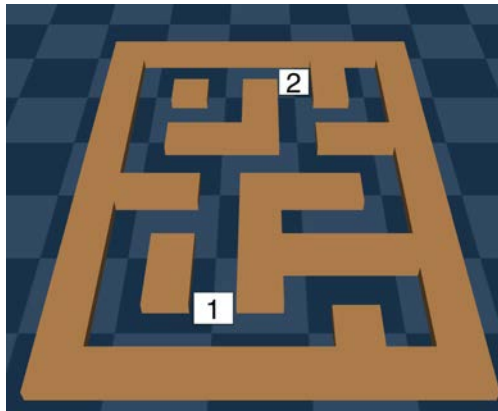


Figure 5.3: D4RL’s maze environment, with annotations for areas discussed in the paper.

lack of any other paths to explore. As a result,  $P(A|F, S = s_i)$  is not much higher than  $P(A|S = s_i)$  for such states, making the mutual information quantity  $MI(A, F|S = s_i)$  lower than the mutual information quantity at point 1.

### 5.3 MIRL-TD3 vs. Traditional TD3

In the next two sections, we discuss the experiments that compare our proposed approach MIRL-TD3 against traditional TD3.

In the fixed-goal settings, specifically the case where the goal is a fixed location, but the initialization of the agent is random, MIRL-TD3 performs comparably to traditional TD3. The reason for this is because TD3 agents without exploration bonuses find themselves stuck in local minima. Furthermore, the additional reward signal was also not useful enough because the agent would get stuck at the local minima, not allowing for optimal action prediction. Refer to figures 5.4, 5.5, and 5.6 for these.

### 5.4 MIRL-TD3 vs. Traditional TD3 with Exploration Bonus

Due to the issue of agents getting stuck in local optima, we moved to integrate a count-based exploration reward in the environment.

With the integration of a count-based exploration bonus, the performance of the agents drastically improved. We found that MIRL-TD3 slightly improved training time, but the overall affect was not pronounced in this setting. The reason for this was that the existence

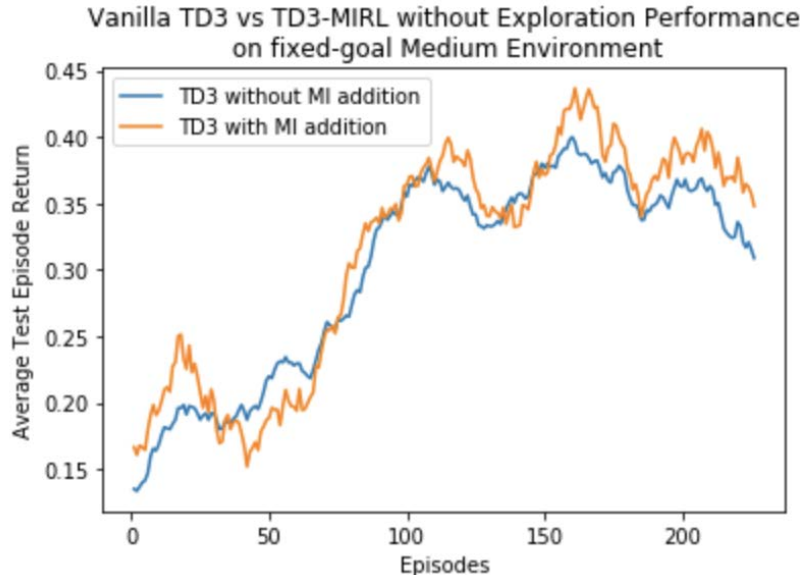


Figure 5.4: Comparison of performance of MIRL-TD3 versus TD3 in D4RL’s fixed-goal medium maze environment without exploration bonuses. We smooth out the graph using a 15-epoch moving average. We report our average over 3 seeds.

of a fixed goal decreased the importance of decision points as training went on. To provide some insight into this, consider the top right branch of the large-maze. When the maze consistently has a goal set at the absolute top right, an agent at the branch will always learn to go rightward. As a result of this, the mutual information quantity tends to 0 since the correct action is almost fully informed by just the current state.

Finally, we took to the setting of a variable goal, and a fixed initial point (to decrease the complexity of the problem that the Q-functions need to model). In this setting, we found that in the large and medium mazes, MIRL-TD3 resulted in quicker convergence, as well as better performance when compared to the vanilla TD3 agent (Figures 5.7, 5.8, 5.9). The agents learned to visit decision points and randomly fully explore certain paths from decision points before returning. In the u-maze, however, the improvement of MIRL-TD3 was not evident. This is attributed to the lack of decision points in that environment because of its simple structure as a "U".

For our other experiments that were attempted, as well as for the specific parameters used in our experiment, please refer to the appendix in Chapter 9.

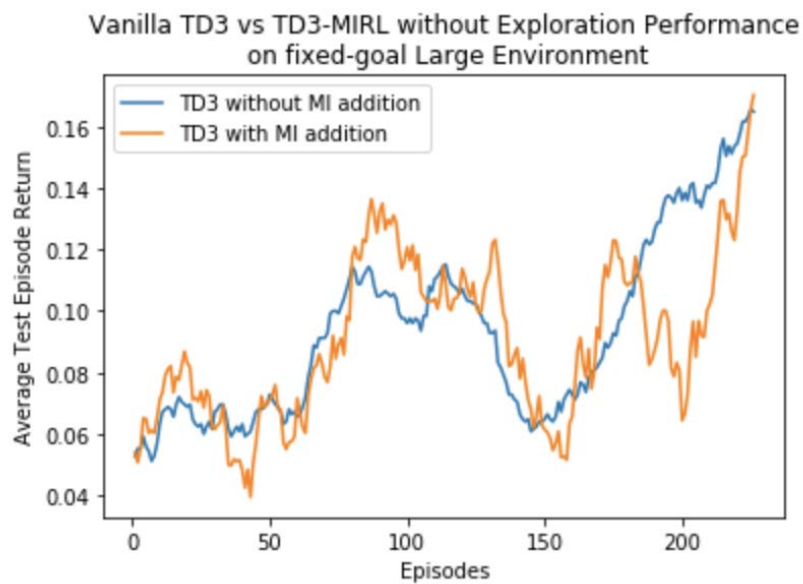


Figure 5.5: Comparison of performance of MRL-TD3 versus TD3 in D4RL’s fixed-goal large maze environment. We smooth out the graph using a 15-epoch moving average. We report our average over 3 seeds.

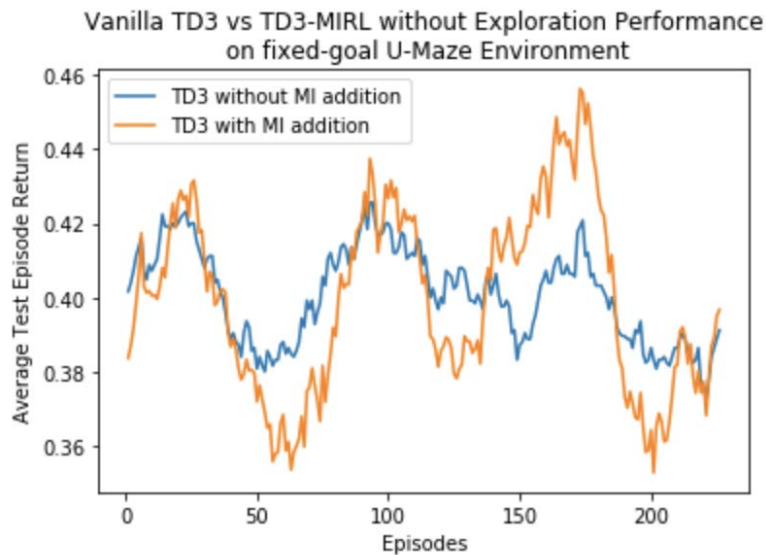


Figure 5.6: Comparison of performance of MIRL-TD3 versus TD3 in D4RL’s fixed-goal umaze environment without exploration bonuses. We smooth out the graph using a 15-epoch moving average. We report our average over 3 seeds.

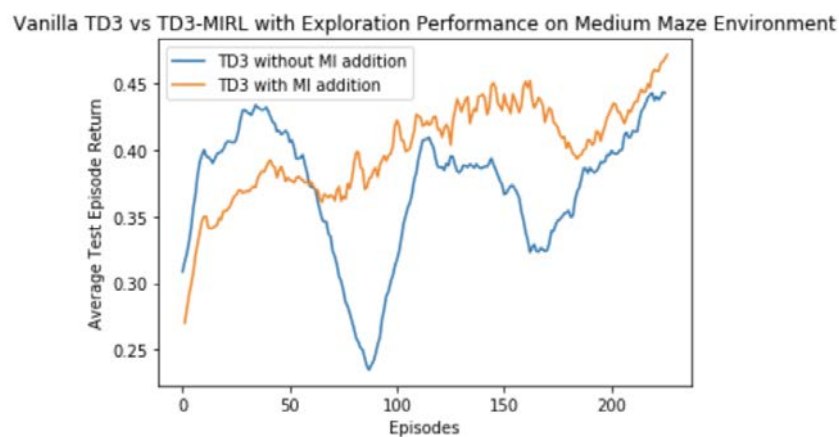


Figure 5.7: Comparison of performance of MIRL-TD3 versus TD3 in D4RL’s random-goal medium maze environment. We smooth out the graph using a 15-epoch moving average. We report our average over 3 seeds.

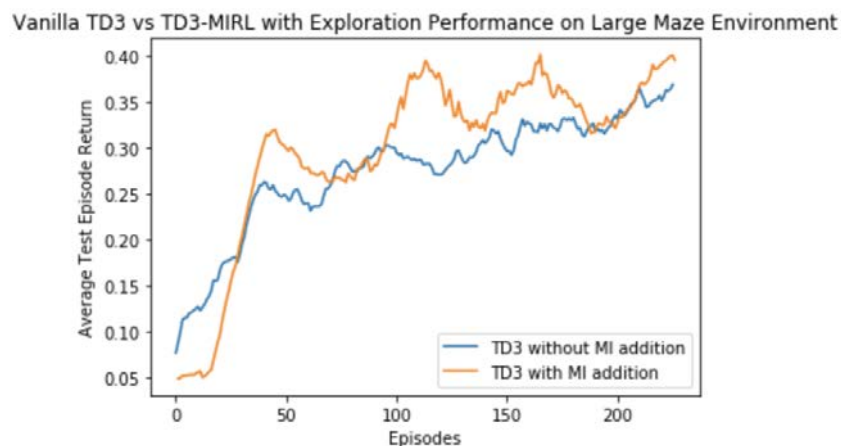


Figure 5.8: Comparison of performance of MIRL-TD3 versus TD3 in D4RL’s random-goal large maze environment. We smooth out the graph using a 15-epoch moving average. We report our average over 3 seeds.

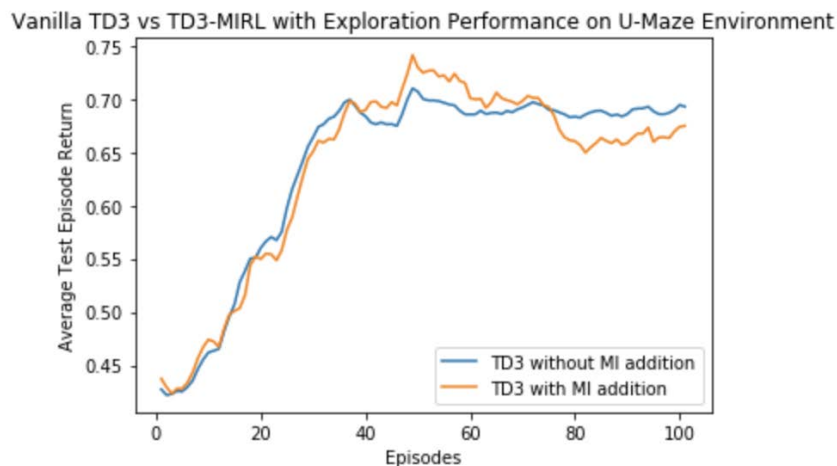


Figure 5.9: Comparison of performance of MIRL-TD3 versus TD3 in D4RL’s random-goal umaze environment. We smooth out the graph using a 15-epoch moving average. We report our average over 3 seeds.

# Chapter 6

## Conclusion

Within deep reinforcement learning, learning to efficiently explore complicated environments to solve a variety of tasks is an unsolved problem. Our motivation is to find a clear and efficient method for exploration to discover effective policies, improving on the work of [4, 1, 3, 5].

In order to facilitate efficient exploration, we propose the algorithm MIRL: Mutual Information for Beneficial Exploration in RL. MIRL involves using the mutual information between an action and a future from a given state in order to improve exploration. Agents are incentivized to visit “decision states” that lead to highly specialized futures.

We hypothesized that encouraging agents to take low mutual information paths unless the state is a local maximum of mutual information will result in efficient exploration that leads to faster learning and improved performance when compared to TD3. From our experiments we found that in the case of a maze with a random goal position with an exploration incentive, MIRL helps with quicker convergence and finds optimal behavior by leveraging decision points in the maze space. However, in the case of a fixed goal position, MIRL’s benefits were not as pronounced.

# Chapter 7

## Limitations

There are a few limitations of our work which we explore below.

1. Computational complexity and memory overhead: With MIRL, replay buffers are required to maintain closest neighbors, and mutual information is calculated for each state that is being updated on. The calculation of nearest neighbors for up to 1M points in the replay buffer takes significant amount of time, and even with approximate nearest neighbors, there is still an additional overhead that is noticeable. This is something that needs to be managed, especially in high dimensional environments.
2. Mutual Information in non-maze environments: It is easy to understand the notion of decision states from the perspective of maze-based environments as branch points, but in non-maze based environments, it is unclear whether this approach is useful.
3. Tuning at Environment-level: The MIRL algorithm also incorporates the “time-reversal” of states so that MI can be calculated for them, which requires the knowledge of what a reverse time trajectory looks like. In the maze, for example, this was tantamount to the negation of the action, the negation of the velocity, and the switching of the next state and the current state. There is some thought that is required to understand which parts of the state should be negated/changed to mimic the time-reversal.



# Chapter 8

## Future Work

In future work, we intend on applying MIRL to other reinforcement learning algorithms aside from TD3. Perhaps, we may even attempt to use MIRL to see results in offline reinforcement learning.

Furthermore, it should also be noted that MIRL relies on the maintenance of nearest neighbors for points in its replay buffer. As a result, when the state-space gets even larger (like with 100 dimensions), this nearest-neighbors array will be very expensive to keep track of. It would be interesting to combine MIRL with an encoder for the state space to decrease dimensionality and avoid this issue. This is a definite avenue for future work!

Finally, we also intend on applying MIRL to non maze-based environments in the future. Specifically, we hope that MIRL can be used in environments even involving coordination tasks, or other continuous control tasks like the ones part of the MuJoCo suite.

# Chapter 9

## Appendix

### 9.1 Predictors— Additional Information

When we first began the approach of using predictors to calculate  $P(A|S)$  and  $P(A|F, S)$  we attempted to use batch reinforcement learning in order to train an actor of a reinforcement learning agent. Unfortunately, this proved to be quite difficult, with significant mean-squared-error values. We then moved to a direct Multi-layer perceptron model, which proved to be more computationally efficient, and also better-performing.

Regarding the parameters used for the perceptron, for both predictors, we used a 2-layer neural net of 256 neurons each. We used an adam optimizer with a 0.001 learning rate, and a  $10^{-5}$  weight decay.

Regarding the way the predictors were trained, we train them using 100-state size batches at regular intervals during training, which happen after a burn-in period. We have a burn in period, because without this, the actor has not yet learned useful methods of tackling the maze, which means the "action" predictors will be relatively random.

### 9.2 Mutual Information Calculation— Additional Information

Since the calculation of Mutual Information in this project is quite dense, it is explained in detail in this section.

We seek to calculate the quantity  $MI(A; F|S = s_i)$ . First, we sample the replay buffer, getting a batch of states. Let's consider a single state  $s_i$  and its corresponding action  $a_i$  as doing the calculation for all states is simply vectorized.

NOTE: at certain intervals, the replay buffer calculates its closest matches for all indices using FAISS. For this project, we use 25000 episodes. As a result of this, at timestep  $t$ , the

replay buffer always has its closest matching states from  $0 \dots t - 25K$ .

With state  $s_i$ , we first calculate  $P(A = a_i | S = s_i)$  by passing the state through the  $\phi_{past}$  predictor which outputs a specific distribution over the action space, with a standard deviation that is preset to 0.01. Then, we calculate the probability of taking action  $a_i$  at this state. This gives us the  $P(A|S = s_i)$  quantity.

For the  $P(A|S = s_i, f = s_{i+l})$ , where  $l$  is the lookahead number of states, we calculate this quantity by first finding states that are close "matches" of  $s_i$ . To do this, we take a look at our replay buffer, which keeps track of the  $1.5 * k$  closest states to  $s_i$ . We then ensure that for each of these  $1.5 * k$  matching states, there exists a future state at the specified lookahead. To tangibly put this, suppose we have a matching state at index  $q$ . We ensure that index  $q + l$  also has a state that belongs to the same episode as index  $q$ . If this is the case,  $s_q$  is a valid match for  $s_i$ . After keeping the valid matches, we truncate the valid matches to only  $k$  matches.

Note: The probability of having less than  $k$  valid matches after storing  $1.5k$  possible matches initially, with  $k=10$  is as follows:

$$1 - P(15Valid) - P(14valid) - P(13valid) - P(12valid) - P(11valid) - P(10valid)$$

The probability of a particular index being invalid is  $5/1000$ , because only the last 5 states of an episode are invalid (1000 is the episode length). Using a simple binomial distribution, this calculation works out to  $7.5^{-11}$ , which is very negligible.

With the  $k$  matches, we now get the actions  $a_q$  that correspond to each of the matches. We calculate  $P(A = a_q | S = s_q, F = s_{q+l})$  for each of the matches by passing them through the  $\phi_{future}$  network. We then weight these matches according to their L-2 distance from  $s_i$ .

As mentioned earlier, we compute the following:

$$P(A|F, S_i) = \frac{\sum_{q=1}^k (\Phi(s_i'^q) * \phi_{future}(a_i'^q))}{\sum_{q=1}^k (\Phi(s_i'^q))}$$

Where  $\Phi(s_i'^q)$  is the normal pdf of a gaussian centered at  $S_i$  with a predefined variance of 0.01 evaluated at the  $q$ th match of  $s_i$ .

Ultimately, putting together the calculated probabilities from our two predictors  $\phi_{past}$  and  $\phi_{future}$ , we can compute:

$$MI(A, F | S = s_i) = E_{A, F}(\log(\frac{P(A|F, S = s_i)}{P(A|S = s_i)}))$$

# Bibliography

- [1] Richard Y. Chen et al. “UCB and InfoGain Exploration via *nboldsymbol{Q}*-Ensembles”. In: *CoRR* abs/1706.01502 (2017). arXiv: 1706.01502. URL: <http://arxiv.org/abs/1706.01502>.
- [2] Scott Fujimoto, Herke van Hoof, and David Meger. “Addressing Function Approximation Error in Actor-Critic Methods”. In: *CoRR* abs/1802.09477 (2018). arXiv: 1802.09477. URL: <http://arxiv.org/abs/1802.09477>.
- [3] Anirudh Goyal et al. *InfoBot: Transfer and Exploration via the Information Bottleneck*. 2019. arXiv: 1901.10902 [stat.ML].
- [4] Kimin Lee et al. “SUNRISE: A Simple Unified Framework for Ensemble Learning in Deep Reinforcement Learning”. In: *CoRR* abs/2007.04938 (2020). arXiv: 2007.04938. URL: <https://arxiv.org/abs/2007.04938>.
- [5] Kendall Lowrey et al. “Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control”. In: *CoRR* abs/1811.01848 (2018). arXiv: 1811.01848. URL: <http://arxiv.org/abs/1811.01848>.