UNIVERSITY OF CALIFORNIA SAN DIEGO

State-based Policy Representation for Deep Policy Learning

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Fangchen Liu

Committee in charge:

Professor Hao Su, Chair
Professor Sicun Gao
Professor Zhuowen Tu

2020

The thesis of Fangchen Liu is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am very fortunate to join the lab led by Prof. Hao Su in 2018, where I got the courage and inspiration to explore many exciting and fundamental problems in the field of artificial intelligence. Hao is a brave explorer and pure researcher with curiosity on fundamental problems, who has served as my role model during my academic studies. From Hao, I have learned to be passionate, dedicated, and ambitious in research and in life.

I am very lucky to know my great labmates, who are my collaborators and also my close friends, particularly, Zhiao Huang, Jiayuan Gu, Tongzhou Mu, Zhiwei Jia, Shuang Liu, Zhan Ling, Fanbo Xiang, Yuzhe Qin, and many others. They are indeed honest, dedicated and solid, and our collaborations have enhanced my skills and enlarged my research views. I also owe enormous thanks to other friends and researchers who have offered me incredible support, especially Huazhe Xu, Dequan Wang, Xinyue Zhang and Siyu Yao. Also, I am very grateful to my parents, for their unconditional love, trust and support.

I would thank so many exceptional professors at UC San Diego, such as Prof. Zhuowen Tu, Prof. Xiaolong Wang, Prof. Sicun Gao and Prof. Ravi Ramamoorthi, who have offered me helps and suggestions in my graduate studies, and also demonstrated what great researchers are like. The experience in UC San Diego will always be the greatest treasure in my life.

Chapter 2, in full, is a material found in the Thirty-third Conference on Neural Information Processing Systems 2019. Huang, Zhiao; Liu, Fangchen; Su, Hao. Mapping State Space Using Landmarks for Universal Goal Reaching. The thesis author was the co-first author of this paper.

Chapter 3, in full, is a material found in the Eighth International Conference on Learning Representations 2020. Liu, Fangchen; Ling, Zhan; Mu, Tongzhou; Su, Hao. State Alignment-based Imitation Learning. The thesis author was the first author of this paper.

VITA

2014-2018	B. S. in Computer Science *summa cum laude*, Peking University, China
2018-2020	M. S. in Computer Science, University of California San Diego

PUBLICATIONS

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, **Fangchen Liu**, Minghua Liu, Hanxiao Jiang, Yifu Yuan, Li Yi, He Wang, Angel Chang, Leonidas Guibas, Hao Su, "SAPIEN: a SimulAted Part-based Interactive ENvironment", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Fangchen Liu, Zhan Ling, Tongzhou Mu, Hao Su, "State Alignment-based Imitation Learning", *Eighth International Conference on Learning Representations (ICLR)*, 2020.

Zhiao Huang[†], **Fangchen Liu**[†], Hao Su, "Mapping State Space Using Landmarks for Universal Goal Reaching", *Thirty-third Conference on Neural Information Processing Systems (NeurIPS)*, 2019. ([†] indicates equal contribution)

Bo Sun, Nian-hsuan Tsai, **Fangchen Liu**, Ronald Yu, Hao Su, "Adversarial Defense by Stratified Convolutional Sparse Coding", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

ABSTRACT OF THE THESIS

State-based Policy Representation for Deep Policy Learning

by

Fangchen Liu

Master of Science in Computer Science

University of California San Diego, 2020

Professor Hao Su, Chair

Reinforcement Learning has achieved noticeable success in many fields, such as video game playing, continuous control, and the game of Go. One the other hand, current approaches usually require large sample complexity, and also lack the transferability to similar tasks. Imitation learning, also known as “learning from demonstrations”, is possible to mitigate the former problem by providing successful experiences. However, current methods usually assume the expert and imitator are the same, which lack flexibility and robustness when the dynamics change.

Generalizability is the core of artificial intelligence. An agent should be able to apply its knowledge for novel tasks after training in similar environments, or providing related demonstrations. Given current observation, it should have the ability to predict what can happen

(modeling), and what need to happen (planning). This brings out challenges on how to represent the knowledge and how to utilize the knowledge by learning from interactions or demonstrations.

In this thesis, we will systematically study two important problems, the universal goal-reaching problem and the cross-morphology imitation learning problem, which are representative challenges in the field of reinforcement learning and imitation learning. Laying out our research work that attends to these challenging tasks unfolds our roadmap towards the holy-grail goal: make the agent generalizable by learning from observations and model the world.

Chapter 1

Introduction

Learning a good policy for complicated tasks has attracted a lot of attention in recent years, with the rising trends and noticeable capacity of deep reinforcement learning. Some methods are value-centric by utilizing Bellman equations, while other methods focus more on directly optimizing policy, or combine value and policy (also known as actor-critic methods). From the standard definition for a sequential decision-making process used in the above methods, a policy π is defined as a probabilistic distribution of action a at certain state s, i.e. $\pi(a|s)$, which can be directly optimized or induced from value function.

While π can be regarded as certain action distribution every step, one can also rethink the essence of π from a high-level perspective. To fulfill an ultimate goal, there are certain pre-conditions that should be satisfied. For example, to use a coffee machine, a robot needs to grasp a cup, put the cup under the machine outlet, and then push some buttons. These "milestones" are subgoals for the final challenging goal, and need to be finished sequentially by the learned policy. Thus, a successful policy needs to figure out the dependency and relationships of these subgoals purely from interactions.

Reinforcement learning methods usually require massive explorations to discover such a structure of goals, especially when the state and action spaces are large and episodes are long.

That is quite common in many RL problems that we are interested in, such as robot learning and video game playing. To solve this problem, researchers introduce a hierarchical structure and extend the current RL framework, which will be briefly reviewed in Chapter 2.

Imitation Learning can mitigate the exploration problem by learning from successful demonstrations. The sample complexity can be largely reduced if the agent can mimic the expert's behavior and reproduce the trajectory. Some methods use behavior cloning to memorize the action sequence in a supervised way, while another important line in imitation learning is inverse reinforcement learning, which learns the imitator policy in another RL problem, whose rewards are generated to encourage the current policy and to approach the expert policy. These methods will be reviewed in Chapter 3. For all the above approaches, there is a natural assumption that the expert and the imitator are the same kinds of agents. Therefore, directly copying actions or matching the distribution of actions will produce a good imitation policy, even without utilizing subgoals. However, when it comes to a more challenging setting, where agent dynamics are different but the task remains the same, all the methods cannot be directly applied.

How can a 7-DoF robot arm learn to manipulate objects from a human video demonstration? For the robot, a lot of important subgoals should be accomplished in the same way as the human expert, even though the actions to reach them are never the same. In this setting, the imitation policy can also be regarded as transitions between subgoals in the demonstration, but no longer from an action distribution perspective.

In this thesis, we will introduce two papers that systematically study the above problems in reinforcement learning and imitation learning, and use a state-based high-level policy representation for planning and cross-morphology imitation. These representative states selected from experience or demonstration are a compact representation of a successful policy, as well as an abstraction for the environment or task, and can also be regarded as some knowledge representation during the learning process.

Chapter 2

Reinforcement Learning with State-based Environment Model

2.1 Introduction

Reinforcement learning (RL) allows training agents for planning and control tasks by feedbacks from the environment. While significant progress has been made in the standard setting of achieving a goal known at training time, e.g., to reach a given flag as in MountainCar [Moo90], very limited efforts have been exerted on the setting when goals at evaluation are unknown at training time. For example, when a robot walks in an environment, the destination may vary from time to time. Tasks of this kind are unanimous and of crucial importance in practice. We call them Universal Markov Decision Process (UMDP) problems following the convention of [LPS18].

Pioneer work handles UMDP problems by learning a *Universal Value Function Approximator* (UVFA). In particular, [SHGS15] proposed to approximate a goal-conditioned value function $V(s,g)$[1] by a multi-layer perceptron (MLP), HER [AWR^{+}17] proposed a framework called *hindsight experience replay* (HER) to smartly reuse past experience to fit the universal

[1]s is the current state and g is the goal.

value function by TD-loss. However, for complicated policies of long-term horizon, the UVFA learned by networks is often not good enough.

This is because UVFA has to memorize the cumulative reward between all the state-goal pairs, which is a daunting job. In fact, the cardinality of state-goal pairs grows by a high-order polynomial over the horizon of goals.

While the general UMDP problem is extremely difficult, we consider a family of UMDP problems whose state space is a low-dimension manifold in the ambient space. Most control problems are of this type and geometric control theory has been developed in the literature [BL04].

Our approach is inspired by manifold learning, e.g., Landmark MDS [DST04]. We abstract the state space as a small-scale map, whose nodes are landmark states selected from the experience replay buffer, and edges connect nearby nodes with weights extracted from the learned local UVFA. A network is still used to fit the local UVFA accurately. The map allows us to run high-level planning using pairwise shortest path algorithm, and the local UVFA network allows us to derive an accurate local decision. For a long-term goal, we first use the local UVFA network to direct to a nearby landmark, then route among landmarks using the map towards the goal, and finally reach the goal from the last landmark using the local UVFA network.

Our method has improved sample efficiency over purely network learned UVFA. There are three main reasons. First, the UVFA estimator in our framework only needs to work well for local value estimation. The network does not need to remember for faraway goals, thus the load is alleviated. Second, for long-range state-goal pairs, the map allows propagating accurate local value estimations in a way that neural networks cannot achieve. Consider the extreme case of having a long-range state-goal pair never experienced before. A network can only guess the value by extrapolation, which is known to be unreliable. Our map, however, can reasonably approximate the value as long as there is a path through landmarks to connect them. Lastly, the map provides a strong exploration ability and can help to obtain rewards significantly earlier, especially in the sparse reward setting. This is because we choose the landmarks from the replay buffer using a farthest-point sampling strategy, which tends to select states that are closer to the boundary of the visited space. In experiments, we compared our methods on several challenging environments and have outperformed baselines.

Our contributions are: First, We propose a sample-based method to map the visited state space using landmarks. Such a graph-like map is a powerful representation of the environment, maintains both local connectivity and global topology. Second, our framework will simultaneously map the visited state space and execute the planning strategy, with the help of a locally accurate value function approximator and the landmark-based map. It is a simple but effective way to improve the estimation accuracy of long-range value functions and induces a successful policy at the early stage of training.

2.2 Related work

Variants of goal-conditioned decision-making problems have been studied in literature [SMD$^+$11, MDL18, SHGS15, PGDL18]. We focus on the goal-reaching task, where the goal is a subset of the state space. The agent receives meaningful rewards if and only if it has reached the goal, which brings significant challenges to existing RL algorithms. A significant recent approach along the line is Hindsight Experience Replay (HER) by [AWR$^+$17]. They proposed to relabel the reached states as goals to improve data efficiency. However, they used only a single neural network to represent the Q value, learned by DDPG [LHP$^+$15]. This makes it hard to model the long-range distance. Our method overcomes the issue by using a sample-based map to represent the global structure of the environment. The map allows to propagate rewards to distant states more efficiently. It also allows to factorize the decision-making for long action sequences into a high-level planning problem and a low-level control problem.

Model-based reinforcement learning algorithms usually need to learn a local forward model of the environment, and then solve the multi-step planning problem with the learned model

[HLF+18, OSL17, SHM+16, HWL17, SJA+18, YSSF19]. These methods rely on learning an accurate local model and require extra efforts to generalize to the long term horizon [KST+18]. In comparison, we learn a model of environment in a hierarchical manner, by a network-based local model and a graph-based global model (map). Different from previous works to fit forward dynamics in local models, our local model distills local cumulative rewards from environment dynamics. In addition, our global model, as a small graph-based map that abstracts the large state space, supports reward propagation at long range. One can compare our framework with Value Iteration Networks (VIN) [TWT+16]. VIN focused on the 2D navigation problem. Given a predefined map of known nodes, edges, and weights, it runs the value iteration algorithm by ingeniously simulating the process through a convolutional neural network [LBBH98]. In contrast, we construct the map based upon the learned local model.

Sample-Based Motion Planning (SBMP) has been widely studied in the robotics context [HNR68, LaV98, KSO94]. The traditional motion planning algorithm requires the knowledge of the model. Recent work has combined deep learning and deep reinforcement learning for [IP18, QY18, KB19, FOR+18]. In particularly, PRM-RL addressed the 2D navigation problem by combining a high-level shortest path-based planner and a low-level RL algorithm. To connect nearby landmarks, it leveraged a physical engine, which depends on sophisticated domain knowledge and limits its usage to other general RL tasks. In the general RL context, our work shows that one can combine a high-level planner and a learned local model to solve RL problems more efficiently. Some recent work also utilize the graph structure to perform planning [SDK18, ZLS+18], however, unlike our approach that discovers the graph structure in the process of achieving goals, both [SDK18, ZLS+18] require supervised learning to build the graph. Specifically, [SDK18] need to learn a Siamese network to judge if two states are connected, and [ZLS+18] need to learn the state-attribute mapping from human annotation.

Our method is also related to hierarchical RL research [LPS18, KNST16, NGLL18b]. The sampled landmark points can be considered as sub-goals. [LPS18, NGLL18b] also used

HER-like relabeling technique to make the training more efficient. These work attack more general RL problems without assuming much problem structure. Our work differs from previous work in how high-level policy is achieved. In their methods, the agent has to learn the high-level policy as another RL problem. In contrast, we exploit the structure of our universal goal reaching problem and find the high-level policy by solving a pairwise shortest path problem in a small-scale graph, thus more data-efficient.

2.3 Background

2.3.1 Reinforcement Learning and Notation

First, we introduce the standard reinforcement learning setting. For simplification, we assume that the environment is fully observable, which can be described as a *Markov Decision Process* (MDP) defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0, \gamma \rangle$. Here, $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space, respectively, $\mathcal{P}$ represents the transition probability $\mathcal{P}(s'|s,a)$, $\mathcal{R}$ is the reward function, ρ_0 is the distribution of the initial state, and $\gamma \in (0,1]$ is the discounting factor.

The policy π of an agent maps the state space to an action $\pi : \mathcal{S} \to \mathcal{A}$. The goal of reinforcement learning is to find an optimal policy π^* that maximizes the expectation of the accumulated future rewards $J(\pi) = E_{\rho_0,\pi}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})]$, according to the initial state distribution ρ_0.

We also define the state value function V_π as $V_\pi(s) = E_\pi[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})|s_0 = s]$. The optimal value function V^* satisfies $V^*(s) = \max_\pi E_\pi[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})|s_0 = s]$ for any $s \in \mathcal{S}$, and the policy π^* to make $V_{\pi*}(s) = V^*(s)$ for every $s \in \mathcal{S}$, is called optimal policy.

Similarly, we define the Q function $Q_\pi(s,a) = E_\pi[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})|s_0 = s, a_0 = a]$ and $Q^*(s,a) = \max_\pi E_\pi[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})|s_0 = s, a_0 = a]$. It is easy to show the following equation holds, which is called the *Bellman Equation*:

$$Q(s,a) = r(s,a) + \gamma E_{s' \sim \mathcal{P}(s'|s,a)} \max_{a'} Q(s',a')$$

An important algorithm based on the *Bellman Equation* to calculate the V^* and π^* is value iteration:

$V_{n+1}(s) = \max_a Q_n(s,a), \forall s$ where $Q_n(s,a) = r(s,a) + \gamma E_{s'} \mathcal{P}(s'|s,a) V_n(s')$

It is well known that the value function V_n converges to V^* as $n \to \infty$.

2.3.2 Universal Markov Decision Process (UMDP)

Universal Markov Decision Process (UMDP) extends an MDP with a set of goals $\mathcal{G}$. UMDP has reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathcal{R}$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. Every episode starts with a goal selected from $\mathcal{G}$ by the environment and is fixed for the whole episode.

We aim to find a goal conditioned policy $\pi : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$ to maximize the expected cumulative future return $V_{g,\pi}(s_0) = E_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, g)]$, which called goal-conditioned value, or universal value. Universal Value Function Approximators (UVFA) [SHGS15] use neural network to model $V(s,g) \approx V_{g,\pi^*}(s)$ where π^* is the optimal policy, and apply Bellman equation to train it in a bootstrapping way. Usually, the reward in UMDP is sparse to train the network. For a given goal, the agent can receive non-trivial rewards only when it can reach the goal. This brings a challenge to the learning process.

Hindsight Experience Replay (HER) [AWR+17] proposes goal-relabeling to train UVFA in sparse reward setting. The key insight of HER is to "turn failure to success", i.e., to make a failed trajectory become success, by replacing the original failed goals with the goals it has achieved. This strategy gives more feedback to the agent and improves the data efficiency for sparse reward environments. Our framework relies on HER to train an accurate low-level policy.

2.4 Universal Goal Reaching

Problem Definition: Our universal goal reaching problem refers to a family of UMDP tasks. The state space of our UDMP is a low-dimension manifold in the ambient space. Many useful planning problems in practice are of this kind. Example universal goal reaching environments include labyrinth walking (e.g., AntMaze [DCH+16]) and robot arm control (e.g., FetchReach [PAR+18]). Their states can only transit in a neighborhood of low-dimensionality constrained by the degree of freedom of actions.

Following the notions in Sec 2.3.2, we assume that a goal g in goal space $\mathcal{G}$ which is a subset of the state space $\mathcal{S}$. For example, in a labyrinth walking game with continuous locomotion, the goal can be to reach a specific location in the maze at any velocity. Then, if the state s is a vector consisting of the location and velocity, a convenient way to represent the goal g would be a vector that only contains the dimensions of location, i.e., the goal space is a projection of the state space.

The universal goal reaching problem has a specific transition probability and reward structure. At every time step, the agent moves into a local neighborhood based on the metric in the state space, which might be perturbed by random noise. It also receives some negative penalty (usually a constant, e.g., -1 in the experiments) unless it has arrived at the vicinity of the goal. A 0 reward is received if the goal is reached. To maximize the accumulated reward, the agent has to reach the goal in fewest steps. Usually the only non-trivial reward 0 appears rarely, and the universal goal reaching problem falls in the category of *sparse reward* environments, which are hard-exploration problems for RL.

A Graph View: Assume that a policy π takes at most steps T to move from s to g and the reward at each step r_k's absolute value is bounded by R_{max}. Let $w_\pi(s,t)$ be the expected total reward along the trajectory, and $d_\pi(s,t) = -w_\pi(s,t)$ for all s,t. If $\gamma \approx 1$, we can show that UVFA

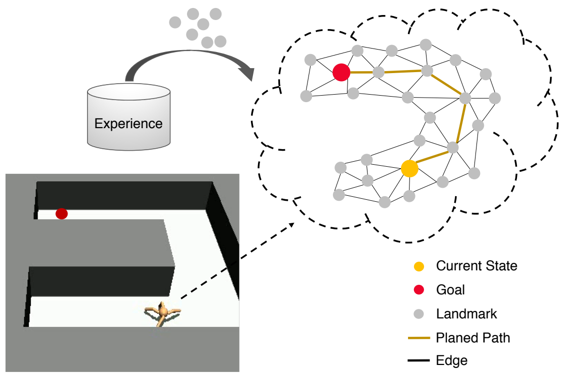


Figure 2.1: An illustration of our framework. The agent is trying to reach the other side of the maze by planning on a landmark-based map. The landmarks are selected from its past experience, and the edges between the landmarks are formed by a UVFA.

$V_\pi(s,g)$ can be approximated as:

$$V_\pi(s,g) \approx E[w_\pi(s,g)] = E[-d_\pi(s,g)] \tag{2.1}$$

This suggests us to view the MDP as a directed graph, whose nodes are the state set $\mathcal{S}$, and edges are sampled according to the transition probability in the MDP. The general value iteration for RL problems is exactly the shortest path algorithm in terms of $d_\pi(s,g)$ on this directed graph. Besides, because the nodes form a low-dimensional manifold, nodes that are far away in the state space can only be reached by a long path.

The MDP of our universal goal reaching problem is a large-scale directed graph whose nodes are in a low-dimensional manifold. This structure allows us to estimate the all-pair shortest paths accurately by a landmark based coarsening of the graph.

2.5 Approach

In this paper, we choose deep RL algorithms such as DQN and DDPG for discrete and continuous action space, respectively. UVFA [SHGS15] is a goal-conditioned extension of the original DQN, while HER (Sec 2.3.2), can produce more informative feedback for UVFA learning. Our algorithm is thus based upon HER, and the extension of this approach for other algorithms is also straightforward.

2.5.1 Basic Idea

Our approach aims at addressing the fundamental challenges in UVFA learning. As characterized in the previous section, the UVFA estimation solves a pair-wise shortest path problem, and the underlying graph has a node space of high cardinality. Note that UVFA has to memorize the distance between every state-goal pairs, through trajectory samples from the starting state to the goal, which is much larger than the original state space.

Such large set of state-goal pairs poses the challenge. First, it takes longer time to sample enough state-goal pairs. Particularly, at the early stage, only few state-goal samples have been collected, so learning from them requires heavy extrapolation by networks, which is well known to be unreliable. Second, memorizing all the experiences is too difficult even for large networks.

We propose a map to abstract the visited state space by landmarks and edges to connect them. This abstraction is reasonable due to the underlying structure of our graph — a low-dimensional manifold [GH05]. We also learn local UVFA networks that only needs to be accurate in the neighborhood of landmarks. As illustrated in Figure 2.1, an ant robot is put in an "U" Maze to reach a given position. It should learn to model the maze as a small-scale map based on its past experiences.

This solution addresses the challenges. For the UVFA network, it only needs to remember experiences in a local neighborhood. Thus, the training procedure requires much lower sample complexity. The map decomposes a long path into piece-wise short ones, and each of which is from an accurate local network.

Our framework contains three components: a value function approximator trained with hindsight experience replay, a map that is supported by sampled landmarks, and a planner that can find the optimal path with the map. We will introduce them in Sec 2.5.2, Sec 2.5.3, and Sec 2.5.4, respectively.

2.5.2 Learning a Local UVFA with HER

Specifically, we define the following reward function for goal reaching problem:

$$r_t = \mathcal{R}(s_t, a_t, g) = \begin{cases} 0 & |s'_t - g| \leq \delta \\ -1 & otherwise \end{cases}$$

Here s'_t is the next observation after taking action a_t. We first learn a UVFA based on HER, which has proven its efficiency for UVFA. In experiments (see Sec 2.6.3), we find out that the agent trained with HER does master the skill to reach goals of increasing difficulty in a curriculum way. However, the agent can seldom reach the most difficult goals constantly, while the success rate of reaching easier goals remains stable. All these observations prove that HER's value and policy is locally reliable.

One can pre-train the HER agent and then build map for planner. However, as an off-policy algorithm, HER can work with arbitrary exploration policy. Thus we use the planner based on current local HER agent as the exploration policy and train the local HER agent jointly. We sample long horizon trajectories with the planner and store them into the replay buffer. We change the replacement strategy in HER, ensuring that the replaced goals are sampled from the near future within a fixed number of steps to increase the agent's ability to reach nearby goals at the early stage.

The UVFA trained in this step will be used in the planner for two purposes: (1) to estimate the distance between two local states belonging to the same landmark, or between two nearby landmarks; and (2) to decide whether two states are close enough so that we can trust the distance estimation from the network. Although the learned UVFA is imperfect globally, it is enough for the two local usages.

2.5.3 Building a Map by Sampling Landmarks

After training the UVFA, we will obtain a distance estimation $d(s,g)$[2], a policy for any state-goal pair (s,g), and a replay buffer that contains all the past experiences. We will build a landmark-based map to abstract the state space based on the experiences.

Landmark Sampling The replay buffer stores visited states during training. Instead of localizing few important states that play a key role in connecting the environment, we seek to sample many states to cover the visited state space.

Limited by computation budget, we first uniformly sample a big set of states from the replay buffer, and then use the farthest point sampling (FPS) algorithm [AV07] to select landmarks to support the explored state space. The metric for FPS can either be the Euclidean distance between the original state representation or the pairwise value estimated by the agent.

We compare different sampling strategies in Section 2.6.3, and demonstrate the advantage of FPS in abstracting the visited state space and exploration.

Connecting Nearby Landmarks We first connect landmarks that have a reliable distance estimation from the UVFA and assign the UVFA-estimated distance between them as the weight of the connecting edge.

Since UVFA is accurate locally but unreliable for long-term future, we choose to only connect nearby landmarks. The UVFA is able to return a distance between any pair (s,g), so we connect the pairs with distance below a preset threshold τ, which should ensure that all the edges are reliable, as well as the whole graph is connected.

[2]If the algorithm returns a Q function, we will calculate the value by selecting the optimal action and calculate the Q function and convert to d by Eq. 2.1

With these two steps, we have built a directed weighted graph which can approximate the visited state space. This graph is our map to be used for high-level planning. Such map induces a new environment, where the action is to choose to move to another landmark.

2.5.4 Planning with the Map

We can now leverage the map and the local UVFA network to estimate the distance between any state-goal pairs, which induces a reliable policy for the agent to reach the goal.

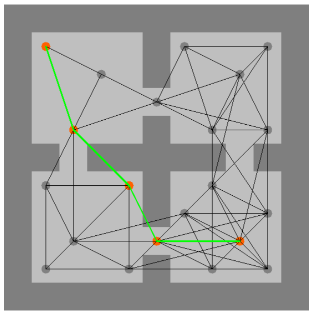
For a given pair of (s,g), we can plan the optimal path between (s,g) by selecting a serial of landmarks $l_1, \cdots, l_k$, so that the approximated distance will be $\bar{d}(s,g) = \min_{l_1,\cdots,l_k} d(s,l_1) + \sum_{i=1}^{k-1} d(l_i,l_{i+1}) + d(l_k,g)$. The policy from s to g can then be approximated as: $\bar{\pi}(s,g) = \pi(s,l_1) + \sum_{i=1}^{k-1} \pi(l_i,l_{i+1}) + \pi(l_k,g)$. Here the summation of π is the concatenation of the corresponding action sequence.

In our implementation, we run the shortest path algorithm to solve the above minimization problem. To speed up the pipeline, we first calculate the pairwise distances $d(l_i,g)$ between each landmark l_i and the goal g when episode starts. When the agent is at state s, we can choose the next subgoal by finding $g_{next} = \arg\min_{l_i} d(s,l_i) + d(l_i,g)$.

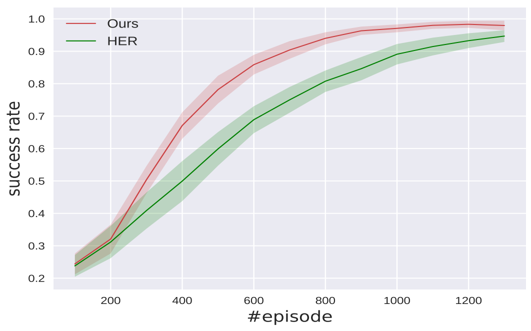
2.6 Experiments

2.6.1 FourRoom: An Illustrative Example

We first demonstrate the merits of our method in the FourRoom environment, where the action space is discrete. The environment is visualized in Figure 2.2a. There are walls separating the space into four rooms, with narrow openings to connect them. For this discrete environment, we use DQN [MKS$^+$13] with HER [AWR$^+$17] to learn the Q value. Here, we use the one-hot representation of the x-y position as the input of the network. The initial states and the goals are



(a) FourRoom



(b) Success Rate

Figure 2.2: The results on FourRoom Environment. Figure 2.2a shows the sampled landmarks and the planned path based on our algorithm. Figure 2.2b indicates the success rate to reach the goal.

randomly sampled during training.

We compare our method with DQN on success reaching rate, and their performances are shown in Figure 2.2b. Our method can achieve better accuracy at the early stage.

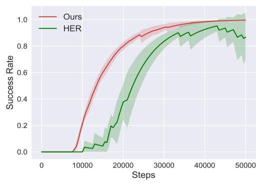
2.6.2 Continuous Control

In this section, we will compare our method with HER on challenging classic control tasks and MuJoCo [TET12a] goal-reaching environments.

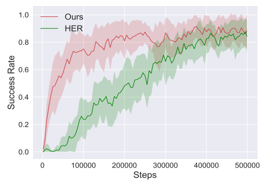
The results compared with HER are shown in Figure 2.3. Our method trains UVFA with planner and HER. It is evaluated under the test setting, using the model and replay buffer at corresponding training steps.

In the **2DReach** and **2DPush** task (shown in Figure 2.3b), we can see our method achieves better performance. When incorporating with control tasks, for **BlockedFetchReach** and **FetchPush** environments, the results still show that our performance is better than HER, but the improvement is not so remarkable. We guess this comes from the strict time limit of the two environments, which is only 50. We observe that pure HER can finally learn well, when the task horizon is not very long.

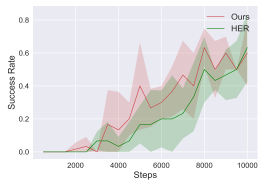
We expect that building maps would be more helpful for long-range goals, which is



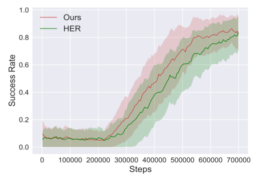
(a) 2DReach



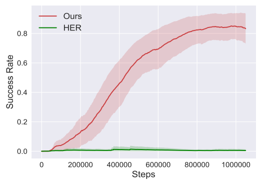
(b) 2DPush



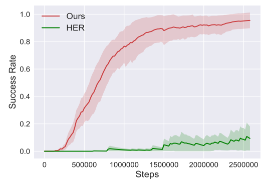
(c) BlockedFetchReach



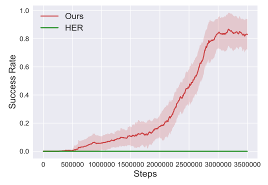
(d) FetchPush



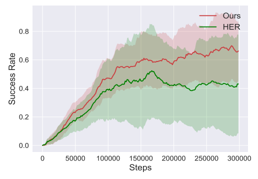
(e) PointMaze



(f) AntMaze



(g) Complex AntMaze



(h) Acrobot

Figure 2.3: Experiments on the continuous control environments. The red curve indicates the performance of our method at different training steps.

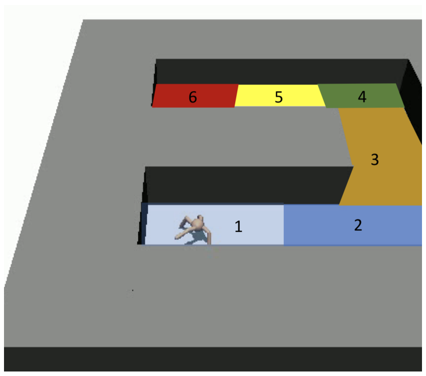
evidenced in the environments with longer episode length. Here we choose **PointMaze** and **AntMaze** with scale 12×12. For training, the agent is born at a random position to reach a random goal in the maze. For testing, the agent should reach the other side of the "U-Maze" within 500 steps. For these two environments, the performance of planning is significantly better and remains stable, while HER can hardly learn a reliable policy. Results are shown in Figure 2.3e and Figure 2.3f.

We also evaluate our method on classic control, and more complex navigation + locomotion task. Here we choose **Complex Antmaze** and **Acrobot**, and results are shown in Figure 2.3h and Figure 2.3g. The advantage over baseline demonstrates our method is applicable to complicated navigation tasks as well as general MDPs.

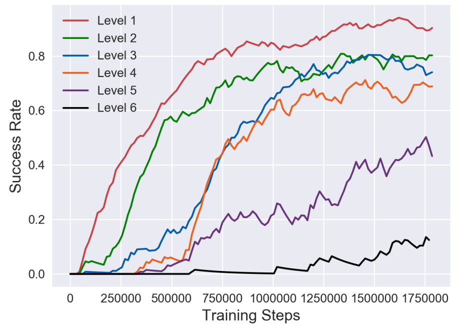
2.6.3 Ablation Study

We study some key factors that affect our algorithm on AntMaze.

Choice of clip range and landmarks There are two main hyper-parameters for the planner – the number of landmarks and the edge clipping threshold τ. Figure 2.5a shows the



(a) Multi-level AntMaze



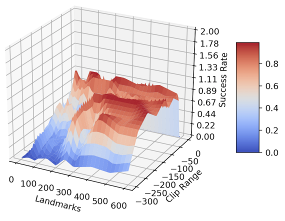
(b) Success Rate

Figure 2.4: AntMaze of multi-level difficulty. Figure 2.4b is the average steps and success rate to reach different level of goals, respectively.

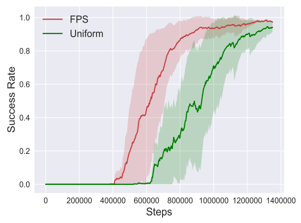
evaluation result of the model trained after 0.8M steps in AntMaze. We see that our method is generally robust under different choices of hyper-parameters. Here τ is the negative distance between landmarks. If it's too small, the landmarks will be isolated and can't form a connected graph. The same problem comes when the landmarks are not enough.

The local accuracy of HER We evaluate our model trained between $0 \sim 2.5$M steps, for goals of different difficulties. We manually define the difficulty level of goals, as shown in Figure 2.4a. Goal's difficulty increases from Level 1 to Level 6. We plot the success rate as well as the average steps to reach these goals. We find out that, for the easier goals, the agent takes less time and less steps to master the skill. The success rate and average steps also remain more stable during the training process, indicating that our base model is more reliable and stable in the local area.

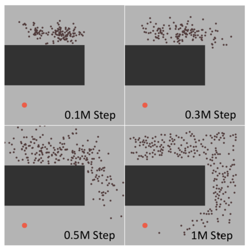
Landmark sampling strategy comparison Our landmarks are dynamically sampled from the replay buffer by iterative FPS algorithm using distances estimated by UVFA, and get updated at the beginning of every episode. The FPS sampling tends to find states at the boundary of the visited space, which implicitly helps exploration. We test FPS and uniform sampling in fix-start AntMaze (The ant is born at a fixed position to reach the other side of maze for both training and testing). Figure 2.5b shows that FPS has much higher success rate than uniform sampling.



(a) Hyperparameters of the planner



(b) FPS vs. Uniform Sampling



(c) Landmark-based Map

Figure 2.5: Figure 2.5a shows the relationship with the landmarks and clip range in the planner. Figure 2.5b shows FPS outperforms uniform sampling. And Figure 2.5c is the landmark-based map at different training steps constructed by FPS.

Figure 2.5c shows landmark-based graph at four training stages. Through FPS, landmarks expand gradually towards the goal (red dot), even if it only covers a small proportion of states at the beginning.

Chapter 2, in full, is a material found in Thirty-third Conference on Neural Information Processing Systems 2019. Huang, Zhiao; Liu, Fangchen; Su, Hao. Mapping State Space Using Landmarks for Universal Goal Reaching. The thesis author was the co-first author of this paper.

Chapter 3

Imitation Learning with State Prediction and Alignment

3.1 Introduction

Learning from demonstrations (imitation learning, abbr. as IL) is a basic strategy to train agents for solving complicated tasks. Imitation learning methods can be generally divided into two categories: behavior cloning (BC) and inverse reinforcement learning (IRL). Behavior cloning [RGB11a] formulates a supervised learning problem to learn a policy that maps states to actions using demonstration trajectories. Inverse reinforcement learning [Rus98, NR00] tries to find a proper reward function that can induce the given demonstration trajectories. GAIL [HE16] and its variants [FLL, QBY18, XHW$^+$19] are the recently proposed IRL-based methods, which uses a GAN-based reward to align the distribution of state-action pairs between the expert and the imitator.

Although state-of-the-art BC and IRL methods have demonstrated compelling performance in standard imitation learning settings, e.g. control tasks [HE16, FLL, QBY18, XHW$^+$19] and video games [APB$^+$18a], these approaches are developed based on a strong assumption:

the expert and the imitator share **the same** dynamics model; specifically, they have the same action space, and any feasible state-action pair leads to the same next state in probability for both agents. The assumption brings severe limitation in practical scenarios: Imagine that a robot with a low speed limit navigates through a maze by imitating another robot which moves fast, then, it is impossible for the slow robot to execute the exact actions as the fast robot. However, the demonstration from the fast robot should still be useful because it shows the path to go through the maze.

We are interested in the imitation learning problem under a relaxed assumption: Given an imitator that shares the same state space with the expert but their dynamics may be different, we train the imitator to follow the state sequence in expert demonstrations as much as possible. This is a more general formulation since it poses fewer requirements on the experts and makes demonstration collection easier. Due to the dynamics mismatch, the imitator becomes more likely to deviate from the demonstrations compared with the traditional imitation learning setting. Therefore, it is very important that the imitator should be able to resume to the demonstration trajectory by itself. Note that neither BC-based methods nor GAIL-based IRL methods have learned to handle dynamics misalignment and deviation correction.

To address the issues, we propose a novel approach with four main features: 1) *State-based*. Compared to the majority of literature in imitation learning, our approach is state-based rather than action-based. Not like BC and IRL that essentially match state-action pairs between the expert and the imitator, we only match states. An inverse model of the imitator dynamics is learned to recover the action; 2) *Deviation Correction*. A state-based β-VAE [HMP+17] is learned as the prior for the next state to visit. Compared with ordinary behavior cloning, this VAE-based next state predictor can advise the imitator to return to the demonstration trajectory when it deviates. The robustness benefits from VAE's latent stochastic sampling; 3) *Global State Alignment*. While the VAE can help the agent to correct its trajectory to some extent, the agent may still occasionally enter states that are far away from demonstrations, where the VAE has no clue how to correct it. So we have to add a global constraint to align the states in demonstration and imitation. Inspired by GAIL that uses reward to align the distribution of state-action pairs, we also formulate an IRL problem whose maximal cumulative reward is the Wasserstein Distance between states of demonstration and imitation. Note that we choose not to involve state-action pairs as in GAIL[HE16], or state-state pairs as in an observation-based GAIL [TWS18b], because our state-only formulation imposes weaker constraints than the two above options, thus providing more flexibility to handle different agent dynamics; 4) *Regularized Policy Update*. We combine the prior for next state learned from VAE and the Wasserstein distance-based global constraint from IRL in a unified framework, by imposing a Kullback-Leibler divergence based regularizer to the policy update in the Proximal Policy Optimization algorithm.

To empirically justify our ideas, we conduct experiments in two different settings. We first show that our approach can achieve similar or better results on the standard imitation learning setting, which assumes the same dynamics between the expert and the imitator. We then evaluate our approach in the more challenging setting that the dynamics of the expert and the imitator are different. In a number of control tasks, we either change the physics properties of the imitators or cripple them by changing their geometries. Existing approaches either fail or can only achieve very low rewards, but our approach can still exhibit decent performance. Finally, we show that even for imitation across agents of completely different actuators, it is still possible for the state-alignment based method to work. Surprisingly, a point mass and an ant in MuJoCo [TET12b] can imitate each other to navigate in a maze environment.

3.2 Related work

Imitation learning is widely used in solving complicated tasks where pure reinforcement learning might suffer from high sample complexity, like robotics control [LYCL17, YA17, PML$^+$18], autonomous vehicle [FLL, Pom89], and playing video game [APB$^+$18b]. Behavioral cloning [BS99] is a straight-forward method to learn a policy in a supervised way. However, behavioral cloning suffers from the problem of compounding errors as shown by [RB10], and this can be somewhat alleviated by interactive learning, such as DAGGER [RGB11a]. Another important line in imitation learning is inverse reinforcement learning [Rus98, NR00, AN04, ZMBD08, FLL], which finds a cost function under which the expert is uniquely optimal.

Since IRL can be connected to min-max formulations, works like GAIL, SAM [HE16, BK18] utilize this to directly recover policies. Its connections with GANs [GPAM$^+$14] also lead to f-divergence minimization [KBS$^+$19, NCT16] and Wasserstein distance minimization [XHW$^+$19]. One can also extend the framework from matching state-action pairs to state distribution matching, such as [TWS18b, SVBB19, SI17]. Other works [APB$^+$18a, LGAL18, PKM$^+$18] also learn from observation alone, by defining reward on state and using IRL to solve the tasks. Works like [LEP$^+$19, LEP$^+$] also use state-based reward for exploration. [TWS18a, ESSI18] will recover actions from observations by learning an inverse model or latent actions. However, our work aims to combine the advantage of global state distribution matching and local state transition alignment, which combines the advantage of BC and IRL through a novel framework.

3.3 Backgrounds

3.3.1 Variational Autoencoders

[KW13, RMW14] provides a framework to learn both a probabilistic generative model $p_\theta(\mathbf{x}|\mathbf{z})$ as well as an approximated posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$. β-VAE is a variant VAE that introduces an adjustable hyperparameter β to the original objective:

$$\mathcal{L}(\theta,\phi;\mathbf{x},\mathbf{z},\beta) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \beta D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})\right) \tag{3.1}$$

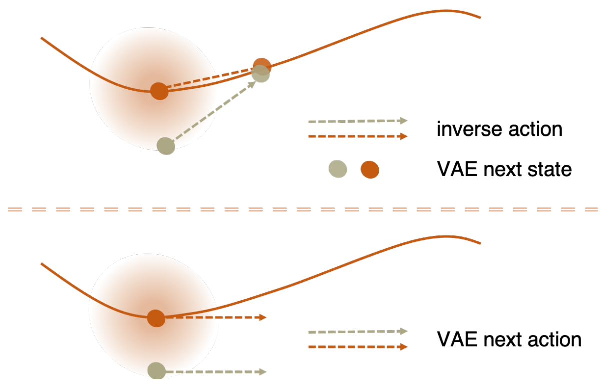


Figure 3.1: Using VAE as a state predictive model will be more self-correctable because of the stochastic sampling mechanism. But this won't happen when we use VAE to predict actions.

Larger β will penalize the total correlation [CLGD18] to encourage more disentangled latent representations, while smaller β often results in sharper and more precise reconstructions.

3.3.2 Wasserstein distance

The Wasserstein distance between two density functions $p(x)$ and $q(x)$ with support on a compact metric space (M,d) has an alternative form due to Kantorovich-Rubenstein duality [Vil08]:

$$\mathcal{W}(p,q) = \sup_{\phi \in \mathcal{L}_1} \mathbb{E}_{p(x)}[\phi(x)] - \mathbb{E}_{q(x)}[\phi(x)] \tag{3.2}$$

Here, $\mathcal{L}_1$ is the set of all 1-Lipschitz functions from $\mathcal{M}$ to $\mathbb{R}$. Compared with the prevalent KL-divergence and its extension, the f-divergence family, Wasserstein distance has a number of advantages theoretically and numerically. Please refer to [ACB17] and [Sol18] for a detailed discussion.

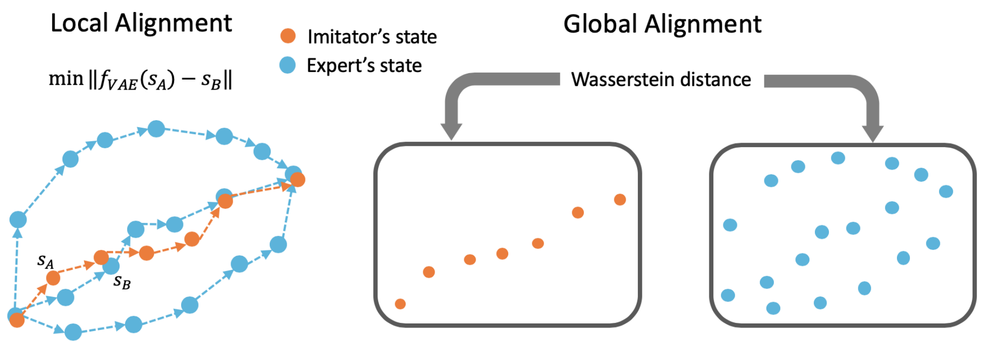


Figure 3.2: Visualization of state alignment

3.4 Approach

3.4.1 Overview

Our imitation learning method is based on state alignment from both local and global perspectives. For local alignment, the goal is to follow the transition of the demonstration as much as possible, and allow the return to the demonstration trajectory whenever the imitation deviates. To achieve both goals, we use a β-VAE [HMP+17] to generate the next state (Figure 3.2 Left). For global alignment, we set up an objective to minimize the Wasserstein distance between the states in the current trajectory and the demonstrations (Figure 3.2 Right). There has to be a framework to naturally combine the local alignment and global alignment components. We resort to the reinforcement learning framework by encoding the local alignment as policy prior and encoding the global alignment as reward over states. Using Proximal Policy Optimization (PPO) by [SWD+17] as the backbone RL solver, we derive a regularized policy update. To maximally exploit the knowledge from demonstrations and reduce interactions with the environment, we adopt a pre-training stage to produce a good initialization based on the same policy prior induced by the local alignment. Our method is summarized in Algorithm **??**. In the rest parts of this section, we will introduce all the components of our method in details.

3.4.2 Local Alignment by State Predictive VAE

To align the transition of states locally, we need a predictive model to generate the next state which the agent should target at. And then we can train an inverse dynamics model to recover the corresponding action, so as to provide a direct supervision for policy.

Instead of using an ordinary network to memorize the subsequent states, which will suffer from the same issue of compounding errors as behavioral cloning [RB10, RGB11b], we propose to use VAE to generate the next state based on the following two reasons. First, as shown in [DWA$^+$18], VAE is more robust to outliers and regularize itself to find the support set of a data manifold, so it will generalize better for unseen data. Second, because of the latent stochastic sampling, the local neighborhood of a data point will have almost the same prediction, which is self-correctable when combined with a precise inverse dynamics model as illustrated in Figure 3.1.

We can also use a VAE to generate action based on the current state. But if the agent deviated from the demonstration trajectory a little bit, this predicted action is not necessarily guide the agent back to the trajectory, as shown in Figure 3.1. And in Sec **??**, we conduct experiments to compare the state predictive VAE and the action predictive VAE.

Instead of the vanilla VAE, we use β-VAE to balance the KL penalty and prediction error, with formulation shown in (3.1). In Sec 3.5, we discuss the effects of the hyper-parameter β in different experiment settings as one of the ablation studies.

3.4.3 Global Alignment by Wasserstein Distance

Due to the difference of dynamics between the expert and the imitator, the VAE-based local alignment cannot fully prevent the imitator from deviating from demonstrations. In such circumstances, we still need to assess whether the imitator is making progress in learning from the demonstrations. We, therefore, seek to control the difference between the state visitation distribution of the demonstration and imitator trajectories, which is a global constraint.

Note that using this global constraint alone will not induce policies that follow from the demonstration. Consider the simple case of learning an imitator from experts of the same dynamics. The expert takes cyclic actions. If the expert runs for 100 cycles with a high velocity and the imitator runs for only 10 cycles with a low velocity within the same time span, their state distribution would still roughly align. That is why existing work such as GAIL aligns state-action occupancy measure. However, as shown later, our state-based distribution matching will be combined with the local alignment component, which will naturally resolve this issue. The advantage of this state-based distribution matching over state-action pair matching as in GAIL or state-next-state pair matching in [TWS18b] is that the constraint becomes loosened.

We use IRL approach to achieve the state distribution matching by introducing a reinforcement learning problem. Our task is to design the reward to train an imitator that matches the state distribution of the expert.

Before introducing the reward design, we first explain the computation of the Wasserstein distance between the expert trajectories $\{\tau_e\}$ and imitator trajectory $\{\tau\}$ using the Kantorovich duality:

$$\mathcal{W}(\tau_e, \tau) = \sup_{\phi \in \mathcal{L}_1} \mathbb{E}_{s\sim\tau_e}[\phi(s)] - \mathbb{E}_{s\sim\tau}[\phi(s)] \tag{3.3}$$

where ϕ is the Kantorovich's potential, and serves as the discriminator in WGAN [ACB17]. ϕ is trained with a gradient penalty term as WGAN-GP introduced in [GAA+17]

After the rollout of imitator policy is obtained, the potential ϕ will be updated by (3.3). Assume a transition among an imitation policy rollout of length T is (s_i, s_{i+1}). To provide a dense signal every timestep, we assign the reward as:

$$r(s_i, s_{i+1}) = \frac{1}{T}[\phi(s_{i+1}) - \mathbb{E}_{s\sim\tau_e}\phi(s)] \tag{3.4}$$

We now explain the intuition of the above reward. By solving (3.3), those states of higher probability in demonstration will have a larger ϕ value. The reward in (3.4) will thus encourage the imitator to visit such states.

Maximizing the curriculum reward will be equivalent to

$$J(\pi) = \sum_{t=1}^{T} \mathbb{E}_{s_t, s_{t+1} \sim \pi}[r(s_t, s_{t+1})] = \sum_{t=1}^{T} \frac{\mathbb{E}_{s_{t+1}}[\phi(s_{t+1}) - \mathbb{E}_{s \sim \tau_e}[\phi(s)]]}{T} = -\mathcal{W}(\tau_e, \tau)$$

In other words, the optimal policy of this MDP best matches the state visitation distributions w.r.t Wasserstein distance.

3.4.4 Regularized PPO Policy Update Objective

As mentioned in the second paragraph of Sec 3.4.3, the global alignment has to be combined with local alignment. This is achieved by adding a prior to the original clipped PPO objective.

We maximize the following unified objective function:

$$J(\pi_\theta) = L^{CLIP}(\theta) - \lambda D_{KL}\left(\pi_\theta(\cdot|s_t) \middle\| \, p_a\right) \tag{3.5}$$

We will explain the two terms in detail. $L^{CLIP}(\theta)$ denotes the clipped surrogate objective used in the original PPO algorithm:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \hat{A}_t, \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}, 1-\varepsilon, 1+\varepsilon \right) \hat{A}_t \right) \right], \tag{3.6}$$

where $\hat{A}_t$ is an estimator of the advantage function at timestep t. The advantage function is calculated based on a reward function described in Sec 3.4.3.

The D_{KL} term in (3.5) serves as a regularizer to keep the policy close to a learned policy prior p_a. This policy prior p_a is derived from the state predictive VAE and an inverse dynamics model. Assume the β-VAE is $f(s_t) = s_{t+1}$ and the inverse dynamics model is $g_{inv}(s_t, s_{t+1}) = a$. To solve the case when the agents have different dynamics, we learn a state prediction network and use a learned inverse dynamics to decode the action. We define the action prior as

$$p_a(a_t|s_t) \propto \exp\left(-\left\|\frac{g_{inv}(s_t, f(s_t)) - a_t}{\sigma}\right\|^2\right) \tag{3.7}$$

where the RHS is a pre-defined policy prior, a Gaussian distribution centered at $g_{inv}(s_t, f(s_t))$. σ controls how strong the action prior is when regularizing the policy update, which is a hyper-parameter. Note that the inverse model can be further adjusted during interactions.

L^{CLIP} is computed through the advantage $\hat{A}_t$ and reflects the global alignment. The policy prior is obtained from the inverse model and local β-VAE, which makes the D_{KL} serve as a local alignment constraint. Furthermore, our method can be regard as a combination of BC and IRL because our KL-divergence based action prior encodes the BC policy and we update the policy leveraging reward.

We would note that our state-alignment method augments state distribution matching by taking relationships of two consecutive states into account with robustness concern.

3.4.5 Pre-training

We pretrain the state predictive VAE and the inverse dynamics model, and then obtain the policy prior in (3.7), which is a Gaussian distribution. For pre-training, we want to initialize PPO's Gaussian policy π by this prior p_a, by minimizing the KL-divergence between them. Practically, we use direct supervision from $g_{inv}(s_t, f(s_t))$ and σ in (3.7) to directly train both the mean and variance of the policy network, which is more efficient during the pre-training stage. During the online interaction, the update rule of PPO's policy is by optimizing (3.5), and the variance will be further adjusted for all the dimensions of the action space.

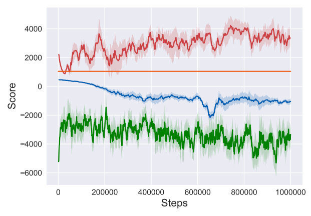
3.5 Experiments

We conduct two different kinds of experiments to show the superiority of our method. In Sec 3.5.1, we compare our method with behavior cloning [BS99], GAIL [HE16], and AIRL [FLL] in control setting where the expert and the imitator have different dynamics model, e.g., both of them are ant robots but the imitator has shorter legs. In Sec 3.5.1, we further evaluate in the traditional imitation learning setting. Finally, in Sec **??**, we conduct ablation study to show the contribution of the components.

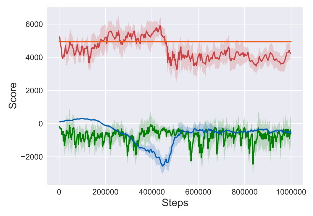
3.5.1 Imitation Learning across Agents of Different Action Dynamics

Actors of Modified Physics and Geometry Properties

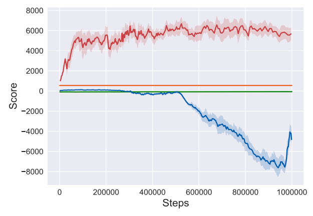
We create environments using MuJoCo [TET12b] by changing some properties of experts, such as density and geometry of the body. We create 6 different environments: Heavy/Light/Disabled Ant/Swimmer. The Heavy/Light agents have modified density, and the disabled agents have modified head/tail/leg lengths. The demonstrations are collected from the standard Ant-v2 and Swimmer-v2. More descriptions of the environments can be founded in the Appendix.



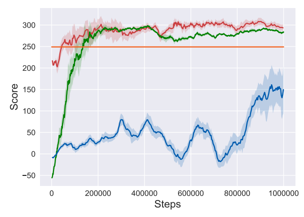
(a) DisabledAnt



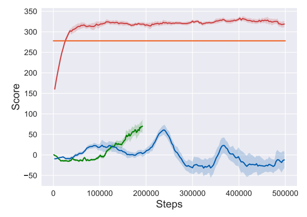
(b) LightAnt



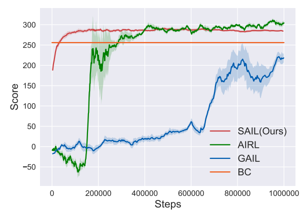
(c) HeavyAnt



(d) DisableSwimmer



(e) LightSwimmer



(f) HeavySwimmer

Figure 3.3: Comparison with BC, GAIL and AIRL when dynamics are different from experts.

Figure 3.3 demonstrates the superiority of our methods over all the baselines. Our approach is the most stable in all the 6 environments and shows the leading performance in each of them. GAIL seems to be the most sensitive to dynamics difference. AIRL, which is designed to solve imitation learning for actors of different dynamics, can perform on par with our method in two swimmer-based environments (DisabledSwimmer and HeavySwimmer) that have relatively lower dimensional action space (2D for swimmer versus 8D for ants).

Interestingly, the stability and performance of vanilla behavior cloning are quite reasonable, although it failed to move about in the DisabledAnt and HeavyAnt environments. In the other four games, BC agents do not die but just move less efficiently, so they have a sub-optimal yet still reasonable score.

Actors of Heterogeneous Action Dynamics

We consider a challenging setting that the imitator and demonstrator are functionally different. One typical example of expert/imitator pair in practice would be a human and a humanoid robot. We consider a much simplified version but with similar nature – a Point and an Ant in MuJoCo. In this task, even if the state space cannot be exactly matched, there are still some shared dimensions across the state space of the imitator and the actor, e.g., the location of the center of mass, and the demonstration should still teach the imitator in these dimensions.

We use the same setting as many hierarchical RL papers, such as HIRO and Near-Optimal RL [NGLL18a, NGLL18b]. The agent need to reach a goal position in a maze, which is represented by (x,y) coordinates. We also know that the first two dimensions of states are the position of the agent. The prior knowledge includes: (1) the goal space (or the common space that need to be matched) (2) the projection from the state space to the goal space (select the first two dimensions of the states).

The first task is that the Ant should reach the other side of the maze from several successful demonstrations of a Point robot. As shown in Figure 3.4, the maze structure for the ant and point

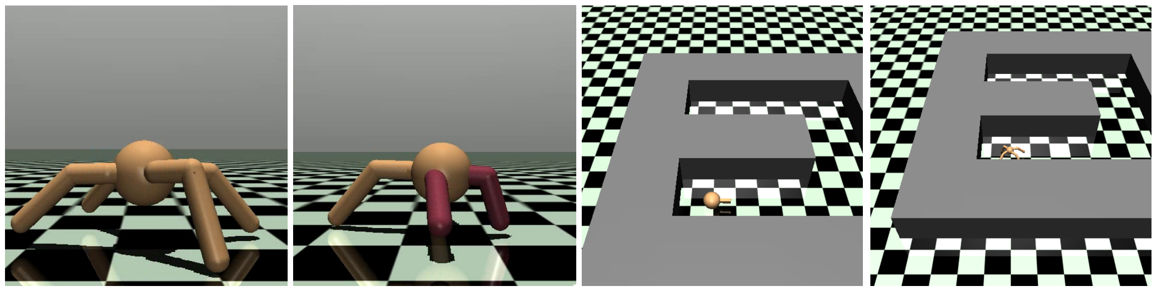


Figure 3.4: Imitation Learning of Actors with Heterogeneous Action Dynamics.

mass is exactly the same.

To solve this problem, we first pre-train an VAE on the demonstrations, and use this VAE to propose the next "subgoal" for the Ant. This VAE is trained on the goal space (i.e. the first two dimensions) of the Point robot's trajectory. Then we train an inverse model for Ant, which will generate an action based on the Ant's current state (high dimensional) and goal predicted by VAE (2 dimensional. After 1M training steps, the agent has success rate of 0.8 to reach the other side of the maze.

3.5.2 Actors of the Same Dynamics (Standard Imitation Learning)

We also evaluate our algorithm on 6 non-trivial control tasks in MuJoCo: Swimmer, Hopper, Walker, Ant, HalfCheetach, and Humanoid. We first collect demonstration trajectories with Soft Actor-Critic, which can learn policies that achieve high scores in most of these environments[1]. For comparison, we evaluate our method against 3 baselines: behavior cloning, GAIL, and AIRL[2]. Also, to create even stronger baselines for the cumulative reward and imitator run-time sample complexity, we initialize GAIL with behavior cloning, which would obtain higher scores in Swimmer and Walker. Lastly, to evaluate how much each algorithm depends on the amount of demonstrations, we sampled demonstration trajectories of ten and fifty episodes.

[1]We collect near-optimal demonstration on Swimmer using TRPO due to the limited performance of SAC.
[2]AIRL and EAIRL[QBY18] have similar performance, and we only compare to AIRL.

Table 3.1 depicts representative results in Hopper and HalfCheetah[3]. The advantage of our methods over BC should be attributed to the inherent data augmentation by VAE. On Hopper-v2, we are significantly better with 10 demos but are just on par if the demos are increased to 50. On HalfCheetah-v2, the demo cheetah runs almost perfectly (12294 scores); in other words, the demo provides limited instruction when the imitator is even slightly off the demo states, thus the robustness from VAE becomes critical.

Chapter 3, in full, is a material found in Eighth International Conference on Learning Representations 2020. Liu, Fangchen; Ling, Zhan; Mu, Tongzhou; Su, Hao. State Alignment-based Imitation Learning. The thesis author was the first author of this paper.

Table 3.1: Performance on Hopper-v2 and HalfCheetah-v2

	Hopper-v2		HalfCheetah-v2	
# Demo	10	50	10	50
Expert	3566 ± 1.24		12294.22 ± 273.59	
BC	1318.76 ± 804.36	3525.87 ± 160.74	971.42 ± 249.62	4813.20 ± 1949.26
GAIL	3372.66 ± 130.75	3363.97 ± 262.77	474.42 ± 389.30	-175.83 ± 26.76
BC-GAIL	3132.11 ± 520.65	3130.82 ± 554.54	578.85 ± 934.34	1597.51 ± 1173.93
AIRL	3.07 ± 0.02	3.31 ± 0.02	-146.46 ± 23.57	755.46 ± 10.92
Our init	3412.58 ± 450.97	3601.16 ± 300.14	1064.44 ± 227.32	7102.29 ± 910.54
Our final	$\mathbf{3539.56 \pm 130.36}$	$\mathbf{3614.19 \pm 150.74}$	$\mathbf{1616.34 \pm 180.76}$	$\mathbf{8817.32 \pm 860.55}$

[3]Results for other environments can be founded in the Appendix.

Appendix A

Environment Settings

A.1 Goal-reaching Environments

We test our algorithms on the following environments: **2DReach** A green point in a 2D U-maze aims to reach the goal represented by a red point, as shown in Figure A.1a. The size of the maze is 15×15. The state space and the goal space are both in this 2D maze. At each step, the agent can move within $[-1,1] \times [-1,1]$ as δ_x, δ_y in x and y directions.

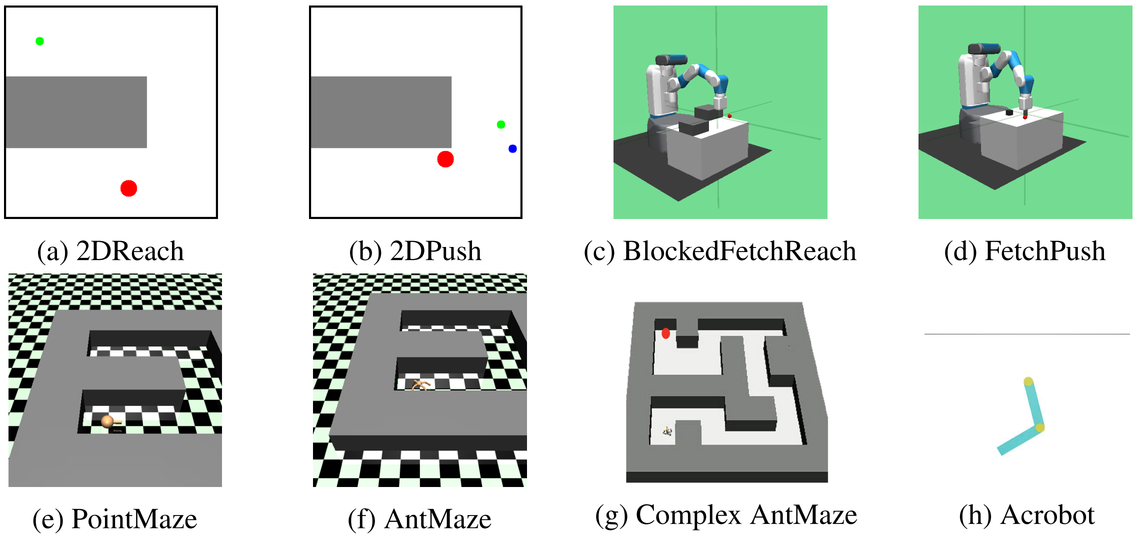


Figure A.1: The environments we use for continuous control experiments.

2DPush The green point A now need to push a blue point B to a given goal (red point) lying in the same U-maze as 2DReach, as shown in Figure A.1b. Once A has reached B, B will follow the movement of A. In this environment, the state is a 4-dim vector that contains the location of both A and B.

BlockedFetchReach & FetchPush We need to control a gripper to either reach a location in 3d space or push an object in the table to a specific location, as shown in Figure A.1c and Figure A.1d. Since the original FetchReach implemented in OpenAI gym [BCP$^+$16] is very easy to solve, we further add some blocks to increase the difficulty. We call this new environment BlockedFetchReach.

PointMaze & AntMaze As shown in Figure A.1e and Figure A.1f, a point mass or an ant is put in a 12×12 U-maze. Both agents are trained to reach a random goal from a random location and tested under the most difficult setting to reach the other side of maze within 500 steps. The states of point and ant are 7-dim and 30-dim, including positions and velocities.

Complex AntMaze As shown in Figure A.1g, an ant is put in a 56×56 complex maze. It is trained to reach a random goal from a random location and tested under the most difficult setting to reach the farthest goal (indicated as the red point) within 1500 steps.

Acrobot As shown in Figure A.1h, an acrobot includes two joints and two links. Goals are states that the end-effector is above the black line at specific joint angles and velocities. The states and goals are both 6-dim vectors including joint angles and velocities.

A.2 Cross-Morphology Imitation Learning Environments

PointMaze & AntMaze As shown in Figure 3.4, a point mass or an ant is put in a 24×24 U-maze. The task is to make the agent reach the other side of U-maze with the demonstration from the point mass. The ant is trained to reach a random goal in the maze from a random location, and should reach the other side of the maze. The state space of ant is 30-dim, which contains the positions and velocities.

HeavyAnt Two times of original Ant's density. Two times of original gear of the armature.

LightAnt One tenth of original Ant's density.

DisabledAnt Two front legs are 3 quarters of original Ant's legs.

HeavySwimmer 2.5 times of original Swimmer's density.

LightSwimmer One twentieth of original Swimmer's density.

DisabledSwimmer Make the last joint 1.2 times longer and the first joint 0.7 times of the original length

The exact results of these environments are listed in Table A.1, A.2. All the statistics are calculated from 20 trails.

Table A.1: Performance on modifeid Swimmer

	DisabledSwimmer	LightSwimmer	HeavySwimmer
BC	249.09 ± 1.53	277.99 ± 3.41	255.95 ± 2.5
GAIL	228.46 ± 2.02	-4.11 ± 0.51	254.91 ± 1.35
AIRL	283.42 ± 3.69	67.58 ± 25.09	**301.27 ± 5.21**
SAIL(Ours)	**287.71 ± 2.31**	**342.61 ± 6.14**	286.4 ± 3.2

Table A.2: Performance on modified Ant

	DisabledAnt	HeavyAnt	LightAnt
BC	1042.45 ± 75.13	550.6 ± 77.62	**4936.59 ± 53.42**
GAIL	-1033.54 ± 254.36	-1089.34 ± 174.13	-971.74 ± 123.14
AIRL	-3252.69 ± 153.47	-62.02 ± 5.33	-626.44 ± 104.31
SAIL(Ours)	**3305.71 ± 67.21**	**5608.47 ± 57.67**	4335.46 ± 82.34

Appendix B

Imitation Benchmark Experiments

We use six MuJoCo [TET12b] control tasks. The name and version of the environments are listed in Table B.1, which also list the state and action dimension of the tasks with expert performance and reward threshold to indicate the minimum score to solve the task. All the experts are trained by using SAC [HZAL18] except Swimmer-v2 where TRPO [SLA$^+$15] get higher performance.

Table B.1: Performance on benchmark control tasks

Environment	State Dim	Action Dim	Reward threshold	Expert Performance
Swimmer-v2	8	2	360	332
Hopper-v2	11	3	3800	3566
Walker2d-v2	17	6	-	4924
Ant-v2	111	8	6000	6157
HalfCheetah-v2	17	6	4800	12294
Humanoid-v2	376	17	1000	5187

The exact performance of all methods are list in Table B.2, B.3, B.4, B.5, B.6, B.7. We compare GAIL[HE16], behavior cloning, GAIL with behavior cloning initilization and AIRL to our method containing. Means and standard deviations are calculated from 20 trajectories after the agents converge and the number total interactions with environments is less than one million environment steps.

Table B.2: Performance on Swimmer-v2 with different trajectories

Swimmer-v2				
#Demo	5	10	20	50
Expert	332.88 ± 1.24			
BC	328.85 ± 2.26	331.17 ± 2.4	332.17 ± 2.4	330.65 ± 2.42
GAIL	304.64 ± 3.16	271.59 ± 11.77	56.16 ± 5.99	246.73 ± 5.76
BC-GAIL	313.80 ± 3.42	326.58 ± 7.87	294.93 ± 12.21	315.68 ± 9.99
AIRL	332.11 ± 2.57	338.43 ± 3.65	335.67 ± 2.72	**340.08** ± **2.70**
Our init	**332.36** ± **3.62**	335.78 ± 0.34	**336.23** ± **2.53**	334.03 ± 2.11
Our final	332.22 ± 3.23	**339.67** ± **3.21**	336.18 ± 1.87	336.31 ± 3.20

Table B.3: Performance on Hopper-v2 with different trajectories

Hopper-v2				
#Demo	5	10	20	50
Expert	3566 ± 1.24			
BC	1471.40 ± 637.25	1318.76 ± 804.36	1282.46 ± 772.24	3525.87 ± 160.74
GAIL	3300.32 ± 331.61	3372.66 ± 130.75	3201.97 ± 295.27	3363.97 ± 262.77
BC-GAIL	**3122.23** ± **358.65**	3132.11 ± 520.65	3111.42 ± 414.28	3130.82 ± 554.54
AIRL	4.12 ± 0.01	3.07 ± 0.02	4.11 ± 0.01	3.31 ± 0.02
Our init	2322.49 ± 300.93	3412.58 ± 450.97	3314.03 ± 310.32	3601.16 ± 300.14
Our final	3092.26 ± 670.72	**3539.56** ± **130.36**	**3516.81** ± **280.98**	**3610.19** ± **150.74**

Table B.4: Performance on Walker2d-v2 with different trajectories

Walker2d-v2				
#Demo	5	10	20	50
Expert	5070.97 ± 209.19			
BC	1617.34 ± 693.63	**4425.50** ± **930.62**	4689.30 ± 372.33	**4796.24** ± **490.05**
GAIL	1307.21 ± 388.55	692.16 ± 145.34	1991.58 ± 446.66	751.21 ± 150.18
BC-GAIL	**3454.91** ± **792.40**	2094.68 ± 1425.05	3482.31 ± 828.21	2896.50 ± 828.18
AIRL	-7.13 ± 0.11	-7.39 ± 0.09	-3.74 ± 0.13	-4.64 ± 0.09
Our init	1859.10 ± 720.44	2038.90 ± 260.78	4509.82 ± 1470.65	4757.58 ± 880.45
Our final	2681.20 ± 530.67	3764.14 ± 470.01	**4778.82** ± **760.34**	4780.73 ± 360.66

Table B.5: Performance on Ant-v2 with different trajectories

Ant-v2				
#Demo	5	10	20	50
Expert	6190.90 ± 254.18			
BC	**3958.20 ± 661.28**	3948.88 ± 753.41	5424.01 ± 473.05	5852.79 ± 572.97
GAIL	340.02 ± 59.02	335.25 ± 89.19	314.35 ± 52.13	284.18 ± 32.40
BC-GAIL	-1081.30 ± 673.65	-1177.27 ± 618.67	-13618.45 ± 4237.79	-1166.16 ± 1246.79
AIRL	-839.32 ± -301.54	-386.43 ± 156.98	-586.07 ± 145.43	-393.90 ± 145.13
Our init	1150.82 ± 200.87	3015.43 ± 300.70	5200.58 ± 870.74	5849.88 ± 890.56
Our final	1693.59 ± 350.74	**3983.34 ± 250.99**	**5980.37 ± 420.16**	**5988.65 ± 470.03**

Table B.6: Performance on HalfCheetah-v2 with different trajectories

HalfCheetah-v2				
#Demo	5	10	20	50
Expert	12294.22 ± 208.41			
BC	225.42 ± 147.16	971.42 ± 249.62	2782.76 ± 959.67	4813.20 ± 1949.26
GAIL	-84.92 ± 43.29	474.42 ± 389.30	-116.70 ± 34.14	-175.83 ± 26.76
BC-GAIL	1362.59 ± 1255.57	578.85 ± 934.34	3744.32 ± 1471.90	1597.51 ± 1173.93
AIRL	**782.36 ± 48.98**	-146.46 ± 23.57	1437.25 ± 25.45	755.46 ± 10.92
Our init	267.71 ± 90.38	1064.44 ± 227.32	3200.80 ± 520.04	7102.74 ± 910.54
Our final	513.66 ± 15.31	**1616.34 ± 180.76**	**6059.27 ± 344.41**	**8817.32 ± 860.55**

Table B.7: Performance on Humanoid-v2 with different trajectories

Humanoid-v2				
#Demo	5	10	20	50
Expert	5286.21 ± 145.98			
BC	**1521.55± 272.14**	**3491.07± 518.64**	4686.05 ±355.74	4746.88 ±605.61
GAIL	485.92± 27.59	486.44 ±27.18	477.15± 22.07	481.14± 24.37
BC-GAIL	363.68 ±44.44	410.03 ±33.07	487.99± 30.77	464.91 ±33.21
AIRL	79.72 ± 4.27	87.15 ± 5.01	-1293.86 ± 10.70	84.84 ± 6.46
Our init	452.31 ± 190.12	1517.63 ± 110.45	4610.25 ± 2750.86	4776.83 ± 1320.46
Our final	1225.58 ± 210.88	2190.43 ± 280.18	**4716.91 ±680.29**	**4780.07 ± 700.01**

Bibliography

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[AN04] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[APB$^+$18a] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. In *Advances in Neural Information Processing Systems*, pages 2930–2941, 2018.

[APB$^+$18b] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. In *Advances in Neural Information Processing Systems*, pages 2930–2941, 2018.

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[AWR$^+$17] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[BCP$^+$16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[BK18] Lionel Blondé and Alexandros Kalousis. Sample-efficient imitation learning via generative adversarial nets. *arXiv preprint arXiv:1809.02064*, 2018.

[BL04] Francesco Bullo and Andrew D. Lewis. *Geometric Control of Mechanical Systems*, volume 49 of *Texts in Applied Mathematics*. Springer Verlag, New York-Heidelberg-Berlin, 2004.

[BS99] Michael Bain and Claude Sommut. A framework for behavioural cloning. *Machine intelligence*, 15(15):103, 1999.

[CLGD18] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 2610–2620, 2018.

[DCH$^+$16] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. *CoRR*, abs/1604.06778, 2016.

[DST04] Vin De Silva and Joshua B Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, 2004.

[DWA$^+$18] Bin Dai, Yu Wang, John Aston, Gang Hua, and David Wipf. Connections with robust pca and the role of emergent sparsity in variational autoencoder models. *The Journal of Machine Learning Research*, 19(1):1573–1614, 2018.

[ESSI18] Ashley D Edwards, Himanshu Sahni, Yannick Schroecker, and Charles L Isbell. Imitating latent policies from observation. *arXiv preprint arXiv:1805.07914*, 2018.

[FLL] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *ICLR 2018*.

[FOR$^+$18] Aleksandra Faust, Kenneth Oslund, Oscar Ramirez, Anthony Francis, Lydia Tapia, Marek Fiser, and James Davidson. Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5113–5120. IEEE, 2018.

[GAA$^+$17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[GH05] Andrew V Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 156–165. Society for Industrial and Applied Mathematics, 2005.

[GPAM$^+$14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[HE16] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[HLF$^+$18] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

[HMP$^+$17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

[HNR68] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[HWL17] Mikael Henaff, William F Whitney, and Yann LeCun. Model-based planning with discrete and continuous actions. *arXiv preprint arXiv:1705.07177*, 2017.

[HZAL18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[IP18] Brian Ichter and Marco Pavone. Robot motion planning in learned latent spaces. *CoRR*, abs/1807.10366, 2018.

[KB19] Tobias Klamt and Sven Behnke. Towards learning abstract representations for locomotion planning in high-dimensional state spaces. *arXiv preprint arXiv:1903.02308*, 2019.

[KBS$^+$19] Liyiming Ke, Matt Barnes, Wen Sun, Gilwoo Lee, Sanjiban Choudhury, and Siddhartha Srinivasa. Imitation learning as f-divergence minimization. *arXiv preprint arXiv:1905.12888*, 2019.

[KNST16] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[KSO94] Lydia Kavraki, Petr Svestka, and Mark H Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*, volume 1994. Unknown Publisher, 1994.

[KST$^+$18] Nan Rosemary Ke, Amanpreet Singh, Ahmed Touati, Anirudh Goyal, Yoshua Bengio, Devi Parikh, and Dhruv Batra. Modeling the long term future in model-based reinforcement learning. 2018.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[LaV98] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LEP$^+$] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Ruslan Salakhutdinov, and Sergey Levine. State marginal matching with mixtures of policies.

[LEP$^+$19] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[LGAL18] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125. IEEE, 2018.

[LHP$^+$15] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[LPS18] Andrew Levy, Robert Platt, and Kate Saenko. Hierarchical reinforcement learning with hindsight. *arXiv preprint arXiv:1805.08180*, 2018.

[LYCL17] Hoang M Le, Yisong Yue, Peter Carr, and Patrick Lucey. Coordinated multi-agent imitation learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1995–2003. JMLR. org, 2017.

[MDL18] Jiayuan Mao, Honghua Dong, and Joseph J Lim. Universal agent for disentangling environments and tasks. 2018.

[MKS$^+$13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[Moo90] Andrew William Moore. Efficient memory-based learning for robot control. Technical report, 1990.

[NCT16] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279, 2016.

[NGLL18a] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018.

[NGLL18b] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018.

[NR00] Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[OSL17] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In *NIPS*, 2017.

[PAR$^+$18] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *CoRR*, abs/1802.09464, 2018.

[PGDL18] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep RL for model-based control. *CoRR*, abs/1802.09081, 2018.

[PKM$^+$18] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Trans. Graph.*, 37(6), November 2018.

[PML$^+$18] Deepak Pathak, Parsa Mahmoudieh, Michael Luo, Pulkit Agrawal, Dian Chen, Fred Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. *international conference on learning representations*, 2018.

[Pom89] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[QBY18] Ahmed H Qureshi, Byron Boots, and Michael C Yip. Adversarial imitation via variational inverse reinforcement learning. *arXiv preprint arXiv:1809.06404*, 2018.

[QY18] Ahmed H. Qureshi and Michael C. Yip. Deeply informed neural sampling for robot motion planning. *CoRR*, abs/1809.10252, 2018.

[RB10] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010.

[RGB11a] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[RGB11b] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[Rus98] Stuart J Russell. Learning agents for uncertain environments. In *COLT*, volume 98, pages 101–103, 1998.

[SDK18] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018.

[SHGS15] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320, 2015.

[SHM$^+$16] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016.

[SI17] Yannick Schroecker and Charles L Isbell. State aware imitation learning. In *Advances in Neural Information Processing Systems*, pages 2911–2920, 2017.

[SJA$^+$18] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks. *CoRR*, abs/1804.00645, 2018.

[SLA$^+$15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[SMD$^+$11] Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[Sol18] Justin Solomon. Optimal transport on discrete domains. *AMS Short Course on Discrete Differential Geometry*, 2018.

[SVBB19] Wen Sun, Anirudh Vemula, Byron Boots, and J Andrew Bagnell. Provably efficient imitation learning from observation alone. *arXiv preprint arXiv:1905.10948*, 2019.

[SWD$^+$17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[TET12a] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[TET12b] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[TWS18a] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[TWS18b] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[TWT$^+$16] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2154–2162, 2016.

[Vil08] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[XHW$^+$19] Huang Xiao, Michael Herman, Joerg Wagner, Sebastian Ziesche, Jalal Etesami, and Thai Hong Linh. Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113*, 2019.

[YA17] Gu Ye and Ron Alterovitz. Guided motion planning. In *Robotics research*, pages 291–307. Springer, 2017.

[YSSF19] Tianhe Yu, Gleb Shevchuk, Dorsa Sadigh, and Chelsea Finn. Unsupervised visuomotor control through distributional planning networks. *CoRR*, abs/1902.05542, 2019.

[ZLS$^+$18] Amy Zhang, Adam Lerer, Sainbayar Sukhbaatar, Rob Fergus, and Arthur Szlam. Composable planning with attributes. *arXiv preprint arXiv:1803.00512*, 2018.

[ZMBD08] Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. 2008.